Sensur av hovedoppgaver

Universitetet i Sørøst-Norge
Fakultet for teknologi og maritime fag

Prosjektnummer: **2020-10**
For studieåret: **2018/2020**
Emnekode: **SFHO3201-1 19H Bacheloroppgave**

Prosjektnavn
TRYE App
Utført i samarbeid med: TRYE AS

Ekstern veileder: Hans Kristian Nilsen

Sammendrag: En fullstendig løsning som består av både software og hardware for automatisk utleie av elektriske stisykler

Stikkord:
- Applikasjon
- Hardware
- Interfacing

Tilgjengelig: JA

Prosjekt deltagere og karakter:

Navn	Karakter
Andreas Røed Kjønnerud	
Tobias Hylleseth	
Joachim nordholmen	
Dawit Abamachu	
Eebbaa Dhugaasaa	

Dato: 15. juni 2020

____
José M. M. Ferreira
Intern Veileder

____
Karoline Moholth
Intern Sensor

____
Emil Moholth
Ekstern Sensor

TRYE APP

University of South-Eastern Norway

TRYE

Group Members	Supervisors	Sensors
Andreas Røed Kjønnerud	External Supervisor	External Sensor
Tobias Hylleseth	Hans Kristian Nilsen	Emil Moholth
Eebbaa Dhugaasaa Bantii	Internal Supervisor	Internal Sensor
T	José M. M. Ferreira	Karoline Moholth Mcclenaghan
Dawit Abamachu		

Abstract

The market for renting out electric scooters has expanded rapidly throughout the world in the last years, with projected annual compound growth rates of an impressive 18.9% [9]. In the last couple of years it has also become a huge business here in Norway, and there is massive competition for the customers. What TRYE is trying to achieve, is to make a similar automated renting solution, but to rent out their electric mountain-bikes. This is an innovative concept that has not been done in Norway, and the system consists of a robust hardware solution that can fit on any electrical bike, with a complimentary automatic renting software.

The TRYE App has therefore the potential to both be profitable and easy to maintain. This report explains how we went on about creating the solution.

Contents

List of figures

Abbreviations

ADC Analog to Digital Conversion.

AS Admin System.

ASR Admin System Requirements.

AWS Amazon Web Server.

CS Control System.

CSR Control System Requirements.

ECC Elliptic Curve Cryptography.

ECDSA Elliptic Curve Digital Signature Algorithm.

eMTB electrical Mountain Bike.

GCP Google Cloud Platform.

GND Ground.

GPS Global Positioning System.

HTTP HyperText Transfer Protocol.

IDE Integrated Development Environment.

IoT Internet of things.

JSON javaScript Object Notation.

LTE Long-Term Evolution.

MA Mobile Application.

MAC Medium Access Control.

MAR Mobile Application Requirements.

MQTT Message Queuing Telemetry Transport.

NMEA National Marine Electronics Association.

OTP One Time Password.

PS Power System.

PSR Power System Requirements.

RX Receiver Data.

SM Shimano Mountain Bike.

SWOT Strengths, Weaknesses, Opportunities, Threats.

TS Tracking System.

TSR Tracking System Requirements.

TX Transmitter Data.

UML Unified Modelling Language.

USB Universal Serial Buss.

USN User Story Number.

UUID Universal Unique Identifier.

VCC Voltage at Common Collector.

WS Web Server.

WSR Web Server Requirements.

Chapter 1

Introduction

The project is divided into six major systems. These are referred to as epics (main systems) for the final product. The systems are listed below:

- Software application for customers which is a mobile application (MA).
- Admin application for the TRYE owners called the admin system (AS).
- Web server which will host our development workspace and database (WS).
- GPS tracking system (TS).
- Power system (PS).
- A control system that consists of a microcontroller (CS).

The task of the MA is to make it easy for the users to communicate with the bike trough the TRYE server. This includes making it possible to book bikes, make payments, and unlock bikes. The task of the AS is to give the company owners at TRYE an overview of the customers and bikes. The task of the CSR is to make it possible to send and receive important data from the app with a mobile IoT network, and both lock and unlock bikes depending on if the bike has been booked. The TSR are achieved with a GPS module and its task is to give location data to the hardware system which is then sent to the software. The PSR are responsible for giving power to the microcontroller from the bike.

Chapter 2

State Of The Art

2.1 Background

We are now seeing all kinds of different small battery-driven transportation devices all over the world, with everything from scooters to hoverboards. To find out why there has been a massive growth in these devices in recent years, we have to go back to 2002 when the Segway was launched. Upon launch, it was extremely anticipated, and it seemed to be the future of transportation.

Figure 2.1: The Segway. (source:wikimedia)

The Segway had a futuristic look, but 3 main concerns made it more of a luxury gadget than a common transportation device:

- It had a price of 4950 dollars which makes it too expensive for most people.
- It was very heavy (12KG), which meant it was not very portable.
- The batteries did not have a great capacity at the time which meant charging had to be done frequently.

Ever since the early 2000s, the capacity of batteries has greatly improved, and the price of the batteries has been substantially reduced. This means that newer devices such as electrical scooters and hoverboards have become a preferred transportation device as they are both affordable and portable. It is especially popular for the younger generation who don't have the possibility of driving a car.

This means that some people might want to temporarily use these transportation devices for traveling through densely populated areas while avoiding queues and being environmentally friendly. This opened a whole new market for renting electrical transport devices, especially electrical scooters.

The two main factors that created the marked for renting out electrical scooters is that it is profitable for the owners and that the range of the electric scooter is up to about 30 miles for the customer.

Statistics prove that there has been a price-drop for batteries used for transportation of an impressive 86%[13] from 2010 to 2016. It is, therefore, safe to say that this is one of the reasons why the electric scooter rental market was established in 2017.

As mentioned above, the electrical scooter rental explosion started in 2017 when Lime started using an automated renting solution for their electrical scooters[5]. After their huge success in the United States of America, similar companies across Europe started taking inspiration from their solution. In 2019, VOI launched its scooters in Oslo as the first electrical scooter rental company in Norway[21]. The reason we went on about researching electrical scooter rental solutions, is because it is the only solution on the market that is similar to what we are trying to achieve.

2.2 Current state

In the current market, there are several different scooter rental companies in the larger cities of Norway, making it a huge market with many competitors fighting for the same customers. We will be testing two of the biggest operators in the Norwegian market which are VOI and CIRC.

The business idea is quite simple. Customers download an app on their phone where they can locate the scooters and activates them by adding a payment solution that supports either a credit card or PayPal. Then they will be automatically charged by how long they ride, and a startup fee. The scooters can be found anywhere in the cities, and you can park them close to wherever you want.

It is clear that the different technical solutions are inspired by one another, and they all contain hardware and a software solution. When it comes to the software solution, the appearance of the apps (User Interface) is all different, meanwhile, the functionality of the apps are very similar. The software solutions contain a login solution, booking solution, and payment solution. The hardware solutions contain an IoT communication system, a tracking system, and an unlocking system.

2.3 Cost and risks

The expenditures for creating an automatic rental solution is mostly accounted for working hours of testing and integrating the core functionalities. As the hardware components mostly consist of a cheap microcontroller, the cost of the hardware adds up to a very reasonable figure. Most of the risks associated with the solution are related to malicious users trying to hack or exploit our system. This brings us to another significant expenditure of creating this solution, security. A lot of work hours are needed to test and verify that none of the features in the system can be exploited. This is to protect sensitive customer data like credit card information, location information, and personal information. A leak of this kind is to be avoided, as it might end up with a lawsuit as well as damaging customer privacy. A known security concern is that malicious customers could find a way to unlock bikes without actually paying. There have also been reports of scooters being stolen, and that the hardware is replaced so someone could use it as their personal scooter[12].

Our cost assumptions were confirmed by looking at an article from a professional electrical scooter rental app developer. They assume a development time of 147 days of development and the total cost of 7056 dollars[10]. This means that the cost of development time and scooters with hardware are the two main expenditures.

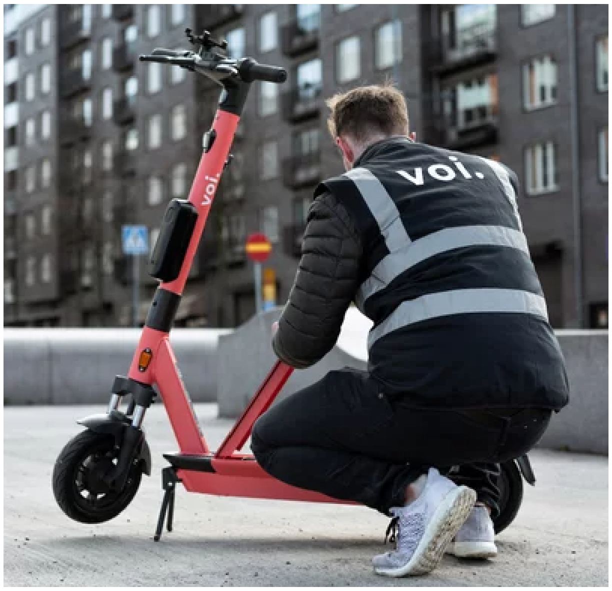


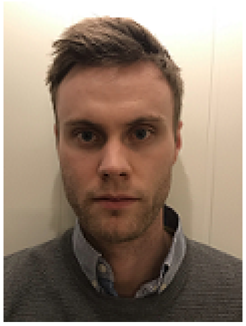
Figure 2.2: VOI scooters. The ”black box” in the front is where the hardware is placed. (source:voiscooters.com)

Chapter 3

Project Overview

3.1 Our team, employer and key persons

3.1.1 Our team of computer scientists



Software Developer
Product Owner
Document Manager
Andreas Kjønnerud

Software Developer
Process Manager
Finance Manager
Tobias Hylleseth

Hardware Developer
Verification Manager
Joachim Nordholmen

Hardware Developer
Requirements
Eebbaa Dhugaasaa

Hardware Developer
Risk Manager
Dawit Abamachu

3.1.2 Our university

The University of Southeast-Norway is the fourth-largest university in Norway and has more than 18000 students and around 1600 employees. The university offers profession-oriented and education directed at the current work sector, research, and conveys knowledge with high international quality. It used to be a university-college when it was established in January 2016 when the college in Buskerud and Telemark merged. On the 4th of May 2018 it received its university status[6].

Our campus in Kongsberg has a wide variety of subjects, but it is most known for its studies within technology and science. Our group all are on the program computer science.

Internal sensor

Karoline Moholth McClenaghan

University lecturer, Faculty for technology, science, and maritime subjects

Institute for science and industrial systems at University of Southeast-Norway, Campus Kongsberg

Our internal sensor is working with grading the student's work during all the bachelor projects in Engineering. She will decide together with our external sensor the individual grades on all the project group members. Our internal sensor naturally takes responsibility that the censoring process is fair and that we are graded correctly. Our internal sensor also takes care of the internal supervisor and students during the process and gives them what support they need.

Internal supervisor

José M. M. Ferreira

Professor, Faculty for technology, science, and maritime subjects

Institute for science and industrial systems at University of Southeast-Norway, Campus Kongsberg

The internal supervisor works with the project group and guides them during the bachelor project. The supervisor has weekly meetings with the group to discuss the progress. The supervisor's main task is to guide the group on how they perform the project but can also help with technical or other aspects students might ask about. The internal supervisor is also present in the panel which will grade the students to give their insights.

3.1.3 Our employer

TRYE AS is a newly founded company that is working on an innovative renting solution for eMTB's. It is owned 50/50 by its founders Hans Kristian Nilsen and Simon Haugan. Currently they are renting out bikes manually from their office in Mjøndalen. They take bookings by call or their booking system on their website. They have high expertise in eMTB's since it is one of their hobbies that they are trying to make a living out of. Renting out eMTB is a new market in Norway, with very few competitors at the moment.

External supervisors

Simon Haugan and Hans Kristian Nilsen

TRYE

External supervisors are representatives from the company who is both a customer and a resource to the group. This means the supervisors should both supervise the group by giving them specified requirements and all the equipment the group needs to complete the project. They should also be available for the group to discuss with and give their input and information that the group needs. They have no responsibility when it comes to the censoring of the project, but they are expected to attend presentations and give their input to the evaluation procedure.

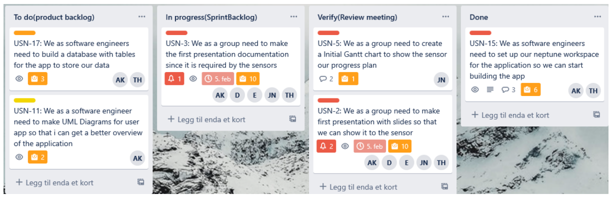
3.2 Process Model

The process model that we made use of in this project is heavily inspired by SCRUM, however, it is not strictly SCRUM. We have on the other hand been very interested in having an agile approach to our system development. In our model, we have focused on having an agile approach to our project development and we have picked the tools necessary to achieve our goals.

Changes to the SCRUM are listed below:

- User Stories visualized with the help of Trello.
- Modifications done to the SCRUM-specific roles

In order to have a successful project development, it is important to pick the right process model. For our project, we decided that the combination of SCRUM and software-oriented Trello and User Stories would be an essential tool for us to perform as best as possible. As this bachelor thesis is software-heavy, our development is extremely agile and changes can be done several times a day depending on the limitations/opportunities we find throughout the development.



To do(product backlog)

USN-17: We as software engineers need to build a database with tables for the app to store our data

USN-11: We as a software engineer need to make UML Diagrams for user app so that i can get a better overview of the application

In progress(SprintBacklog)

USN-3: We as a group need to make the first presentation documentation since it is required by the sensors

Verify(Review meeting)

USN-5: We as a group need to create a Initial Gantt chart to show the sensor our progress plan

USN-2: We as a group need to make first presentation with slides so that we can show it to the sensor

Done

USN-15: We as software engineers need to set up our neptune workspace for the application so we can start building the app

Figure 3.1: Example of our trello board

Progress has been realized with the help of weekly sprints, daily stand-up meetings and a detailed list of end goals to keep track of the progress made. More about this is explained within the Project Workflow part of the report.

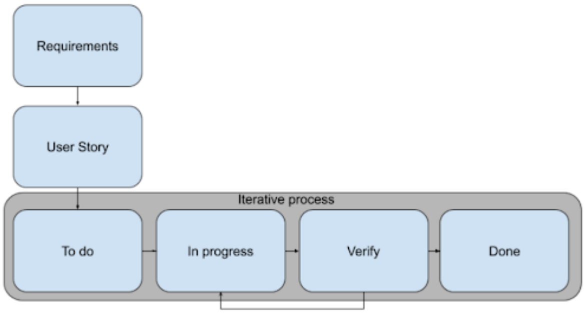


Figure 3.2: Process model

3.2.1 Project Workflow

Following the fundamentals of agile, our project development starts with a proper analysis of the requirements given. From the refined and hierarchically built requirements, we were able to build short and concise User Stories, and the development process took place.

The User Stories are put into Trello, which is an excellent tool for handling User Stories and is initially put in the "To Do" section of the tool. When a group member starts working on a User Story, the User Story is then put in the "In progress" table in Trello. From here on out, the work done is put into the "Verify" table, then, thoroughly verified, and if substantial improvements are needed for the User Story to be finalized, it is moved back to the "In progress" table. On the other hand, if the User Story being verified has fulfilled the desired requirements, the User Story is finished and is moved to the "Done" table.

Although this seems like a long and overly complicated workflow, it has been wonderful to work with for the past months. However, it is important to emphasize that without a proper requirement analysis, and close cooperation with our employee (TRYE), the development might have failed. As Systems can be large and extremely complex, breaking problems into smaller fragments is a must, and is an important part of a successful design process.
You can not expect the employer to be familiar with agile approaches and tools, so visualizing the problem for the employer is a crucial part of figuring out exactly what needs to be produced. Nobody wants to buy a solution that does not fit their problem.

Throughout our development, we have had 14 weekly sprints. A sprint is a period of time that can be specified to fit specific projects. In our case, one sprint lasted one working week. One working week is in our case from Monday to Friday. To supplement the sprints, and to ensure that everything were going as planned, I as the Process Manager implemented the use of daily stand-up meetings both in the morning and afternoon, as well as Sprint planning meetings and Sprint conclusion meetings.

The stand-up meetings in the morning were a maximum of ten minutes, and started with the questions, "What am I going to work on today", and "Why did I choose to work on this specifically". These were done at exactly 9:00 AM every morning and after every team member had explained their thoughts and plan, we started working. One of the reasons why the stand-up meetings were such a success was that it allowed us to work on different parts of the system, and still know how the other group members were doing. This was especially important when everyone had to do their work from home. As a group we worked hard to start the day at the same time, and also end it together. This enabled us to have a short conclusion of our working day, and to report to the other group members how we did throughout the day.

The principles were essentially the same as the stand-up meetings in the morning, however, the questions were flipped to ”What did I end up working with” and ”Am I on track to finish my work within my original estimation”.

In our project, a sprint started on Mondays, and on Monday morning we had longer morning meetings where we planned the workload throughout the week. This included making User Stories if this was not done before, estimating the workload needed for each, and making sure we had enough work to do throughout the entire week. On Friday evenings it was time to conclude how we did in the sprint.

3.2.2 User Stories

A User Story is fundamentally just an informal description of one or more features of a system. As our system contains software, hardware, and cooperation between the two, our User Story differentiates from the common understanding of User Stories. When documenting our work done in the User Stories, we started off with giving it a status. ”To do”, ”In Progress” and ”Done” were the main status phrases to visualize progress. However, we did also make use of more technical terms such as ”Waiting for”, meaning a bottleneck prevented the User Story from being finished and ”Verify” to express that the User Story required a verification.

3.2.3 User Stories workload explained

When figuring out how we were to measure progress to our development, we decided on making use of a feature in Trello where you can give User Stories a number that estimates the work needed to get done with the User Story. In our first couple of Sprints (A Sprint is in our Project one week of work, 5 days) we tried some different scoring systems to estimate the work needed, meanwhile still being easy to use and easy to understand. Our first solution was as follows: 1 Point meant that the task was easy and would be solved within a short amount of time. 2 Points meant that the task was of medium difficulty and would require more time than the 1 Pointer, but still be a fairly easy task that would not require a lot of time. 3 Points were the maximum of our first scoring system, making User Stories with 3 Points the hardest/most technical tasks.

At the start, it seemed like the 1-2-3-Pointers worked excellent and properly described and made track of our progress. In reality, it performed poorly. The reason for the failure of the first point-system was found out when we on Fridays had our weekly review meeting and found out that we all had different expectations on how time-consuming a 2-Pointer would be, etc. We had defined the time consumption as ”Short amount”, ”Medium amount” and ”Large amount” of time, and we had all different ideas of what the different terms meant. This caused us to steer away from vague definitions to the point system we used throughout the rest of our project.

The new Point system is as following: One day's work equals one point. If a task is thought to require approximately three days of work, three points are allocated to the specific User Story. This made analyzing every sprint a lot easier, as detailed statistics of "How many points per week did we score as a team", "How many points did I as a software engineer get last week". Again, the idea of an approximation is that it is thought to be, not exactly how it will turn out. As the project went along, we learned to approximate more accurately, meaning that cross User Story cooperation became an option, instead of constantly meeting bottlenecks due to unrealistic time approximation.

3.2.4 Epics

An epic is a top-level system where most technical user stories are derived from. The epics were put into its own Trello board to keep an overview of the progress of our main systems for the project. When the project is finished the status of the epics will be reported in epic reports.

3.2.5 Process tools

- Trello

Trello is an extremely powerful tool when it comes to visualizing the progress of User Stories. You can create as many User Stories as you would want, and assign group members to each specific User Story. There is also support for extra features known as "Power-ups" in Trello, and a Power-up we used throughout the entire project was giving the User Story individual points to define the expected workload. Each individual User Story was also given a colored flag to express the urgency of the User Story.

- Zoom

As everyone had to start working from home due to the COVID-19 outbreak, we needed to make use of new tools to be able to talk to anyone we needed to talk with.

- Google Drive

Google Drive is where we have stored our entire stack of documentation, making it possible for us to collaborate more easily than sending files back and forth.

- Discord

Discord made it possible for us to continue with our stand-up meetings and Sprint plannings/conclusions. Within Discord we created three different voice-channels. One for the software development, one for the hardware development with its complementary software, and one for the stand-up meetings[11].

3.2.6 Roles and Responsibilities

At the start of the project we quickly assigned fixed individual roles to each group member. In addition, rotating roles were also assigned. The combination of fixed and rotating roles turned out to be a great addition to avoid a static work environment.

Product Owner

The Product Owner is the sole person responsible for managing the Product Backlog, and maximizing the value of the product resulting from the work of the Development Team.[22] Although this is the formal definition of what a Product Owner does, it fit quite nicely with how we made use of the Product Owner in reality. The Product Owner did set up Trello to ensure that the Product Backlog (In our case User Stories) were visible, but as the Product Owner also worked as a Software Developer every group member had to contribute to the visibility of the backlog. Moreover, as it is not uncommon to see key stakeholders as Product Owners, the Product Owner did his best to ensure the value for our employee (TRYE) was maximized.

Process Manager

The work of the Process Manager is to ensure that the way progress is made and that the way we cooperate is done well. Although the Process Model itself is decided by the group as a whole, it is the responsibility of implementing it. Moreover, as there are no leaders within a stand-up meeting, it is their responsibility to plan it. The same goes for Sprint reviews, and even though SCRUM is a process model that focuses on collaboration instead of commanding, it is the Process Managers responsibility to lay the foundation of a sufficient development environment.

Verification Manager

When work from the product backlog is done, it is the responsibility of the Verification Manager to ensure everyone in the group has a clear idea of how to Verify their work sufficiently. Failure of clarity and prepared documents can end in lack of documentation, and most importantly User Stories that do not fulfill their requirements. As the Verification Manager worked as a Hardware Developer as well, the important work done in the project was to develop detailed templates and routines for documenting, verifying, and validating progress.

Requirement Manager

The Requirement Manager is responsible for creating easy-to-understand requirements derived from the original requirement list we were given. In our project, the requirement manager decided to have hierarchically distributed requirements, while still keeping it crystal clear what requirement would be solved by fulfilling a sub-requirement. In the end we had a split requirement-tree-structure that successfully broke down large and complex requirements into requirements that were easy-to-understand.

Risk Manager

The primary responsibility of the risk manager is to understand the scope of the project and the environment in which we manage it. Then the risk manager can facilitate and coordinate the risk discussion in the group and motivate the group to identify risks in a different parts of the project. After knowing the identified risks, the risk manager takes these risks to the team of the project so that the team can assess, evaluate, and prioritize the risk. When the critical and less critical risks are known and prioritized, the risk manager is responsible for analyzing the risk for the project. When the risk analyzing process has been done, the risk manager can focus on the critical risks that can affect the project positively and negatively. The risk manager has a responsibility to reduce the probability of the harmful risks that may impede the objectives of the project and maximize the opportunity of positive risks that increase the achievement of the project. It is also the responsibility of the risk manager to design the risk responses. It means the risk manager can make the risk management strategies that could help us to avoid and mitigate the risk or if the risk is beyond to control, the manager's decision to accept it.

Finance Manager

The Finance Manager is responsible for keeping track of expenses and making the needed orders of components. In our project there were close to a minimum of needed components to get a working system, which made the role less time consuming than other roles. This meant that the Finance Manager could manage other roles at the same time.

Documentation manager

One of the responsibilities connected to being the documentation manager is to make sure all documents are following the rules of documentation according to references and plagiarizing, and oversee that all members are filling out the documents in the correct way and adding them to the final report as appendixes. He also has the responsibility to make the shell in Overleaf with headers, references, and diagrams. The documentation manager will also look through the final report and find bad grammar and language.

Rotating Roles

In our project we had two Rotating Roles in addition to the fixed roles assigned at the start of the project. These are:

1. Secretary
2. Chair Person

As we were five members in our project team, the Secretary and Chair Person roles were assigned to each member every 5th week. The positives of Rotating Roles is that we avoid a static work environment, and ensure that the workload of each team member totals up to approximately the same.

Secretary

The Secretary is a rotating role that is responsible for writing meeting reports after meetings and documenting communication between the team and external teams/collaborators. Examples of external teams are our Employer (TRYE) and our Internal Supervisor.

Chair Person

The Chair Person is a rotating role that is responsible for writing meeting agendas ahead of meetings, and mail them to the person/team we are meeting with. In our project, meeting agendas were written to our Employer (TRYE) and the Internal Supervisor.

3.3 Project Planning

A good bachelor project requires detailed project planning. It will give our team an overview of the important events and critical stages of the project, and guides the team to meet the expectations of the bachelor project. This chapter explains the project plan, which includes important events and schedules.

3.3.1 Visual representation

A visual representation in the form of a timeline diagram can help the team visualize the entire project. We chose to use Gantt diagrams for time scheduling representation. In the Gantt chart we have critical events such as presentations for the sensors and finish of different versions of the software application and hardware. The overall timeline Gantt chart is shown below but is also in the appendix.

TRYE APP Gantt Chart

ID	Task name	Start date	Due date	Duration (days)	Timeline
1	Project	01-01-2020	06-12-2020	163	🟩
2	Before 1. Presentation			0	
3	- Project planning	01-13-2020	02-06-2020	24	🟩
4	- Requirement analysis	01-13-2020	02-06-2020	24	🟩
5	- Verification Planning	01-29-2020	02-07-2020	9	🟩
6	- Preperation for first presentation	02-07-2020	02-12-2020	5	🟩
7	FIRST PRESENTATION	02-12-2020	02-13-2020	1	🟥
8	Before 2. Presentation			0	
9	- Concept planning	02-13-2020	02-21-2020	8	🟩
10	- App development (Initial Prototype)	02-13-2020	03-04-2020	20	🟩
11	- Hardware development (Initial Prototype)	02-13-2020	03-10-2020	26	🟩
12	- Prototype validation	03-04-2020	03-18-2020	14	🟩
13	- Preparing for second presentation	03-14-2020	03-23-2020	9	🟩
14	SECOND PRESENTATION	03-24-2020	03-25-2020	1	🟥
15	Before 3. Presentation			0	
16	- Configuring the server	03-19-2020	04-03-2020	15	🟩
17	- Finishing app	04-01-2020	04-17-2020	16	🟩
18	- Finishing hardware	04-13-2020	04-29-2020	16	🟩
19	- Verification and Validation	04-28-2020	05-08-2020	10	🟩
20	- Finish Documentation	05-04-2020	05-27-2020	23	🟩
21	- Make last presenation	05-20-2020	05-29-2020	9	🟩
22	- Prepare for last presentation	05-29-2020	06-05-2020	7	🟩
23	THIRD PRESENTATION	06-02-2020	06-12-2020	10	🟥

3.3.2 Presenting and delivering our work

In the time-span of our project, our group needed to attend three obligatory presentations for the sensors, our supervisors, and others who might be interested in it. Before our final presentation, an Expo was supposed to hold at the university but is now moved to a digital platform, where we make posters digitally to show our project. We had a first presentation on the 12th of February where we explained our task given by the company, what kind of company they are, how we go about doing the task, and we showed our process model.

In the second presentation on the 24th of March we had to have more of a technical focus. We gave an insight into our work and the hardware and software system, how they should work together. Basically a summary of our progress so far. Then we talked about our road ahead and what we are going to do next. In these two first presentations, we were given 20 minutes to present. That means we talked for 4 minutes each. When we were done with the presentation there were 10 minutes which was used for questions from the sensor and audience

The third presentation will be on the 11th of June and is the last opportunity for the group to show how good we are and how good our technical solution is. The time set aside for this presentation is an hour. 20 minutes where we try to sell our product, 20 minutes where we explain our technical solution and 20 minutes in the end for sensors to ask questions. On May 25th, our final report has to be delivered. Our final documentation is this report with appendixes, which will account for fifty percent of our grade.

3.3.3 Work time

Our group is using an agile process model where one week is equivalent to one sprint. Our working hours where 09:00 to 15:00 when we still were at the campus, but after the corona lock-down we have had a more open schedule where people can work whenever they want during the day as long as they finish their work. We have a total of 14 sprints. We were working 3 days a week before the exams and the full 5 days after exams. Which means we worked at least 18 hours a week before the exam. After the exam we worked at least 30 hours per week. We also used about 80 hours in January where we worked full weeks for planning and finding an employer.

That means we have planned to work 250 hours before the exam and about 254 hours after the exam. That will be a total of about 504 hours for each group member. That will add up to around 3000 working hours for our entire group.

3.4 Requirements

In this section we discuss how the group understands the two categories of the requirements called project requirements and product requirements. Besides this we will go through how product requirements used to understand the system of interest and derive the stakeholders. In the end, an explanation will be presented in how we made requirement analysis as a group.

3.4.1 Project Requirements

The source of project requirements is USN. The requirements are presented by the university for students of the third year and those registered for taking the bachelor project. The basic project requirements are presented as follows.

Group: The bachelor project should be performed by groups of 4-6 students. Each student should finish 120 study points before the bachelor project starts. The group has responsibility for finding a relevant project from a company and submitting it to the university for approval. An approved project will have a signed contract between the group and the company.

Grading: For grading the project group should have three presentations at different stages in the process according to the university schedule and should submit written documentation. The group members will be graded by internal and external sensors in the scale of A to F at the end of the project.

Company: The company that fulfills the criteria for giving a project will sign a contract with the group and the university based on the terms set by the university.

For this project we are a group of five students, we all are from the department of computer engineering. The company for this project is TRYE. A contract is signed among the three parties the university, the group, and the company.

3.4.2 Product Requirements

As a group, we got a list of product requirements from the company. The company prepared a document that describes the product and its functionalities. Most of the product functionalities are set by the company. Understanding the product requirements from the company is the most important part of the project which helps the group to understand the system we are going to build and consider all possible stakeholders related to the system.

3.4.3 Stakeholders

Stakeholders are any group of individuals or other existing systems that may interact directly or indirectly with the system we are going to build. Considering stakeholders and their concerns during system development will help us to capture more requirements besides the product requirements given by the company.

Stakeholders may have different concerns or concerns that may overlap with one another. As a group, we sorted all possible stakeholders and their concerns. From these concerns some requirements are derived and discussions were made between the group and the company to expand the scope of the product requirement given by the company. Some of the stakeholders for our product are listed down below.

- **Customers or end-users:** are stakeholders that have a direct relation with our system. They may have different concerns related to reliability, performance, security, and how easy the system is to use.
- **Maintenance crew:** are stakeholders that are responsible for maintaining the system in its life cycle. Their main concerns related to the system are how difficult to maintain the system and system accuracy in localizing the bikes remotely.
- **Banks:** are related to our system through their customers. It happens when the end-user makes payment through our system. Their main concerns are how the system processes users' private data such as bank number in a confidential way during the payment process.
- **Network Provider:** This stakeholder is responsible for any communication related to message exchange and internet communication that may happen between our system and other systems.
- **Norwegian Law/Regulations:** are responsible for making sure that the laws and regulations of the country are kept. As a system the system we build should meet the standards of the country and users should accept the rules and regulations given when it comes to using the system.
- **System Developer (our group):** since we are responsible for developing the system, our main concerns are to build a system that satisfies all the stakeholder's needs and a system that is easy to upgrade.
- **Electrical Mountain Bikes:** These are systems that will be interfaced with our system after it is finished. Their main concerns are the system should be easy to interface with their existing sub systems and it should not be big in size.
- **TRYE:** is a company that comes with the idea of building the system. Their main concerns are if the system we build meets their requirements and if the system built is in the standards of currently existing systems in the market.

Considering stakeholder's concerns besides given requirements during the system development life cycle will help to build a system that meets the standards and the stakeholders' needs. As an example, if we take Norwegian law and regulation, It is one of the stakeholders and their main concern is to make sure that laws and regulations are obeyed. End-user which use the product should be based on the rule and regulations, so as a group we need to make sure that the product we design has a mechanism to inform the rules and regulation to end-users and make them accept the terms of the agreement before using the product. This feature is added in the user app, where users should accept the terms before renting or booking a bike. This makes our system to meet the legal issue that may be raised by this organization.

3.4.4 Requirement Analysis

The main goal of this analysis is to derive an organized list of requirements from product and stakeholders requirements. Since requirements have to be measurable, traceable, testable, and verifiable, The final derived requirements should have a category, unique ID, verification ID, and verification method. Requirements shall be used to derive user histories. All validation, tests, and derived user histories should be related to their source of requirements. From the given requirements we sorted out functional requirements. These are a list of functionalities our product should have in order to satisfy TRYE as a customer. Identifying functional requirements at an early stage helps the group for understanding the systems and sub-systems needed to give the functionalities mentioned in the requirement document. The following functionalities are sorted from the requirement document.

- **Function 1:** The system should enable bike users to rent a bike using a mobile app.
- **Function 2:** The system should enable bike users to make payments using a mobile app.
- **Function 3:** The system should send a digital receipt to the user's email address.
- **Function 4:** The system should enable bike users to book a bike using a mobile app.
- **Function 5:** The system should enable bike users to make failure reports using the mobile app.
- **Function 6:** The system should enable paid users to unlock a bike using a mobile app.
- **Function 7:** The system should provide users with a tour suggestion map to their mobile app.

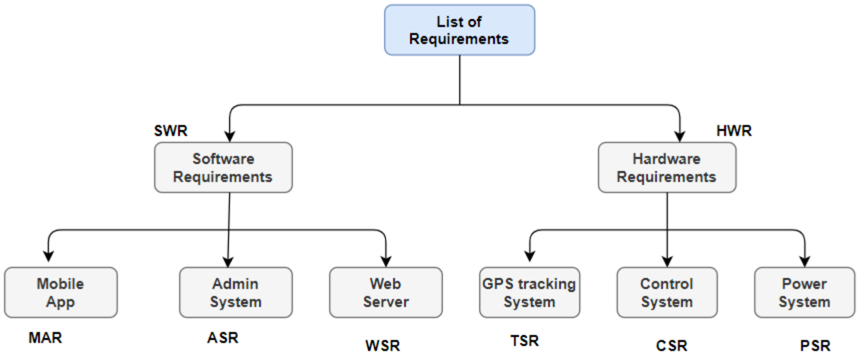


Figure 3.3: Category of Requirements with their corresponding subsystems.

- **Function 8:** The system should help to localize the bike with the help of a GPS tracking system.

The System should be able to interact with other systems such as GPS satellites, Web Servers, Banks and electric Bikes to provide the functionalities listed above. In addition, to accomplish one of the functions, the System should use one or more sub-systems.

The system should have both software and hardware implementations. Categorizing the requirements based on these implementations, and identifying the systems and subsystems needed to accomplish each functionality helped the group to categorize the requirements accordingly. We categorized the requirements as Software requirements(SWR) and Hardware requirements (HWR) as shown in Figure 3.2

These requirements further divided based on the subsystems under each category and a unique name and number are given for identifying the requirements in each subsystem. In Appendix A.7.1 organized list of requirements with a unique ID, description, verification method, and Verification ID is attached while the Product requirement Document from TRYE is attached in Appendix A.13.1

Figure 3.3 summarizes How the group used requirements in different phases of the system development life cycle. It starts where requirements gave us the general system overview at the start phase. In later phases requirements were used by the group to identify all possible system functionalities and further to drive the subsystems with their components.

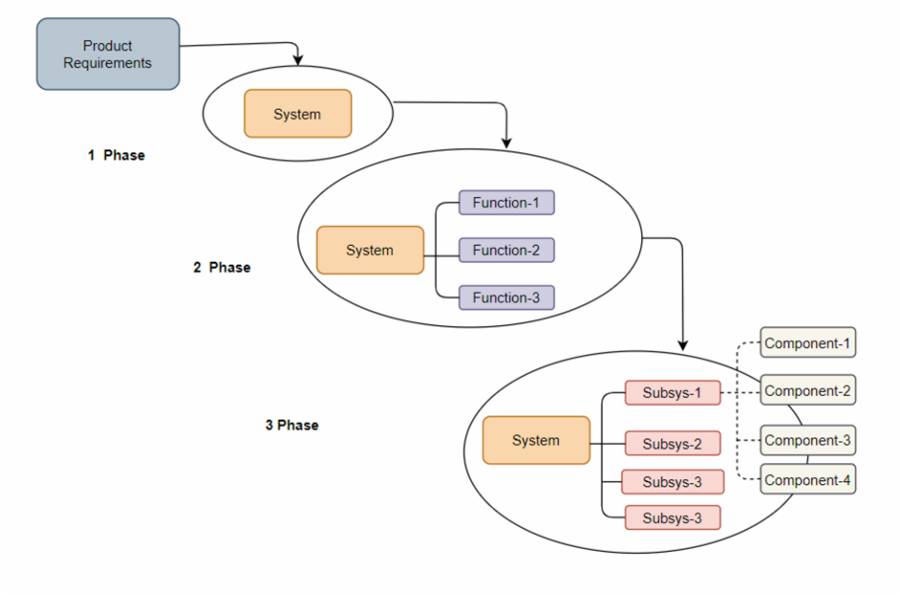


Figure 3.4: Usage of requirements in different phases.

3.5 Verification and Validation

In this section, we discuss the importance of validation and verification, how it should be done, and how the group managed to document the verification prosses to ensure the requirements were satisfied

3.5.1 Validation

Validation is used to ensure that the product made is what the main stakeholder TRYE was expecting. This can be done by talking and demonstrating the product being made and ask for feedback. Our system was divided into two parts software and hardware. The mobile app can be validated by sending TRYE a link to the app, where they can see the progress. This way, TRYE can see the development of the app, and say if this is what they want. To validate hardware, we have to explain what we have done in a high-level language and ask for feedback, as TRYE is not expected to understand the technical diagrams that have been made.

3.5.2 Verification

Verification is used to prove a sub-system is working as intended specified in the requirements. The sub-systems must be verified throughout the project and not only in the end, this is to find faults as early as possible. If a faulty system is carried on being used, it can cause problems to other system build on top of it. Correcting the faulty system might imply having to change large portions of the system as they are dependent on a faulty system, therefore significantly increasing the development time. As soon as a user story is done it should be verified, to do so, we have used four different verification methods: test, analysis, demonstration, and inspection.

1. Test: Here a series of predefined input data is given to the function/system and the output is compared with the expected output, e.g:
   - Software: To test a login form, try logging in with known existing users, known non-existing users, correct password, wrong password, and special characters.
   - Hardware: To test if a battery reader reads the correct battery level. It is possible to measure how long the bikes go in km until it's at 50% charge, after that the bike should go twice as long until reaching 0%. This assumes we want to display the battery as a linear function of how long the bike can travel.
2. Analysis: is verification based on models, simulations, or calculations, this can be used when testing is not possible, e.g:
   - Software: to check if the server can handle the expected number of users, it can instead be calculated, this is done by adding a few users and check how much resources one user uses, then multiply that by the expected number of users.
   - Hardware: to test if the correct battery level is displayed, the battery can be measured, and with a known max and min battery voltage the battery level in percentage can be calculated.
3. Demonstration: use the product as the end-user should be using it, and see if it works as expected, e.g:
   - Software: pen the app on different phones, and check to see if it's acting as it should be
   - Hardware: when the bike receives an unlock bike signal, the motor should turn on and be ready to use
4. Inspection: is a non-destructive way to verify using one or more of the five senses, this can be things like comparing with diagrams, physical manipulation, and so on e.g:
   - Software: check to see if the bikes are where they should be on the map, given a set of coordinates
   - Hardware: given the wheel is spinning a certain speed, check to see if the speed on the display is corresponding with our system

3.5.3 Verification Documentation

To document the verification, there are two files to consider, the first one is the spreadsheet which contains every verification done as seen in appendix A.7.1, and the second one is the document belonging to each of the verification processes as seen in appendix A.7. The spreadsheet is used to get an overview of all the verification's, to help see a quick summary, ID, status, who worked on it, when it was done, and a link to more in-depth documentation. The document that follows each of the verification processes, should contain why this need to be verified or what the consequences of this sub-system not working are, the process used to get the results used to verify, the results and a conclusion where we tell if this is as expected or not.

3.6 Project Risk Management

Risk management is the scale that we can use to cover all areas of our project's difficulties. It meant that the challenges due to technical risk, budget risk, schedule risk, and quality risk. If these risks might occur, they could no doubt limit the success of the project unless they managed. Therefore, after identifying all these potential risks in every part of the project, we need to analyze them, mitigate them, monitor, and follow them to stay the project in control. Since we follow the scrum model's approach for this project, we can benefit the model that contribute to minimizing the threat itself. In the process of risk management, we don't have a standard risk tool to follow. But, the project team can define it on its own and take the only insight to refer to different standards. To achieve the goal of the project, we need to pursue the following risk management tools. These are risk identification, probability-impact matrix, and risk register.

3.6.1 Project risk identification

Before we start to identify risk, we need to understand and know the risk identification techniques that provide the project team on how to detecting risk. A risk is an uncertain event or condition that, if it occurs, affects at least one project objective [19]. Therefore, risk identification techniques are an essential tool used to start with identifying risk. Risk identification is the risk management tool that helps us to collect all information during the identification process and to use them as a base for further risk analysis.

Brainstorming is one of the techniques that we have used in our project to identify the risk. In this case, we sit down and discuss all parts of the project where a chance can probably occur. The group members come with any uncertainty in mind, which relates to the role they are working on and take it for discussion. Then we document these risks for further analysis. It is often our choice or decision on some issues that arise a new threat that we didn't' t see in the previous steps. We also need to register all these new risks. The other technique is the SWOT analysis technique and used to analyze the group member so that we understand and visualize everything early in the project. We can find the SWOT analysis table and their description in Appendix A.8.4.

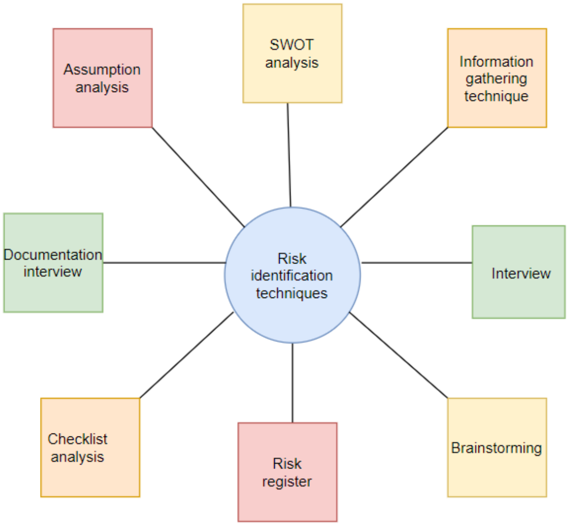
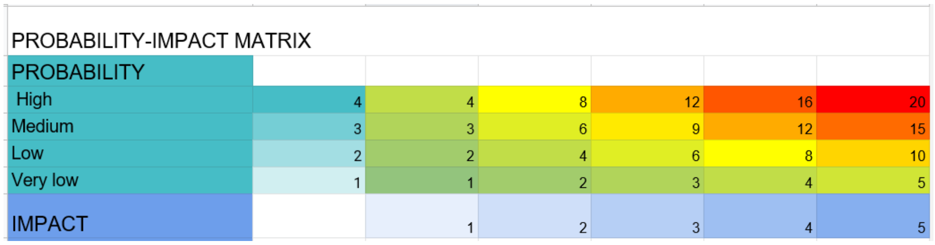


Figure 3.5: Some of risk identification techniques.

3.6.2 Probability-impact matrix

The project uses a probability-impact matrix and a spreadsheet to calculate and determine the relationship between the probability of the risk occurring and the impact of risk identified. The value given for the probability when multiplied by the value given for impact provides the risk with the product, we call it risk product. Risks with the high-risk product have given high priority, while those with a low-risk product have included on a follow list for the future. In this case, we consider the following as the project's weights when calculating the value for risk products. These are project scope, quality, cost, velocity, and credibility.[2]



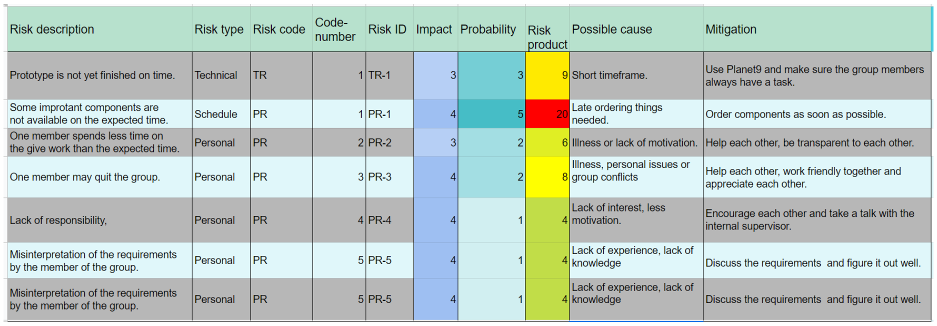
PROBABILITY-IMPACT MATRIX						
PROBABILITY						
High	4	4	8	12	16	20
Medium	3	3	6	9	12	15
Low	2	2	4	6	8	10
Very low	1	1	2	3	4	5
IMPACT		1	2	3	4	5

Figure 3.6: Five by four probability-impact matrix.

The combination of probability and impact conduct the rating of the risks as very low, low, medium, and high. The rating value for probability is given from one to four, whereas the rating value for impact is from one to five. The reason the rating values for impacts has given from one to five is that we consider the impacts as an essential factor that affects the project's objectives. Different colors represent the value of risk products. As it goes from a light green color to a red color, we know that the risk value goes from low to high. It means that the light green color represents the very-low risk; the yellow color represents the medium risk, whereas the red color represents a high risk.

3.6.3 Project risk register

The risk register is a documented result of qualitative and quantitative risk analysis and risk response planning [20]. We used it for documenting, managing, and tracking all risks that have already identified in the project. It also allowed us to describe the risk consistently and follow these harmful risks readily in the project. Moreover, we could use the risk register as a record for future use. We could see different types of risks in the register. We divided them into project risk and technical risk for the ease of clarity of these risks in the risk register.



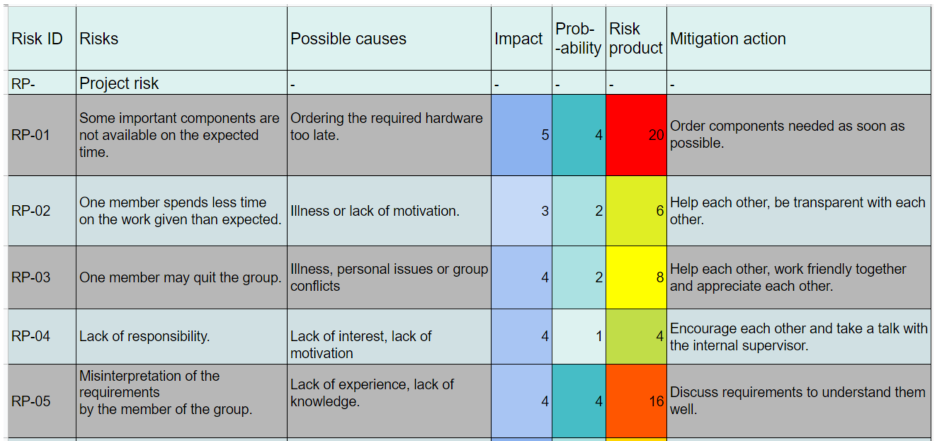
Risk description	Risk type	Risk code	Code-number	Risk ID	Impact	Probability	Risk product	Possible cause	Mitigation
Prototype is not yet finished on time.	Technical	TR	1	TR-1	3	3	9	Short timeframe.	Use Planet9 and make sure the group members always have a task.
Some improtant components are not available on the expected time.	Schedule	PR	1	PR-1	4	5	20	Late ordering things needed.	Order components as soon as possible.
One member spends less time on the give work than the expected time.	Personal	PR	2	PR-2	3	2	6	Illness or lack of motivation.	Help each other, be transparent to each other.
One member may quit the group.	Personal	PR	3	PR-3	4	2	8	Illness, personal issues or group conflicts	Help each other, work friendly together and appreciate each other.
Lack of responsibility,	Personal	PR	4	PR-4	4	1	4	Lack of interest, less motivation.	Encourage each other and take a talk with the internal supervisor.
Misinterpretation of the requirements by the member of the group.	Personal	PR	5	PR-5	4	1	4	Lack of experience, lack of knowledge	Discuss the requirements and figure it out well.
Misinterpretation of the requirements by the member of the group.	Personal	PR	5	PR-5	4	1	4	Lack of experience, lack of knowledge	Discuss the requirements and figure it out well.

Figure 3.7: Shows a table of some chosen risk register.

In the table, we can see the risk description, risk type, risk code (TR-Technical risks, RP-Project risks), code number for risk, risk ID (TR & RP) followed by a number, impact, probability, possible causes, mitigation action, date of risk update, and the owner of the risk. We can find the entire project risk register table with all information in appendix A.8.1.

3.6.4 Project risks

The project risks are hazardous risks that affect the accomplishment of the project when they occur. Therefore, we need to monitor and control them continually. Threats included in project risk are project scope risk, budget risk, schedule risk, personal risk, and quality risk. We need to interpret these risks and decide how to stop or reduce their effects upon the project.



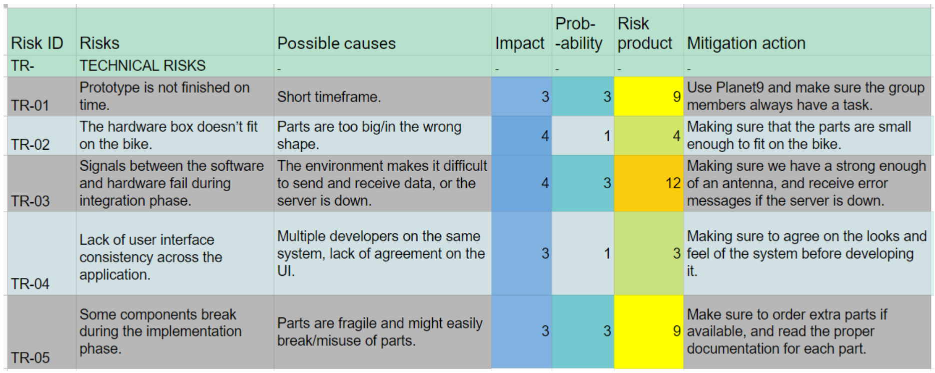
Risk ID	Risks	Possible causes	Impact	Prob- -ability	Risk product	Mitigation action
RP-	Project risk	-	-	-	-	-
RP-01	Some important components are not available on the expected time.	Ordering the required hardware too late.	5	4	20	Order components needed as soon as possible.
RP-02	One member spends less time on the work given than expected.	Illness or lack of motivation.	3	2	6	Help each other, be transparent with each other.
RP-03	One member may quit the group.	Illness, personal issues or group conflicts	4	2	8	Help each other, work friendly together and appreciate each other.
RP-04	Lack of responsibility.	Lack of interest, lack of motivation	4	1	4	Encourage each other and take a talk with the internal supervisor.
RP-05	Misinterpretation of the requirements by the member of the group.	Lack of experience, lack of knowledge.	4	4	16	Discuss requirements to understand them well.

Figure 3.8: Shows a table of some chosen project risks.

In the table, we can see risk ID starting with RP followed by a number, descriptions of risk, causes of risk, impact, probability, risk product, and the mitigation actions. All risks are in chronological order when they get identified. We can find the entire project risk table with all information in appendix A.8.2.

3.6.5 Technical risks

Technical risks mainly due to the incidents of technical difficulties, inappropriate representation of requirements, the inadequacy of design, and insufficient calculations. In general, the occurrence of technical risks in the project could increase development time, decrease product quality, and lower the product's performance [4].



Risk ID	Risks	Possible causes	Impact	Prob- -ability	Risk product	Mitigation action
TR-	TECHNICAL RISKS	-	-	-	-	-
TR-01	Prototype is not finished on time.	Short timeframe.	3	3	9	Use Planet9 and make sure the group members always have a task.
TR-02	The hardware box doesn't fit on the bike.	Parts are too big/in the wrong shape.	4	1	4	Making sure that the parts are small enough to fit on the bike.
TR-03	Signals between the software and hardware fail during integration phase.	The environment makes it difficult to send and receive data, or the server is down.	4	3	12	Making sure we have a strong enough of an antenna, and receive error messages if the server is down.
TR-04	Lack of user interface consistency across the application.	Multiple developers on the same system, lack of agreement on the UI.	3	1	3	Making sure to agree on the looks and feel of the system before developing it.
TR-05	Some components break during the implementation phase.	Parts are fragile and might easily break/misuse of parts.	3	3	9	Make sure to order extra parts if available, and read the proper documentation for each part.

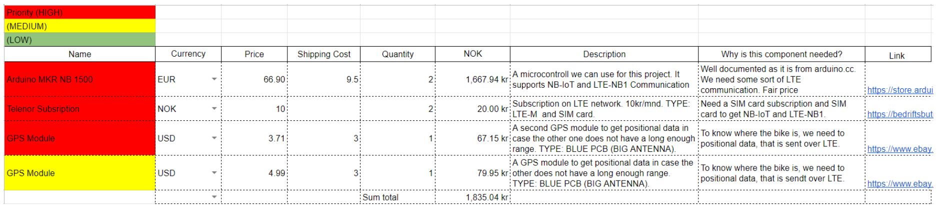
Figure 3.9: Shows a table of some chosen technical risks.

In the table, we can see risk ID starting with RT followed by a number, descriptions of risk, causes of risk, impact, probability, risk product, and the mitigation actions. All risks are in chronological order when they get identified. We can find the entire technical risk table with all information in appendix A.8.3.

3.7 Finance Management

Having a deep understanding of the finances in the system is important when it comes to deployment. In our case, we have only considered the parts needed for our development as true costs. In a regular work environment, it is safe to say that the work hours would of have been the greatest expense.

3.7.1 The cost of development



Priority (HIGH)
(MEDIUM)
(LOW)

Name	Currency	Price	Shipping Cost	Quantity	NOK	Description	Why is this component needed?	Link
Arduino MKR NB 1500	EUR	66.90	9.5	2	1,667.94 kr	A microcontroll we can use for this project. It supports NB-IoT and LTE-NB1 Communication	Well documented as it is from arduino.cc. We need some sort of LTE communication. Fair price	https://store.ardui
Telenor Subsription	NOK	10		2	20.00 kr	Subscription on LTE network. 10kr/mnd. TYPE: LTE-M and SIM card.	Need a SIM card subscription and SIM card to get NB-IoT and LTE-NB1.	https://bedriftsbut
GPS Module	USD	3.71	3	1	67.15 kr	A second GPS module to get positional data in case the other one does not have a long enough range. TYPE: BLUE PCB (BIG ANTENNA).	To know where the bike is, we need to positional data, that is sent over LTE.	https://www.ebay.
GPS Module	USD	4.99	3	1	79.95 kr	A GPS module to get positional data in case the other does not have a long enough range. TYPE: BLUE PCB (BIG ANTENNA).	To know where the bike is, we need to positional data, that is sendt over LTE.	https://www.ebay.
				Sum total	1,835.04 kr			

Figure 3.10: Shows a table containing all the parts bought during the project.

As seen in figure 3.10, the total price of the project ended up at 1 835NOK. However, it is important to note that we bought twice the amount of parts needed for development in case some parts didn't work as expected. When it comes to scaling, the cost of hardware per unit is a total of: 934NOK. If we compare the price of the hardware with the costs of the bike of an approximated 40 000NOK, the hardware addition increases the total cost per bike up to: 40 934NOK. This increases the costs by 2,35% per fully equipped bicycle.

Chapter 4

Technical Work Process

4.1 Versions in software

We decided to split the Applications into different versions to show that we are working iteratively. This means that we might remove features from the previous version if we render them not useful or add new features that we feel are missing. It is also a way for us to get feedback from different types of people during our development. The user experience is an important bit of verifying and validating the performance of our solution. Three versions were planned but only 2 were confirmed finished. We used the two versions as prototypes when testing distinct functionalities and validated that the user experience was as intended. We were planning on 3 versions of the MA and 1 version of the AS. The MA versions were planned as Alpha version 1, Alpha version 2, and Beta version 1. For the ASR we planned one alpha version.

These test versions have been helpful in discussions for further development, and have helped the group understand what features are needed in the application to give users the best possible experience.

We planned to make three different versions of the user application. The three different versions would be Alpha1 which is a fast developed shell for the app. Then comes the Alpha 2 which is supposed to be the user application with all features working. After the Alpha 2, we moved on to Beta 1 which was supposed to be released on the Google Play and Apple Store. The Beta1 is more focused on final debugging and implementing security features. This is our initial time frames for the versions is shown below

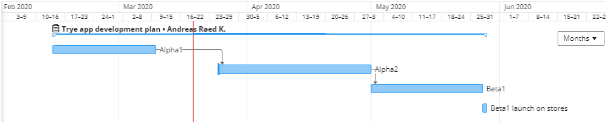


Figure 4.1: Version Gantt chart

4.2 Software concept

In the conceptual face of the software development we started by testing out two different popular scooter rental applications called VOI and CIRC. You can have a look at the test in appendix?. By looking at these apps and discussing what we liked and disliked we started planning the shell and appearance of the app in the first place.

In the draft phase we started with the main menu interface, and all of these interfaces are quite similar in most of the applications we looked at. It is a sidebar menu on the left and a map that shows the location of the rental sites.

But before the customers reach the main menu they have to go through a verification process. The reason for this is that expensive bikes are being rented out and to hold customers responsible we will have to identify them in some way. The easiest way to do this is by SMS verification. The customer enters their phone number and receives a one time password. When their phone number is verified by the customer entering the code, we then can track the booking to that specific phone, which gives TRYE extra safety. The customer also needs to

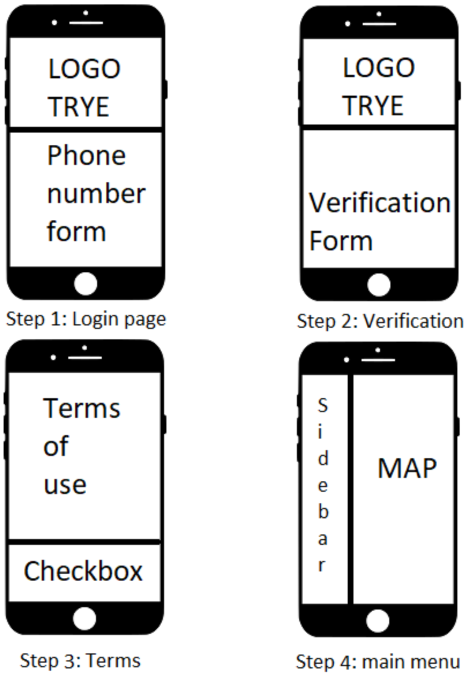


Figure 4.2: Initial visualization of the application

accept the terms set by TRYE. These terms were sent to us by TRYE and are visible in appendix A.13.2. After a user has registered and accepted the terms, they don't have to go through this process the next time they log in.

So in the concept face we made initial thoughts, and made a drawing for the visual UI of the application. And after this drawing we made the app and functionalities from that. We made no UML drawings in the concept face because we wanted to start as soon as possible and learn as much as we could about the development platform and start with a base app that we could use for further planning.

We also looked at the requirements for the three different software systems which are the MAR, ASR, and WSR. We look at which features TRYE considered the most important and prioritized our development from that.

4.2.1 Picking our development environment

In the Software concept, it was already decided that we were going to create an app, the question was just how were we going to do it? A lot of different app developer tools were drafted, and one of the promising development platforms was a software named Planet9, developed by the Norwegian company Neptune Software.

After a brief conversation by email, the entire group was invited for an introduction to the platform in Neptune Software's headquarters in Oslo. The CTO Njål Stabell welcomed us and had a presentation of the strengths and opportunities given by developing in Planet9. As the platform seemed like it had everything we would want to develop a modern application, we were given 10 free licenses for the platform.

The deciding factor that made us pick Planet9 as our development platform was the fact that the platform needed to be hosted on a server, meaning it would be reachable from anywhere in the world. What we didn't know at that moment was that the fact that the platform could be reached from anywhere would be an important factor for us to be able to finish our work on the application with the COVID-19 lock-down, enabling us to continue working as if we were at the office.

4.2.2 How does Planet9 work as a development environment?

Planet9 is an "all-in-one" platform containing both the database layer, API-support (for both external and internal APIs), security layer, and an app development section.

Setting up Planet9 on a server

To use planet9 in a development project you need a host to execute the program. You can run it locally or from a server. It will run in a browser. To be able to cooperate you need license keys.

Choosing a suitable database for our platform

Even though Planet9 comes with a built-in standard-database, it was strongly encouraged to pick another database when developing. The database we chose was PostgreSQL. Then, after deciding to go for PostgreSQL as our database, we had to first host the database somewhere. As we did when setting up the development Platform, this was initially done in AWS, then moved to Google Cloud Platform due to technical difficulties. When the database was up and running, all we had to do was to create a Planet9 database schema in the PostgreSQL console. Then, using the built-in database support, we had successfully connected the database to our development platform, meaning we could start creating the app itself.

The App Developer

One of the strongest features of Planet9 is the built-in App Developer. Planet9 is built upon an open-source framework called OpenUI5, and has extensive support for external API's, meaning it is not limited to "in-house" features. However, OpenUI5 proved to be extremely useful when developing and it was not often that we needed external data objects. OpenUI5 is a framework released by SAP, which is one of the leading development environments for businesses worldwide, meaning there was always a sufficient amount of documentation to help us in the right direction. Another unique feature of the Planet9 App Developer is that it makes use of the modern approach to app development, namely, Low-code. Applications are built hierarchically, with objects within objects. When it came to testing our application, we made use of the built-in "runner", where the app would load up and act like it was on a mobile device.

4.2.3 The SMS Service

One of the goals of our group was to create a modern application made with the help of modern tools. Therefore, it was an obvious choice to go for a registration process needing One-time Passwords delivered to the user's phone number. In the beginning, this seemed like an incredibly difficult task, raising basic questions such as "How do we send SMS's in an App?" and "How much is it going to cost?". However, we knew it was possible, as many large companies had SMS support. The answer to our problem was a service named messageBird. With Planet9's extensive API support, and messageBird being API focused, it was a timely journey filled with a large number of failures before we finally were able to send our first SMS. After we cracked the SMS code, tweaking everything to support our One-Time Password feature was a fairly simple task.

4.3 Web server

4.3.1 Initial server setup

In the initial phase of the project we asked Hans Kristian about buying a database for his company or renting a virtual database. He told us we will have to find a solution that is free during the development phase. So to learn Planet9 we set up the instance on a local server which was done by Tobias.

Planet9 was running fine and we were able to cooperate. The problem was we could only use SQLite when we wanted to use PostgreSQL. And we also had problems with the server not having a 24/7 up-time. Which meant that some days we couldn't develop since the server was down. So we found out we had no other choice but to use a cloud platform to set up our work space.

4.3.2 AWS server setup

To get our Planet9 work space up and running on an external server, we started to check for cloud web services that are free. We found AWS which has both virtual machines to run our programs and you can also set up database instances for free.

For this task Andreas was chosen and he started working on it since he has a lot of experience with working in Linux. The important thing was to get the right version of Planet9. That means the Linux version, and uploading and running it in our virtual instance.

We faced some problems in the start connecting to the instance. What we found out is that setting the firewall security rules is an important aspect of running virtual machines. So for instance since Planet9 runs on port 8080 we only want inbound traffic to our server on port 8080 which anyone can access from any IP address, and the outbound traffic can go anywhere from port 8080 on the servers IP address. Once the security rules were set up we could connect to our instance by typing in the public IP address of the server.

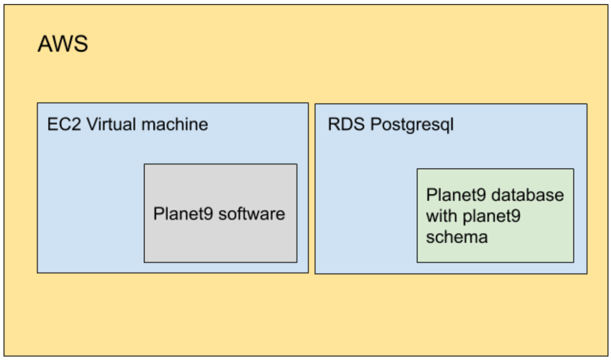


Figure 4.3: AWS server setup visualized

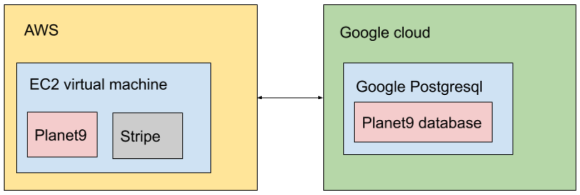


Figure 4.4: Final server setup visualized

4.3.3 Final server setup

During the MA Alpha 2 version development the server setup changed a bit. The reason for this is we had some technical problems with our database so we changed the database to the Google platform. Doing this also made the database more secure since Google will not allow you to have generic security rules. You have to type in the exact IP addresses to the client who has access to the database. This means only Andreas's computer and the web server have access to the database.
A Stripe back end server was also set up using node.js. This server also communicates with Stripe's server. This node program was also running on AWS together with Planet9.

4.4 Alpha 1 MA

Alpha 1 of the User App was worked on from the 14th of February to the 9th of March. Our goal with the first Alpha was to create a working shell for further development. Some of the key features we made a place for in the application were:

1. A Login System
2. A User Database
3. A Main Menu Shell
4. A Basic Map

Further information on each feature is given below.

4.4.1 The OTP login system

The idea of having a OTP Login System was one of the first features we wanted to get to work. Unfortunately, it proved difficult to get working in time for the deadline of our first Alpha. What we did get working was sending SMSs from the API tester built into Planet9. From there we were able to send a ”test” SMS to verify the API as well as fill the SMS API with a Phone number, text, and sender. An example of an SMS we were able to send was ”This is a test SMS”, to phone number x and sender: ”TRYE”.

Being able to have TRYE the sender of the SMS was of great use to us, making our system more credible than having ”Unknown Sender”. To get a visually appealing first Alpha we sent the test SMS with a hard-coded four-digit code that was the ”correct” answer in the Verification tab of the app. However, this limited us to only being able to send an SMS to one phone number at a time, and that the verification code had to be the same every time.

4.4.2 The User database

The initial User Database was purely made up of a ”Phone number” and ”Email-number”, with the phone number as the primary key. The reason we choose the phone number as the primary key was because of the assumption that one person is linked to one phone number, and every phone number is unique. With this solution a person could also get a new phone number, but would have to create a new account.

The User Database has since been worked on a lot, but we were creating an initial OTP login system, we purely wanted a visually appealing first Alpha.

4.4.3 The Main Menu Shell

The Main Menu shell was made up of a typical hamburger-menu with different sub-menus that were made visible when pressing the hamburger-menu. It contained a tab for everything we thought was needed in the app such as ”Profile”, ”Payment”, ”History”, ”FAQ” etc.

The sub-menus were on the other hand not responsible, but more of a shell of what's to come. The design of the menu has however stayed the same throughout the entire development, with the addition of clickable sub-menus for easy navigation throughout the app.

4.4.4 The Map

As shown in the diagrams of our proof of concept, the map, and the side-menu takes up the entire main menu screen of the app. The map itself was built with a free-to-use map API by ESRI Maps, and as the size was set to auto-adjust, it automatically filled the available space on the screen.

When the main menu shell is opened, the map would shrink as the available space shrank. In order to create a visually appealing first Alpha, the map was also filled with TRYE icons to visually show the user where the bikes are located.

4.5 Alpha 2 MA

The Alpha 2 was done in time for the second bachelor presentation on the 24th of March. Key features made/redone were:

1. Redefined Login System
2. User-friendly Map
3. A User Page
4. A Booking system
5. The Unlocking System
6. The History Page
7. The Help Page
8. PIN-code system

With the help of user tests, we were able to find weaknesses in our application and resolve them. The test subjects had a varying degree of technical background, and it was the mix that made it possible for us to create a better product.

4.5.1 Planning for the Alpha 2

When planning our second version of the app, we took a different approach to development where we started by planning the entire app through UML and database diagrams. Also, user tests were done on the app to look at how we could improve the app. We also looked at the requirements and had a talk with TRYE about further development and features.

In the second version of the app all the features were planned from the beginning, but we still had to make some changes to deviate from the original planning to make the app better. We still think the planning helped us with getting started on the second version.

This part consists of the diagrams from our planning process. We also had initial UML diagrams drawn which can be seen in appendix A.4.2 and an initial database diagram in appendix A.4.7. Some of the changes we had to make during the planning process were adding extra columns to some database tables and changing the use case diagram for the main menu where we added the booking and unlocking option and removed the payment and settings tab.

We also planned two of our most important features for the MA, which is the booking sequence and the unlocking sequence during the development process. We made sequence diagrams for these to processes to get a better overview. The diagrams are added to the sections about the two features.

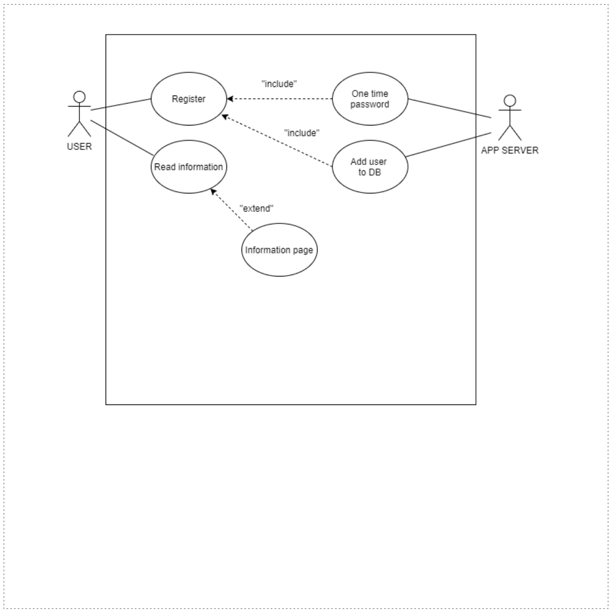


Figure 4.5: Use case diagram for registering process

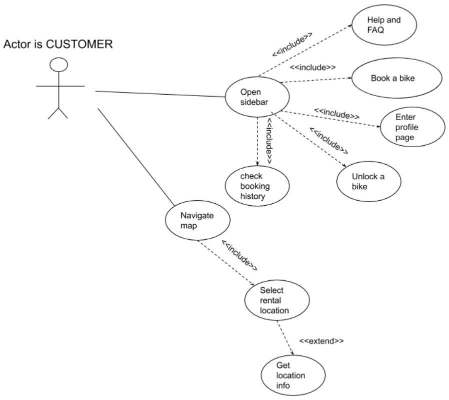


Figure 4.6: Use case diagram for the main menu

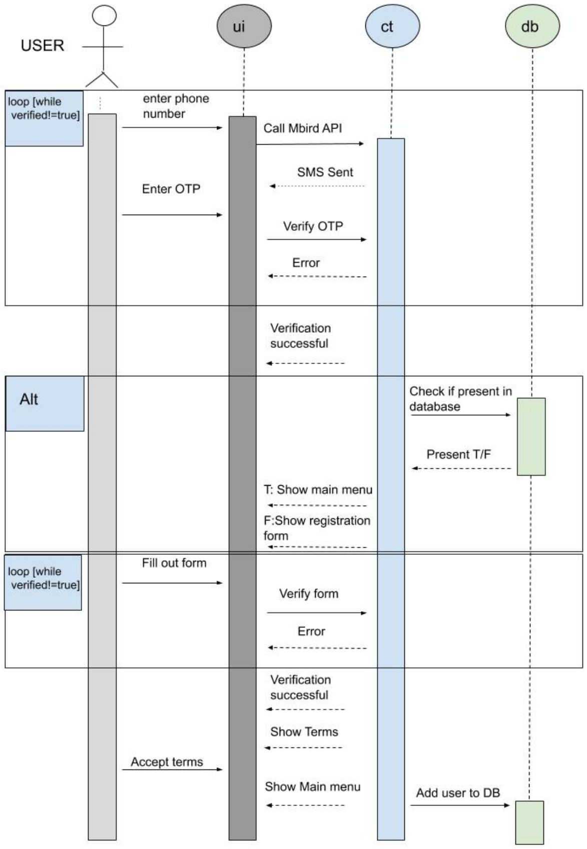
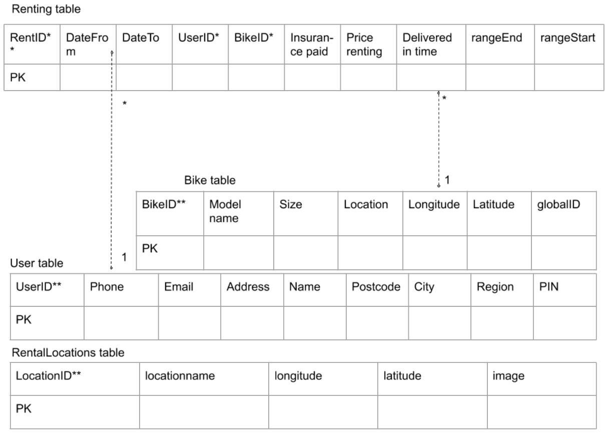


Figure 4.7: Sequence diagram for the registering process

Here is the database diagram where the most noticeable changes from the original diagram found in appendix A.4.7 are adding longitude, latitude, and a global ID to the database. This is used when we communicate with the hardware. We also added rangeStart and rangeEnd to the bookings database. These two entries are used for calculating if bikes are booked in the same time frame. Also, a PIN column is added to the user database to store the different user's PIN-codes.

The relations stays the same and is described in the diagram below, but we have also added a fourth table that has no relations called "rentalLocations", where we store the current rental locations that customers can travel to.



Renting table

RentID**	DateFrom	DateTo	UserID*	BikeID*	Insurance paid	Price renting	Delivered in time	rangeEnd	rangeStart
PK									

Bike table

BikeID**	Model name	Size	Location	Longitude	Latitude	globalID
PK						

User table

UserID**	Phone	Email	Address	Name	Postcode	City	Region	PIN
PK								

RentalLocations table

LocationID**	locationname	longitude	latitude	image
PK				

Figure 4.8: Diagram of database tables and relations

4.5.2 Redefined login system

After consulting with TRYE, we realized that the User Database needed to contain more than just a phone number and an email. TRYE then gave us a list of personal info they needed in case of fraud or needing to contact a customer. This meant that changes needed to be done to the login system, and the redefined login system worked out to the following:

As a User enters our app, the User is requested to enter a phone number. from there the user will have to enter a OTP sent to his/her phone number. If the phone number is successfully verified, the user is then guided over to a registration form asking for a Name, Address, E-mail, etc. If every field is properly filled in, the user is registered as "verified" and is given access to the main menu of the app.

4.5.3 The Map

The Map had no changes since the Alpha 1 with the exception of recreating the icons on the map to be more visible.

We also made the decision of only showing rental locations on in the MA. After a meeting with TRYE where we discusses about the rental locations where the bikes would be stacked at one location in a shed at the ski resorts. This means it would be hard for customers to select a certain bike, which means it is better to visually show the rental locations for the customers on the map.

We also made this decision because of the customers privacy because then we would not have to ask for their location or hide other bikes on the map that are used by other customers. Adding these features would take more planning and development and little effect on the user experience of the MA.

4.5.4 The User page

With the creation of a more complete user registration form, the need for a user page arose. Inside the user page, a user can see the details saved to his/her account. The user page also supports editing user data in case faulty information was put into the initial registration form, or if a user changes address.

We also implemented a delete function where customer can delete their account if they don't want to be customers anymore. But we will still save their old bookings in the bookings database so TRYE can still view the booking history for security reasons.

4.5.5 The Booking system

One of our main requirements for the app was to make a booking system. With the creation of the booking system, the user is able to book a bike. The booking system works as follows: First, the user has to select a start- and end date. Then the user has to pick the preferred bike, and lastly the user has to press the checkout button which says whether or not the bike is available for rent, and if the bike is available the price will show up.

If the bike is available, the user can then proceed to pay for the bike. The payment is done by using the Stripe[7] API. The process is described using a UML sequence diagram below.

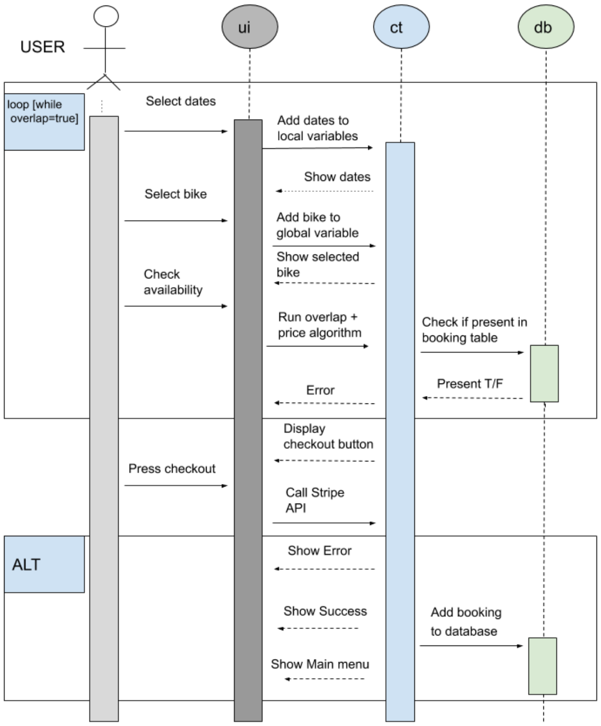


Figure 4.9: Booking sequence

4.5.6 The Unlocking system

The Unlocking System works closely with the booking system. A user should only be able to unlock a bike that is within his/her order period and paid for. The Unlocking System, therefore, checks if today's date is within a rental period. If it is, the bike will pop up as unlock-able, and pressing this button sends a signal to the server asking it to unlock.

The unlocking sequence is described in the diagram below.

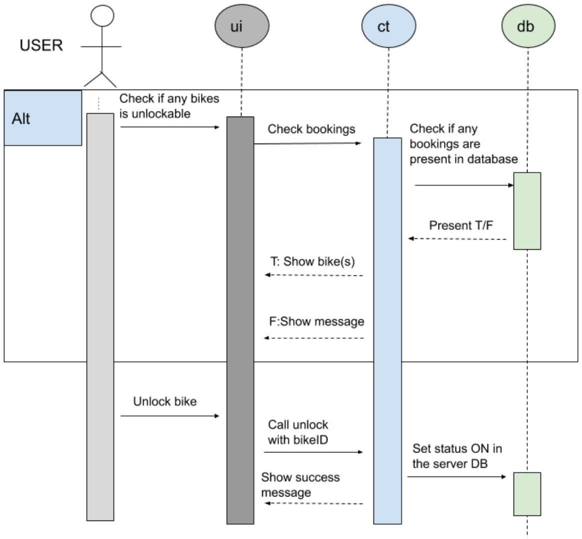


Figure 4.10: Unlocking sequence

4.5.7 The History page

In the History page, a user is able to look at his/her booking history. The information listed in the booking history is as follows:

1. Bike ID (Number)
2. Date From
3. Date To
4. Whether or not the bike is insured
5. Price for that specific booking

4.5.8 The Help page

The Help page contains helpful information such as the phone numbers of TRYE and frequently asked questions.

There was also some final feature fix to the help page before delivering the report. The bike damage reporting system was made in right before delivering the report to complete the MAR-04 requirement.

Since this was not part of any sprint the documentation for this work is put in the report in this section. The work was done and verified by Andreas.

The feature consist of a text area and a button where customers can report any damages or error to bikes. The customer explains the error as good as they can and click report damage.

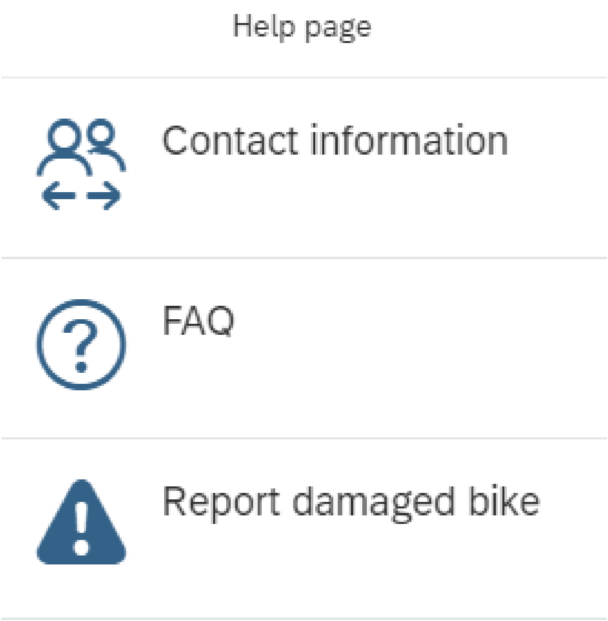


Figure 4.11: Report button added to help page

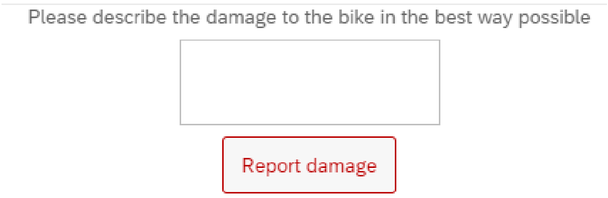
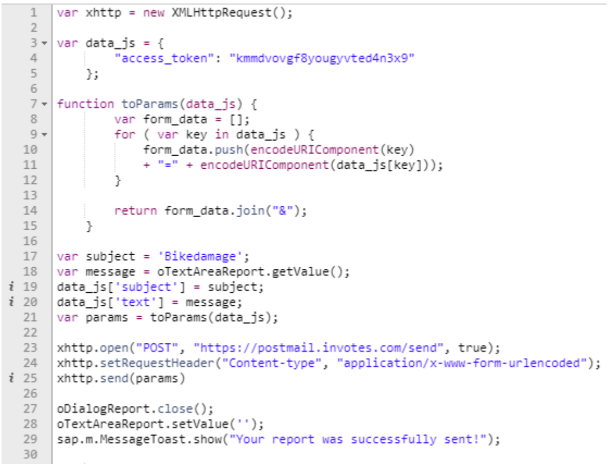

Please describe the damage to the bike in the best way possible

Figure 4.12: The damage report form

When a user is done describing the damage discovered on a bike, they submit the report, and an email is sent to TRYE's email using an email API called Postmail. The API call is done using an Ajax request as shown below.



```javascript
var xhttp = new XMLHttpRequest();

var data_js = {
        "access_token": "kmmdvovgf8yougyvted4n3x9"
    };

function toParams(data_js) {
        var form_data = [];
        for ( var key in data_js ) {
            form_data.push(encodeURIComponent(key)
            + "=" + encodeURIComponent(data_js[key]));
        }

        return form_data.join("&");
    }

var subject = 'Bikedamage';
var message = oTextAreaReport.getValue();
data_js['subject'] = subject;
data_js['text'] = message;
var params = toParams(data_js);

xhttp.open("POST", "https://postmail.invotes.com/send", true);
xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xhttp.send(params)

oDialogReport.close();
oTextAreaReport.setValue('');
sap.m.MessageToast.show("Your report was successfully sent!");
```

Figure 4.13: Calling the email API using Ajax request in javaScript

A test was performed by Andreas where the form was filled out and the submit button was pressed. An email was then received that contained the same text as entered in the report system.

4.5.9 PIN-code system

After spending a good amount of money on messageBird for sending an SMS for each time we tested the application, we found out a personal PIN-code as a second login method would solve the problem of sending out to many unnecessary SMS's.

When customers register for the first time the OTP they received would then become their personal PIN-code which they can use the next time they log in. A tab to change the PIN-code on the user page was also created so that users can change the PIN-code to something they remember. To do this they had to enter their old PIN-code and the new pin 2 times for verification.

We also implemented a recovery system for the PIN-codes where users who have forgotten their personal PIN-code can receive a new one by SMS and that code would update in the database as their new code.

4.5.10 MA Alpha2 Overview and Visuals

To have an overview of the final product we made an extensive class diagram which was started on in appendix A.4.2 which we built on as we developed. Here is the final overview diagram

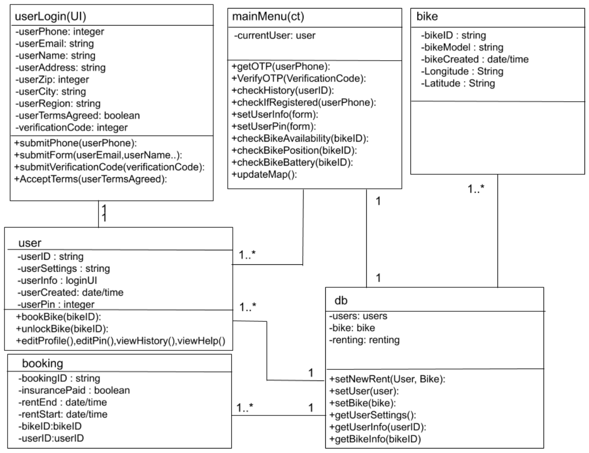
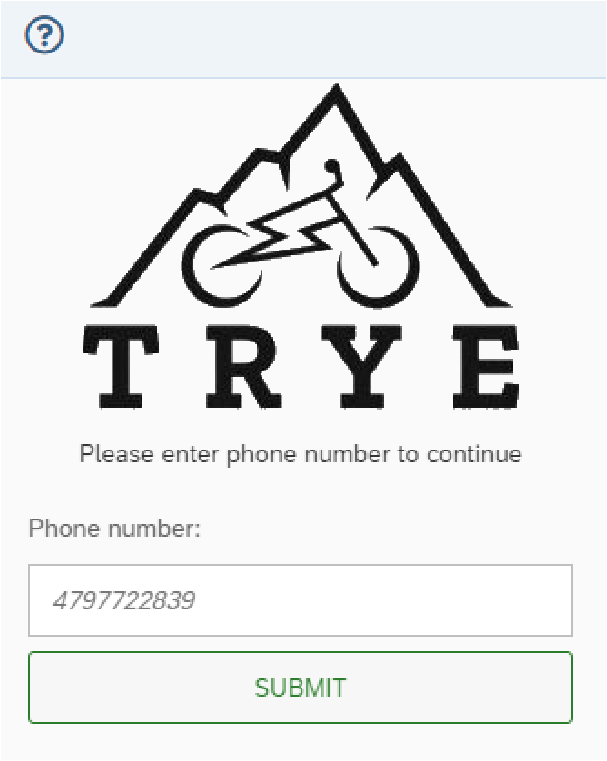


Figure 4.14: Class diagram MA

We have also changed the visual user interface for the users during our development, here you can see how the final MA looks from the registering process to the main menu for a customer.


Please enter phone number to continue

Phone number:

4797722839

Figure 4.15: The initial login page where customers enter their phone number

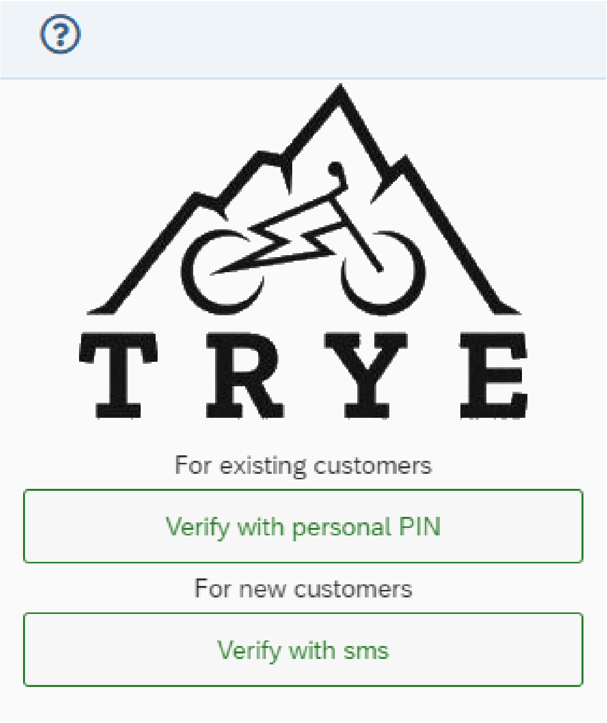
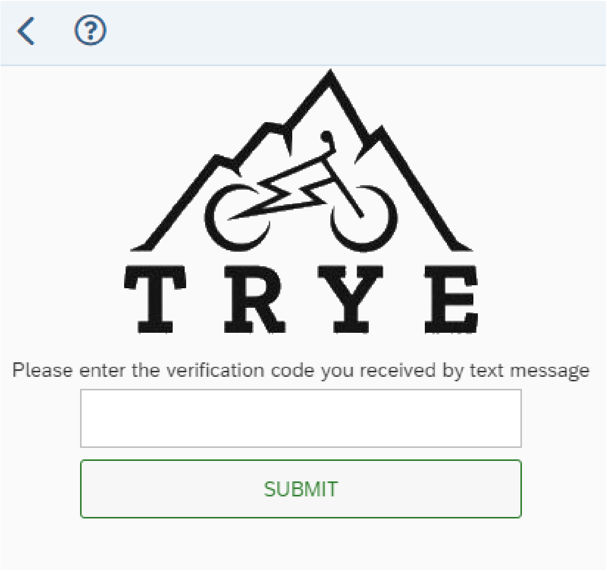
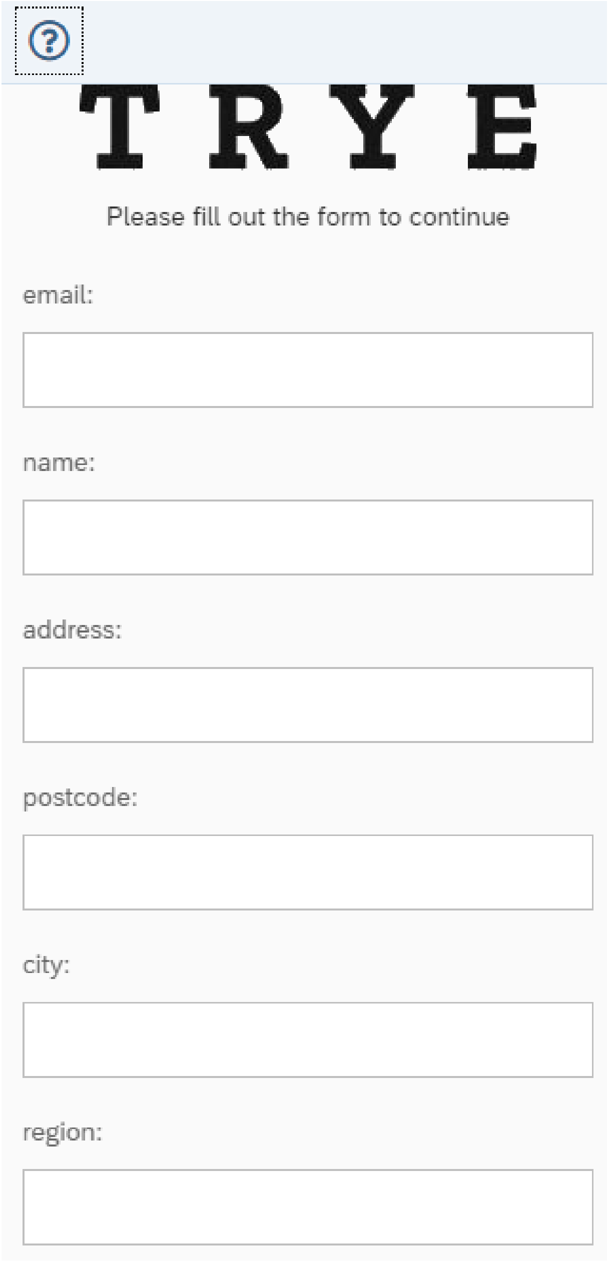


Figure 4.16: On this page existing users can choose to log in with their PIN, while new customers can choose to receive login-information on their phone



Please enter the verification code you received by text message

Figure 4.17: In the verification tab users enter the code they received to verify their phone number



TRYE

Please fill out the form to continue

email:

name:

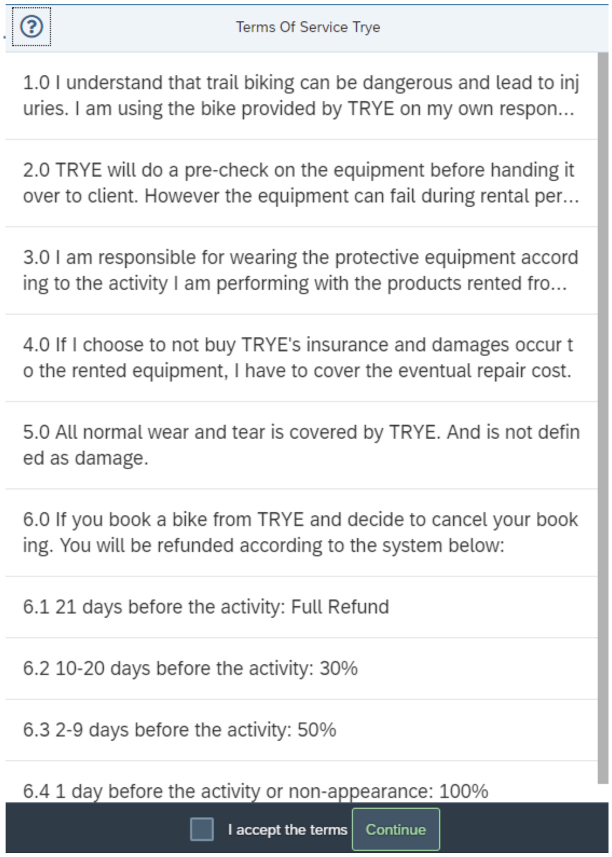
address:

postcode:

city:

region:

Figure 4.18: The registering form for new customers



Terms Of Service Trye

1.0 I understand that trail biking can be dangerous and lead to inj uries. I am using the bike provided by TRYE on my own respon...

2.0 TRYE will do a pre-check on the equipment before handing it over to client. However the equipment can fail during rental per...

3.0 I am responsible for wearing the protective equipment accord ing to the activity I am performing with the products rented fro...

4.0 If I choose to not buy TRYE's insurance and damages occur t o the rented equipment, I have to cover the eventual repair cost.

5.0 All normal wear and tear is covered by TRYE. And is not defin ed as damage.

6.0 If you book a bike from TRYE and decide to cancel your book ing. You will be refunded according to the system below:

6.1 21 days before the activity: Full Refund

6.2 10-20 days before the activity: 30%

6.3 2-9 days before the activity: 50%

6.4 1 day before the activity or non-appearance: 100%

□ I accept the terms

Figure 4.19: Users have to agree to the terms of service before being able to a book bike.

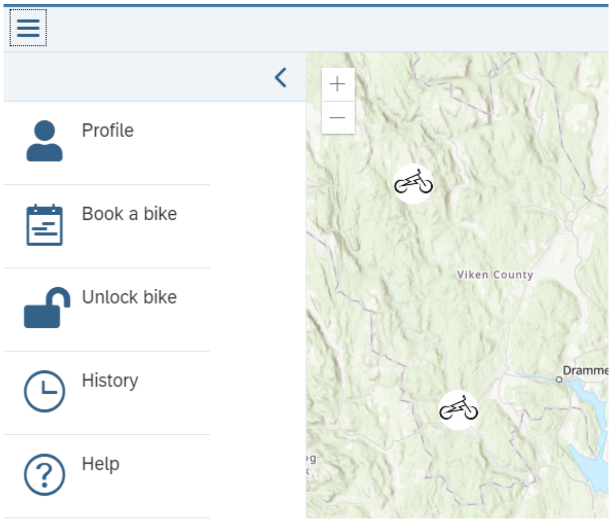


Figure 4.20: A user friendly main menu user interface with a sidebar and the map with a overview of the rental locations

If interested, the visuals of the different functionalities in the main menu can be found in the software user stories in appendix chapter A.4

4.6 Alpha 1 AS

The Admin App Alpha was one of the main requirements for the TRYE app system. The idea of the AS is to give the TRYE admins an overview of the bikes on a map, the current bookings, and manually unlock the bikes. These features come from the sub-requirements TRYE gave us for the AS.

4.6.1 Planning for the AS

To plan the Admin Application we looked at the requirements and tried to start with a simple use case to visualize the features needed in the application.

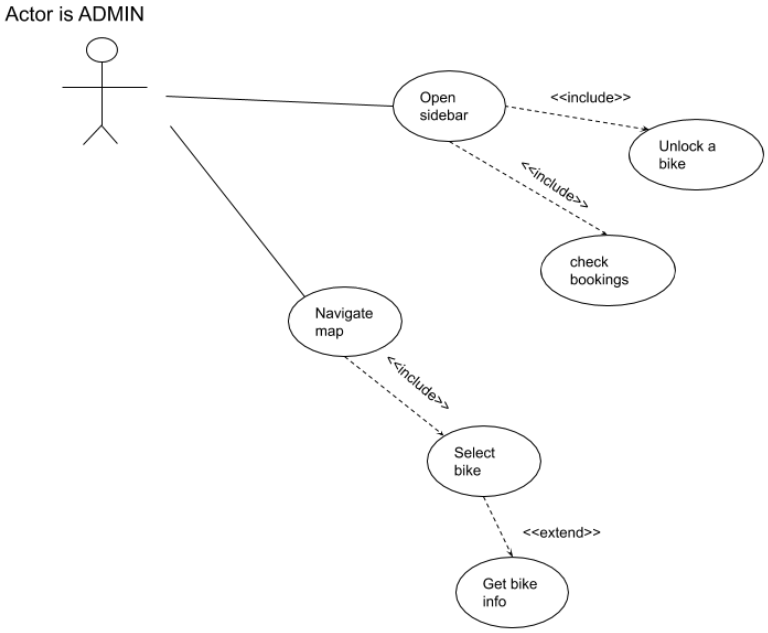


Figure 4.21: Use case diagram of the admin application

This use case diagram was then split in to two sequence diagrams that would explain how the main functionalities of retrieving the bookings and bikes, and unlocking a bike.

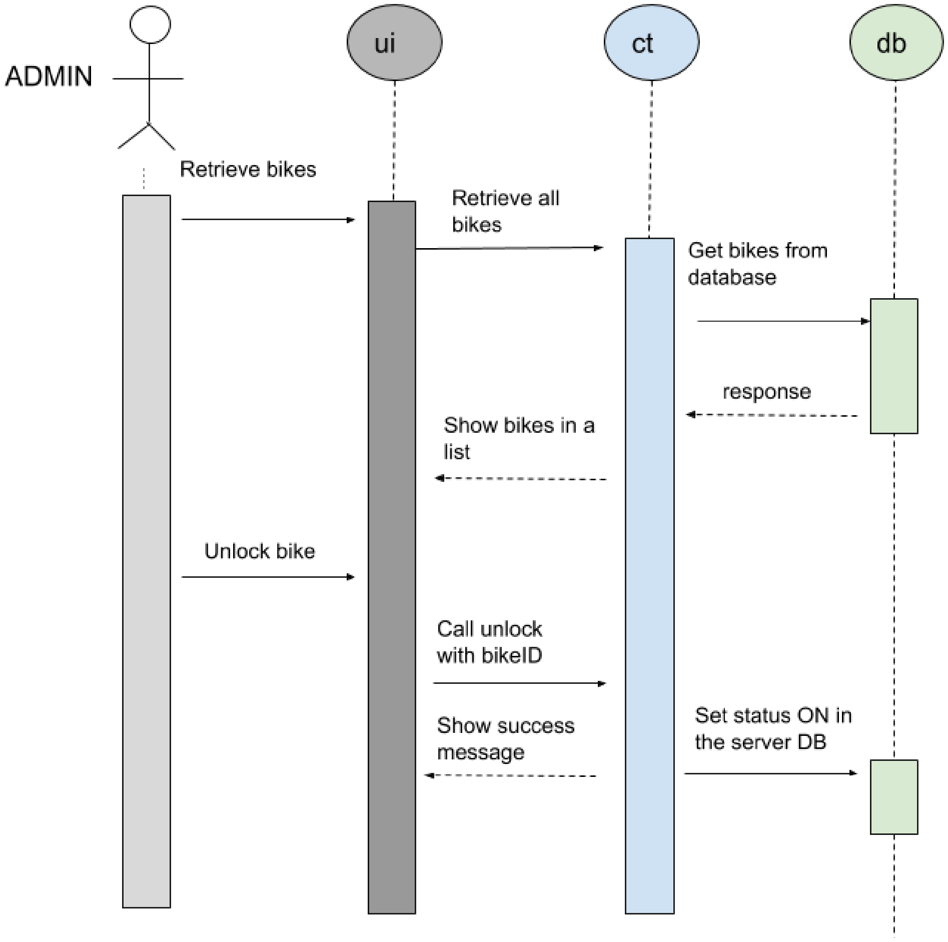


Figure 4.22: Unlocking sequence for admins

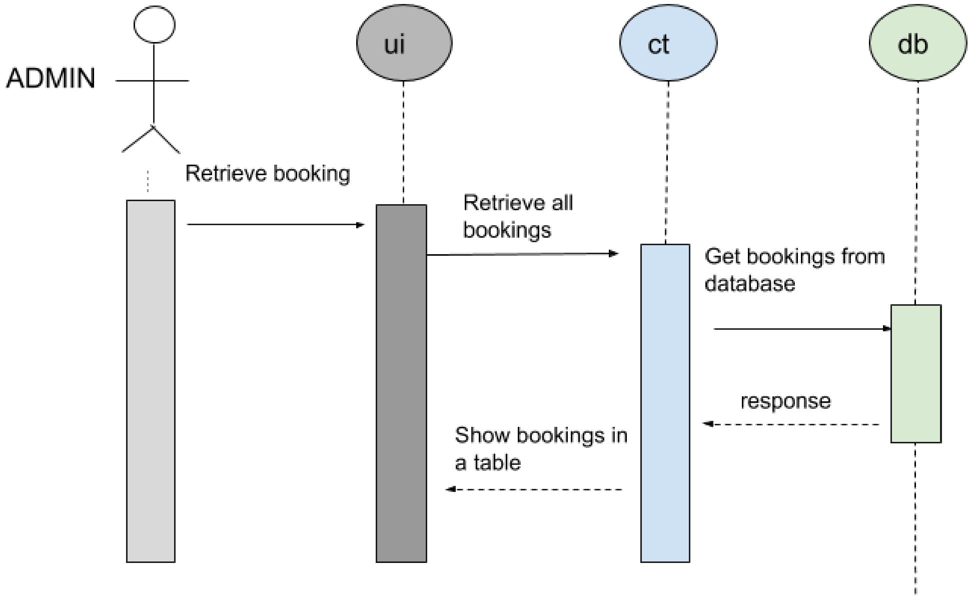


Figure 4.23: Display bookings sequence for admins

Since we had already made the MA we had much of the code from it and the database and server were already set up, which made the development process rapid. Below we have described the systems.

4.6.2 The Unlocking system for admins

The unlocking system is a bit different from the MA. As admins can unlock any bike at any time, it gives the admin freedom to give large groups the bikes they need. This feature is very sought after as it enables a company or a large group to rent the bikes, and TRYE admins can remotely unlock the bikes instead of traveling to the location.

The unlocking system is quite simple, the full list of bikes is retrieved from the database, and shown in a list of clickable buttons. An admin can then click on the bike that they want to unlock, and it will send a signal to the hardware of the bike.

4.6.3 The Booking viewing page

This page was made because TRYE needed to keep track of what bikes are booked. Besides, they also get an overview of which customers have paid insurance, and which customers have not delivered a bike in time.

The bookings view page gets all the bookings from the booking database and displays them in a table so the admin can scroll through the bookings. At the moment no search function is present but it could be a thing for TRYE to implement in the future. Searching for a customer by their phone number would be a great extension to this page.

4.6.4 The Map

The map is where the admins can control where their bikes are at any time. This is important since there are expensive bikes, there could be customers trying to steal or keep the bikes past their rental period.

4.7 Beta 1 MA(Release postponed)

According to our schedule for the software development a beta should have been released before delivering the final report. This has been postponed, and after a talk with TRYE we concluded that the solution is not yet at a stage where it can be released.

The reason TRYE gave us was that if the complete solution is not working (hardware and software) there is no point in a release. Since the hardware is close to finished, we don't have anything other than a working mobile application with a booking system. It has been decided by the Software department that we don't think it is safe to release our Alpha 2 version yet before some security concerns have been tested, and before the app has been tested and checked by a security expert. There are 4 known security concerns we know so far that can be harmful. Some of the known security concerns are:

- Data sent between the app is not yet encrypted so credit card information can be exposed. This is easily fixable with an SSH certificate, but must be done parallel to the release of the native app.
- The Stripe API Key can be exposed since it is located in the app instead of hidden on a protected server.
- As the price is being calculated on the front-end, malicious customers might exploit it and change the price to what they want.
- PIN-codes are not encrypted/hidden in the database, so if an admin account is hacked, the hacker might be able to see the PIN-code of customers.

We estimate that it would take at least another month of development, testing, and verifying before the release of the Beta feels safe and justifiable. TRYE has said that they would want to keep our current software solution and build on it, but they are not sure about the hardware. They said they will use consultants to finish our work, so we hope all our work and efforts have given them a solid foundation to continue developing on. We think the platform is a good solution to their problem and very straightforward.

4.7.1 Software system overview

To have a complete overview of the system and its user stories we made a diagram to visualize the system. We made the diagram with the three components which are the WS, the MA and the AS.

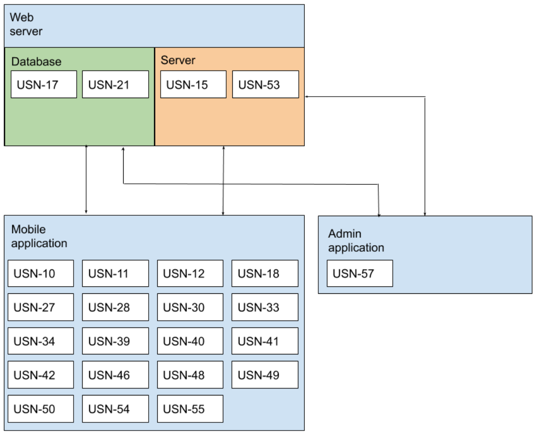


Figure 4.24: Software system

The technology stack for the systems are listed below:

- **Web server:** Linux shell, PostgreSQL and Node.js
- **Mobile application:** HTML5, CSS, javaScript, JSON, Ajax, Rest API, and CRUD
- **Admin system:** HTML5, CSS, javaScript, and CRUD

4.8 Hardware Overview

In Figure 4.25 we can see an overview diagram of the hardware system. The hardware system will consist of an Arduino as the microcontroller, which will receive all the data from various sensors and GPS module and send it to a web server, a voltage converter or power supply which will step down the voltage so our system can handle the voltage, a relay which will be used to turn on or off the bike when the system receives a signal from the web server, and a built-inLTEmodule which will make us able to send data to and from the web server. Our system will also interface with the existing system on the bike, this includes the speed reader and the battery. The battery is then connected to a

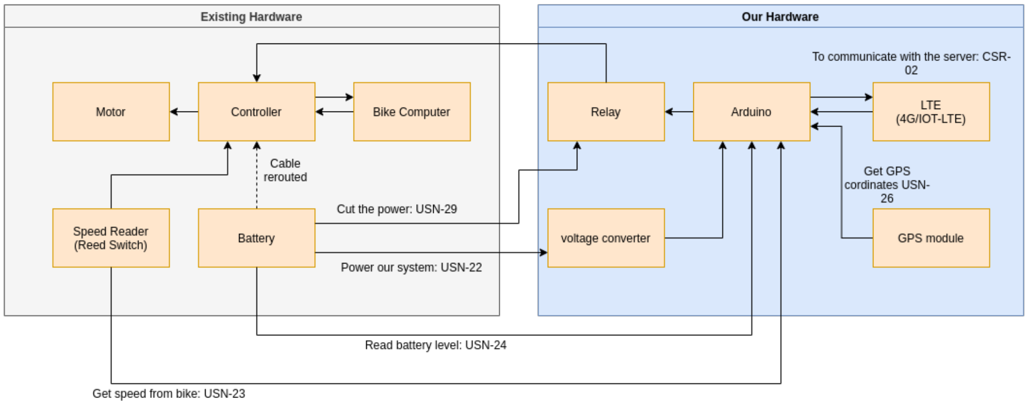


Figure 4.25: System overview of the hardware

battery voltage reader, and the current will be rerouted through a relay. The battery voltage reader will make our system able to get the current bike battery level, so it can be displayed to the end-user. The current will be rerouted to a relay, so the web server can control if the bike is on or off. The lines going between the blocks represent different user stories, except for theLTEmodule which has multiple user stories attached to it and therefore a separate overview diagram as seen in Figure 4.26

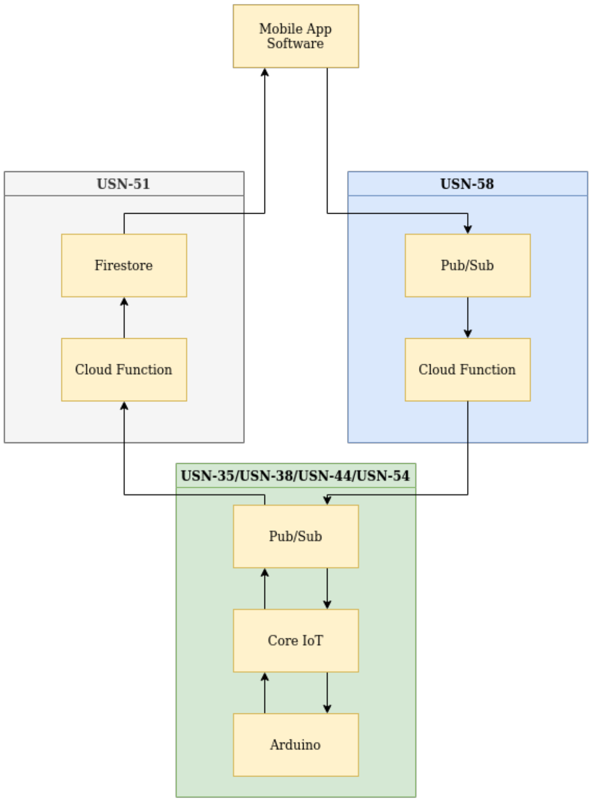


Figure 4.26: Communication overview between the Arduino and the mobile app

In Figure 4.26 or in appendix A.10.2 we can see the communication from the Arduino to the mobile app through the web server. For our system, we decided to go with Google Cloud Platform (GCP) for the web server. Here each of the yellow blocks indicates different GCP services and how they are connected, except the top and bottom blocks. The left side shows the communication from Arduino to the mobile app, and the right side shows the communication from the mobile app to the Arduino. The bigger blocks surrounding multiple yellow blocks show which user story describes the connection between the yellow blocks and how they were configured.

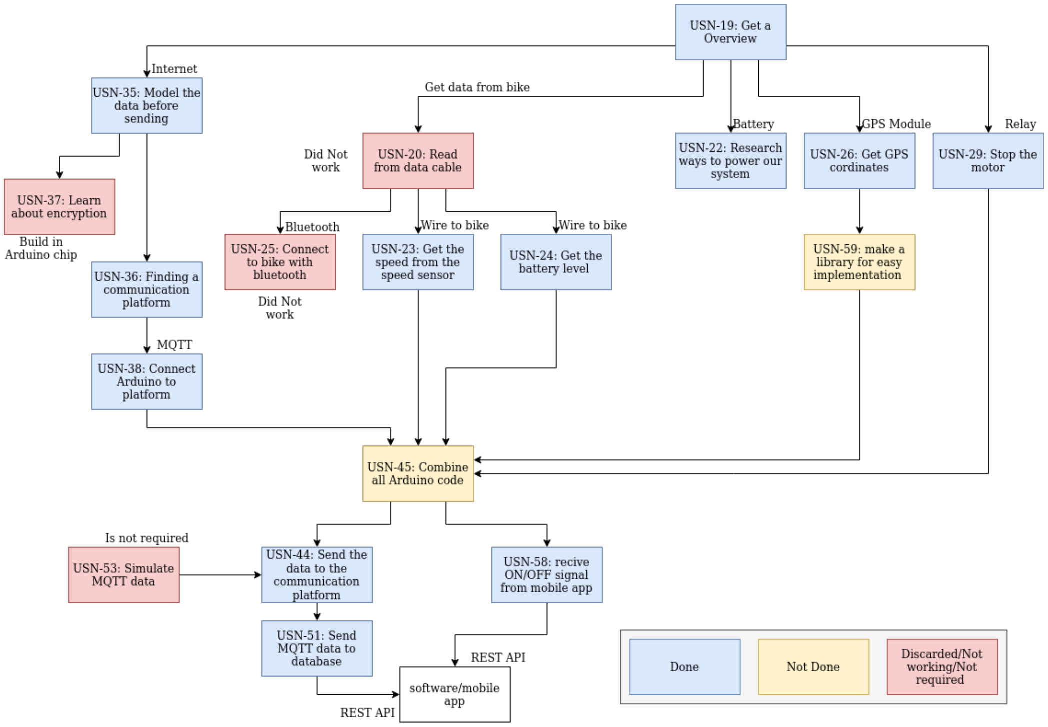

USN-19: Get a Overview
Internet
USN-35: Model the data before sending
Get data from bike
Battery
GPS Module
Relay
Did Not work
USN-20: Read from data cable
USN-22: Research ways to power our system
USN-26: Get GPS cordinates
USN-29: Stop the motor
USN-37: Learn about encryption
Bluetooth
Wire to bike
Wire to bike
Build in Arduino chip
USN-25: Connect to bike with bluetooth
USN-23: Get the speed from the speed sensor
USN-24: Get the battery level
USN-59: make a library for easy implementation
USN-36: Finding a communication platform
Did Not work
MQTT
USN-38: Connect Arduino to platform
USN-45: Combine all Arduino code
Is not required
USN-53: Simulate MQTT data
USN-44: Send the data to the communication platform
USN-58: recive ON/OFF signal from mobile app
USN-51: Send MQTT data to database
REST API
software/mobile app
REST API
Done
Not Done
Discarded/Not working/Not required

Figure 4.27: Every user story used to develop the hardware

In Figure 4.27 or in appendix A.10.1 we can get an overview of all the user stories and how they are connected. It starts with USN-19, which help get an overview of the entire project and where to start, after that it branches into 5 other user stories. The branches talk about how the system was connected to the web server, how our system interfaces with the existing system, how our system is powered, how the system gets GPS data, and how the system starts or stops the motor based on a signal from the web server. After that, all the user stories merge in USN-45 where everything was combined to fit on one Arduino. Blue squares are user stories that have been done, verified and ready for production, yellow squares are things that need extra attaching before production, and red squares are things that did not work out either because it's not required or not feasible to do so.

4.9 Hardware Concept

During the development of the hardware, there were a few different concepts to consider about how to interface with the bike, how to power our system, how the bike and end-user should communicate.

Interface with the bike: To connect with the bike to read e.g. battery level and speed from the bike, there are a few ways to do so. The first is to connect and listen to an already established connection between the bike controller and the bike display. The problem with this is that the communication between the devices can be encrypted or may not follow any official standards, which will lead to a troublesome connection. Another way of connecting to the existing system is with a Bluetooth module built into the bike. The problem with this is the bike has to have Bluetooth to work, and the system will make unnecessary wireless signals which will lead to an easier target for hackers. The third way and the way our system uses is to connect to the battery and speed reader directly. This is probably the easiest way of connecting to the existing system, one problem with this is if we want to connect more modules from the bike to our system will mean more wires and more time setting up each bike. But it also means the system can be modified to any bike.

Power our system: To power our system, we have mainly two ways of doing so. The first is to connect an external battery to the existing system, this will help keep our system running even when the bike battery is out of charge, and we can therefore still see the position of the bike. The problem with this is it will add complexity, cost, and space on the bike. Therefore our system will instead connect directly to the bike battery, without the external battery as our system is used as a prototype to see what's possible.

User to bike communication: As the end-user has to see the position of the bikes we have to think about how our system communicates with the end-user. There are mainly two ways of doing so, the first is to send the data directly from the bike to the mobile app, with a smaller system this is an easy way of sending the data, as the data don't need to be stored or handled other the end-user mobile app. The problem with this is it does not scale. If we have 4 bikes and 4 users each bike as to send one message to each user, giving a total of $4 * 4 = 16$ messages, and this will be exponentially bigger, and will, therefore, add more bandwidth required. If instead, we centralize all the communication to a server we can store all the data, and we don't need to send as many messages. To send 4 bike positions to 4 users we need $4 + 4 = 8$ messages instead of 16, and this will scale linearly and not exponentially

bike opening message: As there already exists systems to track assets with GPS, our system could use one of these instead of communicating with a server over LTE using the Arduino. This would require using the end-users phone to unlock the bike by sending a unlock signal over e.g. Bluetooth. This could be a faster way of developing the system, but it will also mean there exist in total three separate systems on the bike: the existing system, GPS tracking system, and the unlock system. And the GPS module could not be customized to the same extent by using a third party GPS module. But this is still a viable solution for the problem.

4.10 Hardware Alpha 1

Since the implementation for both hardware and software undergo parallel in this project, designing and interfacing the hardware system to the software system and to the existing electrical Mountain bike is a fundamental task for hardware engineers. In this design phase, the main tasks were understanding the hardware system and decomposing it into its subsystems and components. In addition to this understanding of the existing hardware systems of the eMTB is also included.

4.10.1 Identifying the Subsystems

For having a fully complete system, we need to have different subsystems that may communicate with each other or with other systems. The requirements and system functionalities are used to identify the subsystems for the hardware part of our system and the subsystems are shown in Figure 4.28

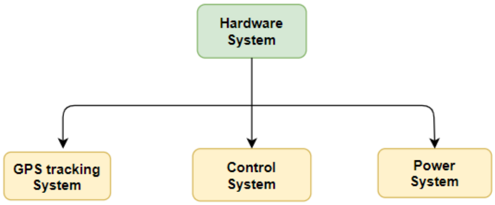


Figure 4.28: Hardware subsystems

GPS tracking system: will communicate with the satellite to fetch location data. The system will be used to update the location of our system.

Power system: will provide an appropriate power supply to our system, subsystems, and components. It helps our system to function without any problem remotely. Power systems should have a power source and voltage regulators to regulate the power and supply to subsystems and components accordingly.

Control System: It is the brain of the system and is responsible for controlling all the subsystems with their components. The control system should enable our system to connect with the internet for communicating with other systems and for exchanging data.

4.10.2 Component selection for Hardware system

After identifying the subsystems as a group we need to select components for the subsystems which will be used to build subsystems that will provide the system functionalities that are mentioned in the product requirements.

Microcontroller: the microcontroller we need for this purpose should accept a SIM(Subscriber Identity Module) card for communication and it should not be big in size for avoiding difficulties during mounting it on the bike. Our group decided to use Arduino as a microcontroller. Arduino is designed with a variety of microprocessors and controllers. It is popular among engineers and hobbyists over the world for projects related to digital and embedded systems. The board is designed with analog and digital pins, which helps to interface and control other systems. Its microcontroller can be programmed in C or C++ on its integrated development environment (IDE). We can find different types of Arduino with different features. For our project we chose Arduino MKR NB 1500.

MKR NB 1500 is the perfect choice for devices in remote locations without an Internet connection, or in situations in which power isn't available like on-field deployments, remote metering systems, solar-powered devices, or other extreme scenarios. The board's main processor is a low power Arm® Cortex®-M0 32-bit SAMD21, like in the other boards within the Arduino MKR family. The Narrowband connectivity is performed with a module from u-blox, the SARA-R410M-02B, a low power chipset operating in the different bands of the IoT LTE cellular range. On top of those, secure communication is ensured through the Microchip® ECC508 crypto chip. Besides that, the PCB includes a battery charger and a connector for an external antenna. This board is designed for global use, providing connectivity on LTE's Cat M1/NB1 bands 1, 2, 3, 4, 5, 8, 12, 13, 18, 19, 20, 25, 26, 28. Operators offering service in that part of the spectrum include Vodafone, AT&T, T-Mobile USA, Telstra, and Verizon, among others. Its USB port can be used to supply power (5 V) to the board. It has a Li-Po charging circuit that allows the board to run on battery power or an external 5-Volt source, charging the Li-Po battery while running on external power. Switching from one source to the other is done automatically.[8]

4G IoT SIM card: This SIM card is a special SIM card that supports IoT services and we chose 4G IoT from Telenor which is the biggest Telecom company that provides a good service in every part of Norway. 4G IoT SIM from Telenor is designed for smart devices that need an internet connection with high speed which uses low data and rely on long battery life. The SIM card works like traditional 2G and 3G but it has better features. Its frequency bands are built on global standards and 4G IoT has the same quality and security features as all other 4G traffic.

GPS Module and Antenna: A GPS module has a tiny processor and antenna

Figure 4.29: Arduino MKR NB 1500 and Its Antenna(source:arduino.cc)

which are used to receive data sent by satellites through RF(Radio Frequency). GPS modules have an antenna that is either inbuilt in or externally attached to it. The antenna needs to communicate with 4 or more satellites to accurately calculate position and time. Things to consider for choosing our GPS module for this project.

- **Size:** GPS module can be found in different size and for our project, we need a Module which is small in size.
- **Update Rate:**The rate at which the module updates its data and update rate is measured in Hertz (Hz).GPS modules mostly have a frequency of 1Hz and more update rate.
- **Power consumption:**Average power consumption for most modules is 30mA at 3.3 V.
- **Accuracy:** It is a measure of giving the correct location data and most modules are able to locate a position within 30 seconds within 10 m radius or below.

For our project, we chose the NEO-6M GPS module. The module is compatible with all Arduino boards and other microcontrollers. The module has four pins VCC, RX, TX, and GND. It communicates with Arduino through RX and TX pins via serial communication. A NEO-6M GPS module with its antenna is shown in Figure 4.30 and it has the following features.
Size: 23mm x 30mm
Update Rate: 1 Hz, 5Hz maximum
Power Supply Voltage:3 V – 5 V
Baud Rate: 9600
Number of Channels: 50 [18]

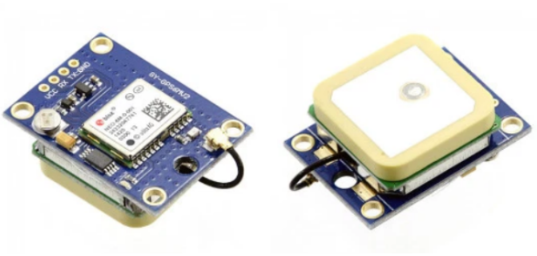


Figure 4.30: NEO-6M GPS Module (source:core-electronics.com)

Relay: is an electrical component that can be used as a switch. It has a set of input pins used to control output terminals. Its main advantage is to control a high power system with an independent low power signal like an Arduino without being damaged. For this project a single channel 5v relay which is shown in Figure 4.31 will be used.

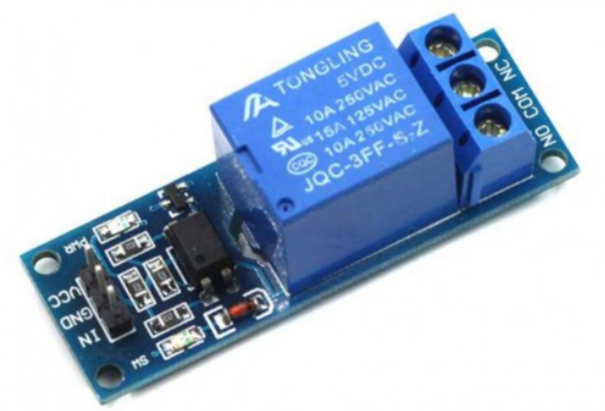


Figure 4.31: A single channel 5 V relay (source:embededstudio.com)

Buck Converter: is used to convert high DC voltage into low DC voltage efficiently. Since many components used in the hardware subsystems use the power of 3.3 - 5 V. Buck converter will be used to manage the power to the components. DC-DC step-Down buck converter which is shown in Figure 4.32 will be used.

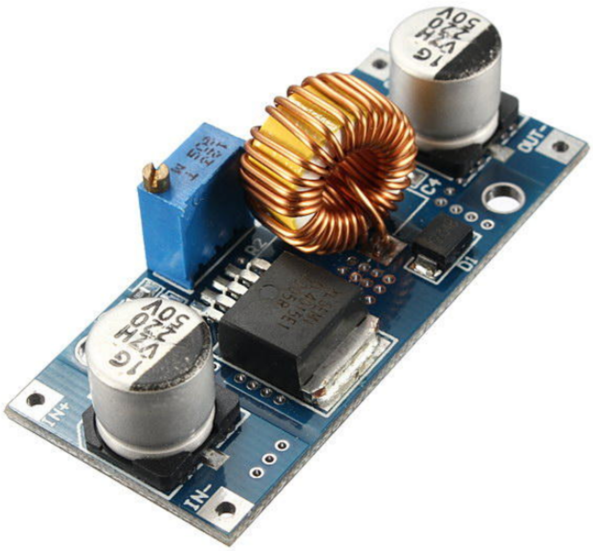


Figure 4.32: DC-DC step down buck converter(source:embededstudio.com)

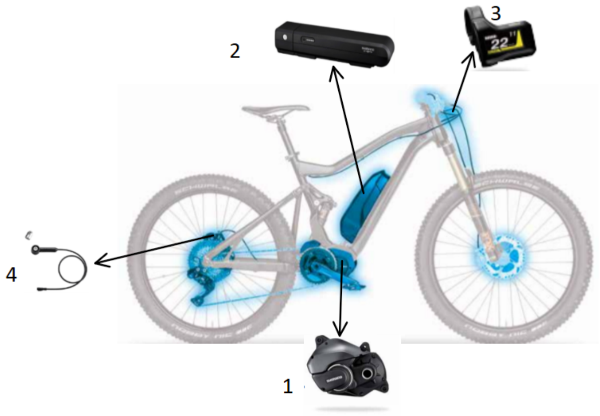
4.10.3 Existing System and its Hardware parts

When we are developing a system, we need to understand how it interfaces with other systems. Doing this is a critical part of the development process, and provides easy to design a system that can interact and mount without any difficulty. The goal is to interface with electric Mountain bikes of type eMTB (E-bike E-Track 27+).

This eMTB is an electric-assist mountain bike and it is powered by Shimano E8000 motor and it has Shimano E8020 500Wh battery which can provide up to 100 km range after fully charged. SM-DUE11-Shimano is used as a magnetic reed switch for measuring the speed of the bike. The bike also has a Shimano cycle computer where users can choose the different modes for the bike and they can see, for example, the battery level and speed. The main subsystems of the electric Bike are shown in Figure 4.33

Motor: The Shimano STEP E8000 250W motor is used in the existing system of eMTB. The motor is designed to provide an assistant by sensing the pedaling. It can provide a power of 250 W and it has 70 Nm of torque. The motor has a connection with the cycle computer and users can choose their mode and the motor adjusts and switches to users mode.

Battery: The eMTB uses Shimano E8020 500 Wh battery. The battery has a capacity of 504 Wh(36v, 14Ah).



1. Motor
2. Battery
3. Cycle Computer
4. Speed sensor unit

Figure 4.33: eMBike subsystems (source:shimano)

Cycle Computer: The cycle computer used for this eMTB is SHIMANO STEPS E8000-Cycle Computer. It has control buttons for powering ON/OFF and changing modes for users. The cycle computer displays bike mode with speed and battery level. The three different modes a user can choose are ECO, BOOST, and TRIAL.

Speed-sensing unit: The eMTB uses SM-DUE 11-Shimano as a speed sensing unit. The unit is a magnetic switch and used for calculating the speed of the wheel and it is mounted on the bike of the back wheel.

4.11 Hardware Alpha 2

During this phase, the development was mostly focused on the bike and its hardware parts, this includes, getting the battery level, getting the speed of the bike, powering the system, and learn the technical limitation of the bike. this will set the system up to be ready to connect to the software part in the beta version

4.11.1 Read Data from Cable on the Bike

To get the speed and battery data from the existing system to our system, listening to the communication between the bike controller and bike computer display was considered a plausible solution. The bike controller is located in the motor housing near the bike pedals, and the bike computer display was located at the handlebar for the end-user to see. As the bike controller display needs data from the bike controller to get battery info, speed, mode, etc there is a cable running between them. This meant it should be possible to connect to this wire and read the signals and decode the battery info and speed from the communication. But this proved to be difficult, and therefore not a viable solution as the system should be easily modifiable to fit other bikes. The problem with this solution was the communication between the units was either encrypted or happening in a non-standard way. This problem can be solved by either talking with Shimano and learn how the units communicate or try with another system that uses a standard protocol to communicate, or it can be worked around by gathering the data directly from the battery and the speed reader. As our solution doesn't require more than the battery level and the speed, the easiest solution was to gather the data our self.

4.11.2 Bluetooth communication

The goal to test the Bluetooth communication of the existing bike system was to extract data that go forth and back from the central controller to the bike computer. These specific data are reading speed, reading the battery level, and updating the firmware. These data are essential for our system.

We visualized that most of the Shimano bike controllers use the smartphone over the Bluetooth to connect to itself to get the speed, battery level, and update the firmware. It was this concept that led us to extract these data and interface them with our system.

Wireshark is a tool used to extract these data, and its function is to capture packets on the wireless channels, like Bluetooth. By using this tool, we can immediately get the bike computer MAC-address and UUID.
However, the problem with the test was that as an indication of the Bluetooth terminal, the bike computer model could not send the speed and battery level via Bluetooth use. The implication was that it was impossible to extract the data from the bike computer. The only thing the Bluetooth could do was that it updates the firmware, customize the bike settings, and diagnosis the problems by use of the E-tube app.

In conclusion, we can say that the bike computer system's complexity steered us to discard this test and lead to looking for another option.

4.11.3 Powering our System

To power our system the voltage first needs to be converted from the 36-Volt from the bike battery to the 5-Volt the Arduino can handle. There are mainly two ways to do so, and that's with either a linear power supply or a buck convert. A linear power supply work by taking an input voltage and running it through a resistor to reduce the output voltage. This makes it simple to use and manufacture and therefore it's cheap. The downside with a linear power supply is the waste of energy, as we have 36-Volt input and the expected output is 5-Volt, the voltage drop over the converter has to be $36 - 5 = 31$ Volt. The efficiency can be calculated using the following formula: $\eta = \frac{V_{out}}{V_{in}} * 100\%$ which gives a $\eta = \frac{5}{36} * 100\% = 13.9\%$ efficiency. The rest of the power will go as heat. This can make the part hot, and therefore need extra cooling, but because of the lack of space this is not a viable solution. instead, the system can use a buck converter. A buck converter is a smaller version of a switching mode power supply (SMPS), which works by turning the voltage on and off at a certain percentage to achieve the desired voltage. If we have an input voltage of 36-Volt and an output voltage of 5-Volt, we can turn the 36-Volt input on and off with a 15% on time and an 85% off time to get the 5-Volt output the Arduino needs, an example of PWM can be seen in figure 4.34. This is an extremely efficient way of converting power. But the parts required are also more complex, this will lead to a slight increase in cost.

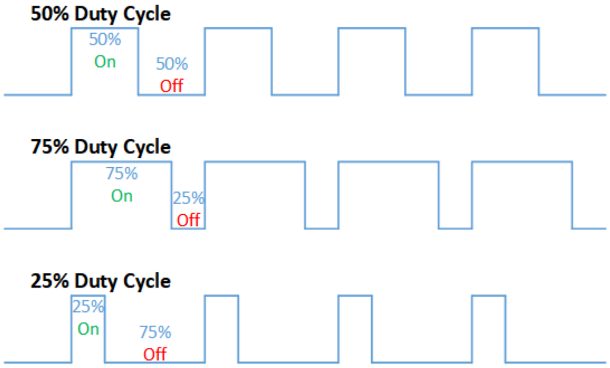


Figure 4.34: figure showing how pulse width modulation works (source:commons.wikimedia.org)

4.11.4 Retrieving speed from the existing system

The objective of retrieving the speed from the existing system is to retrieve the speed and compare it with the speed of our system. The significance of doing this is to control how often the data of GPS coordinate to send to the server using the speed on the bike. It is because when the bike is stationary, it is unnecessary to send the data as frequently as when the bike is moving.

But the problem with the existing bike was that it was laborious to distinguish the signal for speed from the signal for the battery reading since both the cables for speed and battery originated from the central controller and ended to the bike's computer. Moreover, it was complex to utter what kind of coding and decoding protocols used in the existing bike system. We could not even find it in their user manuals and websites.

Since reading speed is a vital task for our system, we must look for another option. From our previous discovery of the existing bike system, we have figured out that the bike has a reed switch mounted on the rear wheel triangle, and the magnet switch, which mounted on the rear wheel. When the magnet gets aligned with the reed switch, we could read a logical value 1, which meant high; otherwise, its logical value is 0, which meant low. By considering this, we could find the difference in time between each passing and use it to find the speed.

To support the achievement of reading the speed, we have used the following components, and these are SM-DUE 11-Shimano, circuit, and Arduino. To read the speed, we considered the magnet attached to the spake of the wheel and the magnet sensor attached to the rear of the bike. As shown in figures 4.36and 4.37, the Arduino connected with the circuit to read the reed switch signal. When the switch is off and on, the Arduino detects, and then count the time between each consecutive 1's. Since one revolution of the wheel meant the two-consecutive 1's in each interval of time, we could calculate the speed that bike travels. The following formula used.

$V = \frac{S}{t}$, V= speed,S= distance traveled, t= time taken
$\omega = \frac{\theta}{t}$, ω = Rotational speed, θ = angle swept, t = time
wheel diameter of our bike 27.5 INCH = 0.7 m is given on the specification.

Lastly, based on the formula above, we made the Arduino code that reads the number of 1's in the given interval for calculating the speed. We also calculated the average speed for every five rotations that were made by the wheel. On top of this, we calculated the momentary speed for the last rotation. If the Arduino doesn't receive the signal within 5 seconds, the program gets a timeout. When this takes place, we can deduce that it is because the bike is stationary, and the speed sets to 0 km/h. The followings are the explanation for the program done. Four different functions that take care of finding the speed of the bike:

- **findDeltaTime()**
- **findAvgDeltaTime()**
- **CheckTimeout()**
- **calculateSpeed()**

Figure 4.35: Illustration for rotational motion of the wheel.(source:SlidePlayer.com)

The following are the function explanations:

- **findDeltaTime():** function, returns the difference in time between each wheel rotation in ms and stores it in an array of an arbitrary length of five. The array is to find the average time later in the program.
- **findAvgDeltaTime():** function, returns the average time calculated by summing the array from findDeltaTime() and then dividing it by five which is the length of the array.
- **CheckTimeout():** the timeout function checks to see if the wheel is not rotating within five seconds. If the bike is stationary, the function **findDeltaTime()** and **findAvgDeltaTime():** will not work as they are dependent on the wheel spinning. So if that is the case, the CheckTimeout overwrites the value from the other functions and sets the change in time to 5 seconds. Which will later return as 0 km/h later in the program
- **calculateSpeed():** function takes the deltaTime and average deltaTime and, and uses that to find the RPM of the wheel and using the circumstance to calculate the speed of the bike. If the speed is below a cutoff speed which is currently set to 2km/h, the speed is set to 0km/h.

In conclusion, the code calculates the speed as expected, but the verification needs the result from the bike computer.

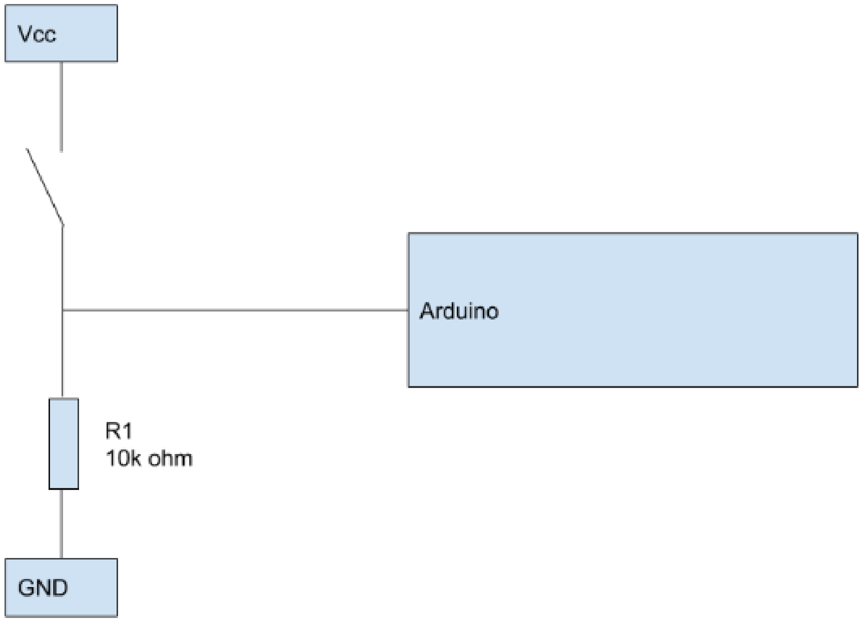


Figure 4.36: A switch with a pulldown resistor.

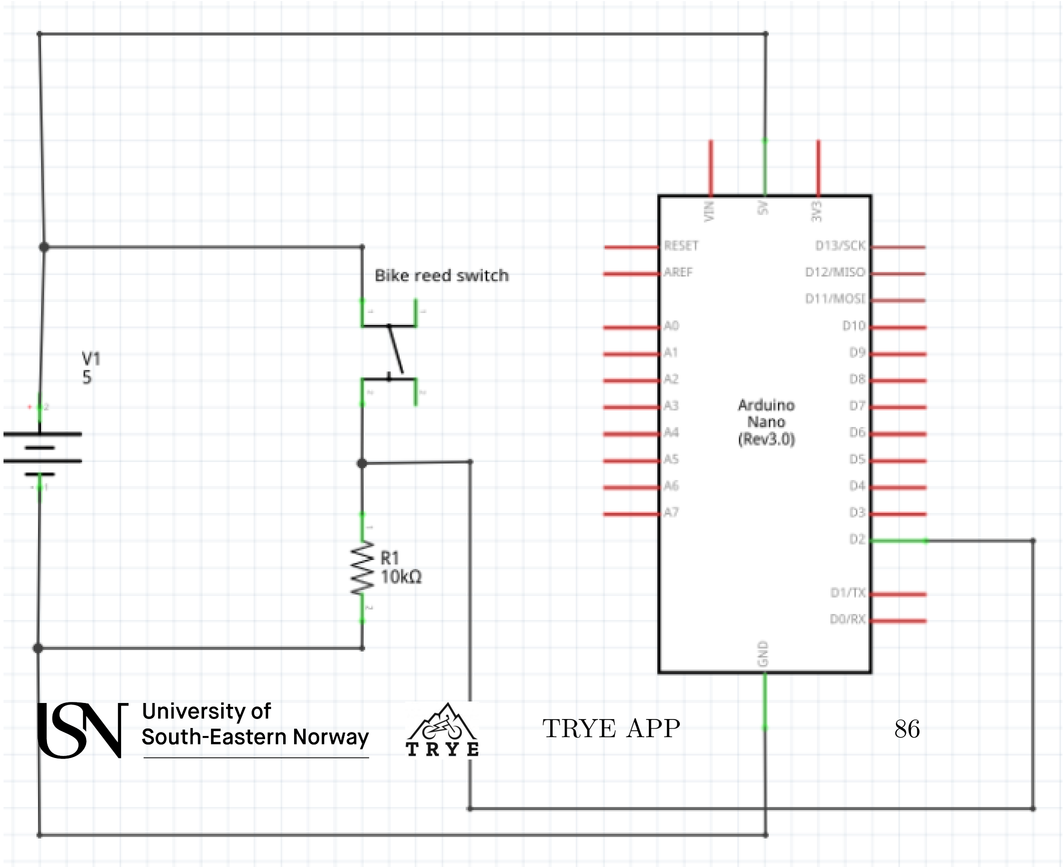


Figure 4.37: A more detailed view of Fig 4.11.

4.11.5 Retrieving Battery Level from the Excising System

The purpose of reading the battery level is to read batter from the exiting bike controller system and provide the readings to the end-users so that they know how much power is left on the bike before they rent it for use.

But the problem with the existing bike controller system was that its controller system was locked, and we could not interface our system with the existing bike controller system. Therefore, the problem drove us to design our battery reading system. We, as a hardware engineer, believe that reading the battery level is a critical task and benefits both the end-users and the company itself. The company benefited by frequently checking all the bikes if the bikes had enough amount of battery. In case the bikes don‘t have enough batter, the maintenance crew could charge it before the next user start. Therefore, the users could always be provided bikes with full of battery. However, when the battery level probably gets lower and not enough battery is available for the users while they are in the field, they could readily know and ride it to the charge station at once.

We designed our reading battery system in a way that we first designed the circuit which used to connect the Arduino with the existing battery system. Arduino is a power source for our reading battery level system and has a maximum of 5 V.

Since the maximum voltage of the existing bike system is 36 V, it is unsafe and risky to connect it straight with Arduino as it only handles 5 V. To avoid such an issue we used the voltage divider, and slimmed the voltage of the existing system down to the corresponding Arduino of 5 V as shown in figure 4.38.

As shown in figure 4.39 below, by using a connection a resistor of 10 Kohm with 1,6 Kohm we found the output voltage of the existing system which corresponded with the voltage of Arduino, which ranges from 0-5 V. Since the Arduino has a 10-bit ADC, the firmware reads the ADC signal that ranges from 0- 1023 bit and maps it 0- 36 V. Following the conversion, the Arduino used to map 0 -36 V to 0- 100% battery level.

We could figure out how the Arduino knows 0% in Volt by discharging the battery and measuring the voltage level using a voltmeter and changing the parameters in the code to match the results. The battery level can then be converted to an easy to understand battery indicator as seen in Figure 4.40. The battery-level indicator in black represents a negative charge, whereas the white color represents the full charge. It means that 0% battery is empty, whereas 100- 81% full charge.

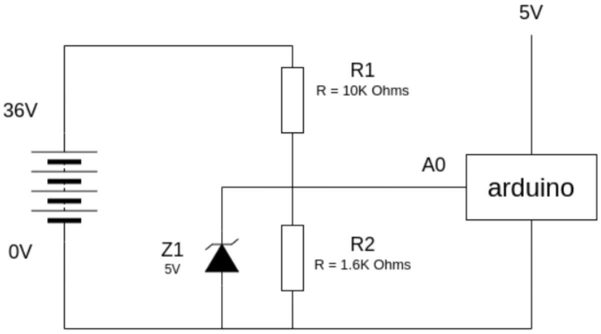
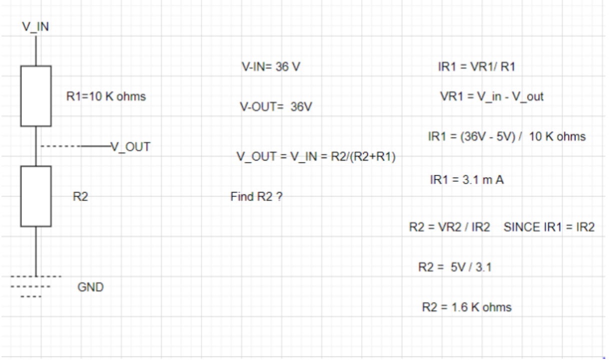


Figure 4.38: A voltage divider giving a 0-5 V on A0.



V-IN= 36 V

V-OUT= 36V

V_OUT = V_IN = R2/(R2+R1)

Find R2 ?

IR1 = VR1/ R1

VR1 = V_in - V_out

IR1 = (36V - 5V) / 10 K ohms

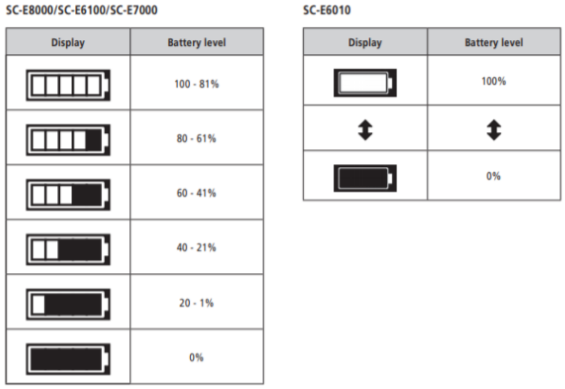
IR1 = 3.1 m A

R2 = VR2 / IR2 SINCE IR1 = IR2

R2 = 5V / 3.1

R2 = 1.6 K ohms

Figure 4.39: Calculation of R2 to find output voltage.



SC-E8000/SC-E6100/SC-E7000

Display	Battery level
▯▯▯▯▯	100 - 81%
▯▯▯▯▮	80 - 61%
▯▯▯▮▮	60 - 41%
▯▯▮▮▮	40 - 21%
▯▮▮▮▮	20 - 1%
▮▮▮▮▮	0%

SC-E6010

Display	Battery level
▯	100%
↕	↕
▮	0%

Figure 4.40: The battery level on existing system.(source:shimano)

4.12 Hardware Beta 1

During the beta 1 phase, the development was mostly focused on setting things together and communicating with the software system. This includes putting all the code on one microcontroller, starting and stopping the motor, getting GPS coordinates, and sending everything to a database in the cloud, ready to be retrieved.

4.12.1 Retrieving GPS Coordinates

The objective of testing the current position and real-time by use of GPS coordinate is to realize and check if the GPS coordinate code gives the same result as expected. The main target is the end-users, and the maintenance crew must be able to locate the eMTB. Therefore, the bike needs to get the GPS coordinate, and store it, and send it to the server. More info given in appendix A.11.5.

However, at this stage, we don‘t mind getting data that we could store on the bike. But, we stick around to search and check the approximate location for the current address and real-time while the GPS module is receiving data from satellites. The concern to do this is to get the latitude, longitude, and the time that we reuse it in the class library and create a return function that is ready to send to the server.

To achieve the objective, we have used the following essential materials. These are laptop, USB serial communication port, NEO-6M GPS module, and its four jumper wires and external antenna, and the Arduino Mega.

For the application, as shown in figure 4.41, we should connect the Arduino with four jumper wires from the GPS module. The GND of the GPS module to GND of Arduino, RX of GPS module to TX of Arduino, TX of GPS module to RX of Arduino, VCC of GPS module to 5 V of Arduino. The USB port connects the Arduino with the laptop to display data on the screen. The external antenna uses to receive the signal from the satellites. We also downloaded two important libraries for GPS to work in Arduino IDE. These are the SoftwareSerial library and TinyGPS plus. These libraries provide most of the NMEA GPS functionality and avoid any mandatory floating-point dependency and ignore all except the few key GPS fields [1].

Following the connection, we download the GPS coordinate code into the Arduino and see for results. When the GPS module receives data, the module starts to blink the blue-light and the data displays. As shown in figure 4.42, the data for the latitude, longitude, and time displayed on the screen of the computer and these data, as shown in figure 4.43, converted into map information and exhibited the current address and real-time as expected [3].

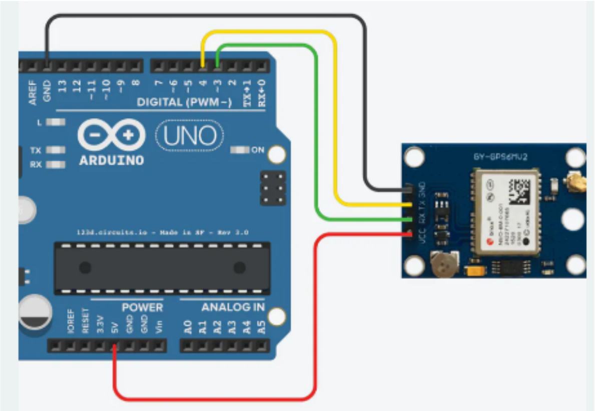
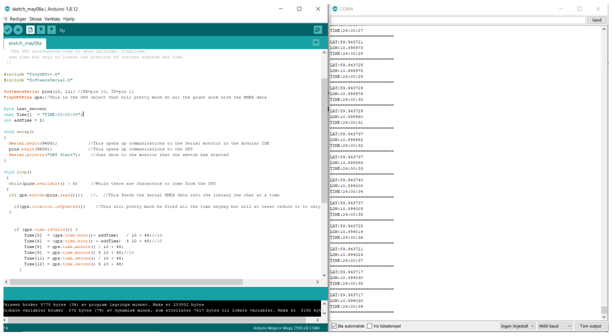


Figure 4.41: The interface of Arduino with GPS NEO-6M Module. (source:arduino project hub)

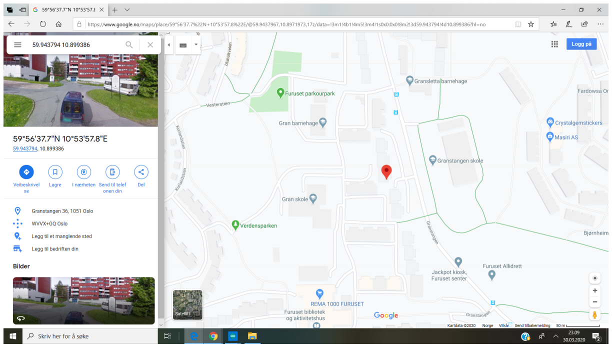

sketch_may08a | Arduino 1.8.12
Fil Rediger Skisse Verktøy Hjelp
Ny
sketch_may08a
COM4
Send
☑ Bla automatisk
□ Vis tidsstempel
Ingen linjeslutt
9600 baud
Tom output
Arduino Mega or Mega 2560 på COM4

```
 -The GPS coordinates code to show latitude, longitude,
 and time but only to locate the position of current address and time.
*/

#include "TinyGPS++.h"
#include "SoftwareSerial.h"

SoftwareSerial pins(10, 11); //RX=pin 10, TX=pin 11
TinyGPSPlus gps;//This is the GPS object that will pretty much do all the grunt work with the NMEA data

byte last_second;
char Time[]  = "TIME:00:00:00";
int addTime = 2;

void setup()
{
  Serial.begin(9600);             //This opens up communications to the Serial monitor in the Arduino IDE
  pins.begin(9600);               //This opens up communications to the GPS
  Serial.println("GPS Start");     //Just show to the monitor that the sketch has started
}

void loop()
 {
  while(pins.available() > 0)     //While there are characters to come from the GPS
 {
  if( gps.encode(pins.read())){   //,  //This feeds the serial NMEA data into the library one char at a time

    if(gps.location.isUpdated())    //This will pretty much be fired all the time anyway but will at least reduce it to only
  {


    if (gps.time.isValid()) {
        Time[5]  = (gps.time.hour()+ addTime)   / 10 + 48;//10
        Time[6]  = (gps.time.hour() + addTime)  % 10 + 48;//10
        Time[8]  = gps.time.minute() / 10 + 48;
        Time[9]  = gps.time.minute() % 10 + 48;//10
        Time[11] = gps.time.second() / 10 + 48;
        Time[12] = gps.time.second() % 10 + 48;
      }
```

```
Skissen bruker 8778 bytes (3%) av program lagrings minnet. Maks er 253952 bytes
Globale variabler bruker  575 bytes (7%) av dynamisk minne, som etterlater 7617 bytes til lokale variabler. Maks er  8192 byt
```

```
TIME:24:00:27
--------------------------
LAT:59.943721
LON:10.898973
TIME:24:00:28
--------------------------
LAT:59.943725
LON:10.898975
TIME:24:00:29
--------------------------
LAT:59.943729
LON:10.898976
TIME:24:00:30
--------------------------
LAT:59.943729
LON:10.898980
TIME:24:00:31
--------------------------
LAT:59.943737
LON:10.898982
TIME:24:00:32
--------------------------
LAT:59.943737
LON:10.898990
TIME:24:00:33
--------------------------
LAT:59.943740
LON:10.899000
TIME:24:00:34
--------------------------
LAT:59.943737
LON:10.899003
TIME:24:00:35
--------------------------
LAT:59.943725
LON:10.899019
TIME:24:00:36
--------------------------
LAT:59.943721
LON:10.899023
TIME:24:00:37
--------------------------
LAT:59.943717
LON:10.899030
TIME:24:00:38
--------------------------
LAT:59.943717
LON:10.899030
TIME:24:00:39
--------------------------
```

Figure 4.42: When latitude, longitude, and time is displayed.



https://www.google.no/maps/place/59°56'37.7%22N+10°53'57.8%22E/@59.9437967,10.8971973,17z/data=!3m1!4b1!4m5!3m4!1s0x0:0x0!8m2!3d59.943794!4d10.899386?hl=no

59.943794 10.899386

59°56'37.7"N 10°53'57.8"E

59.943794, 10.899386

Granstangen 36, 1051 Oslo

WVVX+GQ Oslo

Figure 4.43: Location for current address.

4.12.2 Controlling the motor with Arduino

Controlling the motor using a microcontroller is important for our project to control the bikes. The controller should turn the motor ON only for authorized users. For this project we need to find a mechanism to stop and start a motor using the Arduino MKR NB 1500.

A relay will be used as an electrical switch between the motor and the battery. To control a high power system with an Arduino we need to isolate the Arduino with the help of a relay. We control the relay state by using Arduino and the relay state will be used to control the device state either in ON or OFF modes.

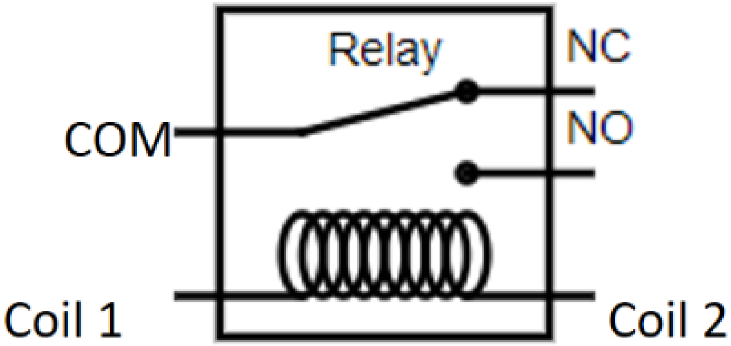


Figure 4.44: Pins of a Relay

Relays are designed for handling and switching high-voltage or high power circuits. They have an electromagnet that can be energized and results in a switch to close or open. As shown in Figure 4.44, relays have 5 pins (COM, Coil 1, Coil 2, NC, NO) and among these 3 pins (NC, COM, and NO) will be connected to the device to be controlled. At the COM terminal electricity enters the relay and NC and NO terminals are used to turn ON and OFF devices. Between pins of Coil 1 and Coil 2, there is an electromagnet coil, and whenever a current passes through them the electromagnet charges and the internal switch connects COM and NO pins as a result the device will be in ON state.

Our main goal is to control the motor of the Shimano eBike by using an Arduino and a relay. As we can see on the wiring of the Shimano eBike in Figure 4.45, the battery is connected directly to the motor by green wire. In normal operation when we press the start button of the bike the motor will get power from the battery directly. Our plan is to use an Arduino to send a control signal ‘0’ or ‘1’ to the relay, as a result the motor will be either in ON or OFF state. Adding the relay and Arduino at the middle of the existing wiring diagram will replace the task of the start button, which means to start the motor we just need to send logical value ‘1’ to the pin of an Arduino where the relay is connected and the relay will act as a switch to turn ON the motor.

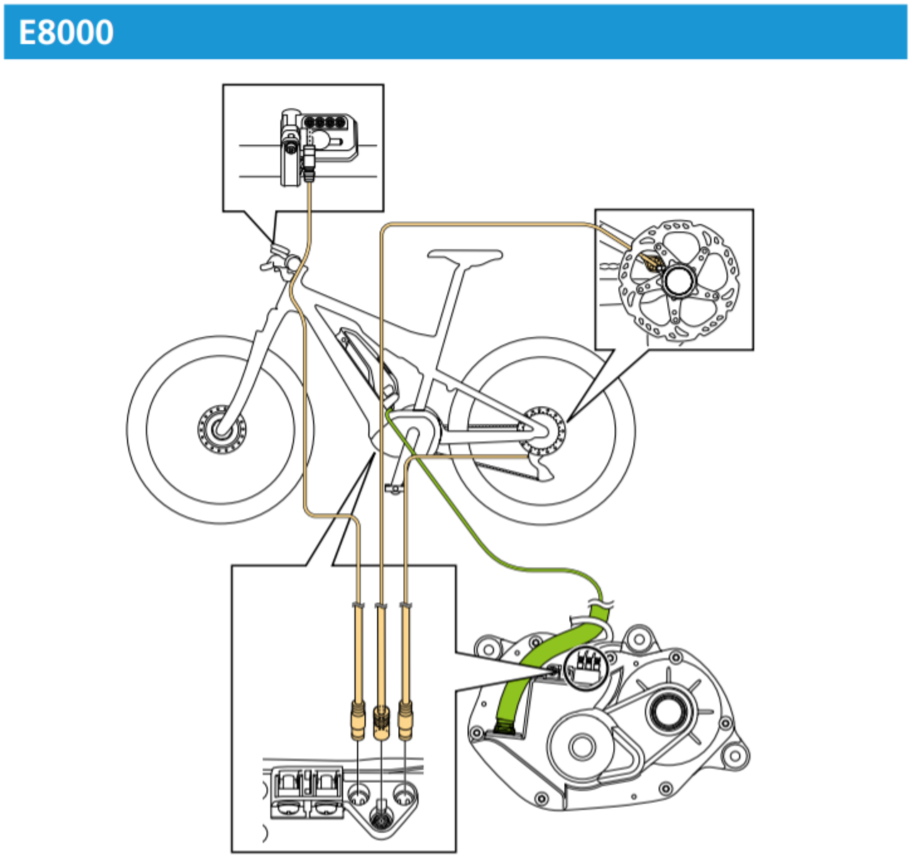


Figure 4.45: Shimano eBike wiring (source:shimano)

Controlling the motor using an Arduino will help to manage control of a bike from our cloud platform by sending commands to the Arduino board to control the state of the motor either ON or OFF. Having these functionality helps us to control our bikes remotely based on the user status, for example users who have completed the renting process on the user app should get the access to turn ON/OFF the bike motors.

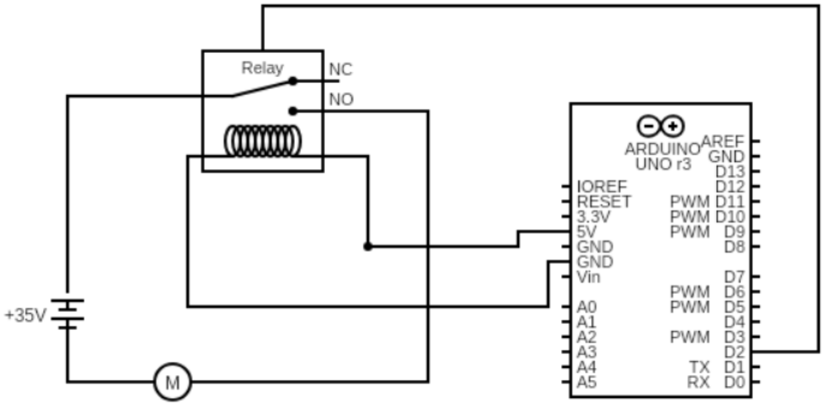


Figure 4.46: Connection between Relay, Arduino, Battery, and Motor

Based on the connection in Figure 4.46 what we need is to send a digital signal either ‘1’ or ‘0’ to digital pin number 2 on the Arduino to trigger the relay to act as a switch and control the motor. Sending ‘0’ will make the circuit to stay as it is where the COM pin and NC are connected and the motor will not get power from the battery and it will be in OFF state. If ‘1’ is sent to the relay the coil will be energized and the relay will switch by connecting the COM with NO and the motor will get power from the battery and it will be in ON state.

4.12.3 Modelling the data

Before sending the data, it’s important to find a standard way of communicating. This makes it a lot easier to send, receive, and process messages sent between nodes. Probably the most used way of structuring the data is with JSON. But to save bandwidth the data could also be sent without any header, so just the raw data is transferred, this makes it harder to understand, process and debug if something goes wrong, an example would be: (1,1.123456, 2.123456, 30), this contains no information about what is what, and the same message in JSON would look like:

```
   [
”bike_id”: 1
”lon”: 1.123456
”lat”: 2.123456
”bat”: 30
]
```

In the JSON message it's clear what the message content, this also makes it easier to update the message in the further if more information is added. So, in this case, JSON seems to be the best option, even with the higher bandwidth cost.

4.12.4 Connecting the Arduino to the internet

For connecting the hardware with the internet, the antenna should be connected to the MKR NB 1500 board and the 4G IoT sim card should be mounted to the board. An Arduino code is used for connecting our board MKR NB 1500 to the network through the 4G network. We included a library called MKRNB. Structure of the MKRNB library:- the library comprises different classes with various functions that provide different functionalities.

- **NB class-** helps the board modem to connect to the network by using the pin number of the sim card.
- **NB-SMS class**- it enables the board modem to send/receive SMS messages.
- **GPRS class-** it helps for connecting to the internet.
- **NBClient**- includes an implementation for a client. This class can create a client that can connect to a server by using GPRS class functions to send and receive data.

Our board operates in 4G and it has a self embedded modem that enables the board to transfer data from serial port to the network. The library abstracts the low level of communication between the modem and the Simcard.

4.12.5 Sending data from the device to the server

Sending speed data, battery level, and location data from the bike to the server is very important for our project. Since the group decided to use the Google Cloud Platform as our server, the data should be sent to this platform from our IoT board. Different tasks and configurations are made on the Google Cloud Platform and on the board to accomplish the task and those configuration tasks are:-

1. Creating a project on the GCP.
2. Configuring and adding the board to a specific project on the platform.
3. Choosing an appropriate protocol for communication between the board and the platform. Where we have the possibility to choose MQTT or HTTP.
4. Generating a public key on the board by running an Arduino code (ECCX08JWSPublicKey).

5. The public key is stored both on the board and on the cloud platform. The key helps for securing communication between the board and the platform and it should be kept secret.

Since MQTT messaging protocol is used for setting the communication between the board and the cloud platform, and Arduino code is loaded with the MQTT library. The following things should be accomplished by the Arduino code for sending the data from the device.

- Connecting the device with the internet.
- Establish MQTT connection between the board and the MQTT server of the GCP which is "mqtt.googleapis.com".
- use mqttClient.subscribe("/devices/" + deviceId + "/states") function to subscribe to Sub/pub topic called state where the device can publish its state data by using a function that can send sensor data in the format explained in the section 4.12.3.
- For sending the data, using a self-made function publishMessage(), this function should send the message which contains the bike-ID, longitude, latitude, and battery level of the bike based on the chosen format. The message part will be put into the MQTT packet as payload together with other packet parameters.

The MQTT server for the Google Cloud Platform is "mqtt.googleapis.com" and the data that will be sent from the bike IoT board will be stored on this broker at Sub/pub and any client that is subscribed for this message will receive the data automatically after the message is published. After the data is published to the MQTT server for this project, what happens next is explained in section 4.12.12.

4.12.6 Managing security during data exchange

While building an IoT application, thinking about security is very important at all levels. The vulnerability at some point in your IoT infrastructure has a big risk of affecting the stakeholders that are related to your IoT application.
As a group understanding the IoT device and Security technologies the device supports will help to know the level of security that can be provided. This may help to identify the possible threats that may happen due to the IoT device and it will be the first step for adding security features to the system if necessary.
Most IoT devices have inbuilt crypto chips for managing and storing keys. These crypto chips facilitate key-based authentication for communication and data exchange in IoT applications. As a group, we identified the crypto chip that is embedded in our Hardware device and identified the features for the crypto chip and we understand the cryptographic operation of the chip.

The Arduino board we use for this project (MKR NB 1500) has an embedded microchip ATECC508A which is used as a crypto chip. This chip is used for generating and storing 256 bit ECC keys.

The ATECC508A implements a complete asymmetric (public/private) key cryptographic signature solution based upon Elliptic Curve Cryptography and the ECDSA signature protocol. The device is designed to securely store multiple private keys along with their associated public keys and certificates. The signature verification command can use any stored or an external ECC public key. Public keys stored within the device can be configured to require validation via a certificate chain to speed up subsequent device authentications. Random private key generation is supported internally within the device to ensure that the private key can never be known outside of the device. The public key corresponding to a stored private key is always returned when the key is generated and it may optionally be computed at a later time.[16]

Arduino has open-source libraries and codes that can be used for generating random key pairs(private key and public keys). The Arduino codes that are designed for crypto chips of ECC508 and ECC608 are the following.

- **ECCX08CSR**:- helps to generate a CSR(Certificate Signing Request) for a private key generated in a crypto chip slot.
- **ECCX08JWSPublickey**:- helps to generate a public key for a private key generated in a crypto chip slot.
- **ECCX08SelfSignedCert**:- helps to generate a self-signed certificate for a private key generated in a crypto chip slot.

During connecting IoT devices to an IoT platform we need to generate keys(private and public keys) and store them in our crypto chip. The private key will be used for generating digital signatures whenever we send data from our board and the public key will be used to authenticate the sender from its digital signature sent with the data.

Besides keys generated, we need to use JSON Web tokens for authenticating IoT devices to the cloud or to communicate with communication bridges such as MQTT. Figure 4.47 shows how IoT devices establish communication with MQTT bridges of GCP by using JSON Web Tokens.
JSON Web token is an open standard (RFC 7519) that is used to securely transmit information. The information contains a digitally signed signature of the sender and this makes the data to be trusted after verification is made at the receiver side.

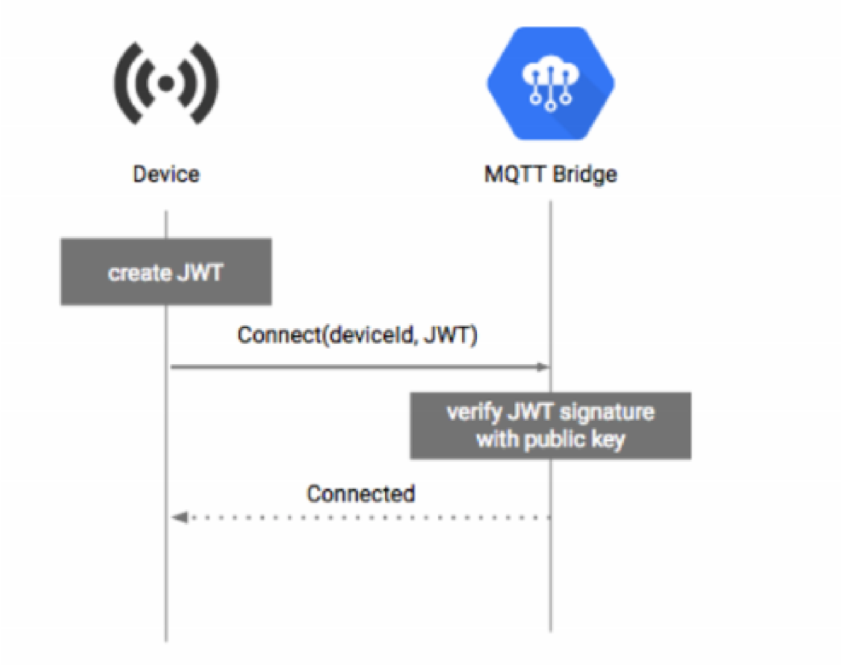


Figure 4.47: Device connection with MQTT bridge using JWT (source:cloud.google.com)

The data we send from our device are battery level, location information with longitude, and latitude. These data are not sensitive data, as a group we aren't afraid of the third party can steal the data. What we worry is that if an attacker can pretend as if it is one of the IoT devices and connects with the IoT platform. This has a dangerous consequence in our entire system because the IoT platform has a connection with the Database and web server for the Mobile app. If the hackers manage to connect with our Cloud platform they can manage a DoS(Denial of Service) attack and this can affect the TRYE AS company economically and they can lose users' trust. Losing trust from customers due to a lack of security can cause brand damage.

Since we are using the Google Cloud Platform, which has good security features and it has many security protocols in different levels of their architecture. The google cloud platform has strong authentication of IoT devices during communication and sending data. The private key of our device is stored in the crypto chip and is isolated from people and software, this makes the communication more secure and difficult for intruders. The group decided to use the existing GCP security features without adding any self-made security measures to the hardware part.

4.12.7 Controlling the device from GCP

For controlling the IoT device from the GCP we need to send messages to tell the device what to do. From GCP there are 2 types of messages that can be sent to control IoT devices that are registered on the device manager. The cloud platform sends the messages with the help of IoT core Admin SDK or Cloud REST API. The message types are Configuration and command message.

Configuration message:this is a message sent by the cloud platform through the cloud IoT core for configuring a device. It is a user-defined data. The data can be structured and also can be formatted in any format such as arbitrary binary data, text, JSON, or serialized protocol buffers. Configuration messages can be used to update firmware, reboot a device, turn on a feature, or change other properties. Devices using MQTT can subscribe to a special MQTT topic: /device/device-id/config for configuration updates. Devices can receive the latest configuration in a message payload. For example a configuration message can be ”bike-id”: ”001”, “power” : ”ON” or ”bike-id” : ”001”, “power” : ”OFF” . To verify the configuration message is correctly applied and the devices are in the correct state, each device can report its state whether it is ON or OFF.

Command message: are commands that can be sent to the IoT devices and they are one-time directive sent to those devices registered to the command topic: /devices/device-Id/commands/”. Command messages are much faster than configuration messages and can be sent more frequently. If the device is not connected to the cloud when the command message is sent, the command message will be lost.
There are different ways of sending these messages to IoT devices. For our project since we need to send different commands based on different scenarios or events, we will use Google cloud function to send the messages to our IoT device(MKR NB 1500). Google cloud functions are serverless computing platforms that can be used to execute code in the cloud. Related to cloud functions, we have two concepts called ***Events*** and ***Triggers***. When a change in the state of something happens the Google infrastructure will raise an event and triggers will be used to connect raised events with cloud functions.

In our project we have different events that may occur when users use the TRYE app for renting or booking a bike. For renting a bike a user should register on a user app and fill the necessary fields before sending it to the server. The result of user authentication on the server-side may cause an event where either a user is allowed to rent a bike or not. The events will trigger their own google cloud function which is responsible to run a JSON code for sending a configuration message to a bike for turning ON or OFF the motor. Since our project has different events that may need to send command or configuration messages to a bike. We need to use a function that triggers based on an event and such function is called background function.

We use background functions when we want to have our Cloud Function invoked indirectly in response to an event, such as a message on a Pub/Sub topic, a change in a Cloud Storage bucket, or a Firebase event [14].

Using background function we can send configuration messages to the pub/subtopic based on an event and the device can receive the message. The configuration message received by the device can be used to control the device and the subsystems that are interfaced with the device. For example, sending an OFF/ON message to the device can be used to control a state of a relay that is connected to the digital pin of our board and this can be used further to control the Shimano motor which is explained in section 4.12.2.

4.12.8 Combining all the Arduino Code

During the development of the system, multiple developers have worked on the code independently, when combining the code it's therefore important to arrange the code in the correct order. For our system, we will have to gather data then send it, if it happens in the other direction it can mean the data being sent is old, and therefore, less reliable. The exact order in which we run the different programs can be seen in figure 4.48

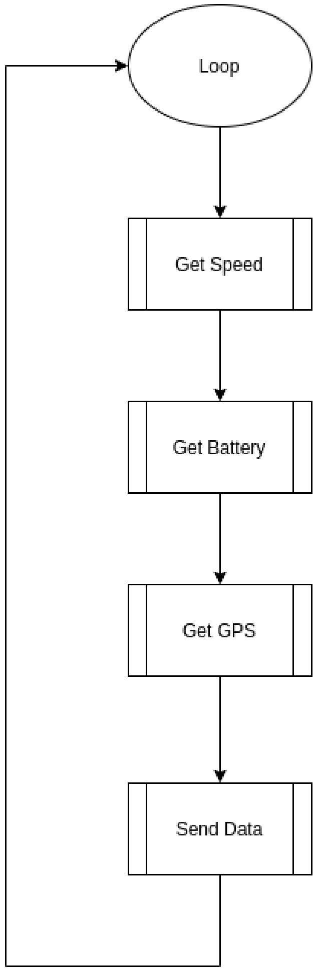


Figure 4.48: The order the code was arranged

4.12.9 Choosing IoT servers

There are many different IoT servers out on the market, some of the biggest might be Microsoft Azure, Amazon Web Server (AWS) and Google Cloud Platform (GCP) and there are smaller more specialized servers such as Thingspeak. But as the platform was chosen to be able to receive IoT data, store the data, and also store the data for the mobile application, the more specialized servers were not an option. As the cost during development should be at a minimum, AWS could not be used either, as both GCP and Microsoft Azure provide a free sample time. (note: normally AWS also provide a free sample time but because of extra demand on the servers during the coronavirus, they had to cut the free sample) That left Microsoft Azure and GCP, from which we choose GCP for further development.

4.12.10 Setting up IoT Server

A common standard for communication with IoT devices is the MQTT protocol. The MQTT protocol is a lightweight, publish-subscribe network protocol which is designed with a small footprint in mind[[17]]. This means we need to use the GCP service IoT core to receive and transmit the MQTT messages between the Arduino and the GCP platform. And because the MQTT protocol is a publish-subscribe based network protocol the GCP pub/sub service also needs to be used to handle the communication.

Setting up IoT core To connect the Arduino to the IoT core, the IoT core first has to be configured, this requires a collection (or register) name, a device name and a public key generated by the encryption chip built into the Arduino. The configuration of the Arduino can be read about in section 4.12.5 and in section 4.12.4

Setting up pub/sub To be able to send and receive messages to the right device, the GCP server needs a pub/sub service. To configure this, each bike will require a topic-id which works like the name for the bike, the topic-id is used to make sure the IoT core sends the messages to the correct bike

4.12.11 Simulating MQTTdata with python

To reduce the cost of development by reducing the bandwidth used by the Arduino, it is possible to use Python to simulate the functionality of the Arduino by sending data with the MQTT protocol. This will have the added benefit of being able to send a large amount of random data, which will simulate multiple devices connected at once. This can in turn be used as a tool for debugging and testing the capabilities of the server. The python script also helped to speed up the development of the system, as testing the GCP server was indented of the state of the Arduino

4.12.12 Sending IoT data to database

When the data has been uploaded to the GCP platform through the IoT core and pub/sub service it needs to be stored in a database for easy retrieval for the mobile application. GCP provides many different ways of storing data in a database, some of which are the SQL service, BigQuery, and Firestore, where we choose to use the Firestore service.

SQL service: might be the easiest way of storing the data in a normal SQL database in GCP. To upload the IoT data to the SQL service, GCP first needs to pull the data from the pub/sub service, to do so it can use the cloud function service. Cloud function works as a scripting environment that can use python, node.js, or GO to control various parts of the GCP's services. The cloud function has the functionality to trigger if there is an event in the pub/sub service, by using this the script will run when new data is available. The problem with using cloud function with the SQL service is it only recently got supported to communicate between them, and therefore essential documentation is missing.

BigQuery: is a NoSQL database made to handle big data, with this service it can be combined with dataflow to automatically upload data from the pub/sub service to the BigQuery service with a job. A job is something that is programmed the same way as a normal SQL call, but instead of selecting from a database it selects from a JSON file uploaded to the pub/sub service, this makes it easy to configure. But as the services are made with a large amount of data in mind is reflected in the price, this means to keep the service running for one month would cost approx. 3000 NOK, this makes it not a viable solution

Firestore: is a NoSQL database made to store data for mobile applications and is a sub-service of what used to be Firebase. Cloud function also used to be part of the Firebase framework and therefore the interaction between these services is well documented, which was the main problem with the SQL service. How the interaction works between cloud function and pub/sub can be read about in the SQL service. And the source code used to pull the data from the pub/sub service and push it to the Firestore service can be seen in appendix A.11.8. Firestore stores data in what's called collection and documents, where one collection can store multiple documents and one document can refer to multiple out collection, this makes a tree that will start with a collection followed by a document followed by a new collection and so on. For our system, we need a collection called "TRYE-bike" with a document for each bike called "bike-xx". Here the collection can be used as a location such as Kongsberg, Drammen, Oslo if the system is expanded in the future. The way data is uploaded to the Firestore database now is it overwrites the last position, this makes sure each bike only uses one database entry which will save space in the database but the historical data will be lost, an alternative is to append the data at the end of the document. This will save the historical data at the cost of taking more space.

4.13 Summary of the Hardware

The hardware system is close to being done, so far the system can calculate the speed, get GPS coordinates, get battery level, store the data in the cloud, send data to the mobile app, and receive data from the mobile app. The only thing missing is getting all the system to work together.

Chapter 5

Conclusion of our work

5.0.1 Summary

Throughout the last semester, we have been through a project filled with challenges. Both the normal challenges a bachelor group meet, but also the COVID-19 pandemic with its restrictions.

COVID-19 forced us to have to work from home. This happened right in front of the second bachelors-presentation and caused us to continue working from home throughout the rest of the project. To handle the challenges of COVID-19, we had to make use of tools to ease our communication. Examples of tools we used throughout the project to continue the development as usual was: Discord for stand-up meetings, Zoom or Google Hangouts for communicating with our supervisor (Or anyone else we needed to talk to), Google Drive for documenting as a group, Skype For Business for having digital presentations, and WhatsApp for keeping in touch with our employer and group members.

Our project was split into two main systems, the software- and the hardware-system. Work was done in respect to the derived requirements given from Trye, and the group made different User Stories from the requirements. The User Stories themselves were stored in Trello, which is a Kanban-style list-making application that proved to be useful in our agile development. Trello did also enable us to visualize progress, with its ”To do”, ”In Progress” and ”Done” list.

To be able to measure progress we had to document every User Story, and on every Friday evening we had a Sprint-review where we went through the progress made that week as a group. To make it easier for us to visualize the progress, a burndown-chart was created to show us as a group how many points we achieved throughout the working week.

5.0.2 Reflection

In retrospect, we should have been more structured when it came to filling out Sprint Reviews, and verify progress.

With not understanding the importance of documenting Sprint Reviews and Verification documents, it became difficult to create genuine burndown-charts and conclude a User Story. This lead to us not knowing how effective the group was. By not knowing exactly how much we were able to produce, planning became less effective in periods. However, when we as a group understood the importance of Sprint Reviews and Verification, planning a work week became easier and estimations became more accurate.

In addition, the group faced a challenge with verifying technical work as we no longer had access to the bike after the pandemic lockdown. Cooperation between the group members went well throughout the project, regardless of the situation we were put in.

The University, staff members and supervisors continued give us the necessary support and guidance for the duration of the project. They also provided us with the tools and necessary information to ensure us that everything would go as planned.

5.0.3 Future Work

Before the system can be finished, some parts are required to complete the system, these include a bike and a company Google account with billing enabled. The bike is required to achieve the work of verification and solve any unexpected faults associated with the connection of our system to the existing system. The company Google account is needed to give the ownership of the entire developed product to TRYE.

Code needs to be combined on the Arduino

Some of the Arduino code still needs to be combined, this includes the code for retrieving the GPS coordinates, connecting the system to the internet, and sending the data to GCP.

Preferably a configuration panel should also be included for a user-friendly configuration for the different bikes. The configuration could include, bike wheel diameter, units (metric/imperial), battery capacity, ID, name, and what data should be sent to the server.

The hardware has to be connected to the bike

Since our sub-system only has been tested independently, unexpected situations could occur when the sub-systems are connected to the existing systems. Therefore the system can not be considered done, before this is verified.

Arduino to GCP communication needs to be formated

The data sent from the Arduino to the GCP needs to be formatted as JSON data as that is required by the GCP sub-service cloud function. This can be done on the Arduino by editing the message sent.

Develop security features in the mobile app

As the payment system demands a safe and secure way of communicating, security methods need to be implemented to move from testing mode to production mode.

Testing the usability of the app

Before launching the app to the user, we should perform a user study to ensure a clear and easy to use interface. This can be done by letting people unfamiliar with the interface try the app, and observing their interaction.

Make a guide to add functionality to the app for TRYE

As TRYE wants to add more bikes to the system at a later time, a good step by step guide should be written to make this process as easy to understand as possible.

6 References

[1] *”Guide to NEO-6M GPS Module with Arduino (I only got inspiration)”.* URL: https://randomnerdtutorials.com/guide-to-neo-6m-gps-module-with-arduino/. (accessed: 21.03.2020).

[2] *”How to analyze risk ?”* URL: https://www.projectmanager.com/training/how-to-analyze-risks-project%20(I%20got%20only%20inspiration). (accessed: 13.03.2020).

[3] *”How to interface GPS module with Arduino (I only got inspiration)”.* URL: https://create.arduino.cc/projecthub/ruchir1674/how-to-interface-gps-module-neo-6m-with-arduino-8f90ad. (accessed: 20.04.2020).

[4] *”Technical risk management (I only got inspiration)”.* URL: https://ieeexplore.ieee.org/abstract/document/4349543. (accessed: 12.03.2020).

[5] *About Lime e-scooters.* URL: https://www.crunchbase.com/organization/limebike#section-overview. (accessed: 11.05.2020).

[6] *About Universitetet i Sørøst-Norge.* URL: http://www.usn.no/om-usn. (accessed: 07.05.2020).

[7] *Accepting a card payment.* URL: https://stripe.com/docs/payments/accept-a-payment. (accessed: 22.05.2020).

[8] *Arduino MKR NB 1500.* URL: https://store.arduino.cc/arduino-mkr-nb-1500-1413. (accessed: 05.02.2020).

[9] *Business Insider projection for the rental market of electrical scooters and bikes.* URL: https://markets.businessinsider.com/news/stocks/the-bike-and-scooter-rental-market-is-projected-to-grow-from-usd-2-5-billion-in-2019-to-reach-usd-10-1-billion-by-2027-at-a-cagr-of-18-9-1028677949. (accessed: 22.05.2020).

[10] *Business insider projection for the rental market of electrical scooters and bikes.* URL: https://www.coruscatesolution.com/e-scooter-hardware-solutions-for-rental-business/. (accessed: 22.05.2020).

[11] *Discord.* URL: https://discord.com/. (accessed: 13.05.2020).

[12] *E-Scooter hacks.* URL: https://mashable.com/article/e-scooter-hacks-bird-lime/. (accessed: 11.05.2020).

[13] *Electric scooters' sudden invasion of American cities, explained.* URL: https://www.vox.com/2018/8/27/17676670/electric-scooter-rental-bird-lime-skip-spin-cities. (accessed: 22.05.2020).

[14] *Google cloud function.* URL: https://cloud.google.com/functions/docs/writing/background. (accessed: 05.02.2020).

[15] *How to do a SWOT Analysis for better strategic planning(I only got inspiration).* URL: `https://articles.bplans.com/how-to-perform-swot-analysis/`. (accessed: 2.02.2020).

[16] *Microchip-ATECC508A.* URL: `http://ww1.microchip.com/downloads/en/DeviceDoc/20005928A.pdf`. (accessed: 29.03.2020).

[17] *MQTT Version 5.0 OASIS Standard Specification.* URL: `https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.pdf`. (accessed: 22.05.2020).

[18] *Neo 6M GPS module.* URL: `https://robu.in/product/ublox-neo-6m-gps-module/`. (accessed: 29.03.2020).

[19] *Risk definition.* URL: `https://pm4id.org/chapter/11-1-defining-risk/`. (accessed: 22.02.2020).

[20] *Risk register definition.* URL: `https://www.sciencedirect.com/topics/engineering/risk-register`. (accessed: 10.02.2020).

[21] *VOI in oslo.* URL: `https://www.elbil24.no/nyheter/slik-leier-du-el-sparkesykkel-i-oslo/70869996`. (accessed: 11.05.2020).

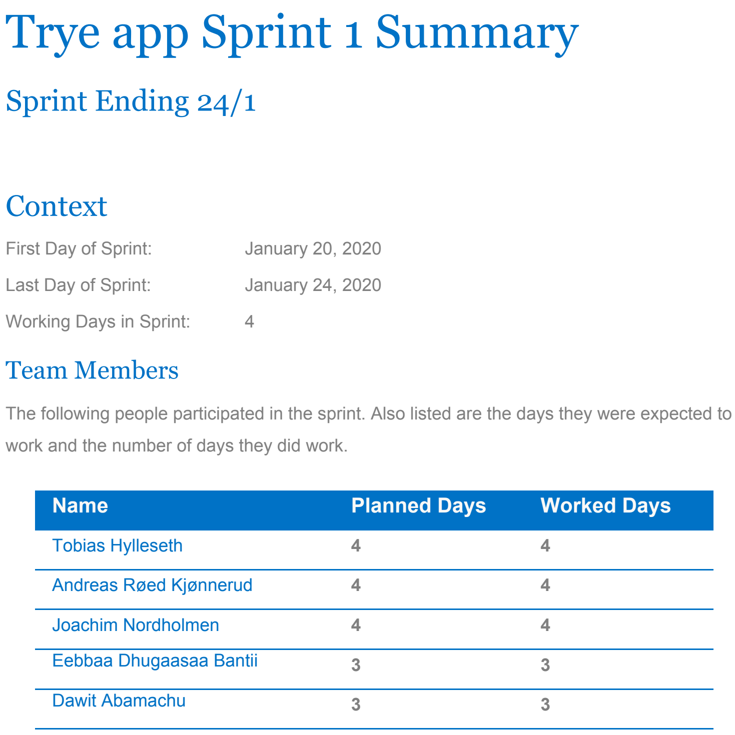
[22] *What is a Product Owner?* URL: `https://www.scrum.org/resources/what-is-a-product-owner`. (accessed: 13.05.2020).

Chapter A

Appendix

A.1 Sprint reports

A.1.1 Sprint 1



Trye app Sprint 1 Summary

Sprint Ending 24/1

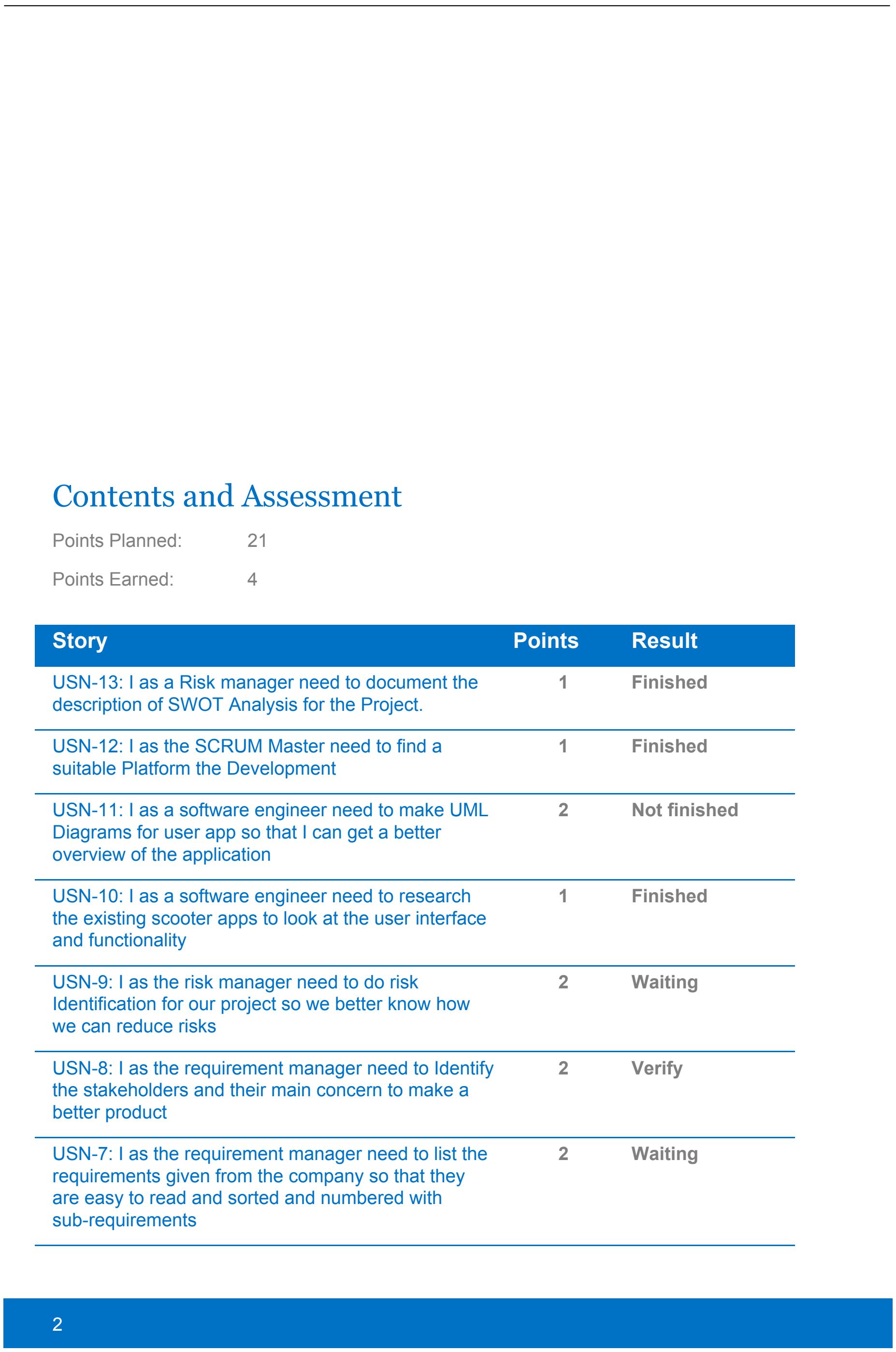
Context

First Day of Sprint:	January 20, 2020
Last Day of Sprint:	January 24, 2020
Working Days in Sprint:	4

Team Members

The following people participated in the sprint. Also listed are the days they were expected to work and the number of days they did work.

Name	Planned Days	Worked Days
Tobias Hylleseth	4	4
Andreas Røed Kjønnerud	4	4
Joachim Nordholmen	4	4
Eebbaa Dhugaasaa Bantii	3	3
Dawit Abamachu	3	3

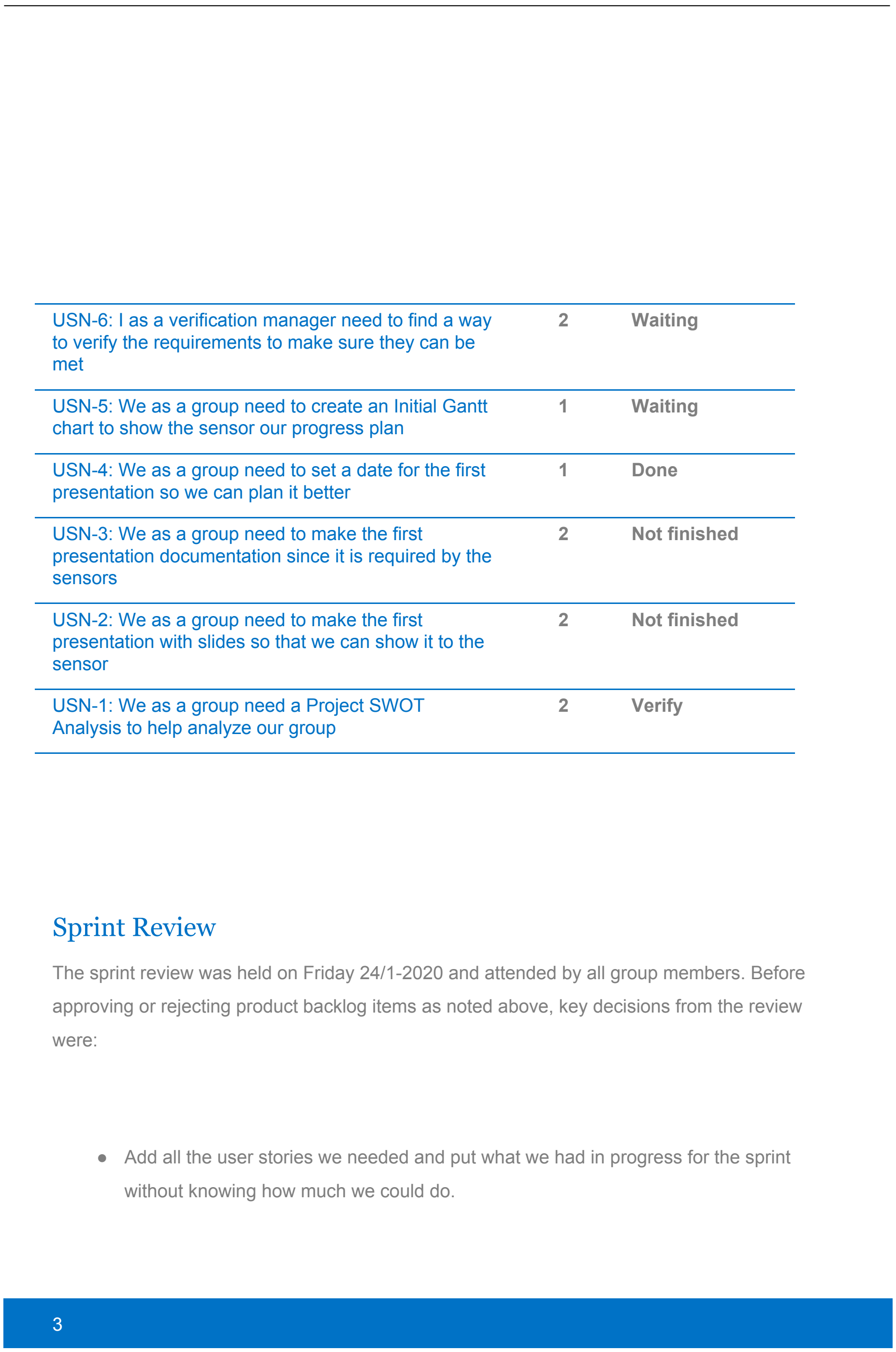


Contents and Assessment

Points Planned: 21

Points Earned: 4

Story	Points	Result
USN-13: I as a Risk manager need to document the description of SWOT Analysis for the Project.	1	Finished
USN-12: I as the SCRUM Master need to find a suitable Platform the Development	1	Finished
USN-11: I as a software engineer need to make UML Diagrams for user app so that I can get a better overview of the application	2	Not finished
USN-10: I as a software engineer need to research the existing scooter apps to look at the user interface and functionality	1	Finished
USN-9: I as the risk manager need to do risk Identification for our project so we better know how we can reduce risks	2	Waiting
USN-8: I as the requirement manager need to Identify the stakeholders and their main concern to make a better product	2	Verify
USN-7: I as the requirement manager need to list the requirements given from the company so that they are easy to read and sorted and numbered with sub-requirements	2	Waiting

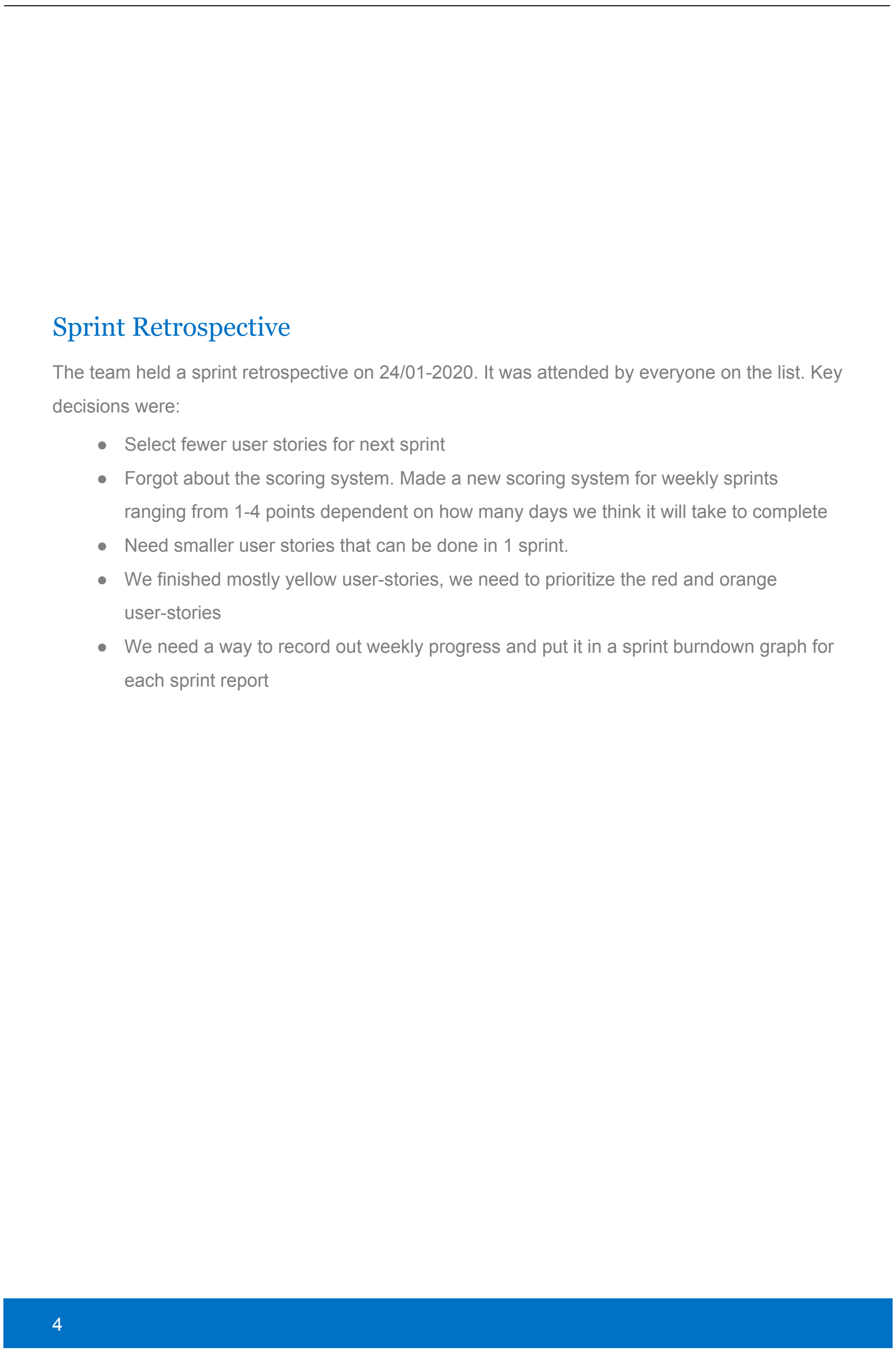


USN-6: I as a verification manager need to find a way to verify the requirements to make sure they can be met	2	Waiting
USN-5: We as a group need to create an Initial Gantt chart to show the sensor our progress plan	1	Waiting
USN-4: We as a group need to set a date for the first presentation so we can plan it better	1	Done
USN-3: We as a group need to make the first presentation documentation since it is required by the sensors	2	Not finished
USN-2: We as a group need to make the first presentation with slides so that we can show it to the sensor	2	Not finished
USN-1: We as a group need a Project SWOT Analysis to help analyze our group	2	Verify

Sprint Review

The sprint review was held on Friday 24/1-2020 and attended by all group members. Before approving or rejecting product backlog items as noted above, key decisions from the review were:

- Add all the user stories we needed and put what we had in progress for the sprint without knowing how much we could do.

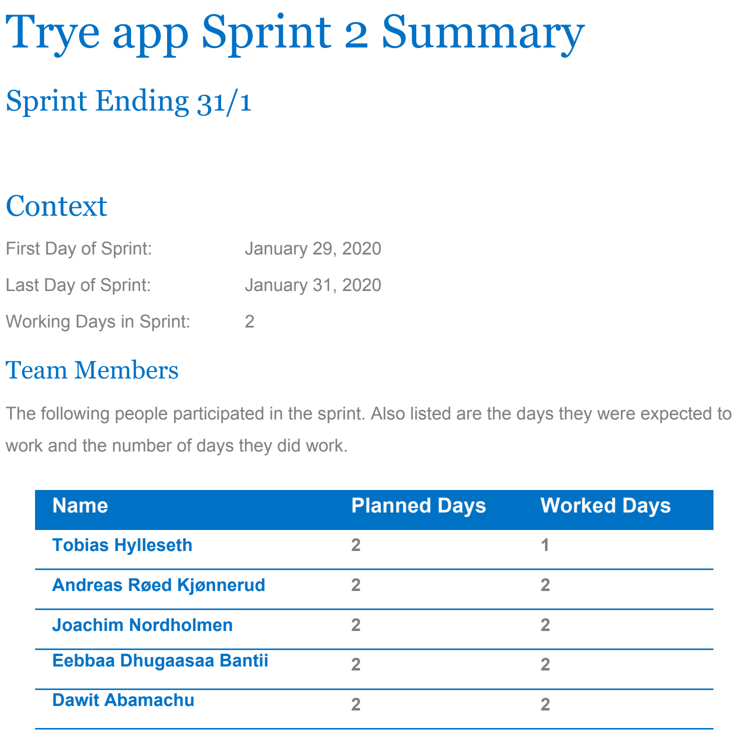


Sprint Retrospective

The team held a sprint retrospective on 24/01-2020. It was attended by everyone on the list. Key decisions were:

- Select fewer user stories for next sprint
- Forgot about the scoring system. Made a new scoring system for weekly sprints ranging from 1-4 points dependent on how many days we think it will take to complete
- Need smaller user stories that can be done in 1 sprint.
- We finished mostly yellow user-stories, we need to prioritize the red and orange user-stories
- We need a way to record out weekly progress and put it in a sprint burndown graph for each sprint report

A.1.2 Sprint 2



Trye app Sprint 2 Summary

Sprint Ending 31/1

Context

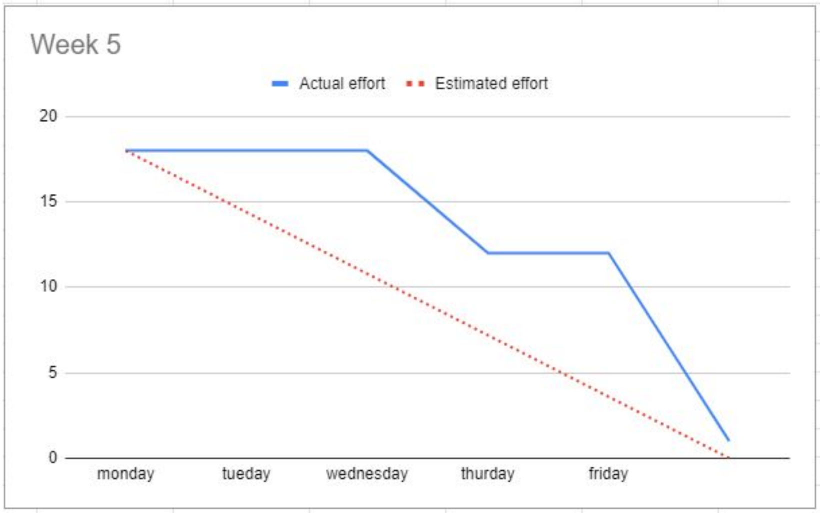
First Day of Sprint:	January 29, 2020
Last Day of Sprint:	January 31, 2020
Working Days in Sprint:	2

Team Members

The following people participated in the sprint. Also listed are the days they were expected to work and the number of days they did work.

Name	Planned Days	Worked Days
Tobias Hylleseth	2	1
Andreas Røed Kjønnerud	2	2
Joachim Nordholmen	2	2
Eebbaa Dhugaasaa Bantii	2	2
Dawit Abamachu	2	2

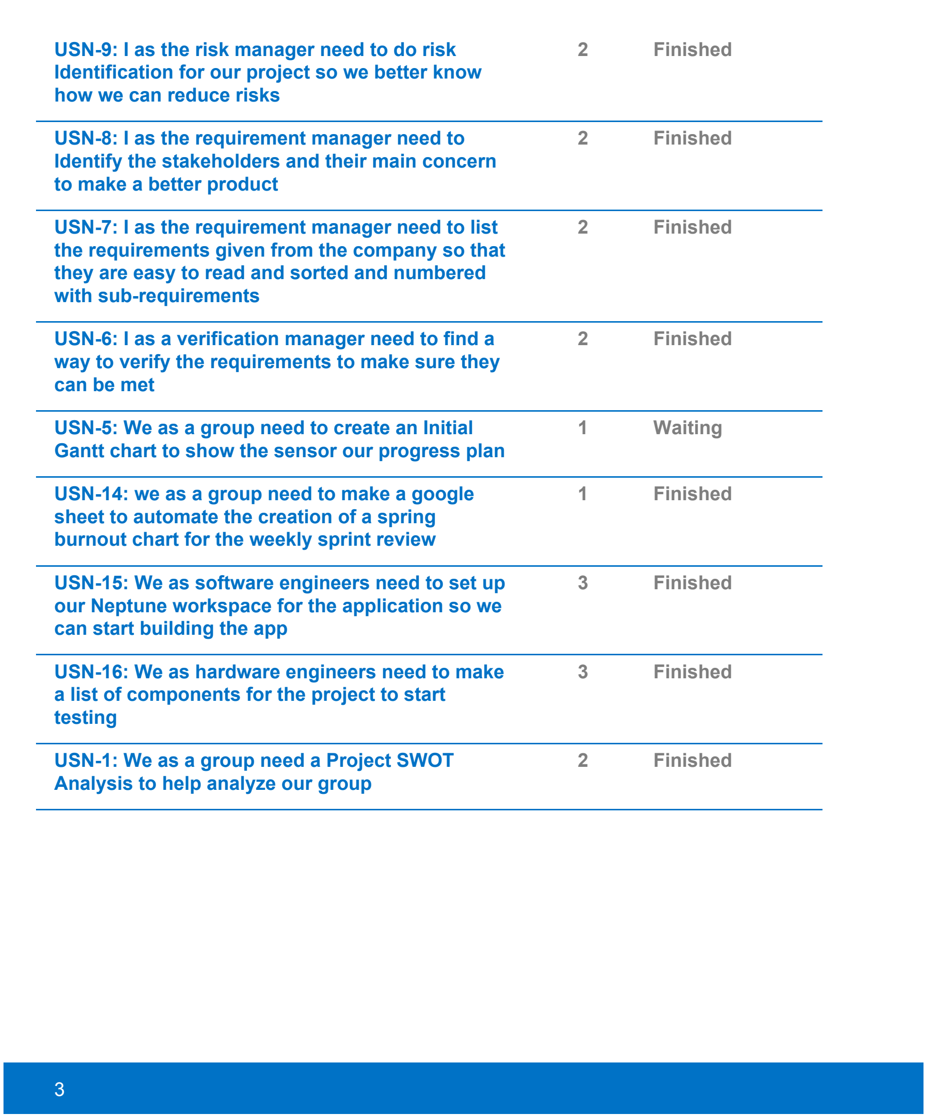
Burndown chart



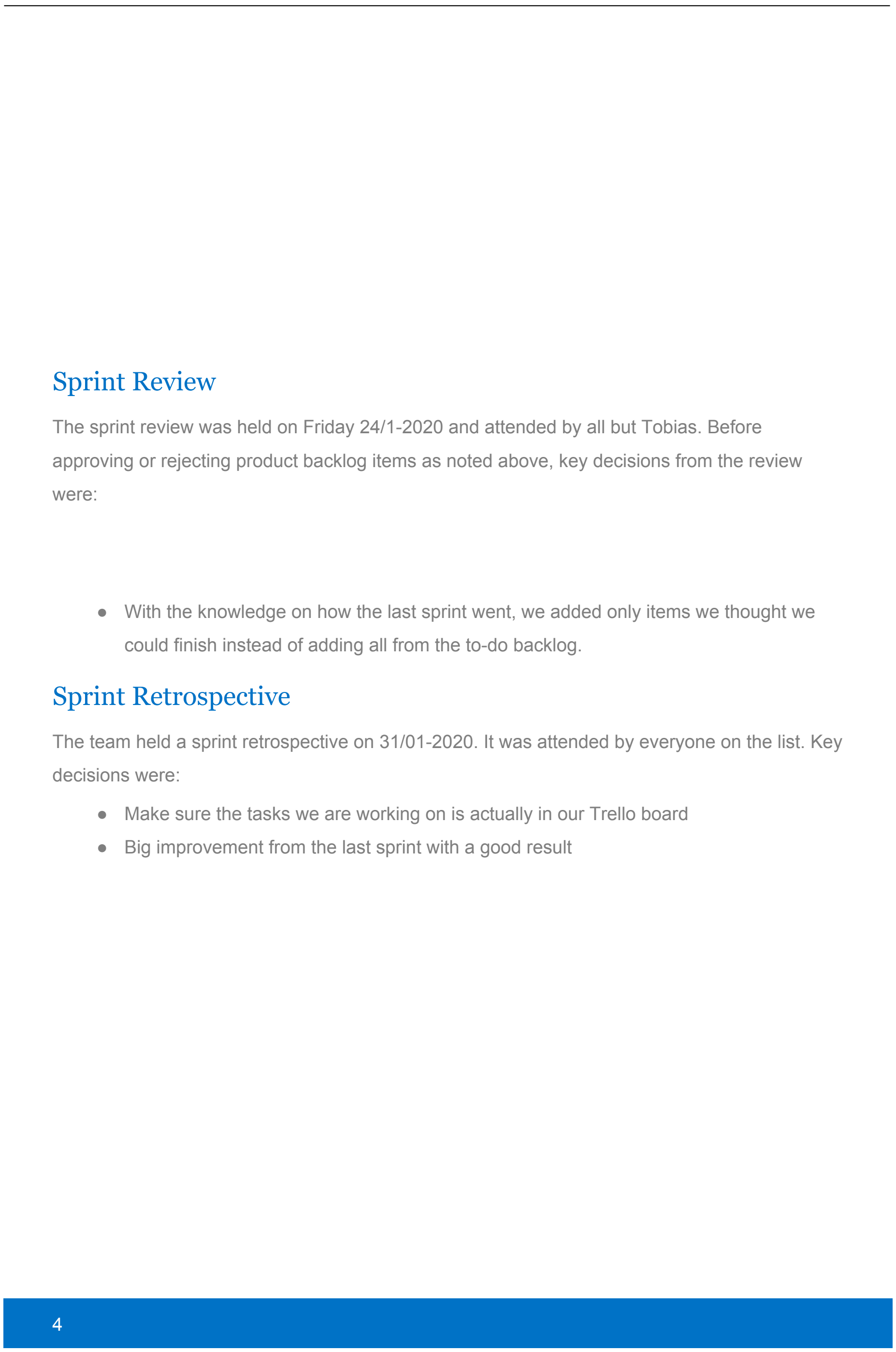
Contents and Assessment

Points Planned:	18
Points Earned:	14

Story	Points	Result



USN-9: I as the risk manager need to do risk Identification for our project so we better know how we can reduce risks	2	Finished
USN-8: I as the requirement manager need to Identify the stakeholders and their main concern to make a better product	2	Finished
USN-7: I as the requirement manager need to list the requirements given from the company so that they are easy to read and sorted and numbered with sub-requirements	2	Finished
USN-6: I as a verification manager need to find a way to verify the requirements to make sure they can be met	2	Finished
USN-5: We as a group need to create an Initial Gantt chart to show the sensor our progress plan	1	Waiting
USN-14: we as a group need to make a google sheet to automate the creation of a spring burnout chart for the weekly sprint review	1	Finished
USN-15: We as software engineers need to set up our Neptune workspace for the application so we can start building the app	3	Finished
USN-16: We as hardware engineers need to make a list of components for the project to start testing	3	Finished
USN-1: We as a group need a Project SWOT Analysis to help analyze our group	2	Finished



Sprint Review

The sprint review was held on Friday 24/1-2020 and attended by all but Tobias. Before approving or rejecting product backlog items as noted above, key decisions from the review were:

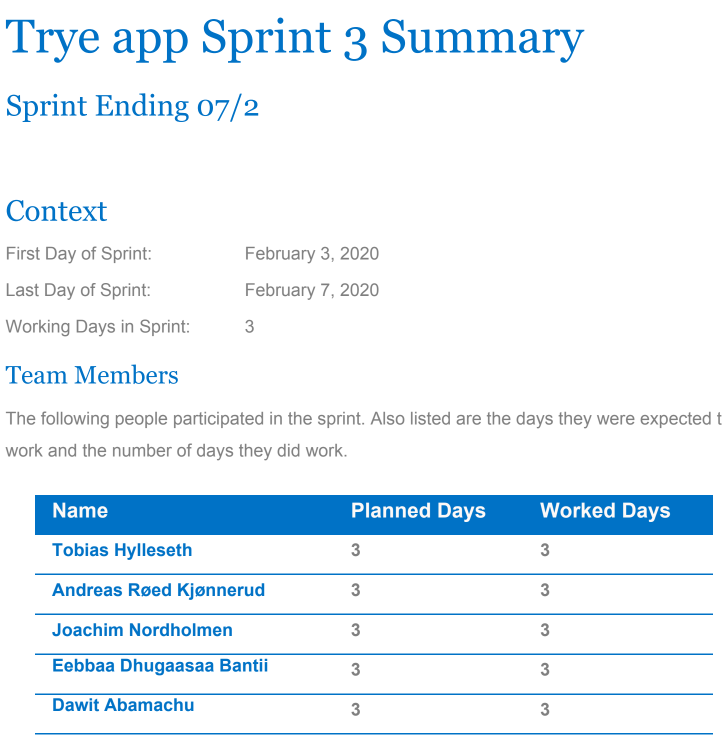
- With the knowledge on how the last sprint went, we added only items we thought we could finish instead of adding all from the to-do backlog.

Sprint Retrospective

The team held a sprint retrospective on 31/01-2020. It was attended by everyone on the list. Key decisions were:

- Make sure the tasks we are working on is actually in our Trello board
- Big improvement from the last sprint with a good result

A.1.3 Sprint 3



Trye app Sprint 3 Summary

Sprint Ending 07/2

Context

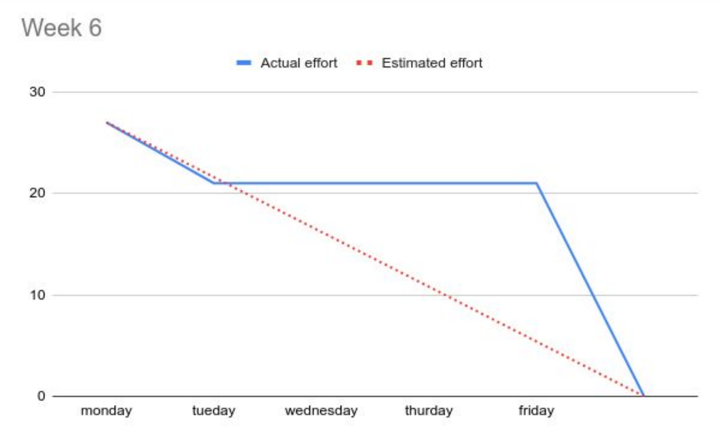
First Day of Sprint:	February 3, 2020
Last Day of Sprint:	February 7, 2020
Working Days in Sprint:	3

Team Members

The following people participated in the sprint. Also listed are the days they were expected to work and the number of days they did work.

Name	Planned Days	Worked Days
Tobias Hylleseth	3	3
Andreas Røed Kjønnerud	3	3
Joachim Nordholmen	3	3
Eebbaa Dhugaasaa Bantii	3	3
Dawit Abamachu	3	3

Burndown chart

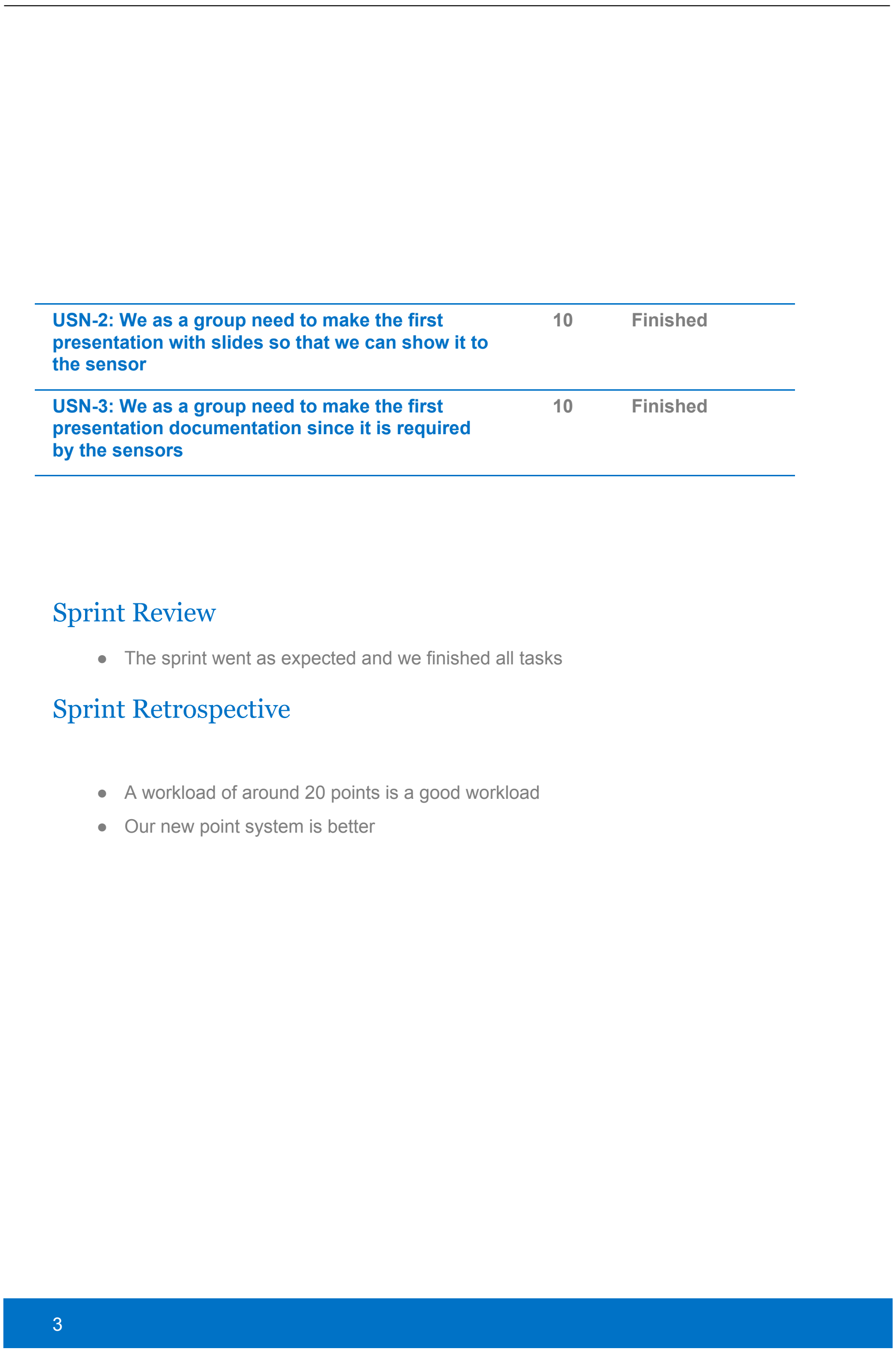


Contents and Assessment

Points Planned: 21

Points Earned: 21

Story	Points	Result
USN-5: We as a group need to create an Initial Gantt chart to show the sensor our progress plan	1	Finished



USN-2: We as a group need to make the first presentation with slides so that we can show it to the sensor	**10**	**Finished**
USN-3: We as a group need to make the first presentation documentation since it is required by the sensors	**10**	**Finished**

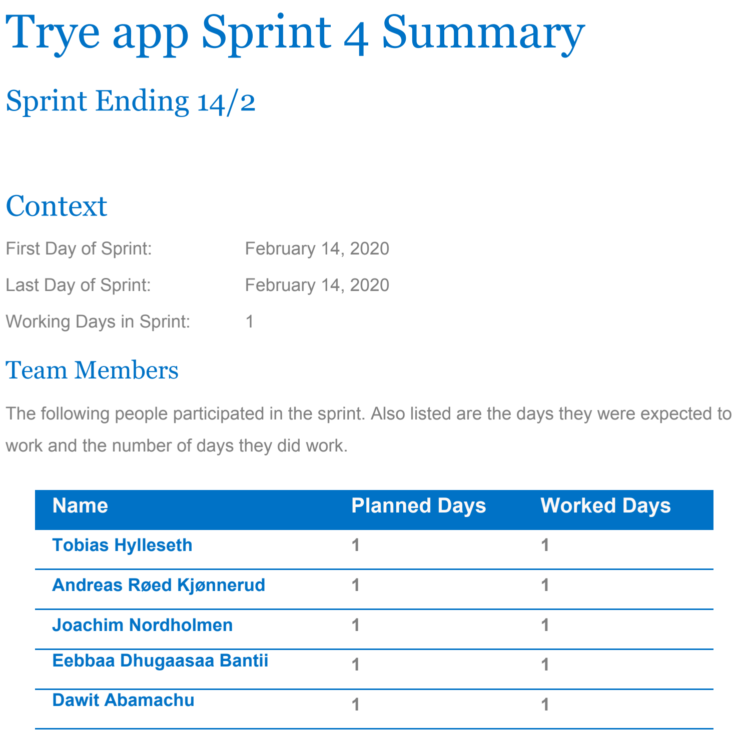
Sprint Review

- The sprint went as expected and we finished all tasks

Sprint Retrospective

- A workload of around 20 points is a good workload
- Our new point system is better

A.1.4 Sprint 4



Trye app Sprint 4 Summary

Sprint Ending 14/2

Context

First Day of Sprint: February 14, 2020

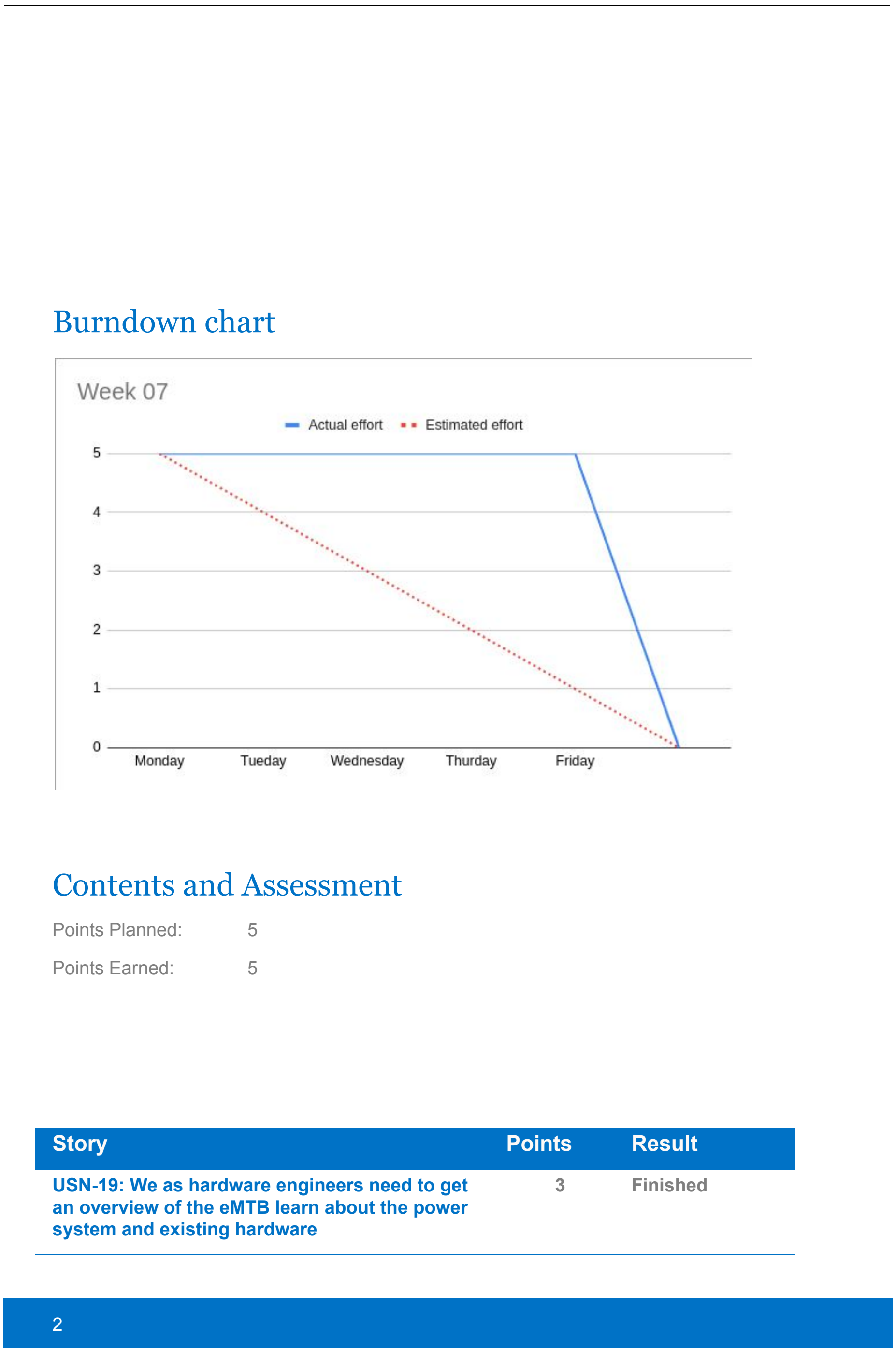
Last Day of Sprint: February 14, 2020

Working Days in Sprint: 1

Team Members

The following people participated in the sprint. Also listed are the days they were expected to work and the number of days they did work.

Name	Planned Days	Worked Days
Tobias Hylleseth	1	1
Andreas Røed Kjønnerud	1	1
Joachim Nordholmen	1	1
Eebbaa Dhugaasaa Bantii	1	1
Dawit Abamachu	1	1



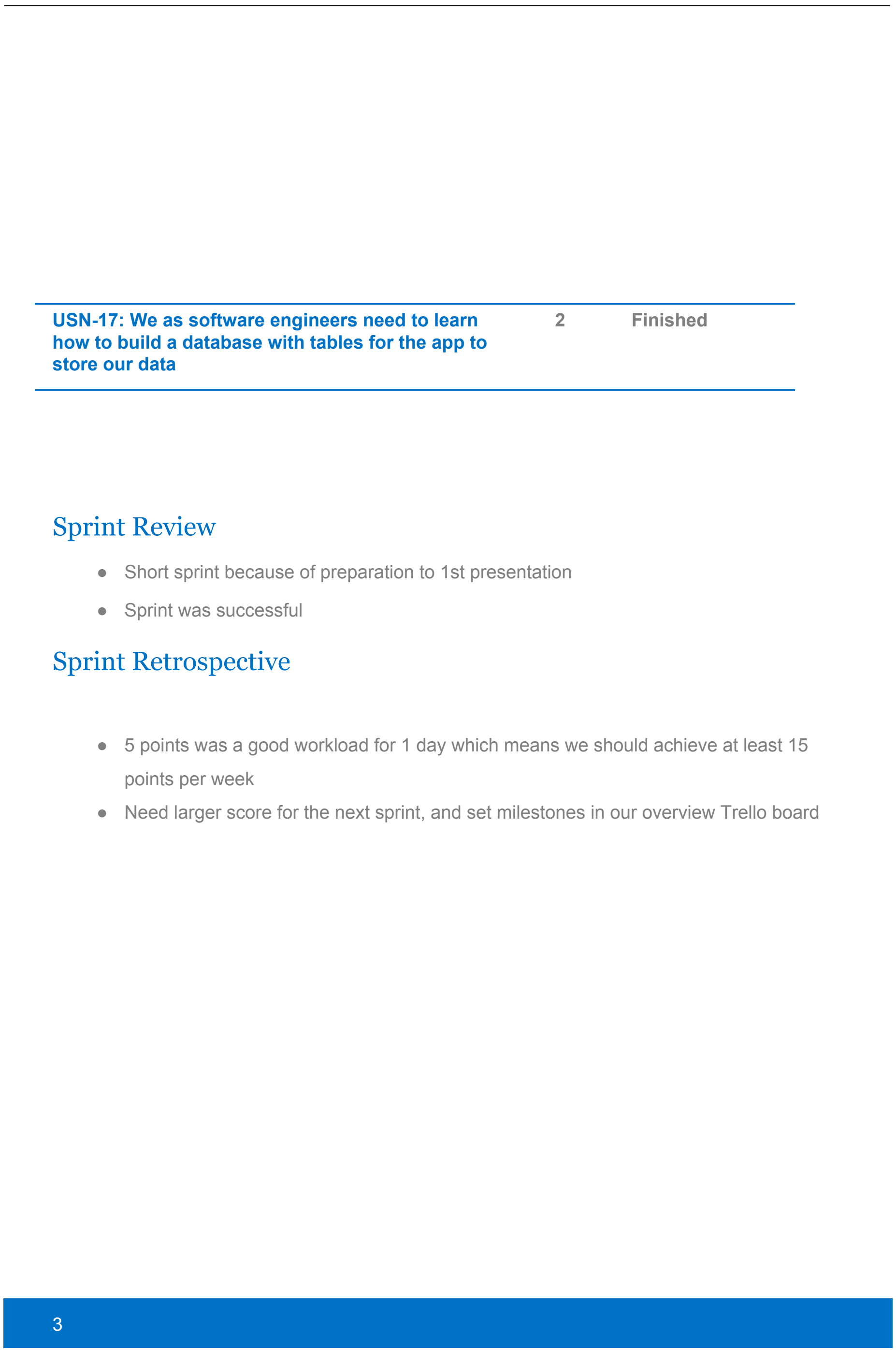
Burndown chart

Week 07
Actual effort
Estimated effort
5
4
3
2
1
0
Monday
Tueday
Wednesday
Thurday
Friday

Contents and Assessment

Points Planned:	5
Points Earned:	5

Story	Points	Result
USN-19: We as hardware engineers need to get an overview of the eMTB learn about the power system and existing hardware	3	Finished



USN-17: We as software engineers need to learn how to build a database with tables for the app to store our data	**2**	**Finished**

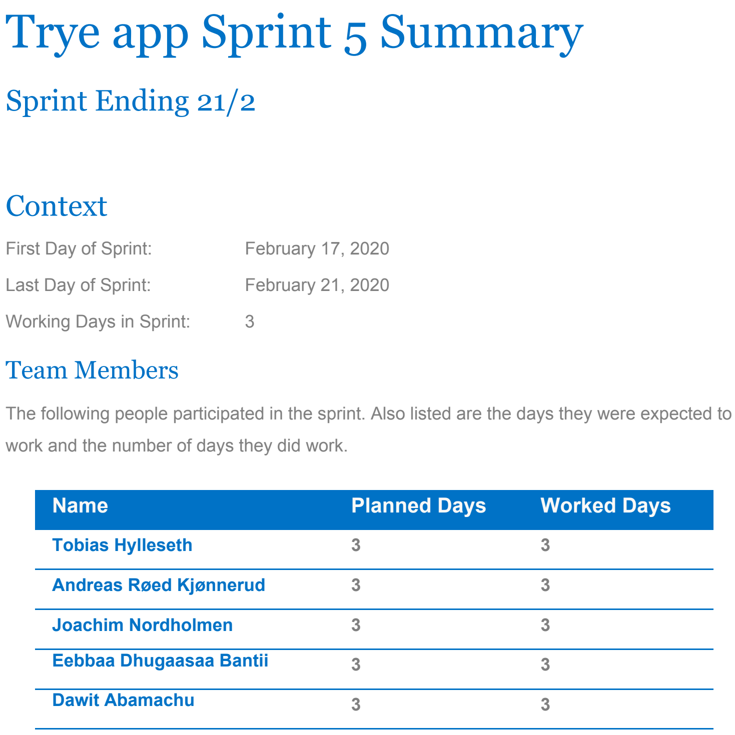
Sprint Review

- Short sprint because of preparation to 1st presentation
- Sprint was successful

Sprint Retrospective

- 5 points was a good workload for 1 day which means we should achieve at least 15 points per week
- Need larger score for the next sprint, and set milestones in our overview Trello board

A.1.5 Sprint 5



Trye app Sprint 5 Summary

Sprint Ending 21/2

Context

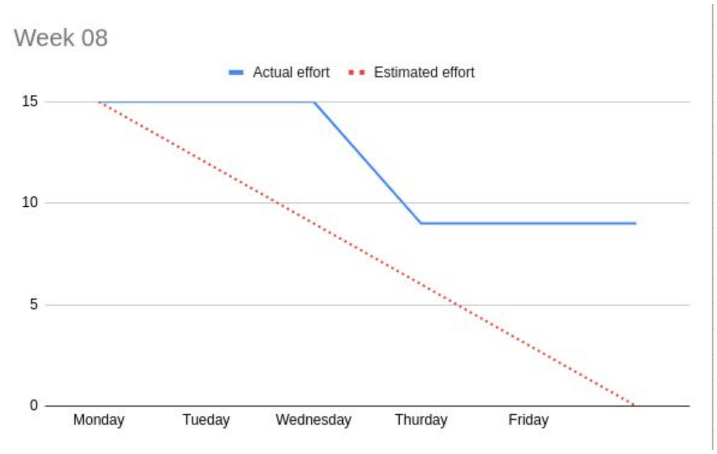
First Day of Sprint:	February 17, 2020
Last Day of Sprint:	February 21, 2020
Working Days in Sprint:	3

Team Members

The following people participated in the sprint. Also listed are the days they were expected to work and the number of days they did work.

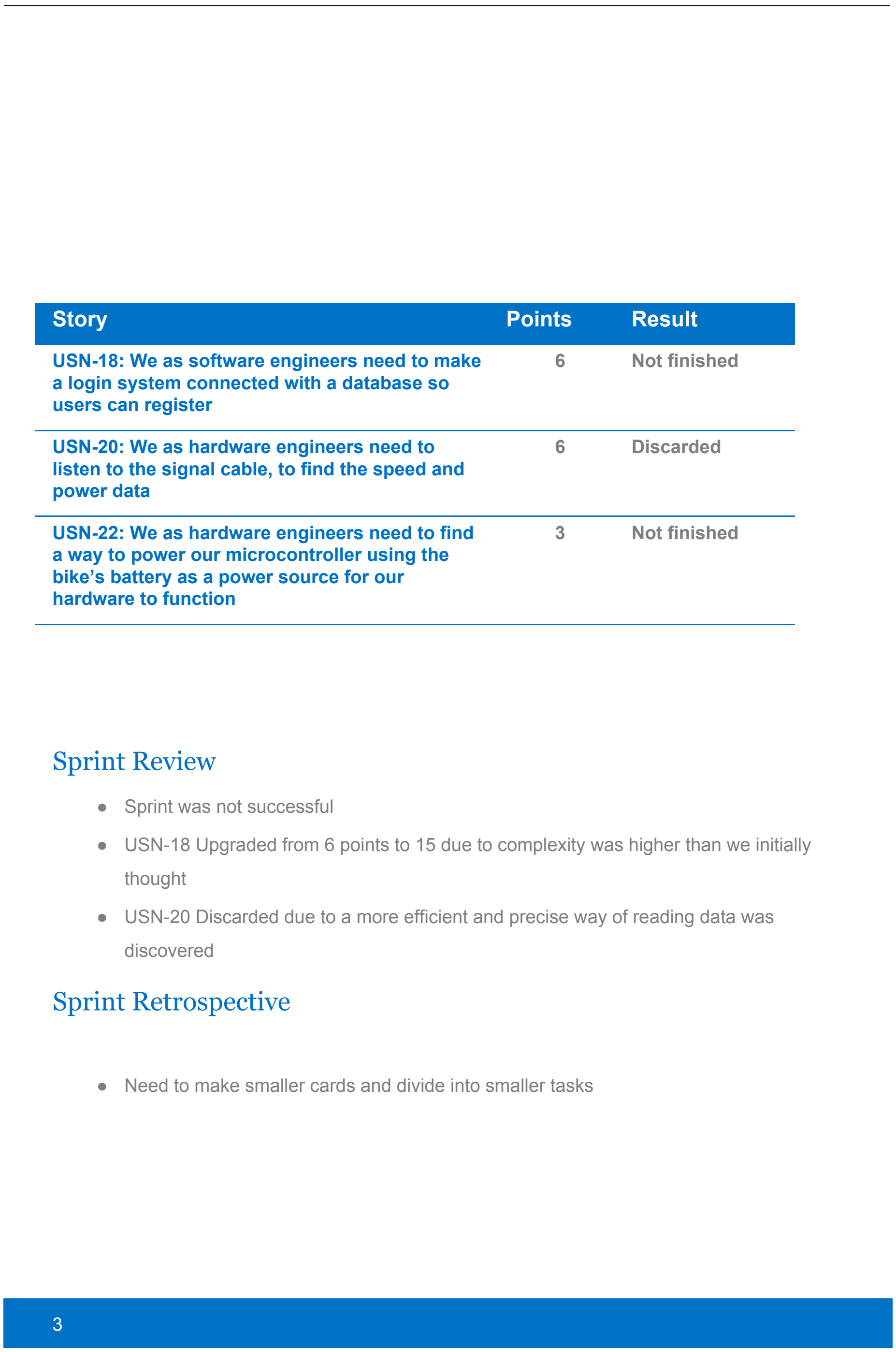
Name	Planned Days	Worked Days
Tobias Hylleseth	3	3
Andreas Røed Kjønnerud	3	3
Joachim Nordholmen	3	3
Eebbaa Dhugaasaa Bantii	3	3
Dawit Abamachu	3	3

Burndown chart



Contents and Assessment

Points Planned:	15
Points Earned:	0



Story	Points	Result
USN-18: We as software engineers need to make a login system connected with a database so users can register	**6**	**Not finished**
USN-20: We as hardware engineers need to listen to the signal cable, to find the speed and power data	**6**	**Discarded**
USN-22: We as hardware engineers need to find a way to power our microcontroller using the bike's battery as a power source for our hardware to function	**3**	**Not finished**

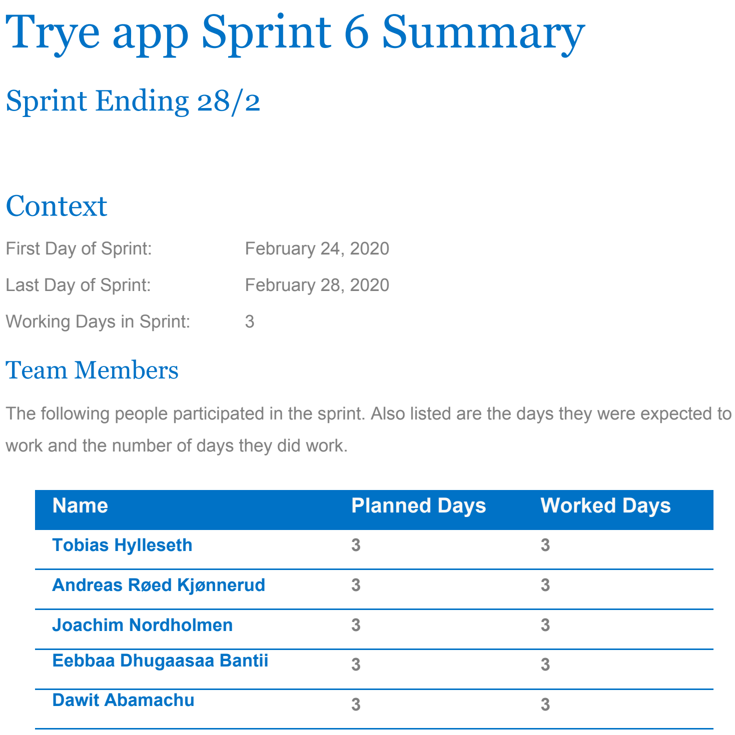
Sprint Review

- Sprint was not successful
- USN-18 Upgraded from 6 points to 15 due to complexity was higher than we initially thought
- USN-20 Discarded due to a more efficient and precise way of reading data was discovered

Sprint Retrospective

- Need to make smaller cards and divide into smaller tasks

A.1.6 Sprint 6



Trye app Sprint 6 Summary

Sprint Ending 28/2

Context

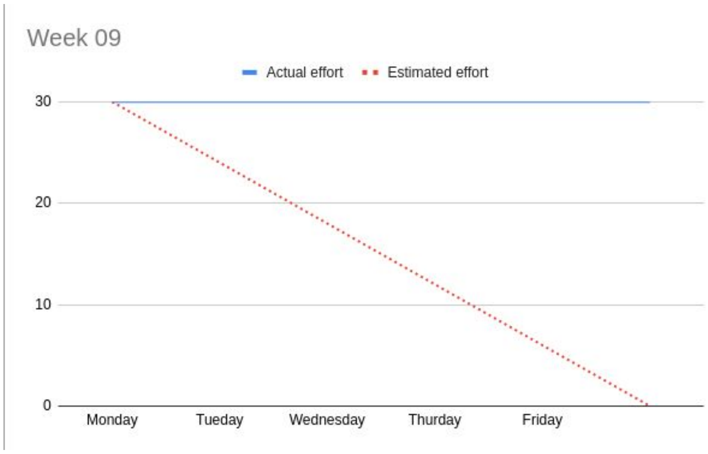
First Day of Sprint:	February 24, 2020
Last Day of Sprint:	February 28, 2020
Working Days in Sprint:	3

Team Members

The following people participated in the sprint. Also listed are the days they were expected to work and the number of days they did work.

Name	Planned Days	Worked Days
Tobias Hylleseth	3	3
Andreas Røed Kjønnerud	3	3
Joachim Nordholmen	3	3
Eebbaa Dhugaasaa Bantii	3	3
Dawit Abamachu	3	3

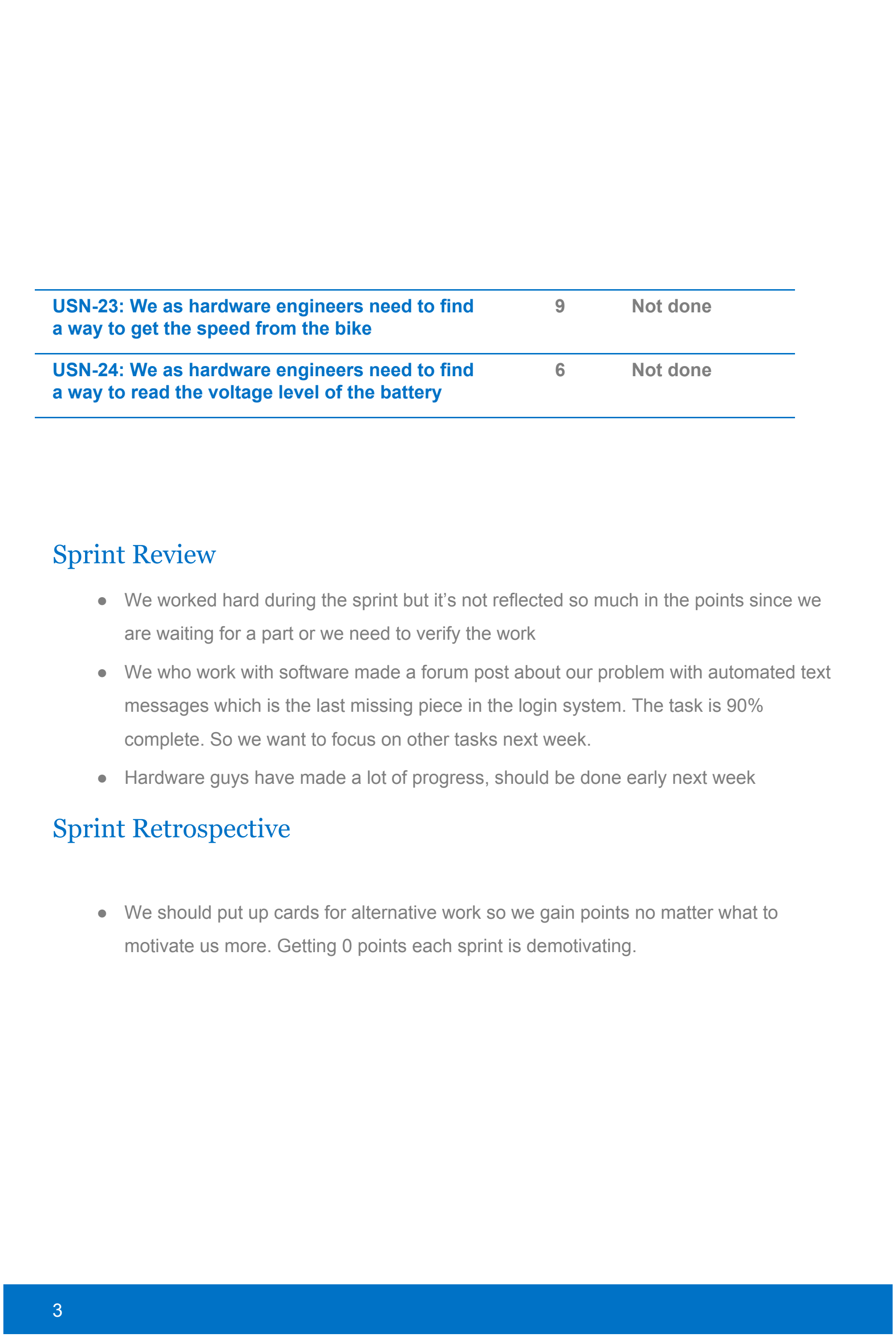
Burndown chart



Contents and Assessment

Points Planned:	30
Points Earned:	0

Story	Points	Result
USN-18: We as software engineers need to make a login system connected with a database so users can register	15	Waiting



USN-23: We as hardware engineers need to find a way to get the speed from the bike	**9**	**Not done**
USN-24: We as hardware engineers need to find a way to read the voltage level of the battery	**6**	**Not done**

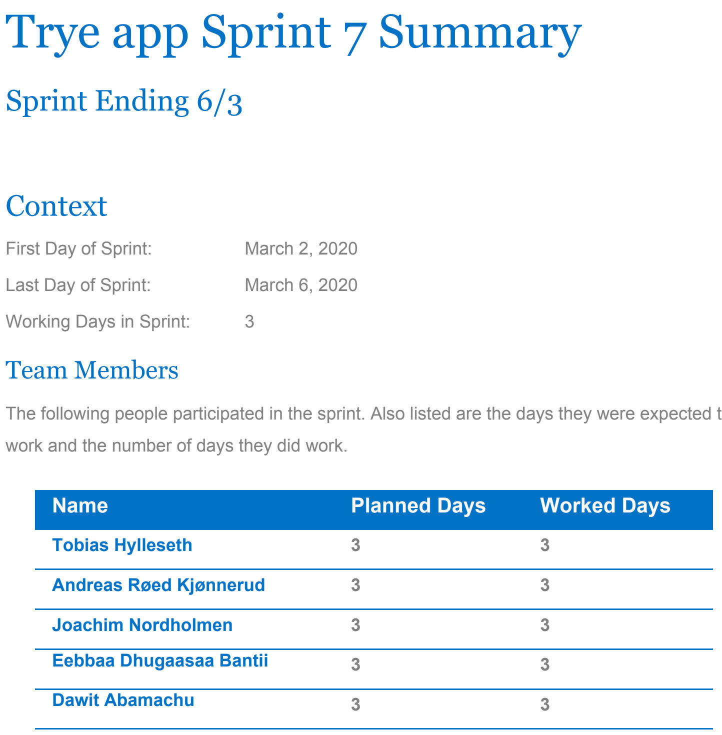
Sprint Review

- We worked hard during the sprint but it’s not reflected so much in the points since we are waiting for a part or we need to verify the work
- We who work with software made a forum post about our problem with automated text messages which is the last missing piece in the login system. The task is 90% complete. So we want to focus on other tasks next week.
- Hardware guys have made a lot of progress, should be done early next week

Sprint Retrospective

- We should put up cards for alternative work so we gain points no matter what to motivate us more. Getting 0 points each sprint is demotivating.

A.1.7 Sprint 7



Trye app Sprint 7 Summary

Sprint Ending 6/3

Context

First Day of Sprint: March 2, 2020

Last Day of Sprint: March 6, 2020

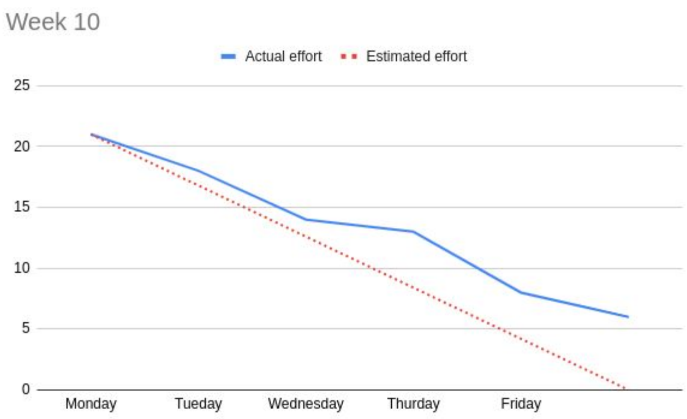
Working Days in Sprint: 3

Team Members

The following people participated in the sprint. Also listed are the days they were expected to work and the number of days they did work.

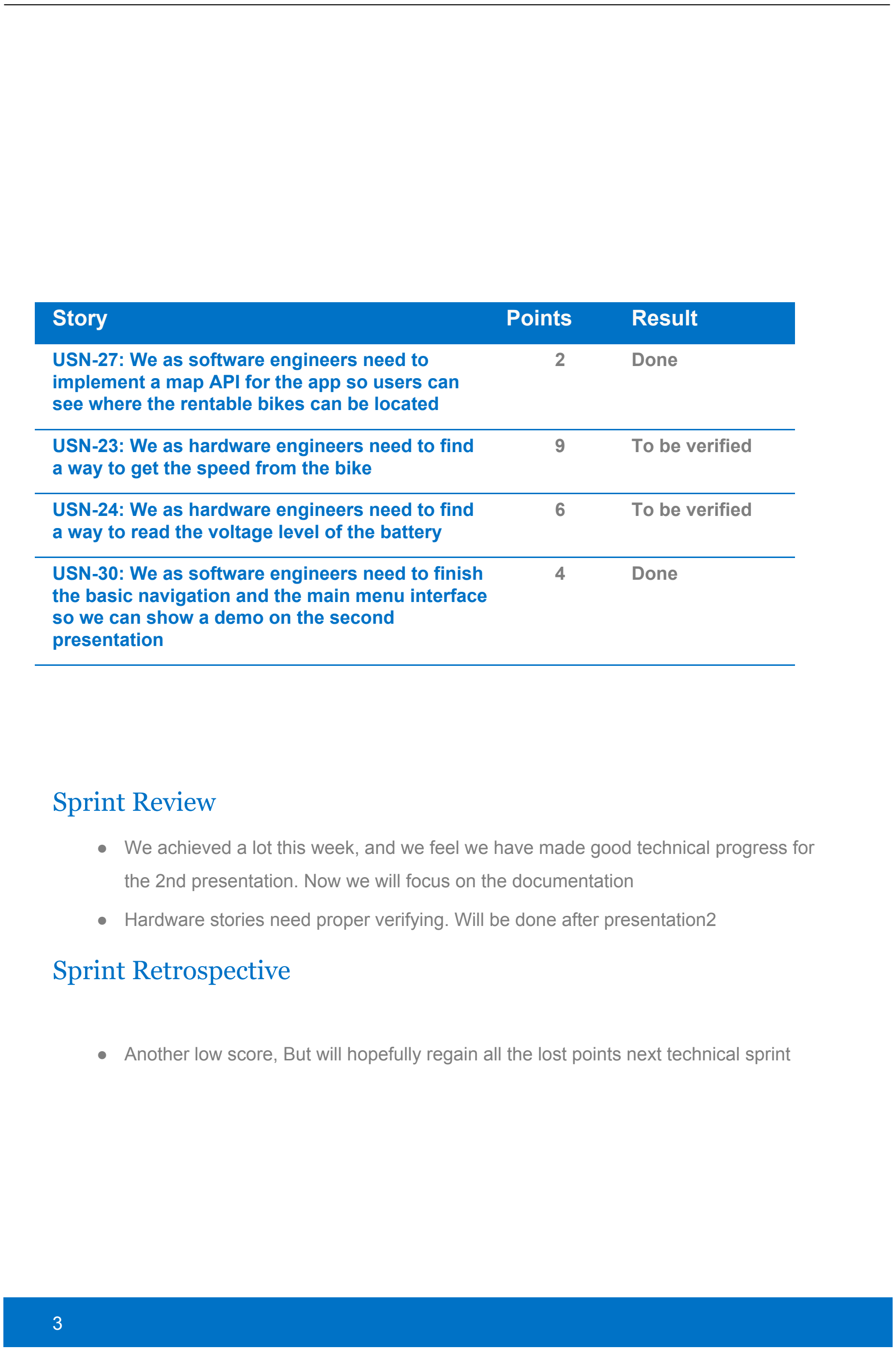
Name	Planned Days	Worked Days
Tobias Hylleseth	3	3
Andreas Røed Kjønnerud	3	3
Joachim Nordholmen	3	3
Eebbaa Dhugaasaa Bantii	3	3
Dawit Abamachu	3	3

Burndown chart



Contents and Assessment

Points Planned:	21
Points Earned:	6



Story	Points	Result
USN-27: We as software engineers need to implement a map API for the app so users can see where the rentable bikes can be located	2	Done
USN-23: We as hardware engineers need to find a way to get the speed from the bike	9	To be verified
USN-24: We as hardware engineers need to find a way to read the voltage level of the battery	6	To be verified
USN-30: We as software engineers need to finish the basic navigation and the main menu interface so we can show a demo on the second presentation	4	Done

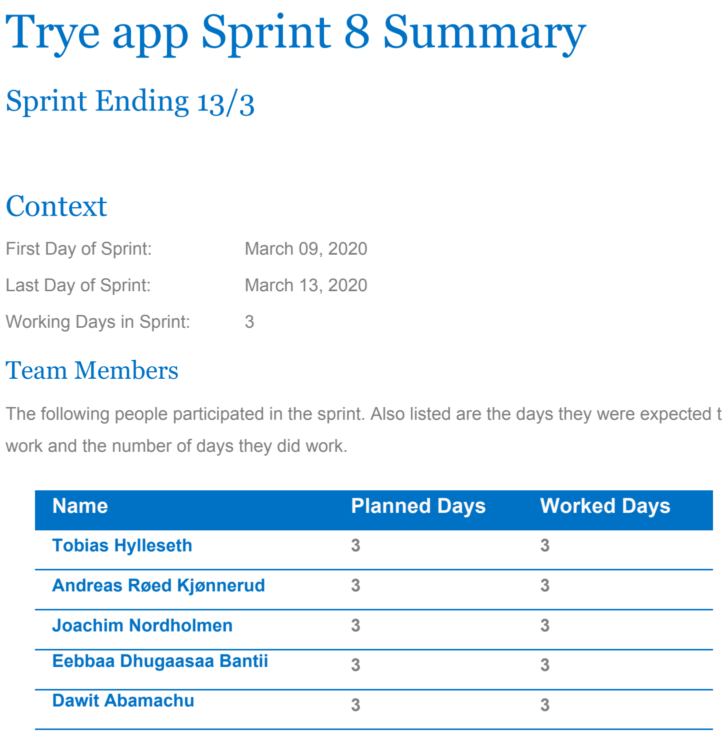
Sprint Review

- We achieved a lot this week, and we feel we have made good technical progress for the 2nd presentation. Now we will focus on the documentation
- Hardware stories need proper verifying. Will be done after presentation2

Sprint Retrospective

- Another low score, But will hopefully regain all the lost points next technical sprint

A.1.8 Sprint 8



Trye app Sprint 8 Summary

Sprint Ending 13/3

Context

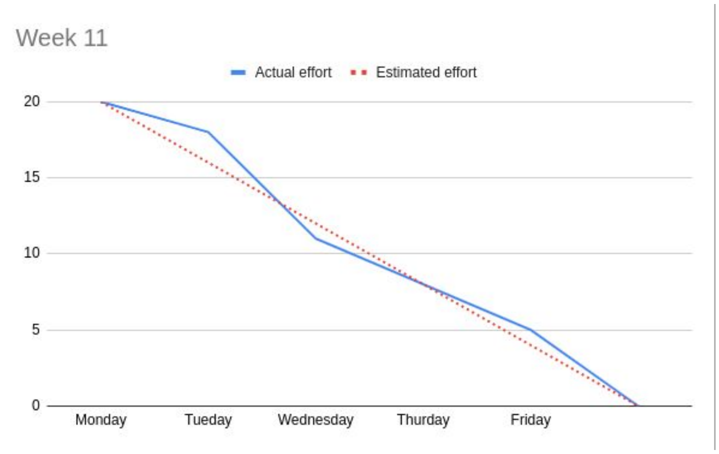
First Day of Sprint:	March 09, 2020
Last Day of Sprint:	March 13, 2020
Working Days in Sprint:	3

Team Members

The following people participated in the sprint. Also listed are the days they were expected to work and the number of days they did work.

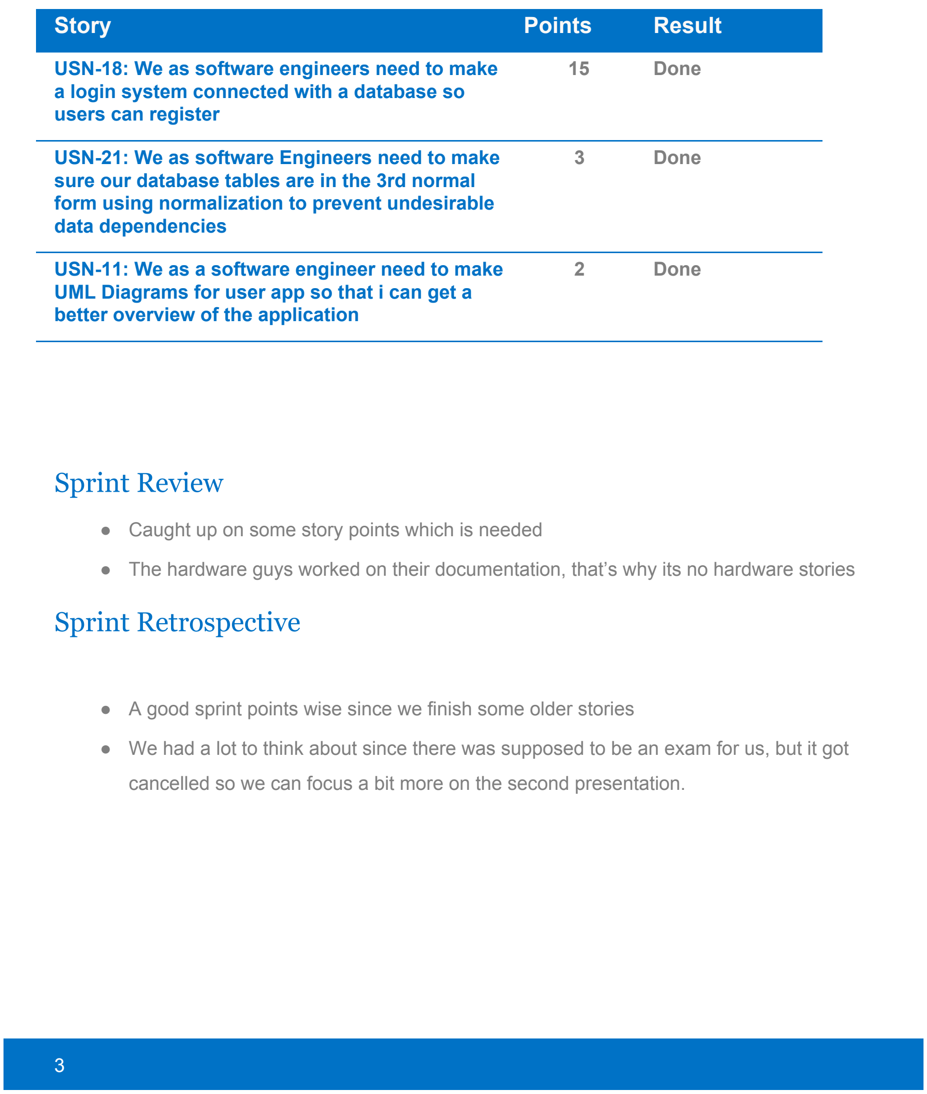
Name	Planned Days	Worked Days
Tobias Hylleseth	3	3
Andreas Røed Kjønnerud	3	3
Joachim Nordholmen	3	3
Eebbaa Dhugaasaa Bantii	3	3
Dawit Abamachu	3	3

Burndown chart



Contents and Assessment

Points Planned:	20
Points Earned:	20



Story	Points	Result
USN-18: We as software engineers need to make a login system connected with a database so users can register	15	Done
USN-21: We as software Engineers need to make sure our database tables are in the 3rd normal form using normalization to prevent undesirable data dependencies	3	Done
USN-11: We as a software engineer need to make UML Diagrams for user app so that i can get a better overview of the application	2	Done

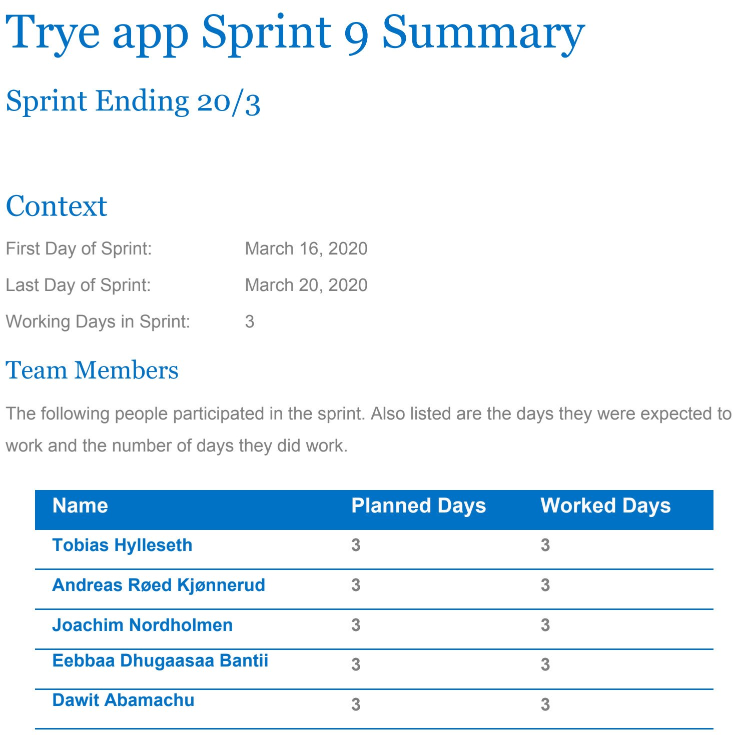
Sprint Review

- Caught up on some story points which is needed
- The hardware guys worked on their documentation, that's why its no hardware stories

Sprint Retrospective

- A good sprint points wise since we finish some older stories
- We had a lot to think about since there was supposed to be an exam for us, but it got cancelled so we can focus a bit more on the second presentation.

A.1.9 Sprint 9



Trye app Sprint 9 Summary

Sprint Ending 20/3

Context

First Day of Sprint: March 16, 2020

Last Day of Sprint: March 20, 2020

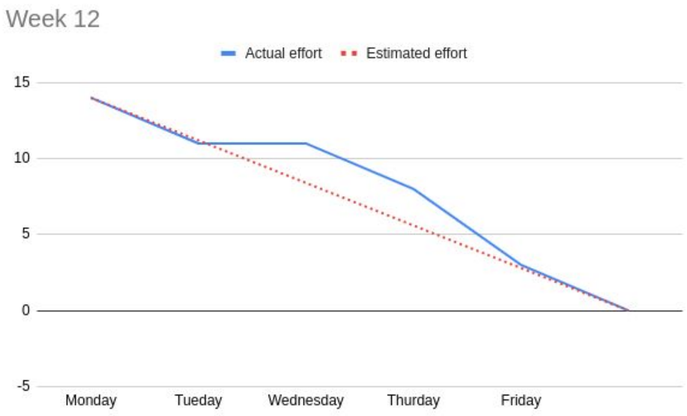
Working Days in Sprint: 3

Team Members

The following people participated in the sprint. Also listed are the days they were expected to work and the number of days they did work.

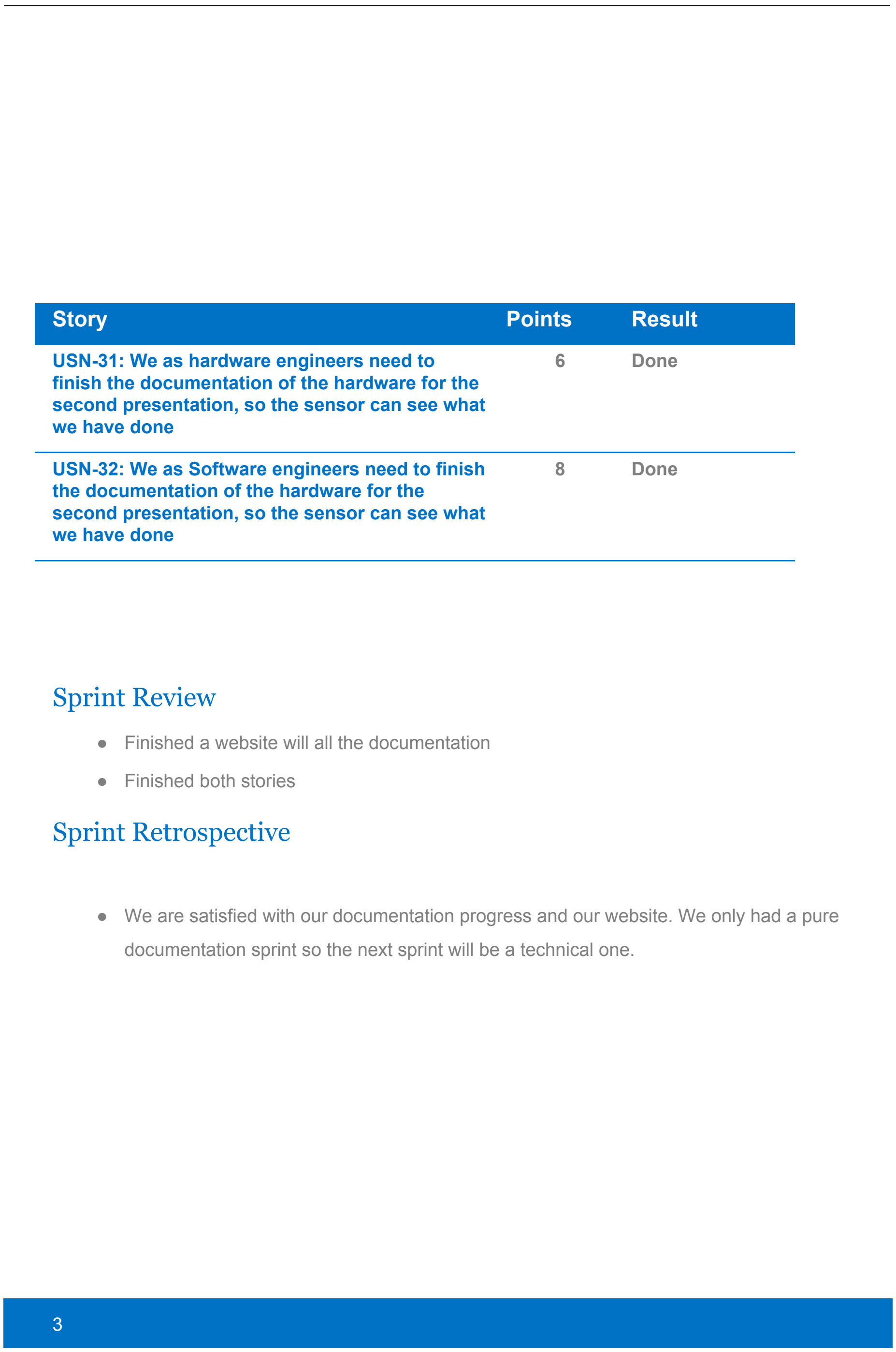
Name	Planned Days	Worked Days
Tobias Hylleseth	3	3
Andreas Røed Kjønnerud	3	3
Joachim Nordholmen	3	3
Eebbaa Dhugaasaa Bantii	3	3
Dawit Abamachu	3	3

Burndown chart



Contents and Assessment

Points Planned:	14
Points Earned:	14



Story	Points	Result
USN-31: We as hardware engineers need to finish the documentation of the hardware for the second presentation, so the sensor can see what we have done	6	Done
USN-32: We as Software engineers need to finish the documentation of the hardware for the second presentation, so the sensor can see what we have done	8	Done

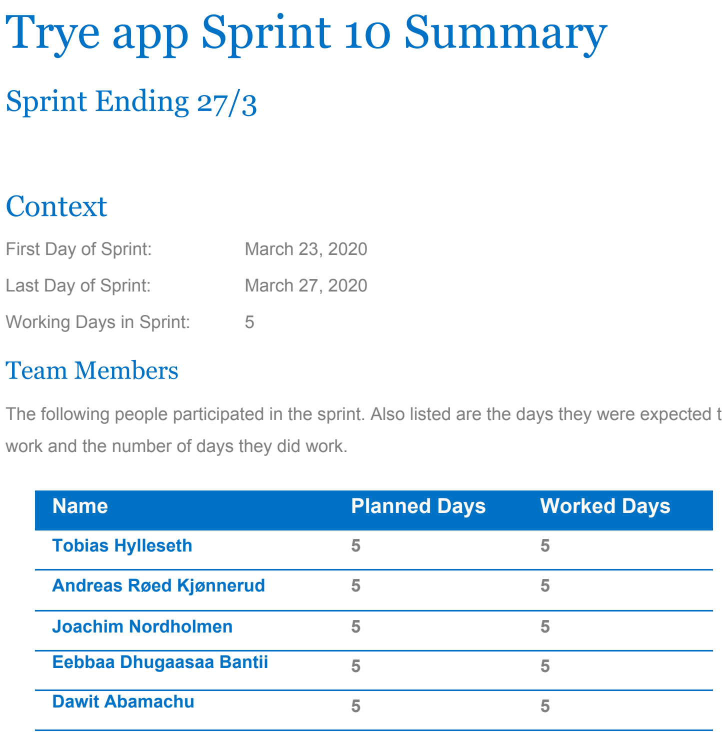
Sprint Review

- Finished a website will all the documentation
- Finished both stories

Sprint Retrospective

- We are satisfied with our documentation progress and our website. We only had a pure documentation sprint so the next sprint will be a technical one.

A.1.10 Sprint 10



Trye app Sprint 10 Summary

Sprint Ending 27/3

Context

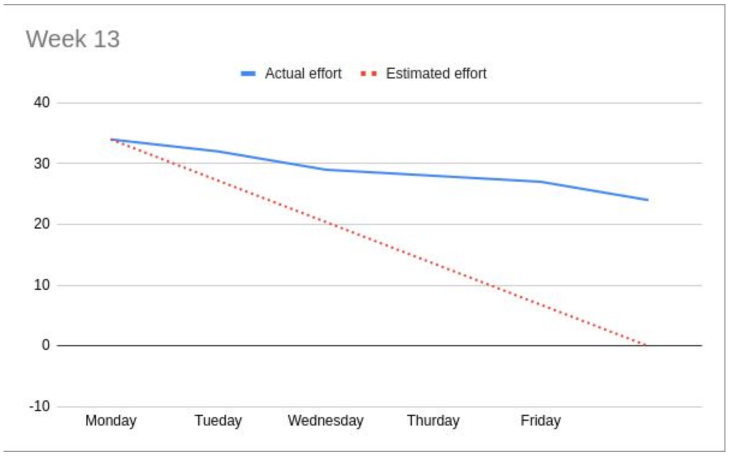
First Day of Sprint:	March 23, 2020
Last Day of Sprint:	March 27, 2020
Working Days in Sprint:	5

Team Members

The following people participated in the sprint. Also listed are the days they were expected to work and the number of days they did work.

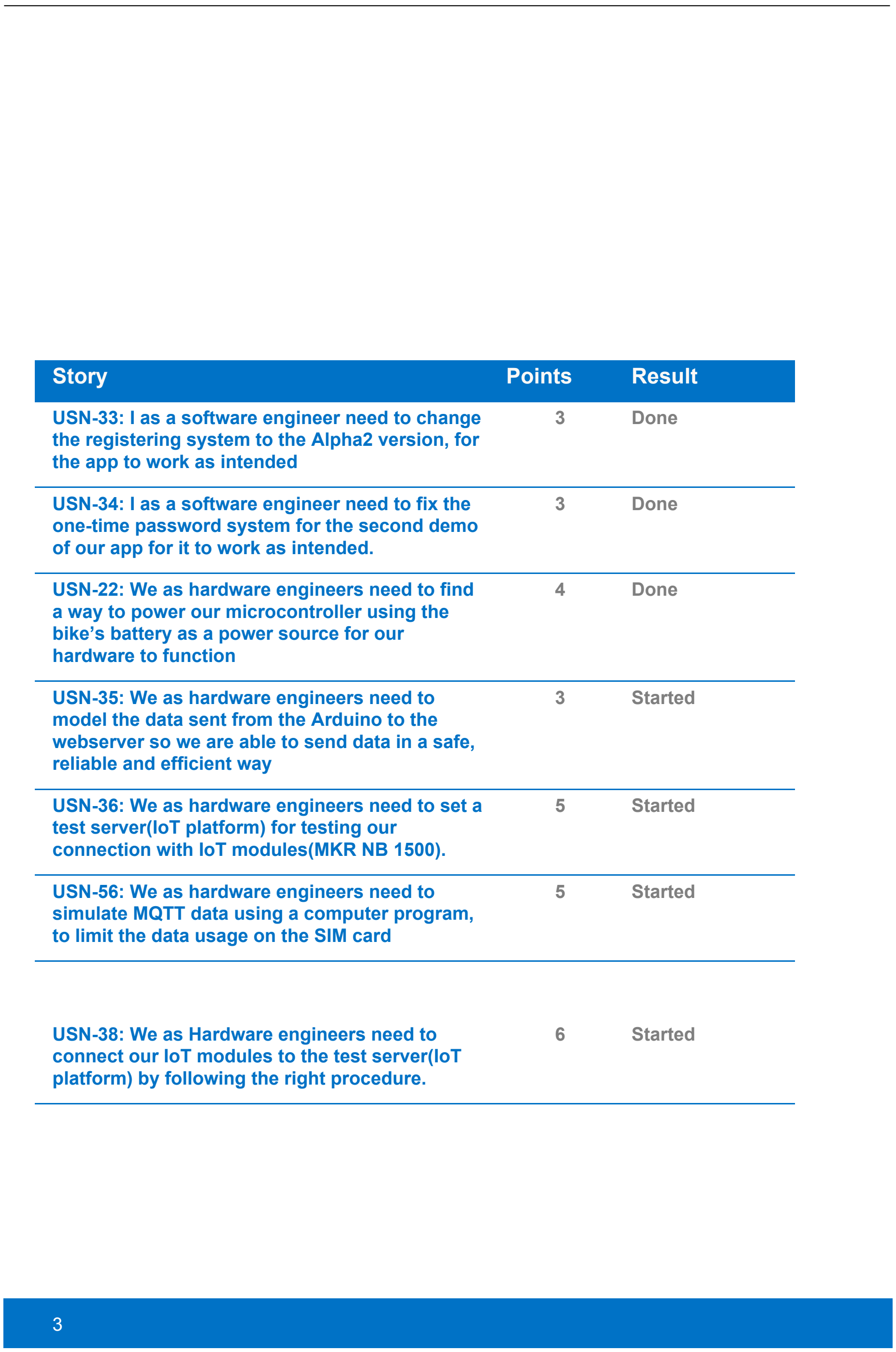
Name	Planned Days	Worked Days
Tobias Hylleseth	5	5
Andreas Røed Kjønnerud	5	5
Joachim Nordholmen	5	5
Eebbaa Dhugaasaa Bantii	5	5
Dawit Abamachu	5	5

Burndown chart



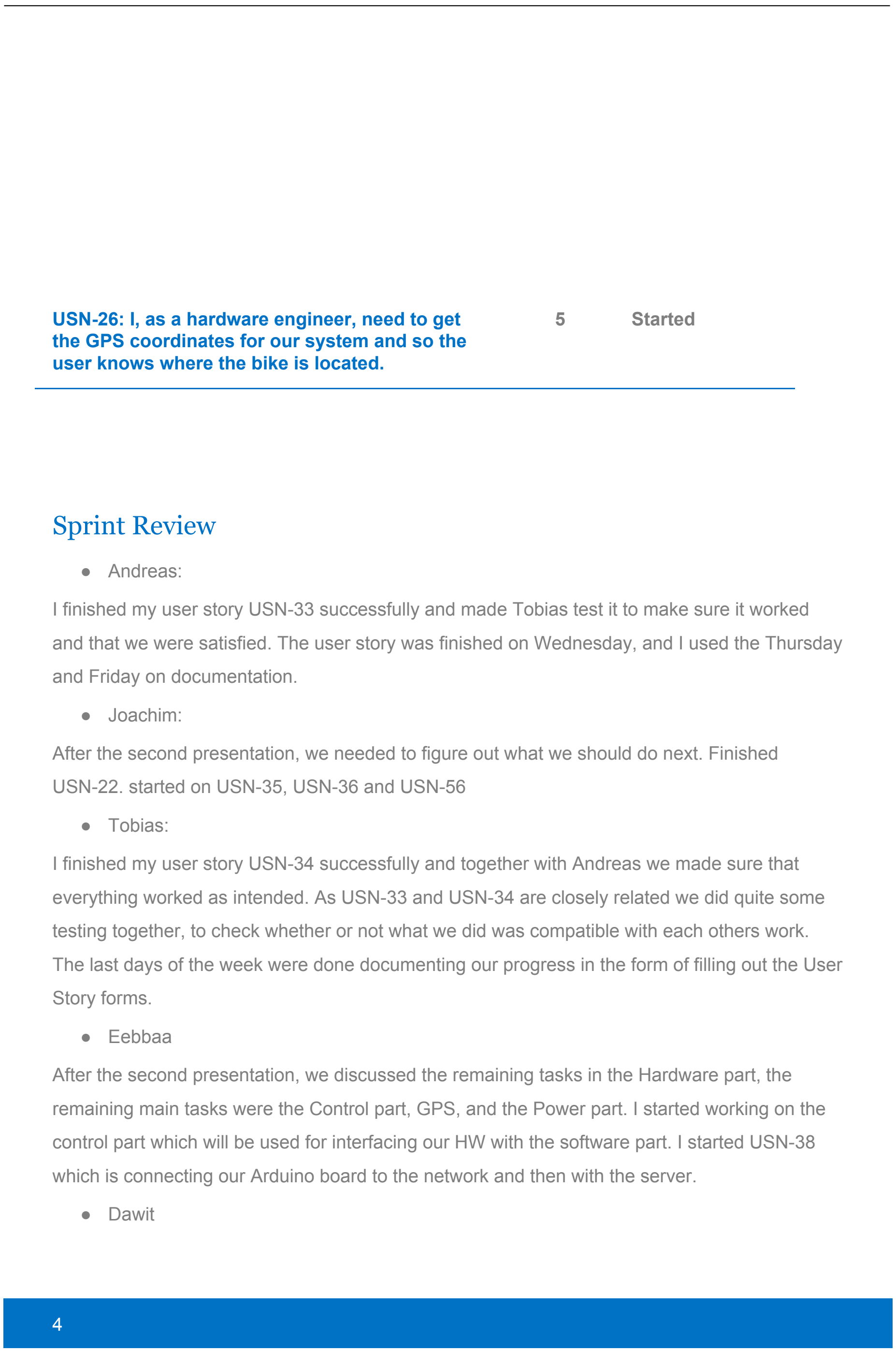
Contents and Assessment

Points Planned:	34
Points Earned:	10



Story	Points	Result
USN-33: I as a software engineer need to change the registering system to the Alpha2 version, for the app to work as intended	3	Done
USN-34: I as a software engineer need to fix the one-time password system for the second demo of our app for it to work as intended.	3	Done
USN-22: We as hardware engineers need to find a way to power our microcontroller using the bike’s battery as a power source for our hardware to function	4	Done
USN-35: We as hardware engineers need to model the data sent from the Arduino to the webserver so we are able to send data in a safe, reliable and efficient way	3	Started
USN-36: We as hardware engineers need to set a test server(IoT platform) for testing our connection with IoT modules(MKR NB 1500).	5	Started
USN-56: We as hardware engineers need to simulate MQTT data using a computer program, to limit the data usage on the SIM card	5	Started
USN-38: We as Hardware engineers need to connect our IoT modules to the test server(IoT platform) by following the right procedure.	6	Started

3



USN-26: I, as a hardware engineer, need to get the GPS coordinates for our system and so the user knows where the bike is located.	**5**	**Started**

Sprint Review

- Andreas:

I finished my user story USN-33 successfully and made Tobias test it to make sure it worked and that we were satisfied. The user story was finished on Wednesday, and I used the Thursday and Friday on documentation.

- Joachim:

After the second presentation, we needed to figure out what we should do next. Finished USN-22. started on USN-35, USN-36 and USN-56

- Tobias:

I finished my user story USN-34 successfully and together with Andreas we made sure that everything worked as intended. As USN-33 and USN-34 are closely related we did quite some testing together, to check whether or not what we did was compatible with each others work. The last days of the week were done documenting our progress in the form of filling out the User Story forms.

- Eebbaa

After the second presentation, we discussed the remaining tasks in the Hardware part, the remaining main tasks were the Control part, GPS, and the Power part. I started working on the control part which will be used for interfacing our HW with the software part. I started USN-38 which is connecting our Arduino board to the network and then with the server.

- Dawit

After the second presentation, since we had left with the power system, IoT system, and the GPS coordinating system, we as a hardware team divided the task to each other. The part I worked with was the GPS coordinating system. Here the goal was to get the GPS coordinates, and so the user knows where the bike locates. Therefore, I first started to search for the GPS module that supports our system. Then selecting GPS module NEO-6M, I used it for interfacing with the Arduino for my next step..

Sprint Retrospective

- Andreas:

I learned how to format the WHERE statement in the database API calls which will be very useful for me in the next user story I will work on.

- Joachim:

I added too many user stories to sprint backlog at a time, meaning I will not be able to finish all the user stories this week, therefore I should put back one of the user stories until next week.

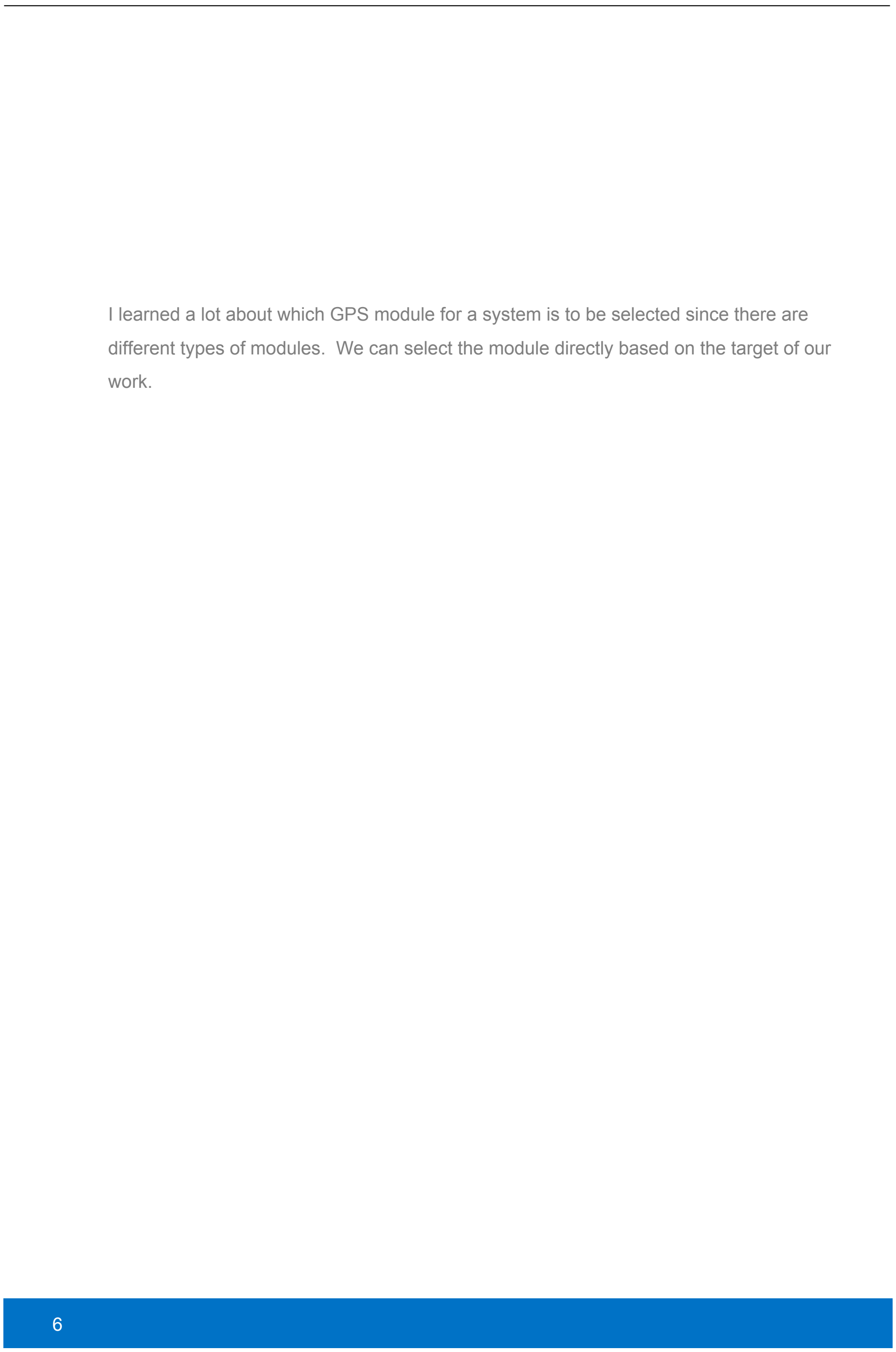
- Tobias:

Together with Andreas, we finalized the registering system, meaning we are on track to a working demo in May. I learned a lot about security when it comes to OTP (One-time password), and implemented a solution that is fit for a native app.

- Eebbaa

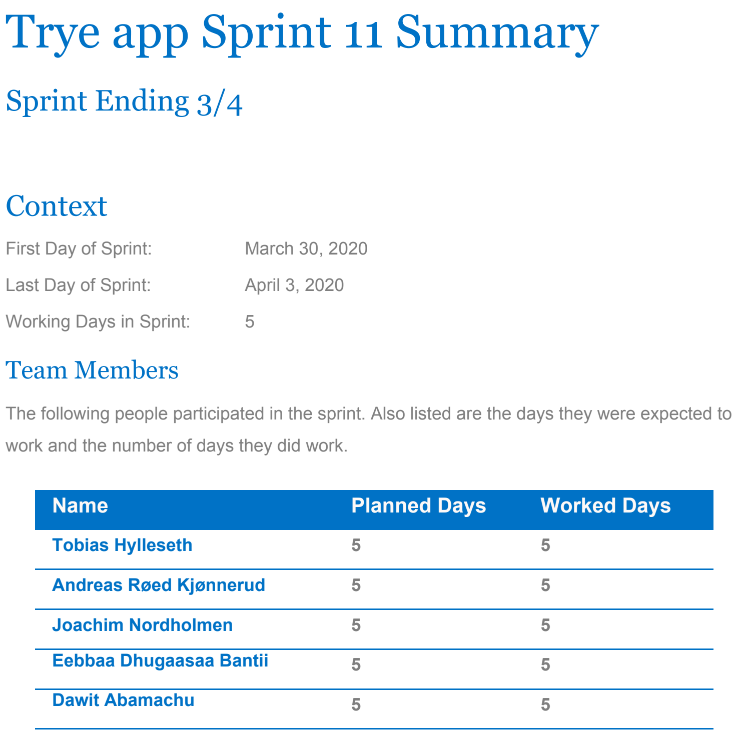
I started to work on our IoT device(Arduino with Telenor 4G sim) I spent time in finding out how Arduino IoT device connects to the internet. the are plenty of examples related to the topic so I used my time to understand what is going during connecting your IoT device and what you need.

- Dawit



I learned a lot about which GPS module for a system is to be selected since there are different types of modules. We can select the module directly based on the target of our work.

A.1.11 Sprint 11



Trye app Sprint 11 Summary

Sprint Ending 3/4

Context

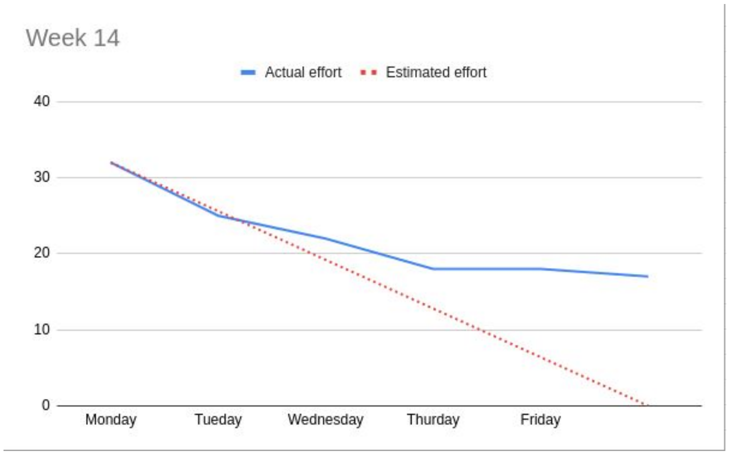
First Day of Sprint:	March 30, 2020
Last Day of Sprint:	April 3, 2020
Working Days in Sprint:	5

Team Members

The following people participated in the sprint. Also listed are the days they were expected to work and the number of days they did work.

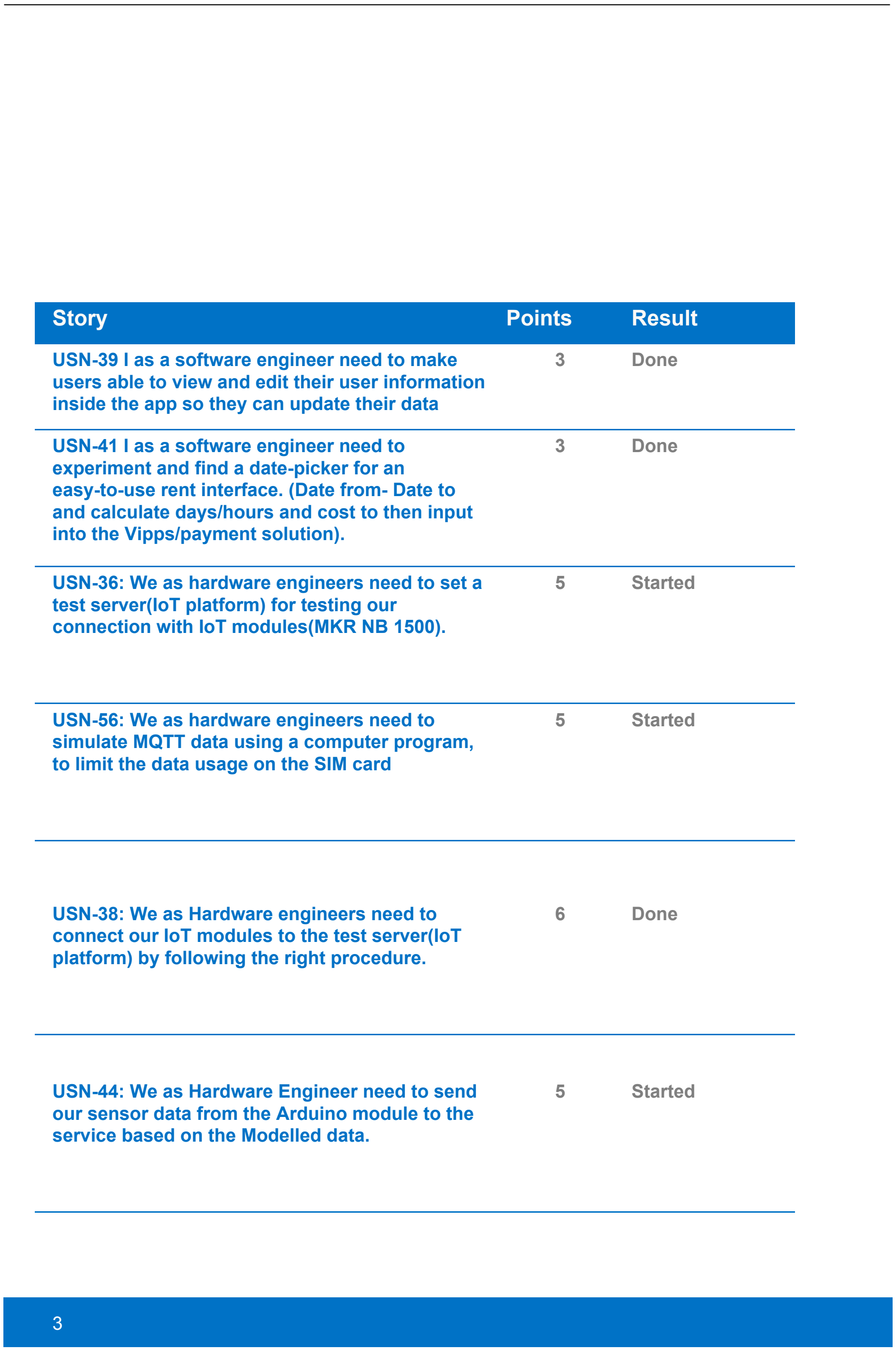
Name	Planned Days	Worked Days
Tobias Hylleseth	5	5
Andreas Røed Kjønnerud	5	5
Joachim Nordholmen	5	5
Eebbaa Dhugaasaa Bantii	5	5
Dawit Abamachu	5	5

Burndown chart



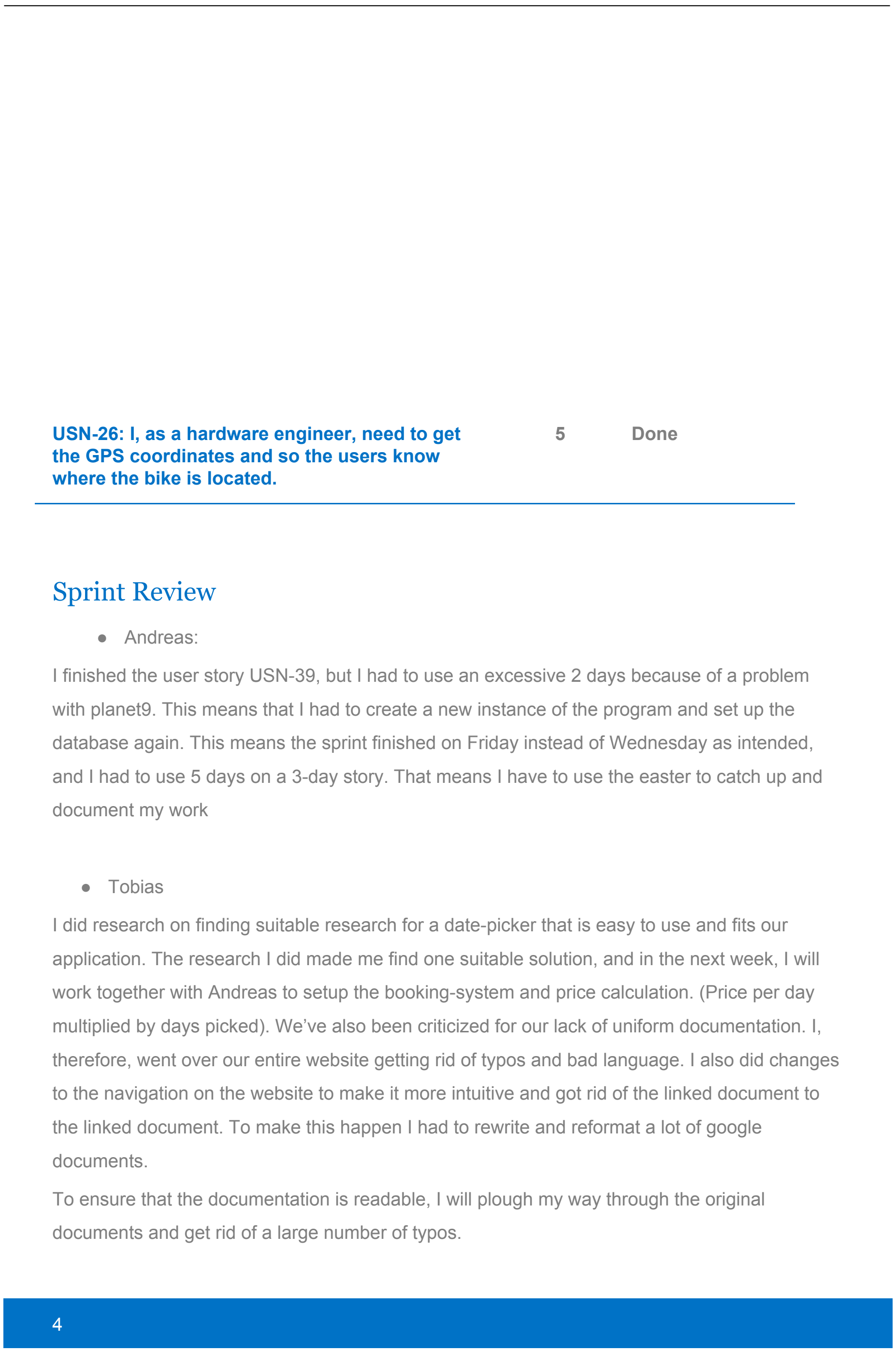
Contents and Assessment

Points Planned:	32
Points Earned:	17



Story	Points	Result
USN-39 I as a software engineer need to make users able to view and edit their user information inside the app so they can update their data	3	Done
USN-41 I as a software engineer need to experiment and find a date-picker for an easy-to-use rent interface. (Date from- Date to and calculate days/hours and cost to then input into the Vipps/payment solution).	3	Done
USN-36: We as hardware engineers need to set a test server(IoT platform) for testing our connection with IoT modules(MKR NB 1500).	5	Started
USN-56: We as hardware engineers need to simulate MQTT data using a computer program, to limit the data usage on the SIM card	5	Started
USN-38: We as Hardware engineers need to connect our IoT modules to the test server(IoT platform) by following the right procedure.	6	Done
USN-44: We as Hardware Engineer need to send our sensor data from the Arduino module to the service based on the Modelled data.	5	Started

3



USN-26: I, as a hardware engineer, need to get the GPS coordinates and so the users know where the bike is located.	**5**	**Done**

Sprint Review

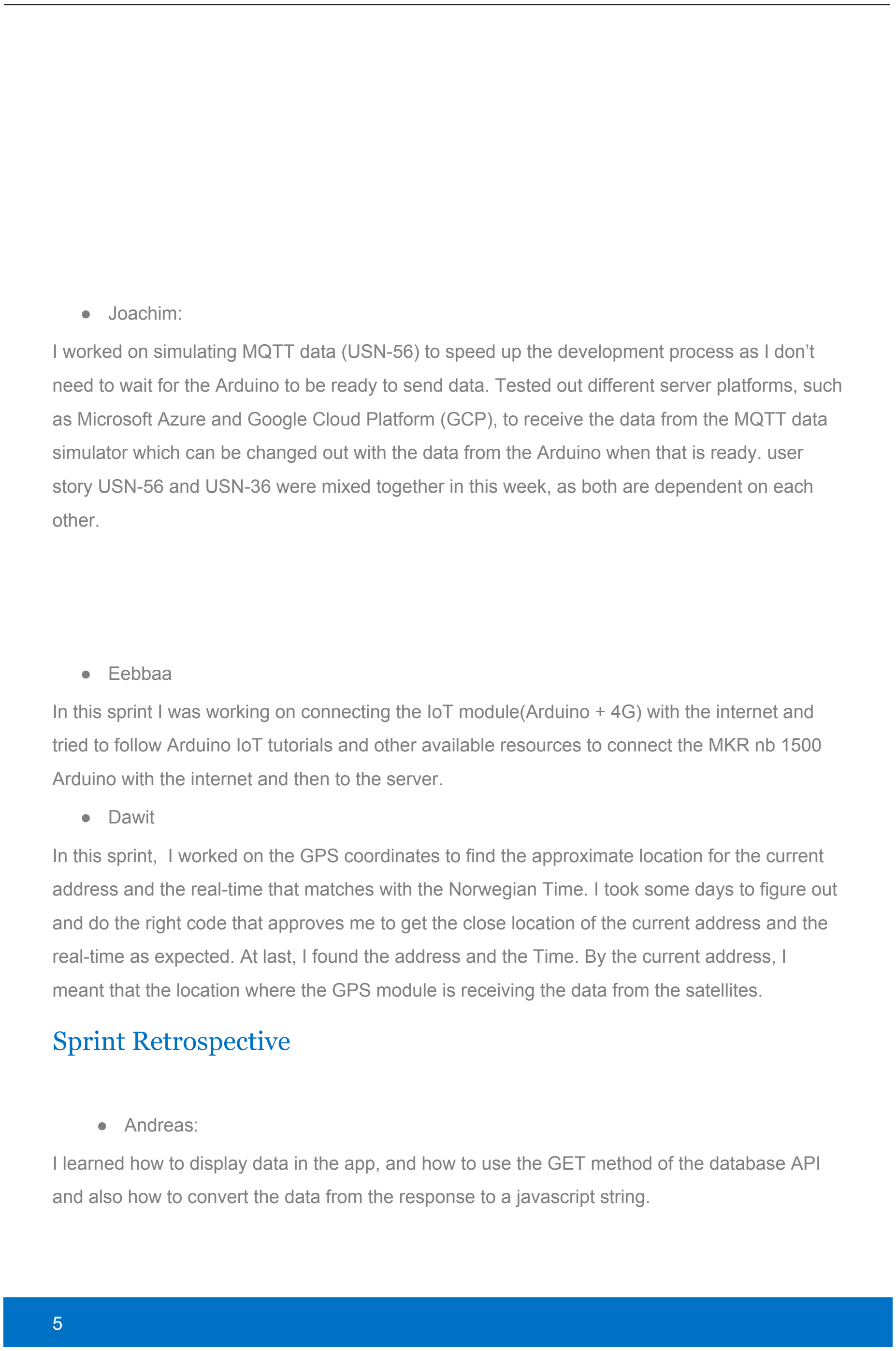
- Andreas:

I finished the user story USN-39, but I had to use an excessive 2 days because of a problem with planet9. This means that I had to create a new instance of the program and set up the database again. This means the sprint finished on Friday instead of Wednesday as intended, and I had to use 5 days on a 3-day story. That means I have to use the easter to catch up and document my work

- Tobias

I did research on finding suitable research for a date-picker that is easy to use and fits our application. The research I did made me find one suitable solution, and in the next week, I will work together with Andreas to setup the booking-system and price calculation. (Price per day multiplied by days picked). We've also been criticized for our lack of uniform documentation. I, therefore, went over our entire website getting rid of typos and bad language. I also did changes to the navigation on the website to make it more intuitive and got rid of the linked document to the linked document. To make this happen I had to rewrite and reformat a lot of google documents.

To ensure that the documentation is readable, I will plough my way through the original documents and get rid of a large number of typos.



- Joachim:

I worked on simulating MQTT data (USN-56) to speed up the development process as I don't need to wait for the Arduino to be ready to send data. Tested out different server platforms, such as Microsoft Azure and Google Cloud Platform (GCP), to receive the data from the MQTT data simulator which can be changed out with the data from the Arduino when that is ready. user story USN-56 and USN-36 were mixed together in this week, as both are dependent on each other.

- Eebbaa

In this sprint I was working on connecting the IoT module(Arduino + 4G) with the internet and tried to follow Arduino IoT tutorials and other available resources to connect the MKR nb 1500 Arduino with the internet and then to the server.

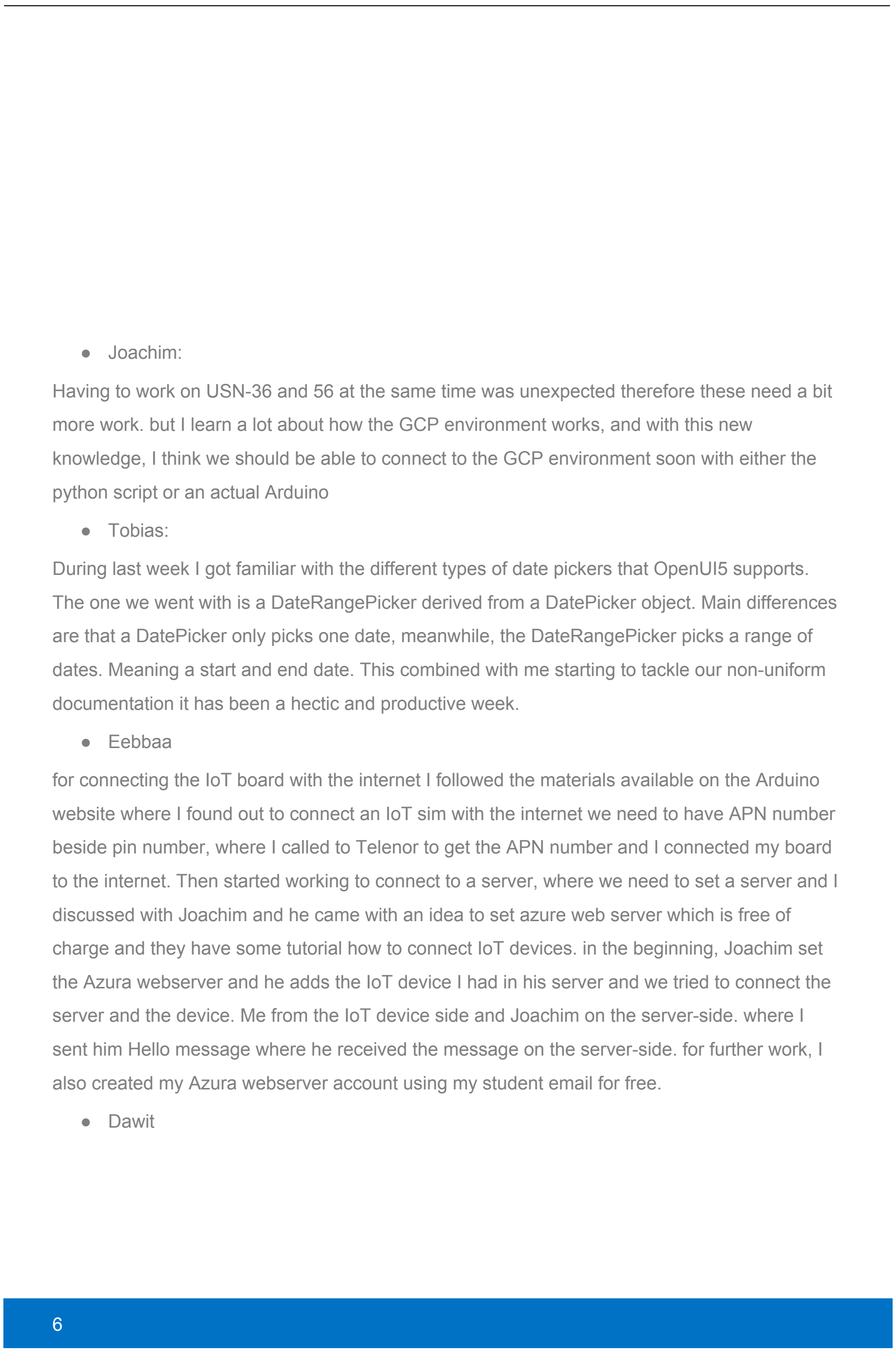
- Dawit

In this sprint, I worked on the GPS coordinates to find the approximate location for the current address and the real-time that matches with the Norwegian Time. I took some days to figure out and do the right code that approves me to get the close location of the current address and the real-time as expected. At last, I found the address and the Time. By the current address, I meant that the location where the GPS module is receiving the data from the satellites.

Sprint Retrospective

- Andreas:

I learned how to display data in the app, and how to use the GET method of the database API and also how to convert the data from the response to a javascript string.



- Joachim:

Having to work on USN-36 and 56 at the same time was unexpected therefore these need a bit more work. but I learn a lot about how the GCP environment works, and with this new knowledge, I think we should be able to connect to the GCP environment soon with either the python script or an actual Arduino

- Tobias:

During last week I got familiar with the different types of date pickers that OpenUI5 supports. The one we went with is a DateRangePicker derived from a DatePicker object. Main differences are that a DatePicker only picks one date, meanwhile, the DateRangePicker picks a range of dates. Meaning a start and end date. This combined with me starting to tackle our non-uniform documentation it has been a hectic and productive week.

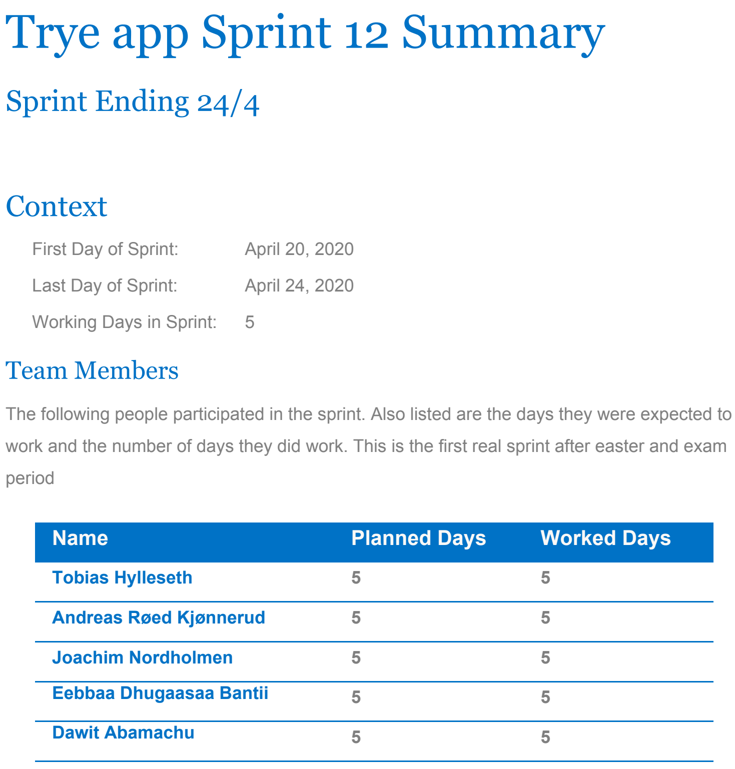
- Eebbaa

for connecting the IoT board with the internet I followed the materials available on the Arduino website where I found out to connect an IoT sim with the internet we need to have APN number beside pin number, where I called to Telenor to get the APN number and I connected my board to the internet. Then started working to connect to a server, where we need to set a server and I discussed with Joachim and he came with an idea to set azure web server which is free of charge and they have some tutorial how to connect IoT devices. in the beginning, Joachim set the Azura webserver and he adds the IoT device I had in his server and we tried to connect the server and the device. Me from the IoT device side and Joachim on the server-side. where I sent him Hello message where he received the message on the server-side. for further work, I also created my Azura webserver account using my student email for free.

- Dawit

I got a big lesson on how to interface the GPS module with the Arduino and find the specific data that I was looking after. I also perceived which environment is suitable for experimenting to receive data from the satellites.

A.1.12 Sprint 12



Trye app Sprint 12 Summary

Sprint Ending 24/4

Context

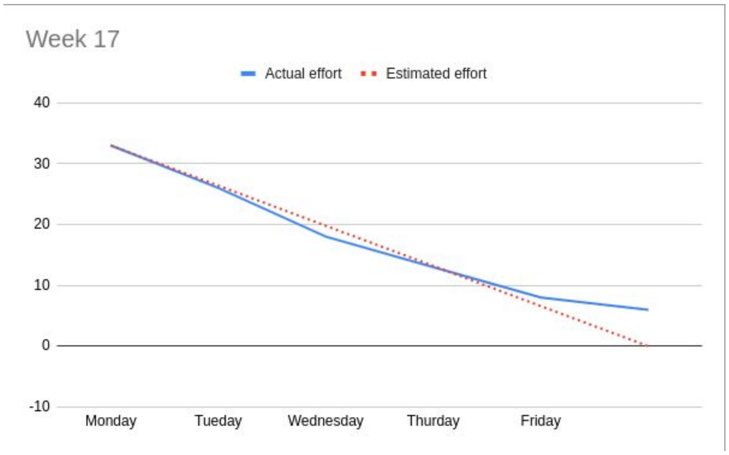
First Day of Sprint:	April 20, 2020
Last Day of Sprint:	April 24, 2020
Working Days in Sprint:	5

Team Members

The following people participated in the sprint. Also listed are the days they were expected to work and the number of days they did work. This is the first real sprint after easter and exam period

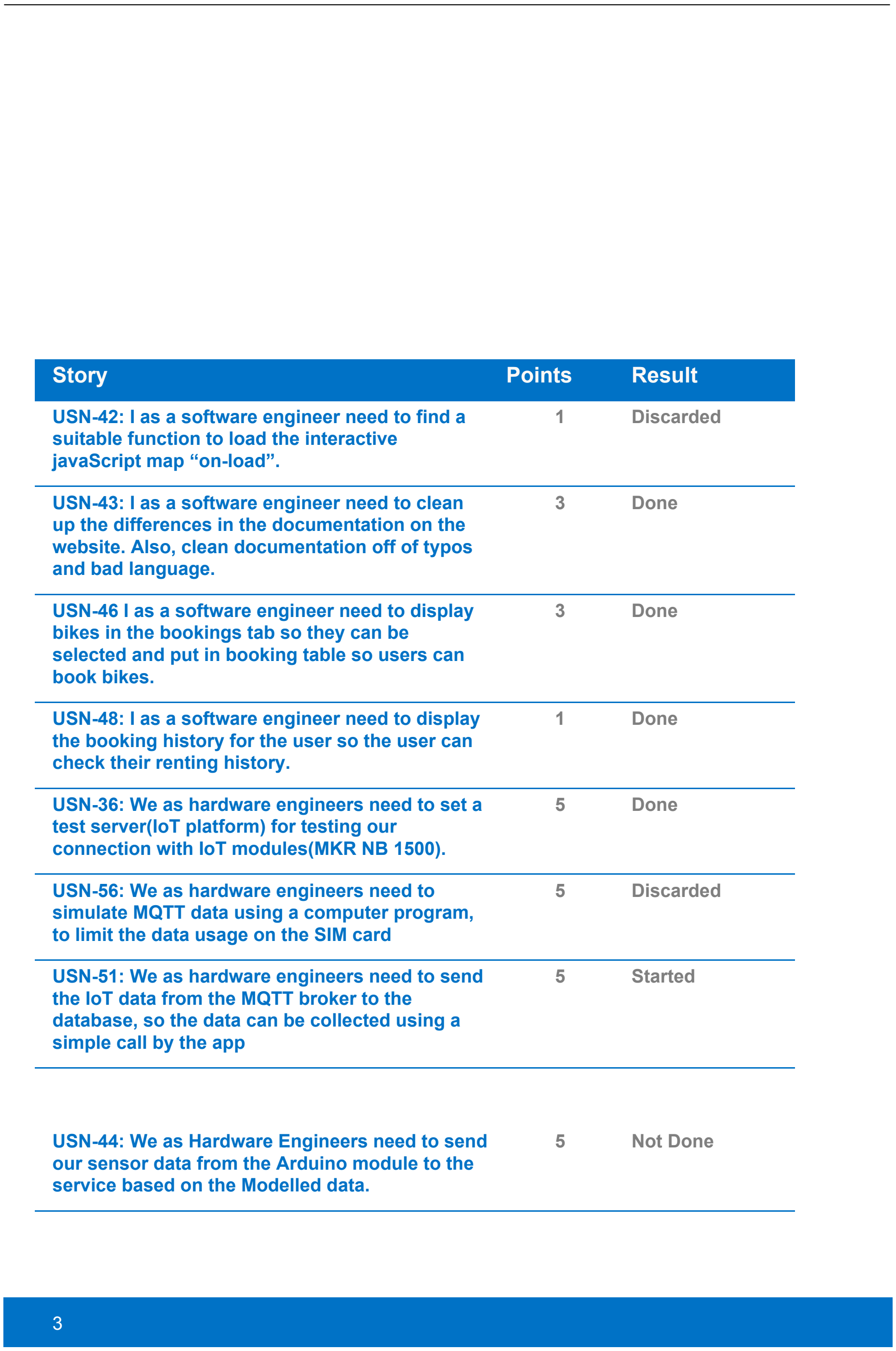
Name	Planned Days	Worked Days
Tobias Hylleseth	5	5
Andreas Røed Kjønnerud	5	5
Joachim Nordholmen	5	5
Eebbaa Dhugaasaa Bantii	5	5
Dawit Abamachu	5	5

Burndown chart



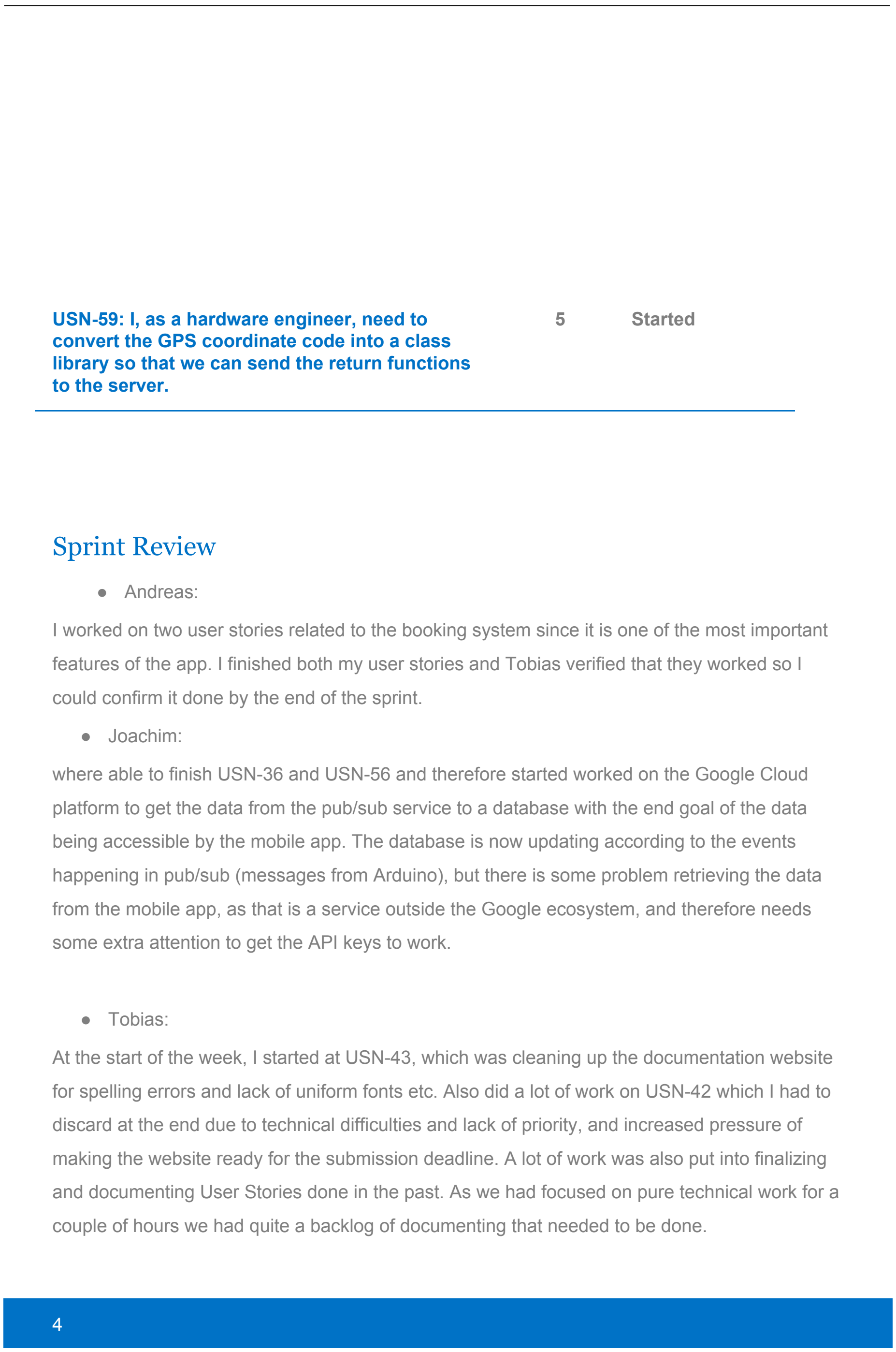
Contents and Assessment

Points Planned:	33
Points Earned:	27



Story	Points	Result
USN-42: I as a software engineer need to find a suitable function to load the interactive javaScript map “on-load”.	1	Discarded
USN-43: I as a software engineer need to clean up the differences in the documentation on the website. Also, clean documentation off of typos and bad language.	3	Done
USN-46 I as a software engineer need to display bikes in the bookings tab so they can be selected and put in booking table so users can book bikes.	3	Done
USN-48: I as a software engineer need to display the booking history for the user so the user can check their renting history.	1	Done
USN-36: We as hardware engineers need to set a test server(IoT platform) for testing our connection with IoT modules(MKR NB 1500).	5	Done
USN-56: We as hardware engineers need to simulate MQTT data using a computer program, to limit the data usage on the SIM card	5	Discarded
USN-51: We as hardware engineers need to send the IoT data from the MQTT broker to the database, so the data can be collected using a simple call by the app	5	Started
USN-44: We as Hardware Engineers need to send our sensor data from the Arduino module to the service based on the Modelled data.	5	Not Done

3



USN-59: I, as a hardware engineer, need to convert the GPS coordinate code into a class library so that we can send the return functions to the server.	**5**	**Started**

Sprint Review

- Andreas:

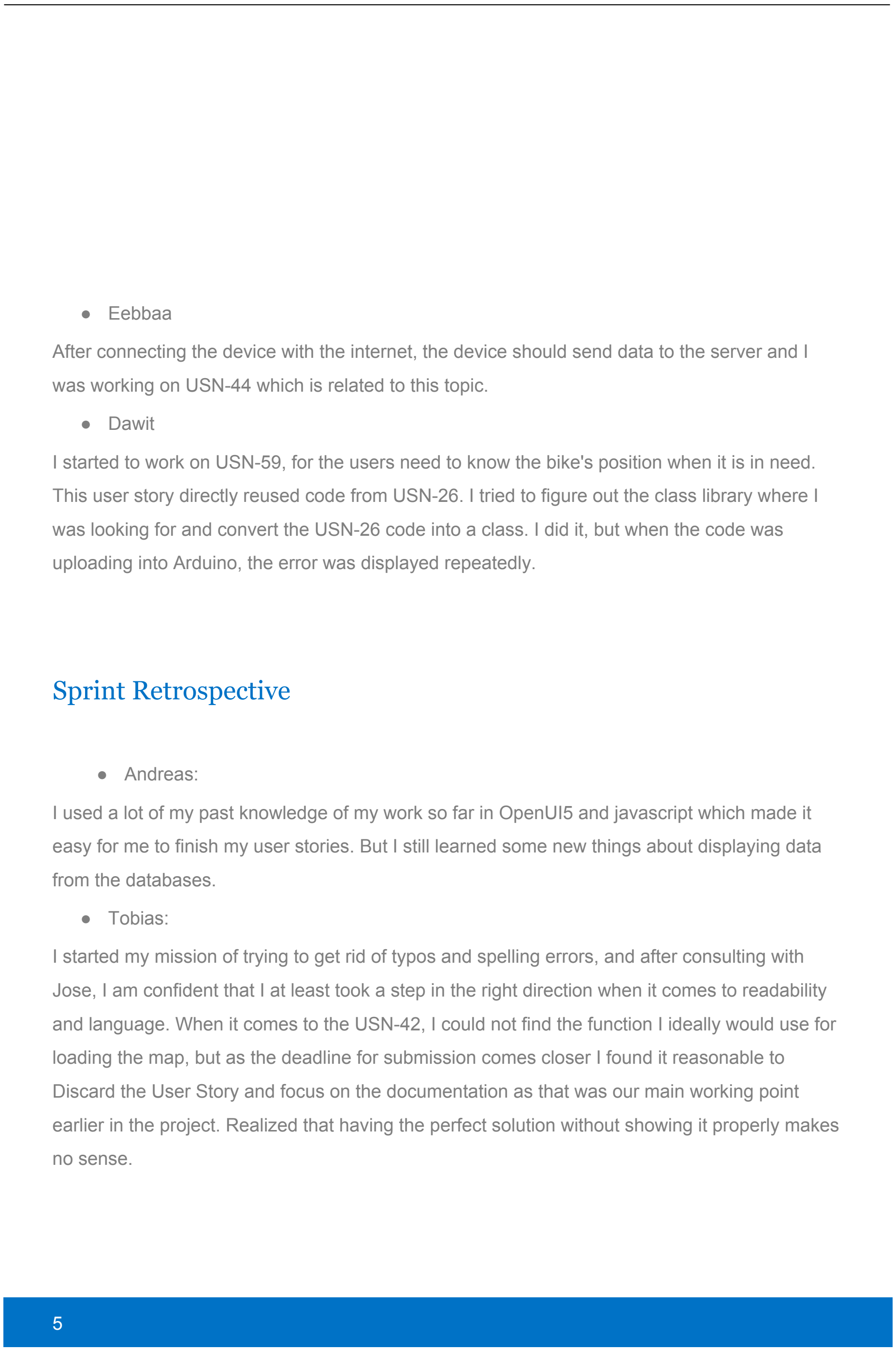
I worked on two user stories related to the booking system since it is one of the most important features of the app. I finished both my user stories and Tobias verified that they worked so I could confirm it done by the end of the sprint.

- Joachim:

where able to finish USN-36 and USN-56 and therefore started worked on the Google Cloud platform to get the data from the pub/sub service to a database with the end goal of the data being accessible by the mobile app. The database is now updating according to the events happening in pub/sub (messages from Arduino), but there is some problem retrieving the data from the mobile app, as that is a service outside the Google ecosystem, and therefore needs some extra attention to get the API keys to work.

- Tobias:

At the start of the week, I started at USN-43, which was cleaning up the documentation website for spelling errors and lack of uniform fonts etc. Also did a lot of work on USN-42 which I had to discard at the end due to technical difficulties and lack of priority, and increased pressure of making the website ready for the submission deadline. A lot of work was also put into finalizing and documenting User Stories done in the past. As we had focused on pure technical work for a couple of hours we had quite a backlog of documenting that needed to be done.



- Eebbaa

After connecting the device with the internet, the device should send data to the server and I was working on USN-44 which is related to this topic.

- Dawit

I started to work on USN-59, for the users need to know the bike's position when it is in need. This user story directly reused code from USN-26. I tried to figure out the class library where I was looking for and convert the USN-26 code into a class. I did it, but when the code was uploading into Arduino, the error was displayed repeatedly.

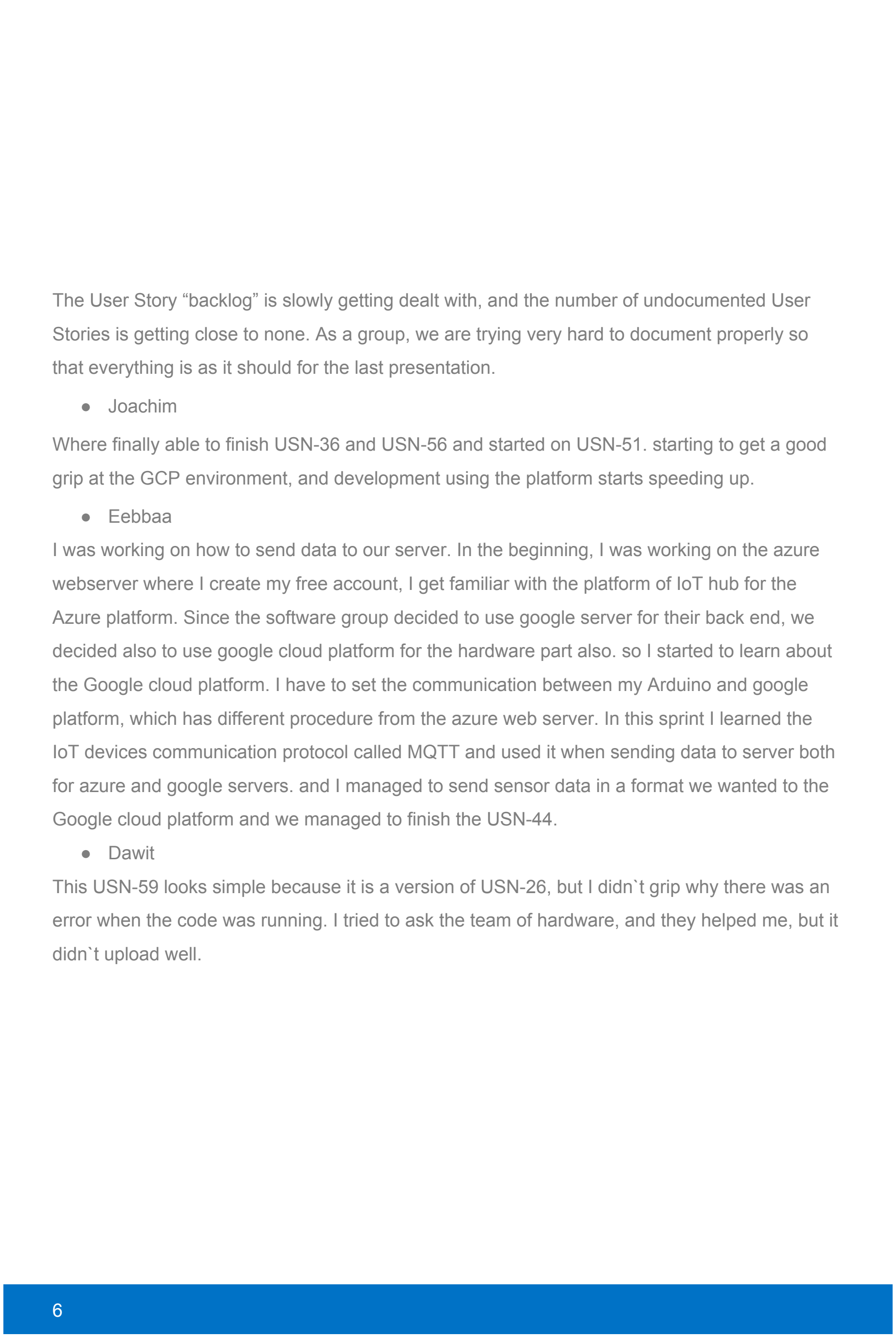
Sprint Retrospective

- Andreas:

I used a lot of my past knowledge of my work so far in OpenUI5 and javascript which made it easy for me to finish my user stories. But I still learned some new things about displaying data from the databases.

- Tobias:

I started my mission of trying to get rid of typos and spelling errors, and after consulting with Jose, I am confident that I at least took a step in the right direction when it comes to readability and language. When it comes to the USN-42, I could not find the function I ideally would use for loading the map, but as the deadline for submission comes closer I found it reasonable to Discard the User Story and focus on the documentation as that was our main working point earlier in the project. Realized that having the perfect solution without showing it properly makes no sense.



The User Story “backlog” is slowly getting dealt with, and the number of undocumented User Stories is getting close to none. As a group, we are trying very hard to document properly so that everything is as it should for the last presentation.

- Joachim

Where finally able to finish USN-36 and USN-56 and started on USN-51. starting to get a good grip at the GCP environment, and development using the platform starts speeding up.

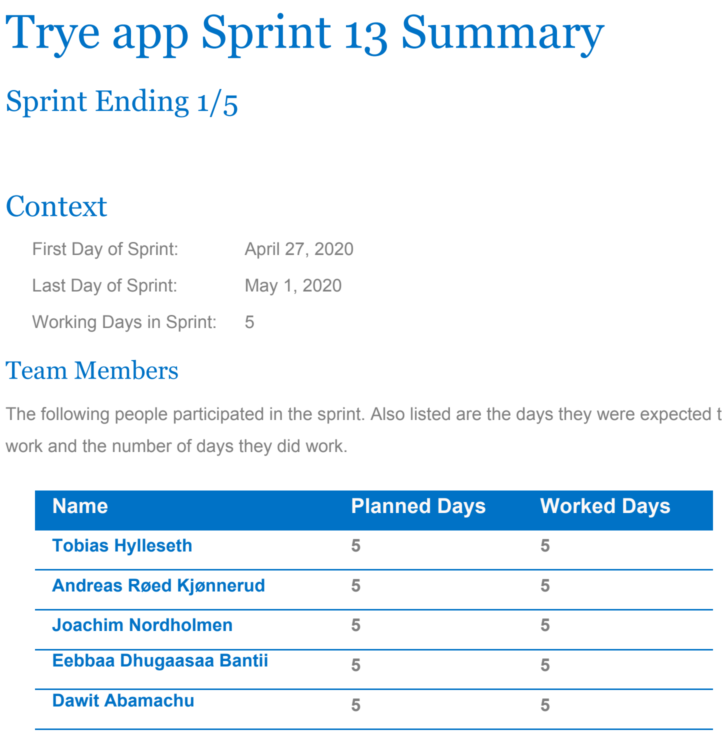
- Eebbaa

I was working on how to send data to our server. In the beginning, I was working on the azure webserver where I create my free account, I get familiar with the platform of IoT hub for the Azure platform. Since the software group decided to use google server for their back end, we decided also to use google cloud platform for the hardware part also. so I started to learn about the Google cloud platform. I have to set the communication between my Arduino and google platform, which has different procedure from the azure web server. In this sprint I learned the IoT devices communication protocol called MQTT and used it when sending data to server both for azure and google servers. and I managed to send sensor data in a format we wanted to the Google cloud platform and we managed to finish the USN-44.

- Dawit

This USN-59 looks simple because it is a version of USN-26, but I didn`t grip why there was an error when the code was running. I tried to ask the team of hardware, and they helped me, but it didn`t upload well.

A.1.13 Sprint 13



Trye app Sprint 13 Summary

Sprint Ending 1/5

Context

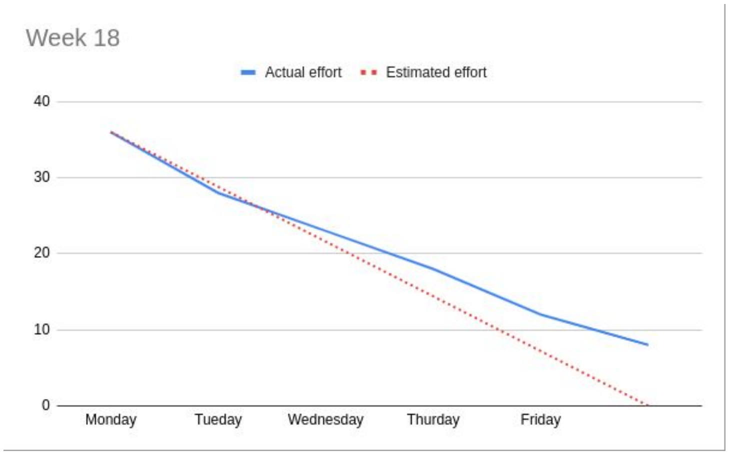
First Day of Sprint:	April 27, 2020
Last Day of Sprint:	May 1, 2020
Working Days in Sprint:	5

Team Members

The following people participated in the sprint. Also listed are the days they were expected to work and the number of days they did work.

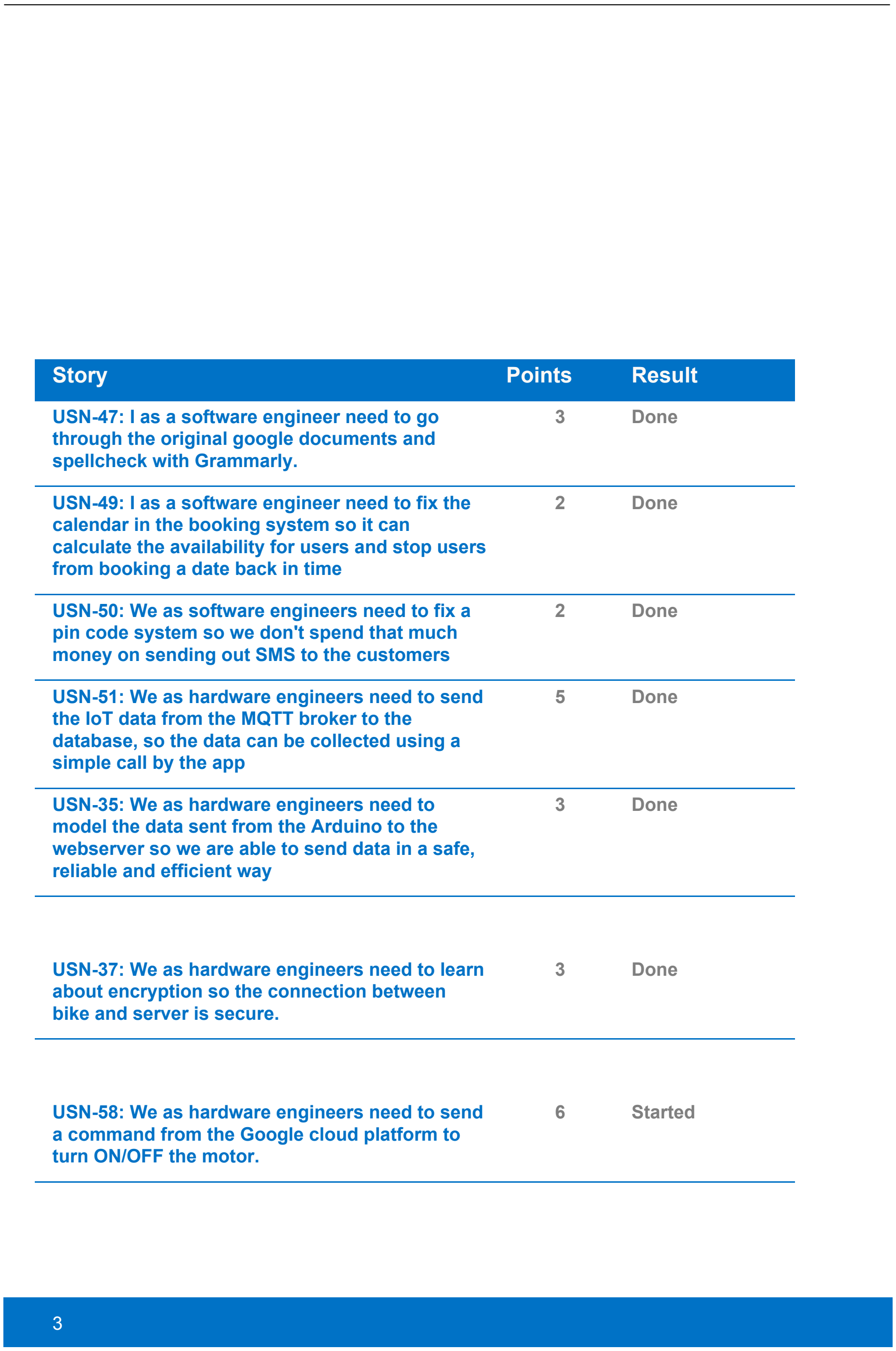
Name	Planned Days	Worked Days
Tobias Hylleseth	5	5
Andreas Røed Kjønnerud	5	5
Joachim Nordholmen	5	5
Eebbaa Dhugaasaa Bantii	5	5
Dawit Abamachu	5	5

Burndown chart



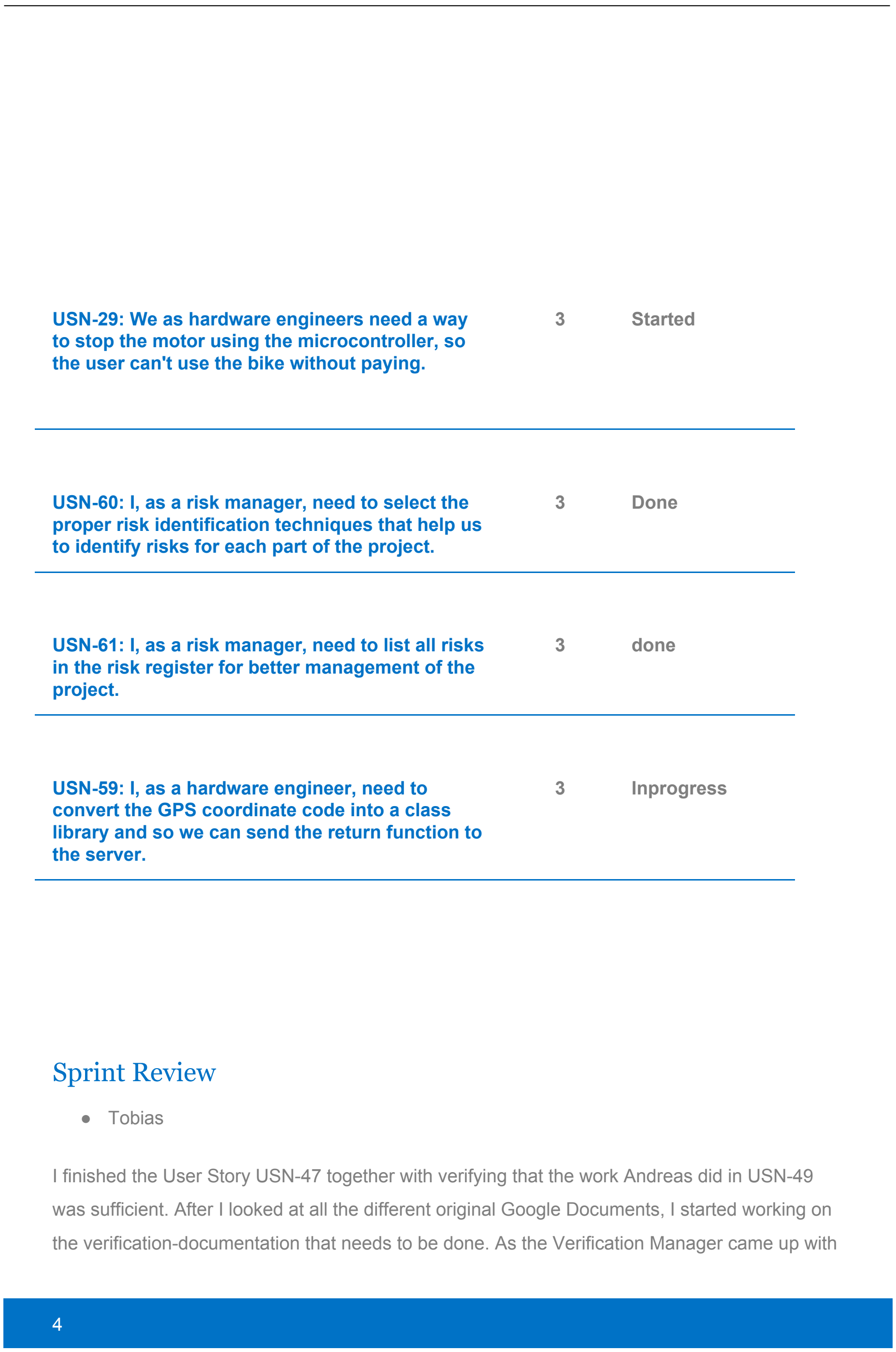
Contents and Assessment

Points Planned:	36
Points Earned:	24



Story	Points	Result
USN-47: I as a software engineer need to go through the original google documents and spellcheck with Grammarly.	3	Done
USN-49: I as a software engineer need to fix the calendar in the booking system so it can calculate the availability for users and stop users from booking a date back in time	2	Done
USN-50: We as software engineers need to fix a pin code system so we don't spend that much money on sending out SMS to the customers	2	Done
USN-51: We as hardware engineers need to send the IoT data from the MQTT broker to the database, so the data can be collected using a simple call by the app	5	Done
USN-35: We as hardware engineers need to model the data sent from the Arduino to the webserver so we are able to send data in a safe, reliable and efficient way	3	Done
USN-37: We as hardware engineers need to learn about encryption so the connection between bike and server is secure.	3	Done
USN-58: We as hardware engineers need to send a command from the Google cloud platform to turn ON/OFF the motor.	6	Started

3

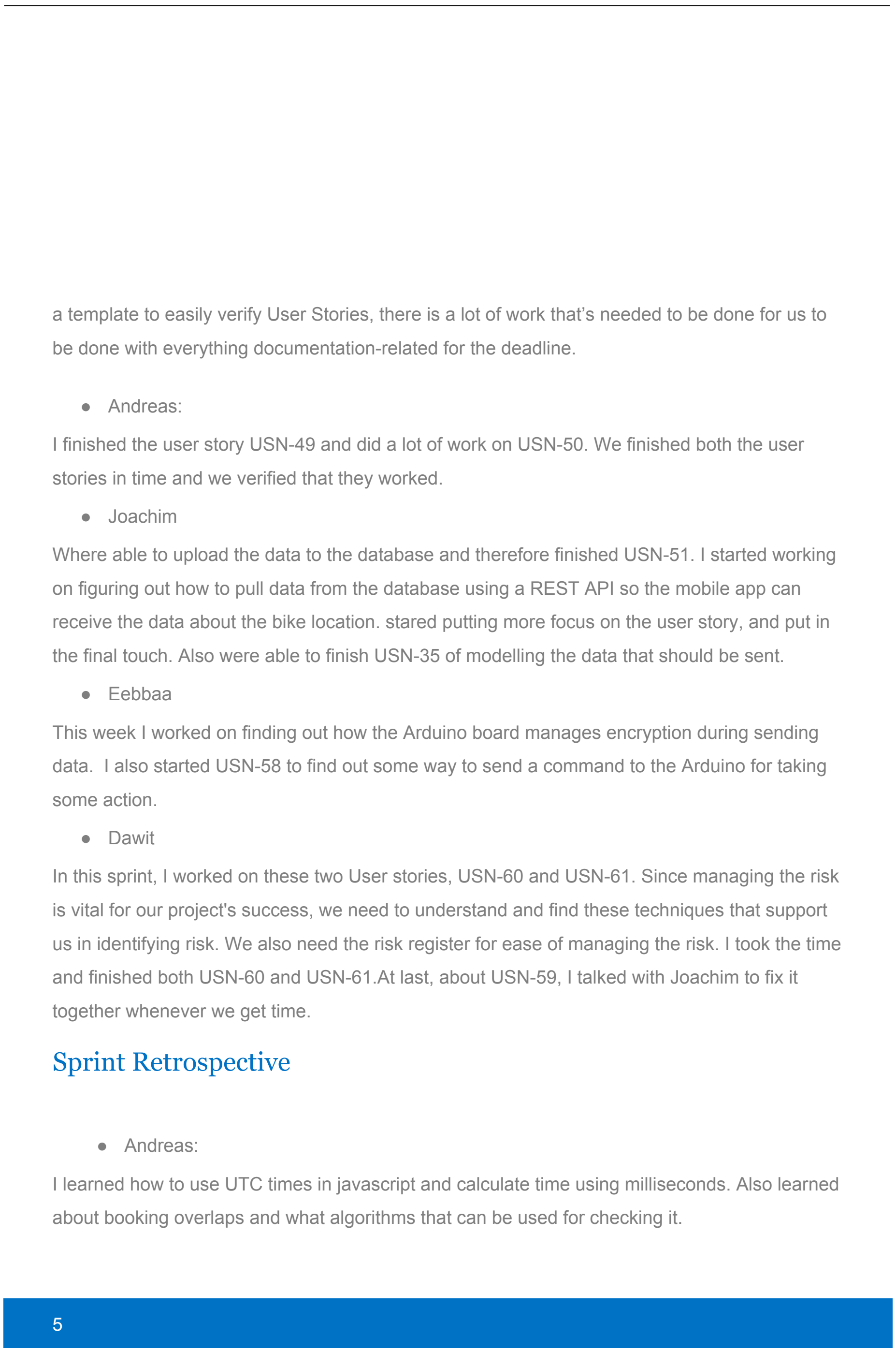


USN-29: We as hardware engineers need a way to stop the motor using the microcontroller, so the user can't use the bike without paying.	3	Started
USN-60: I, as a risk manager, need to select the proper risk identification techniques that help us to identify risks for each part of the project.	3	Done
USN-61: I, as a risk manager, need to list all risks in the risk register for better management of the project.	3	done
USN-59: I, as a hardware engineer, need to convert the GPS coordinate code into a class library and so we can send the return function to the server.	3	Inprogress

Sprint Review

- Tobias

I finished the User Story USN-47 together with verifying that the work Andreas did in USN-49 was sufficient. After I looked at all the different original Google Documents, I started working on the verification-documentation that needs to be done. As the Verification Manager came up with



a template to easily verify User Stories, there is a lot of work that’s needed to be done for us to be done with everything documentation-related for the deadline.

- Andreas:

I finished the user story USN-49 and did a lot of work on USN-50. We finished both the user stories in time and we verified that they worked.

- Joachim

Where able to upload the data to the database and therefore finished USN-51. I started working on figuring out how to pull data from the database using a REST API so the mobile app can receive the data about the bike location. stared putting more focus on the user story, and put in the final touch. Also were able to finish USN-35 of modelling the data that should be sent.

- Eebbaa

This week I worked on finding out how the Arduino board manages encryption during sending data. I also started USN-58 to find out some way to send a command to the Arduino for taking some action.

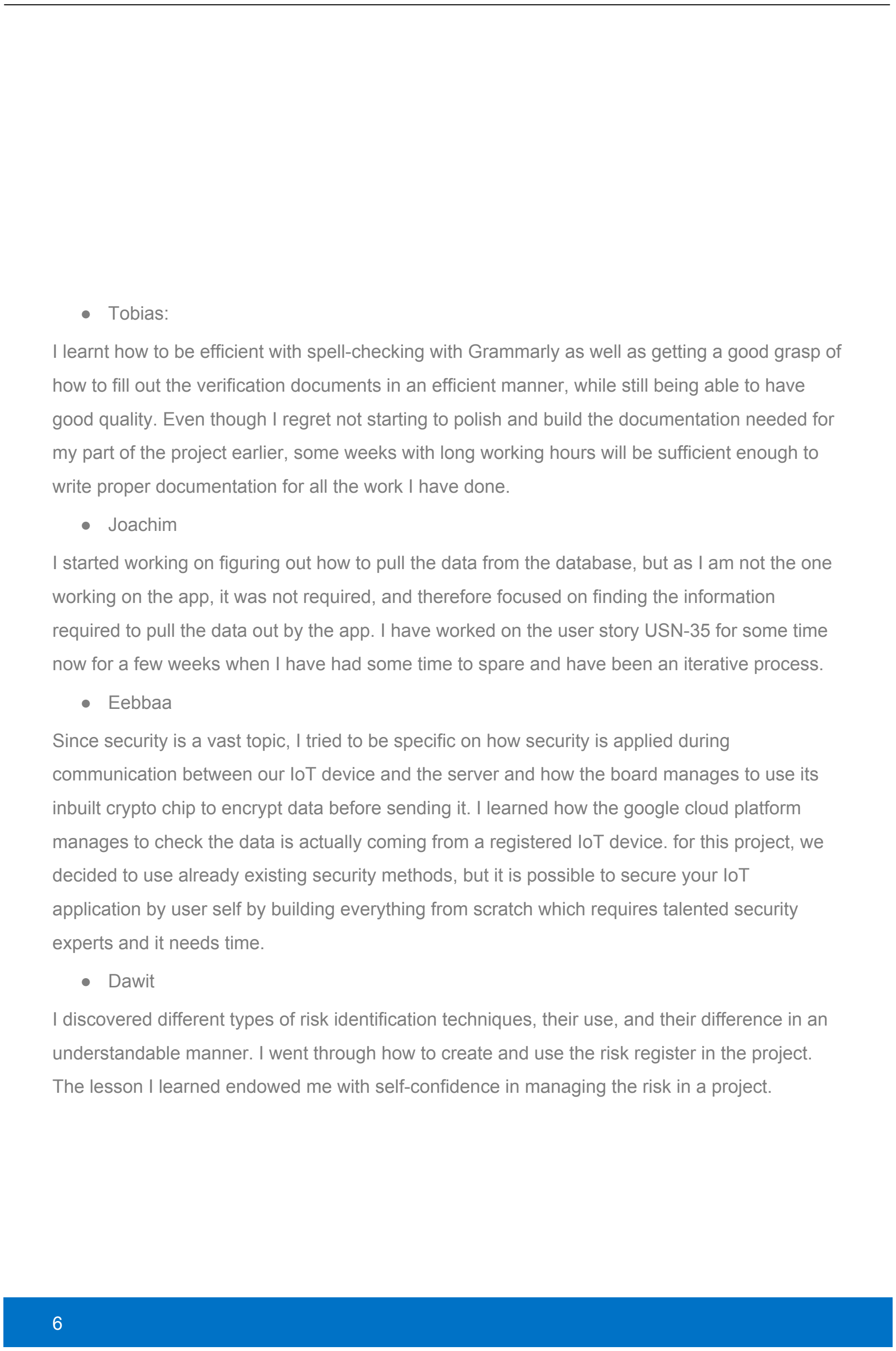
- Dawit

In this sprint, I worked on these two User stories, USN-60 and USN-61. Since managing the risk is vital for our project's success, we need to understand and find these techniques that support us in identifying risk. We also need the risk register for ease of managing the risk. I took the time and finished both USN-60 and USN-61.At last, about USN-59, I talked with Joachim to fix it together whenever we get time.

Sprint Retrospective

- Andreas:

I learned how to use UTC times in javascript and calculate time using milliseconds. Also learned about booking overlaps and what algorithms that can be used for checking it.



- Tobias:

I learnt how to be efficient with spell-checking with Grammarly as well as getting a good grasp of how to fill out the verification documents in an efficient manner, while still being able to have good quality. Even though I regret not starting to polish and build the documentation needed for my part of the project earlier, some weeks with long working hours will be sufficient enough to write proper documentation for all the work I have done.

- Joachim

I started working on figuring out how to pull the data from the database, but as I am not the one working on the app, it was not required, and therefore focused on finding the information required to pull the data out by the app. I have worked on the user story USN-35 for some time now for a few weeks when I have had some time to spare and have been an iterative process.

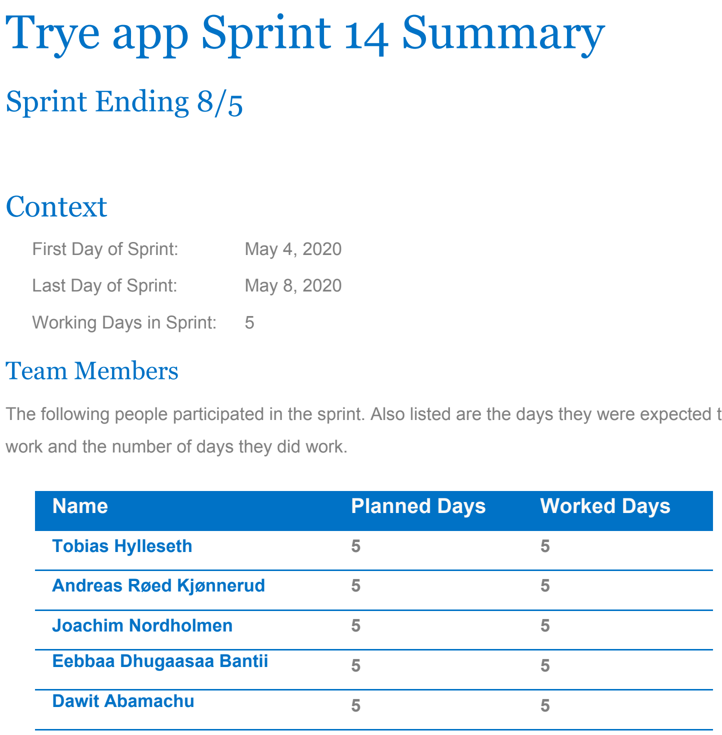
- Eebbaa

Since security is a vast topic, I tried to be specific on how security is applied during communication between our IoT device and the server and how the board manages to use its inbuilt crypto chip to encrypt data before sending it. I learned how the google cloud platform manages to check the data is actually coming from a registered IoT device. for this project, we decided to use already existing security methods, but it is possible to secure your IoT application by user self by building everything from scratch which requires talented security experts and it needs time.

- Dawit

I discovered different types of risk identification techniques, their use, and their difference in an understandable manner. I went through how to create and use the risk register in the project. The lesson I learned endowed me with self-confidence in managing the risk in a project.

A.1.14 Sprint 14



Trye app Sprint 14 Summary

Sprint Ending 8/5

Context

First Day of Sprint: May 4, 2020

Last Day of Sprint: May 8, 2020

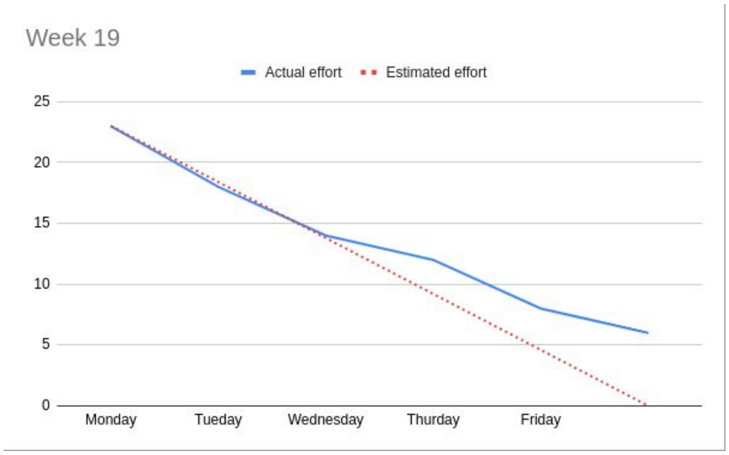
Working Days in Sprint: 5

Team Members

The following people participated in the sprint. Also listed are the days they were expected to work and the number of days they did work.

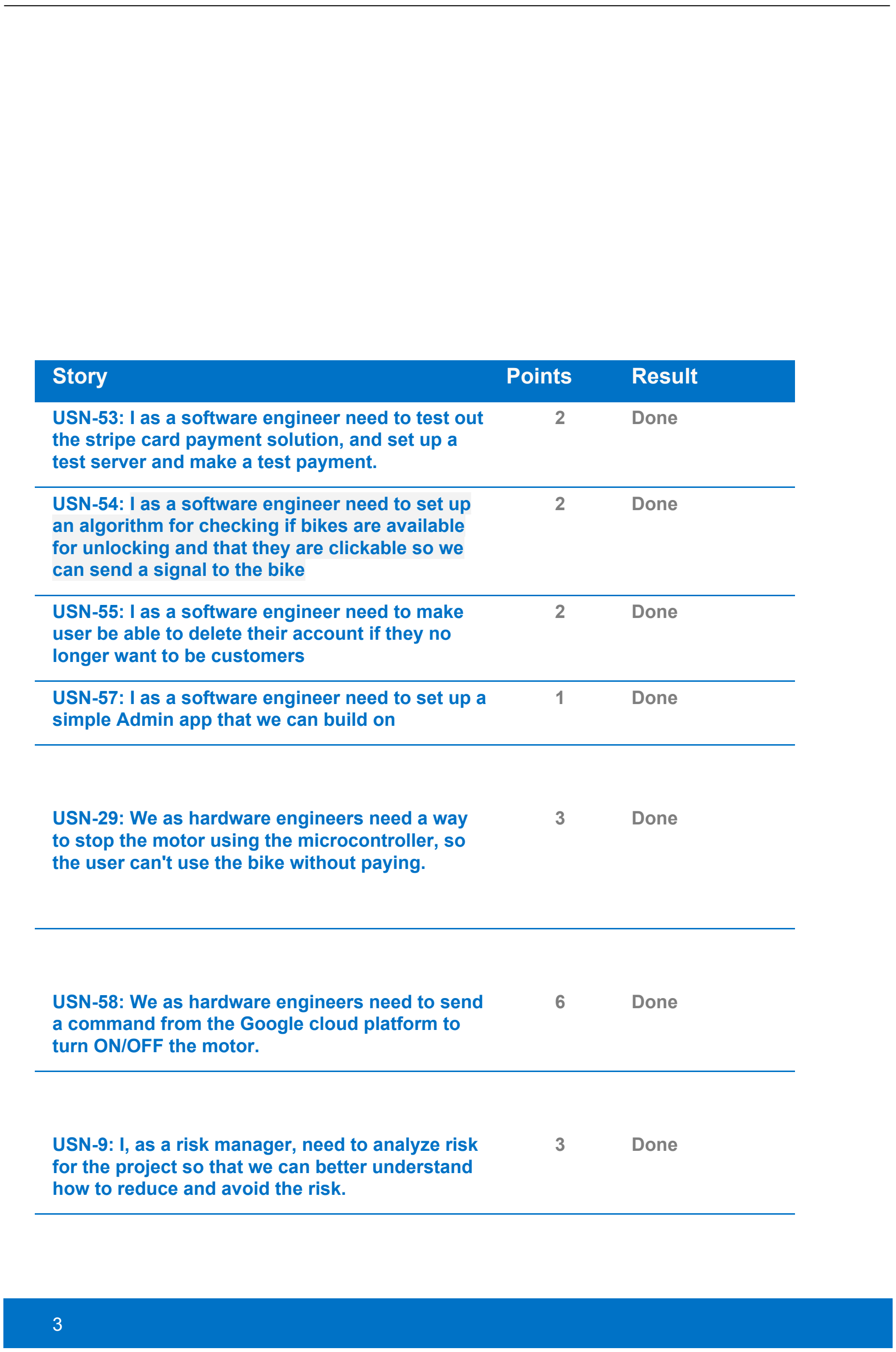
Name	Planned Days	Worked Days
Tobias Hylleseth	5	5
Andreas Røed Kjønnerud	5	5
Joachim Nordholmen	5	5
Eebbaa Dhugaasaa Bantii	5	5
Dawit Abamachu	5	5

Burndown chart



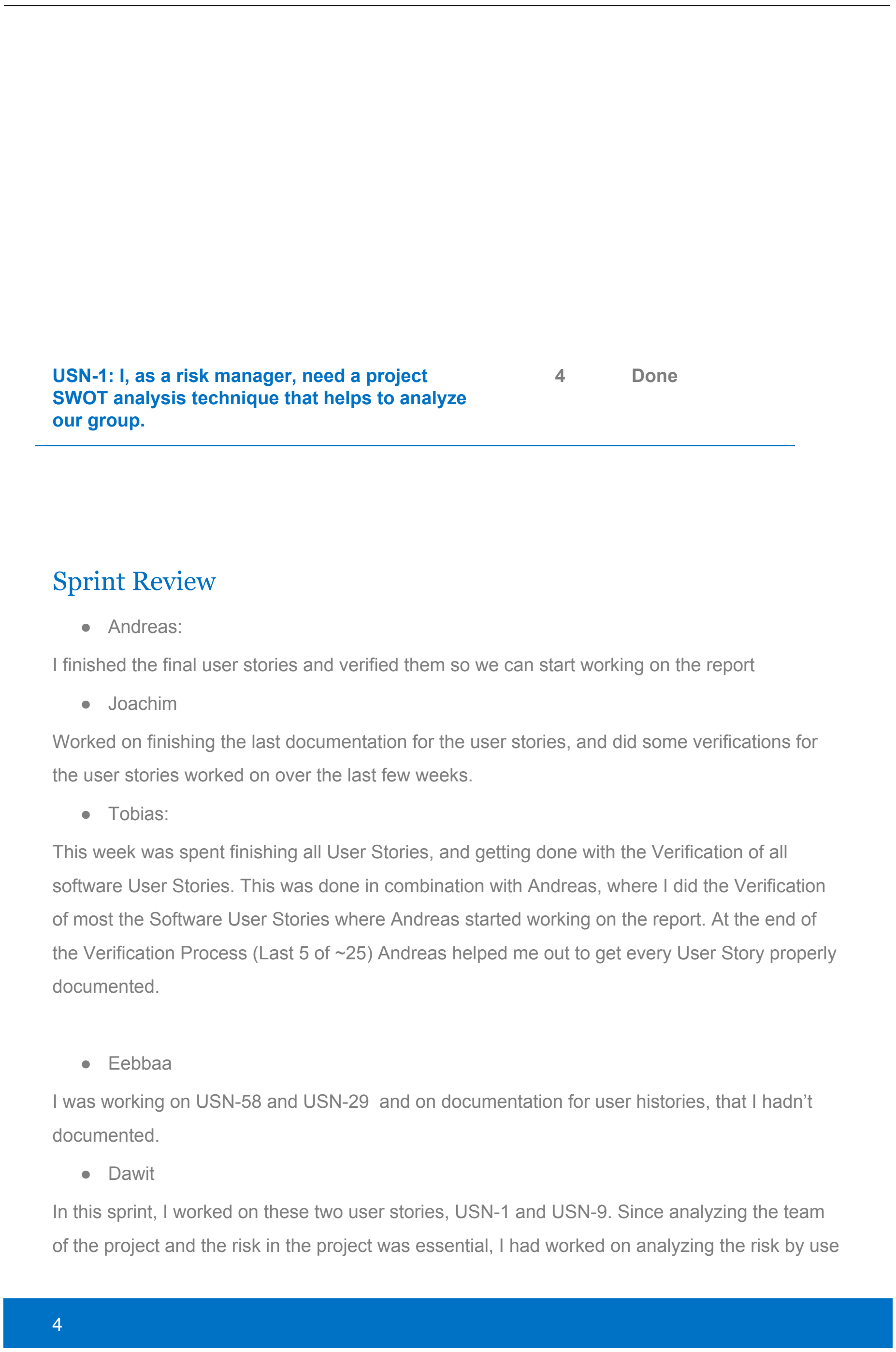
Contents and Assessment

Points Planned:	23
Points Earned:	17



Story	Points	Result
USN-53: I as a software engineer need to test out the stripe card payment solution, and set up a test server and make a test payment.	2	Done
USN-54: I as a software engineer need to set up an algorithm for checking if bikes are available for unlocking and that they are clickable so we can send a signal to the bike	2	Done
USN-55: I as a software engineer need to make user be able to delete their account if they no longer want to be customers	2	Done
USN-57: I as a software engineer need to set up a simple Admin app that we can build on	1	Done
USN-29: We as hardware engineers need a way to stop the motor using the microcontroller, so the user can't use the bike without paying.	3	Done
USN-58: We as hardware engineers need to send a command from the Google cloud platform to turn ON/OFF the motor.	6	Done
USN-9: I, as a risk manager, need to analyze risk for the project so that we can better understand how to reduce and avoid the risk.	3	Done

3



USN-1: I, as a risk manager, need a project SWOT analysis technique that helps to analyze our group.	**4**	**Done**

Sprint Review

- Andreas:

I finished the final user stories and verified them so we can start working on the report

- Joachim

Worked on finishing the last documentation for the user stories, and did some verifications for the user stories worked on over the last few weeks.

- Tobias:

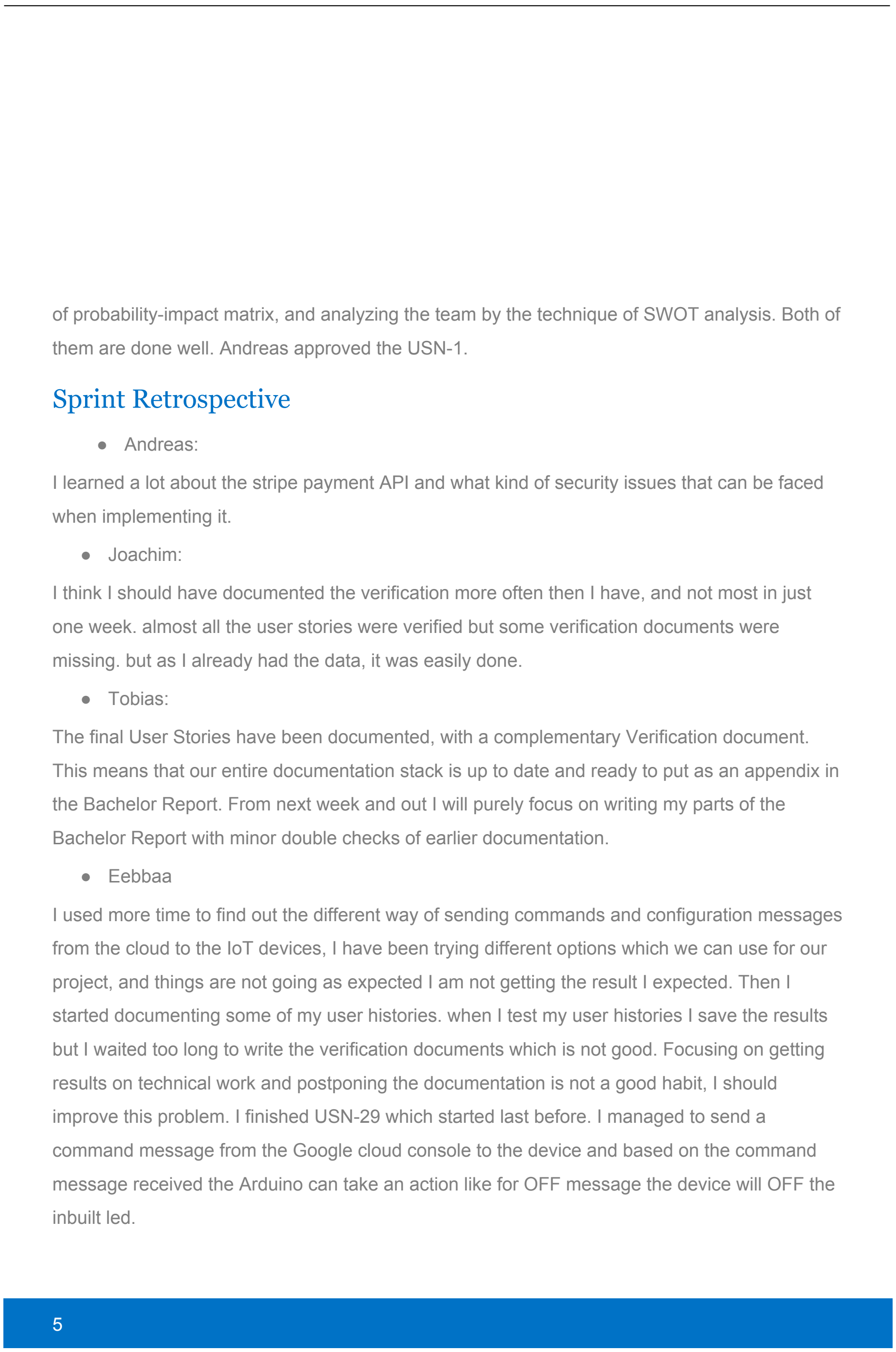
This week was spent finishing all User Stories, and getting done with the Verification of all software User Stories. This was done in combination with Andreas, where I did the Verification of most the Software User Stories where Andreas started working on the report. At the end of the Verification Process (Last 5 of ~25) Andreas helped me out to get every User Story properly documented.

- Eebbaa

I was working on USN-58 and USN-29 and on documentation for user histories, that I hadn't documented.

- Dawit

In this sprint, I worked on these two user stories, USN-1 and USN-9. Since analyzing the team of the project and the risk in the project was essential, I had worked on analyzing the risk by use



of probability-impact matrix, and analyzing the team by the technique of SWOT analysis. Both of them are done well. Andreas approved the USN-1.

Sprint Retrospective

- Andreas:

I learned a lot about the stripe payment API and what kind of security issues that can be faced when implementing it.

- Joachim:

I think I should have documented the verification more often then I have, and not most in just one week. almost all the user stories were verified but some verification documents were missing. but as I already had the data, it was easily done.

- Tobias:

The final User Stories have been documented, with a complementary Verification document. This means that our entire documentation stack is up to date and ready to put as an appendix in the Bachelor Report. From next week and out I will purely focus on writing my parts of the Bachelor Report with minor double checks of earlier documentation.

- Eebbaa

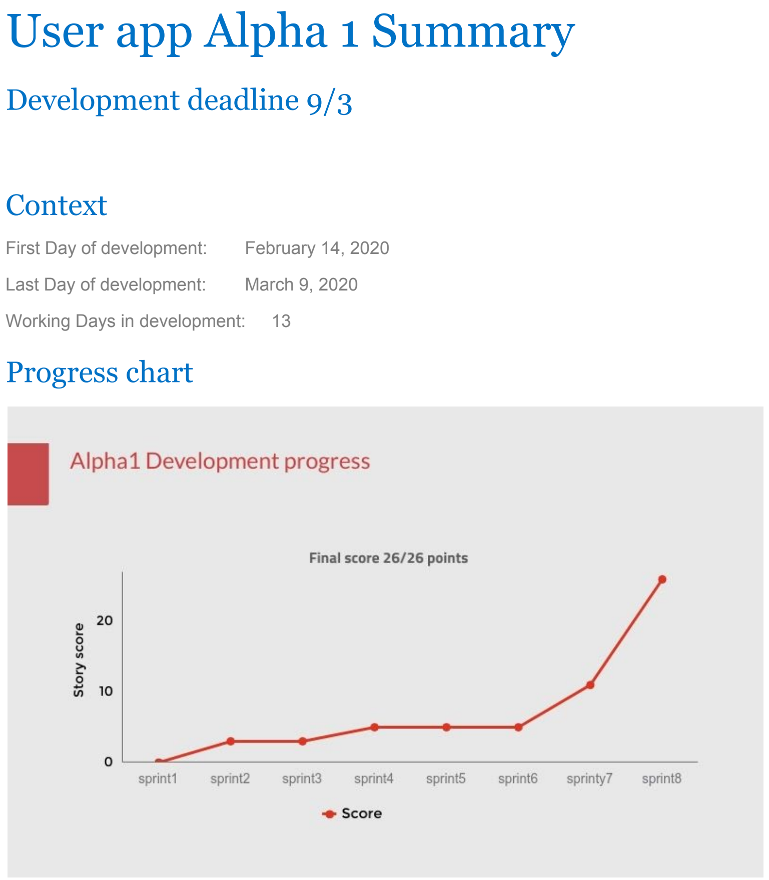
I used more time to find out the different way of sending commands and configuration messages from the cloud to the IoT devices, I have been trying different options which we can use for our project, and things are not going as expected I am not getting the result I expected. Then I started documenting some of my user histories. when I test my user histories I save the results but I waited too long to write the verification documents which is not good. Focusing on getting results on technical work and postponing the documentation is not a good habit, I should improve this problem. I finished USN-29 which started last before. I managed to send a command message from the Google cloud console to the device and based on the command message received the Arduino can take an action like for OFF message the device will OFF the inbuilt led.

- Dawit

I learned a lot while analyzing the risk. Following the work of risk analysis, I fully understood these risks, which needed close attention and less attention. I also learned the mitigation actions to respond to the risk. Without the management of the risk in the project, the project is at risk.

A.2 Version reports

A.2.1 User Alpha Version 1



User app Alpha 1 Summary

Development deadline 9/3

Context

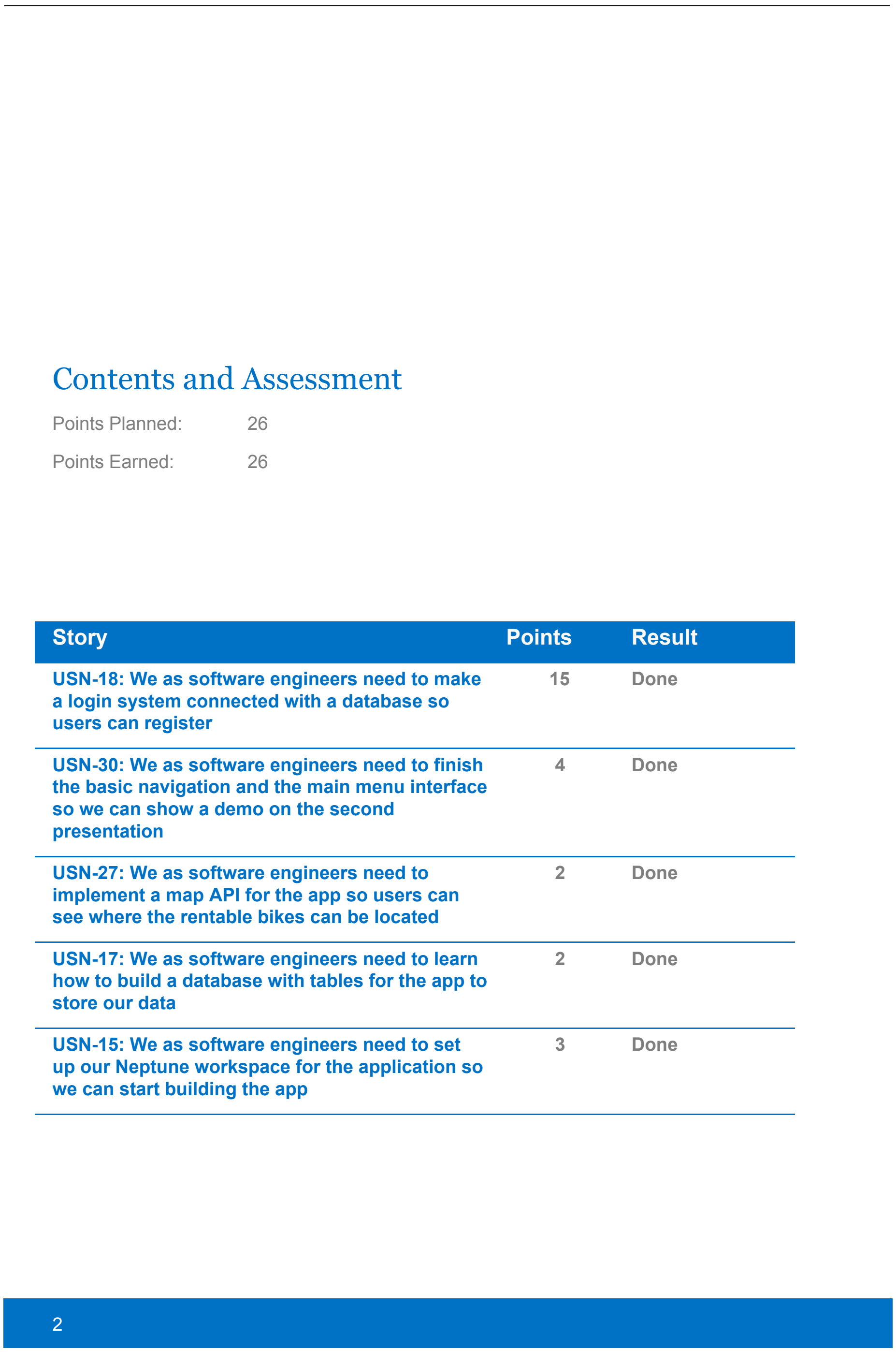
First Day of development: February 14, 2020

Last Day of development: March 9, 2020

Working Days in development: 13

Progress chart

Alpha1 Development progress
Final score 26/26 points
Story score
20
10
0
sprint1
sprint2
sprint3
sprint4
sprint5
sprint6
sprinty7
sprint8
Score



Contents and Assessment

Points Planned:	26
Points Earned:	26

Story	Points	Result
USN-18: We as software engineers need to make a login system connected with a database so users can register	15	Done
USN-30: We as software engineers need to finish the basic navigation and the main menu interface so we can show a demo on the second presentation	4	Done
USN-27: We as software engineers need to implement a map API for the app so users can see where the rentable bikes can be located	2	Done
USN-17: We as software engineers need to learn how to build a database with tables for the app to store our data	2	Done
USN-15: We as software engineers need to set up our Neptune workspace for the application so we can start building the app	3	Done

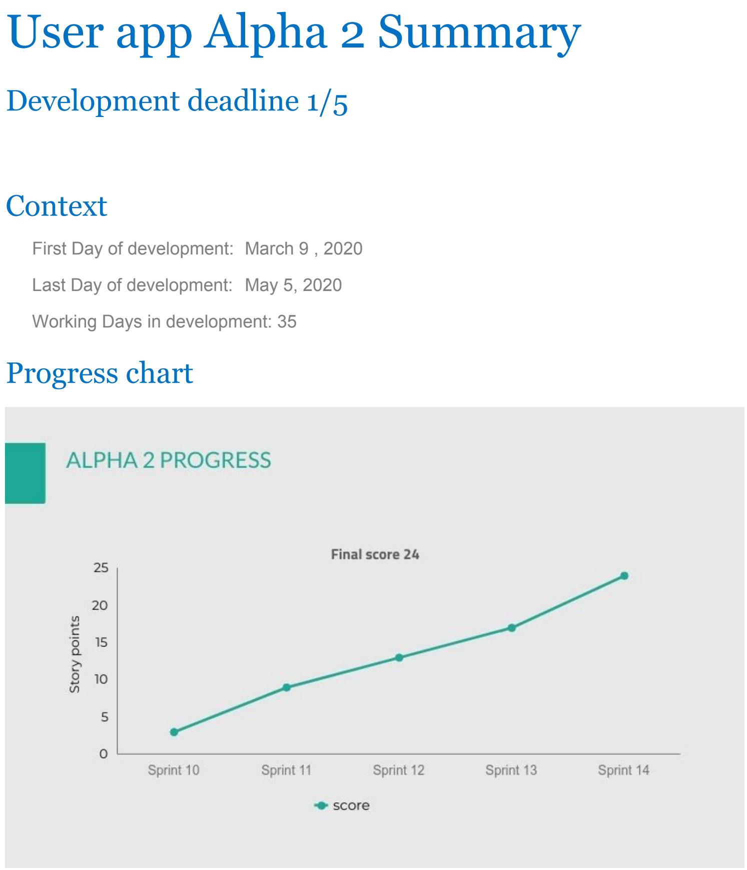
Alpha 1 Review

- The Alpha version works as intended and we and our employers from TRYE both like the prototype and how far we have come in such a short amount of time
- This Alpha version was developed quick without any planning so that we could develop faster. We made this Alpha because the guys from TRYE wanted to see progress. But for the next Alpha, we will do more of the required planning including UML and database planning

Alpha 1 Retrospective

- We have learned a lot about using planet9 and its tools with the app studio, APIs and the database. We will put this to good use when we start developing the next alpha which we want to finish in April.
- We have learned about the syntax of javascript language which will help us with faster development later.

A.2.2 User Alpha Version 2



User app Alpha 2 Summary

Development deadline 1/5

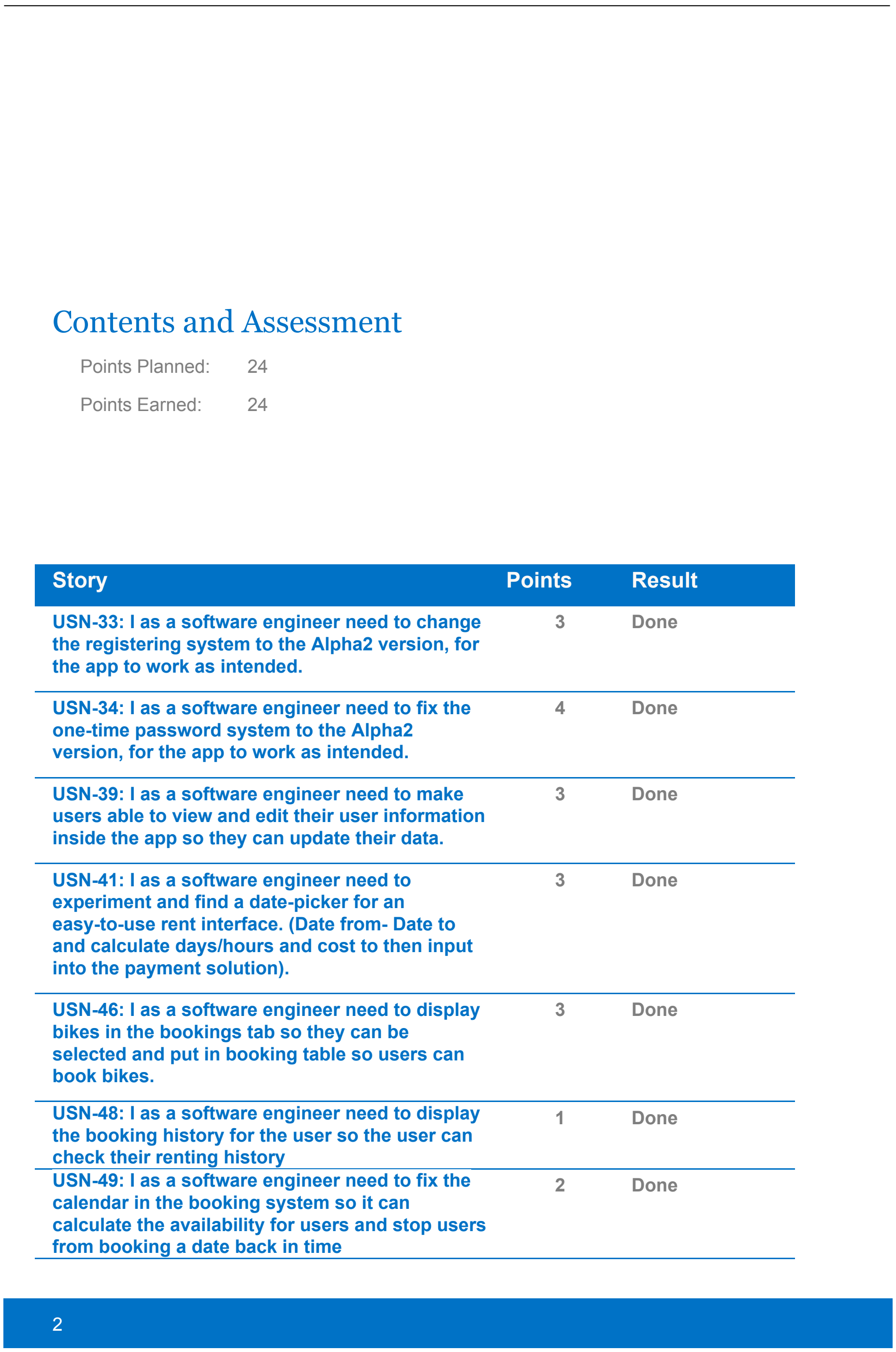
Context

First Day of development: March 9 , 2020

Last Day of development: May 5, 2020

Working Days in development: 35

Progress chart



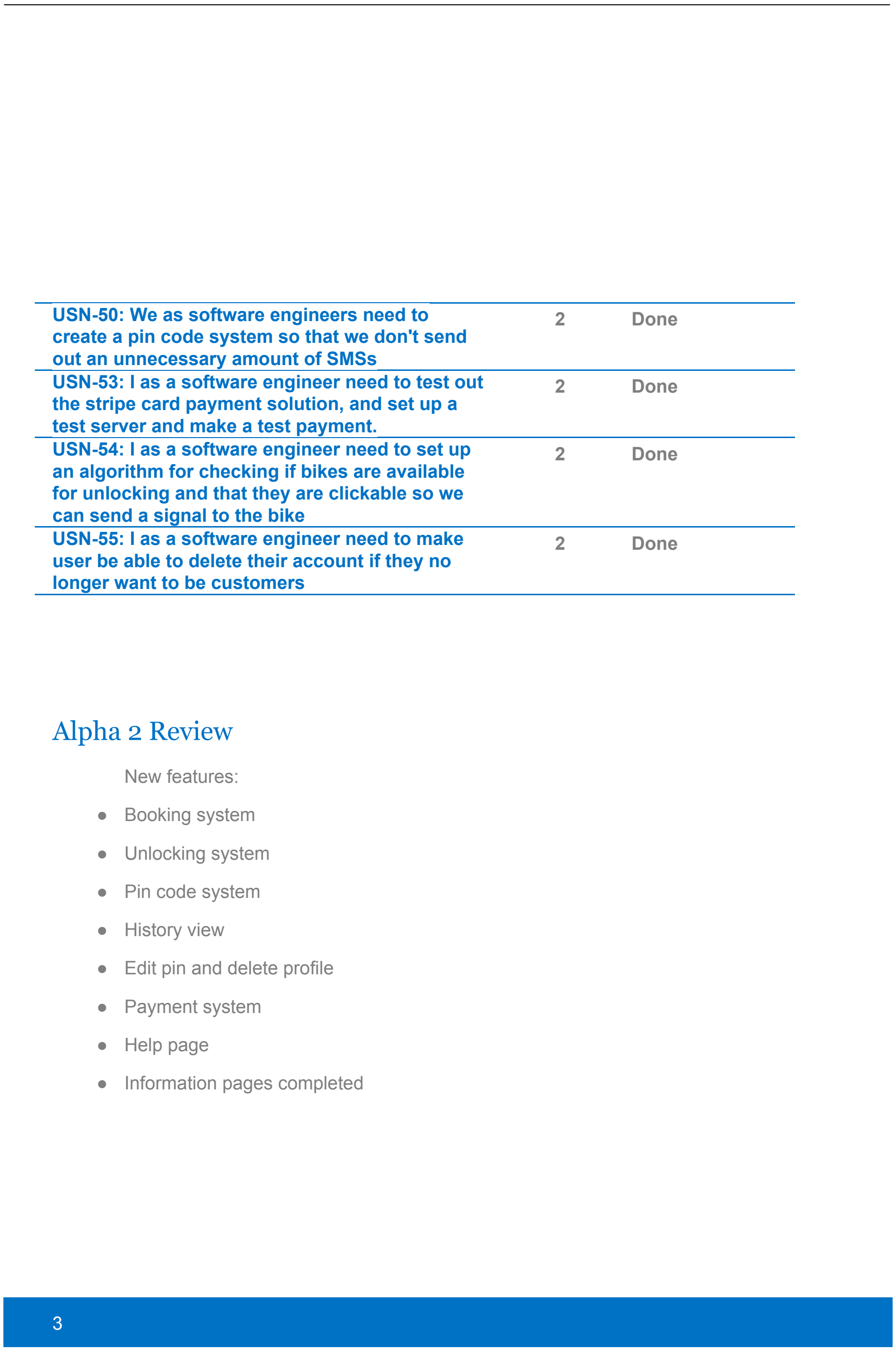
Contents and Assessment

Points Planned: 24

Points Earned: 24

Story	Points	Result
USN-33: I as a software engineer need to change the registering system to the Alpha2 version, for the app to work as intended.	3	Done
USN-34: I as a software engineer need to fix the one-time password system to the Alpha2 version, for the app to work as intended.	4	Done
USN-39: I as a software engineer need to make users able to view and edit their user information inside the app so they can update their data.	3	Done
USN-41: I as a software engineer need to experiment and find a date-picker for an easy-to-use rent interface. (Date from- Date to and calculate days/hours and cost to then input into the payment solution).	3	Done
USN-46: I as a software engineer need to display bikes in the bookings tab so they can be selected and put in booking table so users can book bikes.	3	Done
USN-48: I as a software engineer need to display the booking history for the user so the user can check their renting history	1	Done
USN-49: I as a software engineer need to fix the calendar in the booking system so it can calculate the availability for users and stop users from booking a date back in time	2	Done

2

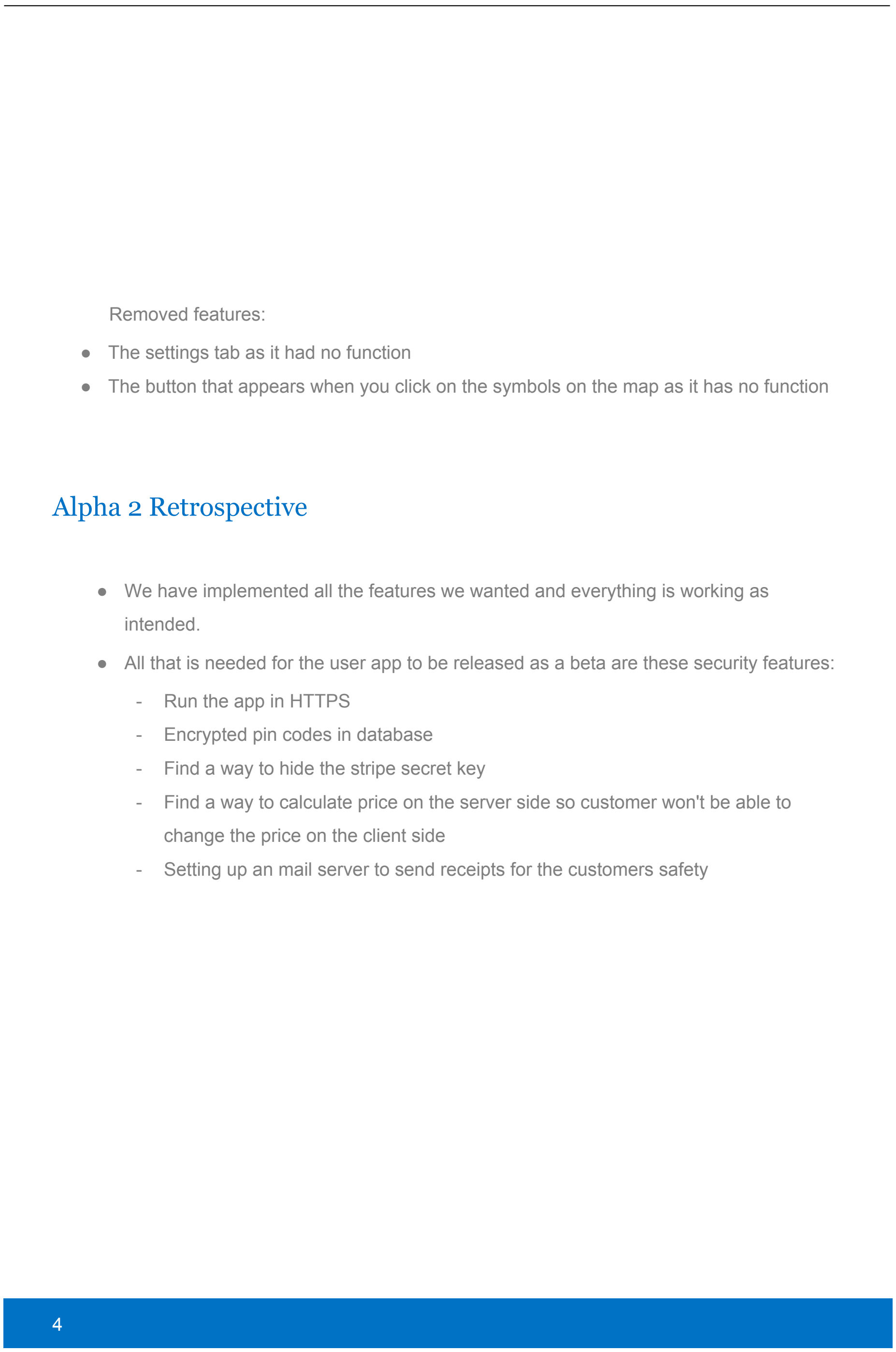


USN-50: We as software engineers need to create a pin code system so that we don't send out an unnecessary amount of SMSs	**2**	**Done**
USN-53: I as a software engineer need to test out the stripe card payment solution, and set up a test server and make a test payment.	**2**	**Done**
USN-54: I as a software engineer need to set up an algorithm for checking if bikes are available for unlocking and that they are clickable so we can send a signal to the bike	**2**	**Done**
USN-55: I as a software engineer need to make user be able to delete their account if they no longer want to be customers	**2**	**Done**

Alpha 2 Review

New features:

- Booking system
- Unlocking system
- Pin code system
- History view
- Edit pin and delete profile
- Payment system
- Help page
- Information pages completed



Removed features:

- The settings tab as it had no function
- The button that appears when you click on the symbols on the map as it has no function

Alpha 2 Retrospective

- We have implemented all the features we wanted and everything is working as intended.
- All that is needed for the user app to be released as a beta are these security features:
  - Run the app in HTTPS
  - Encrypted pin codes in database
  - Find a way to hide the stripe secret key
  - Find a way to calculate price on the server side so customer won't be able to change the price on the client side
  - Setting up an mail server to send receipts for the customers safety

A.2.3 Admin Alpha Version 1

Admin alpha 1 Summary

Development deadline 1/5

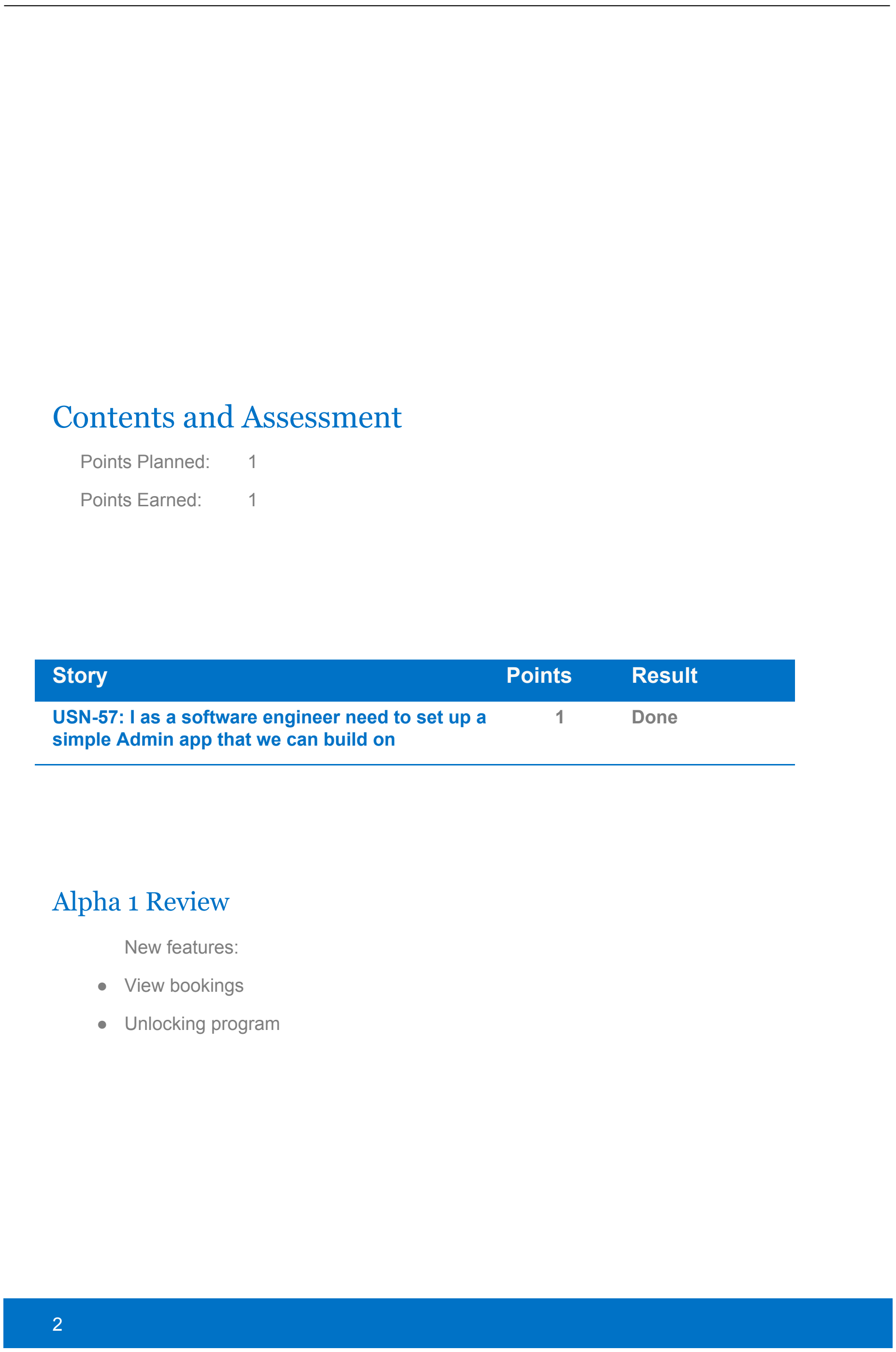
Context

First Day of development: May 4 , 2020

Last Day of development: May 5, 2020

Working Days in development: 2

The development included only one user story so it is no point in a graph.



Contents and Assessment

Points Planned: 1

Points Earned: 1

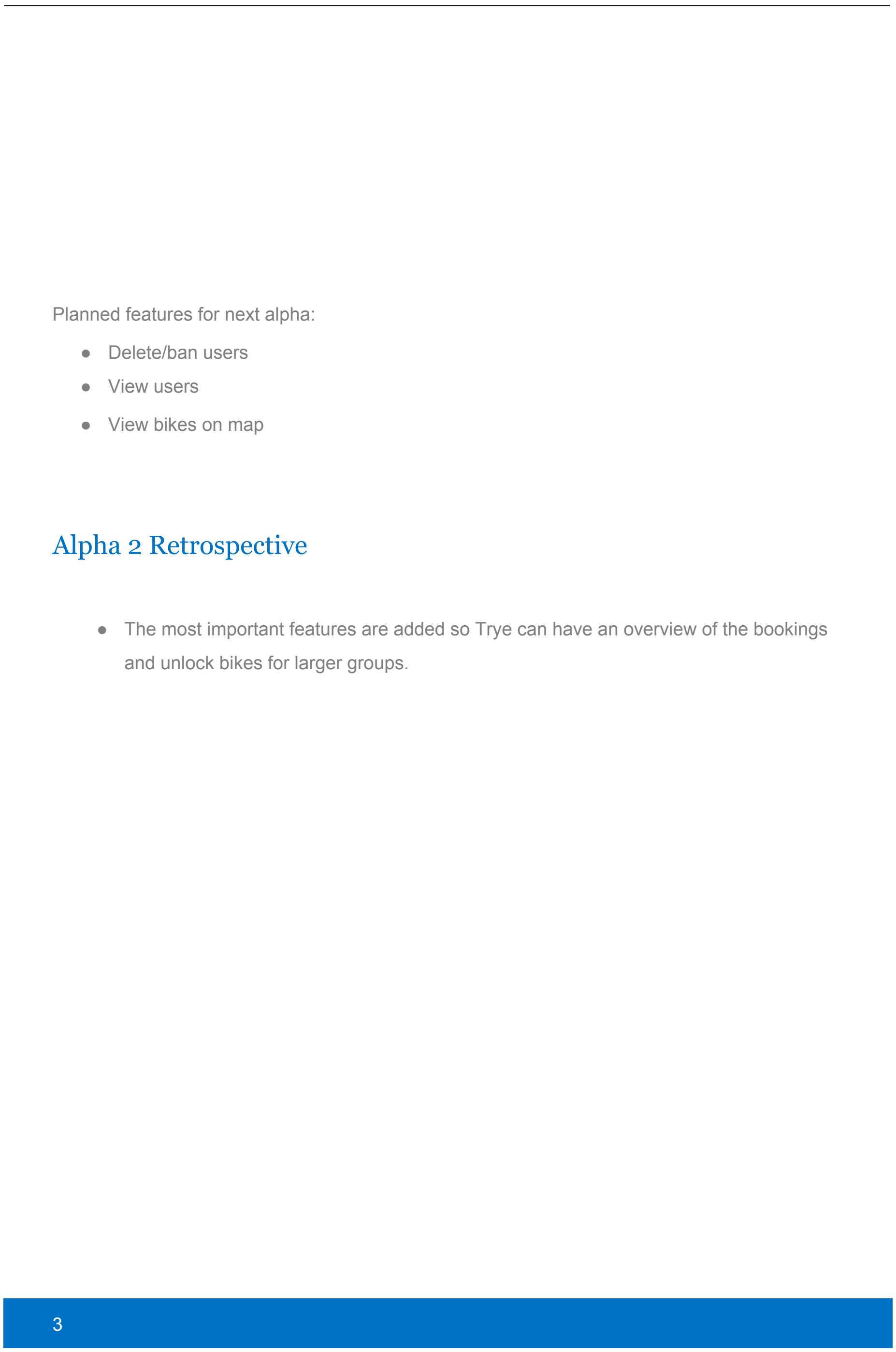
Story	Points	Result
USN-57: I as a software engineer need to set up a simple Admin app that we can build on	1	Done

Alpha 1 Review

New features:

- View bookings
- Unlocking program

2



Planned features for next alpha:

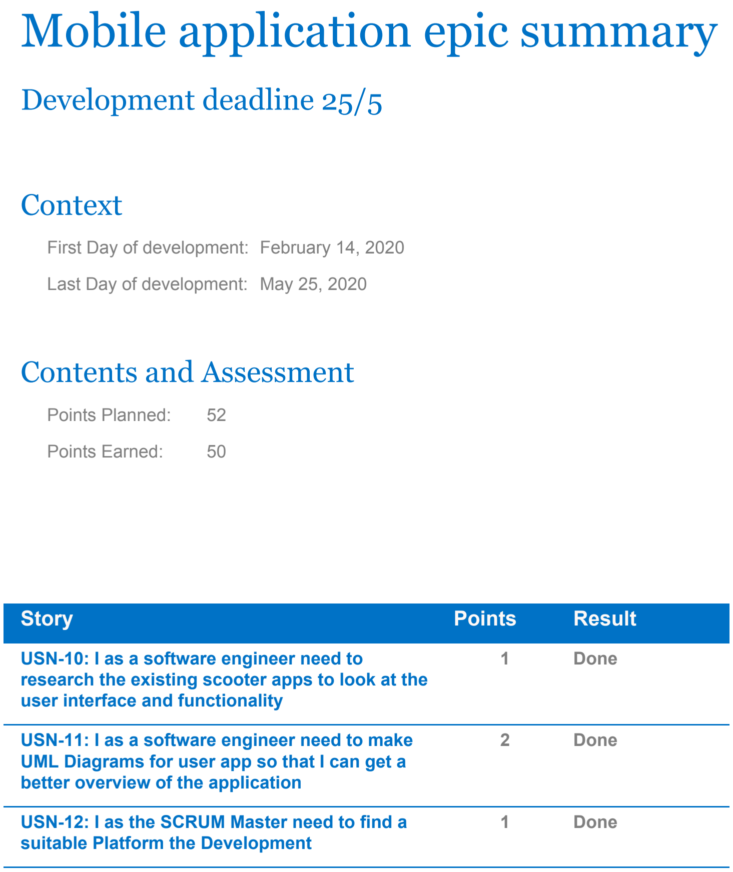
- Delete/ban users
- View users
- View bikes on map

Alpha 2 Retrospective

- The most important features are added so Trye can have an overview of the bookings and unlock bikes for larger groups.

A.3 Epic reports

A.3.1 MA epic



Mobile application epic summary

Development deadline 25/5

Context

First Day of development: February 14, 2020

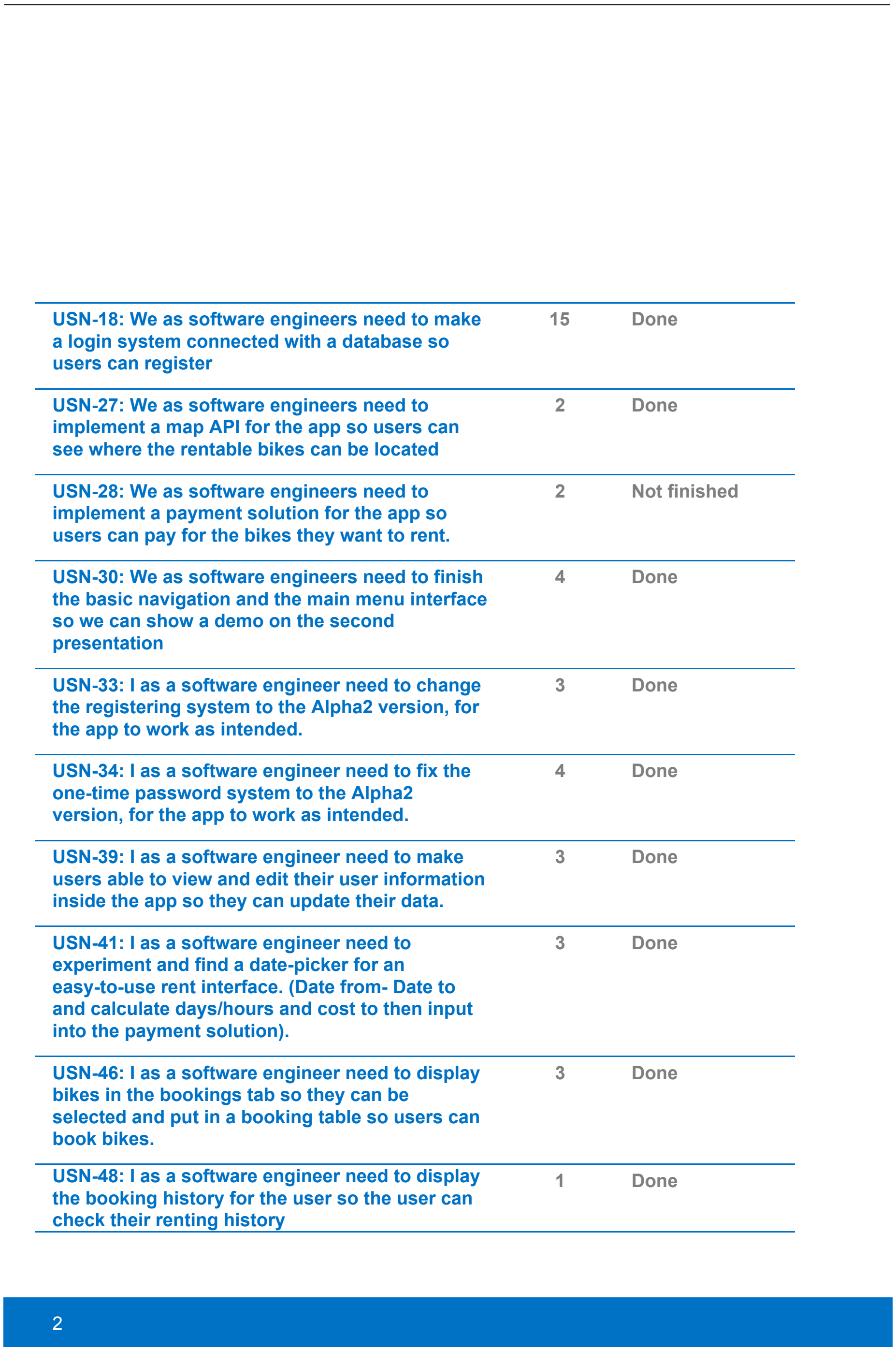
Last Day of development: May 25, 2020

Contents and Assessment

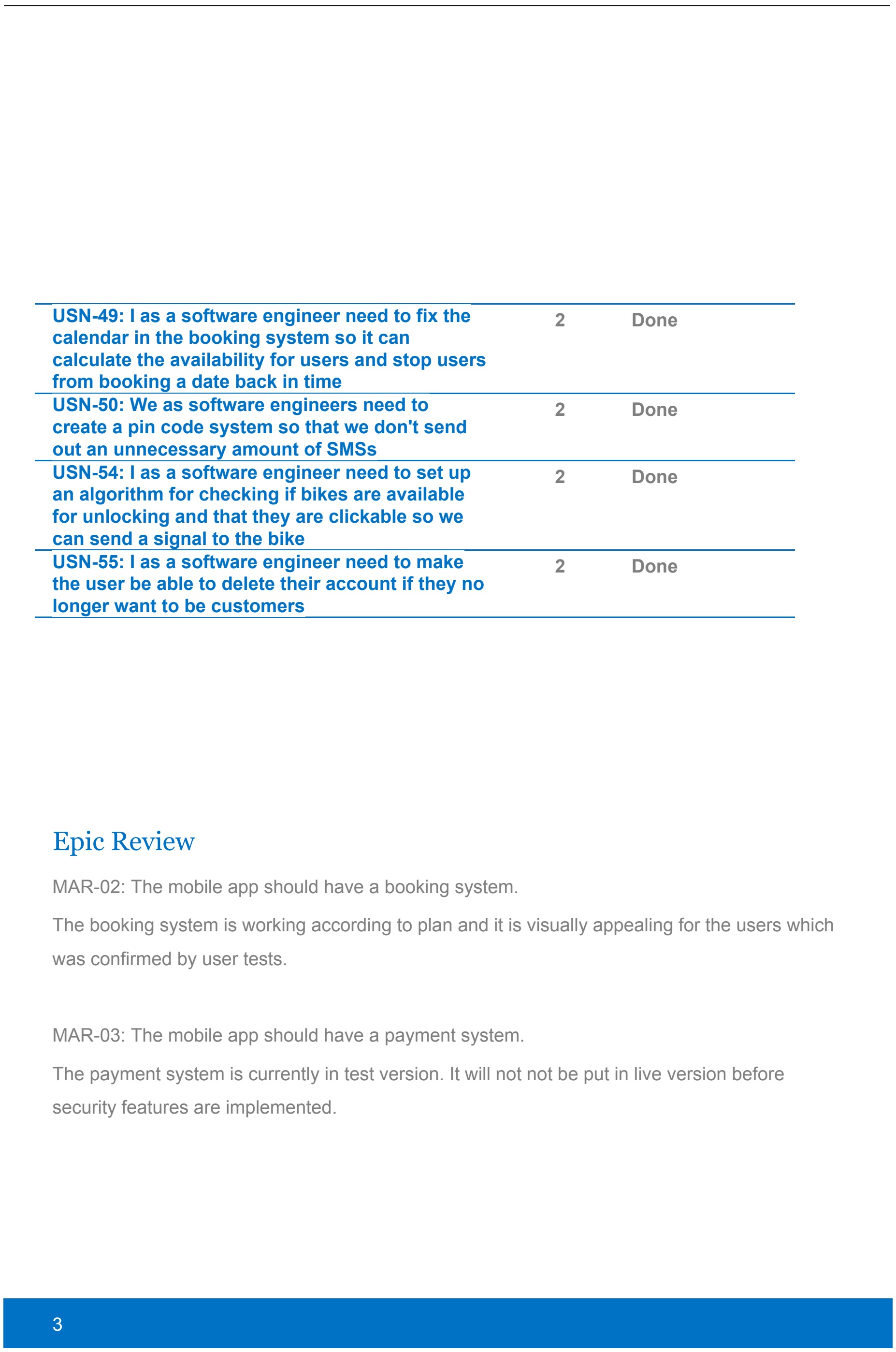
Points Planned: 52

Points Earned: 50

Story	Points	Result
USN-10: I as a software engineer need to research the existing scooter apps to look at the user interface and functionality	1	Done
USN-11: I as a software engineer need to make UML Diagrams for user app so that I can get a better overview of the application	2	Done
USN-12: I as the SCRUM Master need to find a suitable Platform the Development	1	Done



USN-18: We as software engineers need to make a login system connected with a database so users can register	15	Done
USN-27: We as software engineers need to implement a map API for the app so users can see where the rentable bikes can be located	2	Done
USN-28: We as software engineers need to implement a payment solution for the app so users can pay for the bikes they want to rent.	2	Not finished
USN-30: We as software engineers need to finish the basic navigation and the main menu interface so we can show a demo on the second presentation	4	Done
USN-33: I as a software engineer need to change the registering system to the Alpha2 version, for the app to work as intended.	3	Done
USN-34: I as a software engineer need to fix the one-time password system to the Alpha2 version, for the app to work as intended.	4	Done
USN-39: I as a software engineer need to make users able to view and edit their user information inside the app so they can update their data.	3	Done
USN-41: I as a software engineer need to experiment and find a date-picker for an easy-to-use rent interface. (Date from- Date to and calculate days/hours and cost to then input into the payment solution).	3	Done
USN-46: I as a software engineer need to display bikes in the bookings tab so they can be selected and put in a booking table so users can book bikes.	3	Done
USN-48: I as a software engineer need to display the booking history for the user so the user can check their renting history	1	Done



USN-49: I as a software engineer need to fix the calendar in the booking system so it can calculate the availability for users and stop users from booking a date back in time	**2**	**Done**
USN-50: We as software engineers need to create a pin code system so that we don't send out an unnecessary amount of SMSs	**2**	**Done**
USN-54: I as a software engineer need to set up an algorithm for checking if bikes are available for unlocking and that they are clickable so we can send a signal to the bike	**2**	**Done**
USN-55: I as a software engineer need to make the user be able to delete their account if they no longer want to be customers	**2**	**Done**

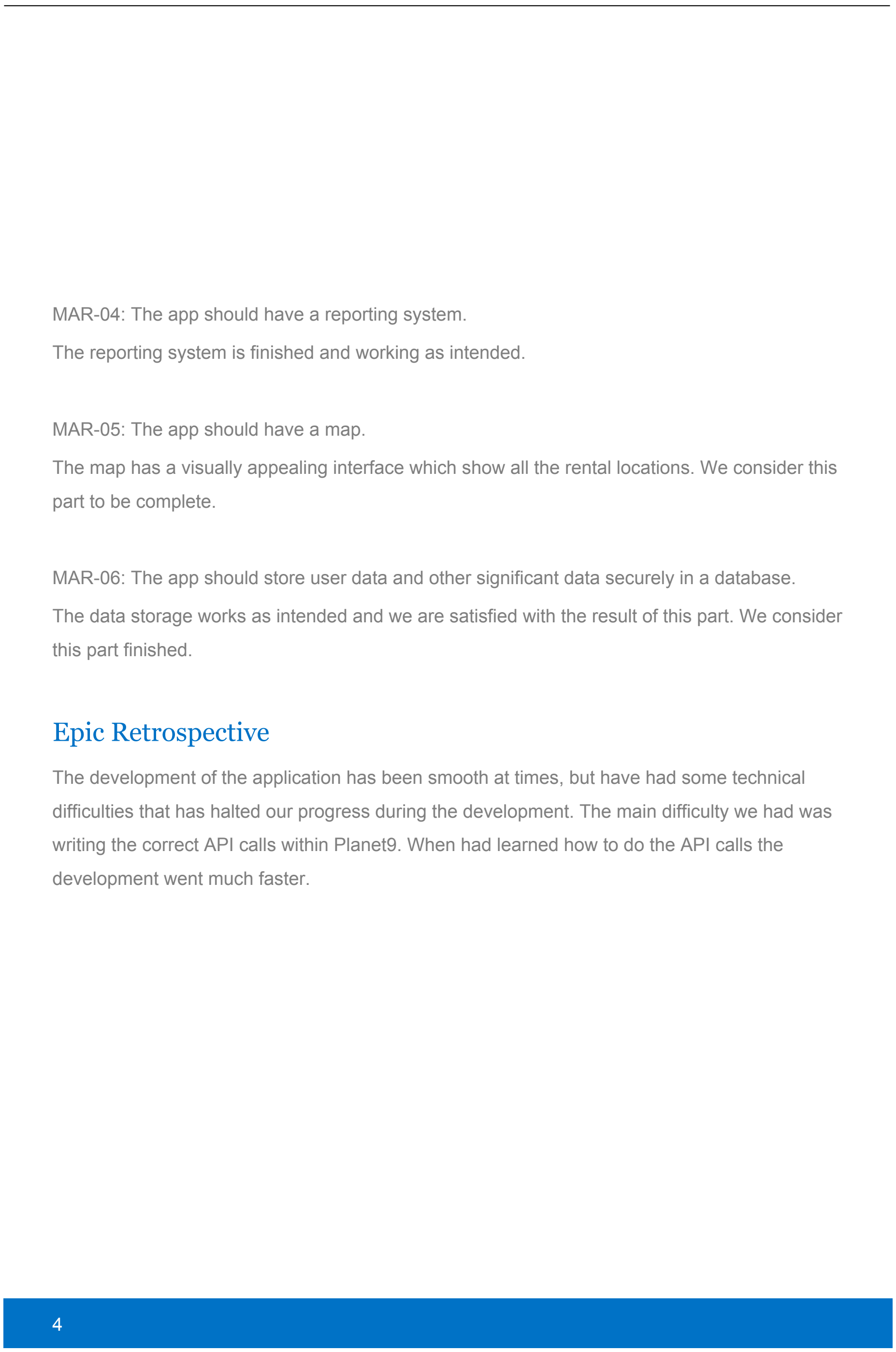
Epic Review

MAR-02: The mobile app should have a booking system.

The booking system is working according to plan and it is visually appealing for the users which was confirmed by user tests.

MAR-03: The mobile app should have a payment system.

The payment system is currently in test version. It will not not be put in live version before security features are implemented.



MAR-04: The app should have a reporting system.

The reporting system is finished and working as intended.

MAR-05: The app should have a map.

The map has a visually appealing interface which show all the rental locations. We consider this part to be complete.

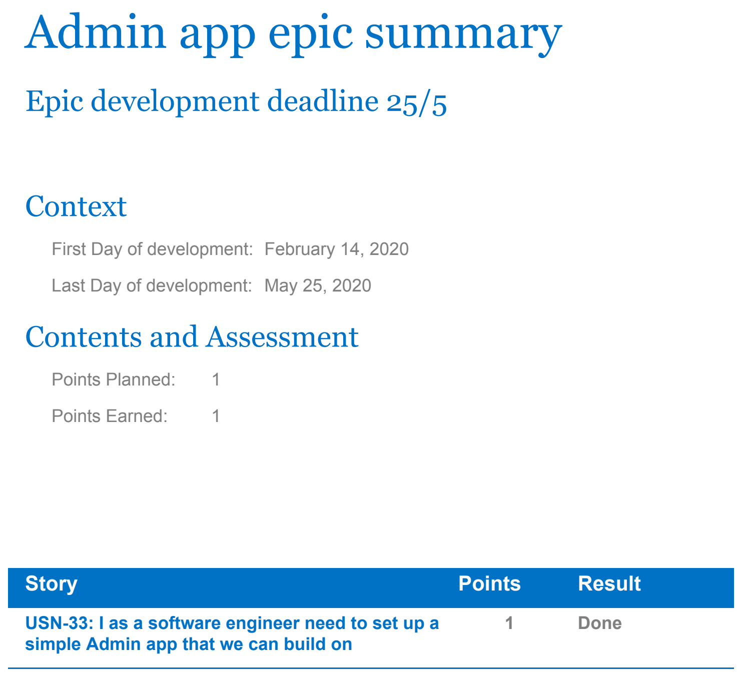
MAR-06: The app should store user data and other significant data securely in a database.

The data storage works as intended and we are satisfied with the result of this part. We consider this part finished.

Epic Retrospective

The development of the application has been smooth at times, but have had some technical difficulties that has halted our progress during the development. The main difficulty we had was writing the correct API calls within Planet9. When had learned how to do the API calls the development went much faster.

A.3.2 AS epic



Admin app epic summary

Epic development deadline 25/5

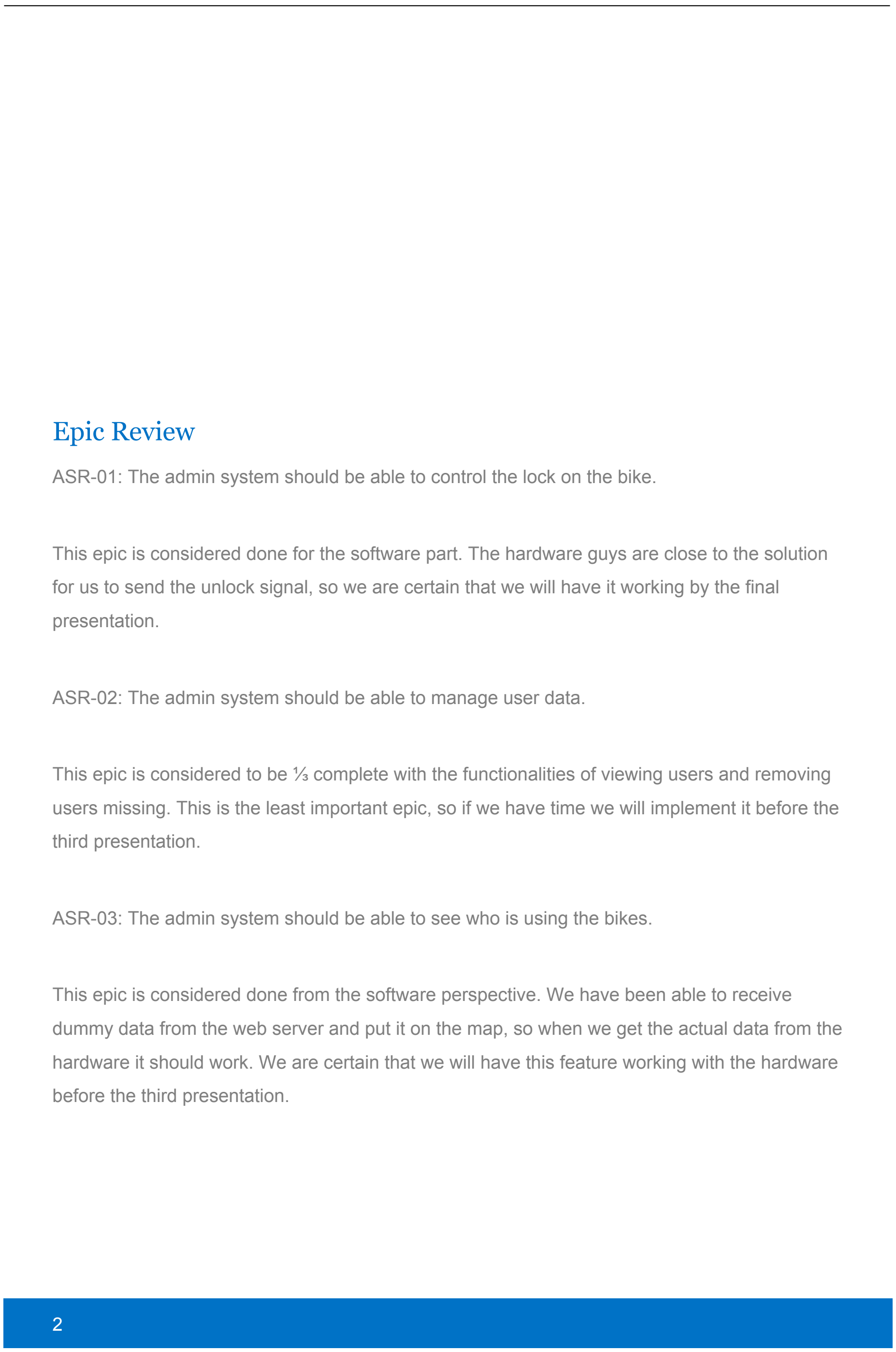
Context

First Day of development:	February 14, 2020
Last Day of development:	May 25, 2020

Contents and Assessment

Points Planned:	1
Points Earned:	1

Story	Points	Result
USN-33: I as a software engineer need to set up a simple Admin app that we can build on	1	Done



Epic Review

ASR-01: The admin system should be able to control the lock on the bike.

This epic is considered done for the software part. The hardware guys are close to the solution for us to send the unlock signal, so we are certain that we will have it working by the final presentation.

ASR-02: The admin system should be able to manage user data.

This epic is considered to be ⅓ complete with the functionalities of viewing users and removing users missing. This is the least important epic, so if we have time we will implement it before the third presentation.

ASR-03: The admin system should be able to see who is using the bikes.

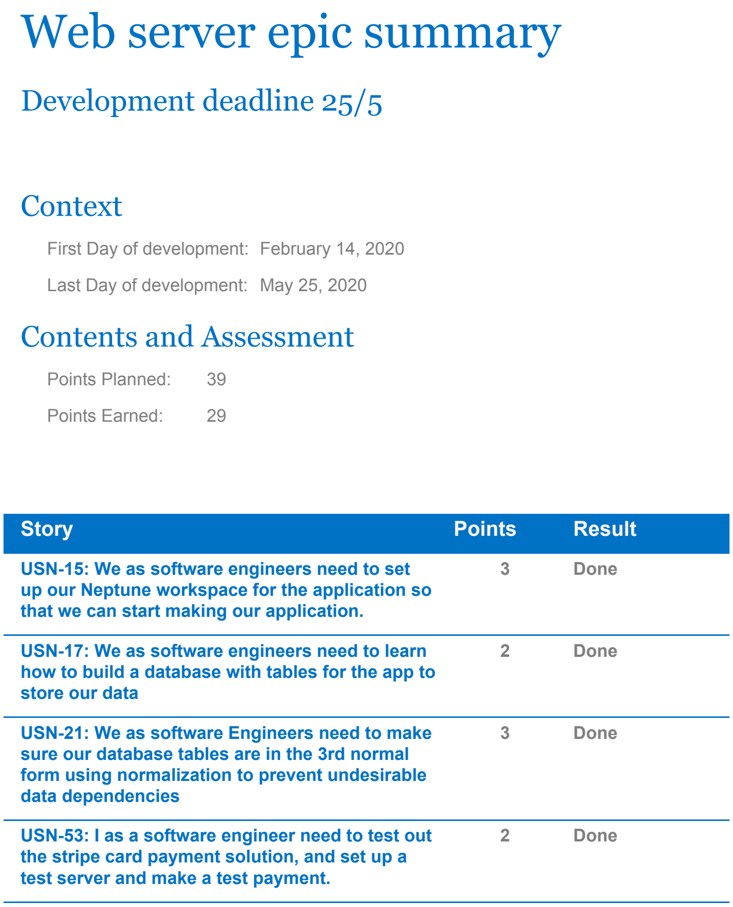
This epic is considered done from the software perspective. We have been able to receive dummy data from the web server and put it on the map, so when we get the actual data from the hardware it should work. We are certain that we will have this feature working with the hardware before the third presentation.

Epic Retrospective

This epic has only 1 point but has been worked on for a total of 4 days since we worked on communication with the hardware system the last days before delivering the final report. This means the real workload is about 4 points.

The epic was not as hard to complete as we had initially thought but that is since we learned much from our mobile application development.

A.3.3 WS epic



Web server epic summary

Development deadline 25/5

Context

First Day of development: February 14, 2020

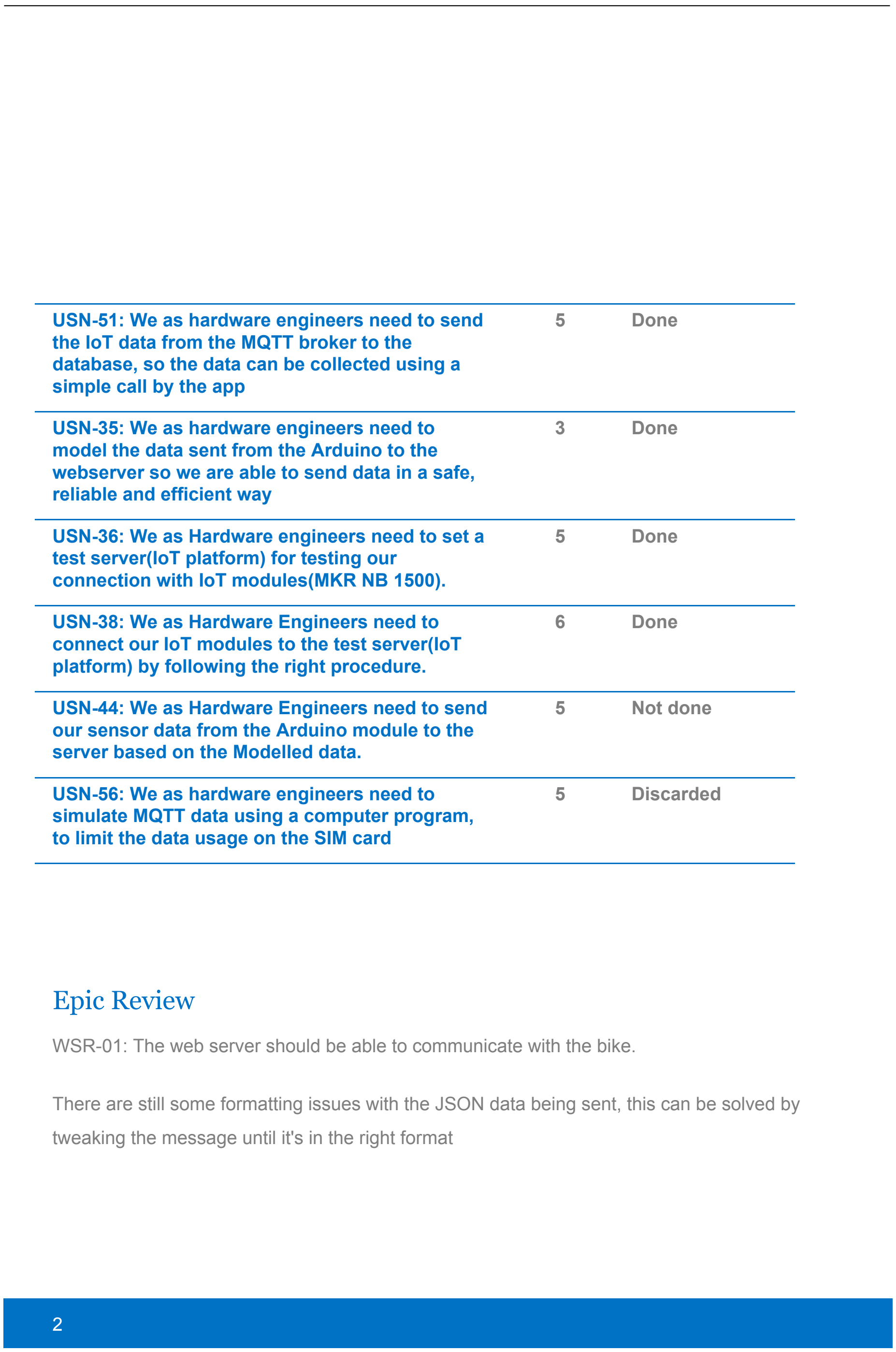
Last Day of development: May 25, 2020

Contents and Assessment

Points Planned: 39

Points Earned: 29

Story	Points	Result
USN-15: We as software engineers need to set up our Neptune workspace for the application so that we can start making our application.	3	Done
USN-17: We as software engineers need to learn how to build a database with tables for the app to store our data	2	Done
USN-21: We as software Engineers need to make sure our database tables are in the 3rd normal form using normalization to prevent undesirable data dependencies	3	Done
USN-53: I as a software engineer need to test out the stripe card payment solution, and set up a test server and make a test payment.	2	Done

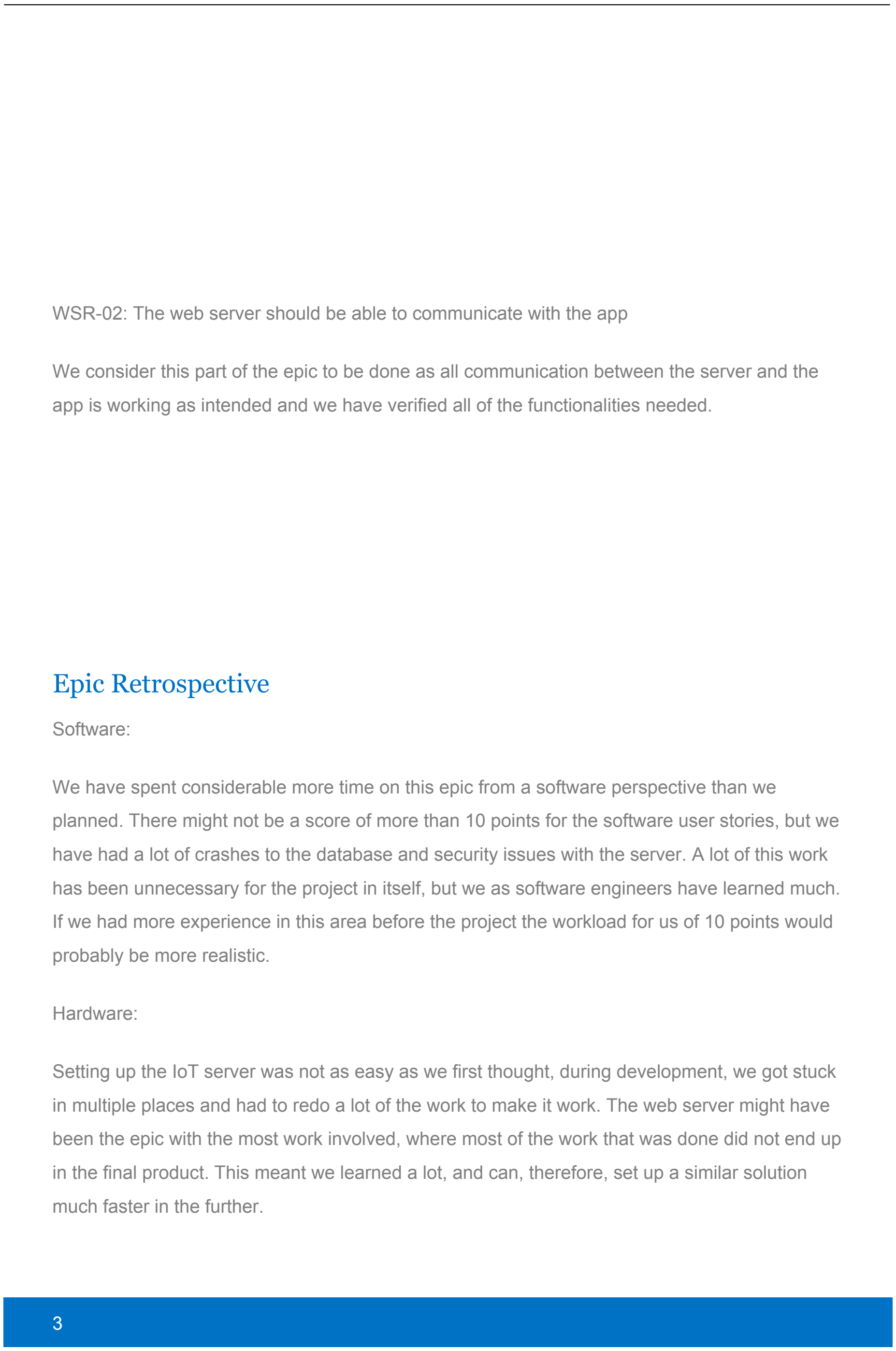


USN-51: We as hardware engineers need to send the IoT data from the MQTT broker to the database, so the data can be collected using a simple call by the app	**5**	**Done**
USN-35: We as hardware engineers need to model the data sent from the Arduino to the webserver so we are able to send data in a safe, reliable and efficient way	**3**	**Done**
USN-36: We as Hardware engineers need to set a test server(IoT platform) for testing our connection with IoT modules(MKR NB 1500).	**5**	**Done**
USN-38: We as Hardware Engineers need to connect our IoT modules to the test server(IoT platform) by following the right procedure.	**6**	**Done**
USN-44: We as Hardware Engineers need to send our sensor data from the Arduino module to the server based on the Modelled data.	**5**	**Not done**
USN-56: We as hardware engineers need to simulate MQTT data using a computer program, to limit the data usage on the SIM card	**5**	**Discarded**

Epic Review

WSR-01: The web server should be able to communicate with the bike.

There are still some formatting issues with the JSON data being sent, this can be solved by tweaking the message until it's in the right format



WSR-02: The web server should be able to communicate with the app

We consider this part of the epic to be done as all communication between the server and the app is working as intended and we have verified all of the functionalities needed.

Epic Retrospective

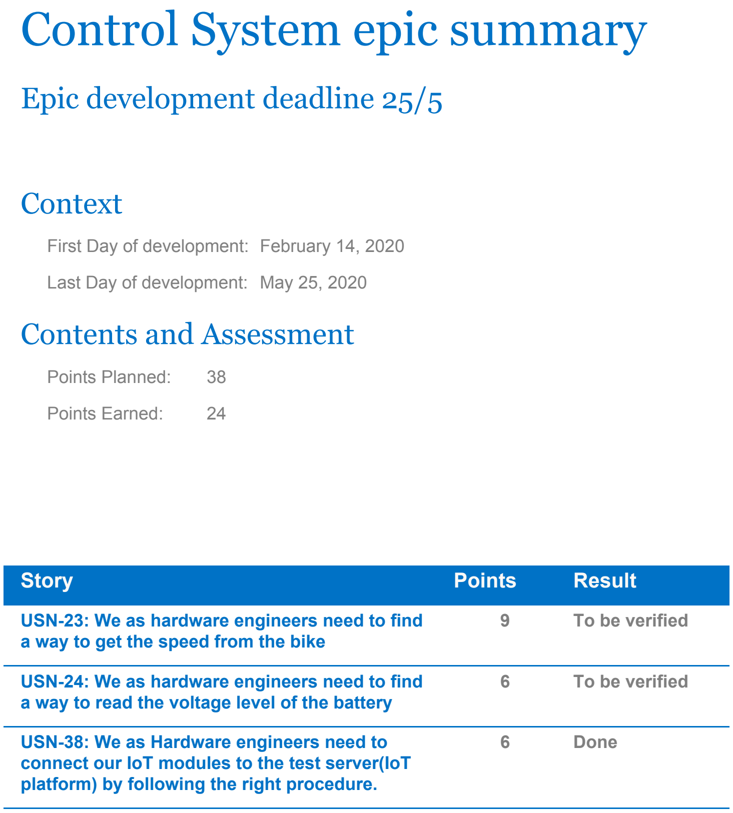
Software:

We have spent considerable more time on this epic from a software perspective than we planned. There might not be a score of more than 10 points for the software user stories, but we have had a lot of crashes to the database and security issues with the server. A lot of this work has been unnecessary for the project in itself, but we as software engineers have learned much. If we had more experience in this area before the project the workload for us of 10 points would probably be more realistic.

Hardware:

Setting up the IoT server was not as easy as we first thought, during development, we got stuck in multiple places and had to redo a lot of the work to make it work. The web server might have been the epic with the most work involved, where most of the work that was done did not end up in the final product. This meant we learned a lot, and can, therefore, set up a similar solution much faster in the further.

A.3.4 CS epic



Control System epic summary

Epic development deadline 25/5

Context

First Day of development: February 14, 2020

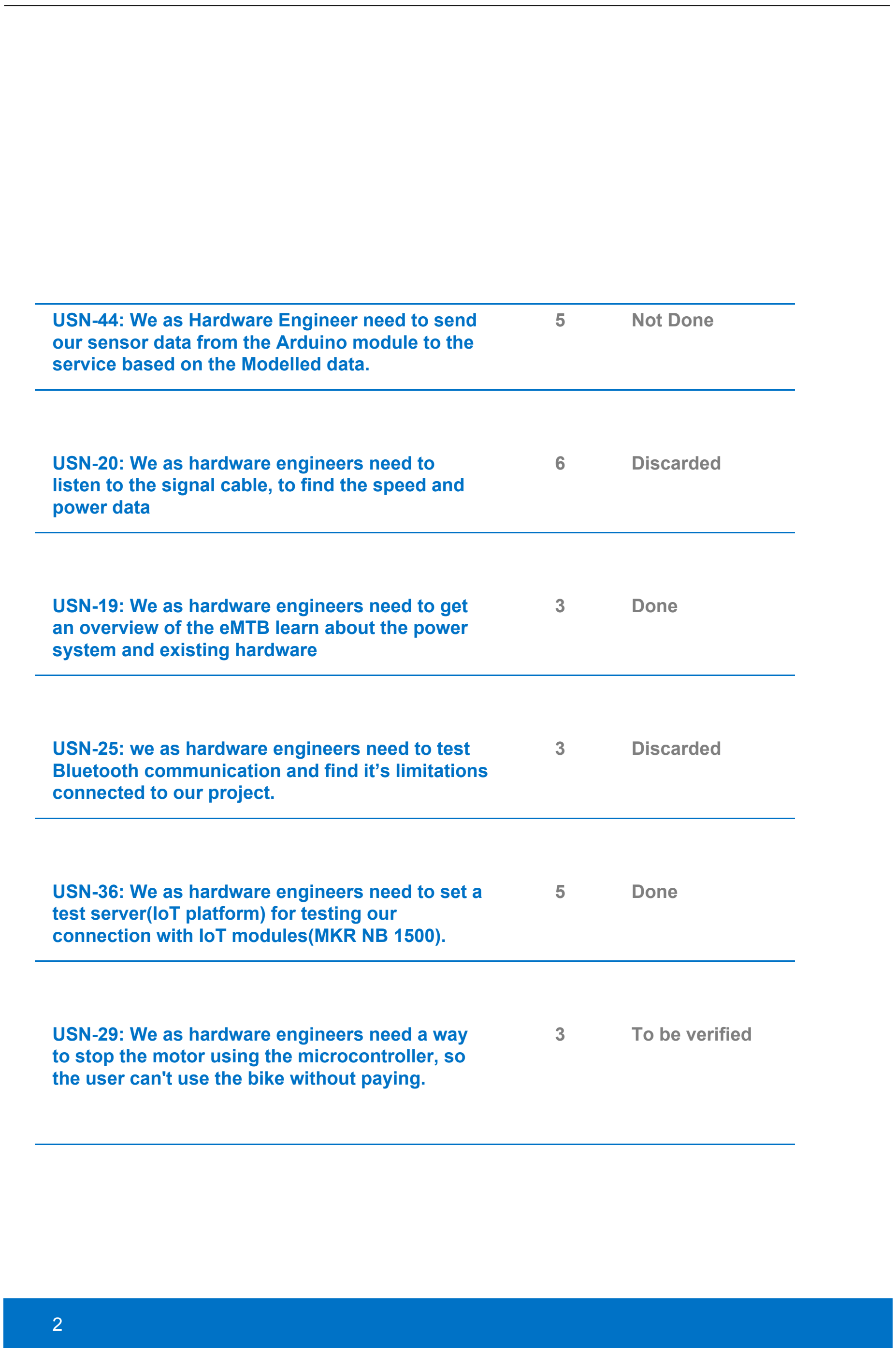
Last Day of development: May 25, 2020

Contents and Assessment

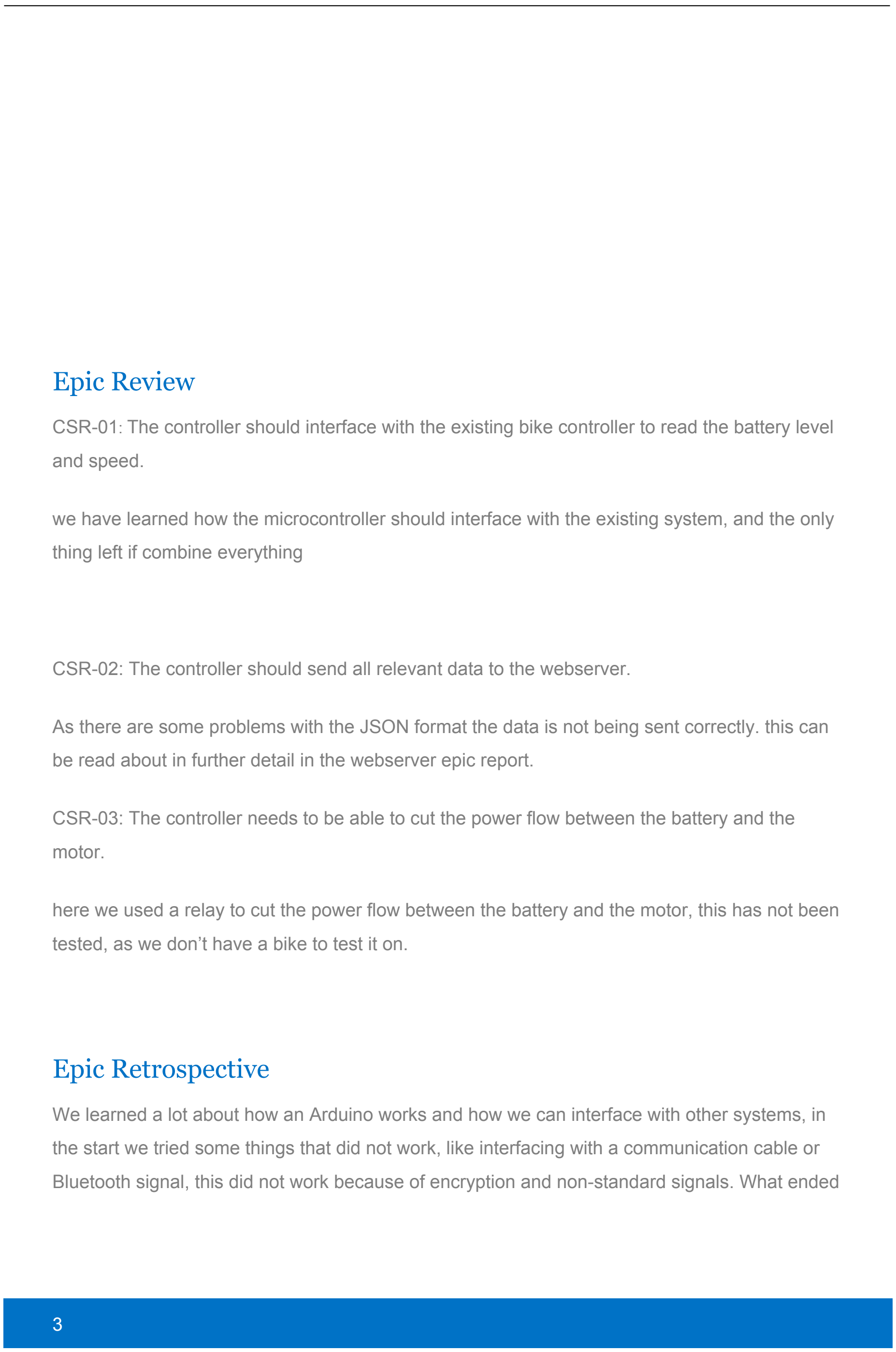
Points Planned: 38

Points Earned: 24

Story	Points	Result
USN-23: We as hardware engineers need to find a way to get the speed from the bike	9	To be verified
USN-24: We as hardware engineers need to find a way to read the voltage level of the battery	6	To be verified
USN-38: We as Hardware engineers need to connect our IoT modules to the test server(IoT platform) by following the right procedure.	6	Done



USN-44: We as Hardware Engineer need to send our sensor data from the Arduino module to the service based on the Modelled data.	5	Not Done
USN-20: We as hardware engineers need to listen to the signal cable, to find the speed and power data	6	Discarded
USN-19: We as hardware engineers need to get an overview of the eMTB learn about the power system and existing hardware	3	Done
USN-25: we as hardware engineers need to test Bluetooth communication and find it’s limitations connected to our project.	3	Discarded
USN-36: We as hardware engineers need to set a test server(IoT platform) for testing our connection with IoT modules(MKR NB 1500).	5	Done
USN-29: We as hardware engineers need a way to stop the motor using the microcontroller, so the user can't use the bike without paying.	3	To be verified



Epic Review

CSR-01: The controller should interface with the existing bike controller to read the battery level and speed.

we have learned how the microcontroller should interface with the existing system, and the only thing left if combine everything

CSR-02: The controller should send all relevant data to the webserver.

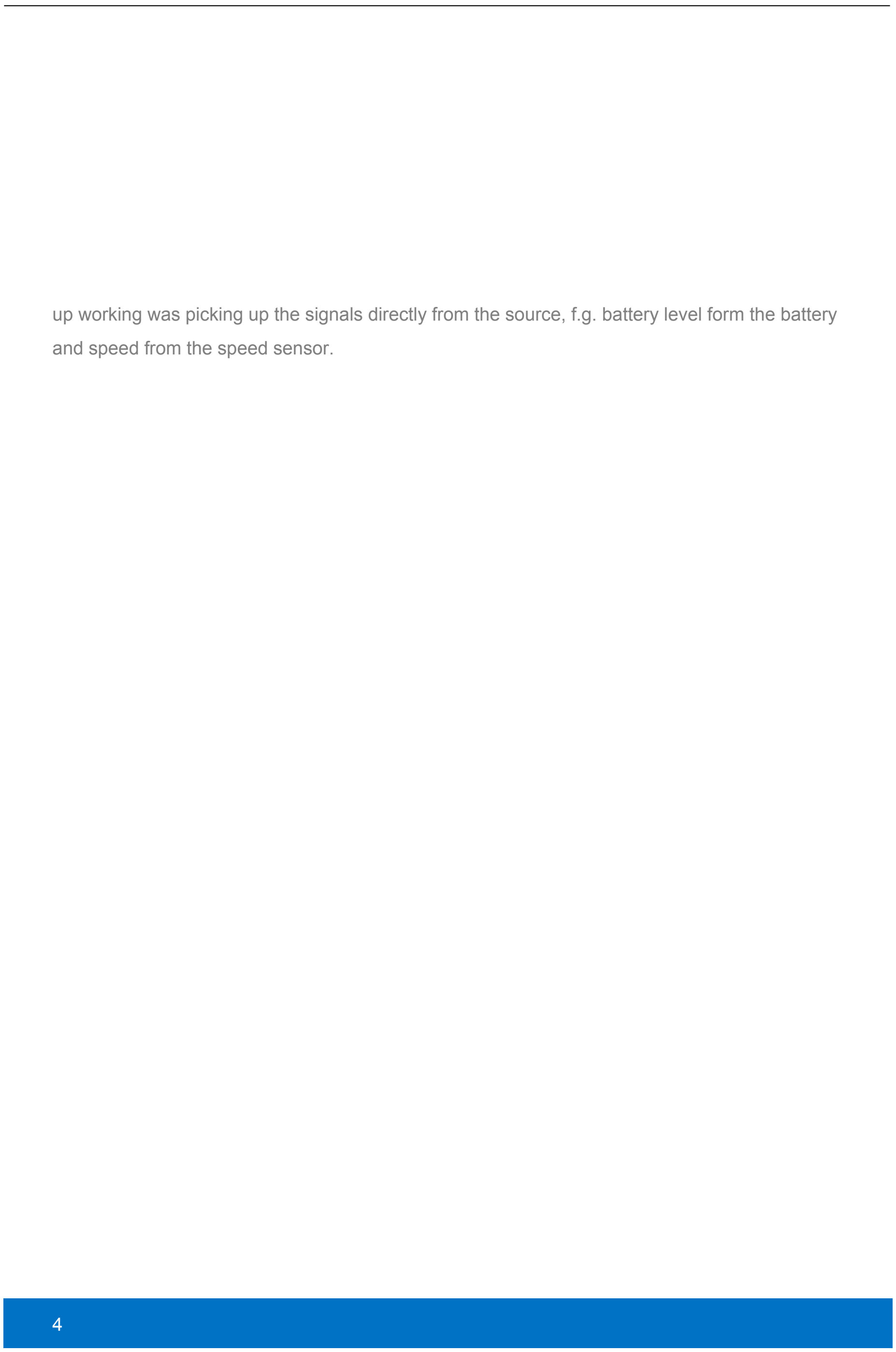
As there are some problems with the JSON format the data is not being sent correctly. this can be read about in further detail in the webserver epic report.

CSR-03: The controller needs to be able to cut the power flow between the battery and the motor.

here we used a relay to cut the power flow between the battery and the motor, this has not been tested, as we don’t have a bike to test it on.

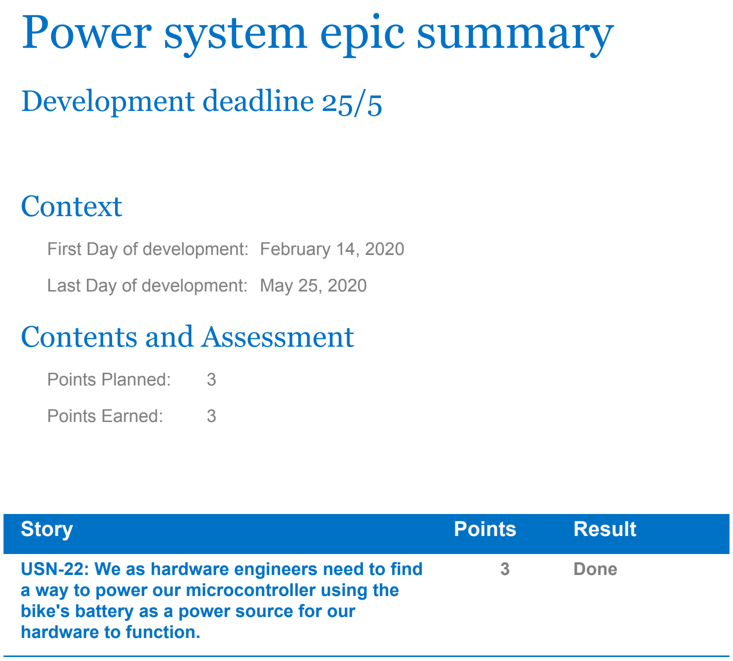
Epic Retrospective

We learned a lot about how an Arduino works and how we can interface with other systems, in the start we tried some things that did not work, like interfacing with a communication cable or Bluetooth signal, this did not work because of encryption and non-standard signals. What ended



up working was picking up the signals directly from the source, f.g. battery level form the battery and speed from the speed sensor.

A.3.5 PS epic



Power system epic summary

Development deadline 25/5

Context

First Day of development: February 14, 2020

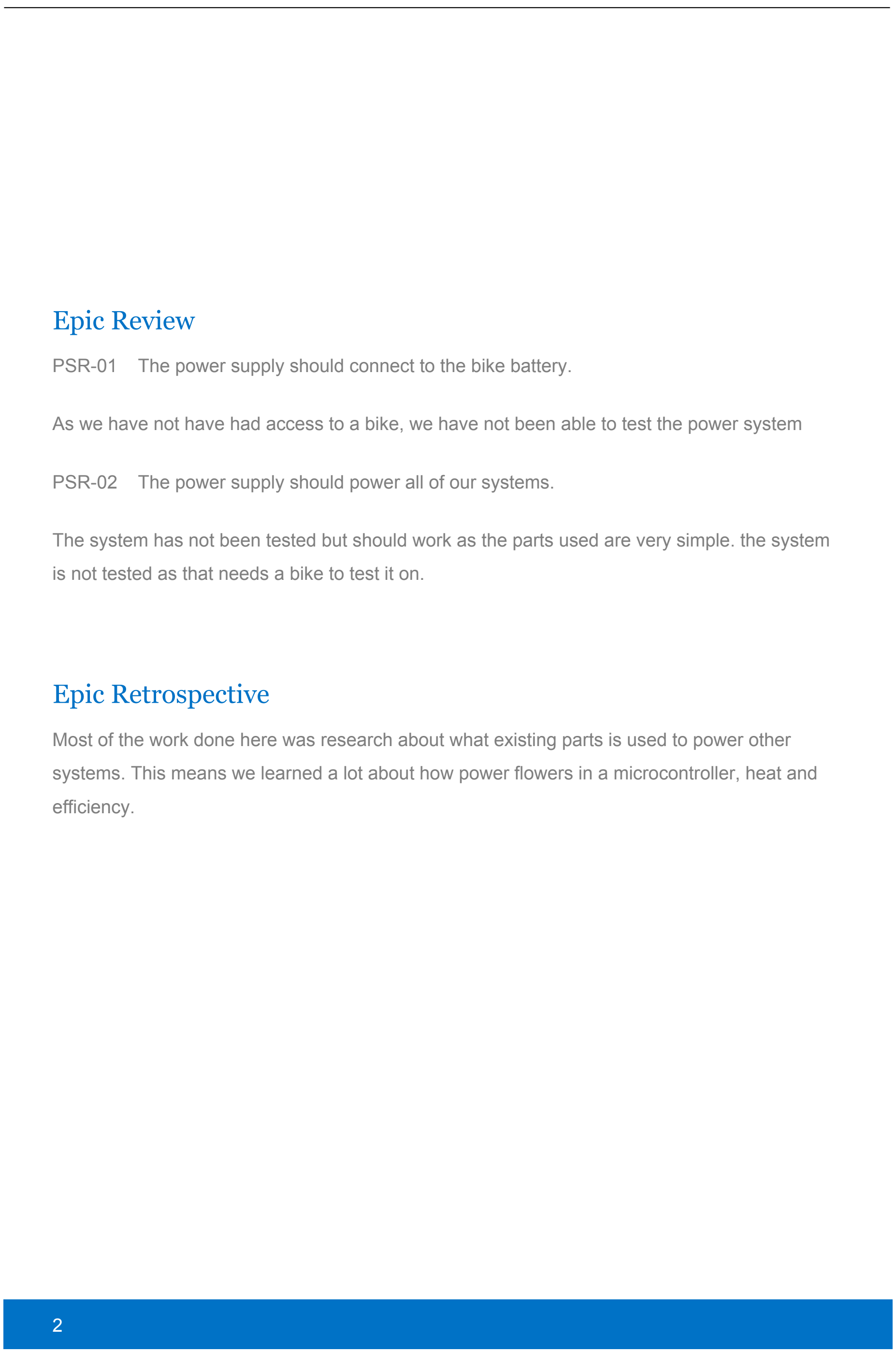
Last Day of development: May 25, 2020

Contents and Assessment

Points Planned: 3

Points Earned: 3

Story	Points	Result
USN-22: We as hardware engineers need to find a way to power our microcontroller using the bike's battery as a power source for our hardware to function.	3	Done



Epic Review

PSR-01 The power supply should connect to the bike battery.

As we have not have had access to a bike, we have not been able to test the power system

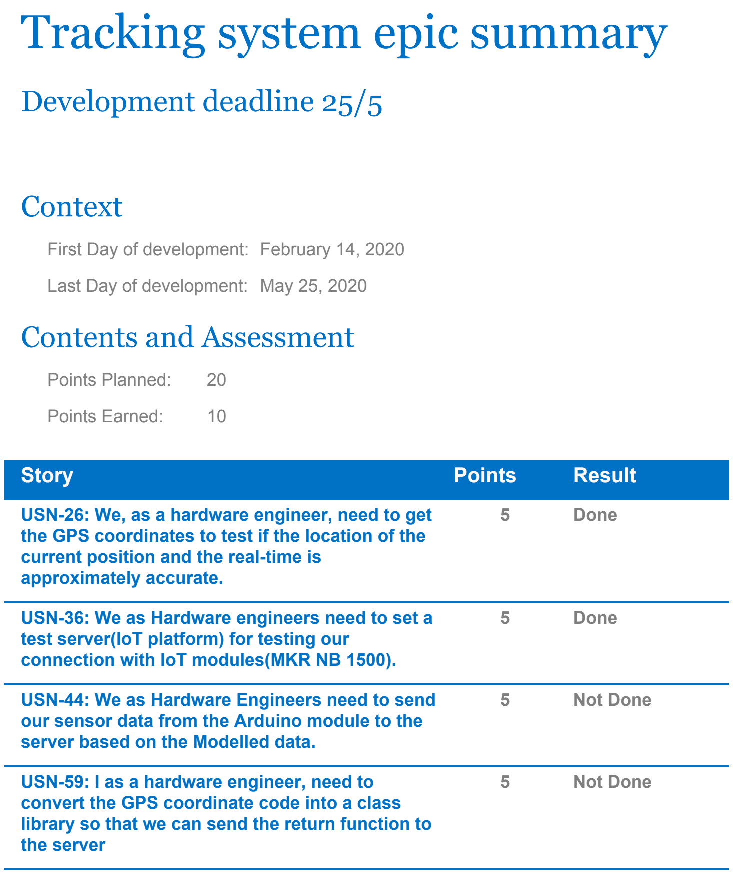
PSR-02 The power supply should power all of our systems.

The system has not been tested but should work as the parts used are very simple. the system is not tested as that needs a bike to test it on.

Epic Retrospective

Most of the work done here was research about what existing parts is used to power other systems. This means we learned a lot about how power flowers in a microcontroller, heat and efficiency.

A.3.6 TS epic



Tracking system epic summary

Development deadline 25/5

Context

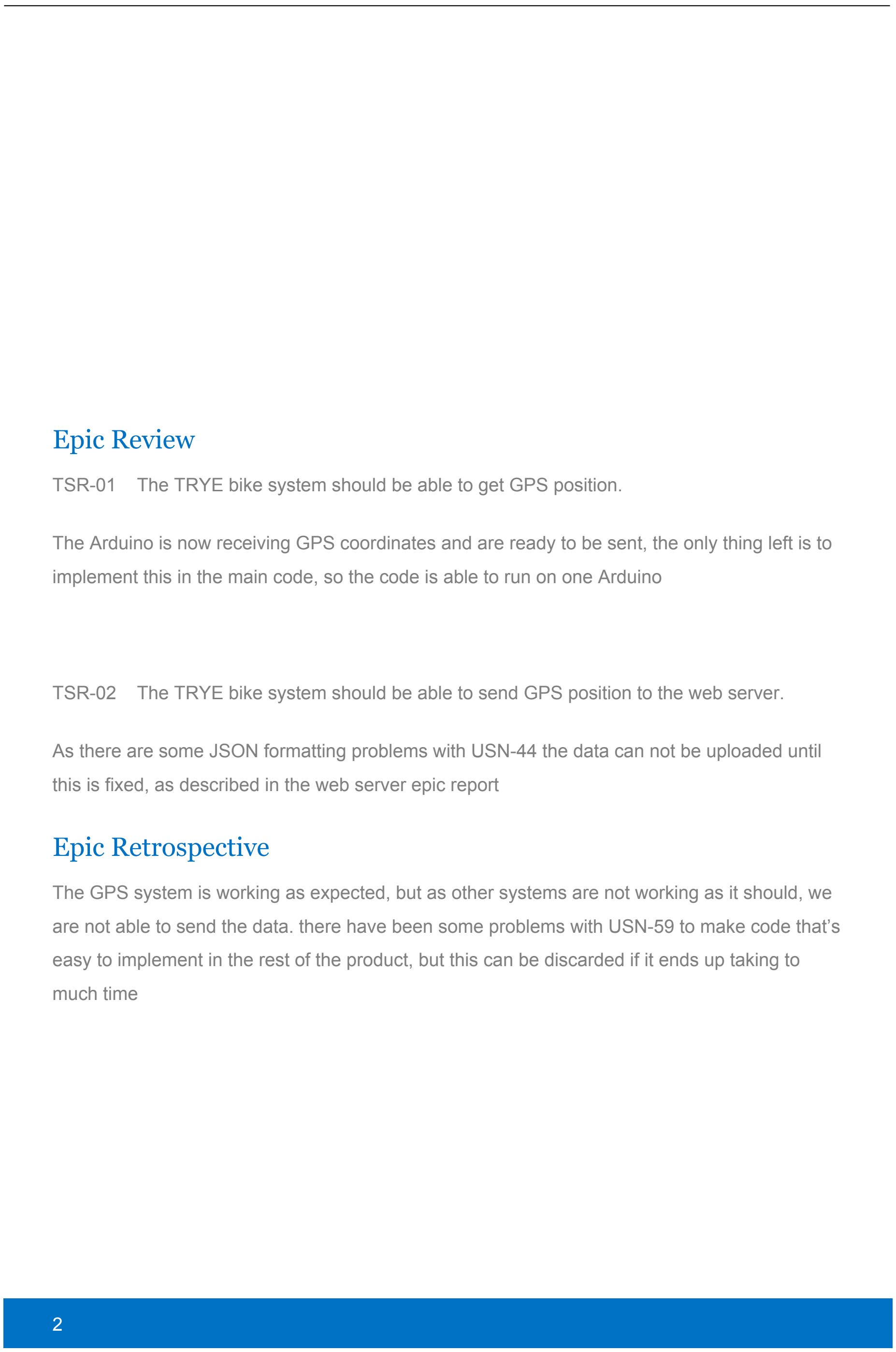
First Day of development: February 14, 2020

Last Day of development: May 25, 2020

Contents and Assessment

Points Planned:	20
Points Earned:	10

Story	Points	Result
USN-26: We, as a hardware engineer, need to get the GPS coordinates to test if the location of the current position and the real-time is approximately accurate.	5	Done
USN-36: We as Hardware engineers need to set a test server(IoT platform) for testing our connection with IoT modules(MKR NB 1500).	5	Done
USN-44: We as Hardware Engineers need to send our sensor data from the Arduino module to the server based on the Modelled data.	5	Not Done
USN-59: I as a hardware engineer, need to convert the GPS coordinate code into a class library so that we can send the return function to the server	5	Not Done



Epic Review

TSR-01 The TRYE bike system should be able to get GPS position.

The Arduino is now receiving GPS coordinates and are ready to be sent, the only thing left is to implement this in the main code, so the code is able to run on one Arduino

TSR-02 The TRYE bike system should be able to send GPS position to the web server.

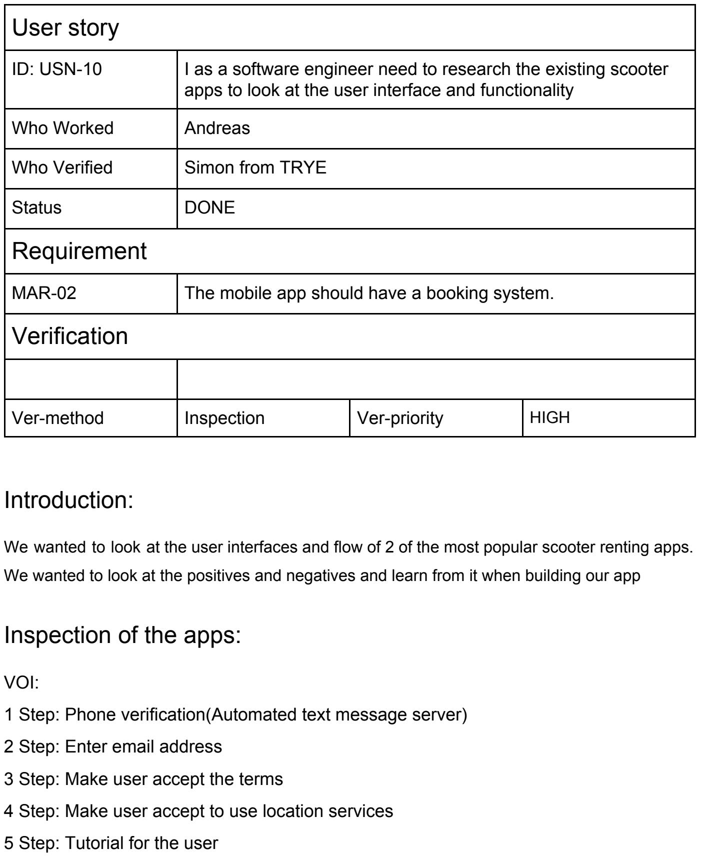
As there are some JSON formatting problems with USN-44 the data can not be uploaded until this is fixed, as described in the web server epic report

Epic Retrospective

The GPS system is working as expected, but as other systems are not working as it should, we are not able to send the data. there have been some problems with USN-59 to make code that’s easy to implement in the rest of the product, but this can be discarded if it ends up taking to much time

A.4 Software User Stories

A.4.1 USN-10 Researching existing scooter apps



User story			
ID: USN-10	I as a software engineer need to research the existing scooter apps to look at the user interface and functionality		
Who Worked	Andreas		
Who Verified	Simon from TRYE		
Status	DONE		
Requirement			
MAR-02	The mobile app should have a booking system.		
Verification			
Ver-method	Inspection	Ver-priority	HIGH

Introduction:

We wanted to look at the user interfaces and flow of 2 of the most popular scooter renting apps.
We wanted to look at the positives and negatives and learn from it when building our app

Inspection of the apps:

VOI:

1 Step: Phone verification(Automated text message server)

2 Step: Enter email address

3 Step: Make user accept the terms

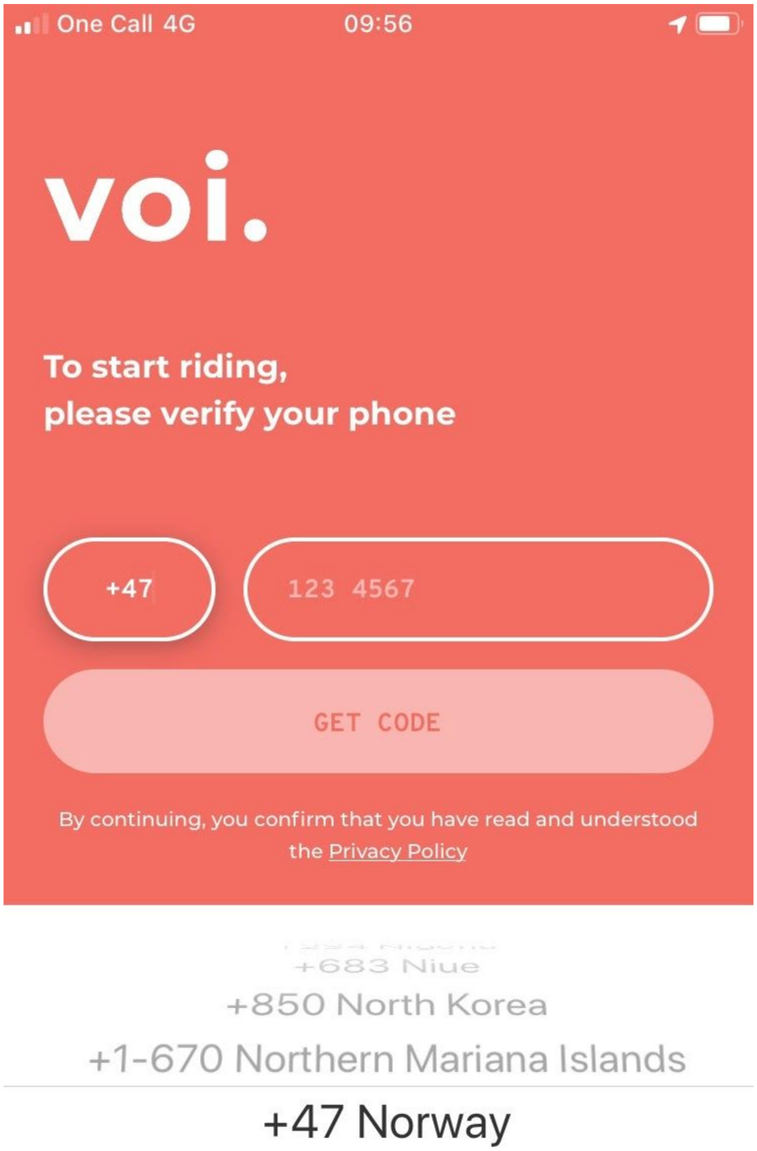
4 Step: Make user accept to use location services

5 Step: Tutorial for the user

App core

Help and Faq
Damaged bike? (Report case)
Bike not working? (FAQ or talk with support)
Issues with the app(Phone died, can’t find location)
Sustainability(How environmentally friendly is our bikes)
Self-insurance (If the user gets injured)
Data handling and user information(Follow GDPR)
Terms of use(FAQ list)
Payments(FAQ list)
Riding and parking(How to contact support and FAQ)

Payment and ride history
General rules
Profile page
Map page(main UI)
Scan to ride
Add payment
Menu



One Call 4G 09:56

voi.

To start riding,
please verify your phone

+47

123 4567

GET CODE

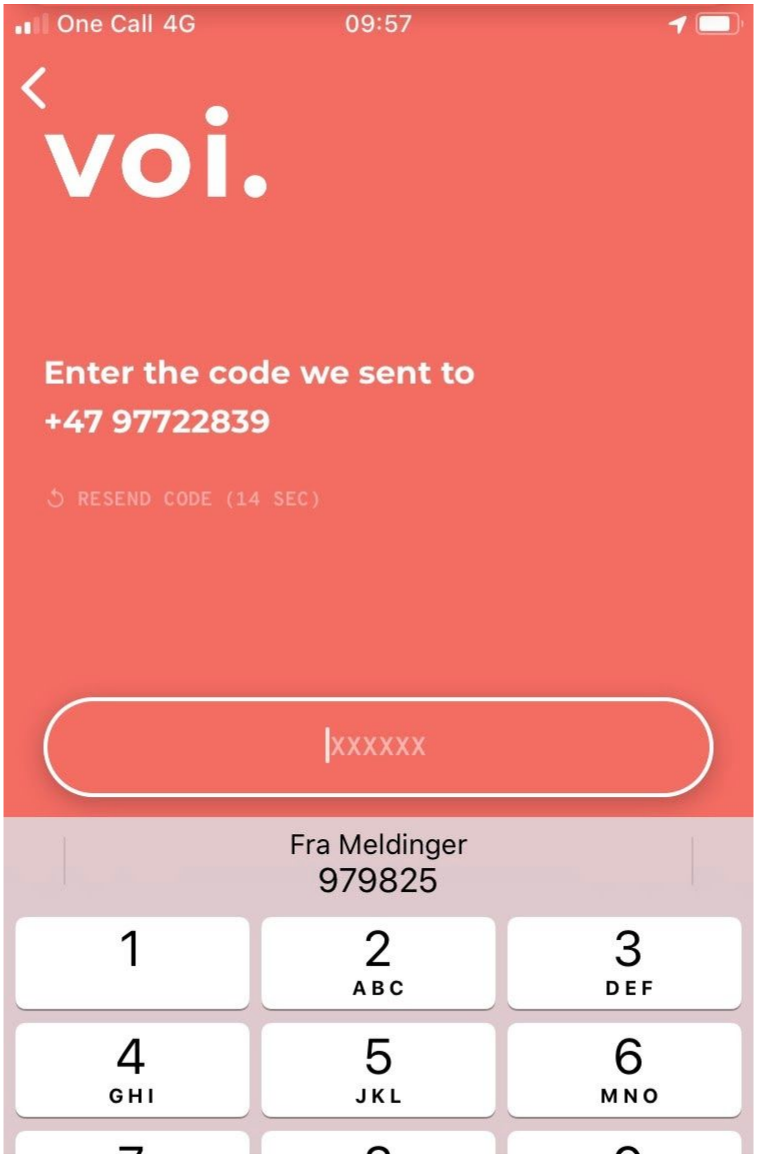
By continuing, you confirm that you have read and understood the Privacy Policy

+683 Niue

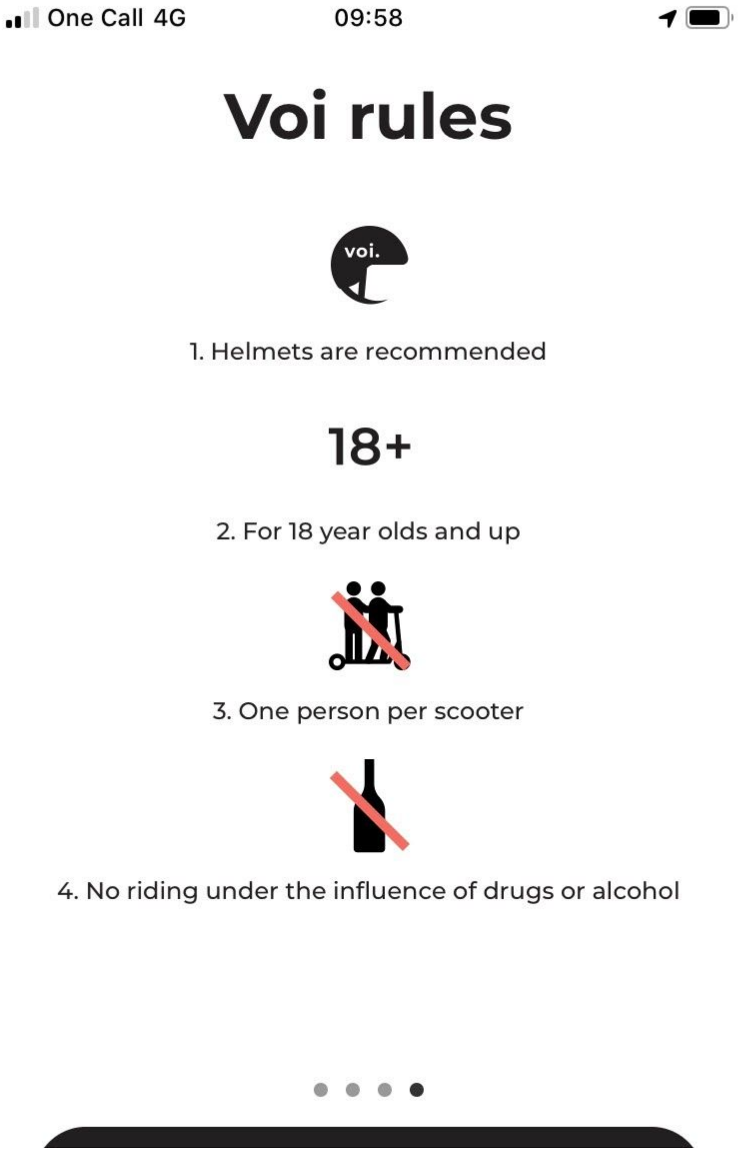
+850 North Korea

+1-670 Northern Mariana Islands

+47 Norway



Enter the code we sent to +47 97722839



One Call 4G 09:58

Voi rules

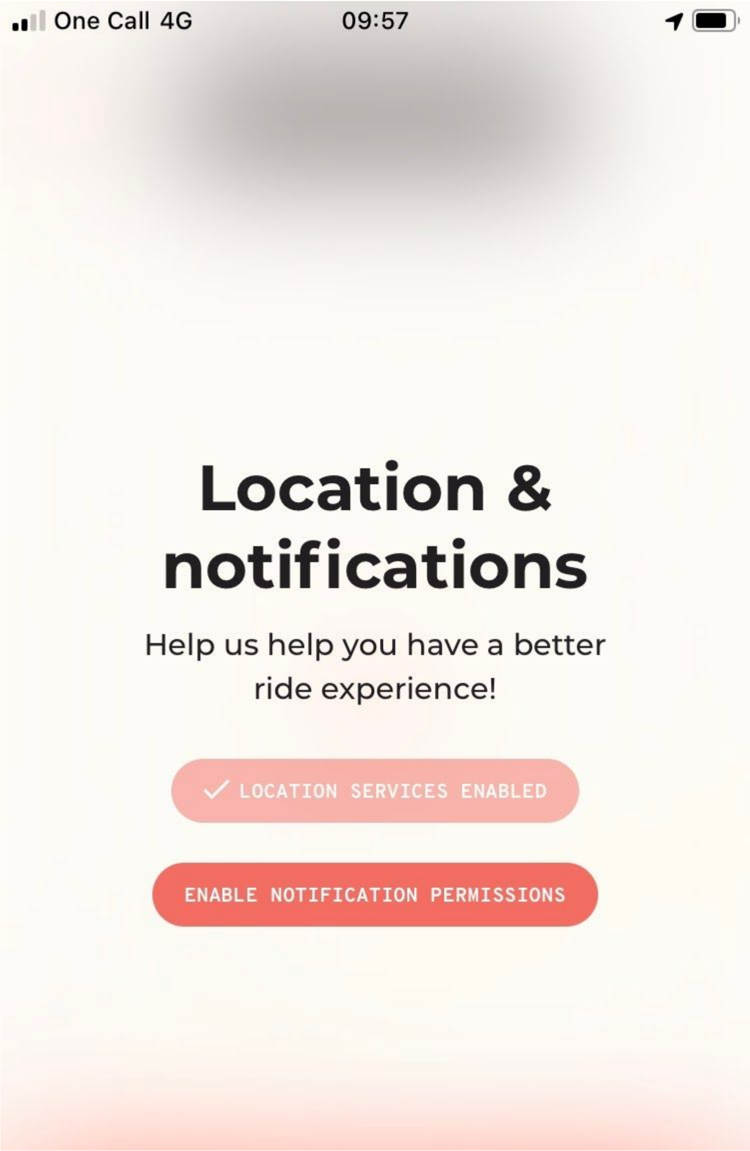
1. Helmets are recommended

18+

2. For 18 year olds and up

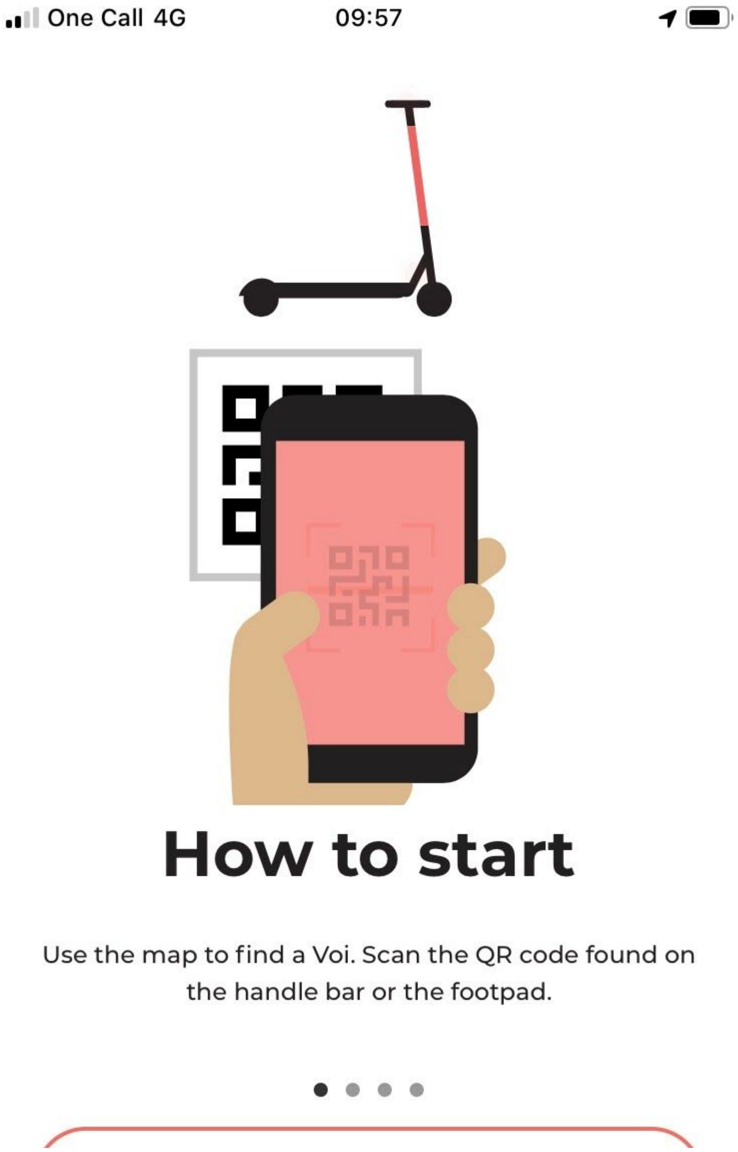
3. One person per scooter

4. No riding under the influence of drugs or alcohol



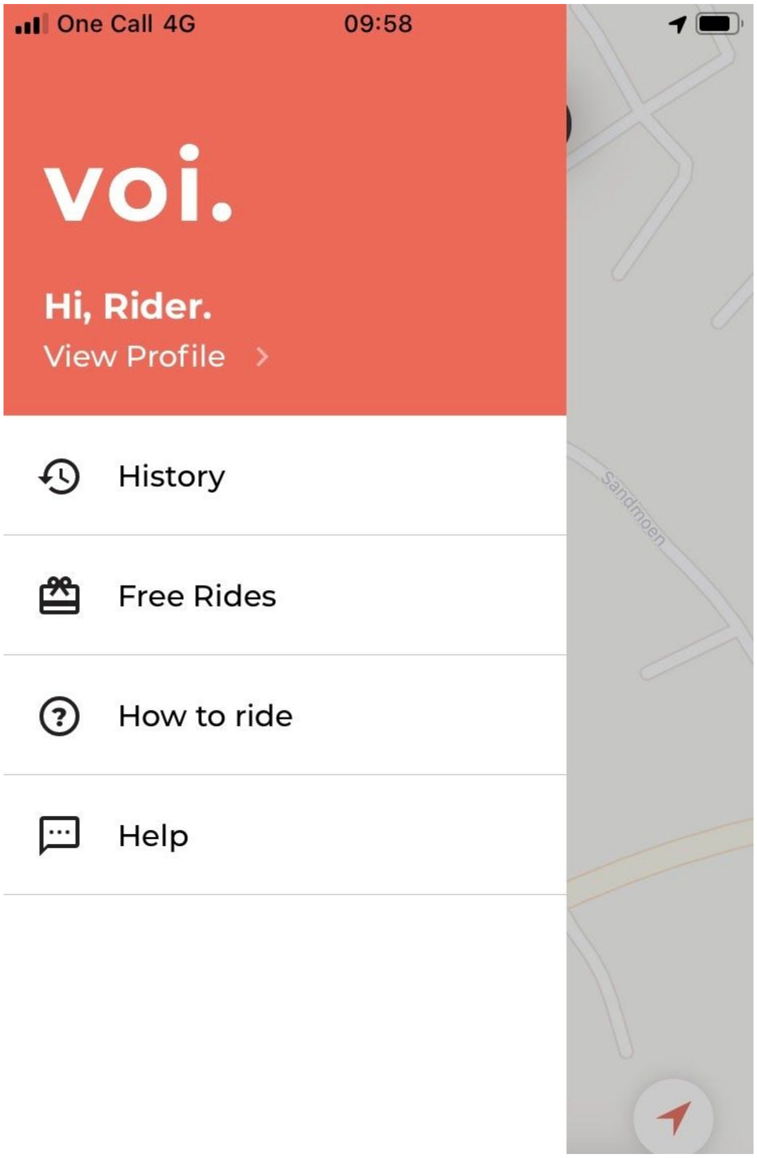
Location & notifications

Help us help you have a better ride experience!



How to start

Use the map to find a Voi. Scan the QR code found on the handle bar or the footpad.



One Call 4G 09:58

voi.

Hi, Rider.

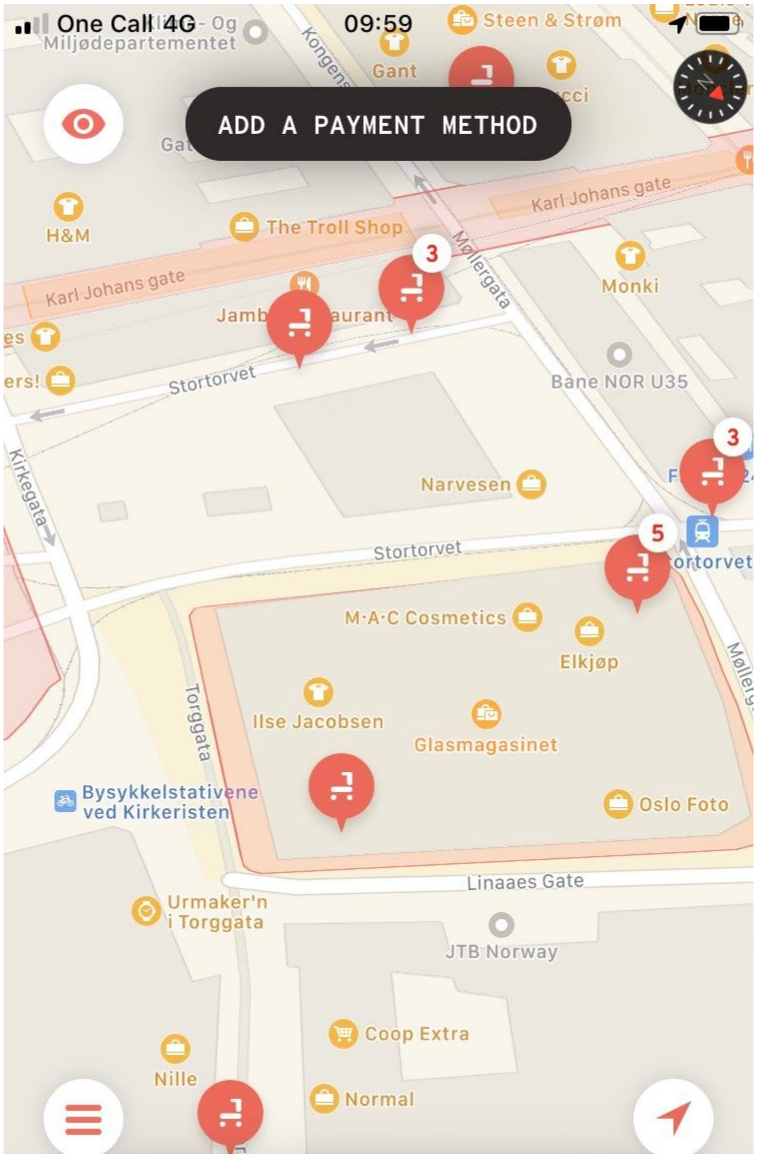
View Profile >

History

Free Rides

How to ride

Help


One Call 4G
09:59
ADD A PAYMENT METHOD
Miljødepartementet
Steen & Strøm
Gant
Karl Johans gate
H&M
The Troll Shop
Møllergata
Monki
Karl Johans gate
Jamb
aurant
Stortorvet
Bane NOR U35
Narvesen
Kirkegata
Stortorvet
M·A·C Cosmetics
Elkjøp
Torggata
Ilse Jacobsen
Glasmagasinet
Bysykkelstativene ved Kirkeristen
Oslo Foto
Linaaes Gate
Urmaker'n i Torggata
JTB Norway
Coop Extra
Nille
Normal
3
3
5

CIRC:

1 Step: Accept the use of location services

2 Step: Continue with Facebook or verify a phone number

3 Step: Enter email and name+surname

4 step: Accept term of use and privacy declaration for data

5 step: Choose and add method of payments

App core

Map page(main UI)

Scan to ride

How to ride

Menu

Find your current location

Help and faq

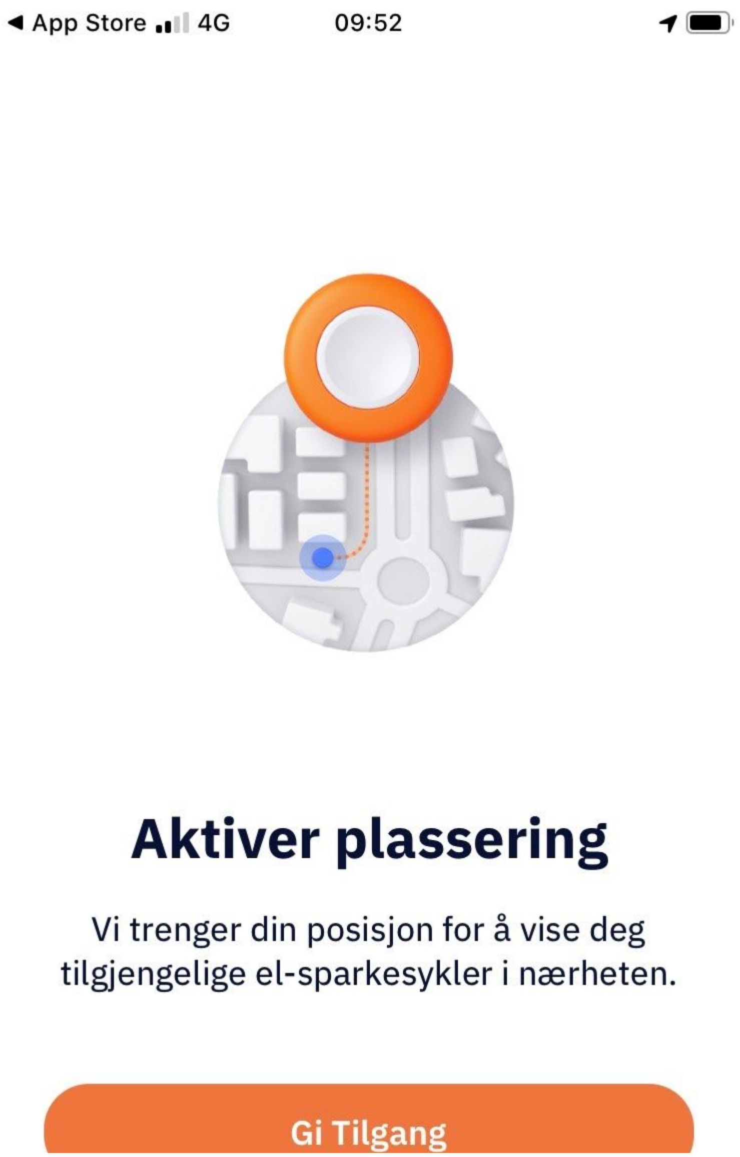
wallet

history

settings page with account information

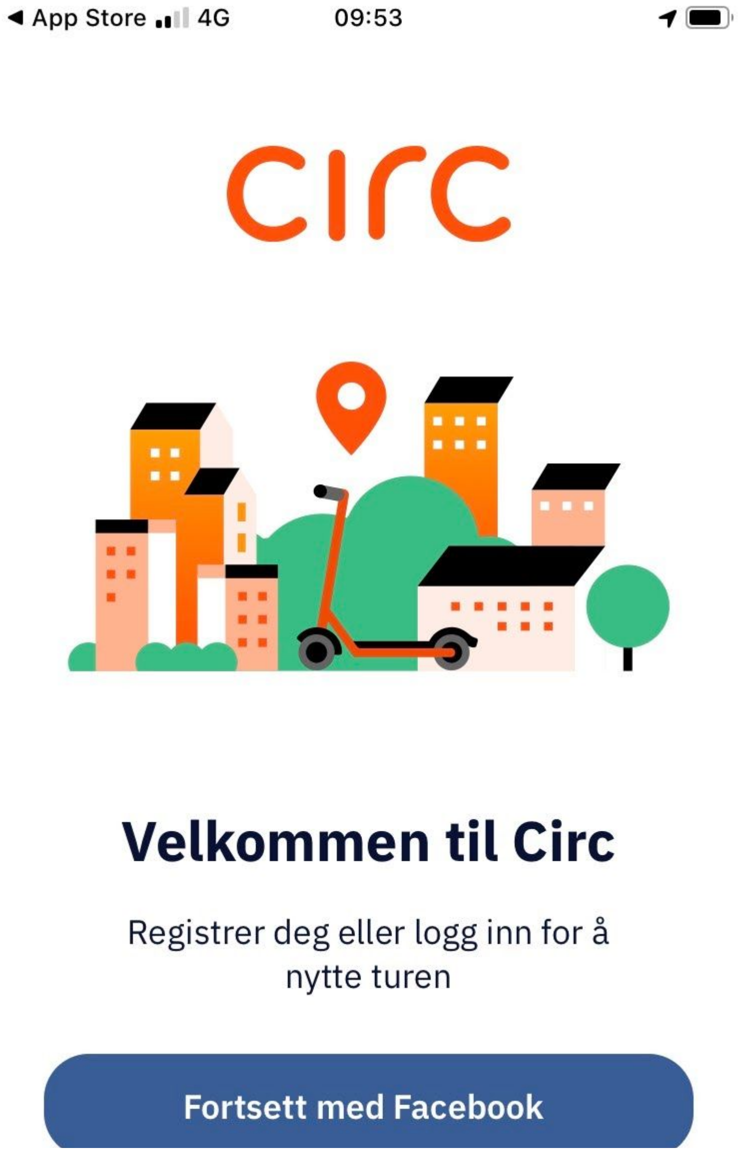
invite your friend

redeem code/campaigns



Aktiver plassering

Vi trenger din posisjon for å vise deg tilgjengelige el-sparkesykler i nærheten.

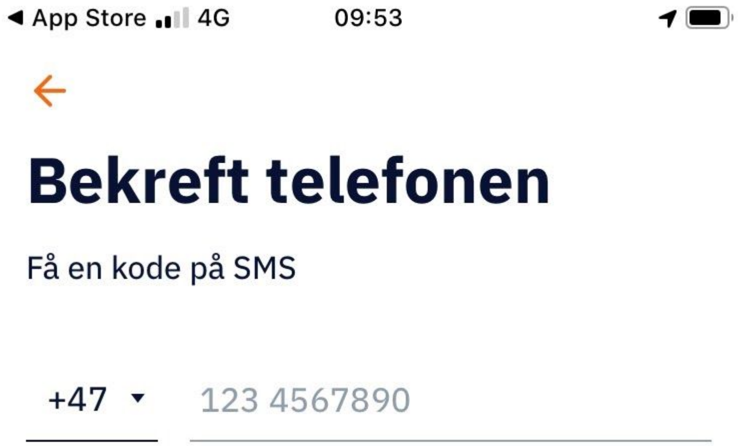


◀ App Store 4G 09:53

circ

Velkommen til Circ

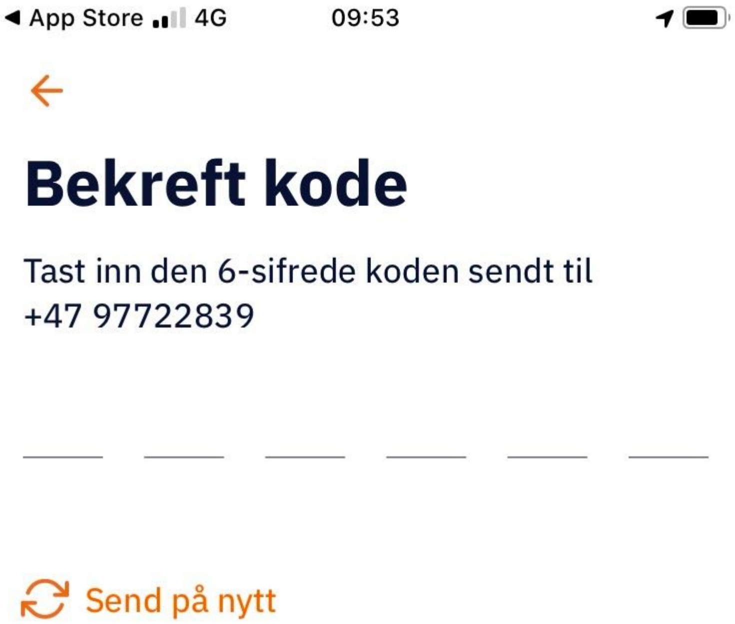
Registrer deg eller logg inn for å nytte turen



Bekreft telefonen

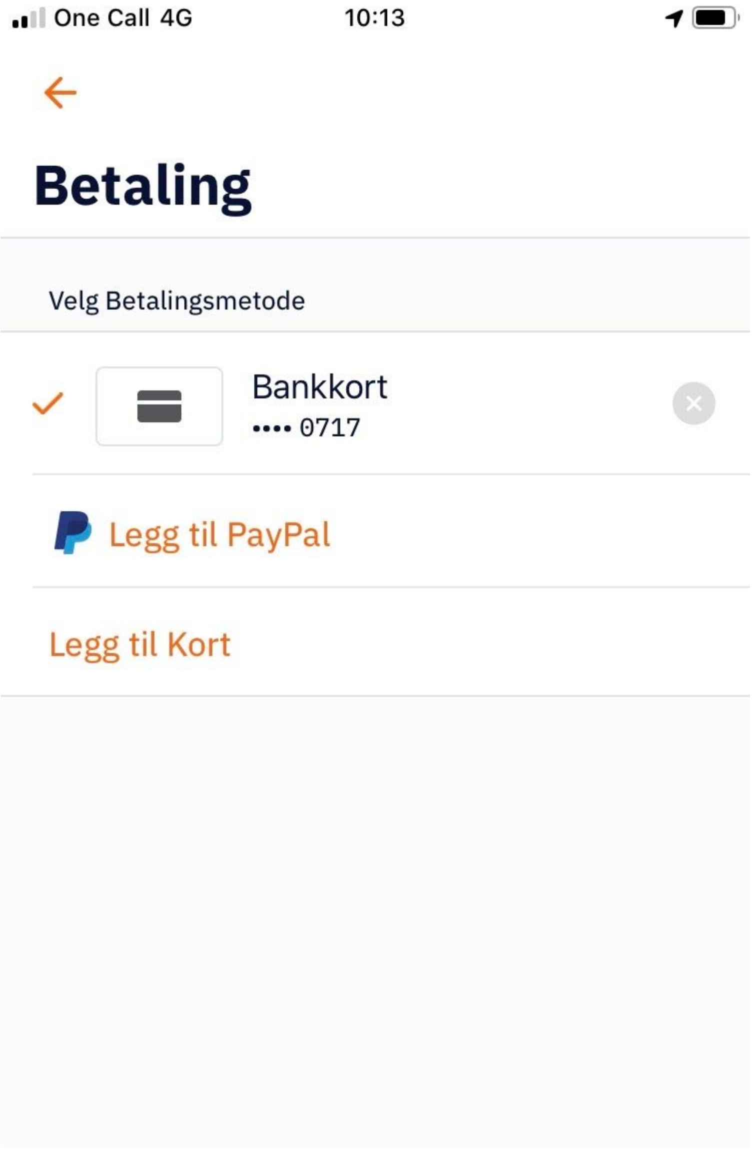
Få en kode på SMS

+47 ▾ 123 4567890



Bekreft kode

Tast inn den 6-sifrede koden sendt til
+47 97722839



One Call 4G 10:13

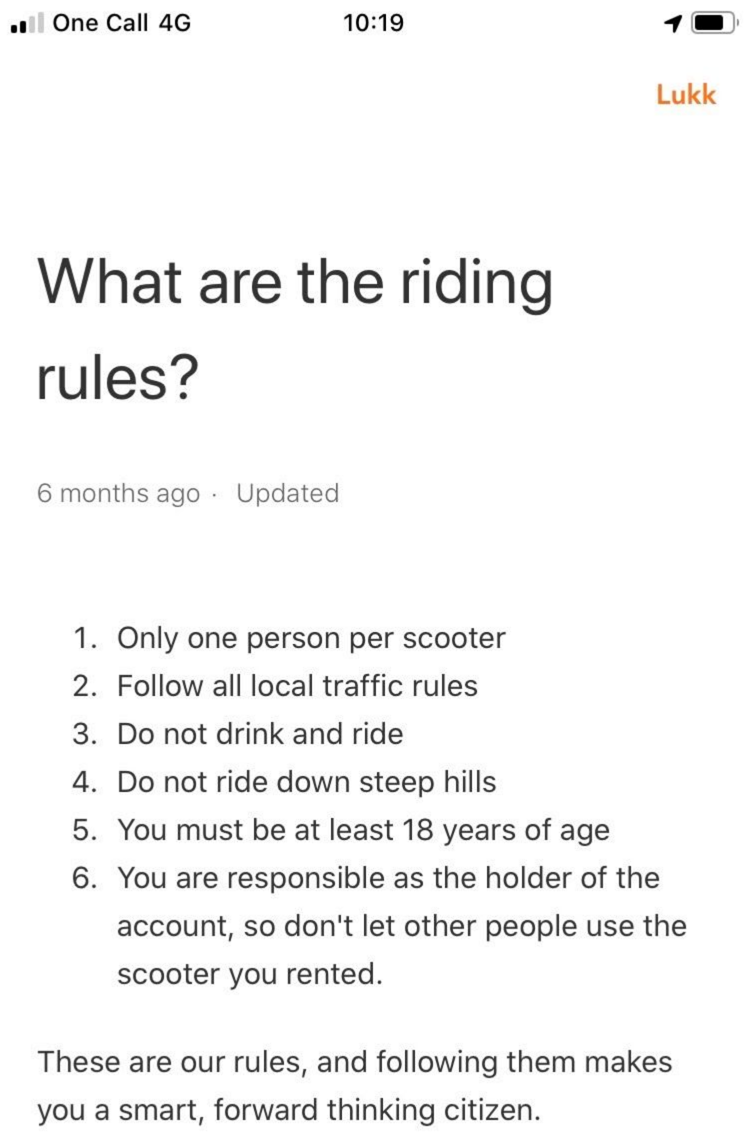
Betaling

Velg Betalingsmetode

✓ Bankkort •••• 0717

Legg til PayPal

Legg til Kort

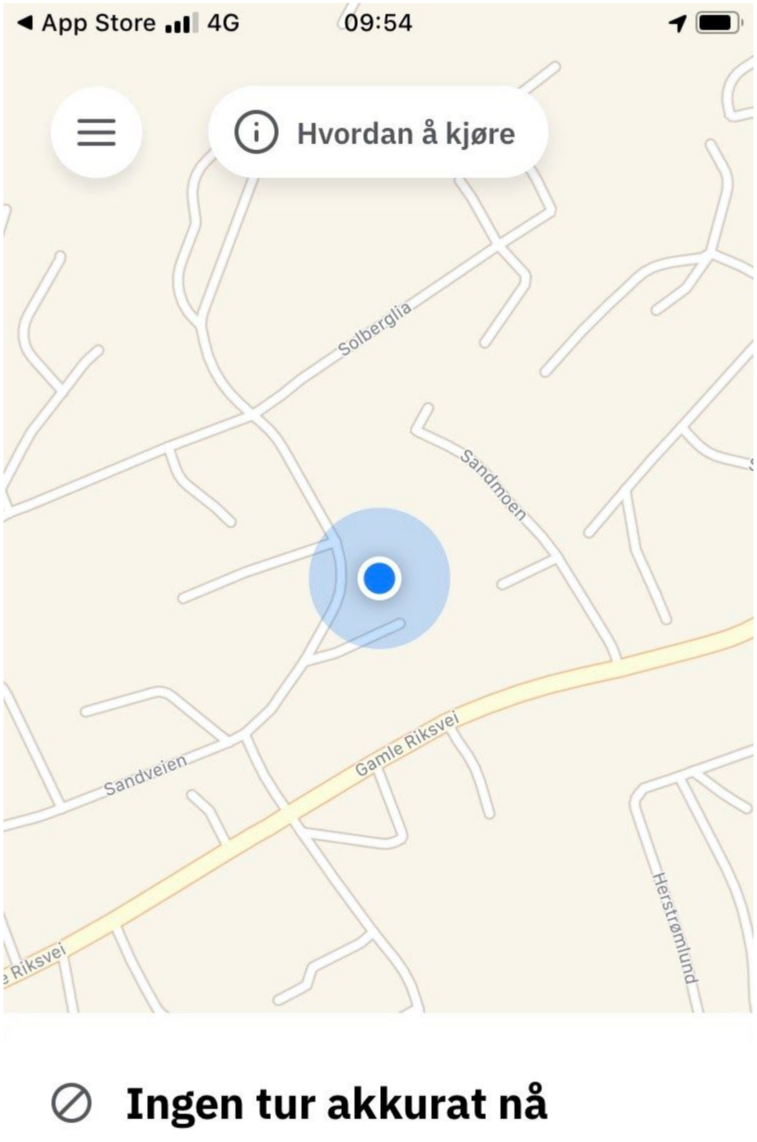


What are the riding rules?

6 months ago · Updated

1. Only one person per scooter
2. Follow all local traffic rules
3. Do not drink and ride
4. Do not ride down steep hills
5. You must be at least 18 years of age
6. You are responsible as the holder of the account, so don't let other people use the scooter you rented.

These are our rules, and following them makes you a smart, forward thinking citizen.



Ingen tur akkurat nå

Conclusion:

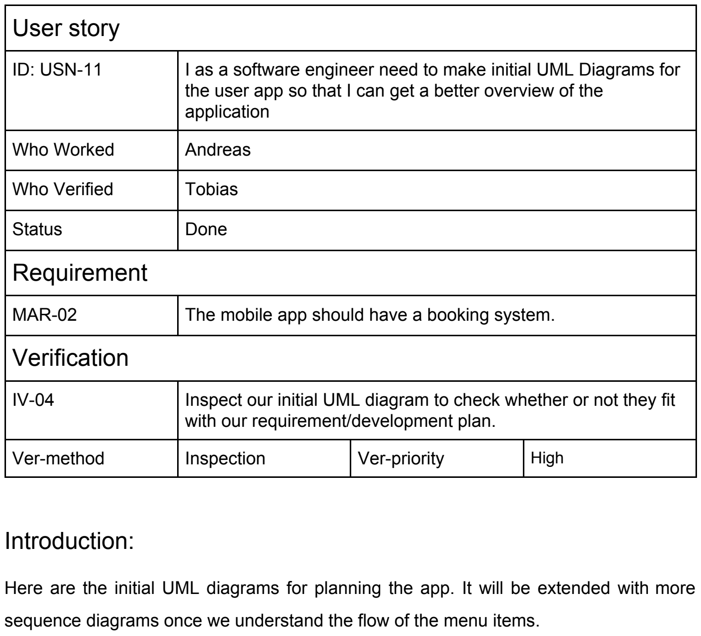
The VOI app has by far the best login interface since it is very simple to use and has fewer steps then the CIRC app. And we as users did not appreciate the way the CIRC app had 5 steps to go through before getting to the user interface where they force payment methods on you and wants to know your location right away instead of in the main menu. We think the VOI way is better here you just verify your phone number, enter an email then you are in the user interface right away, then the user can add payment options and turn on location services after to make it as simple and user friendly as possible. The circ app has a better sidebar menu which is easier to use and understand. The map interface is the same on both apps. Therefore we will model the app in a way that is close to the VOI app login interface. But model the sidebar menu from the CIRC app since we thought theirs is more similar to what we need for our app. It also much better to use because it was easier to understand the menu. The CIRC app had another perk that VOI didn't have, which was app credits. The app credits make it possible for the user to add credits to their account which they then can use for rent, similar to a gift card. This is also something we might want to implement if we are to make a hybrid payment solution with both booking and minute rent. I asked Simon from TRYE about my decision and he agreed with me in a meeting in Holmenkollen 27/1.

Links:

https://apps.apple.com/us/app/circ-electric-scooters/id1446543957

https://apps.apple.com/us/app/voi-scooters-get-magic-wheels/id1395921017

A.4.2 USN-11 Creating UML diagrams to get a better overview of the application



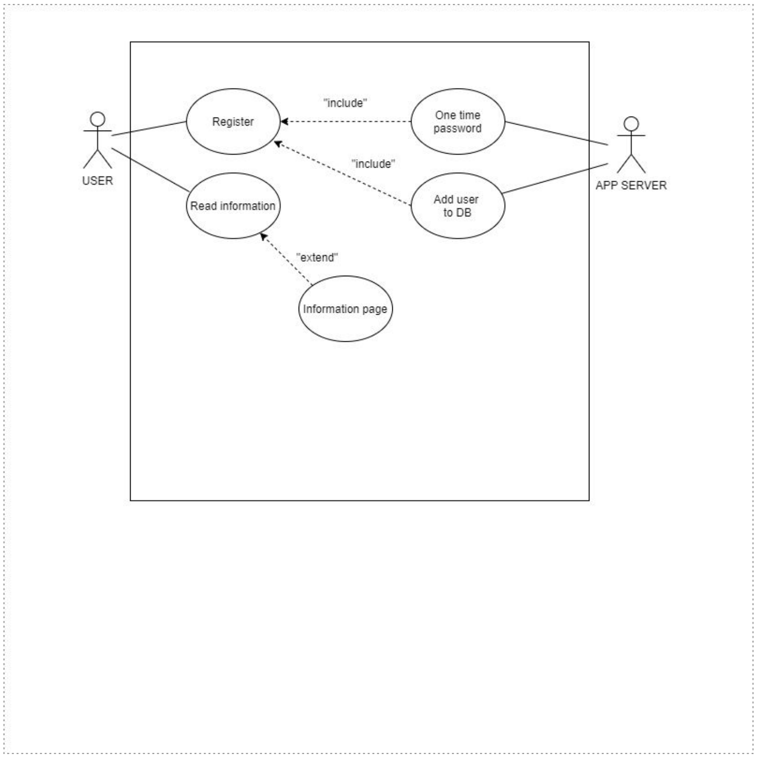
User story			
ID: USN-11	I as a software engineer need to make initial UML Diagrams for the user app so that I can get a better overview of the application		
Who Worked	Andreas		
Who Verified	Tobias		
Status	Done		
Requirement			
MAR-02	The mobile app should have a booking system.		
Verification			
IV-04	Inspect our initial UML diagram to check whether or not they fit with our requirement/development plan.		
Ver-method	Inspection	Ver-priority	High

Introduction:

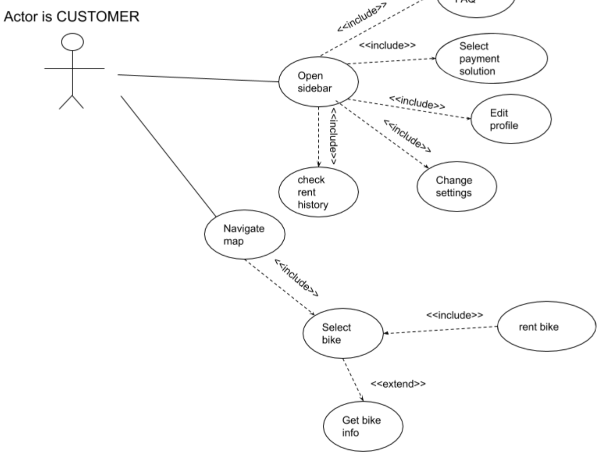
Here are the initial UML diagrams for planning the app. It will be extended with more sequence diagrams once we understand the flow of the menu items.

Figures:

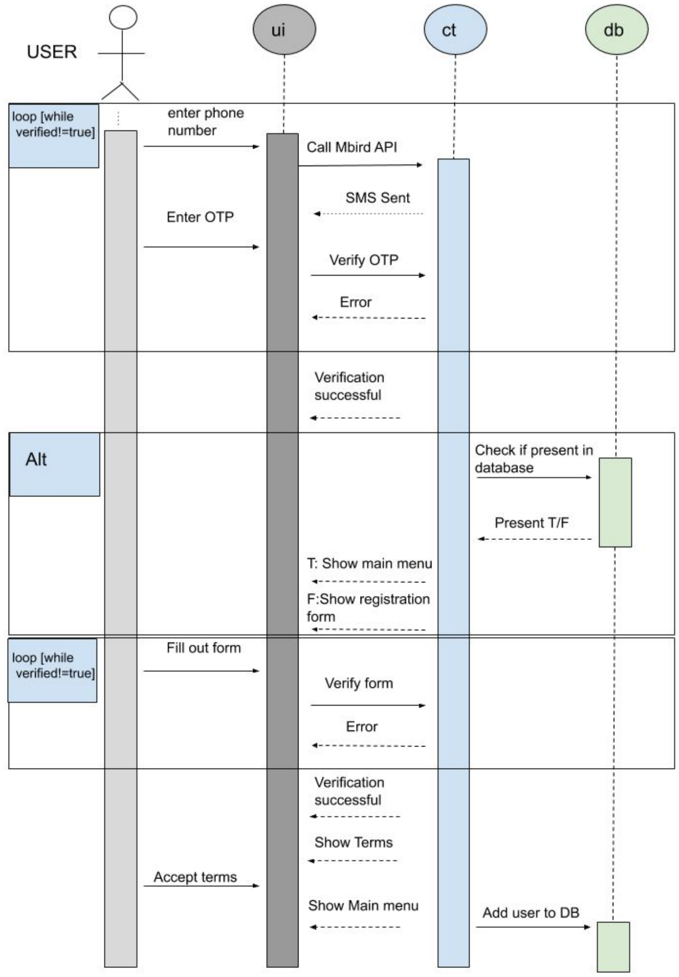
Use case register



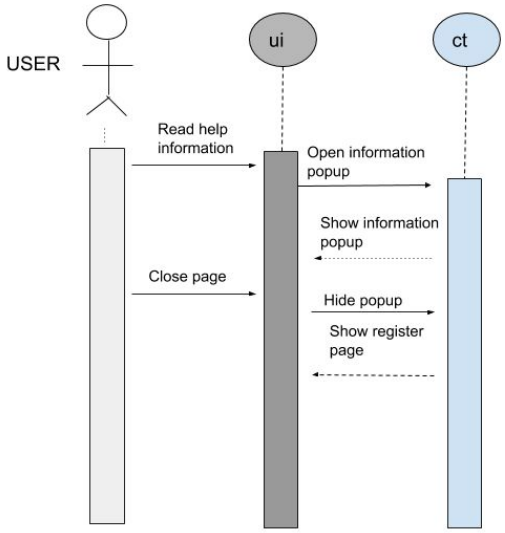
Main menu use case



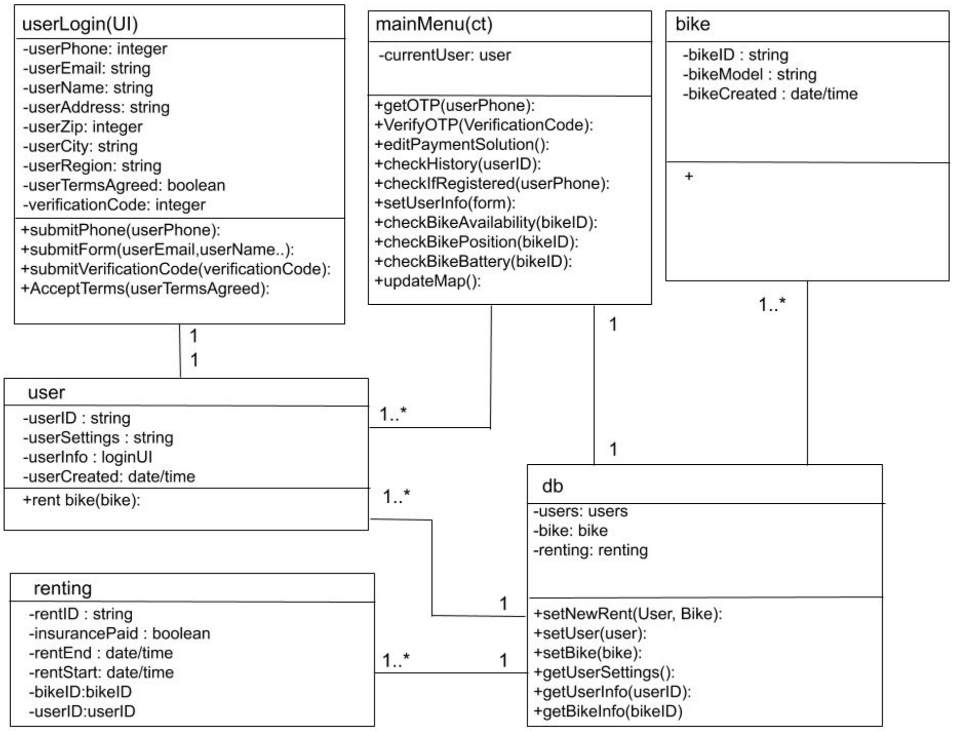
Register sequence



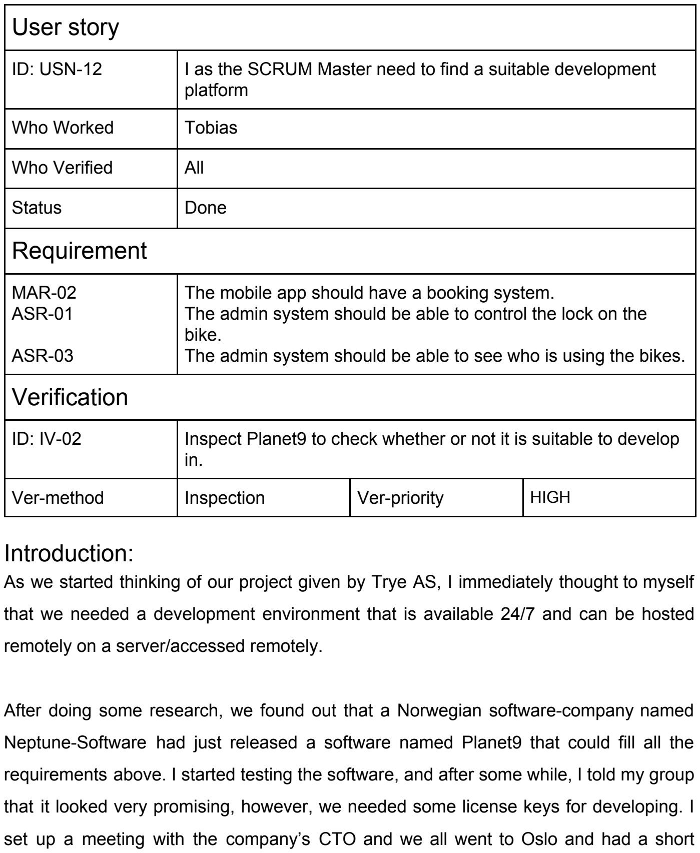
User info sequence



Trye app class diagram



A.4.3 USN-12 Choosing a suitable development platform



User story			
ID: USN-12	I as the SCRUM Master need to find a suitable development platform		
Who Worked	Tobias		
Who Verified	All		
Status	Done		
Requirement			
MAR-02 ASR-01 ASR-03	The mobile app should have a booking system. The admin system should be able to control the lock on the bike. The admin system should be able to see who is using the bikes.		
Verification			
ID: IV-02	Inspect Planet9 to check whether or not it is suitable to develop in.		
Ver-method	Inspection	Ver-priority	HIGH

Introduction:

As we started thinking of our project given by Trye AS, I immediately thought to myself that we needed a development environment that is available 24/7 and can be hosted remotely on a server/accessed remotely.

After doing some research, we found out that a Norwegian software-company named Neptune-Software had just released a software named Planet9 that could fill all the requirements above. I started testing the software, and after some while, I told my group that it looked very promising, however, we needed some license keys for developing. I set up a meeting with the company's CTO and we all went to Oslo and had a short meeting with Njål (the CTO). He gave us 10 free development licenses so that we could build our app alongside some tips and tricks.

Planet9 can be described as a “new generation” of software development as it’s main focus is on being Low-code. This means that the solution and the complexity of the problems are in focus, without having to rely on a syntax-perfect program. Low-code does not imply no-code, and to get the functionality you would want as a developer you need to code small scripts.

Method Used:

Planet9 is built upon many open-source standards such as OpenUI5 which was released the summer of 2019 and is an open-source framework for developing javaScript applications for any phone, any device, and computer. It also makes use of RESTAPI’s, which enables us to connect with all the data sources we need, simple. Combining this with built-in support for remote databases and it is accessible from everywhere, even on mobile, it has been a smart choice.

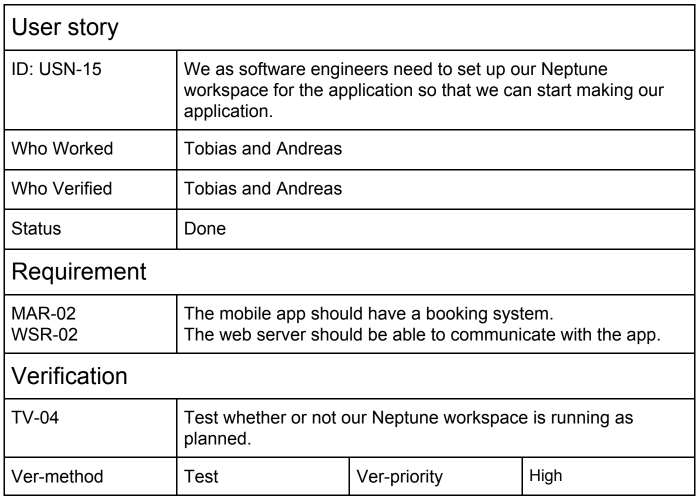
Conclusion:

After using the software development environment for nearly two months I have grown very fond of it. Especially how easy it is to release apps is very useful. Planet9 is also very focused on different user roles and their accessibility. A role can be customized, and there will only be given access to whatever part of the environment the role has given access to. Enables us to do safe beta-tests without having to be afraid of the security aspects of our platform.

Literature:

https://community.neptune-software.com/documentation/planet9

A.4.4 USN-15 Setting up our Neptune work space (Planet9)



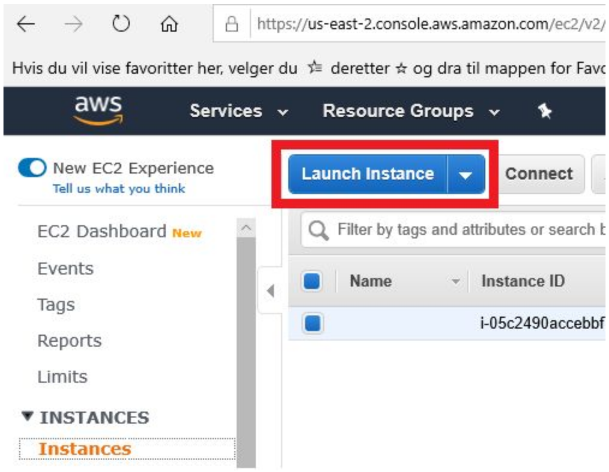
User story			
ID: USN-15	We as software engineers need to set up our Neptune workspace for the application so that we can start making our application.		
Who Worked	Tobias and Andreas		
Who Verified	Tobias and Andreas		
Status	Done		
Requirement			
MAR-02 WSR-02	The mobile app should have a booking system. The web server should be able to communicate with the app.		
Verification			
TV-04	Test whether or not our Neptune workspace is running as planned.		
Ver-method	Test	Ver-priority	High

Setting up Planet9 to run on Amazon Web Services

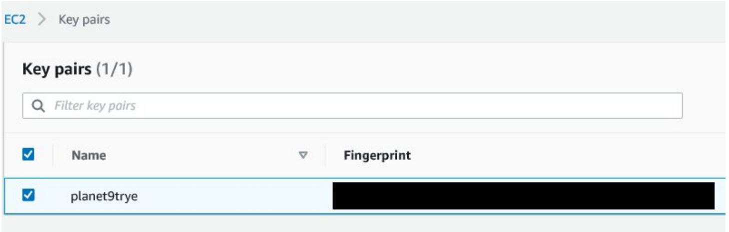
Running an instance of Planet9 in EC2 Linux Server

Author: Andreas
Additions: Tobias

Our first step in the right direction for getting Planet9 to run on an Amazon EC2 server was to create an AWS account (Amazon Web Services) and launch a EC2 instance.



When the instance was done initializing, a KP (Key Pair) was created and we downloaded it locally. The KP is essential for connecting to the server/instance with e.g. Putty with an SSH connection.



EC2 > Key pairs

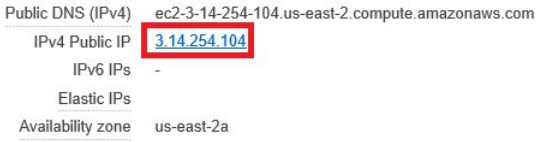
Key pairs (1/1)

Filter key pairs

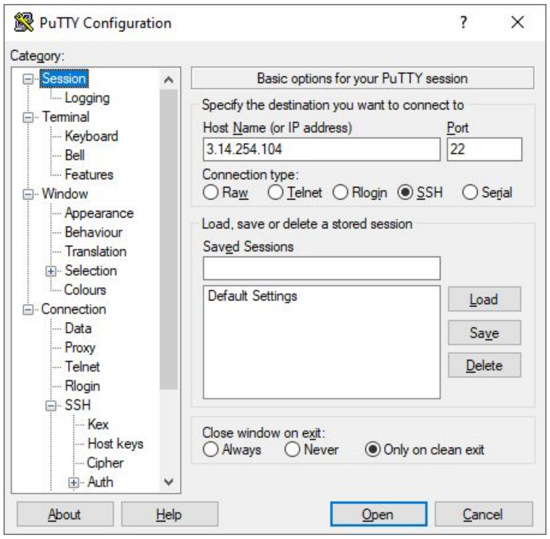
☑	Name	Fingerprint
☑	planet9trye	[redacted]

The KP was downloaded in a PEM-file and then converted to a PPK file(Putty Private Key file).

Next step was to download putty and find the public IP address of our instance:



Public DNS (IPv4)	ec2-3-14-254-104.us-east-2.compute.amazonaws.com
IPv4 Public IP	3.14.254.104
IPv6 IPs	-
Elastic IPs	
Availability zone	us-east-2a



PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 3.14.254.104

Port: 22

Connection type: ○ Raw ○ Telnet ○ Rlogin ● SSH ○ Serial

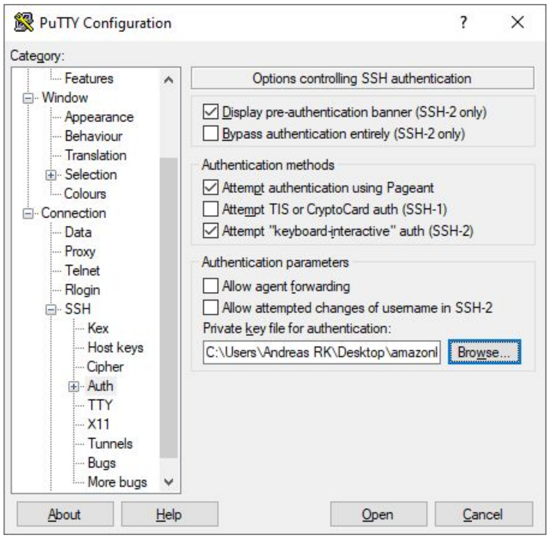
Load, save or delete a stored session

Saved Sessions

Default Settings

Close window on exit: ○ Always ○ Never ● Only on clean exit

Then, we went ahead and entered the IP address of the instance in the Host Name field, specified Port 22, and enabled SSH.



Options controlling SSH authentication

☑ Display pre-authentication banner (SSH-2 only)
☐ Bypass authentication entirely (SSH-2 only)

Authentication methods

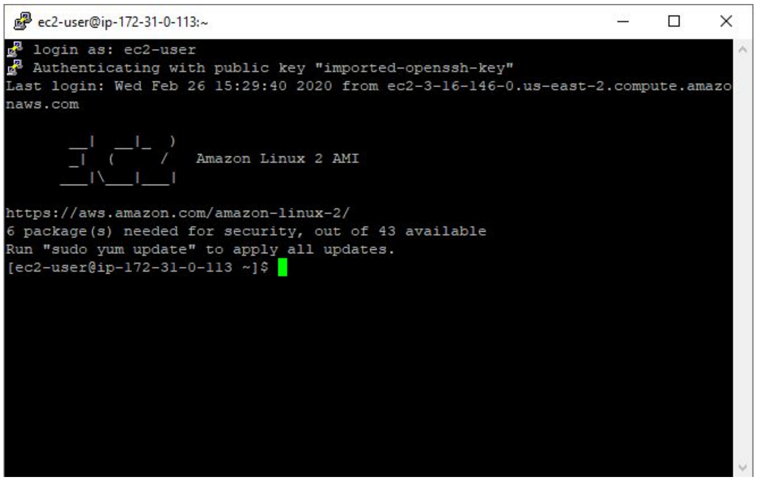
☑ Attempt authentication using Pageant
☐ Attempt TIS or CryptoCard auth (SSH-1)
☑ Attempt "keyboard-interactive" auth (SSH-2)

Authentication parameters

☐ Allow agent forwarding
☐ Allow attempted changes of username in SSH-2

Private key file for authentication:
C:\Users\Andreas RK\Desktop\amazonl

Then, as we had already downloaded the correct KP and converted it to a PPK, all we had to do was to browse our way to the location of our PPK. To find the SSH settings, we had to navigate from Connection -> SSH -> Authentication Parameters -> Private key file for authentication.

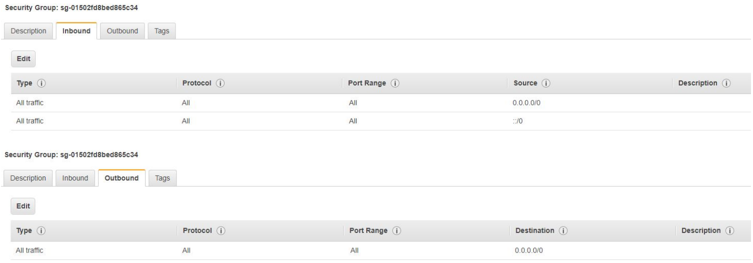


```
login as: ec2-user
Authenticating with public key "imported-openssh-key"
Last login: Wed Feb 26 15:29:40 2020 from ec2-3-16-146-0.us-east-2.compute.amazo
naws.com

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
6 package(s) needed for security, out of 43 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-0-113 ~]$
```

Here we are logged in to our server using the root user ec2-user. The next step is to make the Planet9 software run on the server. I (Andreas) had the documents needed on my dropbox, and here is the list of commands used to install Planet9 and make it run using Linux shell commands

1. $ wget dropbox link 1 (The Planet9 software)
2. $ wget dropbox link 2 (The SQLite node)
3. $ chmod +x planet9-linux (Make program executable)
4. $ sudo yum -y install tmux (Install tmux to keep the program running after disconnecting)
5. $ tmux new -s planet9 (Create new tmux session to keep Planet9 alive 24/7)
6. $ ./planet9-linux (Execute planet9, it will run by default on port 8080)

As we got Planet9 to run on our server, we needed to make changes to the firewall so that it could be reached remotely/make the server public. First of all, we started by making the server public for all IP-addresses both inbound and outbound, as well as making it communicate on all ports and with any protocol.
Note: We might change the firewall settings later in the development due to security reasons.



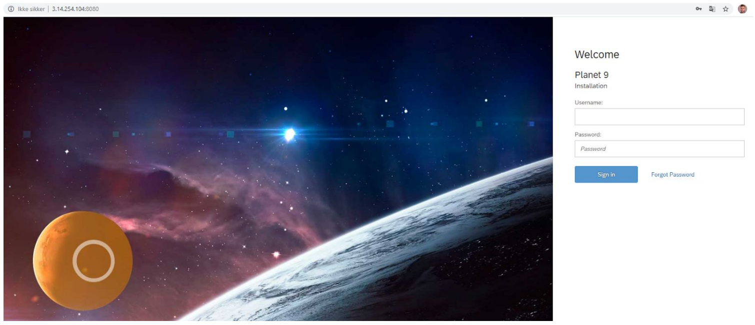
Security Group: sg-01502fd8bed865c34

Type	Protocol	Port Range	Source	Description
All traffic	All	All	0.0.0.0/0	
All traffic	All	All	::/0	

Security Group: sg-01502fd8bed865c34

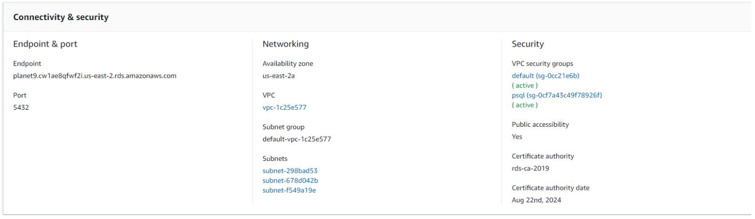
Type	Protocol	Port Range	Destination	Description
All traffic	All	All	0.0.0.0/0	

Now that Planet9 is up and running and the server is public, we simply just enter the IP-address of our server followed with a colon 8080 to specify port 8080 and we are in! Planet9 is now accessible with any device/any browser. Below is a picture showing off the homepage of Planet9 in chrome.



Connecting to an amazon database instance

Here we do the same as we did with the server instance, just that we launch a database instance instead. We also make it public and add firewall rules.



Connectivity & security

Endpoint & port

Endpoint
planet9.cw1ae8qfwf2i.us-east-2.rds.amazonaws.com

Port
5432

Networking

Availability zone
us-east-2a

VPC
vpc-1c25e577

Subnet group
default-vpc-1c25e577

Subnets
subnet-298bad53
subnet-678d042b
subnet-f549a19e

Security

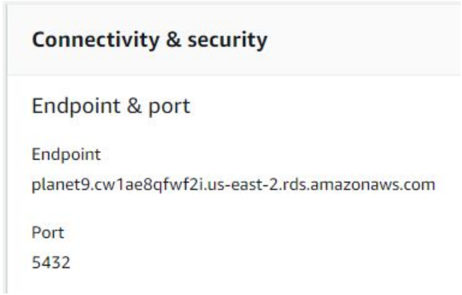
VPC security groups
default (sg-0cc21e6b)
(active)
psql (sg-0cf7a43c49f78926f)
(active)

Public accessibility
Yes

Certificate authority
rds-ca-2019

Certificate authority date
Aug 22nd, 2024

In this screenshot, you can see the public IP endpoint and the port 5432 which is standard for PostgreSQL.



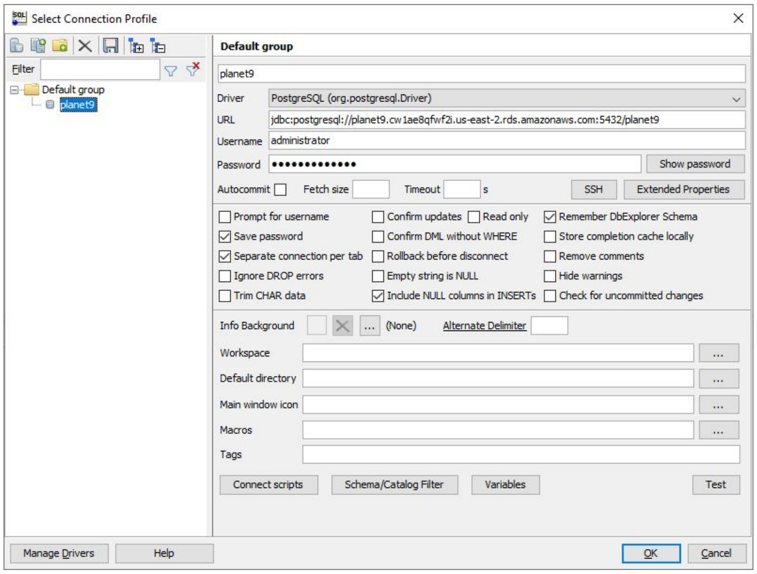
Connectivity & security

Endpoint & port

Endpoint
planet9.cw1ae8qfwf2i.us-east-2.rds.amazonaws.com

Port
5432

We've also created a root user with a password for authentication when connecting to the DB, for then to connect to it using SQL Workbench.



Select Connection Profile

Default group

planet9

Driver: PostgreSQL (org.postgresql.Driver)

URL: jdbc:postgresql://planet9.cw1ae8qfwf2i.us-east-2.rds.amazonaws.com:5432/planet9

Username: administrator

Password: •••••••••••••

Autocommit □ Fetch size Timeout s

□ Prompt for username □ Confirm updates □ Read only ☑ Remember DbExplorer Schema
☑ Save password □ Confirm DML without WHERE □ Store completion cache locally
☑ Separate connection per tab □ Rollback before disconnect □ Remove comments
□ Ignore DROP errors □ Empty string is NULL □ Hide warnings
□ Trim CHAR data ☑ Include NULL columns in INSERTs □ Check for uncommitted changes

Info Background (None) Alternate Delimiter

Workspace

Default directory

Main window icon

Macros

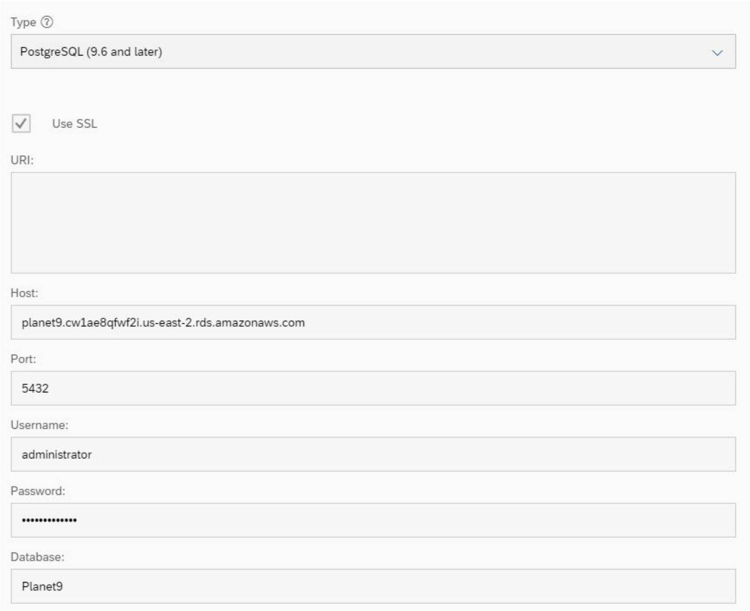
Tags

Here we put in the IP, the port and the password we created.

Then we had to run these commands according to the Planet9 installation manual:

1. CREATE DATABASE planet9;
2. CREATE SCHEMA planet9;

Then we log in to Planet9 and go to the database tab where we fill in all the needed information and click save. After that is done, the database is successfully connected.



Type ⓘ

PostgreSQL (9.6 and later)

☑ Use SSL

URI:

Host:

planet9.cw1ae8qfwf2i.us-east-2.rds.amazonaws.com

Port:

5432

Username:

administrator

Password:

•••••••••••••

Database:

Planet9

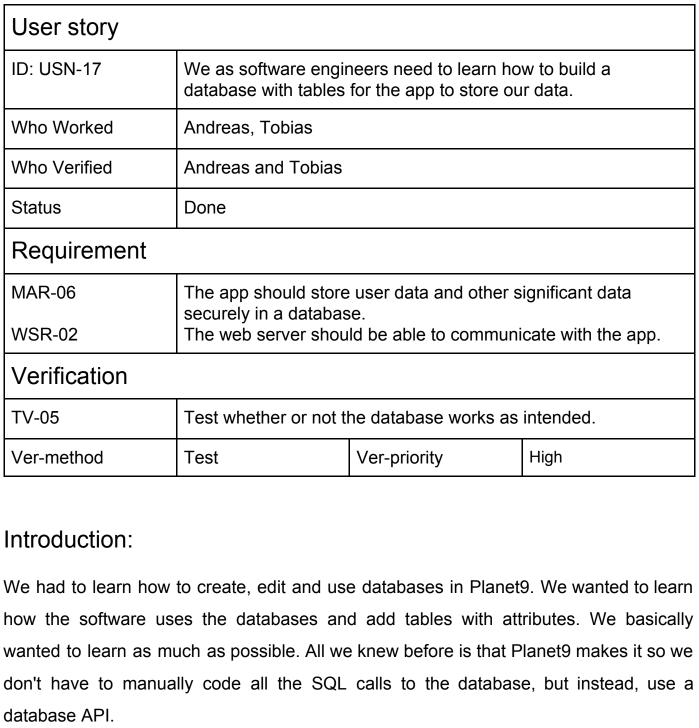
Conclusion:

Even though the setup might look easy, we struggled a lot and spent many hours to make it work. Both the server and the database crashed several times, which meant we had to do all our work all over again. Besides, setting up the security and firewall for the server is something we have never done before, so it took some time to figure that out. However, we can now conclude that the server and database are both running securely on the Amazon Cloud and that they are successfully connected. Both Andreas and Tobias have tested and we can confirm that it is working as intended.

Literature:

https://community.neptune-software.com/documentation/instructions

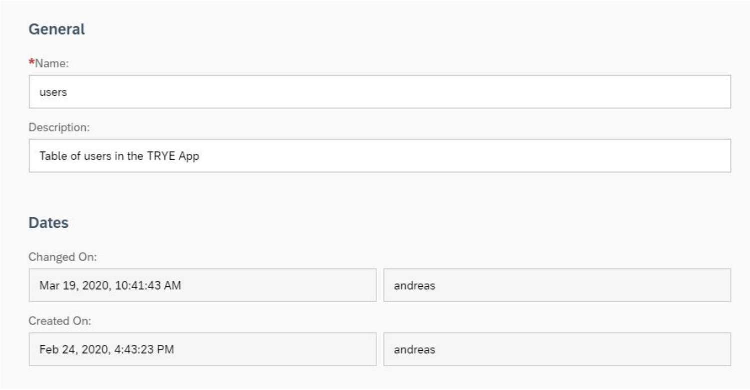
A.4.5 USN-17 Building databases in Planet9



User story			
ID: USN-17	We as software engineers need to learn how to build a database with tables for the app to store our data.		
Who Worked	Andreas, Tobias		
Who Verified	Andreas and Tobias		
Status	Done		
Requirement			
MAR-06 WSR-02	The app should store user data and other significant data securely in a database. The web server should be able to communicate with the app.		
Verification			
TV-05	Test whether or not the database works as intended.		
Ver-method	Test	Ver-priority	High

Introduction:

We had to learn how to create, edit and use databases in Planet9. We wanted to learn how the software uses the databases and add tables with attributes. We basically wanted to learn as much as possible. All we knew before is that Planet9 makes it so we don't have to manually code all the SQL calls to the database, but instead, use a database API.



General

*Name:

users

Description:

Table of users in the TRYE App

Dates

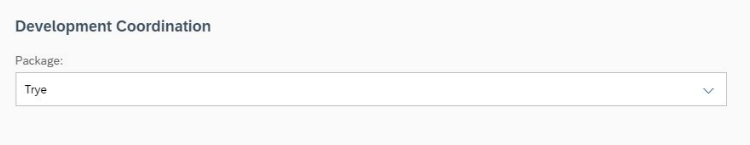
Changed On:

Mar 19, 2020, 10:41:43 AM	andreas

Created On:

Feb 24, 2020, 4:43:23 PM	andreas

The first thing to be done is to define a new table, give it a name and a description.

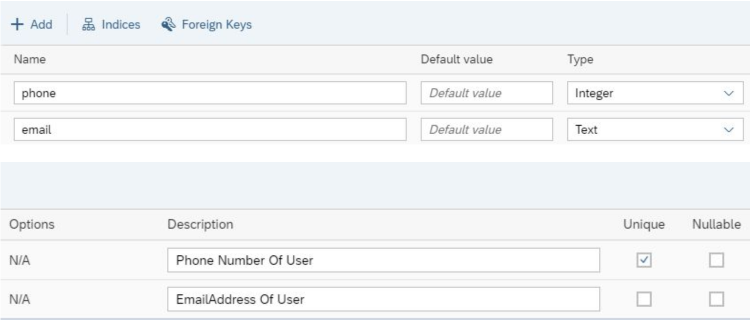


Development Coordination

Package:

Trye

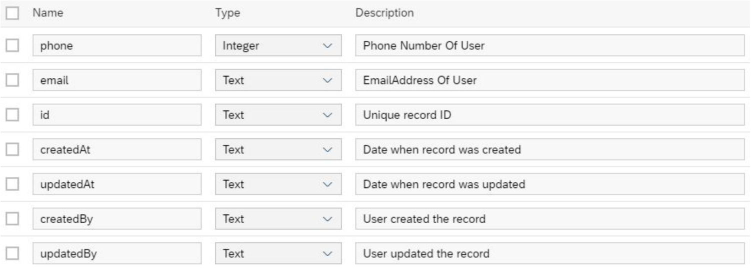
Then add the database to our package so it is easier to find when we create API for the database and add use it en the app editor.



Name	Default value	Type
phone	Default value	Integer
email	Default value	Text

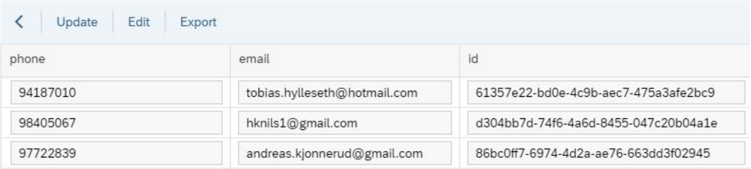
Options	Description	Unique	Nullable
N/A	Phone Number Of User	☑	□
N/A	EmailAddress Of User	□	□

Then it is possible to start adding properties to our database, including the name of the columns and adding the data type and whether or not it is unique.



□	Name	Type	Description
□	phone	Integer	Phone Number Of User
□	email	Text	EmailAddress Of User
□	id	Text	Unique record ID
□	createdAt	Text	Date when record was created
□	updatedAt	Text	Date when record was updated
□	createdBy	Text	User created the record
□	updatedBy	Text	User updated the record

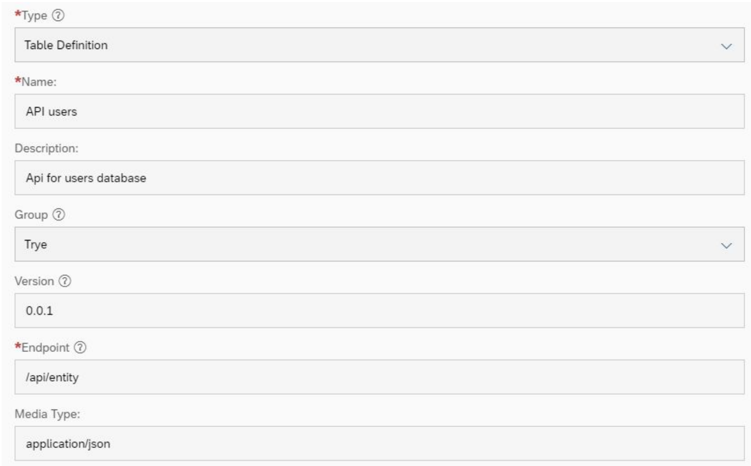
Now it is possible to display the table and see that planet9 will automatically add an id to each entry and the dates and time the record was made and updated.



phone	email	id
94187010	tobias.hylleseth@hotmail.com	61357e22-bd0e-4c9b-aec7-475a3afe2bc9
98405067	hknils1@gmail.com	d304bb7d-74f6-4a6d-8455-047c20b04a1e
97722839	andreas.kjonnerud@gmail.com	86bc0ff7-6974-4d2a-ae76-663dd3f02945

Here are some entries where you can see the unique IDs Planet9 has created to reference the different users.

Now it is possible to connect the database to the API. To do this we need to create a new database API



*Type ⓘ

Table Definition

*Name:

API users

Description:

Api for users database

Group ⓘ

Trye

Version ⓘ

0.0.1

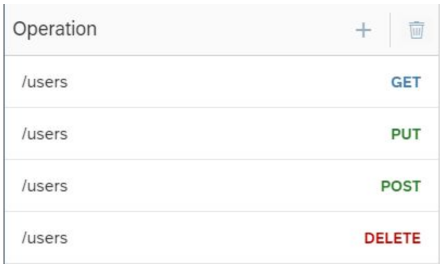
*Endpoint ⓘ

/api/entity

Media Type:

application/json

Here, you can see the four different Operations available for the user's table. If you press on each operation individually, you can customize what it executes based on what's needed in our app. When done, you can run the Operations with a built-in API in the App Designer.



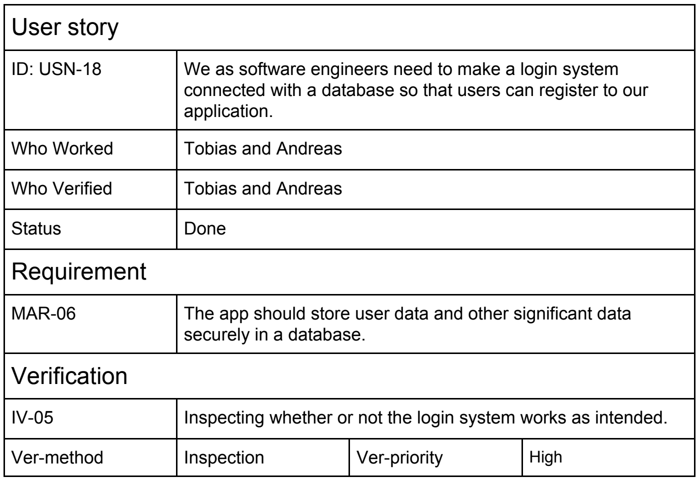
Operation	+ 🗑
/users	GET
/users	PUT
/users	POST
/users	DELETE

The 4 standard Operations for our database API is GET (Read data), PUT (Add data), POST (Update table) and DELETE (Delete data). As probably noticed, this is a slight change to the regular 4 SQL operations (SELECT, UPDATE, INSERT, DELETE).

Conclusion:

The database and the API have now successfully been set up and are ready to be used in the app designer.

A.4.6 USN-18 Creating a login system for our application



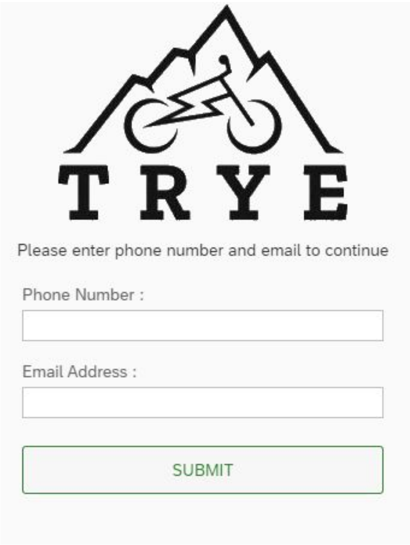
User story			
ID: USN-18	We as software engineers need to make a login system connected with a database so that users can register to our application.		
Who Worked	Tobias and Andreas		
Who Verified	Tobias and Andreas		
Status	Done		
Requirement			
MAR-06	The app should store user data and other significant data securely in a database.		
Verification			
IV-05	Inspecting whether or not the login system works as intended.		
Ver-method	Inspection	Ver-priority	High

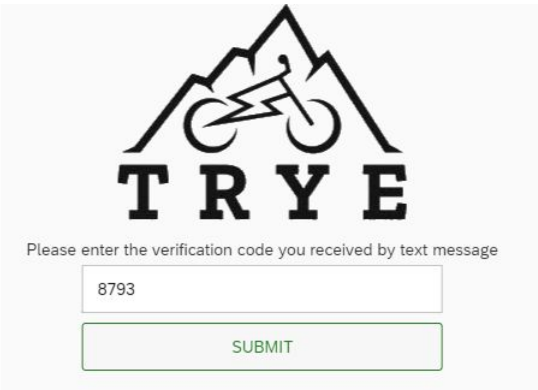
Introduction:

One of the most important features of an app is in our mind a well thought of login-system. After looking at our competitors and how they do login’s we decided to keep it simple and only ask for phone number and e-mail in our first alpha version. This way we keep user input to a minimum, with still getting important information. As we are building a mobile app that is selling quite expensive services (500NOK to several thousand if a user rents a bike for weeks) we took the decision together with the guys at Trye to send the user a one-time-password to the phone number given in the login-form. With the use of OTP (One-time-password), we skip the process of having to remember a password that is long forgotten in exchange for the SMS fee.

Method Used:

The idea is simple. If a user has put in his/her credentials (email and phone number) and successfully put in the correct OTP sent to his/her phone the user will be added to the database. However, due to technical difficulties, we decided to record user data when they press register and as the OTP gets sent, but have a “Verified” Boolean that says whether or not a user is verified.

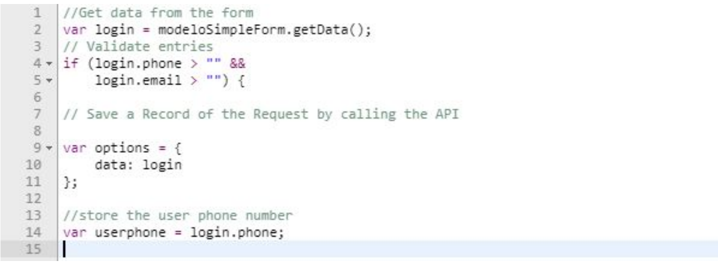




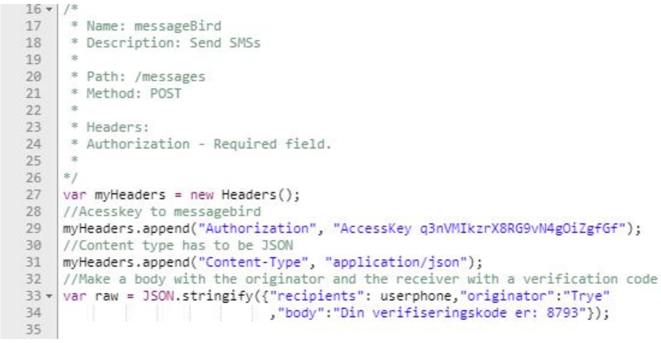
Please enter the verification code you received by text message

8793

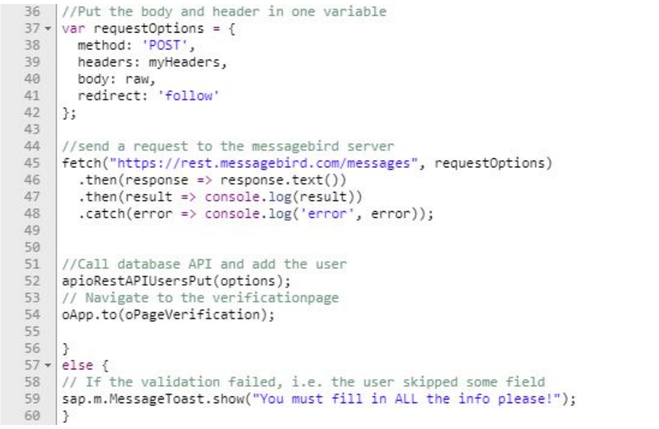
The javaScript code for the submit form with comments.



```
//Get data from the form
var login = modeloSimpleForm.getData();
// Validate entries
if (login.phone > "" &&
    login.email > "") {

// Save a Record of the Request by calling the API

var options = {
    data: login
};

//store the user phone number
var userphone = login.phone;

```

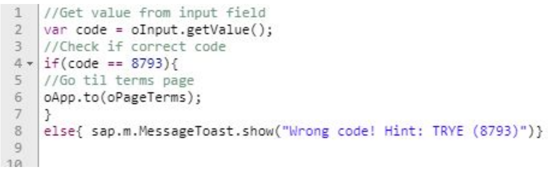


```
/*
 * Name: messageBird
 * Description: Send SMSs
 *
 * Path: /messages
 * Method: POST
 *
 * Headers:
 * Authorization - Required field.
 *
*/
var myHeaders = new Headers();
//Acesskey to messagebird
myHeaders.append("Authorization", "AccessKey q3nVMIkzrX8RG9vN4gOiZgfGf");
//Content type has to be JSON
myHeaders.append("Content-Type", "application/json");
//Make a body with the originator and the receiver with a verification code
var raw = JSON.stringify({"recipients": userphone,"originator":"Trye"
                         ,"body":"Din verifiseringskode er: 8793"});

```



```
//Put the body and header in one variable
var requestOptions = {
  method: 'POST',
  headers: myHeaders,
  body: raw,
  redirect: 'follow'
};

//send a request to the messagebird server
fetch("https://rest.messagebird.com/messages", requestOptions)
  .then(response => response.text())
  .then(result => console.log(result))
  .catch(error => console.log('error', error));


//Call database API and add the user
apioRestAPIUsersPut(options);
// Navigate to the verificationpage
oApp.to(oPageVerification);

}
else {
// If the validation failed, i.e. the user skipped some field
sap.m.MessageToast.show("You must fill in ALL the info please!");
}
```

The javaScript code for the verification with comments

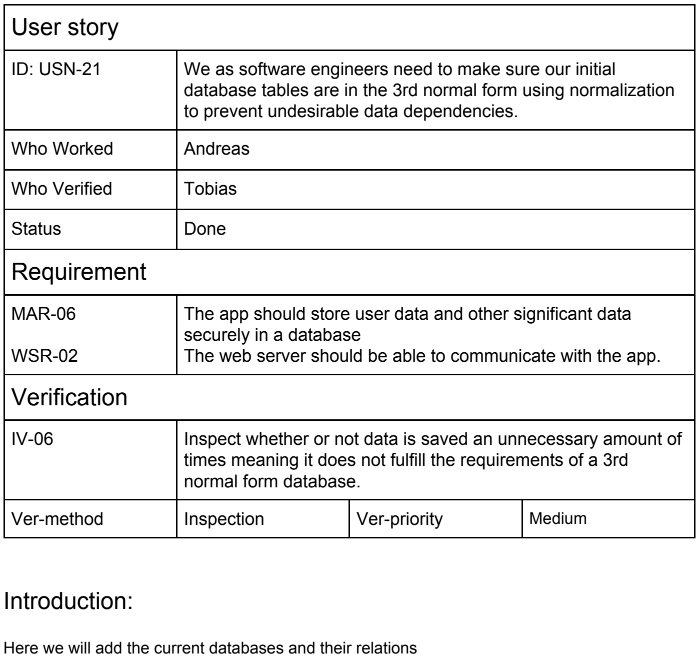


```
//Get value from input field
var code = oInput.getValue();
//Check if correct code
if(code == 8793){
//Go til terms page
oApp.to(oPageTerms);
}
else{ sap.m.MessageToast.show("Wrong code! Hint: TRYE (8793)")}

```

Conclusion:

As of now, it is a working easy to use login-system that records user info and whether the user has come through the verification-page of the app. This is a shell that we will continue to work on for our next alpha.

A.4.7 USN-21 Finalizing our initial databases

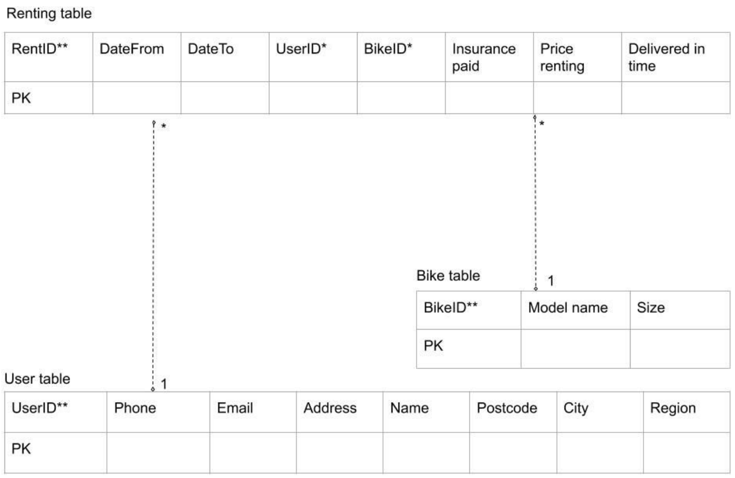


User story			
ID: USN-21	We as software engineers need to make sure our initial database tables are in the 3rd normal form using normalization to prevent undesirable data dependencies.		
Who Worked	Andreas		
Who Verified	Tobias		
Status	Done		
Requirement			
MAR-06 WSR-02	The app should store user data and other significant data securely in a database The web server should be able to communicate with the app.		
Verification			
IV-06	Inspect whether or not data is saved an unnecessary amount of times meaning it does not fulfill the requirements of a 3rd normal form database.		
Ver-method	Inspection	Ver-priority	Medium

Introduction:

Here we will add the current databases and their relations

These tables are in 3rd normal form since for the users all the data rely on the primary key and is unique to each id. Same with the bike table. In the renting table, there is also no dependencies since all the data will depend on the rent ID and will be unique for each rent. You can also see the relations between the tables where 1 user can rent many bikes, but a bike can only have one user. There is a one to many relationship with the bike table also since there can be several bookings in the bookings table with the same bike on different days.



Renting table

RentID**	DateFrom	DateTo	UserID*	BikeID*	Insurance paid	Price renting	Delivered in time
PK							

Bike table

BikeID**	Model name	Size
PK		

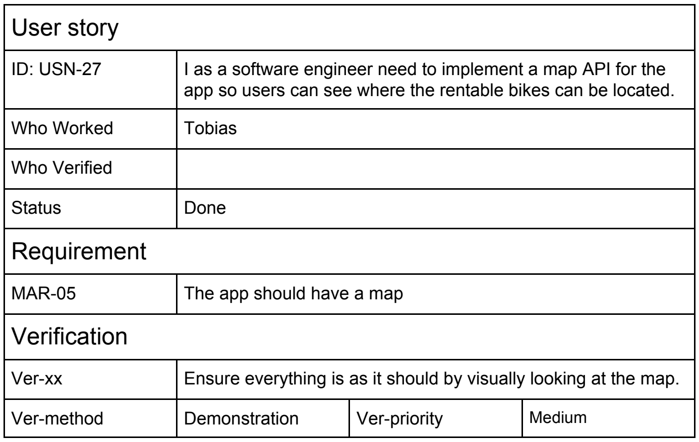
User table

UserID**	Phone	Email	Address	Name	Postcode	City	Region
PK							

Conclusion:

Tables are in 3rd normal form confirmed by the formal method of intuition.

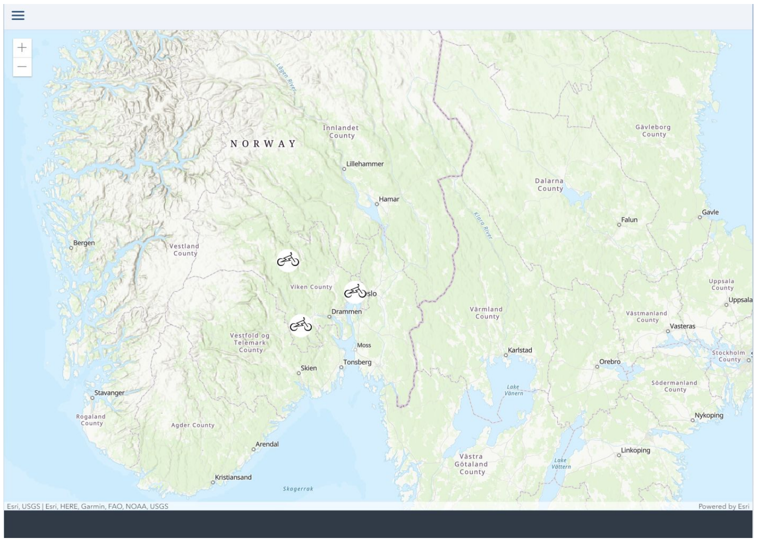
A.4.8 USN-27 Setting up an interactive map in the application



User story			
ID: USN-27	I as a software engineer need to implement a map API for the app so users can see where the rentable bikes can be located.		
Who Worked	Tobias		
Who Verified			
Status	Done		
Requirement			
MAR-05	The app should have a map		
Verification			
Ver-xx	Ensure everything is as it should by visually looking at the map.		
Ver-method	Demonstration	Ver-priority	Medium

Introduction:

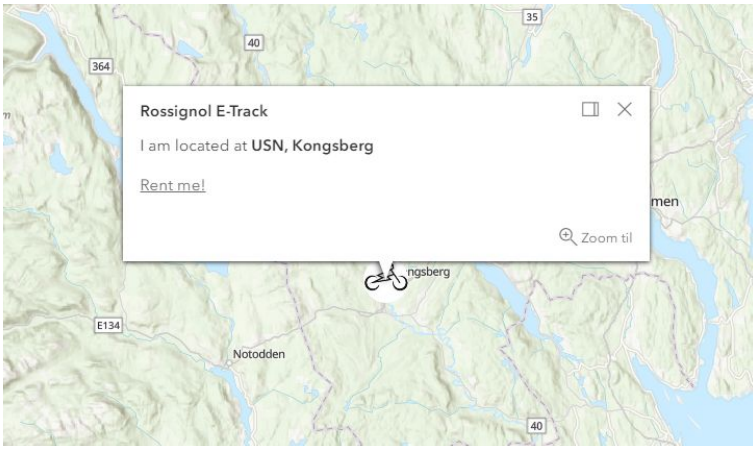
The easiest way to visualize where the rentable bicycles are is with a map. The map solution we have chosen for our first prototype is ESRI maps, which is open-source and free to use.



Method Used:

The map is created with javaScript, and it initialized when the user clicks “Verify One-Time Password”. We had difficulties with creating an “on-load” function, meaning that the map should be initialized when a user is navigating to the main screen. Our solution to this was to have the entire map script put on to the button referring to the main menu.

We have used an API to get ESRI Maps implemented in our app and used symbols and pop-up templates to create clickable bicycles that says where it is located with a link to rent it. The payment solutions and renting-part of the program are not yet created but will be done in time for the third presentation.



Rossignol E-Track

I am located at **USN, Kongsberg**

Rent me!

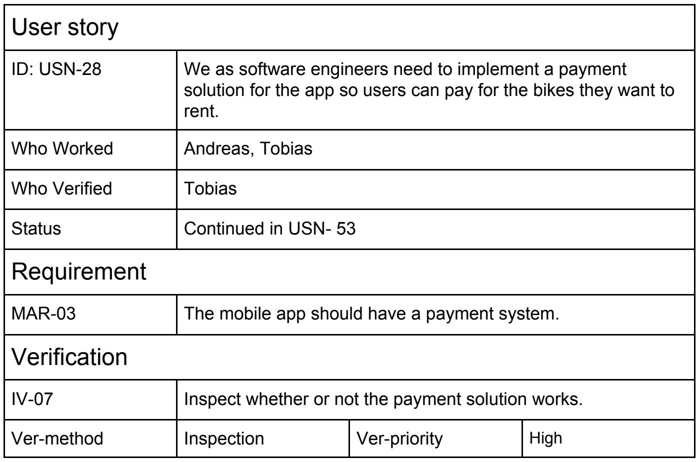
Conclusion:

We have a working map in our app that visualizes the position of the rentable bicycles, the localization of the bikes are implemented so that when the actual bikes send their location, the map will update the location of the bikes. They bike icons are clickable and when clicked, they give the user their location as well as a “zoom to” button that zooms in at that location. In addition, we added a link to Trye’s webpage and their ordinary way of renting the bicycle.

Literature:

https://www.esri.com/en-us/home
https://www.arcgis.com/index.html

A.4.9 USN-28 Implementing a payment solution



User story			
ID: USN-28	We as software engineers need to implement a payment solution for the app so users can pay for the bikes they want to rent.		
Who Worked	Andreas, Tobias		
Who Verified	Tobias		
Status	Continued in USN- 53		
Requirement			
MAR-03	The mobile app should have a payment system.		
Verification			
IV-07	Inspect whether or not the payment solution works.		
Ver-method	Inspection	Ver-priority	High

Introduction:

Having a payment solution is crucial to the useability of our application. The entire point of the application is to make it easy for a customer/user to rent a bike and without a payment solution, Trye would not be profitable.

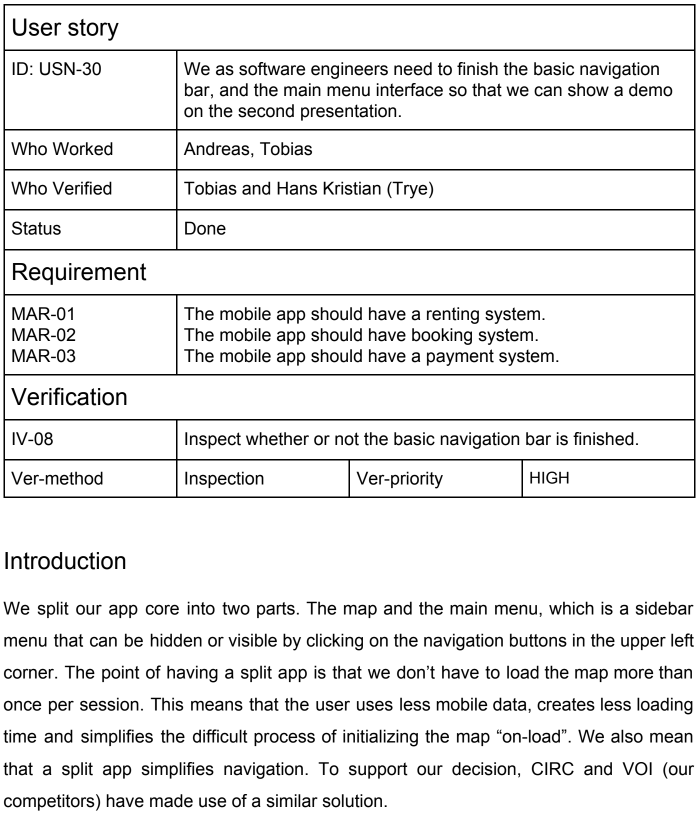
Conclusion:

This User Story documentation is written quite a while after it was originally started, and due to other functionalities having higher priority this User Story was disbanded.

However, after cooperating with Trye we found out that they are already using a payment solution we can integrate into our app (Booqable).

This user story is continued in USN-53.

A.4.10 USN-30 Creating a user friendly main menu navigation bar

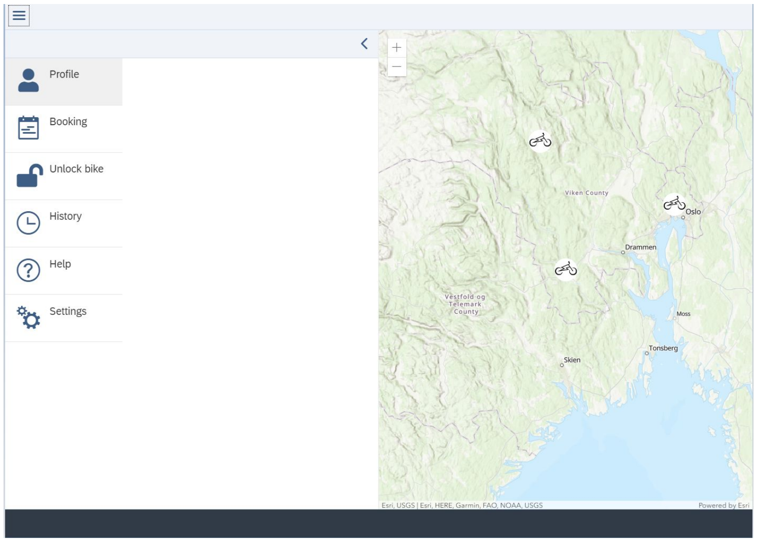


User story			
ID: USN-30	We as software engineers need to finish the basic navigation bar, and the main menu interface so that we can show a demo on the second presentation.		
Who Worked	Andreas, Tobias		
Who Verified	Tobias and Hans Kristian (Trye)		
Status	Done		
Requirement			
MAR-01 MAR-02 MAR-03	The mobile app should have a renting system. The mobile app should have booking system. The mobile app should have a payment system.		
Verification			
IV-08	Inspect whether or not the basic navigation bar is finished.		
Ver-method	Inspection	Ver-priority	HIGH

Introduction

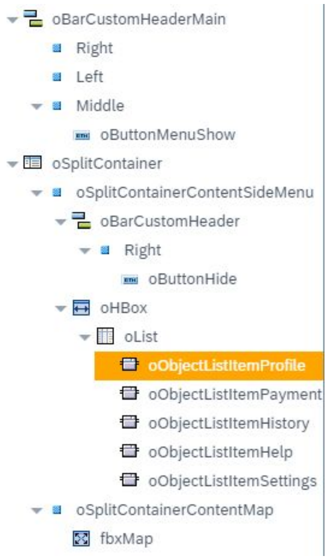
We split our app core into two parts. The map and the main menu, which is a sidebar menu that can be hidden or visible by clicking on the navigation buttons in the upper left corner. The point of having a split app is that we don't have to load the map more than once per session. This means that the user uses less mobile data, creates less loading time and simplifies the difficult process of initializing the map "on-load". We also mean that a split app simplifies navigation. To support our decision, CIRC and VOI (our competitors) have made use of a similar solution.

Sidebar menu



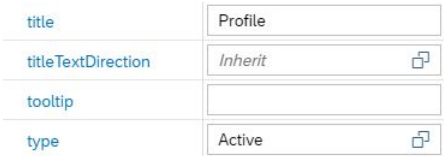
There are 6 different tabs to click. They are:

1) The Profile page where users can view and change their personal information.
2) The Booking tab where a user can book a bike.
3) The Unlock bike tab is where users can unlock a bike they have successfully rented/paid for.
4) The History tab where the user can look at their renting history.
5) The Help tab where the user gets shown how to get help if needed.
6) The Settings tab where the user can change general app settings.


oBarCustomHeaderMain
Right
Left
Middle
oButtonMenuShow
oSplitContainer
oSplitContainerContentSideMenu
oBarCustomHeader
Right
oButtonHide
oHBox
oList
oObjectListItemProfile
oObjectListItemPayment
oObjectListItemHistory
oObjectListItemHelp
oObjectListItemSettings
oSplitContainerContentMap
fbxMap

The image above shows how the menu is built with OpenUI5. As you can see, this way of developing apps makes use of objects that you can drag/drop and sort/rename as one would like. The app development is hierarchical, and the “higher” an object is, the sooner the object will be loaded. For example, our start page is the “top” object.

Developing low-code saves us a lot of time that normally would be spent coding HTML. The sidebar is located inside a SAP.M HBox, which means it will always fill the entire sidebar page. The HBox enables the objects to “auto adjust” vertically, and inside the list, we have the sidebar objects.



title	Profile
titleTextDirection	Inherit
tooltip	
type	Active

The picture above is in the properties section of the Profile object. In our navigation bar, we started by naming the first sidebar menu object “Profile”, set it as an active object and made it clickable. For each of the six sidebar menu objects, we had to embed one line of javaScript code to open their respective dialogue box.
Note: The sidebar menu object can be looked upon as buttons referring to the respective page.

The javaScript needed to open our oDialog objects were:
oDialog”tabname”.open();

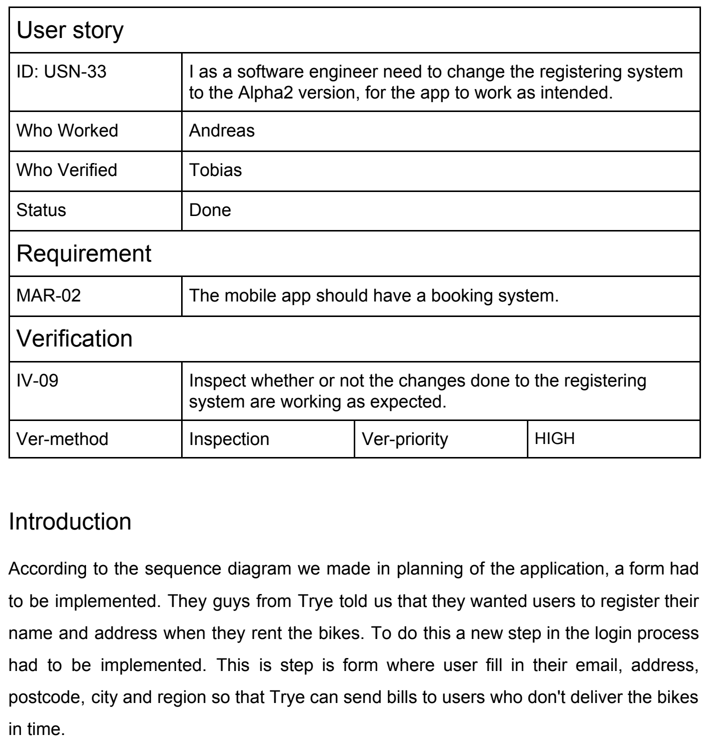
For our Profile tab, we embeded: oDialogProfile.open();

To hide and show the content of our sidebar-menu, we made the use of these two javaScript lines.
oSplitContainer.setShowSecondaryContent(true); //Show sidebar button
oSplitContainer.setShowSecondaryContent(false); //Hide sidebar button

Conclusion:

We have now successfully created the pages needed in our app. The feature has been tested on several mobile phones/different browsers and is user friendly and easy to use. The next step in our development is working on each of the six submenus.

A.4.11 USN-33 Changing the registering system to the Alpha2 version

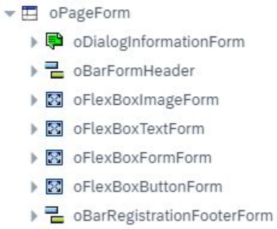


User story			
ID: USN-33	I as a software engineer need to change the registering system to the Alpha2 version, for the app to work as intended.		
Who Worked	Andreas		
Who Verified	Tobias		
Status	Done		
Requirement			
MAR-02	The mobile app should have a booking system.		
Verification			
IV-09	Inspect whether or not the changes done to the registering system are working as expected.		
Ver-method	Inspection	Ver-priority	HIGH

Introduction

According to the sequence diagram we made in planning of the application, a form had to be implemented. They guys from Trye told us that they wanted users to register their name and address when they rent the bikes. To do this a new step in the login process had to be implemented. This is step is form where user fill in their email, address, postcode, city and region so that Trye can send bills to users who don't deliver the bikes in time.

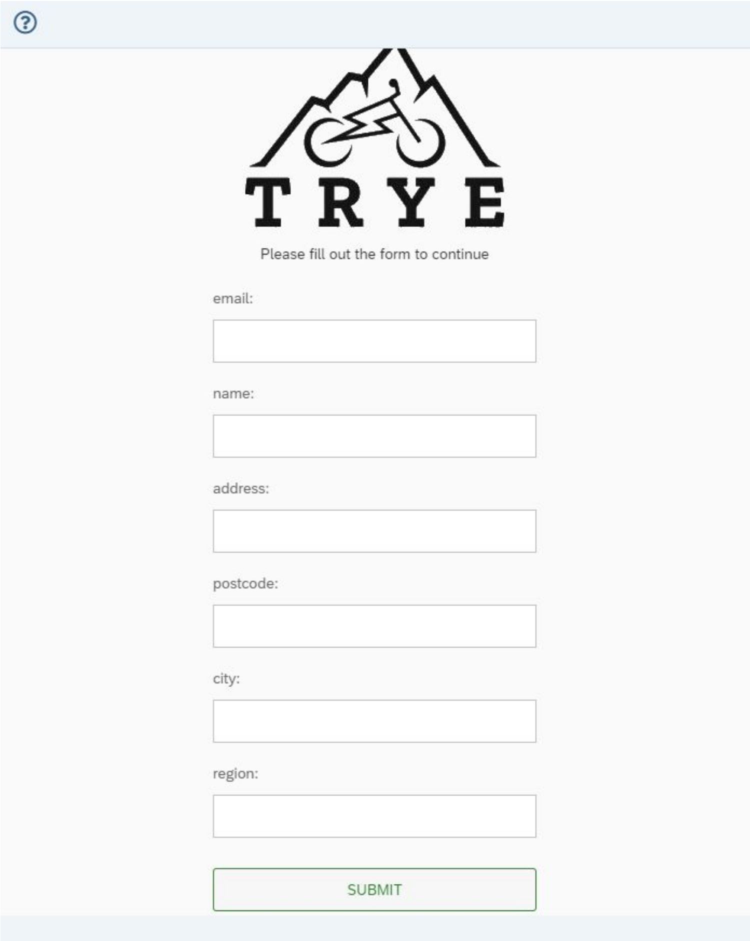
The form page:



```
oPageForm
    oDialogInformationForm
    oBarFormHeader
    oFlexBoxImageForm
    oFlexBoxTextForm
    oFlexBoxFormForm
    oFlexBoxButtonForm
    oBarRegistrationFooterForm
```

The buildup of the form page is quite similar to the first registration page. It has a header and footer with buttons, and an information dialogue to help customers. It also has flexboxes to hold the image, the information text, the form and the accept button.

Visually the form looks like this in the app:



Please fill out the form to continue

email:

name:

address:

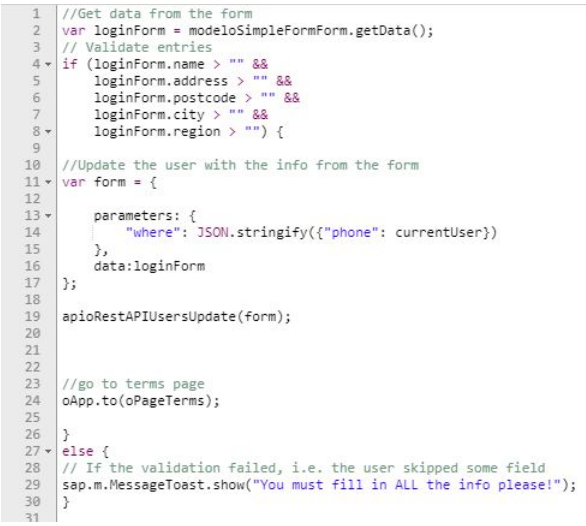
postcode:

city:

region:

Now for the backend code to make be able to put the form data into the database.

When the users register they get added directly into the database with only their phone number added as their primary key. This key we use to know which use to find the user we should update. When the user login we use a global javaScript variable which holds the current user's phone number. This key we use for our POST method. The POST method is a CRUD endpoint method that updates data in the database based on a WHERE parameter. So the code when the submit button is clicked looks like this:

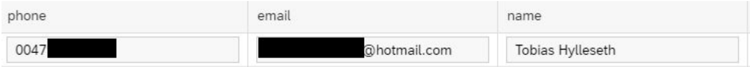


```
//Get data from the form
var loginForm = modeloSimpleFormForm.getData();
// Validate entries
if (loginForm.name > "" &&
    loginForm.address > "" &&
    loginForm.postcode > "" &&
    loginForm.city > "" &&
    loginForm.region > "") {

//Update the user with the info from the form
var form = {

    parameters: {
        "where": JSON.stringify({"phone": currentUser})
    },
    data:loginForm
};

apioRestAPIUsersUpdate(form);



//go to terms page
oApp.to(oPageTerms);

}
else {
// If the validation failed, i.e. the user skipped some field
sap.m.MessageToast.show("You must fill in ALL the info please!");
}

```

First, we collect the data from the form and check if all the fields have been filled our, or we give the customer a message that they need to fill in all the data. If all the data has been filled in, we use the global variable currentUser in the WHERE sentence for the update method to know for which user the data should be updated. Then we send the customer to the next page which is the terms page.

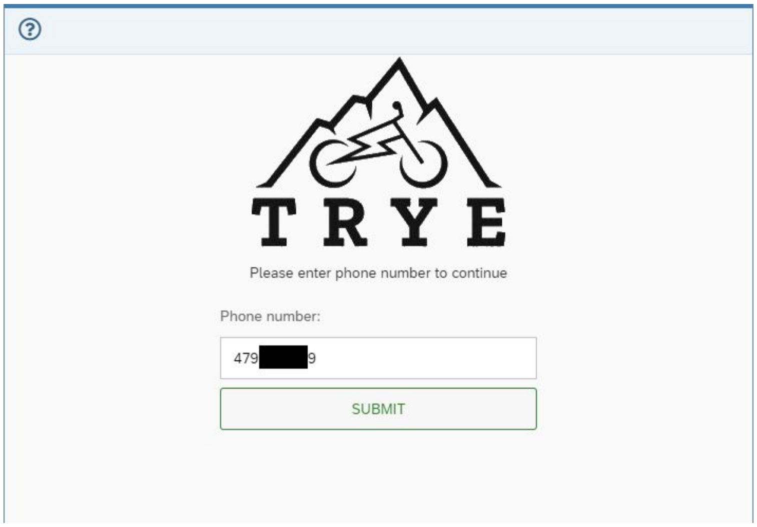
Now to demonstrate the process, I will register myself as a user and fill out the form to verify that is works.

1. Step: the phone registration page



phone	email	name
0047[redacted]	[redacted]@hotmail.com	Tobias Hylleseth

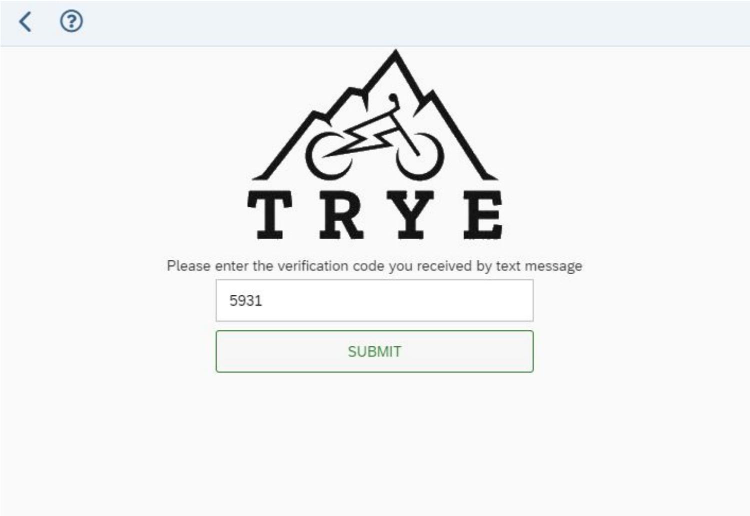
Now our database only contains one user which is Tobias. So I have the database browser open and the app window open:



Please enter phone number to continue

Phone number: 479[redacted]9

Now my phone number is filled in the input field, and submit is clicked.

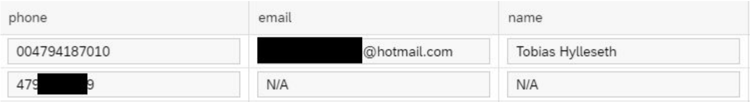


Please enter the verification code you received by text message

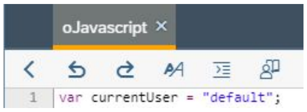
5931

SUBMIT

Then I fill in the one time password. Now I should appear in the database with my phone number and default values:

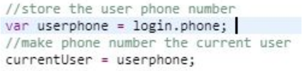


phone	email	name
004794187010	[redacted]@hotmail.com	Tobias Hylleseth
47[redacted]9	N/A	N/A



```
var currentUser = "default";
```

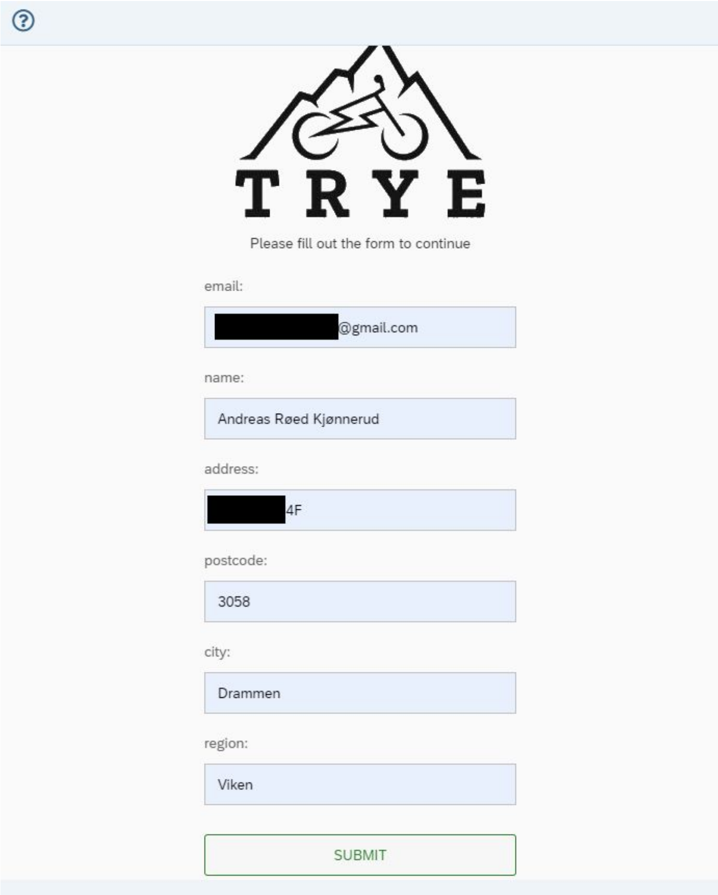
The global javaScript variable is set to the current phone number



```
//store the user phone number
var userphone = login.phone;
//make phone number the current user
currentUser = userphone;
```

We can see that the user has been created for the phone number I entered.

Now for the form:



Please fill out the form to continue

email:

[redacted]@gmail.com

name:

Andreas Røed Kjønnerud

address:

[redacted]4F

postcode:

3058

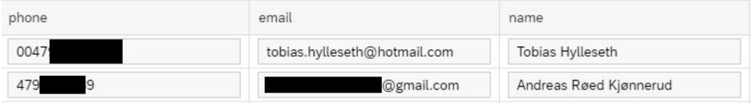
city:

Drammen

region:

Viken

The form is filled out and submit is pressed. This means the user in the database should be updated with the info that was filled in.

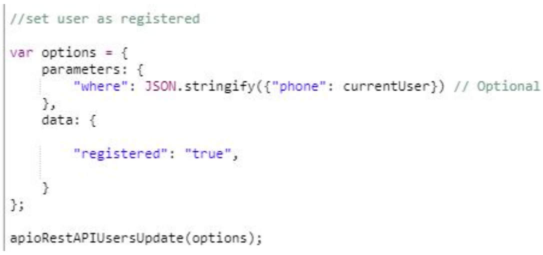


phone	email	name
0047[redacted]	tobias.hylleseth@hotmail.com	Tobias Hylleseth
479[redacted]9	[redacted]@gmail.com	Andreas Røed Kjønnerud

I update the browser and see that the data has been successfully updated.

But according to the sequence diagram for the second alpha version, the users should skip this step the next time they go into the app. So to add this functionality some coding is required.
Setting the registered boolean in the database to true in the terms page and check it in the verification page:

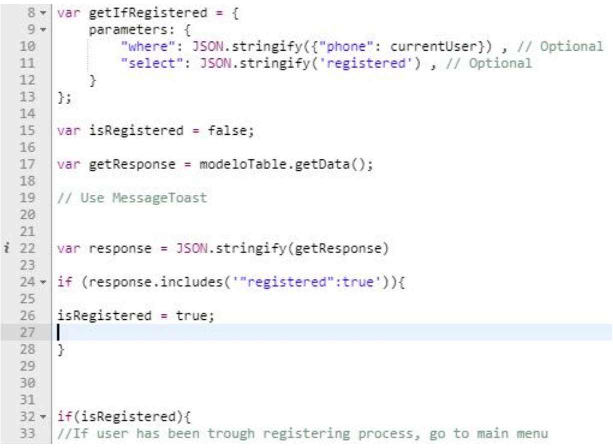
Terms page



```
//set user as registered

var options = {
    parameters: {
        "where": JSON.stringify({"phone": currentUser}) // Optional
    },
    data: {

        "registered": "true",

    }
};

apioRestAPIUsersUpdate(options);
```

‘

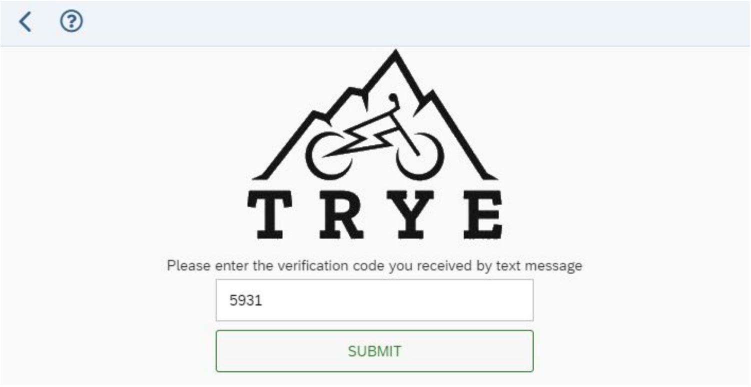
Verification page



```
var getIfRegistered = {
    parameters: {
        "where": JSON.stringify({"phone": currentUser}) , // Optional
        "select": JSON.stringify('registered') , // Optional
    }
};

var isRegistered = false;

var getResponse = modeloTable.getData();

// Use MessageToast


var response = JSON.stringify(getResponse)

if (response.includes('"registered":true')){

isRegistered = true;

}



if(isRegistered){
//If user has been trough registering process, go to main menu
```

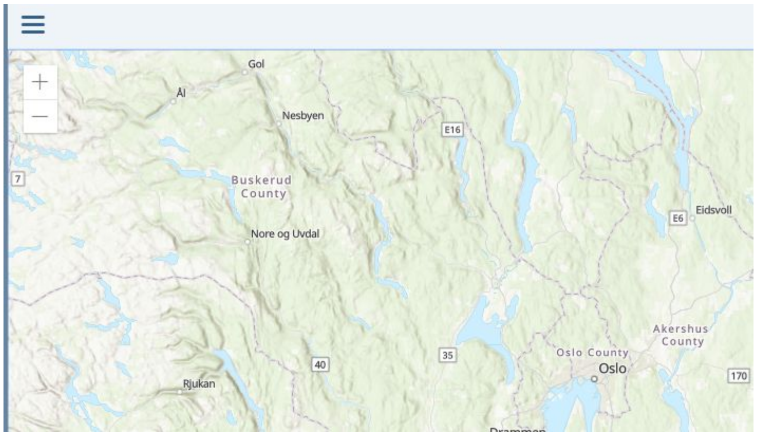
So if the user is registered it should be directed now to the main menu and skip the form and terms pages.

To do a small test I will log in again with the same phone number and see if I am skipping the form this time:



Please enter the verification code you received by text message

5931

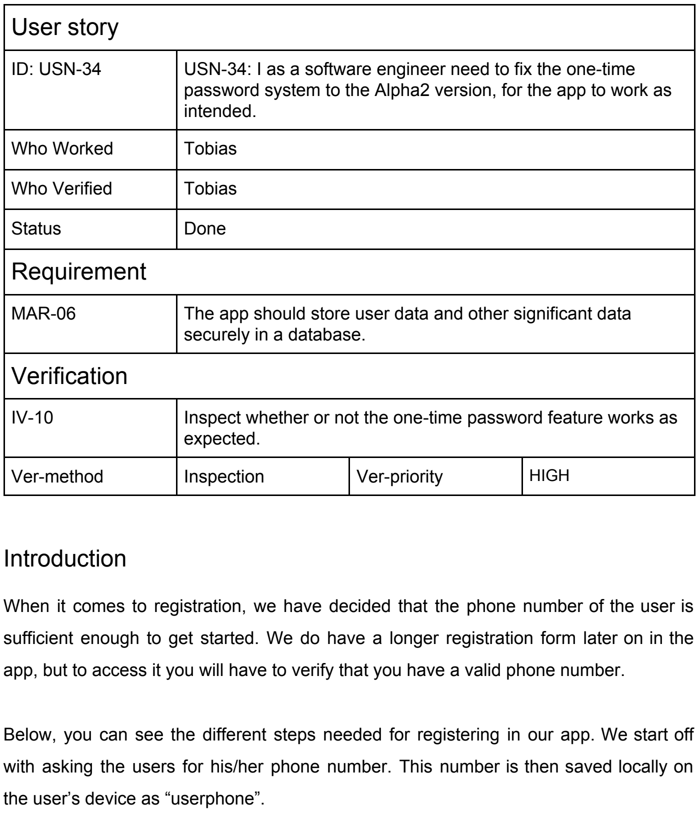


It works as intended. It went straight from the verification tab to the main menu.

Conclusion:

The updated login system is now working as it should according to the Alpha2 planning.

A.4.12 USN-34 Setting up a one-time password feature

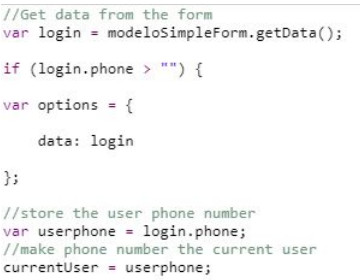


User story			
ID: USN-34	USN-34: I as a software engineer need to fix the one-time password system to the Alpha2 version, for the app to work as intended.		
Who Worked	Tobias		
Who Verified	Tobias		
Status	Done		
Requirement			
MAR-06	The app should store user data and other significant data securely in a database.		
Verification			
IV-10	Inspect whether or not the one-time password feature works as expected.		
Ver-method	Inspection	Ver-priority	HIGH

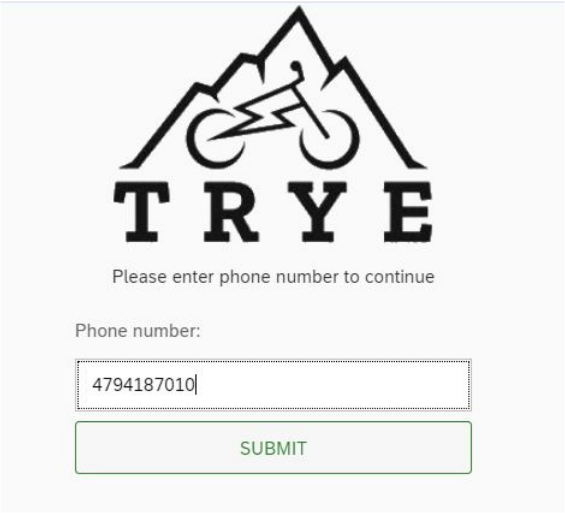
Introduction

When it comes to registration, we have decided that the phone number of the user is sufficient enough to get started. We do have a longer registration form later on in the app, but to access it you will have to verify that you have a valid phone number.

Below, you can see the different steps needed for registering in our app. We start off with asking the users for his/her phone number. This number is then saved locally on the user's device as “userphone”.



```
//Get data from the form
var login = modeloSimpleForm.getData();

if (login.phone > "") {

var options = {

    data: login

};

//store the user phone number
var userphone = login.phone;
//make phone number the current user
currentUser = userphone;
```

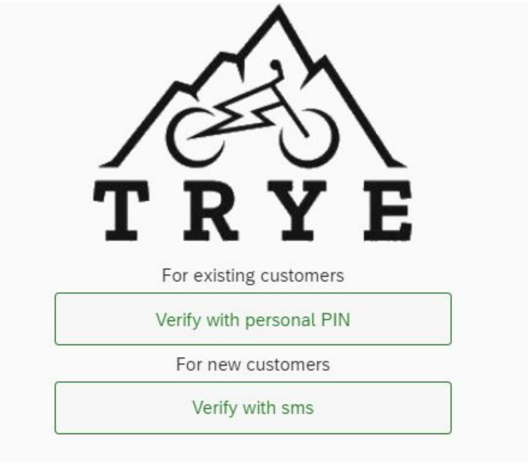


Please enter phone number to continue

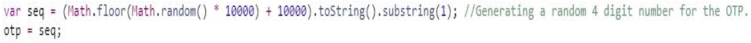
Phone number:

4794187010

Then the user is navigated to a page asking if the user is an existing customer or a new customer.



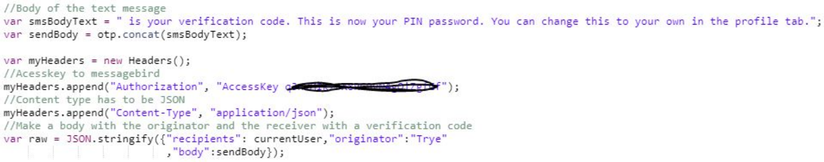
If you press “Verify with SMS”, a randomly generated four-digit is generated and also stored locally on the user’s device. This is done with this line of javaScript.



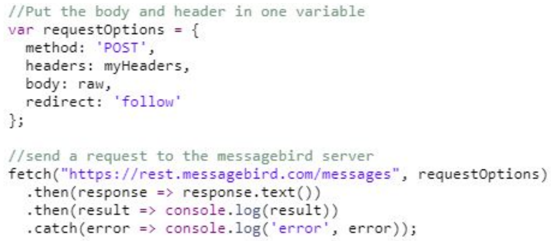
```
var seq = (Math.floor(Math.random() * 10000) + 10000).toString().substring(1); //Generating a random 4 digit number for the OTP.
otp = seq;
```

Then, we set the OTP variable to the value of seq (four random digit generator).

After that, we need to build the SMS body with the correct OTP to be received in an SMS.

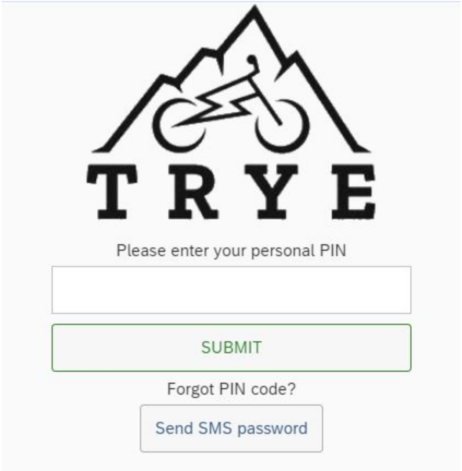


```
//Body of the text message
var smsBodyText = " is your verification code. This is now your PIN password. You can change this to your own in the profile tab.";
var sendBody = otp.concat(smsBodyText);

var myHeaders = new Headers();
//Acesskey to messagebird
myHeaders.append("Authorization", "AccessKey [redacted]");
//Content type has to be JSON
myHeaders.append("Content-Type", "application/json");
//Make a body with the originator and the receiver with a verification code
var raw = JSON.stringify({"recipients": currentUser,"originator":"Trye"
                          ,"body":sendBody});
```



```
//Put the body and header in one variable
var requestOptions = {
  method: 'POST',
  headers: myHeaders,
  body: raw,
  redirect: 'follow'
};

//send a request to the messagebird server
fetch("https://rest.messagebird.com/messages", requestOptions)
  .then(response => response.text())
  .then(result => console.log(result))
  .catch(error => console.log('error', error));
```

To send the user an SMS with the proper body we needed to do some work.
This was done by concatenating the SMS body with the OTP. To send the user an SMS containing the four-digit code, we made use of an SMS sending software that supported API's named messageBird. In the API-call to messageBird we made use of the SMS body we built earlier, as well as the userphone variable. Combined, this ensured that the correct user receives the correct one-time password. The one-time password then gets set as the user's personal PIN-code to avoid having to send one SMS per login. It also makes the login process significantly faster.



Conclusion:

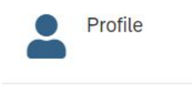
A One-Time Password is generated locally on each individual device upon registration. When going through the registration sequence, an SMS with the respective OTP is sent to the entered phone number and saved as that user’s personal PIN-code.

A.4.13 USN-39 Make users be able to view and edit their user information

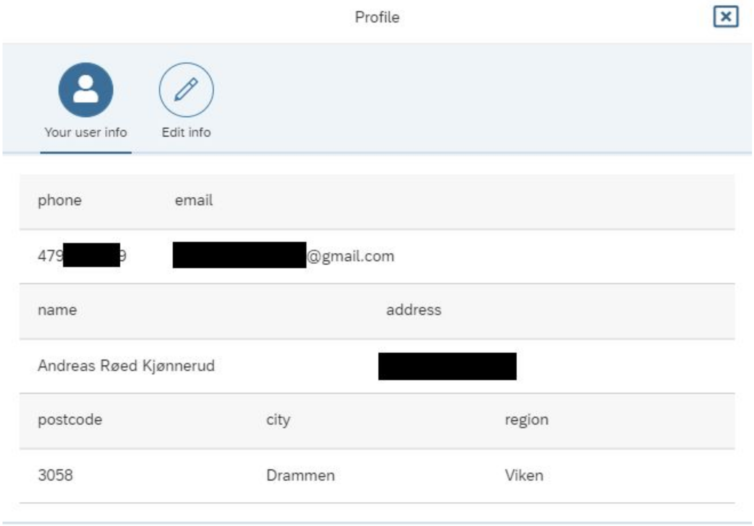
User story			
ID: USN-39	I as a software engineer need to make users able to view and edit their user information inside the app so they can update their data.		
Who Worked	Andreas		
Who Verified	Tobias		
Status	Done		
Requirement			
MAR-02	The mobile app should have a booking system.		
Verification			
IV-11	Inspect whether or not users are able to view and edit their user information inside the app.		
Ver-method	Inspection	Ver-priority	HIGH

Introduction

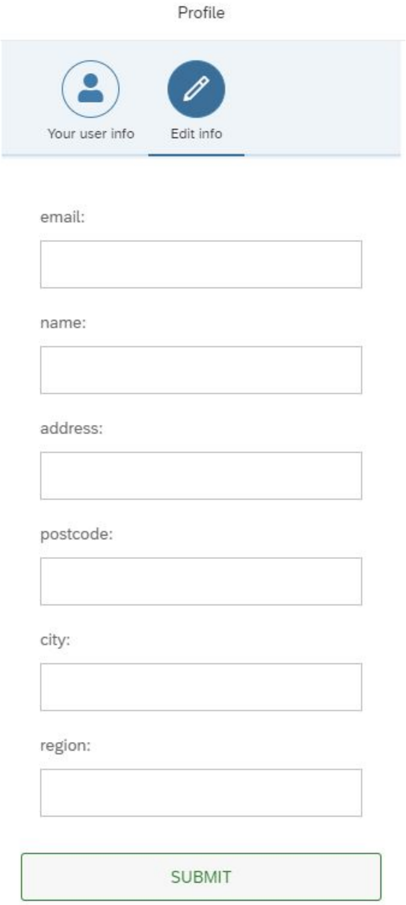
In this user story, the user tab in the sidebar menu of the app is meant to build. It has two core components, displaying the user data for the user and a form tab where the user can edit their data.



When the user clicks on the user button it will display a pop up which has two tabs. The display data tab, and the edit data tab. In the app it looks like this:



phone	email	
479[redacted]9	[redacted]@gmail.com	
name	address	
Andreas Røed Kjønnerud	[redacted]	
postcode	city	region
3058	Drammen	Viken



Profile

Your user info

Edit info

email:

name:

address:

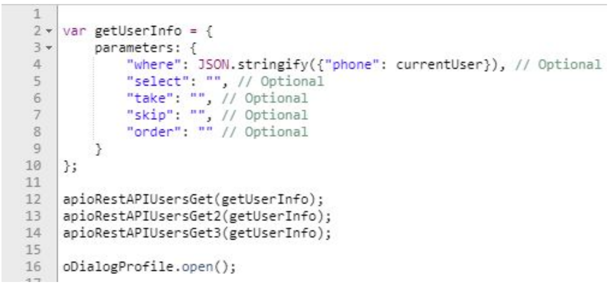
postcode:

city:

region:

Now for the backend stuff.

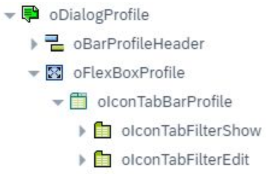
The profile button click event:



```

var getUserInfo = {
    parameters: {
        "where": JSON.stringify({"phone": currentUser}), // Optional
        "select": "", // Optional
        "take": "", // Optional
        "skip": "", // Optional
        "order": "" // Optional
    }
};

apioRestAPIUsersGet(getUserInfo);
apioRestAPIUsersGet2(getUserInfo);
apioRestAPIUsersGet3(getUserInfo);

oDialogProfile.open();
```

Here we use the GET API call to the database to retrieve the data. It will be called 3 times because the data will be split into 3 tables. This is to display the data in a more organized way.

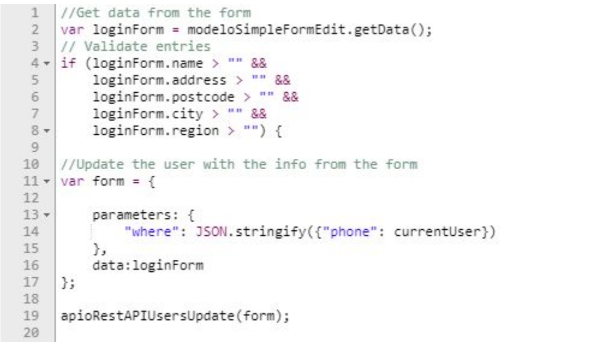
The dialogue window:



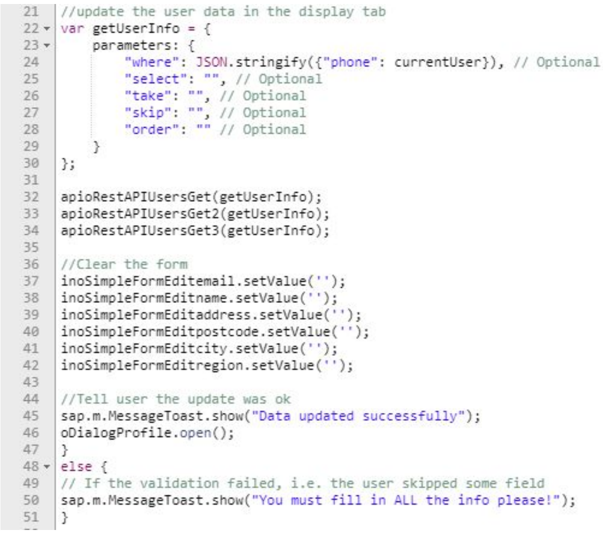
```
▾ oDialogProfile
    ▸ oBarProfileHeader
    ▾ oFlexBoxProfile
        ▾ oIconTabBarProfile
            ▸ oIconTabFilterShow
            ▸ oIconTabFilterEdit
```

An icon tab bar has been added to the dialogue to switch between showing and editing the data in a smooth way in the same dialogue.

The edit submit button with comments

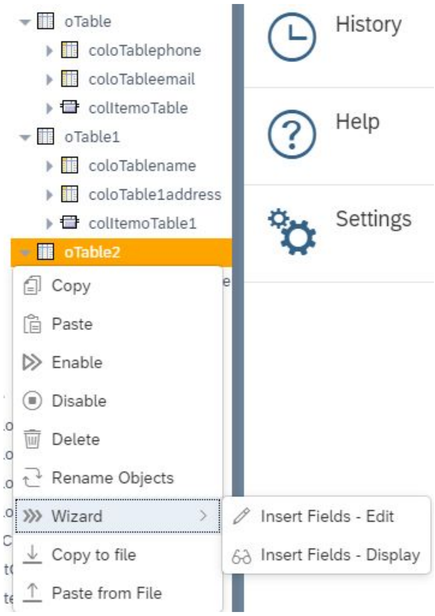


```
//Get data from the form
var loginForm = modeloSimpleFormEdit.getData();
// Validate entries
if (loginForm.name > "" &&
    loginForm.address > "" &&
    loginForm.postcode > "" &&
    loginForm.city > "" &&
    loginForm.region > "") {

//Update the user with the info from the form
var form = {

    parameters: {
        "where": JSON.stringify({"phone": currentUser})
    },
    data:loginForm
};

apioRestAPIUsersUpdate(form);

```

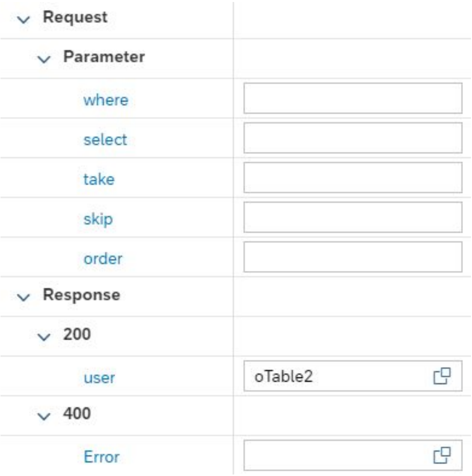


```
//update the user data in the display tab
var getUserInfo = {
    parameters: {
        "where": JSON.stringify({"phone": currentUser}), // Optional
        "select": "", // Optional
        "take": "", // Optional
        "skip": "", // Optional
        "order": "" // Optional
    }
};

apioRestAPIUsersGet(getUserInfo);
apioRestAPIUsersGet2(getUserInfo);
apioRestAPIUsersGet3(getUserInfo);

//Clear the form
inoSimpleFormEditemail.setValue('');
inoSimpleFormEditname.setValue('');
inoSimpleFormEditaddress.setValue('');
inoSimpleFormEditpostcode.setValue('');
inoSimpleFormEditcity.setValue('');
inoSimpleFormEditregion.setValue('');

//Tell user the update was ok
sap.m.MessageToast.show("Data updated successfully");
oDialogProfile.open();
}
else {
// If the validation failed, i.e. the user skipped some field
sap.m.MessageToast.show("You must fill in ALL the info please!");
}
```

To display the fields we use a Planet9 wizard which will create the table with the info we need:



And 3 APIS are added where the 200 success message with the data needs to be linked to each table in this way:

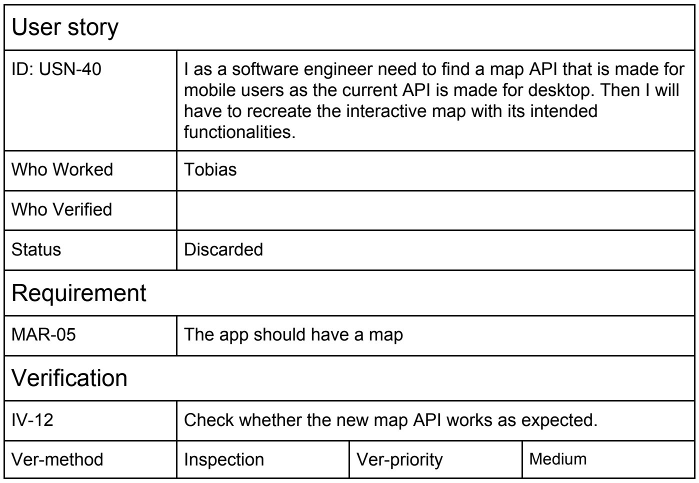


Request	
Parameter	
where	
select	
take	
skip	
order	
Response	
200	
user	oTable2
400	
Error	

Conclusion:

The Profile page is working as intended which was confirmed by both me Andreas and Tobias testing it.

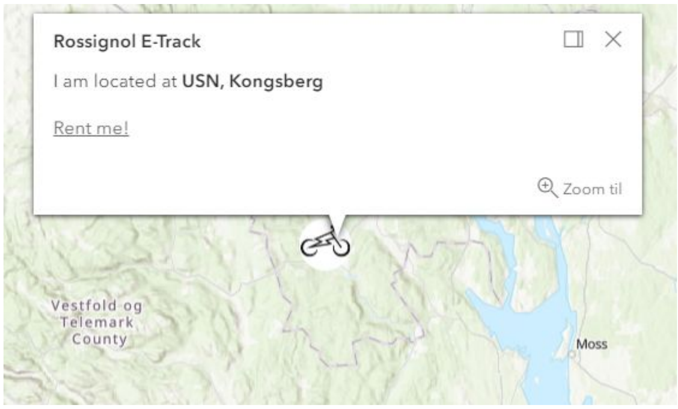
A.4.14 USN-40 Finding a map made for mobile



User story			
ID: USN-40	I as a software engineer need to find a map API that is made for mobile users as the current API is made for desktop. Then I will have to recreate the interactive map with its intended functionalities.		
Who Worked	Tobias		
Who Verified			
Status	Discarded		
Requirement			
MAR-05	The app should have a map		
Verification			
IV-12	Check whether the new map API works as expected.		
Ver-method	Inspection	Ver-priority	Medium

Introduction:

The map we started using was provided by ESRI, and was and is free to use. However, due to the API being free to use, it did not work as good on mobile phones as it did in a PC browser. The only limitation we met with ESRI's "old" and "free" solution was that if you pressed an icon, a text box would appear that was too large for a mobile display.



Rossignol E-Track

I am located at **USN, Kongsberg**

Rent me!

As you can see, the pop-up window is fairly large compared to the fixed-size Trye bike logo. On small screens, there are issues with the pop-up not showing at all/only showing the title.

Method:

To start off, I did extensive research on how to change the “popupTemplate”. A “popupTemplate” is the pop-up ESRI makes use of when clicking on icons. After a while, I noticed something weird, and that was that when clicking links I thought would lead to ESRI I would in fact be redirected to a service named ArcGIS. This halted my progress until I figured out that ArcGIS was ERSI’s “new premium brand” with full support for mobile devices, at a pricy cost.

Eventually, I had to give up on the idea of recreating the already made map and making it more “mobile” friendly.

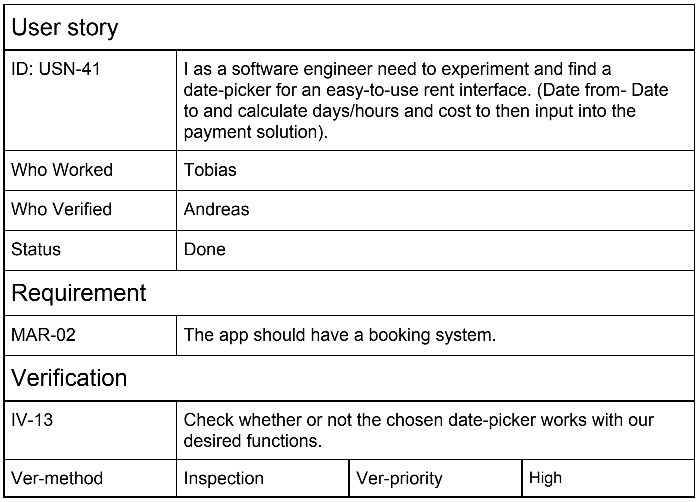
The solution we came up with instead was to simplify our already made map. As we are not aiming for the same functionalities as the scooter renting apps where you press the nearest scooter and rent it, we want to show the customer at what cabins you are able to rent a bike. The renting part of our application will be made in the respective sub-menu, and when clicking on the Trye logo the popup will only print the name of the cabin.

Conclusion:

As ESRI’s free service has limited support for mobile popups, we have decided to not make use of the full range of the popup properties. This means that we only keep the title, not the “description” part of the popup. With only printing the title, we avoid having the inconsistent UI we had earlier.

If Trye decides to put time and effort into making the interactive map even “nicer”, investing in ArcGIS’s service might be a good investment. As of now, we see no substantial improvement with switching to the pay-per-use service that supports mobile popups by default.

A.4.15 USN-41 Finding a suitable date picker for an easy-to-use rent interface

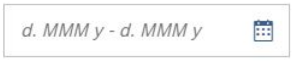


User story			
ID: USN-41	I as a software engineer need to experiment and find a date-picker for an easy-to-use rent interface. (Date from- Date to and calculate days/hours and cost to then input into the payment solution).		
Who Worked	Tobias		
Who Verified	Andreas		
Status	Done		
Requirement			
MAR-02	The app should have a booking system.		
Verification			
IV-13	Check whether or not the chosen date-picker works with our desired functions.		
Ver-method	Inspection	Ver-priority	High

Introduction:

Our employer's focus (Trye) when it came to how the renting of the bicycles would be done was to focus on entire days of renting. In addition, the guys at Trye wanted a one day break between each lease, to ensure that the bikes are well looked after (quality checks will be done in the start) and that the battery is fully charged. The entire experience Trye is providing will dramatically fall in value if the battery is just 20% charged when a customer unlocks the bike.

As of many of the functionalities we've already implemented in the application, I found an already created OpenUI5 object named DatePicker. This object fulfilled half of my requirements for a suitable date-picker, namely the Date from-. After some research, I found an object derived from the original DatePicker named DateRangeSelection. DateRangeSelection fulfilled all my requirements for a date-picker, and I was ready to develop.



d. MMM y - d. MMM y

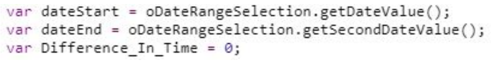
Method Used:

The functionalities needed to be done were:

1) Calculate how many days the user-selected period is.
2) Calculate the price based on Trye's price table and the already calculated user-selected period.

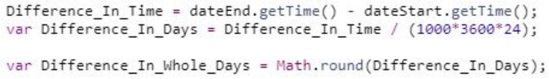
To achieve the first function, I needed to save the start-date and the end-date in two variables. Retrieving the value from the date-picker was harder than originally thought, but luckily Andreas helped me out and got it to work in a quick fifteen minutes.

I also defined a “Difference In Time”.



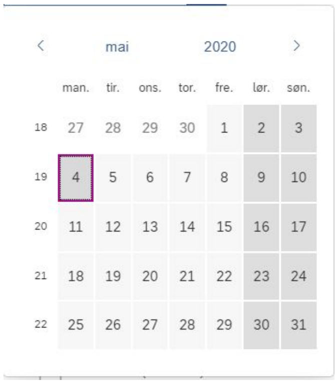
```
var dateStart = oDateRangeSelection.getDateValue();
var dateEnd = oDateRangeSelection.getSecondDateValue();
var Difference_In_Time = 0;
```

I was unsure of how the “DateValue” was formatted, so to calculate the time difference, I used a built-in javaScript function called .getTime(); This would format the time difference how I wanted it, making it easier to calculate the number of days in a rent-period.



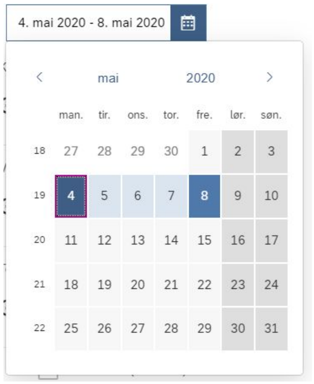
```
Difference_In_Time = dateEnd.getTime() - dateStart.getTime();
var Difference_In_Days = Difference_In_Time / (1000*3600*24);

var Difference_In_Whole_Days = Math.round(Difference_In_Days);
```

The logic is simple, the “end” date minus the “start” date gave me the millisecond difference in time. By first dividing it with 1000 (1 second), then 3600s (1 hour), then 24h (1 day), we got the number of days in the selected period. However, I encountered a tiny bug when calculating the difference in time, and I always ended up with 0,999997 days or so. This made it hard to calculate the price for the selected period, so I used a Math.round function to round the number to the nearest integer (whole number).



< mai 2020 >

	man.	tir.	ons.	tor.	fre.	lør.	søn.
18	27	28	29	30	1	2	3
19	4	5	6	7	8	9	10
20	11	12	13	14	15	16	17
21	18	19	20	21	22	23	24
22	25	26	27	28	29	30	31

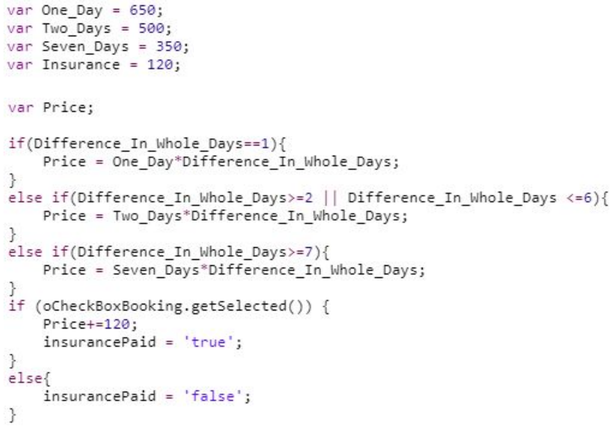


4. mai 2020 - 8. mai 2020

< mai 2020 >

	man.	tir.	ons.	tor.	fre.	lør.	søn.
18	27	28	29	30	1	2	3
19	4	5	6	7	8	9	10
20	11	12	13	14	15	16	17
21	18	19	20	21	22	23	24
22	25	26	27	28	29	30	31

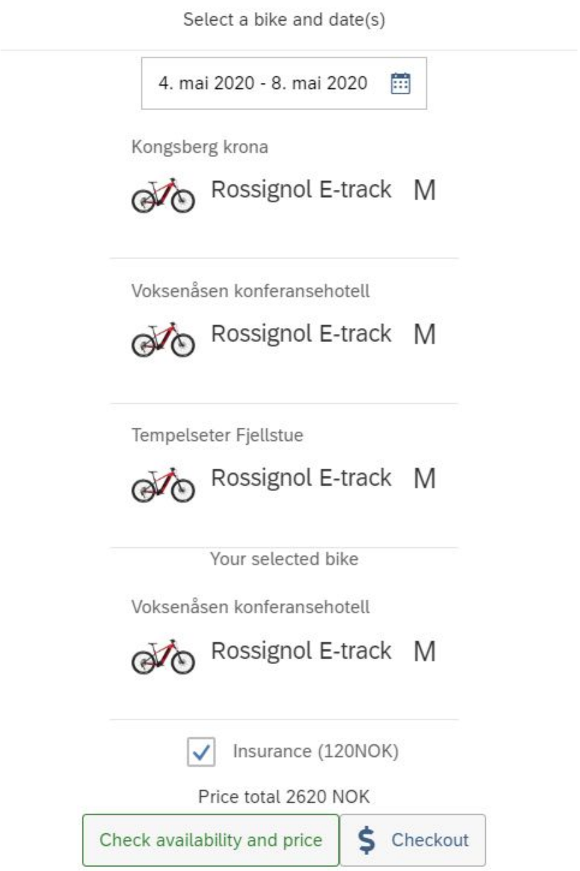
Then, for the second function (price-calculation), we just need to use the "Difference_In_Days" variable and multiply it with the price per day. Trye's price table was: One day = 650 NOK/Day, Two Days = 500NOK/Day, Seven Days=350NOK/Day and Insurance Per Period = 120NOK. To be able to get the correct price for the period, I made a simple if, else if-logic that checked whether or not the rent period was: 1 Day, 2-6 Days, 7 Or More Days and multiplied the price per day with the Difference In Days.



```
var One_Day = 650;
var Two_Days = 500;
var Seven_Days = 350;
var Insurance = 120;

var Price;

if(Difference_In_Whole_Days==1){
    Price = One_Day*Difference_In_Whole_Days;
}
else if(Difference_In_Whole_Days>=2 || Difference_In_Whole_Days <=6){
    Price = Two_Days*Difference_In_Whole_Days;
}
else if(Difference_In_Whole_Days>=7){
    Price = Seven_Days*Difference_In_Whole_Days;
}
if (oCheckBoxBooking.getSelected()) {
    Price+=120;
    insurancePaid = 'true';
}
else{
    insurancePaid = 'false';
}
```

Conclusion:

The date-picker and price-calculation functions are now done and work as intended. A customer/user can select the days he/she want to rent a specific bicycle and can check availability and price. Checking availability is done by Andreas in User Story 49 (USN-49). Below you can see the full booking interface and the calculated price from 4th of May to the 8th of May with insurance.



Select a bike and date(s)

4. mai 2020 - 8. mai 2020

Kongsberg krona

Rossignol E-track M

Voksenåsen konferansehotell

Rossignol E-track M

Tempelseter Fjellstue

Rossignol E-track M

Your selected bike

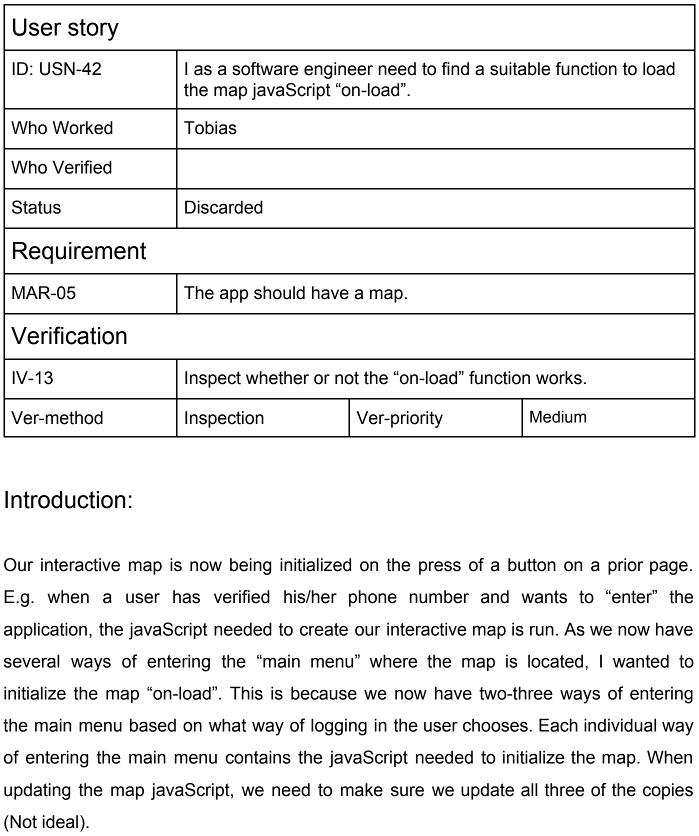
Voksenåsen konferansehotell

Rossignol E-track M

☑ Insurance (120NOK)

Price total 2620 NOK

A.4.16 USN-42 Initializing map when loading the main menu



User story			
ID: USN-42	I as a software engineer need to find a suitable function to load the map javaScript “on-load”.		
Who Worked	Tobias		
Who Verified			
Status	Discarded		
Requirement			
MAR-05	The app should have a map.		
Verification			
IV-13	Inspect whether or not the “on-load” function works.		
Ver-method	Inspection	Ver-priority	Medium

Introduction:

Our interactive map is now being initialized on the press of a button on a prior page. E.g. when a user has verified his/her phone number and wants to “enter” the application, the javaScript needed to create our interactive map is run. As we now have several ways of entering the “main menu” where the map is located, I wanted to initialize the map “on-load”. This is because we now have two-three ways of entering the main menu based on what way of logging in the user chooses. Each individual way of entering the main menu contains the javaScript needed to initialize the map. When updating the map javaScript, we need to make sure we update all three of the copies (Not ideal).

Method Used:

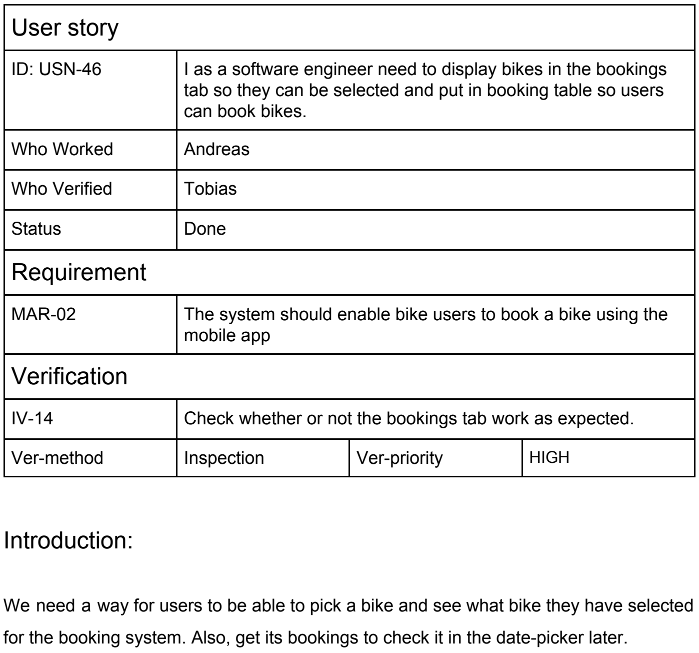
As simple as it sounds, this problem sits deeper than it would seem. OpenUI5 supports javaScript, but as the map javaScript is defined within a function (The javaScript within OpenUI5 doesn't need a declared function.) it becomes harder to execute the function "on-load".

A possible fix for not needing to declare a function, making it easier to make the javaScript run "on-load" would be to create a map API that returns the javaScript needed.

Conclusion:

After a week or two of on- and off- research, I decided that it would not benefit the progress of our project to redo the entire structure of our app for cleaner code. If Trye continues the development after our project is done, a restructuring of how the map works might be good. Combined with USN-40 (Finding a map made for mobile), this would be a positive addition to the app.

A.4.17 USN-46 Displaying bikes in the bookings tab so they can be selected

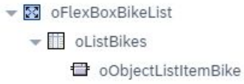


User story			
ID: USN-46	I as a software engineer need to display bikes in the bookings tab so they can be selected and put in booking table so users can book bikes.		
Who Worked	Andreas		
Who Verified	Tobias		
Status	Done		
Requirement			
MAR-02	The system should enable bike users to book a bike using the mobile app		
Verification			
IV-14	Check whether or not the bookings tab work as expected.		
Ver-method	Inspection	Ver-priority	HIGH

Introduction:

We need a way for users to be able to pick a bike and see what bike they have selected for the booking system. Also, get its bookings to check it in the date-picker later.

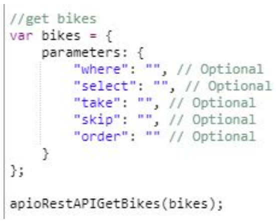
To do this an object list is created which retrieves all bikes from our database automatically when the user clicks on the bookings tab.



```
oFlexBoxBikeList
    oListBikes
        oObjectListItemBike
```

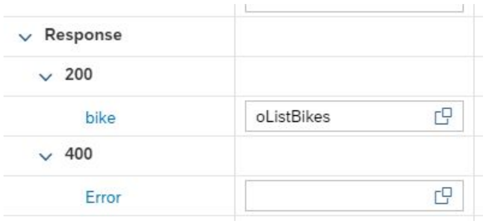
So here we have a list which should autofill by getting the data from the database and filling up the list.

Here is how the list is retrieved in the start when users enter their pin and click submit:



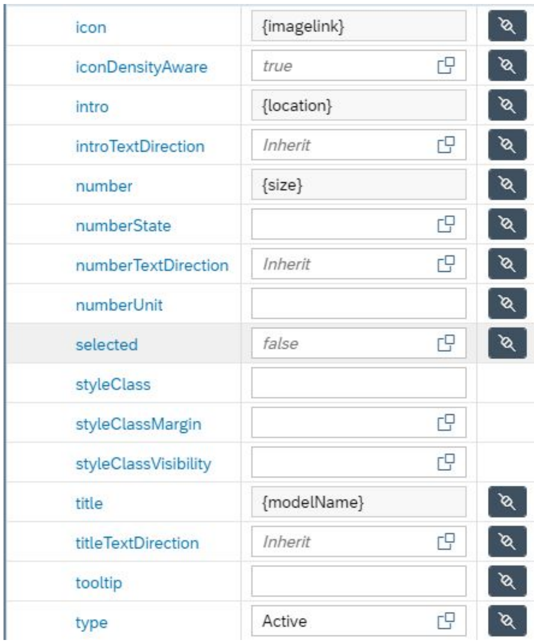
```
//get bikes
var bikes = {
    parameters: {
        "where": "", // Optional
        "select": "", // Optional
        "take": "", // Optional
        "skip": "", // Optional
        "order": "" // Optional
    }
};

apioRestAPIGetBikes(bikes);
```

This data is now sent to the list with the 200 success response:



Response	
200	
bike	oListBikes
400	
Error	

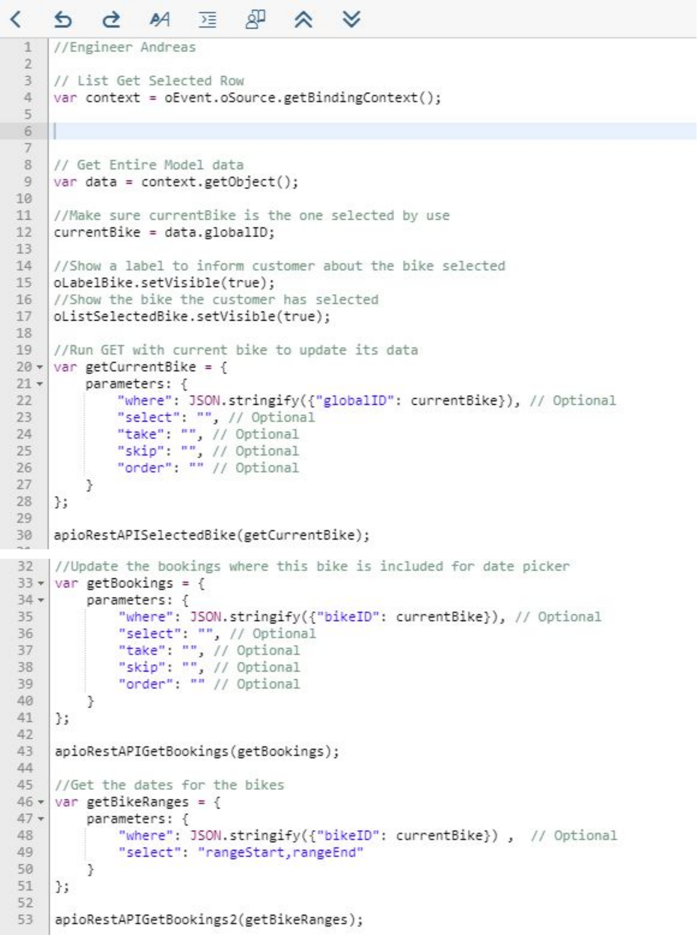
Now that we have the data we can display it by giving its data to the selected field in the object:



icon	{imagelink}
iconDensityAware	true
intro	{location}
introTextDirection	Inherit
number	{size}
numberState	
numberTextDirection	Inherit
numberUnit	
selected	false
styleClass	
styleClassMargin	
styleClassVisibility	
title	{modelName}
titleTextDirection	Inherit
tooltip	
type	Active

We also set the object to type active so it can be clicked.

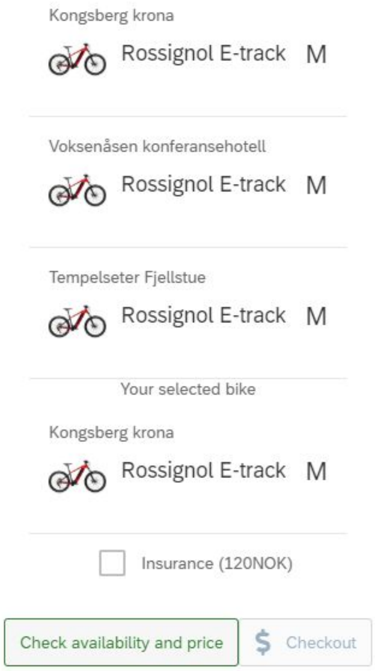
Now for the code when the object is clicked. All the code is commented with its function:



```
//Engineer Andreas

// List Get Selected Row
var context = oEvent.oSource.getBindingContext();



// Get Entire Model data
var data = context.getObject();

//Make sure currentBike is the one selected by use
currentBike = data.globalID;

//Show a label to inform customer about the bike selected
oLabelBike.setVisible(true);
//Show the bike the customer has selected
oListSelectedBike.setVisible(true);

//Run GET with current bike to update its data
var getCurrentBike = {
    parameters: {
        "where": JSON.stringify({"globalID": currentBike}), // Optional
        "select": "", // Optional
        "take": "", // Optional
        "skip": "", // Optional
        "order": "" // Optional
    }
};

apioRestAPISelectedBike(getCurrentBike);
```

```
//Update the bookings where this bike is included for date picker
var getBookings = {
    parameters: {
        "where": JSON.stringify({"bikeID": currentBike}), // Optional
        "select": "", // Optional
        "take": "", // Optional
        "skip": "", // Optional
        "order": "" // Optional
    }
};

apioRestAPIGetBookings(getBookings);

//Get the dates for the bikes
var getBikeRanges = {
    parameters: {
        "where": JSON.stringify({"bikeID": currentBike}) ,  // Optional
        "select": "rangeStart,rangeEnd"
    }
};

apioRestAPIGetBookings2(getBikeRanges);
```

So here is the final result in the user interface:



Kongsberg krona

Rossignol E-track M

Voksenåsen konferansehotell

Rossignol E-track M

Tempelseter Fjellstue

Rossignol E-track M

Your selected bike

Kongsberg krona

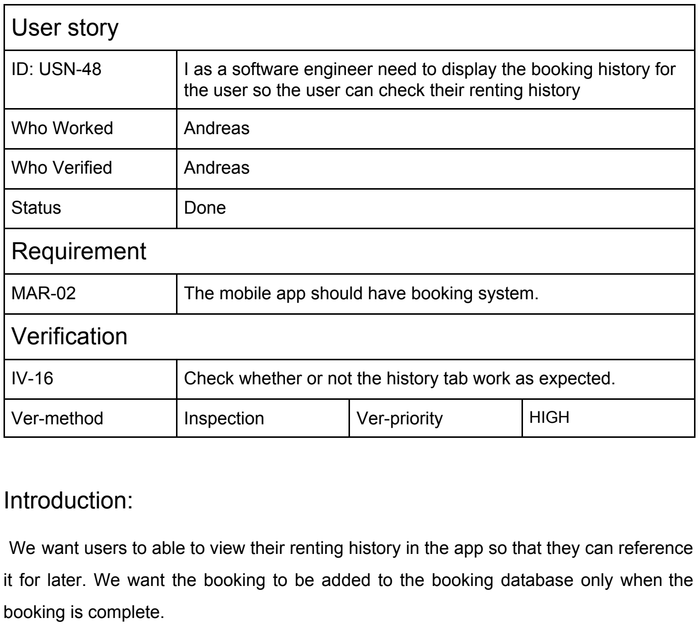
Rossignol E-track M

□ Insurance (120NOK)

Conclusion:

The list is working as intended and we have debugged and tested that it is successful.

A.4.18 USN-48 Displaying the booking history for the user



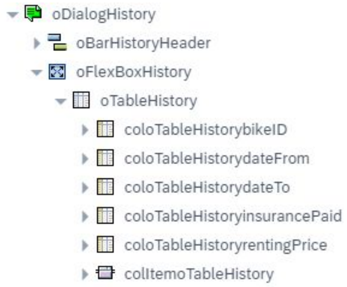
User story			
ID: USN-48	I as a software engineer need to display the booking history for the user so the user can check their renting history		
Who Worked	Andreas		
Who Verified	Andreas		
Status	Done		
Requirement			
MAR-02	The mobile app should have booking system.		
Verification			
IV-16	Check whether or not the history tab work as expected.		
Ver-method	Inspection	Ver-priority	HIGH

Introduction:

We want users to able to view their renting history in the app so that they can reference it for later. We want the booking to be added to the booking database only when the booking is complete.

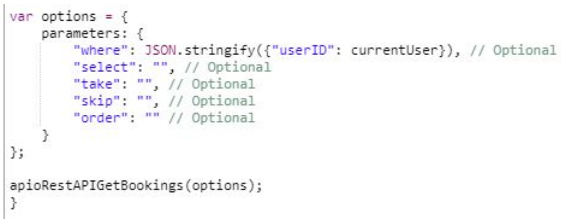
The history tab

We want to show the renting history by showing a table with columns. This was done by making a history page with a table:



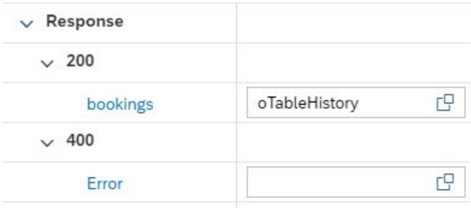
```
oDialogHistory
    oBarHistoryHeader
    oFlexBoxHistory
        oTableHistory
            coloTableHistorybikeID
            coloTableHistorydateFrom
            coloTableHistorydateTo
            coloTableHistoryinsurancePaid
            coloTableHistoryrentingPrice
            colItemoTableHistory
```

Now we need to call the Bookings api with data from the current user. This is done when pressing the submit button in the pin page.



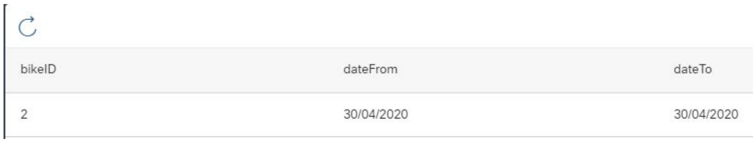
```
var options = {
    parameters: {
        "where": JSON.stringify({"userID": currentUser}), // Optional
        "select": "", // Optional
        "take": "", // Optional
        "skip": "", // Optional
        "order": "" // Optional
    }
};

apioRestAPIGetBookings(options);
}
```

Then give the success response to historytable:



Response	
200	
bookings	oTableHistory
400	
Error	

This is how the table looks with a refresh button:



bikeID	dateFrom	dateTo
2	30/04/2020	30/04/2020

Now when the refresh button is pressed the table is refreshed

Conclusion:

They history tab is working as intended and we have run several tests to confirm this.

A.4.19 USN-49 Calculate the availability for users

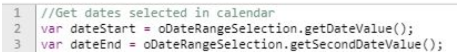
User story			
ID: USN-49	I as a software engineer need to fix the calendar in the booking system so it can calculate the availability for users and stop users from booking a date back in time		
Who Worked	Andreas		
Who Verified	Tobias		
Status	Done		
Requirement			
MAR-02	The mobile app should have booking system.		
Verification			
IV-17	Check whether or not the bookings tab work as expected.		
Ver-method	Inspection	Ver-priority	HIGH

Introduction:

When users book bikes we need to make sure users have selected a bike, valid dates and check if the dates selected available for the selected bike.

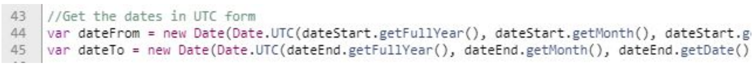
The availability button:

When users book bikes they can select dates and a bike for renting. But before they can checkout and pay we need to make sure bookings don’t overlap. First thing is getting the dates from the calendar

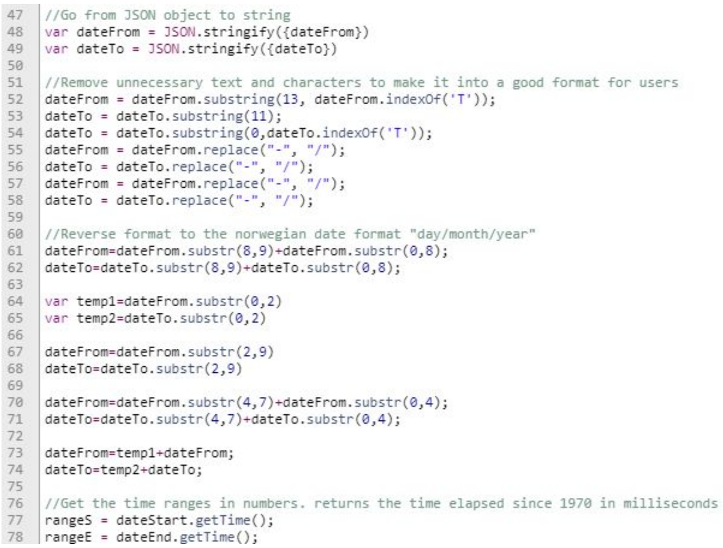


```
//Get dates selected in calendar
var dateStart = oDateRangeSelection.getDateValue();
var dateEnd = oDateRangeSelection.getSecondDateValue();
```

Now we need to make 4 parameters. The dates in a format for humans to read and ranges in milliseconds for calculations, the process is described here with comments:

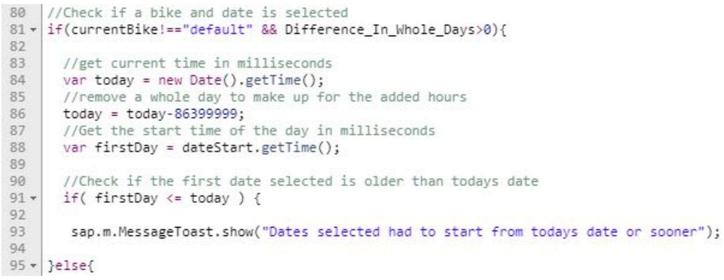


```
//Get the dates in UTC form
var dateFrom = new Date(Date.UTC(dateStart.getFullYear(), dateStart.getMonth(), dateStart.g
var dateTo = new Date(Date.UTC(dateEnd.getFullYear(), dateEnd.getMonth(), dateEnd.getDate()
```



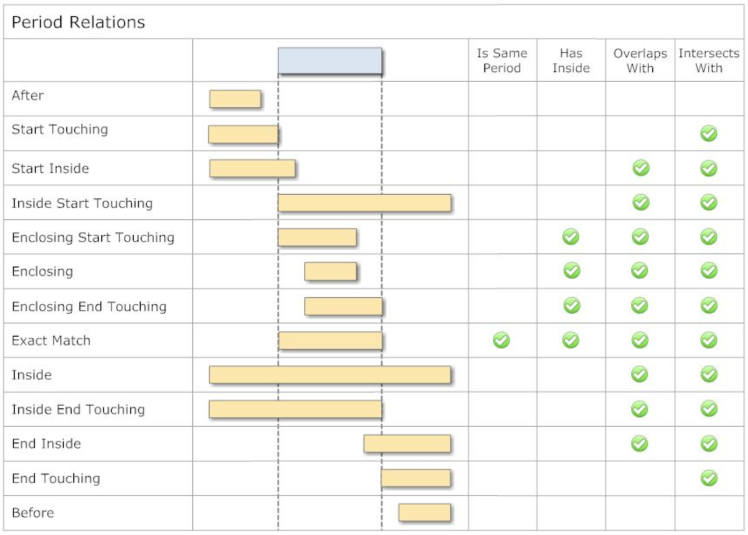
```
//Go from JSON object to string
var dateFrom = JSON.stringify({dateFrom})
var dateTo = JSON.stringify({dateTo})

//Remove unnecessary text and characters to make it into a good format for users
dateFrom = dateFrom.substring(13, dateFrom.indexOf('T'));
dateTo = dateTo.substring(11);
dateTo = dateTo.substring(0,dateTo.indexOf('T'));
dateFrom = dateFrom.replace("-", "/");
dateTo = dateTo.replace("-", "/");
dateFrom = dateFrom.replace("-", "/");
dateTo = dateTo.replace("-", "/");

//Reverse format to the norwegian date format "day/month/year"
dateFrom=dateFrom.substr(8,9)+dateFrom.substr(0,8);
dateTo=dateTo.substr(8,9)+dateTo.substr(0,8);

var temp1=dateFrom.substr(0,2)
var temp2=dateTo.substr(0,2)

dateFrom=dateFrom.substr(2,9)
dateTo=dateTo.substr(2,9)

dateFrom=dateFrom.substr(4,7)+dateFrom.substr(0,4);
dateTo=dateTo.substr(4,7)+dateTo.substr(0,4);

dateFrom=temp1+dateFrom;
dateTo=temp2+dateTo;

//Get the time ranges in numbers. returns the time elapsed since 1970 in milliseconds
rangeS = dateStart.getTime();
rangeE = dateEnd.getTime();
```

Then we need some if sentences to check first if a bike and dates are selected. Also if a date selected is not older than today's date.



```
//Check if a bike and date is selected
if(currentBike!=="default" && Difference_In_Whole_Days>0){

  //get current time in milliseconds
  var today = new Date().getTime();
  //remove a whole day to make up for the added hours
  today = today-86399999;
  //Get the start time of the day in milliseconds
  var firstDay = dateStart.getTime();

  //Check if the first date selected is older than todays date
  if( firstDay <= today ) {

   sap.m.MessageToast.show("Dates selected had to start from todays date or sooner");

}else{
```

If the initial checks are okay then we come to the hard part. Get bookings from a specific bike and check all possible cases of overlap. It will be a long if sentence and so this this diagram here shows all possible cases:

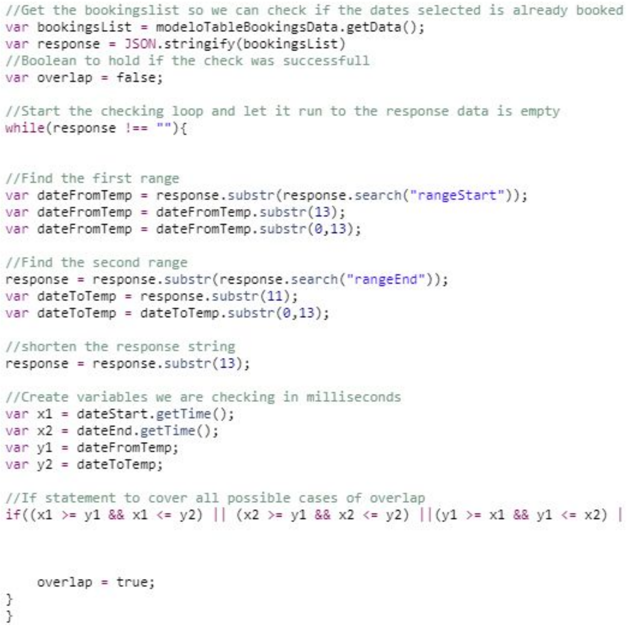


Period Relations

	Is Same Period	Has Inside	Overlaps With	Intersects With
After				
Start Touching				✓
Start Inside			✓	✓
Inside Start Touching			✓	✓
Enclosing Start Touching		✓	✓	✓
Enclosing		✓	✓	✓
Enclosing End Touching		✓	✓	✓
Exact Match	✓	✓	✓	✓
Inside			✓	✓
Inside End Touching			✓	✓
End Inside			✓	✓
End Touching				✓
Before				

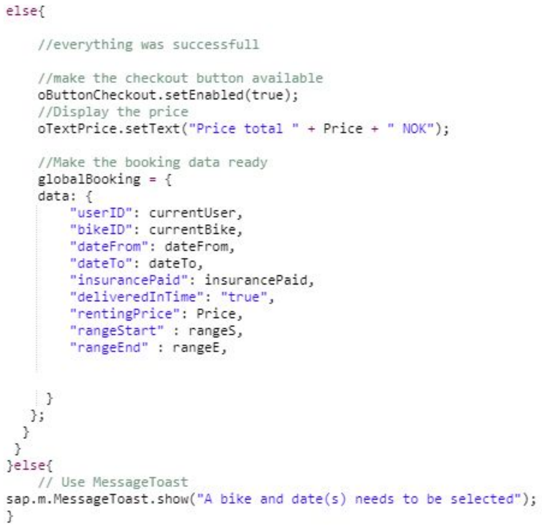
The if sentence to cover all this will be formatted like this for each entry in the bookings table:
if((x1 >= y1 && x1 <= y2) || (x2 >= y1 && x2 <= y2) ||(y1 >= x1 && y1 <= x2) || (y2 >= x1 && y2 <= x2))

Now for the loop, it is a string loop which gets dates from the string, checks them and the shortens the string until it is empty:



```
//Get the bookingslist so we can check if the dates selected is already booked
var bookingsList = modeloTableBookingsData.getData();
var response = JSON.stringify(bookingsList)
//Boolean to hold if the check was successfull
var overlap = false;

//Start the checking loop and let it run to the response data is empty
while(response !== ""){


//Find the first range
var dateFromTemp = response.substr(response.search("rangeStart"));
var dateFromTemp = dateFromTemp.substr(13);
var dateFromTemp = dateFromTemp.substr(0,13);

//Find the second range
response = response.substr(response.search("rangeEnd"));
var dateToTemp = response.substr(11);
var dateToTemp = dateToTemp.substr(0,13);

//shorten the response string
response = response.substr(13);

//Create variables we are checking in milliseconds
var x1 = dateStart.getTime();
var x2 = dateEnd.getTime();
var y1 = dateFromTemp;
var y2 = dateToTemp;

//If statement to cover all possible cases of overlap
if((x1 >= y1 && x1 <= y2) || (x2 >= y1 && x2 <= y2) ||(y1 >= x1 && y1 <= x2) |



    overlap = true;
}
}
```

If all the checks are successful we then put the data in a variable ready for putting it in the database:



```
else{

    //everything was successfull

    //make the checkout button available
    oButtonCheckout.setEnabled(true);
    //Display the price
    oTextPrice.setText("Price total " + Price + " NOK");

    //Make the booking data ready
    globalBooking = {
    data: {
        "userID": currentUser,
        "bikeID": currentBike,
        "dateFrom": dateFrom,
        "dateTo": dateTo,
        "insurancePaid": insurancePaid,
        "deliveredInTime": "true",
        "rentingPrice": Price,
        "rangeStart" : rangeS,
        "rangeEnd" : rangeE,


     }
   };
  }
 }
}else{
    // Use MessageToast
sap.m.MessageToast.show("A bike and date(s) needs to be selected");
}
```

Now the customer can click the checkout button and that the payment process.

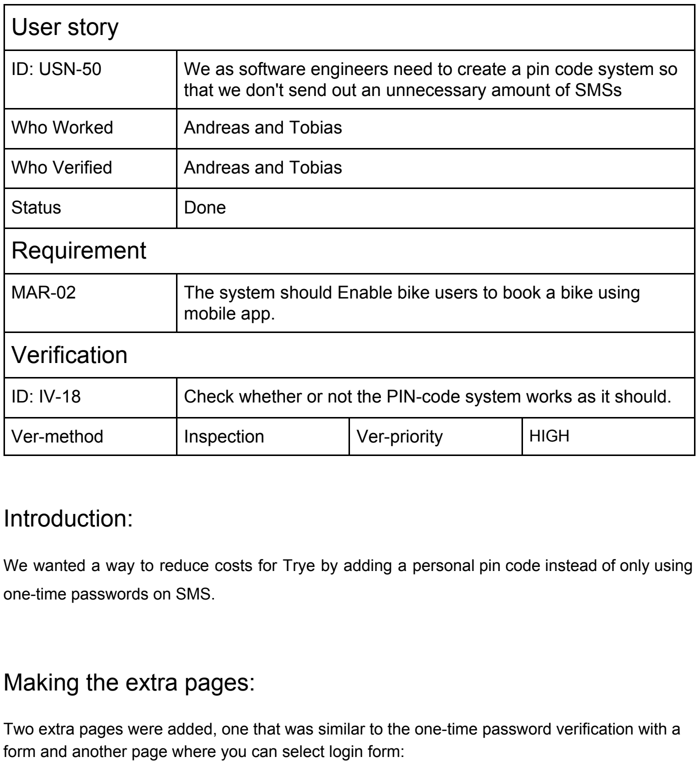
Conclusion:

The booking system works as intended which is proven by several tests by Andreas and Tobias.

Period Relations Source:

https://www.codeproject.com//KB/datetime/TimePeriod/PeriodRelations.png

A.4.20 USN-50 Fixing a PIN-code system



User story			
ID: USN-50	We as software engineers need to create a pin code system so that we don't send out an unnecessary amount of SMSs		
Who Worked	Andreas and Tobias		
Who Verified	Andreas and Tobias		
Status	Done		
Requirement			
MAR-02	The system should Enable bike users to book a bike using mobile app.		
Verification			
ID: IV-18	Check whether or not the PIN-code system works as it should.		
Ver-method	Inspection	Ver-priority	HIGH

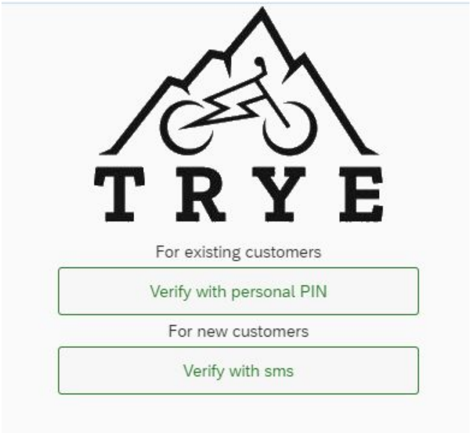
Introduction:

We wanted a way to reduce costs for Trye by adding a personal pin code instead of only using one-time passwords on SMS.

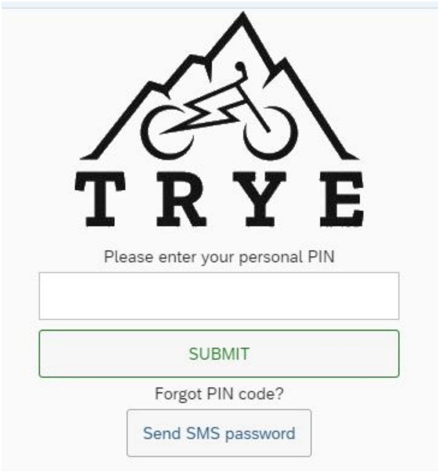
Making the extra pages:

Two extra pages were added, one that was similar to the one-time password verification with a form and another page where you can select login form:

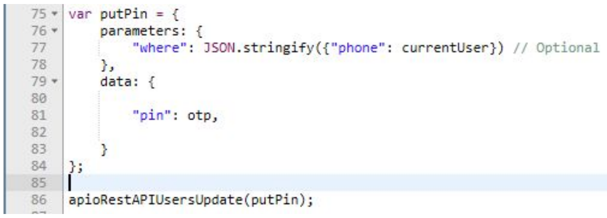
Password select page:



The pin password page with password recovery option:

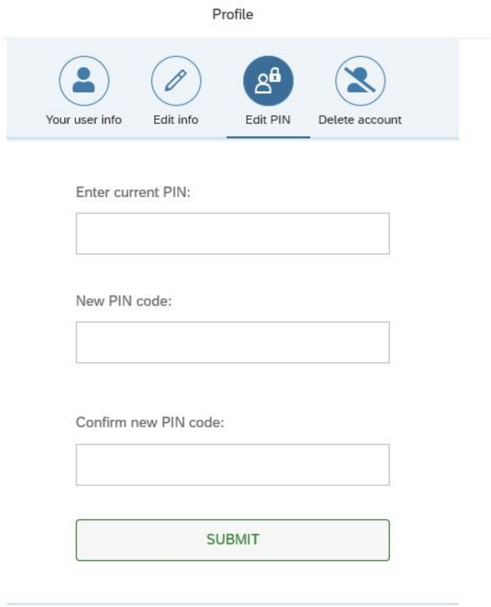


When customers registers they will be assigned the same code as the one time password:

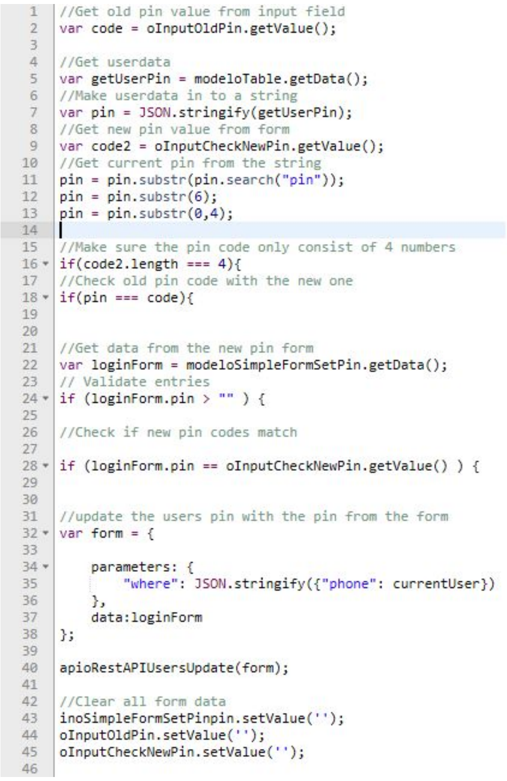


```
var putPin = {
    parameters: {
        "where": JSON.stringify({"phone": currentUser}) // Optional
    },
    data: {

        "pin": otp,

    }
};

apioRestAPIUsersUpdate(putPin);
```

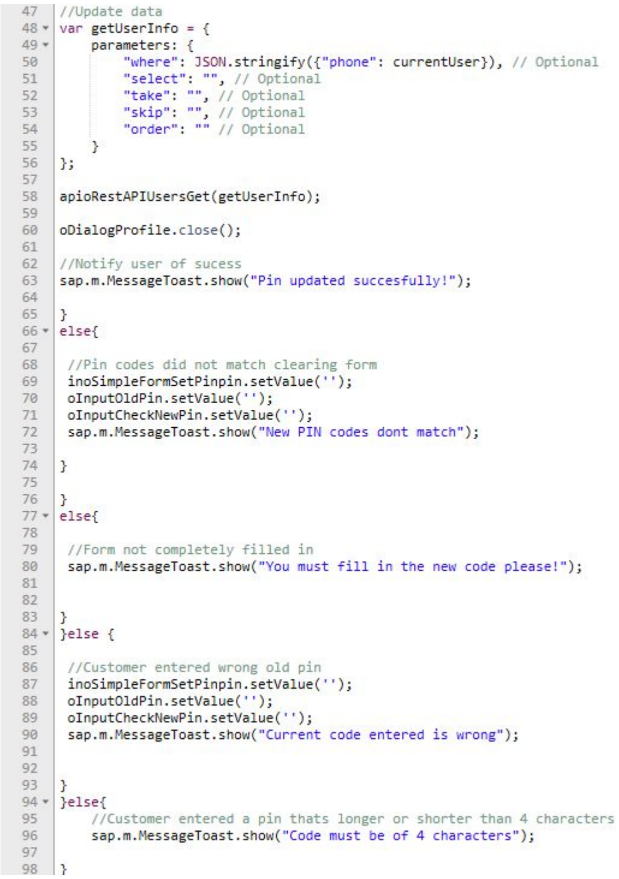
After they have been through the registration process they can change the personal PIN to any 4 digit code they like. The pin changing page looks like this:



Now for customers to change their pin we need to check if they know the old pin code and also type in the new one 2 times for verification. This is done through first retrieving the old pin from the database and then a series of if statements. Code described with comments:

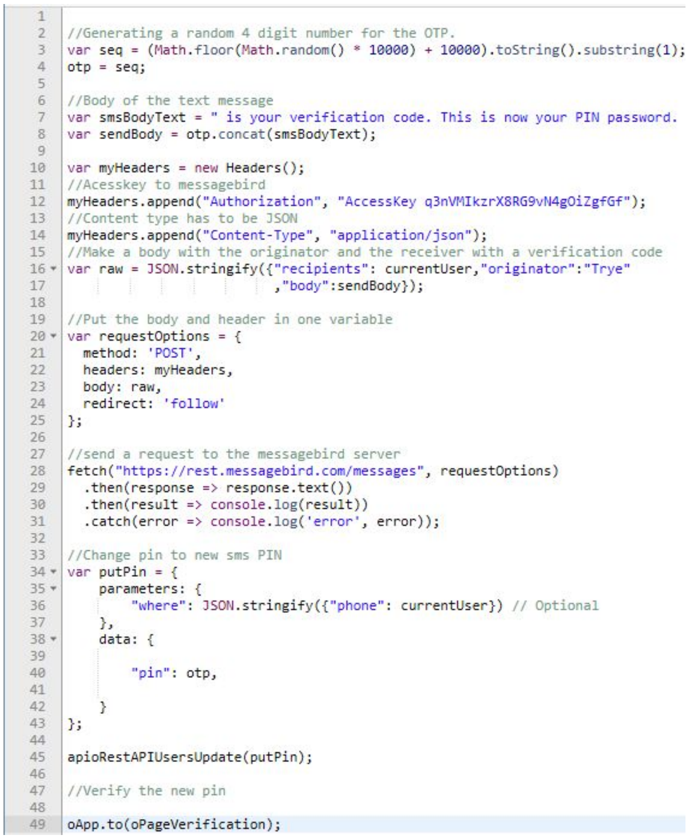


```
//Get old pin value from input field
var code = oInputOldPin.getValue();

//Get userdata
var getUserPin = modeloTable.getData();
//Make userdata in to a string
var pin = JSON.stringify(getUserPin);
//Get new pin value from form
var code2 = oInputCheckNewPin.getValue();
//Get current pin from the string
pin = pin.substr(pin.search("pin"));
pin = pin.substr(6);
pin = pin.substr(0,4);

//Make sure the pin code only consist of 4 numbers
if(code2.length === 4){
//Check old pin code with the new one
if(pin === code){


//Get data from the new pin form
var loginForm = modeloSimpleFormSetPin.getData();
// Validate entries
if (loginForm.pin > "" ) {

//Check if new pin codes match

if (loginForm.pin == oInputCheckNewPin.getValue() ) {


//update the users pin with the pin from the form
var form = {

    parameters: {
        "where": JSON.stringify({"phone": currentUser})
    },
    data:loginForm
};

apioRestAPIUsersUpdate(form);

//Clear all form data
inoSimpleFormSetPinpin.setValue('');
oInputOldPin.setValue('');
oInputCheckNewPin.setValue('');

```



```
//Update data
var getUserInfo = {
    parameters: {
        "where": JSON.stringify({"phone": currentUser}), // Optional
        "select": "", // Optional
        "take": "", // Optional
        "skip": "", // Optional
        "order": "" // Optional
    }
};

apioRestAPIUsersGet(getUserInfo);

oDialogProfile.close();

//Notify user of sucess
sap.m.MessageToast.show("Pin updated succesfully!");

}
else{

 //Pin codes did not match clearing form
 inoSimpleFormSetPinpin.setValue('');
 oInputOldPin.setValue('');
 oInputCheckNewPin.setValue('');
 sap.m.MessageToast.show("New PIN codes dont match");

}

}
else{

 //Form not completely filled in
 sap.m.MessageToast.show("You must fill in the new code please!");


}
}else {

 //Customer entered wrong old pin
 inoSimpleFormSetPinpin.setValue('');
 oInputOldPin.setValue('');
 oInputCheckNewPin.setValue('');
 sap.m.MessageToast.show("Current code entered is wrong");


}
}else{
    //Customer entered a pin thats longer or shorter than 4 characters
    sap.m.MessageToast.show("Code must be of 4 characters");

}
```

Now code for the recovery option if the pin is forgotten by a customer with comments:



```

//Generating a random 4 digit number for the OTP.
var seq = (Math.floor(Math.random() * 10000) + 10000).toString().substring(1);
otp = seq;

//Body of the text message
var smsBodyText = " is your verification code. This is now your PIN password.
var sendBody = otp.concat(smsBodyText);

var myHeaders = new Headers();
//Acesskey to messagebird
myHeaders.append("Authorization", "AccessKey q3nVMIkzrX8RG9vN4gOiZgfGf");
//Content type has to be JSON
myHeaders.append("Content-Type", "application/json");
//Make a body with the originator and the receiver with a verification code
var raw = JSON.stringify({"recipients": currentUser,"originator":"Trye"
                                          ,"body":sendBody});

//Put the body and header in one variable
var requestOptions = {
  method: 'POST',
  headers: myHeaders,
  body: raw,
  redirect: 'follow'
};

//send a request to the messagebird server
fetch("https://rest.messagebird.com/messages", requestOptions)
  .then(response => response.text())
  .then(result => console.log(result))
  .catch(error => console.log('error', error));

//Change pin to new sms PIN
var putPin = {
    parameters: {
        "where": JSON.stringify({"phone": currentUser}) // Optional
    },
    data: {

        "pin": otp,

    }
};

apioRestAPIUsersUpdate(putPin);

//Verify the new pin

oApp.to(oPageVerification);
```

Conclusion:

The change pin tab is working as intended after testing and debugging done by Andreas and Tobias.

A.4.21 USN-53 Make a test payment with Stripe

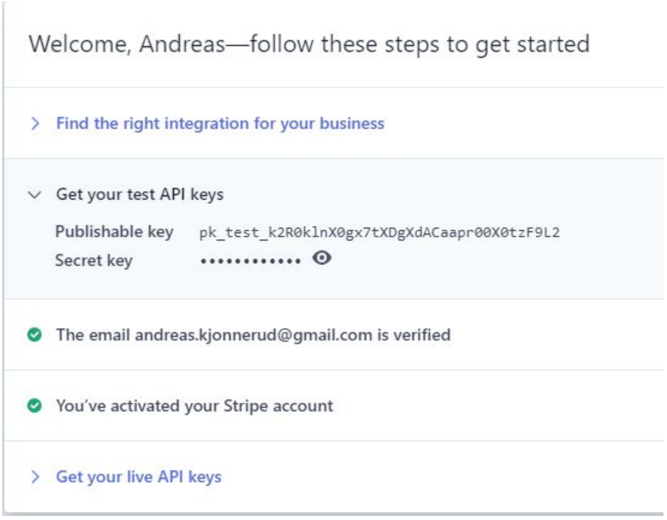
User story			
ID: USN-53	I as a software engineer need to test out the stripe card payment solution, and set up a test server and make a test payment.		
Who Worked	Andreas		
Who Verified	Andreas		
Status	Done		
Requirement			
MAR-03 WSR-02	The mobile app should have a payment system. The web server should be able to communicate with the app.		
Verification			
ID: IV-19	Check whether or not the Stripe payment system works as it should.		
Ver-method	Inspection	Ver-priority	High

Introduction:

Since there is problems with implementing Vipps because of the time waiting to get the api key from them and Paypal has technical difficulties with merchants in norway, stripe came up as an alternative solution. In the first place this user story was created to get familiar with the API in a test environment and make one successful payment to confirm that its working and is ready to be implemented in the app with a live key.

Stripe and the test payment:

To start with a Stripe account was created with Trye's company credentials. To get the license to use their keys a form was submitted and Stripe accepted it within 24 hours. Here is the dashboard with confirmation:



Welcome, Andreas—follow these steps to get started

Get your test API keys

Publishable key pk_test_k2R0klnX0gx7tXDgXdACaapr00X0tzF9L2

Secret key ••••••••••••

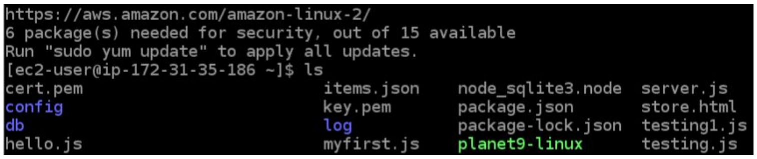
The email andreas.kjonnerud@gmail.com is verified

You've activated your Stripe account

Now that we have the key i found a default stock HTML code for a store online. The only thing needed to be changed in this code is one line and that is the inclusion of the Stripe javascript file. This equals to one line of code in the header of the HTML file:
<script src="https://checkout.stripe.com/checkout.js"></script>.

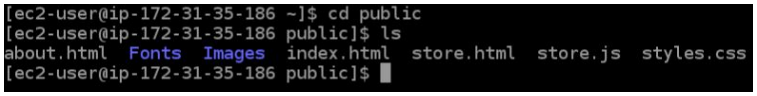
Now the idea is that we use the publishable key to start the payment, and when the details is entered and the purchase is confirmed for the user, what actually happens is that the “false” payment creates a token, which is sent with the secret key to the stripe server, and then the “real” charging of the credit card happens. That is why it is important to keep the secret key hidden or else anyone can charge Trye’s customers what they want. The server is setup using node.js. The script is run on the same server as the planet9 application only from port 3000.

Now for the server script we are using node.js with express which means we can parse the entire store script code from one folder inside the linux server. This means we have the server.js script in the root of the amazon server:



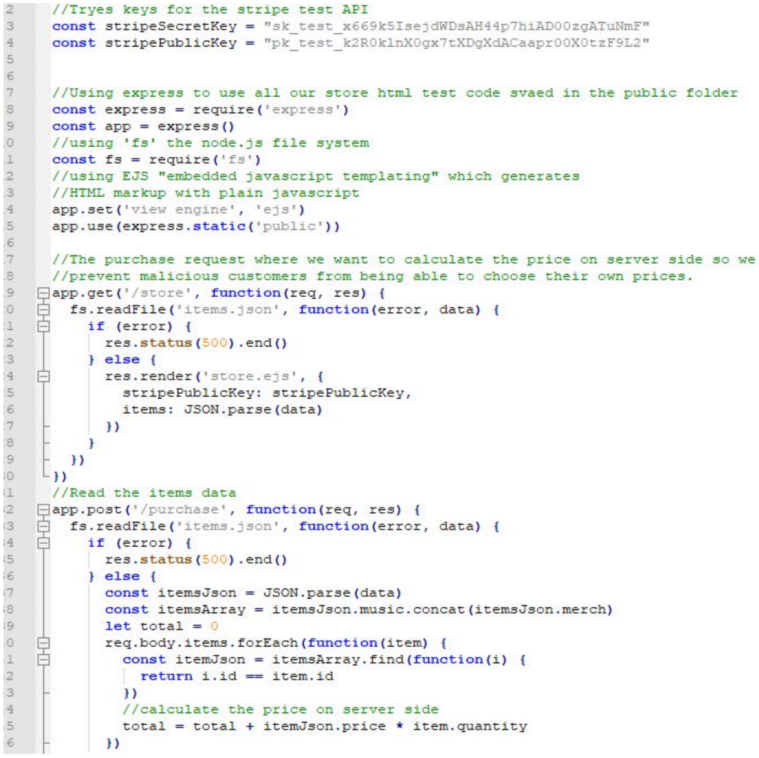
```
https://aws.amazon.com/amazon-linux-2/
6 package(s) needed for security, out of 15 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-35-186 ~]$ ls
cert.pem                         items.json    node_sqlite3.node  server.js
config                           key.pem       package.json       store.html
db                               log           package-lock.json  testing1.js
hello.js                         myfirst.js    planet9-linux      testing.js
```

Now a folder called public is created that will be referenced to express later to run the default store. Using first mkdir then wget from my dropbox i can fetch the entire folder into the linux server. This is how the content of the folder looks:



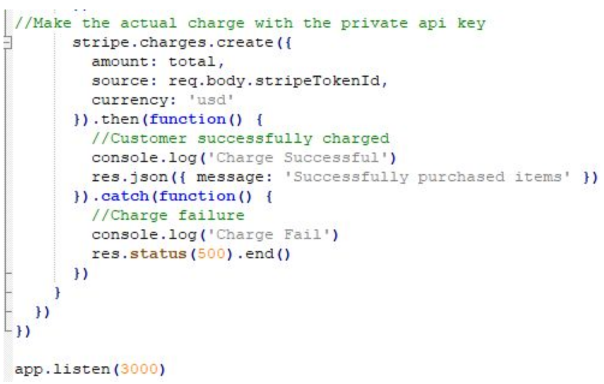
```
[ec2-user@ip-172-31-35-186 ~]$ cd public
[ec2-user@ip-172-31-35-186 public]$ ls
about.html  Fonts  Images  index.html  store.html  store.js  styles.css
[ec2-user@ip-172-31-35-186 public]$ 
```

Now we can make the server script. It will first run the HTML code and then do the test payment when users add items to the cart and click pay. Then it will create a token for the actual charge. Here is the code for the server with comments:



```
//Tryes keys for the stripe test API
const stripeSecretKey = "sk_test_x669k5IsejdWDsAH44p7hiAD00zgATuNmF"
const stripePublicKey = "pk_test_k2R0klnX0gx7tXDgXdACaapr00X0tzF9L2"


//Using express to use all our store html test code svaed in the public folder
const express = require('express')
const app = express()
//using 'fs' the node.js file system
const fs = require('fs')
//using EJS "embedded javascript templating" which generates
//HTML markup with plain javascript
app.set('view engine', 'ejs')
app.use(express.static('public'))

//The purchase request where we want to calculate the price on server side so we
//prevent malicious customers from being able to choose their own prices.
app.get('/store', function(req, res) {
  fs.readFile('items.json', function(error, data) {
    if (error) {
      res.status(500).end()
    } else {
      res.render('store.ejs', {
        stripePublicKey: stripePublicKey,
        items: JSON.parse(data)
      })
    }
  })
})
//Read the items data
app.post('/purchase', function(req, res) {
  fs.readFile('items.json', function(error, data) {
    if (error) {
      res.status(500).end()
    } else {
      const itemsJson = JSON.parse(data)
      const itemsArray = itemsJson.music.concat(itemsJson.merch)
      let total = 0
      req.body.items.forEach(function(item) {
        const itemJson = itemsArray.find(function(i) {
          return i.id == item.id
        })
        //calculate the price on server side
        total = total + itemJson.price * item.quantity
      })
```

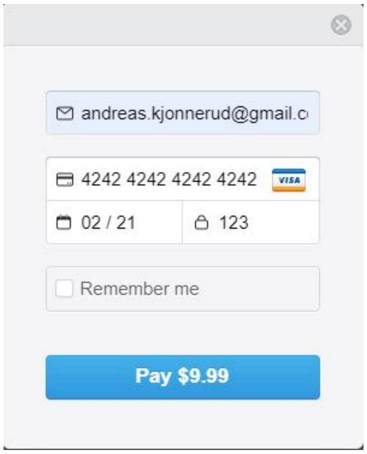
The secret key here is no problem to show as it is not the live key, but only the test key, so the customer will not actually be charged on their credit cards.



```
//Make the actual charge with the private api key
      stripe.charges.create({
        amount: total,
        source: req.body.stripeTokenId,
        currency: 'usd'
      }).then(function() {
        //Customer successfully charged
        console.log('Charge Successful')
        res.json({ message: 'Successfully purchased items' })
      }).catch(function() {
        //Charge failure
        console.log('Charge Fail')
        res.status(500).end()
      })
    }
  })
})

app.listen(3000)
```

Now the payment server is ready to run on port 3000.
To start the program we run: node server.js inside the amazon web server.
Now to test the store, it is opened in the browser by typing the ip address and the port:
http://3.19.14.3:3000/

Now i will add a item of the value 9.99 dollars and make a test payment with the test cars 424242424242 and see if it works:



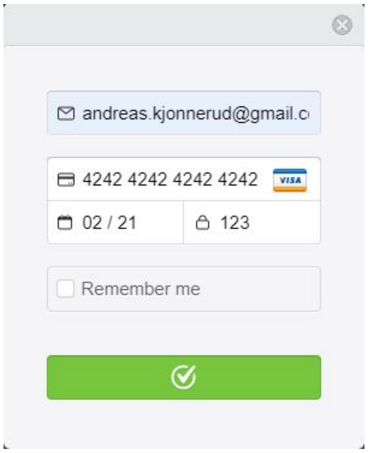
andreas.kjonnerud@gmail.c

4242 4242 4242 4242

02 / 21

123

□ Remember me



andreas.kjonnerud@gmail.c

4242 4242 4242 4242

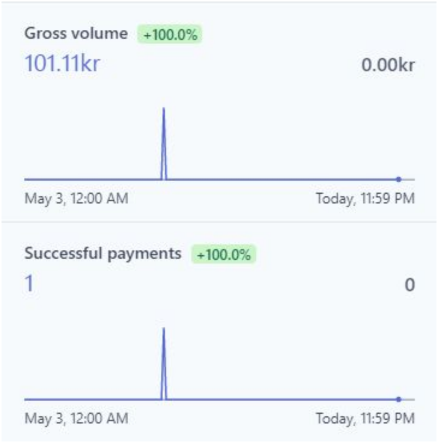
02 / 21

123

□ Remember me

Now we can see that the payment went through.

To confirm we check the Stripe dashboard:



Gross volume +100.0%

101.11kr

0.00kr

May 3, 12:00 AM

Today, 11:59 PM

Successful payments +100.0%

1

0

May 3, 12:00 AM

Today, 11:59 PM

Now 101.11kr is added to our test account, that means we can confirm that the payment went through.

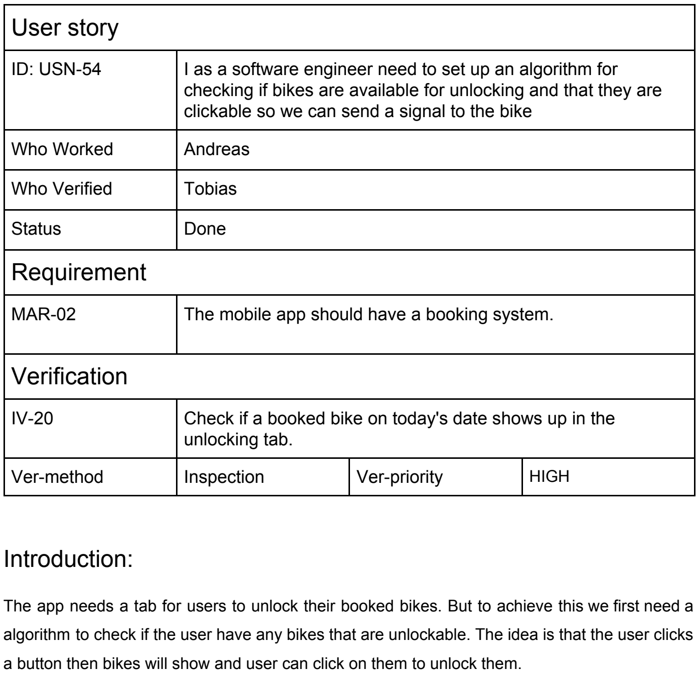
Conclusion:

We now know how to use stripe and are ready to implement it in our app with live keys.

Literature:

https://stripe.com/docs/payments/accept-a-payment

A.4.22 USN-54 Checking if bikes are available for unlocking



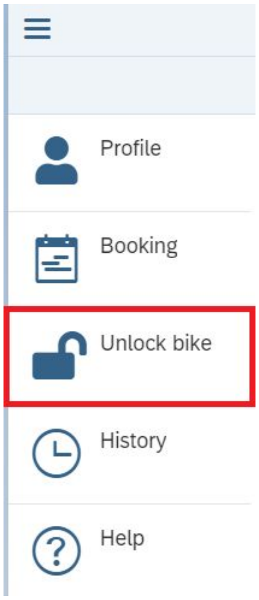
User story			
ID: USN-54	I as a software engineer need to set up an algorithm for checking if bikes are available for unlocking and that they are clickable so we can send a signal to the bike		
Who Worked	Andreas		
Who Verified	Tobias		
Status	Done		
Requirement			
MAR-02	The mobile app should have a booking system.		
Verification			
IV-20	Check if a booked bike on today's date shows up in the unlocking tab.		
Ver-method	Inspection	Ver-priority	HIGH

Introduction:

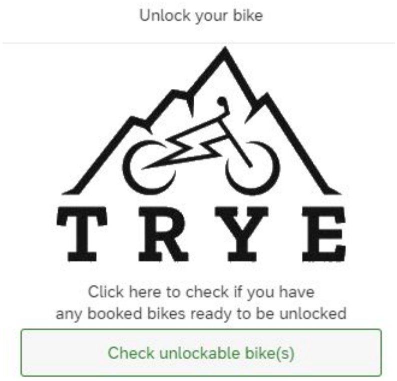
The app needs a tab for users to unlock their booked bikes. But to achieve this we first need a algorithm to check if the user have any bikes that are unlockable. The idea is that the user clicks a button then bikes will show and user can click on them to unlock them.

Check if bikes are available when clicking the button:

A new button is created for the user when they enter the app main menu:

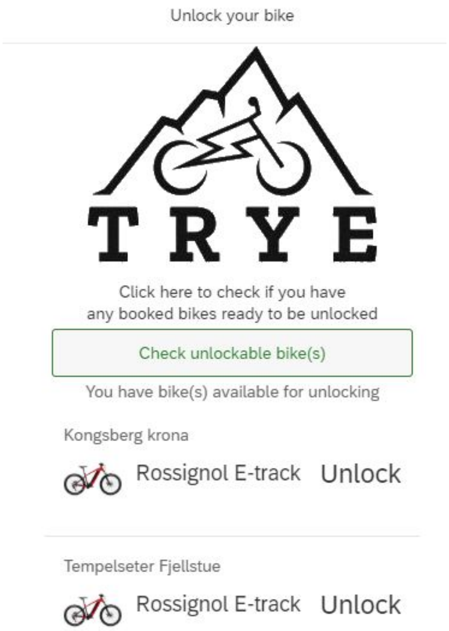


It will open a dialogue with a button that will look like this:



Click here to check if you have any booked bikes ready to be unlocked

When button is pressed the algorithm will find any bikes that are within the range of today's date, so that users can unlock it several times during their rent, and also unlock multiple bikes up to 4 bikes. The reason for this is usually a customer won't need to rent more than 4 bikes which is a normal family or group size. For bigger group they will have to book manually or make more accounts. Now 2 bikes are booked for today, so when the button is pressed the two booked bikes will show:



Unlock your bike

Click here to check if you have any booked bikes ready to be unlocked

You have bike(s) available for unlocking

Kongsberg krona

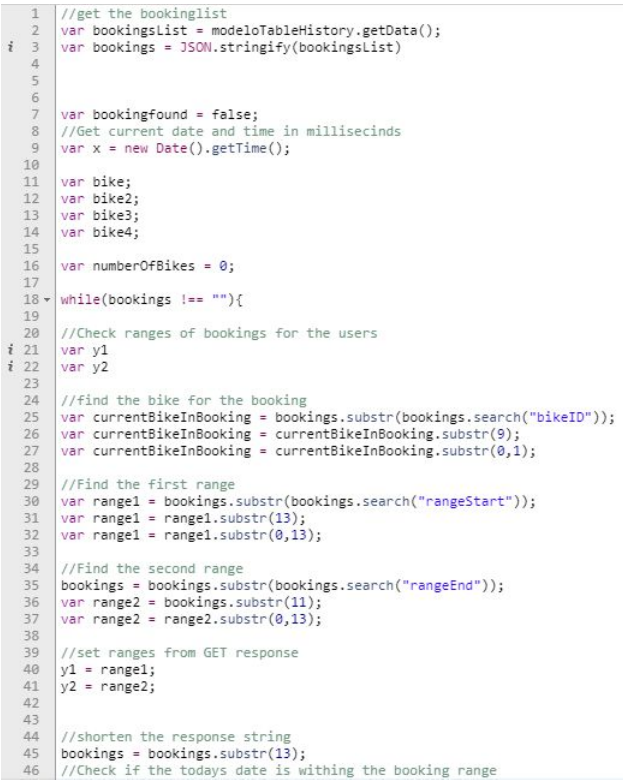
Rossignol E-track

Tempelseter Fjellstue

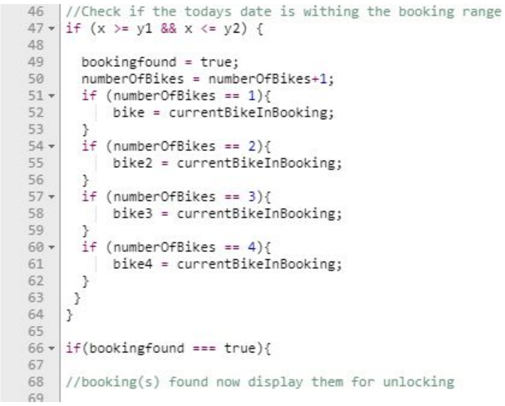
Rossignol E-track

Now for the backend code:

What is used is the same type of loop which was used in the check for booking algorithm. We get all the renting data from the current user and find all ranges of the bookings, and if we have a match we add a bikeID in a variable which we will use to retrieve the bikes in the end.

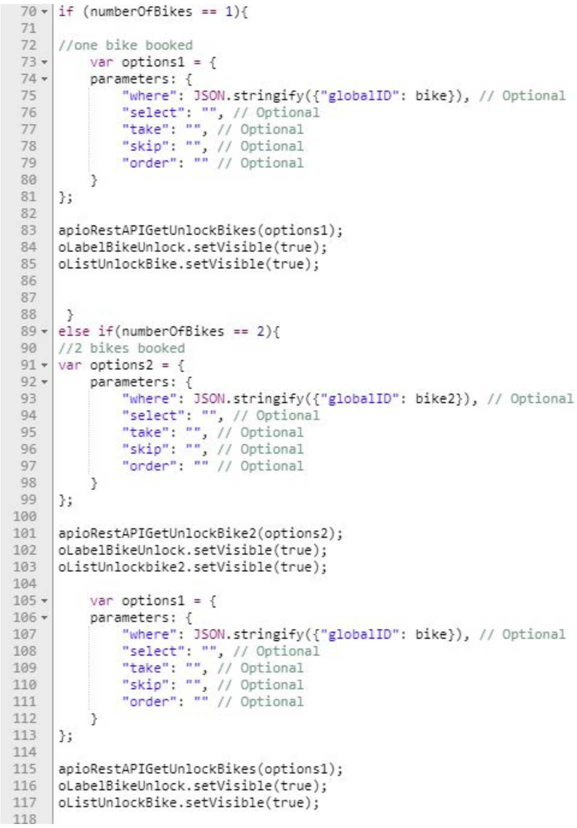


```
//get the bookinglist
var bookingsList = modeloTableHistory.getData();
var bookings = JSON.stringify(bookingsList)



var bookingfound = false;
//Get current date and time in millisecinds
var x = new Date().getTime();

var bike;
var bike2;
var bike3;
var bike4;

var numberOfBikes = 0;

while(bookings !== ""){

//Check ranges of bookings for the users
var y1
var y2

//find the bike for the booking
var currentBikeInBooking = bookings.substr(bookings.search("bikeID"));
var currentBikeInBooking = currentBikeInBooking.substr(9);
var currentBikeInBooking = currentBikeInBooking.substr(0,1);

//Find the first range
var range1 = bookings.substr(bookings.search("rangeStart"));
var range1 = range1.substr(13);
var range1 = range1.substr(0,13);

//Find the second range
bookings = bookings.substr(bookings.search("rangeEnd"));
var range2 = bookings.substr(11);
var range2 = range2.substr(0,13);

//set ranges from GET response
y1 = range1;
y2 = range2;


//shorten the response string
bookings = bookings.substr(13);
//Check if the todays date is withing the booking range
```

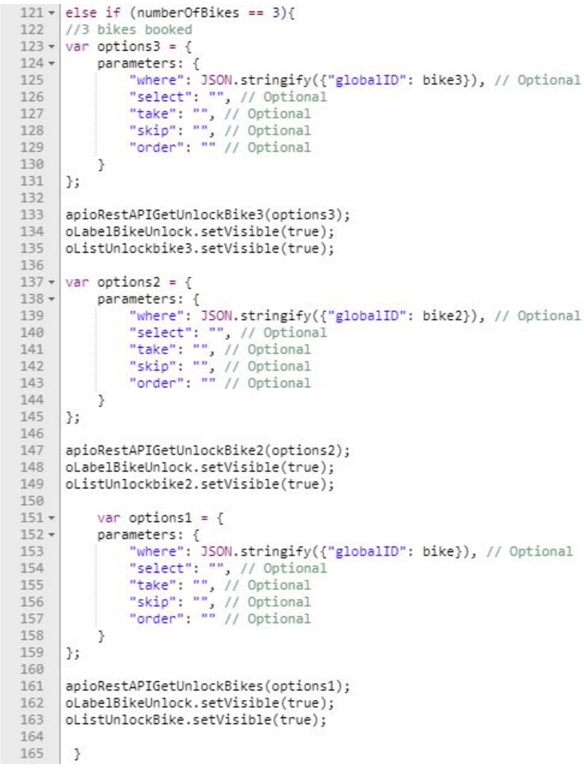


```
//Check if the todays date is withing the booking range
if (x >= y1 && x <= y2) {

  bookingfound = true;
  numberOfBikes = numberOfBikes+1;
  if (numberOfBikes == 1){
      bike = currentBikeInBooking;
  }
  if (numberOfBikes == 2){
      bike2 = currentBikeInBooking;
  }
  if (numberOfBikes == 3){
      bike3 = currentBikeInBooking;
  }
  if (numberOfBikes == 4){
      bike4 = currentBikeInBooking;
  }
 }
}

if(bookingfound === true){

//booking(s) found now display them for unlocking
```

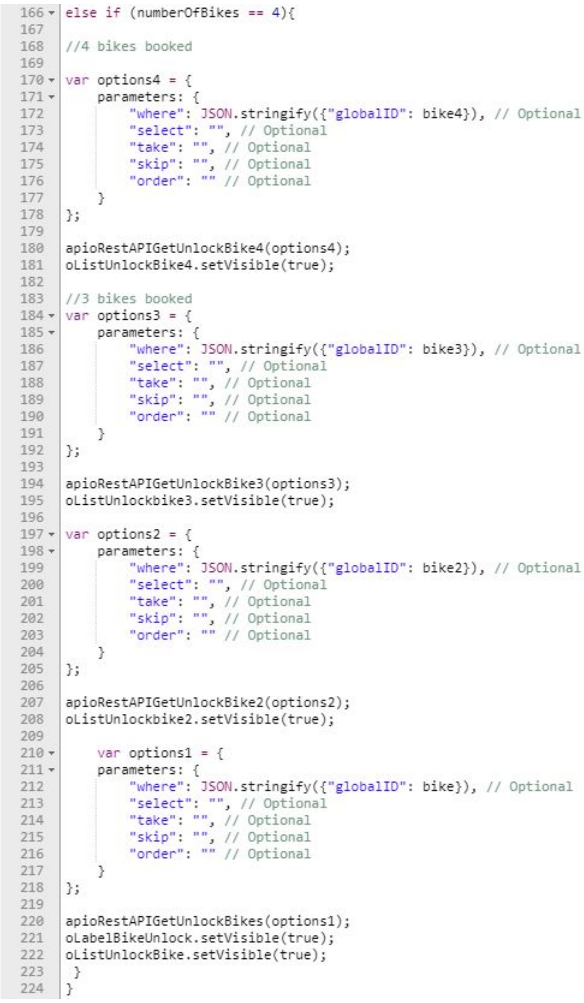
Now we can make the get calls. Since the has been no solution to make GET requests in the planet9 app designer, and the example code dont work it has been coded manually for each of the scenarios:



```
if (numberOfBikes == 1){

//one bike booked
    var options1 = {
    parameters: {
        "where": JSON.stringify({"globalID": bike}), // Optional
        "select": "", // Optional
        "take": "", // Optional
        "skip": "", // Optional
        "order": "" // Optional
    }
};

apioRestAPIGetUnlockBikes(options1);
oLabelBikeUnlock.setVisible(true);
oListUnlockBike.setVisible(true);


 }
else if(numberOfBikes == 2){
//2 bikes booked
var options2 = {
    parameters: {
        "where": JSON.stringify({"globalID": bike2}), // Optional
        "select": "", // Optional
        "take": "", // Optional
        "skip": "", // Optional
        "order": "" // Optional
    }
};

apioRestAPIGetUnlockBike2(options2);
oLabelBikeUnlock.setVisible(true);
oListUnlockbike2.setVisible(true);

    var options1 = {
    parameters: {
        "where": JSON.stringify({"globalID": bike}), // Optional
        "select": "", // Optional
        "take": "", // Optional
        "skip": "", // Optional
        "order": "" // Optional
    }
};

apioRestAPIGetUnlockBikes(options1);
oLabelBikeUnlock.setVisible(true);
oListUnlockBike.setVisible(true);

```

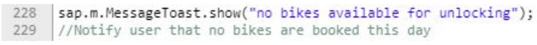


```
else if (numberOfBikes == 3){
//3 bikes booked
var options3 = {
    parameters: {
        "where": JSON.stringify({"globalID": bike3}), // Optional
        "select": "", // Optional
        "take": "", // Optional
        "skip": "", // Optional
        "order": "" // Optional
    }
};

apioRestAPIGetUnlockBike3(options3);
oLabelBikeUnlock.setVisible(true);
oListUnlockbike3.setVisible(true);

var options2 = {
    parameters: {
        "where": JSON.stringify({"globalID": bike2}), // Optional
        "select": "", // Optional
        "take": "", // Optional
        "skip": "", // Optional
        "order": "" // Optional
    }
};

apioRestAPIGetUnlockBike2(options2);
oLabelBikeUnlock.setVisible(true);
oListUnlockbike2.setVisible(true);

    var options1 = {
    parameters: {
        "where": JSON.stringify({"globalID": bike}), // Optional
        "select": "", // Optional
        "take": "", // Optional
        "skip": "", // Optional
        "order": "" // Optional
    }
};

apioRestAPIGetUnlockBikes(options1);
oLabelBikeUnlock.setVisible(true);
oListUnlockBike.setVisible(true);

 }
```



```
else if (numberOfBikes == 4){

//4 bikes booked

var options4 = {
    parameters: {
        "where": JSON.stringify({"globalID": bike4}), // Optional
        "select": "", // Optional
        "take": "", // Optional
        "skip": "", // Optional
        "order": "" // Optional
    }
};

apioRestAPIGetUnlockBike4(options4);
oListUnlockBike4.setVisible(true);

//3 bikes booked
var options3 = {
    parameters: {
        "where": JSON.stringify({"globalID": bike3}), // Optional
        "select": "", // Optional
        "take": "", // Optional
        "skip": "", // Optional
        "order": "" // Optional
    }
};

apioRestAPIGetUnlockBike3(options3);
oListUnlockbike3.setVisible(true);

var options2 = {
    parameters: {
        "where": JSON.stringify({"globalID": bike2}), // Optional
        "select": "", // Optional
        "take": "", // Optional
        "skip": "", // Optional
        "order": "" // Optional
    }
};

apioRestAPIGetUnlockBike2(options2);
oListUnlockbike2.setVisible(true);

    var options1 = {
    parameters: {
        "where": JSON.stringify({"globalID": bike}), // Optional
        "select": "", // Optional
        "take": "", // Optional
        "skip": "", // Optional
        "order": "" // Optional
    }
};

apioRestAPIGetUnlockBikes(options1);
oLabelBikeUnlock.setVisible(true);
oListUnlockBike.setVisible(true);
 }
}
```

And in the end if no bikes are found the user is notified:



```
sap.m.MessageToast.show("no bikes available for unlocking");
//Notify user that no bikes are booked this day
```

Conclusion:

The algorithm works as intended and Andreas and Tobias have verified it by inspection. This means we are ready to integrate the unlocking function with the hardware.

A.4.23 USN-55 Making user be able to delete their account

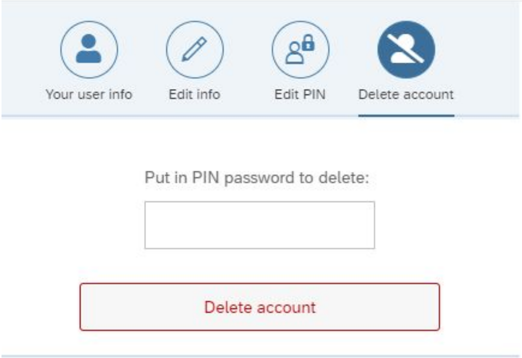
User story			
ID: USN-55	I as a software engineer need to make user be able to delete their account if they no longer want to be customers		
Who Worked	Andreas		
Who Verified	Andreas		
Status	Done		
Requirement			
MAR-02	The mobile app should have a booking system.		
Verification			
IV-21	Check whether or not the account deletion works as it should		
Ver-method	Inspection	Ver-priority	HIGH

Introduction:

This is a function where users enter their pin code to delete their data and be sent back to the registering page of the app after deletion.

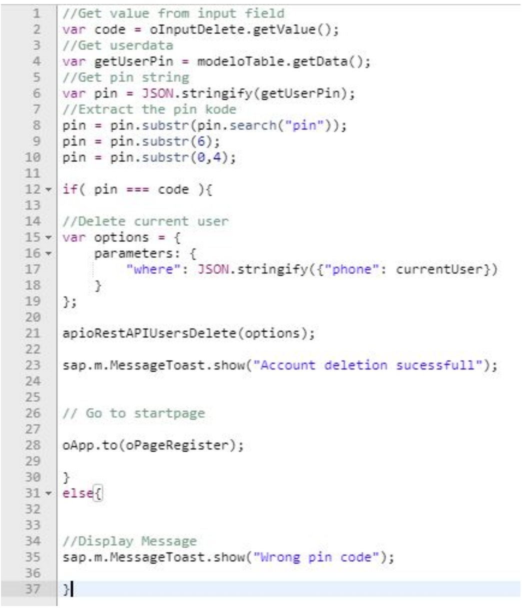
This is to make users that don't want their data stored in our database anymore can delete their account and register again later if they regret deleting it.

The user interface is quite simple with a form that user fill in their personal pin code:



Put in PIN password to delete:

When users press the button the code entered in the form will be compared with the pin code in the database related to the user and they will receive a success or error message. And if it is a success they will be redirected to the first page. Here is the code:

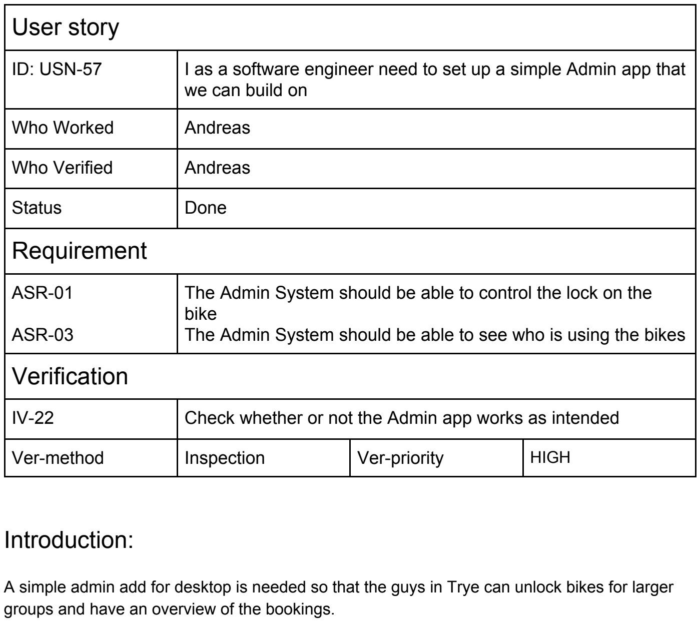


```
//Get value from input field
var code = oInputDelete.getValue();
//Get userdata
var getUserPin = modeloTable.getData();
//Get pin string
var pin = JSON.stringify(getUserPin);
//Extract the pin kode
pin = pin.substr(pin.search("pin"));
pin = pin.substr(6);
pin = pin.substr(0,4);

if( pin === code ){

//Delete current user
var options = {
    parameters: {
        "where": JSON.stringify({"phone": currentUser})
    }
};

apioRestAPIUsersDelete(options);

sap.m.MessageToast.show("Account deletion sucessfull");


// Go to startpage

oApp.to(oPageRegister);

}
else{


//Display Message
sap.m.MessageToast.show("Wrong pin code");

}
```

Conclusion:

The delete function works as intended which was verified by Andreas deleting his account and checking it in the database.

A.4.24 USN-57 Setting up a simple Admin app

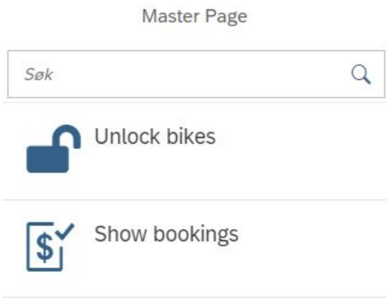


User story			
ID: USN-57	I as a software engineer need to set up a simple Admin app that we can build on		
Who Worked	Andreas		
Who Verified	Andreas		
Status	Done		
Requirement			
ASR-01 ASR-03	The Admin System should be able to control the lock on the bike The Admin System should be able to see who is using the bikes		
Verification			
IV-22	Check whether or not the Admin app works as intended		
Ver-method	Inspection	Ver-priority	HIGH

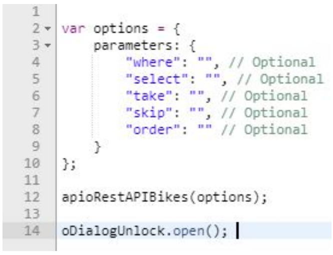
Introduction:

A simple admin add for desktop is needed so that the guys in Trye can unlock bikes for larger groups and have an overview of the bookings.

The admin app is build in a split-app template which is pretty similar to the mobile app. There will be put in 2 components in the first place which is the bike unlock page and the view bookings page. These buttons to redirect to those pages are in the main side menu:

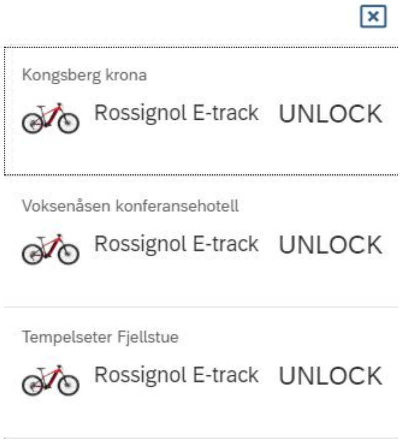


When unlock bikes is clicked we need to call the bike api and list all current bikes, and put it into a list of clickable bikes:



```

var options = {
    parameters: {
        "where": "", // Optional
        "select": "", // Optional
        "take": "", // Optional
        "skip": "", // Optional
        "order": "" // Optional
    }
};

apioRestAPIBikes(options);

oDialogUnlock.open();
```

When the button is clicked the list will look like this:



☒

Kongsberg krona

Rossignol E-track UNLOCK

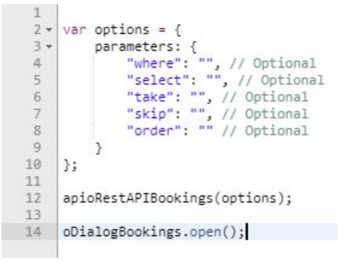
Voksenåsen konferansehotell

Rossignol E-track UNLOCK

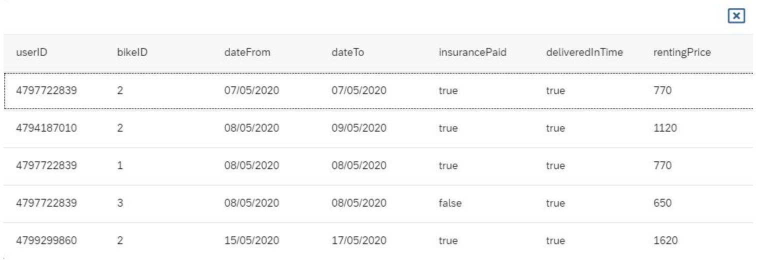
Tempelseter Fjellstue

Rossignol E-track UNLOCK

When the bookings tab is clicked it will also make an database api call with all current bookings and show it in a table:



```

var options = {
    parameters: {
        "where": "", // Optional
        "select": "", // Optional
        "take": "", // Optional
        "skip": "", // Optional
        "order": "" // Optional
    }
};

apioRestAPIBookings(options);

oDialogBookings.open();
```



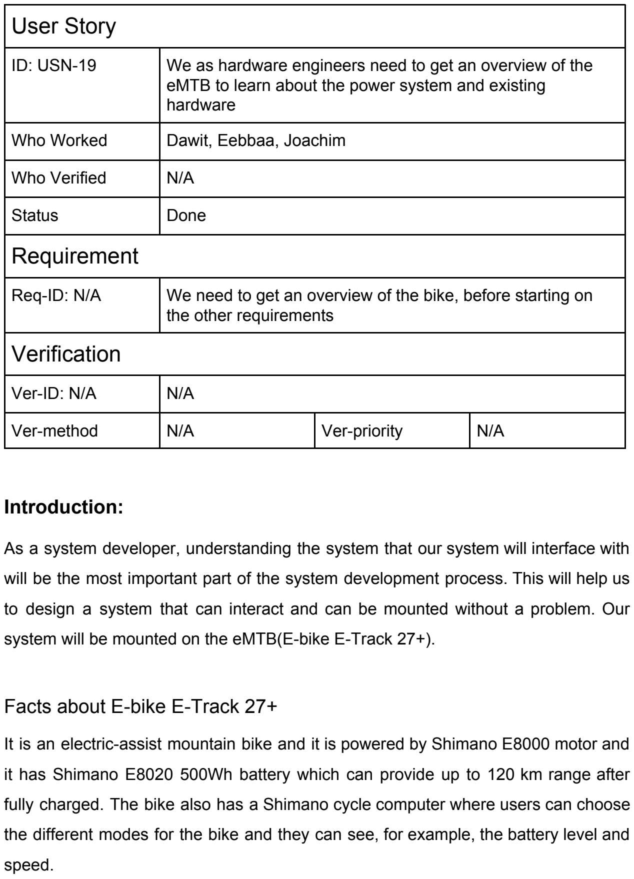
userID	bikeID	dateFrom	dateTo	insurancePaid	deliveredInTime	rentingPrice
4797722839	2	07/05/2020	07/05/2020	true	true	770
4794187010	2	08/05/2020	09/05/2020	true	true	1120
4797722839	1	08/05/2020	08/05/2020	true	true	770
4797722839	3	08/05/2020	08/05/2020	false	true	650
4799299860	2	15/05/2020	17/05/2020	true	true	1620

Conclusion:

The first Alpha of the admin app is finished and is working as intended. Future features that might be added are: map with tracking, view customer list and ban/delete a user.

A.5 Hardware User Stories

A.5.1 USN-19 - Getting an overview of our electric mountainbike, includes the power system and hardware



User Story			
ID: USN-19	We as hardware engineers need to get an overview of the eMTB to learn about the power system and existing hardware		
Who Worked	Dawit, Eebbaa, Joachim		
Who Verified	N/A		
Status	Done		
Requirement			
Req-ID: N/A	We need to get an overview of the bike, before starting on the other requirements		
Verification			
Ver-ID: N/A	N/A		
Ver-method	N/A	Ver-priority	N/A

Introduction:

As a system developer, understanding the system that our system will interface with will be the most important part of the system development process. This will help us to design a system that can interact and can be mounted without a problem. Our system will be mounted on the eMTB(E-bike E-Track 27+).

Facts about E-bike E-Track 27+

It is an electric-assist mountain bike and it is powered by Shimano E8000 motor and it has Shimano E8020 500Wh battery which can provide up to 120 km range after fully charged. The bike also has a Shimano cycle computer where users can choose the different modes for the bike and they can see, for example, the battery level and speed.

Method Used:

To understand more about E-bike E-Track 27+ we read the user manual for getting detailed information about the system and we identified the main subsystems that are important for our system. Fig-1 shows the E-bike we will use for this project and Fig-2 shows the names of the components on the system and Fig-3 shows the subsystems we need most for our system.

Bike subsystems that are necessary for building our system and their specifications are discussed below.

1 Battery (Shimano E8020 500wh)

- Battery life: *1,000 cycles * after 1,000 cycles full charging still more than 300Wh=60% (reference)
- Weight: 3,050 g
- Capacity: 504Wh (36V, 14Ah) [2]

2. Motor (Shimano e8000 MTB drive, 250 W)

The Shimano STEPS E8000 250W motor without frame has a short rear centre to increase slack for the suspension and large tires.
It provides stable assist power and a sense of direct pedalling, with both assistance on and off. It has 70 Nm of torque and a power of 250 W and a 24 mm axle bottom bracket. [3]
Features:
- Power output with stability assistance
- Greater separation (suspension / cover)
- 70 Nm (max.), 250 W
- Lighter than the DU-E 6002
- Improves the handling of the bicycle
- Feeling of direct pedalling when switching on and off the power-assisted
- Compact transmission unit
- Power output

- Lightweight
- Transmission unit characteristics

3. Cycle Computer (SHIMANO STEPS E8000 - Cycle Computer)

This bike computer has a control button which is used for powering on/off the bike computer and changing the different modes of the bike assistant system. the different modes can be chosen are. [4]

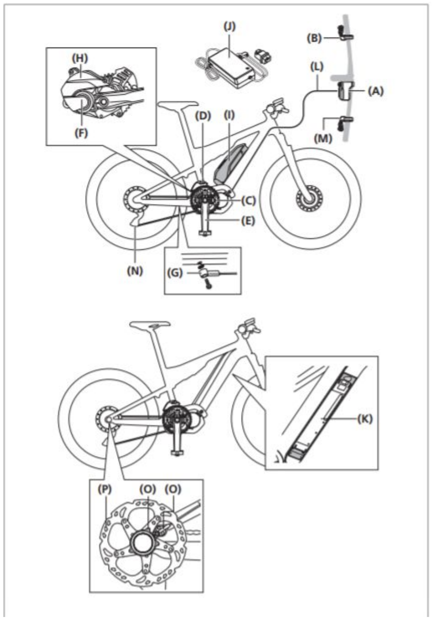
- Eco
- Boost
- Trial

4. SM-DUE 11-Shimano:

It is a magnetic switch and used for calculating the speed of the wheel and it is mounted on the bike of the back wheel.

Figures:

Fig-1 eMTB for the project



(A) Cycle computer/Junction [A]:
SC-E8000/SC-E6010/SC-E6100/
SC-E7000/EW-EN100

(B) Assist switch:
SW-E8000-L/SW-E6010/SW-E7000

(C) Chainring unit:
SM-CRE80/SM-CRE80-B/
SM-CRE80-12-B

(D) Chain device:
SM-CDE80

(E) Crank arm:
FC-E8000/FC-E8050/FC-M8050

(F) Drive unit:
DU-E8000

(G) Speed sensor:
SM-DUE10

(H) Drive unit cover:
SM-DUE80-A
(type that covers drive unit ports)
SM-DUE80-B
(type that covers drive unit ports
and the frame installation bolts)

(I) Battery (external type)/
Battery mount (external type):
BT-E8010/BM-E8010

(J) Battery charger:
EC-E6000

(K) Battery (built-in type)/
Battery mount (built-in type):
BT-E8020/BM-E8020

(L) Electric wire: EW-SD50

When using electronic gear shifting

(M) Shifting switch:
SW-M9050-R/SW-M8050-R/
SW-E6010/SW-E7000

(N) Rear derailleur (DI2):
RD-M9050/RD-M8050

(O) Speed sensor:
SM-DUE11

(P) Disc brake rotor:
RT-EM300/RT-EM600/RT-EM800/
RT-EM810/RT-EM900/RT-EM910

Fig-2 Names of components

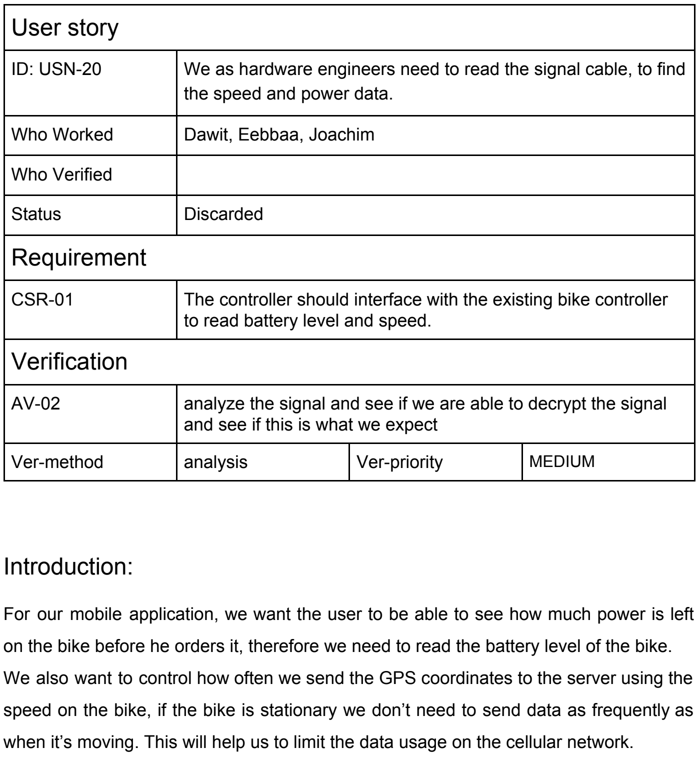
Shimano E8020 500wh
Shimano e8000 MTB drive, 250 W
Shimano e8000 - Cycle Computer
SM-DUE11-Shimano

Fig-3 Subsystems we need for our systems

Literature:

1. https://si.shimano.com/pdfs/dm/DM-E8000-09-ENG.pdf
2. https://bike.shimano.com/en-EU/product/component/mtb-ebike-e8000/BT-E8020.html
3. https://www.deporvillage.net/shimano-steps-e8000-250w-250w-motor-without-frame?country=NO&gclid=Cj0KCQiA7aPyBRChARIsAJfWCgJcmICFm0mDbVi5LJr-L-5nN2ceApXDe2irea5oAV1IA7JIAnNeK5QaAljWEALw_wcB
4. https://bike.shimano.com/en-EU/product/component/mtb-ebike-e8000/SC-E8000.html
5. https://www.sykkelpikene.no/shimano/110352/shimano-speed-sensor-340-mm-e8000-sm-due10?gclid=Cj0KCQiAnL7yBRD3ARIsAJp_oLbU2VCUFOCQmOsGSndmR4vW9iL62ppsXvuwe_xAMagFEwuBTIxeicYaAmdCEALw_wcB
6. https://www.youtube.com/watch?v=F2CJaIM9Uzc
7. https://www.shimano-steps.com/e-bikes/europe/en/product-information/mtb/e8000

A.5.2 USN-20 - Locating read data off of the signal cable to retrieve speed and power data



User story			
ID: USN-20	We as hardware engineers need to read the signal cable, to find the speed and power data.		
Who Worked	Dawit, Eebbaa, Joachim		
Who Verified			
Status	Discarded		
Requirement			
CSR-01	The controller should interface with the existing bike controller to read battery level and speed.		
Verification			
AV-02	analyze the signal and see if we are able to decrypt the signal and see if this is what we expect		
Ver-method	analysis	Ver-priority	MEDIUM

Introduction:

For our mobile application, we want the user to be able to see how much power is left on the bike before he orders it, therefore we need to read the battery level of the bike. We also want to control how often we send the GPS coordinates to the server using the speed on the bike, if the bike is stationary we don't need to send data as frequently as when it's moving. This will help us to limit the data usage on the cellular network.

Methods Used:

On the bike, there is a bike computer/display mounted on the handlebar and this displays the battery level, speed, mode etc. The bike computer needs to get the data from the main controller mounted inside the Shimano motor.

So we disassembled the bike for a better look for the cables that come out from the main controller. We found three cables coming out of the main controller, one connected to the battery, another connected to a reed switch (magnet switch) to read the speed of the wheel and the third is going to the bike computer. To get the data (speed and battery level) we tried to read the signal going to the bike computer. But the reading we got was difficult to conclude which signal is for speed and which one is for battery reading. It is difficult to say what kind of coding and decoding protocols are used in their systems, nothing is explained in their user manuals and websites.

Conclusion:

Because of the difficulty of reading the signal, we decided to find another way of getting the speed and battery reading to our system, which is documented as USN-23 (for reading speed) and USN-24 (for reading battery level) and therefore, USN-20 will be discarded.

Figures:

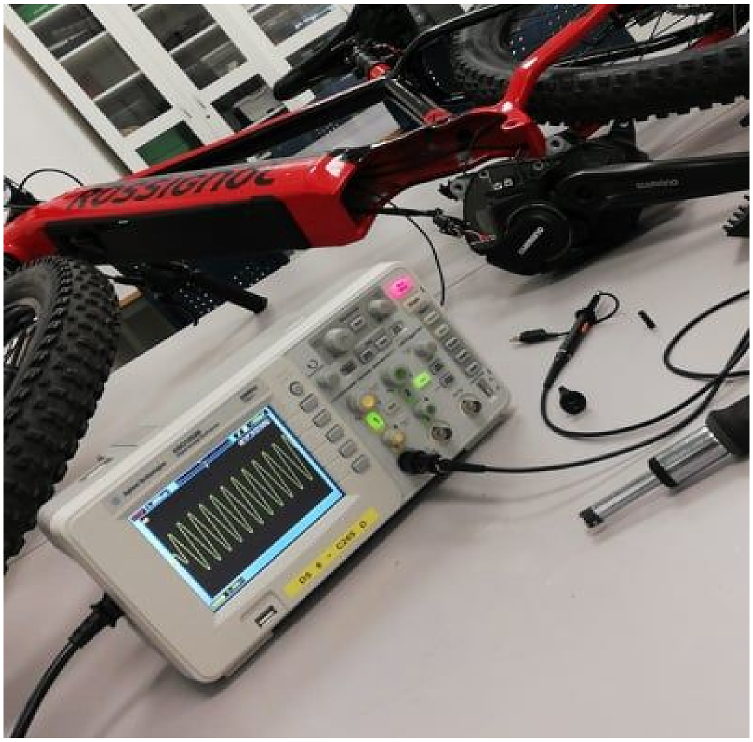


Fig1: A oscilloscope connected to signal cable between the main controller and bike computer

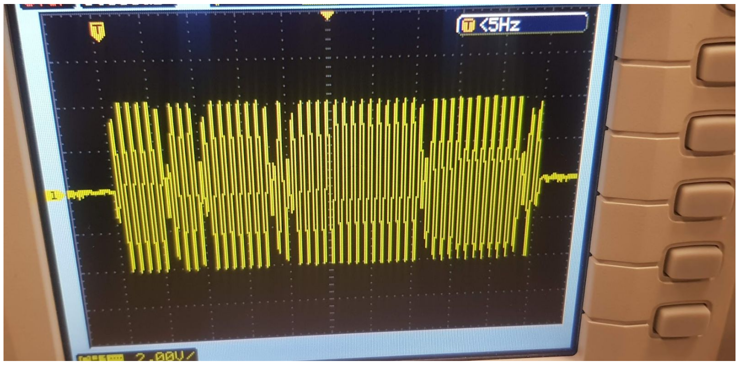
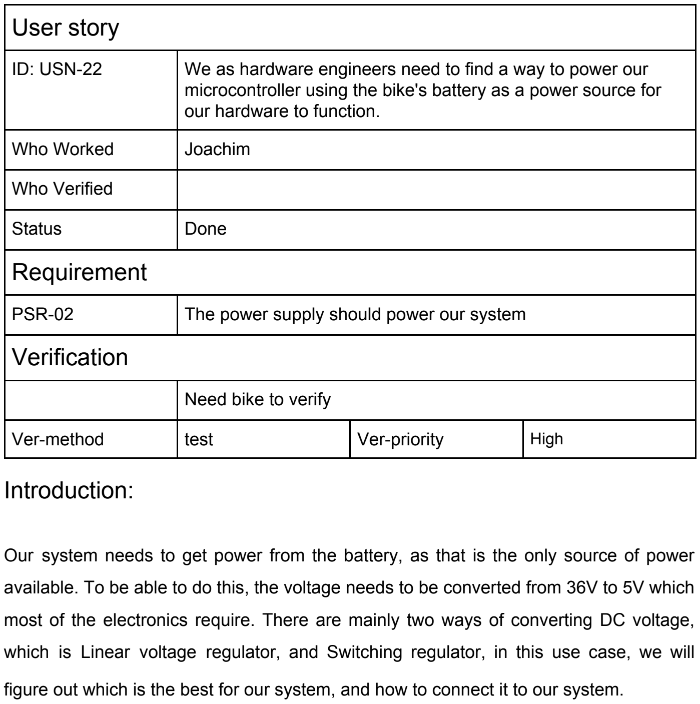


Fig 2: Some of the signal we got from the main controller

A.5.3 USN-22 - Researching ways of powering our micro-contoller



User story			
ID: USN-22	We as hardware engineers need to find a way to power our microcontroller using the bike's battery as a power source for our hardware to function.		
Who Worked	Joachim		
Who Verified			
Status	Done		
Requirement			
PSR-02	The power supply should power our system		
Verification			
	Need bike to verify		
Ver-method	test	Ver-priority	High

Introduction:

Our system needs to get power from the battery, as that is the only source of power available. To be able to do this, the voltage needs to be converted from 36V to 5V which most of the electronics require. There are mainly two ways of converting DC voltage, which is Linear voltage regulator, and Switching regulator, in this use case, we will figure out which is the best for our system, and how to connect it to our system.

Components Used:

- SM-DUE 11-Shimano(with magnetic switch and magnet)
- Microcontroller(Arduino)

Methods Used:

Since reading the speed is essential for our project, we need to figure out another option. For reading the speed we have considered the magnet that is attached to the wheel spake and magnet sensor that is attached on the bike near to it. To read the signals from the reed switch we used an Arduino connected as seen in figure Fig1 and Fig2. The Arduino will detect when the switch is on and off, and count the time between each consecutive 1's. Two consecutive 1's in a given interval of time shows one revolution of the wheel. From this, we can calculate the speed at which the bike is travelling using the following formula.

Speed = (distance traveled / time taken) …………….....for linear motion

Rotational speed = (angle swept/ time taken)…………...for rotational motion

1 Revolution = 360 degree = 2*pi Radians.

Wheel diameter of our bike 27.5 INCH = 0.7 m is given on the specification.

Fig-3 illustration for rotational motion of the wheel

so based on this concept we tried to make an Arduino code that reads the number of 1’s in the given interval for calculating the speed. In this case, we tried to calculate the average speed for every five rotations made by the wheel. and also we tried to calculate the momentary speed for the last rotation. If the Arduino does not receive a signal within 5 seconds then the program gets a timeout, when this happens it is because the bike is stationary, and the speed is set to 0 km/h.

Explanation of the Program:

there are 4 different functions that takes care of finding the speed of the bike:

- *findDeltaTime()*
- *findAvgDeltaTime()*
- *CheckTimeout()*
- *calculateSpeed()*

findDeltaTime() function, returns the difference in time between each wheel rotation in ms and stores it in an array of an arbitrary length of five. The array is to find the average time later in the program.

findAvgDeltaTime() function, returns the average time calculated by summing the array from *findDeltaTime()* and then dividing it by five which is the length of the array.

CheckTimeout(): the timeout function checks to see if the wheel is not rotating within five seconds. If the bike is stationary, the function *findDeltaTime()* and *findAvgDeltaTime()* will not work as they are dependent on the wheel spinning. So if that is the case, the CheckTimeout overwrites the value from the other functions and sets the change in time to 5 seconds. Which will later return as 0 km/h later in the program

calculateSpeed() function takes the deltaTime and average deltaTime and, and uses that to find the RPM of the wheel and using the circumstance to calculate the speed of the bike. If the speed is below a cutoff speed which is currently set to 2km/h, the speed is set to 0km/h.

Conclusion

The code calculates the speed as we expected but we need to verify the result with the bike computer result which is mounted on the existing system.

Arduino-code:
https://drive.google.com/drive/folders/1mZtdGGWajH00zXwDTy55GxgRBbEkVkdx

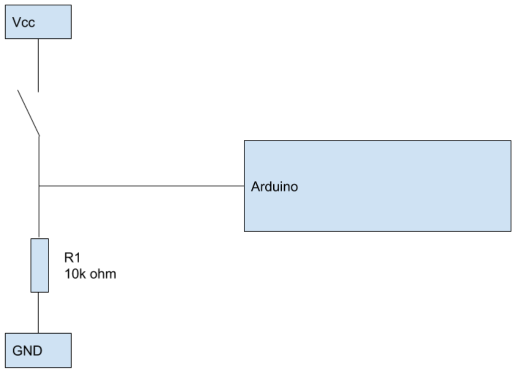


Fig1: A switch with a pulldown resistor

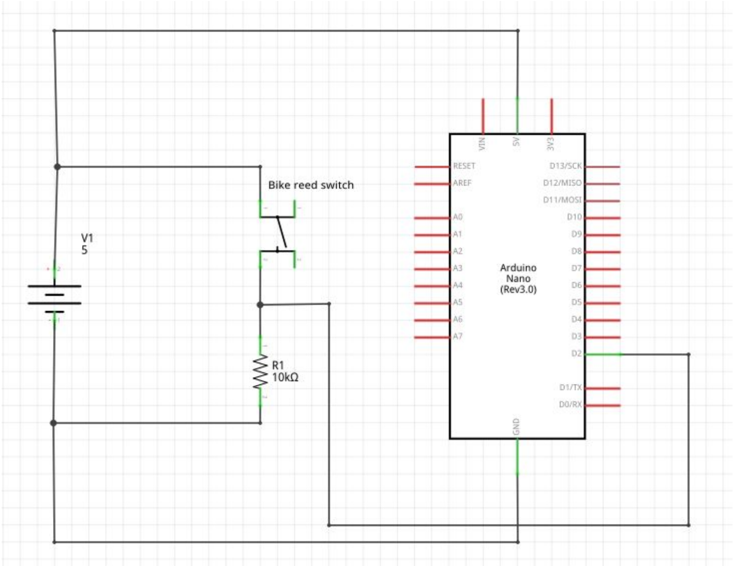
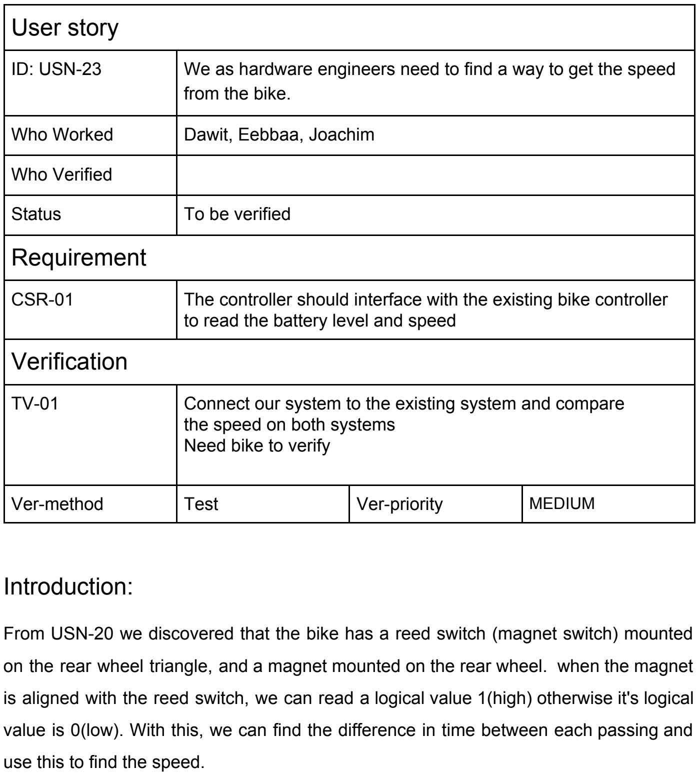


Fig2: A more detailed view of Fig1

A.5.4 USN-23 - Finding out how to retrieve the speed of the bike



User story			
ID: USN-23	We as hardware engineers need to find a way to get the speed from the bike.		
Who Worked	Dawit, Eebbaa, Joachim		
Who Verified			
Status	To be verified		
Requirement			
CSR-01	The controller should interface with the existing bike controller to read the battery level and speed		
Verification			
TV-01	Connect our system to the existing system and compare the speed on both systems Need bike to verify		
Ver-method	Test	Ver-priority	MEDIUM

Introduction:

From USN-20 we discovered that the bike has a reed switch (magnet switch) mounted on the rear wheel triangle, and a magnet mounted on the rear wheel. when the magnet is aligned with the reed switch, we can read a logical value 1(high) otherwise it's logical value is 0(low). With this, we can find the difference in time between each passing and use this to find the speed.

Components Used:

- SM-DUE 11-Shimano(with magnetic switch and magnet)
- Microcontroller(Arduino)

Methods Used:

Since reading the speed is essential for our project, we need to figure out another option. For reading the speed we have considered the magnet that is attached to the wheel spake and magnet sensor that is attached on the bike near to it. To read the signals from the reed switch we used an Arduino connected as seen in figure Fig1 and Fig2. The Arduino will detect when the switch is on and off, and count the time between each consecutive 1's. Two consecutive 1's in a given interval of time shows one revolution of the wheel. From this, we can calculate the speed at which the bike is travelling using the following formula.

Speed = (distance traveled / time taken) …………….....for linear motion

Rotational speed = (angle swept/ time taken)…………...for rotational motion

1 Revolution = 360 degree = 2*pi Radians.

Wheel diameter of our bike 27.5 INCH = 0.7 m is given on the specification.

Fig-3 illustration for rotational motion of the wheel

so based on this concept we tried to make an Arduino code that reads the number of 1's in the given interval for calculating the speed. In this case, we tried to calculate the average speed for every five rotations made by the wheel. and also we tried to calculate the momentary speed for the last rotation. If the Arduino does not receive a signal within 5 seconds then the program gets a timeout, when this happens it is because the bike is stationary, and the speed is set to 0 km/h.

Explanation of the Program:

there are 4 different functions that takes care of finding the speed of the bike:

- *findDeltaTime()*
- *findAvgDeltaTime()*
- *CheckTimeout()*
- *calculateSpeed()*

findDeltaTime() function, returns the difference in time between each wheel rotation in ms and stores it in an array of an arbitrary length of five. The array is to find the average time later in the program.

findAvgDeltaTime() function, returns the average time calculated by summing the array from *findDeltaTime()* and then dividing it by five which is the length of the array.

CheckTimeout(): the timeout function checks to see if the wheel is not rotating within five seconds. If the bike is stationary, the function *findDeltaTime()* and *findAvgDeltaTime()* will not work as they are dependent on the wheel spinning. So if that is the case, the CheckTimeout overwrites the value from the other functions and sets the change in time to 5 seconds. Which will later return as 0 km/h later in the program

calculateSpeed() function takes the deltaTime and average deltaTime and, and uses that to find the RPM of the wheel and using the circumstance to calculate the speed of the bike. If the speed is below a cutoff speed which is currently set to 2km/h, the speed is set to 0km/h.

Conclusion

The code calculates the speed as we expected but we need to verify the result with the bike computer result which is mounted on the existing system.

Arduino-code:
https://drive.google.com/drive/folders/1mZtdGGWajH00zXwDTy55GxgRBbEkVkdx

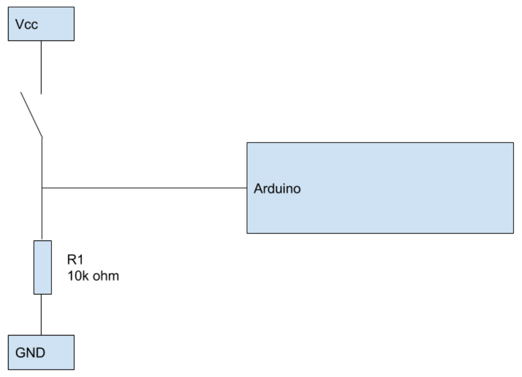


Fig1: A switch with a pulldown resistor

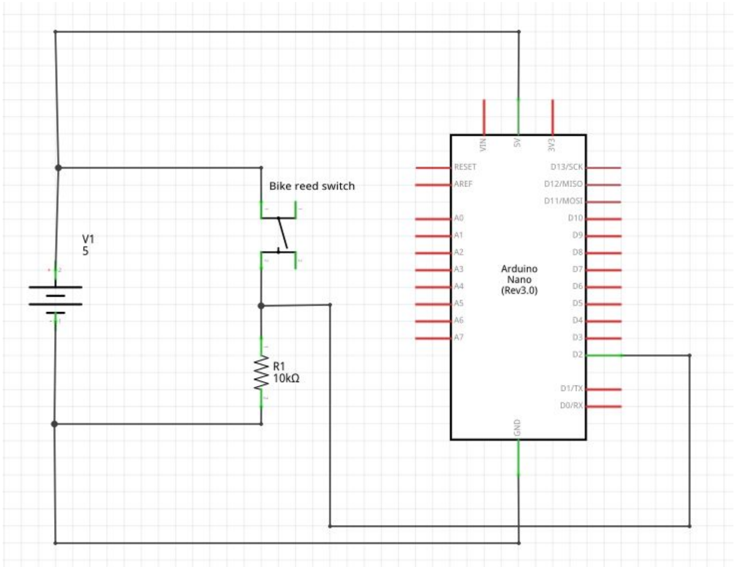
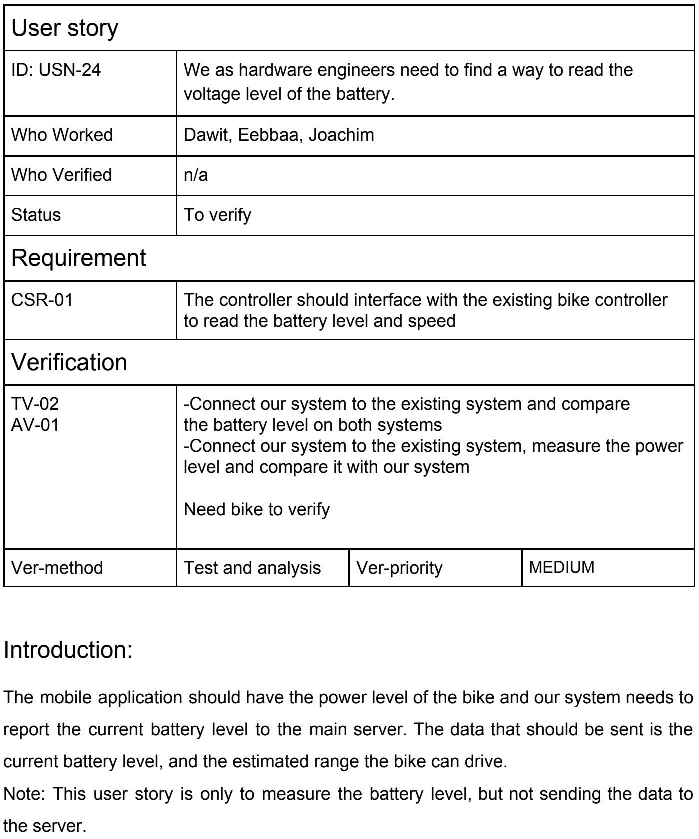


Fig2: A more detailed view of Fig1

A.5.5 USN-24 - Finding out how to retrieve the voltage level of the battery



User story			
ID: USN-24	We as hardware engineers need to find a way to read the voltage level of the battery.		
Who Worked	Dawit, Eebbaa, Joachim		
Who Verified	n/a		
Status	To verify		
Requirement			
CSR-01	The controller should interface with the existing bike controller to read the battery level and speed		
Verification			
TV-02 AV-01	-Connect our system to the existing system and compare the battery level on both systems -Connect our system to the existing system, measure the power level and compare it with our system Need bike to verify		
Ver-method	Test and analysis	Ver-priority	MEDIUM

Introduction:

The mobile application should have the power level of the bike and our system needs to report the current battery level to the main server. The data that should be sent is the current battery level, and the estimated range the bike can drive.

Note: This user story is only to measure the battery level, but not sending the data to the server.

Methods Used:

Hardware: the expected maximum voltage level on the battery is 36 Volt and as the Arduino only can handle 0-5V on its analogue pin, because of this the voltage has to be stepped down by using a voltage divider. By using a resistor of 10k Ohms and 1,6k Ohms connected as seen in Fig1, this way the output voltage to the Arduino ranges from 0-5 Volt. the Arduino has a 10 bit ADC (Analog Digital Converter) that means every reading will range from 0-1023

Firmware: the firmware reads the digital conversion of the analogue signal ranging from 0-1023 and maps it to 0-36 Volt. The. Arduino then needs to map the 0-36 Volt to a 0-100% battery level. To do this the Arduino needs to know what 0% is in Volt, this can be figured out by discharging the battery and measuring the voltage level using a voltmeter and changing the parameters in the code to match the results.ch

The battery level can then be converted to an easy to understand battery indicator as seen in Fig2. The battery level indicator that is shown from black color to white is the battery level with the empty charge to full charge. 0% means that the battery is empty, whereas 100-81% is fully charged.

Figures

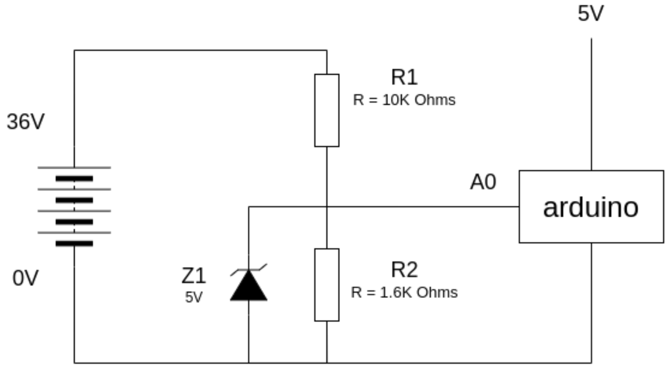
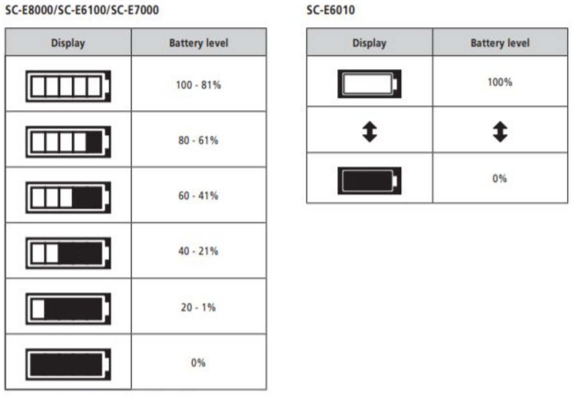


Fig1: A voltage divider giving 0-5 volt on A0.



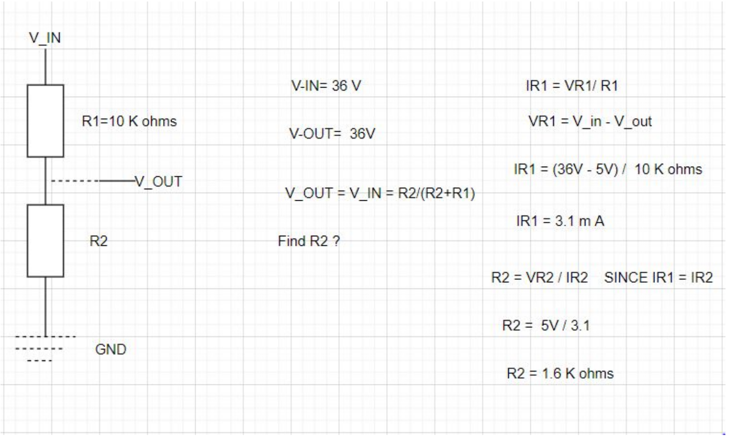
SC-E8000/SC-E6100/SC-E7000

Display	Battery level
□□□□□	100 - 81%
□□□□■	80 - 61%
□□□■■	60 - 41%
□□■■■	40 - 21%
□■■■■	20 - 1%
■■■■■	0%

SC-E6010

Display	Battery level
□	100%
↕	↕
■	0%

Fig2: The battery level indicator, on the existing system.



V-IN= 36 V

V-OUT= 36V

V_OUT = V_IN = R2/(R2+R1)

Find R2 ?

IR1 = VR1/ R1

VR1 = V_in - V_out

IR1 = (36V - 5V) / 10 K ohms

IR1 = 3.1 m A

R2 = VR2 / IR2   SINCE IR1 = IR2

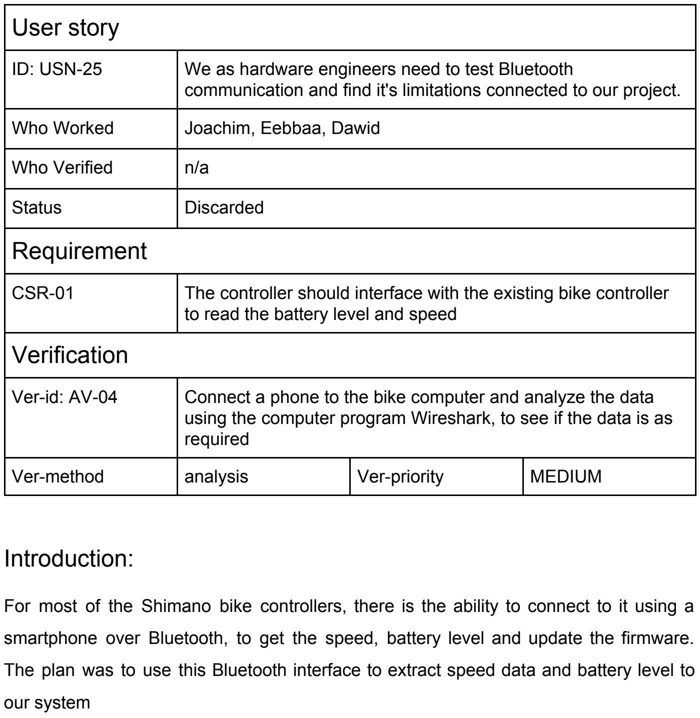
R2 = 5V / 3.1

R2 = 1.6 K ohms

Therefore, we select an R2 = 1.6Kohm.

Fig3: Calculating the R2 shown in Fig1

A.5.6 USN-25 - Finding the limitations associated with Bluetooth communication



User story			
ID: USN-25	We as hardware engineers need to test Bluetooth communication and find it's limitations connected to our project.		
Who Worked	Joachim, Eebbaa, Dawid		
Who Verified	n/a		
Status	Discarded		
Requirement			
CSR-01	The controller should interface with the existing bike controller to read the battery level and speed		
Verification			
Ver-id: AV-04	Connect a phone to the bike computer and analyze the data using the computer program Wireshark, to see if the data is as required		
Ver-method	analysis	Ver-priority	MEDIUM

Introduction:

For most of the Shimano bike controllers, there is the ability to connect to it using a smartphone over Bluetooth, to get the speed, battery level and update the firmware. The plan was to use this Bluetooth interface to extract speed data and battery level to our system

Method Used:

To find out what data is sent back and forth between the bike computer and a smartphone it’s possible to use a tool called Wireshark. What Wireshark does is “capturing” packets on wireless channels, like Bluetooth. This is an easy and fast way to get the bike computers MAC-address and UUID (Universally Unique IDentifier).

By using this information it’s feasible to connect to the bike computer using a Bluetooth terminal. In such a way, while the bike computer and phone are communicating, the terminal shows all the data that is sent and received in between. When we got that data, we figured out the bike computer of the model (Insert model number) does not send speed and battery level over Bluetooth, and was therefore not possible to extract the data. The only thing that you can do over Bluetooth is update firmware, customize bike setting, and diagnose problems using the E-Tube app on this exact model. There are two E-Tube apps, the one used for our model is the E-Tube project, which is the one used to update firmware and change settings, and then there is E-Tube Ride which shows speed and battery level. E-Tube Ride is not supported by our bike computer as it does not send out speed and battery level over Bluetooth.

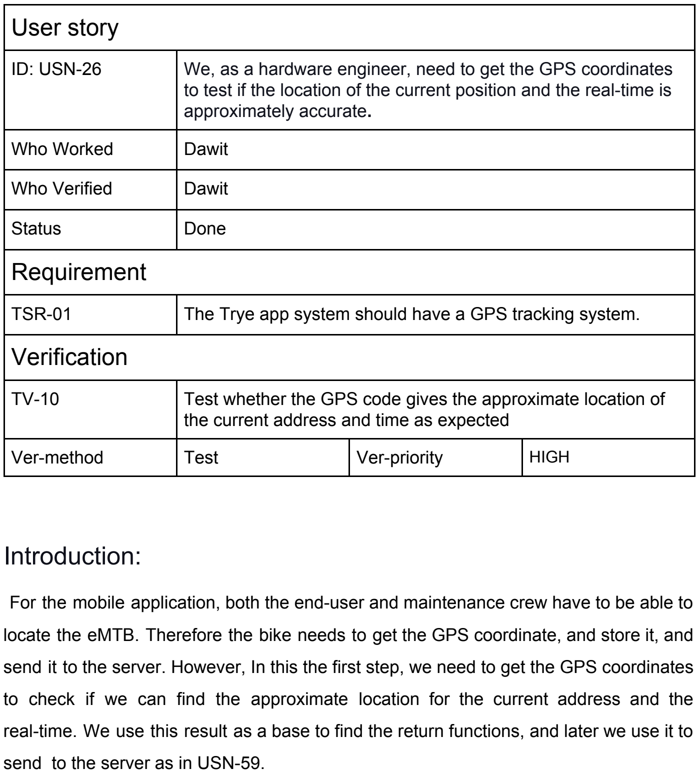
Conclusion:

Because the bike computer does not send out speed and battery level over Bluetooth we decided to discard this user story and continue with USN-23 and USN-24 instead.

Literature:

E-Tube Ride: https://e-tuberide.shimano.com/?lang=en
E-Tube Project: https://e-tubeproject.shimano.com/about/

A.5.7 USN-26 - Finding out how to retrieve the GPS coordinates



User story			
ID: USN-26	We, as a hardware engineer, need to get the GPS coordinates to test if the location of the current position and the real-time is approximately accurate**.**		
Who Worked	Dawit		
Who Verified	Dawit		
Status	Done		
Requirement			
TSR-01	The Trye app system should have a GPS tracking system.		
Verification			
TV-10	Test whether the GPS code gives the approximate location of the current address and time as expected		
Ver-method	Test	Ver-priority	HIGH

Introduction:

For the mobile application, both the end-user and maintenance crew have to be able to locate the eMTB. Therefore the bike needs to get the GPS coordinate, and store it, and send it to the server. However, In this the first step, we need to get the GPS coordinates to check if we can find the approximate location for the current address and the real-time. We use this result as a base to find the return functions, and later we use it to send to the server as in USN-59.

Method Used:

In the beginning, we have planned to use the NEO-7M GPS module. Eventually, we have changed our minds to use the NEO-6M GPS module. It is because when we compare NEO-7M with the NEO-6M module for the same conditions, its sensor receives the data slower than the NEO-6M module. The NEO-7M module is also blinking the light as if it were collecting the data from the satellites. However, in the case of the NEO-6M module, the sensor receives the data faster and blinks blue-light whenever it receives data from the satellites. This module is also comparatively cheap and straightforward to use. These are the main reasons that we have selected the NEO-6M module for GPS tracking.

To apply the NEO-6M module, as shown in figure 1 below, we have connected it with its four jumper wires to the Arduino. Then the Arduino is also connecting with the computer by using the USB serial communication. The GPS module to Arduino, use a +5V from the power side of the Arduino and any ground pin. The two pins work for serial communication. By doing this, we are on the right track to get the GPS coordinates that use for positioning the bike in real-time. The data that we are looking for are Latitude, Longitude, and Time.

To accomplish this task, we have done the code and loaded it into the Arduino. The code loaded into the Arduino used by the use of two important libraries, such as TinyGPS ++ and SoftwareSerial. They are used to provide most of the NMEA GPS functionality, avoid any mandatory floating point dependency, and ignore all except the few key GPS fields.We can find the code in figure 2 below.

Conclusion:

Using the code for GPS coordinate, we have found that the latitude, longitude, and time where latitude and longitude are the locations of the current place, and the time is the same as the Norwegian time. However, the code we have used in this user story needs to convert to the class library so that we can send it to the server to locate the position of the bike and find the real-time. We can do this in USN-59.

Figures:

Fig1. The interface of Arduino with GPS Module(NEO-6M)

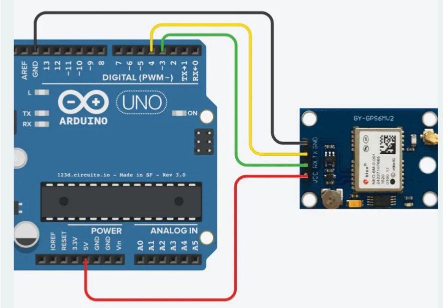


Fig 2. GPS coordinates code

1.[Link_code]

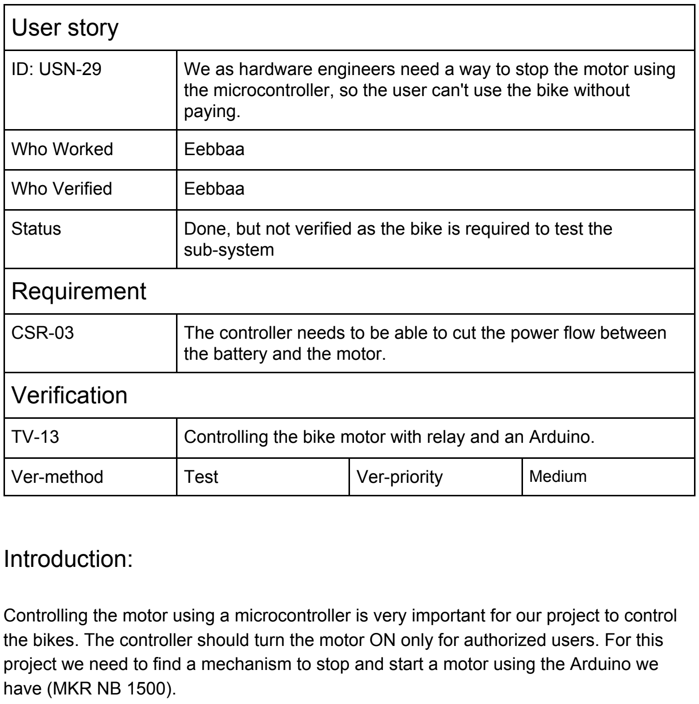
Literature:

[1]https://randomnerdtutorials.com/guide-to-neo-6m-gps-module-with-arduino/

[2]https://create.arduino.cc/projecthub/ruchir1674/how-to-interface-gps-module-neo-6m-with-arduino-8f90ad

[3]https://www.arduino.cc/en/Guide/ArduinoUn

A.5.8 USN-29 Finding a way to stop the motor using a microcontroller



User story			
ID: USN-29	We as hardware engineers need a way to stop the motor using the microcontroller, so the user can't use the bike without paying.		
Who Worked	Eebbaa		
Who Verified	Eebbaa		
Status	Done, but not verified as the bike is required to test the sub-system		
Requirement			
CSR-03	The controller needs to be able to cut the power flow between the battery and the motor.		
Verification			
TV-13	Controlling the bike motor with relay and an Arduino.		
Ver-method	Test	Ver-priority	Medium

Introduction:

Controlling the motor using a microcontroller is very important for our project to control the bikes. The controller should turn the motor ON only for authorized users. For this project we need to find a mechanism to stop and start a motor using the Arduino we have (MKR NB 1500).

Method Used:

A relay will be used as an electrical switch between the motor and the battery. Relay is a programmable switch that can be controlled by microcontrollers and can be used to control devices. To control a high power system with an Arduino we need to isolate the Arduino with the help of a relay. We control the relay state by using Arduino and the relay state will be used to control the device state either in ON or OFF modes.

Relays are designed for handling and switching high-voltage or high power circuits. They have an electromagnet that can be energized and results in a switch to close or open. Relays have 5 pins (COM, Coil 1, Coil 2, NC, NO) and among these 3 pins(NC, COM, and NO) will be connected to the device to be controlled. At the COM terminal electricity enters the relay and NC and NO terminals are used to turn ON and OFF devices. Between pins of Coil 1 and Coil 2 there is an electromagnet coil, and whenever a current passes through them the electromagnet charges and the internal switch connects COM and NO pins as a result the device will be in ON state.

Our main goal is to control the motor of the Shimano eBike by using an Arduino and a relay. As we can see on the wiring of the Shimano eBike in Figure-2, the battery is connected directly to the motor by green wire. In normal operation when we press the start button of the bike the motor will get power from the battery directly. Our plan is to use an Arduino to send a control signal '0' or '1' to the relay, as a result the motor will be either in ON or OFF state. Adding the relay and Arduino at the middle of the existing wiring diagram will replace the task of the start button, which means to start the motor we just need to send logical value '1' to the pin of an Arduino where the relay is connected and the relay will act as a switch to turn ON the motor. How the battery, relay, and the Arduino board should be connected as shown in Figure-3.

Controlling the motor using an Arduino will help us to manage control of a bike from our cloud platform by sending commands to the Arduino board to control the state of the motor either ON or OFF. Having these functionality helps us to control our bikes remotely based on the user status, for example users who have completed the renting process on the user app should get the access to turn ON/OFF the bike motors.

How the cloud platform sends commands to the Arduino board based on the status of users will be discussed on USN-58.

Contribution:

Joachim:

- Helped brainstorm with using a relay to control a motor
- Showed the basic usage of a relay, and how it worked in the Arduino environment

Conclusion:

Based on the connection in Figure-3 what we need is to send a digital signal(‘1’ or ‘0’) from Arduino to trigger the relay to act as a switch to control the motor. Using Relay makes the motor control easy as controlling LED light using an Arduino by sending ‘0’ or ‘1’.

Arduino-code:

https://drive.google.com/drive/folders/1OFulkHalwmcBEO-SRi6yuWyUCd81gVvX

Figures:

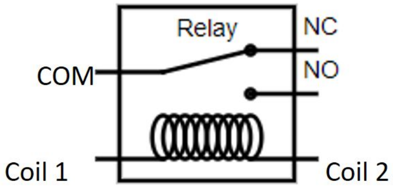


Figure-1 Pins of a Relay

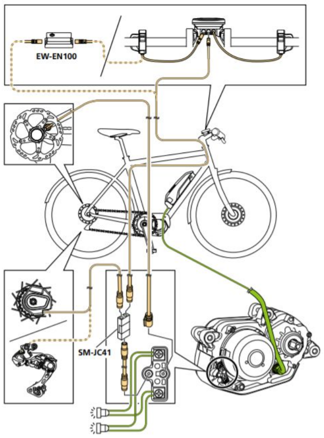


Figure-2 Wiring of Shimano eBike

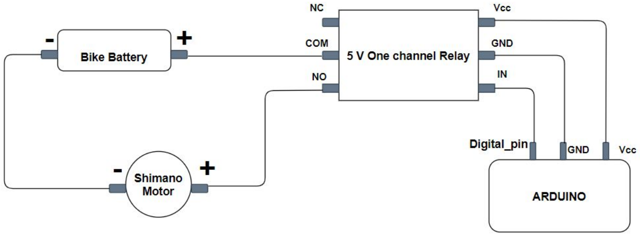
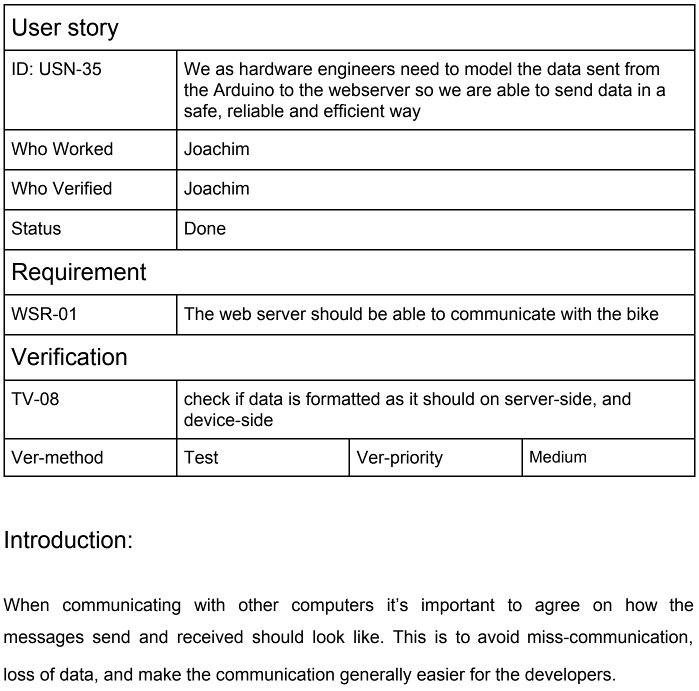


Figure-3 Connection between Battery, Arduino, Relay, and Motor

Literature:

1. https://si.shimano.com/pdfs/sm/SM-SHIMANO_STEPS_US-000.pdf
2. https://www.circuitbasics.com/setting-up-a-5v-relay-on-the-arduino/
3. https://lastminuteengineers.com/one-channel-relay-module-arduino-tutorial/

A.5.9 USN-35 Model data sent from Arduino to the web server



User story			
ID: USN-35	We as hardware engineers need to model the data sent from the Arduino to the webserver so we are able to send data in a safe, reliable and efficient way		
Who Worked	Joachim		
Who Verified	Joachim		
Status	Done		
Requirement			
WSR-01	The web server should be able to communicate with the bike		
Verification			
TV-08	check if data is formatted as it should on server-side, and device-side		
Ver-method	Test	Ver-priority	Medium

Introduction:

When communicating with other computers it's important to agree on how the messages send and received should look like. This is to avoid miss-communication, loss of data, and make the communication generally easier for the developers.

Method Used:

There are many ways we can format our data. The example data used in this user story is as followed:

Bike id = 5

longitude = 1.123456

Latitude = 2.234567

Battery level = 65%

The first way we can do this is to only send the data:

{5, 1.123456, 2.234567, 65}

This will save a lot of overhead, but it is harder to know what is what when receiving this data, which makes it harder to program on the server-side.

Another option is to send the type of data and data on the same line and shorten the text, so it’s still understandable and not so much overhead, this can be seen here:

id: 5

lon: 1.123456

lat: 2.234567

bat : 65%

This is very close to a JSON format, which is a widely used standard to communicate between computers. With JSON format the data will look like this:

```
[
        ‘id’: 5
        ‘lon’: 1.123456
        ‘lat’: 2.234567
        ‘bat’: 65%
]
```

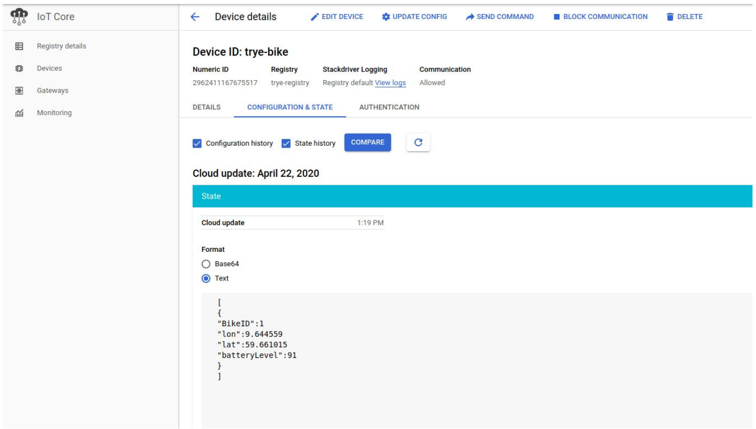
This will make it easier to find the data on the server-side, as there exists JSON parser for also any programing language that is used today. Image of the message on the server can be viewed in FIG-1

Conclusion:

We ended up formating the messages as JSON format, to make it easier to read on the server-side.

Figures:

FIG-1



Device ID: trye-bike

Numeric ID	Registry	Stackdriver Logging	Communication
2962411167675517	trye-registry	Registry default View logs	Allowed

☑ Configuration history ☑ State history

Cloud update: April 22, 2020

Cloud update 1:19 PM

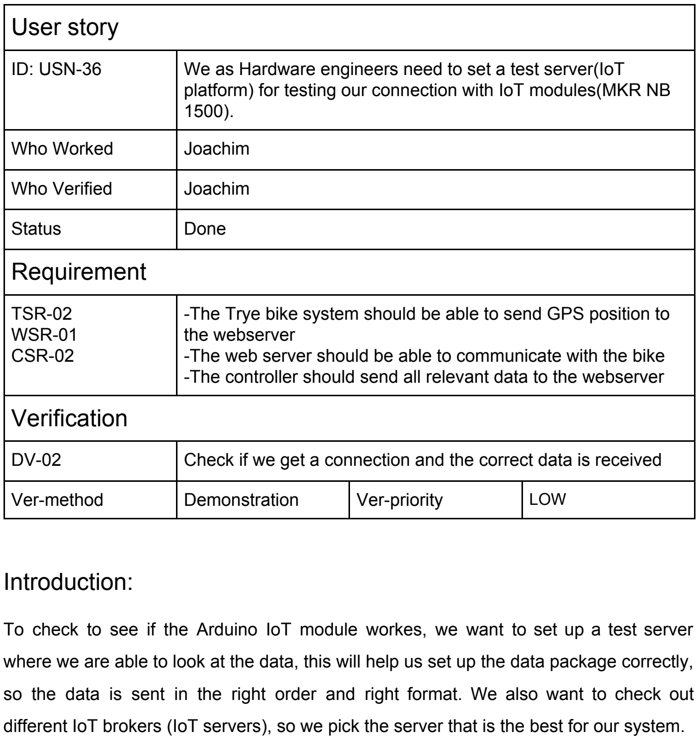
Format

○ Base64

● Text

```
[
{
"BikeID":1
"lon":9.644559
"lat":59.661015
"batteryLevel":91
}
]
```

A.5.10 USN-36 Finding a web server for communication



User story			
ID: USN-36	We as Hardware engineers need to set a test server(IoT platform) for testing our connection with IoT modules(MKR NB 1500).		
Who Worked	Joachim		
Who Verified	Joachim		
Status	Done		
Requirement			
TSR-02 WSR-01 CSR-02	-The Trye bike system should be able to send GPS position to the webserver -The web server should be able to communicate with the bike -The controller should send all relevant data to the webserver		
Verification			
DV-02	Check if we get a connection and the correct data is received		
Ver-method	Demonstration	Ver-priority	LOW

Introduction:

To check to see if the Arduino IoT module workes, we want to set up a test server where we are able to look at the data, this will help us set up the data package correctly, so the data is sent in the right order and right format. We also want to check out different IoT brokers (IoT servers), so we pick the server that is the best for our system.

Method Used:

Platform selection:

There are many IoT brokers on the market, some of which is built into Microsoft Azure, Amazon Web Server (AWS) and Google Cloud Platform (GCP), and has a wide selection of other services that work with them out of the box, and there are also some smaller providers which focus more only on the IoT communication and therefore don’t have the wide selection of services that the previously mentioned providers have. These smaller services can be easier to set up, as they don’t need to be as customizable as the other services, one of these platforms is Thingspeak. Thingspeak provides the option to process the data with Matlab and could be useful to find the speed of the bike using the GPS.

As our system needed to store the data systematically and send it securely to the mobile app, we chose to go with one of the bigger platforms. AWS, GCP, and Microsoft Azure usually give out a free sample period, but because of extra demand for servers when setting up this system, AWS chose to limit their free samples. The platform we decided to use was GCP as this provides an easy-to-use interface, has the services we need and the mobile app database can easily be set up in this environment, so we get all the data on the same platform. Microsoft Azure should also work.

Background:

Most IoT devices communicate with a protocol called MQTT, which is a lightweight and fast communication protocol. Because we decided to use MQTT, the services in GCP have to support it as well. Services that support MQTT usually also support devices to server communication using HTTP as well, although it requires more bandwidth. With support for HTTP and MQTT protocol, it is possible to use an IoT broker (IoT server) as a relay, going from the Arduino to the mobile app, unless we want to store the data in a more systematic way, like in a database.

Getting an encryption key from the Arduino:

GCP requires a known public key, to allow a device to connect to it, because of this we need to generate an encryption key (or token). The Arduino MKR NB 1500 has a build-in encryption chip which makes it easy to generate and use an encryption key. This is done using the Arduino IDE by first installing the “Arduino SAMD” board and “ArduinoECCX08” library, then upload the example code “ECCX08JWSPublicKey” to the Arduino which helps you generate and store the key. This key is required to set up communication between Arduino and GCP.

Setting up GCP (Google Cloud Platform):

GCP provides a number of different services, the one required to set up a communication between an IoT (Arduino) device and GCP is called *IoT Core*. But before we can set up an IoT Core, we first need to create a project, which we named “trye-bike-rental”. When a project is created we can set up an IoT core, here we need to set up a register and a device. A Register can be seen as a folder containing multiple devices. We named the register “trye-register” and the device “trye-bike” which will be changed to “trye-bikeID01” in the final production. Once this is set up, we can use an Arduino to update the state of the “trye-bike” and therefore see the data packet sent, which can be seen in Fig1. The modeling of the data packets can be read about in USN-35 and the programming of the Arduino to work with this example can be read about in USN-44

Contribution:

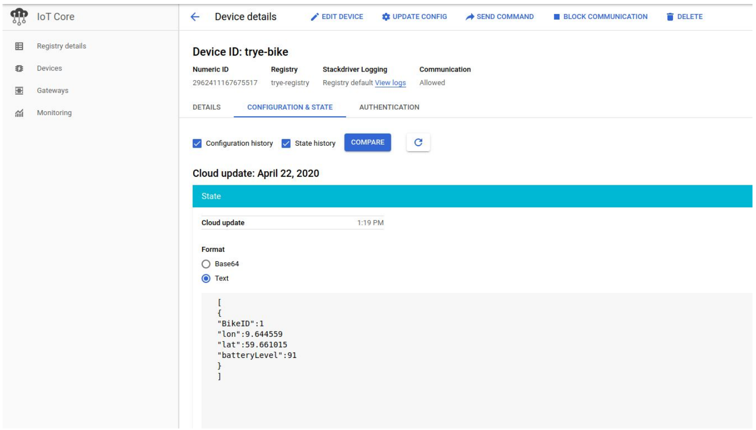
Eebbaa:

- Eebbaa has the Arduino, therefore he configured and sent the messages from the Arduino once GCP was ready.
- Helped to brainstorm and select which platform to use.

Conclusion:

It is now possible to communicate between the Arduino and the GCP server, but it is not possible to relay the data any further. This is because we store the data as a device state and not an event in GCP, this is easily fixable by using another GCP service called pub/sub (publish/subscribe), this can be read about in USN-51

Figures:



Device ID: trye-bike

Numeric ID	Registry	Stackdriver Logging	Communication
2962411167675517	trye-registry	Registry default View logs	Allowed

☑ Configuration history ☑ State history

Cloud update: April 22, 2020

Cloud update 1:19 PM

Format

○ Base64

● Text

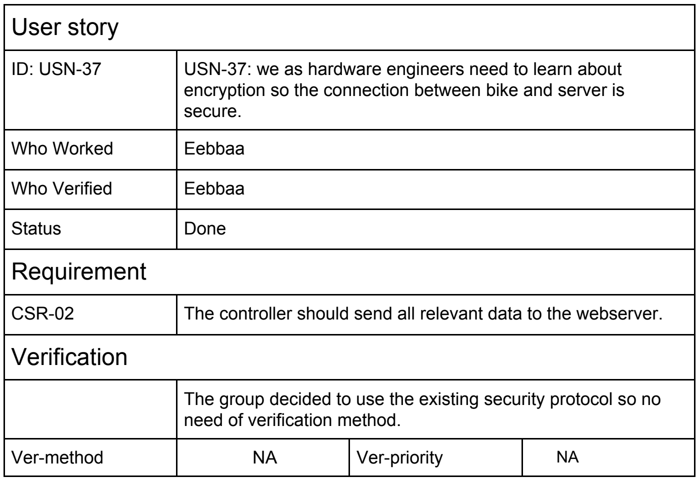
```
[
{
"BikeID":1
"lon":9.644559
"lat":59.661015
"batteryLevel":91
}
]
```

Fig1: Here we can see the device “trye-bike” with the state sent from the Arduino

Literature:

Appendix:

A.5.11 USN-37 Encryption method on our IoT device



User story			
ID: USN-37	USN-37: we as hardware engineers need to learn about encryption so the connection between bike and server is secure.		
Who Worked	Eebbaa		
Who Verified	Eebbaa		
Status	Done		
Requirement			
CSR-02	The controller should send all relevant data to the webserver.		
Verification			
	The group decided to use the existing security protocol so no need of verification method.		
Ver-method	NA	Ver-priority	NA

Introduction:

While building an IoT application, thinking about security is very important at all levels. The vulnerability at some point in your IoT infrastructure has a big risk of affecting the stakeholders that are related to your IoT application.

As a system developer understanding your IoT devices and Security technologies your IoT device supports will help to know the level of security you can provide. This may help to identify the possible threats that may happen due to your IoT device and it will be the first step in adding some security features to your system if necessary.

Here we will discuss how our IoT device manages security during data exchange with other components of our system.

Method Used:

IoT devices have inbuilt crypto chips for managing and storing keys. These crypto chips facilitate key-based authentication for communication and data exchange in IoT applications. As a group we identified the crypto chip that is embedded in our Hardware device and identified the features for the crypto chip and we understand the cryptographic operation of the chip.

The Arduino board we use for this project (MKR NB 1500) has an embedded microchip ATECC508A which is used as a crypto chip. This chip is used for generating and storing 256 bit ECC keys.

The ATECC508A implements a complete asymmetric (public/private) key cryptographic signature solution based upon Elliptic Curve Cryptography and the ECDSA signature protocol. The device is designed to securely store multiple private keys along with their associated public keys and certificates. The signature verification command can use any stored or an external ECC public key. Public keys stored within the device can be configured to require validation via a certificate chain to speed up subsequent device authentications. Random private key generation is supported internally within the device to ensure that the private key can never be known outside of the device. The public key corresponding to a stored private key is always returned when the key is generated and it may optionally be computed at a later time.[1]

Arduino has open-source libraries and codes that can be used for generating random key pairs(private key and public keys). The Arduino codes that are designed for crypto chips of ECC508/ECC608 are the following.

- ECCX08CSR:- helps to generate a CSR(Certificate Signing Request) for a private key generated in a crypto chip slot.

- ECCX08JWSPublickey:- helps to generate a public key for a private key generated in a crypto chip slot.

- ECCX08SelfSignedCert:- helps to generate a self-signed certificate for a private key generated in a crypto chip slot.

When we connect our IoT device to an IoT platform we need to generate keys(private and public keys) and store them in our crypto chip. Figure-1 shows where to store the generated keys. The private key will be used for generating digital signatures whenever we send data from our board and the public key will be used to authenticate the sender from its digital signature sent with the data.

Besides keys generated, we need to use JSON Web tokens for authenticating IoT devices to the cloud or to communicate with communication bridges such as MQTT. Figure-2 shows how IoT devices establish communication with MQTT bridges by using JSON Web Tokens.

JSON Web token is an open standard(RFC 7519) that is used to securely transmit information. The information contains a digitally signed signature of the sender and this makes the data to be trusted after verification is made at the receiver side.

JSON web token has a format of [axxxx . bxxxxx. cxxxxx]. The first is the header part which includes the information about the algorithm name and type of JWT used in base 64 URL encoded format. The second part is the payload which contains the claim of the user details or additional metadata in the base 64 URL encoded format. The third part is a digital signature, When we say digital signature it is analogous with handwritten signature on a post which is used to identify the sender.

In a digital signature, we need to generate a unique signature from the data to be sent and the private key by using cryptographic algorithms called ECC(Elliptical Curve Cryptography). This crypto algorithm will take the data in the JSON web token format for the first two parts(Header and payload) and the private key for generating unique digital signatures. The process of creating a Digital signature is shown at number 2 and 3 in Figure-3. Where the crypto chip takes the Hashed value of JSON web token for the header and payload and combines with the private key for generating ECDSA signature. This signature will attach to the JSON web token as the last part in the JSON Web token format. At this stage the data is ready to be sent to the IoT platform. The IoT platform at the receiver will verify the device by using the public key as shown at number 6 in Figure-3.

Conclusion:

Our IoT device uses the internet through the Telenor sim card and for using the internet the device uses its pin number and APN number for authenticating the sim card. After the IoT sim card is authenticated by Telenor, the IoT device can be added to the IoT platform using its Device_ID and public key generated on the device. IoT platforms are designed with security protocols in different layers of their platform. Once our device is added to the platform using the procedures given by the IoT platform company. Our device can use different protocols for sending and receiving data in secure communication channels. The data we send from our device are battery level, location information with longitude, and latitude. These data are not sensitive, as a group we aren't afraid of the third party can see the data. What we worry is that if an attacker can pretend as if it is one of the IoT devices and connects with the IoT platform. This has a dangerous consequence in our entire system because the IoT platform has a connection with Database and web server for the User app. If the hackers manage to connect with our Cloud platform they can manage a DoS(Denial of Service) attack and this can affect the TRYE AS company economically and they can lose users' trust. Losing trust from customers due to a lack of security can cause brand damage.

We are using the Google Cloud platform, which has good security features and they have many security protocols in different levels of their architecture. The google cloud platform has strong authentication of IoT devices during communication and sending data. The private key of our device is stored in the crypto chip and is isolated from people and software, this makes the communication more secure and difficult for hackers.

In Addition our user app has no direct connection with our IoT device. The device sends data to the platform and the webserver gets device data from the platform and the device gets configuration commands from the cloud based on the user's status for example paid users(authenticated user) will get the service through the cloud to start the motor of the bike.

Google has the following recommendation for keeping the IoT applications secure.

The following security recommendations are not enforced by Cloud IoT Core but will help you secure your devices and connections.

- Keep the private key secret.
- Use TLS 1.2 when communicating with `mqtt.googleapis.com` or `mqtt.2030.ltsapis.goog` on ports 8883 and 443. To maintain TLS connections:
  - Verify that the server certificate is valid using a Google root CA certificate.
  - Perform regular security-related firmware updates to keep server certificates up-to-date.
  - Read this security note for detailed TLS requirements and future compatibility.
- Each device should have a unique public/private key pair. If multiple devices share a single key and one of those devices is compromised, an attacker could impersonate all of the devices that have been configured with that one key.
- Keep the public key secure when registering it with Cloud IoT Core. If an attacker can tamper with the public key and trick the provisioner into swapping the public key and registering the wrong public key, the attacker will subsequently be able to authenticate on behalf of the device.

- The key pair used to authenticate the device to Cloud IoT Core should not be used for other purposes or protocols.
- Depending on the device's ability to store keys securely, key pairs should be rotated periodically. When practical, all keys should be discarded when the device is reset.
- If your device runs an operating system, make sure you have a way to securely update it. Android Things provides a service for secure updates. For devices that don't have an operating system, ensure that you can securely update the device's software if security vulnerabilities are discovered after deployment.
- Ensure that you have a way to update root certificates. For more details, see the Google Internet Authority site.
- Ensure that the clock on the device is not tampered with. If the device clock is compromised, a powerful attacker can trick the device into issuing tokens that will be valid in the future, circumventing the expiration time of the token. For best results, use the Google Public NTP Server.   [4]

Figures:

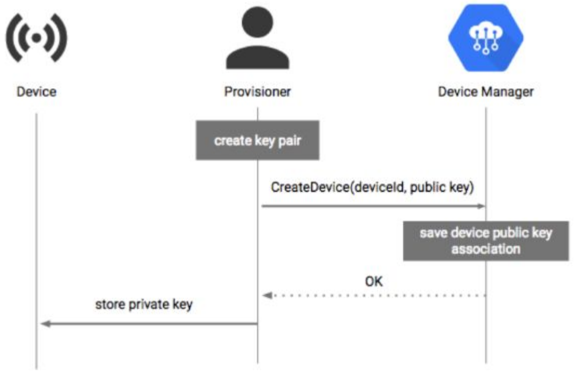

Device
Provisioner
Device Manager
create key pair
CreateDevice(deviceId, public key)
save device public key association
OK
store private key

Figure-1 creation and storage of private and public keys.

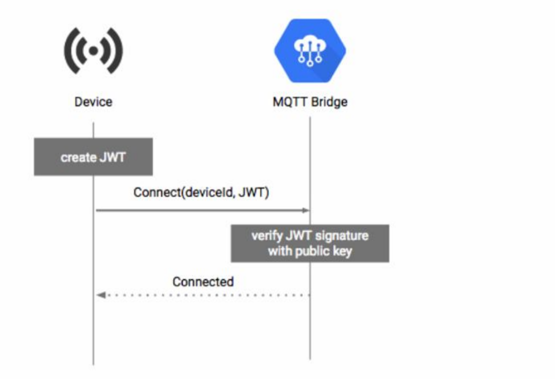


Figure-2 How IoT device establishes a connection with MQTT Bridge using JSON web token.

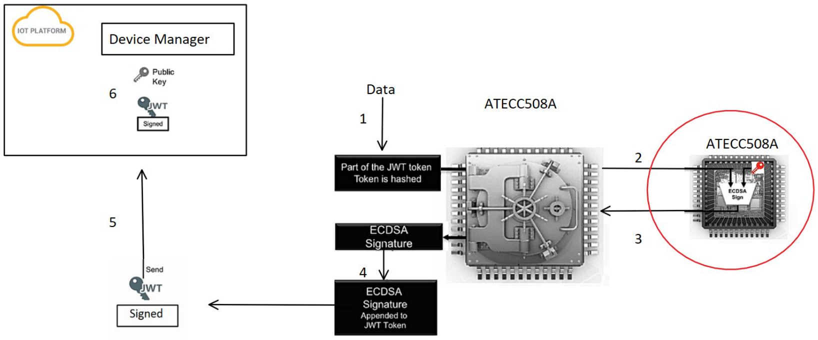


Figure-3 steps for sending data from IoT device with Digital signature to IoT platform

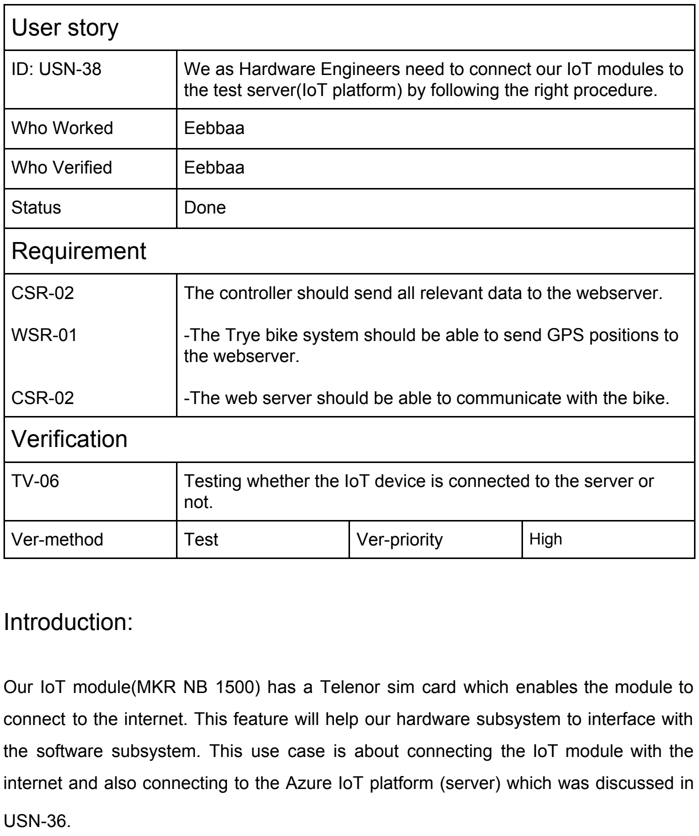
Literature:

1. http://ww1.microchip.com/downloads/en/DeviceDoc/20005928A.pdf
2. https://cloud.google.com/iot/docs/how-tos/credentials/jwts
3. https://www.youtube.com/watch?v=TmA2QWSLSPg&t=190s
4. https://cloud.google.com/iot/docs/concepts/device-security

Appendix:

ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
CSR	Certificate Signing Request
MQTT	Message Query Telemetry Transport

A.5.12 USN-38 Connecting our IoT Module to a IoT Platform



User story			
ID: USN-38	We as Hardware Engineers need to connect our IoT modules to the test server(IoT platform) by following the right procedure.		
Who Worked	Eebbaa		
Who Verified	Eebbaa		
Status	Done		
Requirement			
CSR-02 WSR-01 CSR-02	The controller should send all relevant data to the webserver. -The Trye bike system should be able to send GPS positions to the webserver. -The web server should be able to communicate with the bike.		
Verification			
TV-06	Testing whether the IoT device is connected to the server or not.		
Ver-method	Test	Ver-priority	High

Introduction:

Our IoT module(MKR NB 1500) has a Telenor sim card which enables the module to connect to the internet. This feature will help our hardware subsystem to interface with the software subsystem. This use case is about connecting the IoT module with the internet and also connecting to the Azure IoT platform (server) which was discussed in USN-36.

Method Used:

For connecting the hardware with the internet, the antenna should be connected to the MKR NB 1500 board and the 4G IoT sim card should be mounted to the board. An Arduino code is made to check if the board can connect to the network. The Arduino code(NB_scanner) will print out the IMEI number and scans for nearby networks and gives us the signal strength of the network between 0 and 31 where 0 is minimum signal strength while 31 is maximum and prints the result on the serial monitor, the result we got is shown in figure 1.

For connecting our board(MKR NB 1500 + 4G/LTE) to the network and internet through the 4G network we included a library called MKRNB.

Our board operates in 4G and it has a self embedded modem that enables the board to transfer data from serial port to the network. The library abstracts the low level of communication between the modem and the Simcard.

Structure of the MKRNB library:- the library comprises different classes with various functions that provide different functionalities.

1. NB class- helps the board modem to connect to the network by using the pin number of the sim card.
2. NB-SMS class- it enables the board modem to send/receive SMS messages.
3. GPRS class- it helps for connecting to the internet.
4. NBClient - includes an implementation for a client. This class can create a client that can connect to a server by using GPRS class functions to send and receive data.

After the connection with the network is succeeded, the IoT module should be added to the IoT platform(test server). To do that the IoT module needs to have an SHA-key. To get the SHA key an Arduino code(ECCX08SelfSignedCert) is loaded to the board for generating a self-signed certificate for a private key to the crypto chip and the result is shown in Figure-2. This key will be used for securing communication between the module and the IoT platform. The security feature of the crypto chip is discussed on the USN-37.

After adding our IoT board(MKR NB 1500 + 4G/LTE) to the Azure IoT hub. We can connect our device to the Azure IoT hub by using the MQTT protocol.

MQTT is a Client-Server publish/subscribe messaging transport protocol. It is lightweight, open, simple, and designed to be easy to implement. These characteristics make it ideal for use in many situations, including constrained environments such as for communication in Machine to Machine (M2M) and Internet of Things (IoT) contexts where a small code footprint is required and/or network bandwidth is at a premium. The protocol runs over TCP/IP, or over other network protocols that provide ordered, lossless, bidirectional connections.[4]

Contribution:

Joachim:

- Helped find the code for connection to the GCP platform
- Provided necessary certificate to connect to GCP

Conclusion:

Connecting the MKR NB 1500 board to the internet is accomplished and the board connects to the Telenor network through IoT Telenor 4G sim card. Connecting the board to the IoT hub (test server) is also accomplished and checked by sending a Hello message to the server and the server received the message and the message was seen in the log file of the server.

Arduino-code:

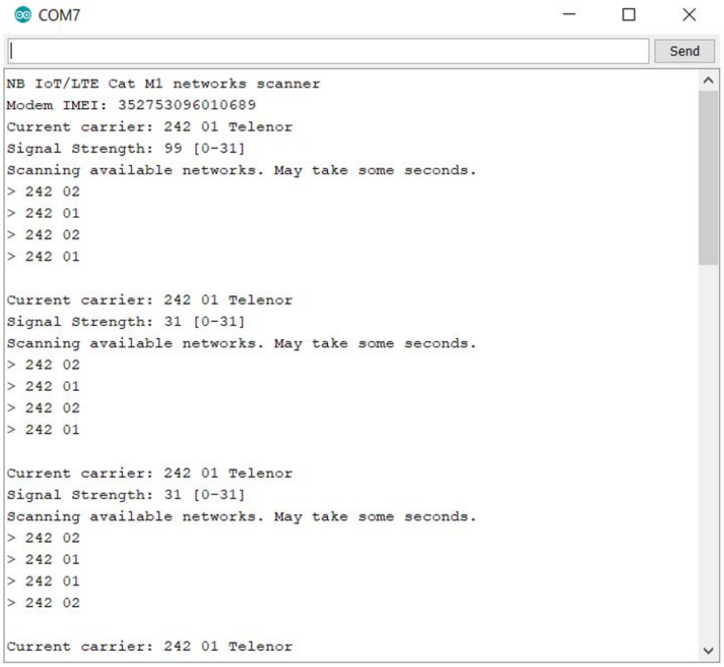
1. NB_scanner
   https://drive.google.com/drive/folders/1Qd-Z840bCUuJrs0ObDkwxYfx0r_iEniy
2. ECCX08SelfSignedCert

https://drive.google.com/drive/folders/1_h3lOEOZSPKfhApzBck_eY9DTggwe21t

3. Azure_IoT_Hub_NB

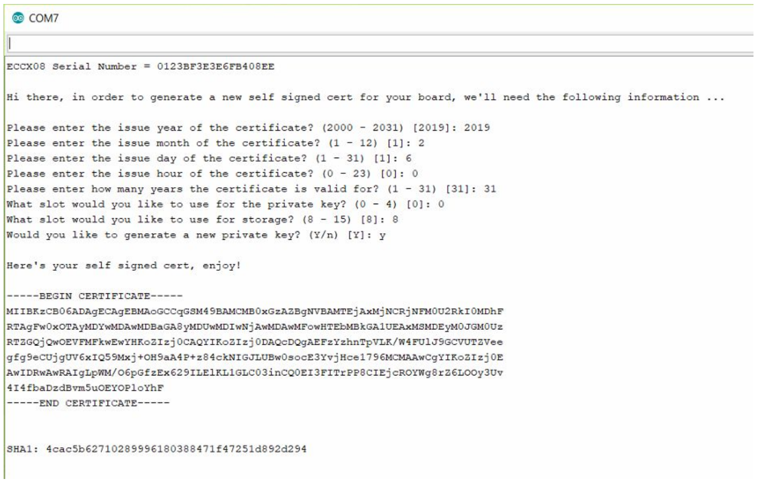
   https://drive.google.com/drive/folders/1GgG-_XZtXSTbwa60ttNaXqPwTh9BgxPX

Figures:



```
NB IoT/LTE Cat M1 networks scanner
Modem IMEI: 352753096010689
Current carrier: 242 01 Telenor
Signal Strength: 99 [0-31]
Scanning available networks. May take some seconds.
> 242 02
> 242 01
> 242 02
> 242 01

Current carrier: 242 01 Telenor
Signal Strength: 31 [0-31]
Scanning available networks. May take some seconds.
> 242 02
> 242 01
> 242 02
> 242 01

Current carrier: 242 01 Telenor
Signal Strength: 31 [0-31]
Scanning available networks. May take some seconds.
> 242 02
> 242 01
> 242 01
> 242 02

Current carrier: 242 01 Telenor
```

Figure-1: Serial Monitor result by running Arduino code NB_scanner



```
ECCX08 Serial Number = 0123BF3E3E6FB408EE

Hi there, in order to generate a new self signed cert for your board, we'll need the following information ...

Please enter the issue year of the certificate? (2000 - 2031) [2019]: 2019
Please enter the issue month of the certificate? (1 - 12) [1]: 2
Please enter the issue day of the certificate? (1 - 31) [1]: 6
Please enter the issue hour of the certificate? (0 - 23) [0]: 0
Please enter how many years the certificate is valid for? (1 - 31) [31]: 31
What slot would you like to use for the private key? (0 - 4) [0]: 0
What slot would you like to use for storage? (8 - 15) [8]: 8
Would you like to generate a new private key? (Y/n) [Y]: y

Here's your self signed cert, enjoy!

-----BEGIN CERTIFICATE-----
MIIBKzCB06ADAgECAgEBMAoGCCqGSM49BAMCMB0xGzAZBgNVBAMTEjAxMjNCRjNFM0U2RkI0MDhF
RTAgFw0xOTAyMDYwMDAwMDBaGA8yMDUwMDIwNjAwMDAwMFowHTEbMBkGA1UEAxMSMDEyM0JGM0Uz
RTZGQjQwOEVFMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEFzYzhnTpVLK/W4FUlJ9GCVUTZVee
gfg9eCUjgUV6xIQ59Mxj+OH9aA4P+z84ckNIGJLUBw0socE3YvjHce1796MCMAAwCgYIKoZIzj0E
AwIDRwAwRAIgLpWM/O6pGfzEx629ILElKL1GLC03inCQ0EI3FITrPP8CIEjcROYWg8rZ6LOOy3Uv
4I4fbaDzdBvm5uOEYOPloYhF
-----END CERTIFICATE-----


SHA1: 4cac5b62710289996180388471f47251d892d294
```

Figure-2: Serial monitor result while running ECCX08SelfSignedCert code.

Literature:

1. https://www.arduino.cc/en/Tutorial
2. https://create.arduino.cc/projecthub
3. https://create.arduino.cc/projecthub/Arduino_Genuino/securely-connecting-an-arduino-nb-1500-to-azure-iot-hub-af6470?ref=part&ref_id=64346&offset=0&fbclid=IwAR1j9mooSXGVDGkK-80zGilNxtgTyha9OKRH4jycS3gTgyJuH_i0ZvCbXm0

4. "MQTT Version 5.0 OASIS Standard Specification" pdf-file
   Link = https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.pdf
5. https://platformio.org/lib/show/5570/ArduinoECCX08/examples?file=ECCX08CSR.ino

Appendix:

- SHA ………...………………...Secure Hash Algorithm.
- IMEI number ……….. International Mobile Station Equipment Identity.
- GPRS………………….General Packet Radio Service.
- MQTT protocol………. Message Query Telemetry Transport protocol.

A.5.13 USN-44 Sending bike data to the server

User story			
ID: USN-44	We as Hardware Engineers need to send our sensor data from the Arduino module to the server based on the Modelled data.		
Who Worked	Eebbaa		
Who Verified	Eebbaa		
Status	Done		
Requirement			
CSR-02 WSR-01 CSR-02	- The controller should send all relevant data to the webserver. -The Trye bike system should be able to send GPS positions to the webserver. -The web server should be able to communicate with the bike.		
Verification			
TV-13	Testing if the device sends the GPS location data and battery level in the format we need.		
Ver-method	Test	Ver-priority	High

Introduction:

Sending speed data, battery level, and location data from the bike to the server is very important for our project. Since the group finally decided to use the Google cloud platform as our server-side, the data should be sent to this platform from our IoT board.

Method Used:

Different tasks and configuration is made on the google cloud platform and on the board, and those tasks are:-

1. Configuring and adding the board to a specific project on the platform.
2. Choosing an appropriate protocol for communication between the board and the platform. Where we have the possibility to choose MQTT or HTTP.
3. Generating a public key on the board by running an Arduino code (ECCX08JWSPublicKey).
4. The public key is stored both on the board and on the cloud platform. The key helps for securing communication between the board and the platform and it should be kept secret. The public key generated on the board for this user history is shown in Figure.1.

An Arduino code is loaded with the MQTT library. This library enables the module to publish an event message or state message and subscribe to different sub/pub topics to receive messages on those topics. The message will be published to the MQTT server and any IoT client that subscribed for this topic will receive the message from the MQTT server.

The MQTT server for the Google cloud platform is "mqtt.googleapis.com" and the data that will be sent from the bike IoT board will be stored on this broker at Sub/pub and any client that is subscribed for this message will receive the data automatically after the message is published. In our case the bike data will be published for the cloud storage to access it and the cloud storage should subscribe to this respective topic.

Figure-2 shows the state message and configuration message exchange through sub/pub between the bike and other GCP components. The black line on the figure shows the message that will be published to the pub/sub and it has all the bike information such as Bike-Id, bike location information with longitude and latitude, and battery level. Cloud storage is subscribed to this message and it can access the data. The pub/sub application helps to exchange a message between the IoT device and the server and also among IoT devices. After the message is sent from the bike to the cloud storage, the data will be handled further accordingly and it is discussed in the USN-51.

The Arduino code (GCP_IoT_core_NB) is loaded on the MKR NB 1500 board for sending the data to the server. The code will help the board to

1. Connect to the network with a pin number and APN number.
2. Make an MQTT connection between the board and the MQTT server of the GCP which is "mqtt.googleapis.com". In this step the IoT board will subscribe to Sub/pub configuration command messages and publish to Sub/Pub event messages. The Arduino code has two important functions.

    - *void publishMessage(**int** bikeID, **String** lon, **String** lat, **int** batteryLevel):* this function will send the message which contains the bike-ID, longitude, latitude, and battery level of the bike. The message part will be put into the MQTT packet as payload together with other packet parameters. The message sent to the MQTT server has a JSON format and can be viewed on device configuration and state on the IoT core and can be viewed both in text format or base-64 format.

    - *void onMessageReceived(int messageSize):* this function will retrieve the message sent to the device as command and displays to the serial monitor. This function will be modified and can be used to receive a command that can change the configuration of the Arduino, for example, we can send a logical value ‘1’ or ‘0’ to a certain pin of the Arduino board, and this can be used for controlling bike motor by changing the digital pin value the motor is connected. It is discussed in USN-58.

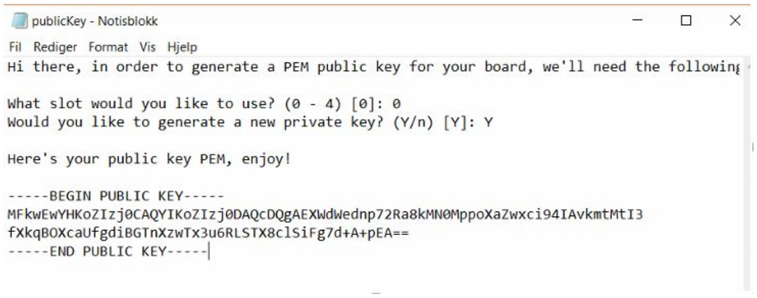
Conclusion:

Sending the data from the bike to The GCP is accomplished and the message we sent can be viewed by other IoT devices that are subscribed to the message. If we have many bikes, all the bikes can send their data to one Sub/pub topic, and with their (specific ID + location_data + Battery_level) and cloud storage can subscribe to this topic and the cloud storage can receive data sent by all bikes. Having all data from the bikes in one storage makes things easy for extracting and using the data we need for further storage in databases or displaying in real-time on a user app.

Arduino-code:

1. GCP_IoT_Core_NB
   https://drive.google.com/drive/folders/1kHnc5LB7wuhquW9LOMd-XPQ7TeYaA3Um
2. ECCX08JWSPublicKey
   https://drive.google.com/drive/folders/1qlNN14RljvTH59bf0vVGIPj7qpD2qRhB

Figures:



```
Hi there, in order to generate a PEM public key for your board, we'll need the following

What slot would you like to use? (0 - 4) [0]: 0
Would you like to generate a new private key? (Y/n) [Y]: Y

Here's your public key PEM, enjoy!

-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEXWdWednp72Ra8kMN0MppoXaZwxci94IAvkmtMtI3
fXkqBOXcaUfgdiBGTnXzwTx3u6RLSTX8clSiFg7d+A+pEA==
-----END PUBLIC KEY-----
```

Figure 1: public key generated for securing communication.

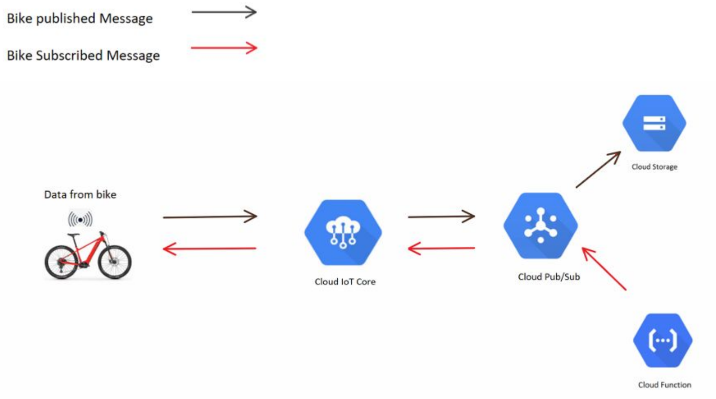


Figure 2:Pub/Sub data exchange illustration between Bike and other GCP components.

Literature:

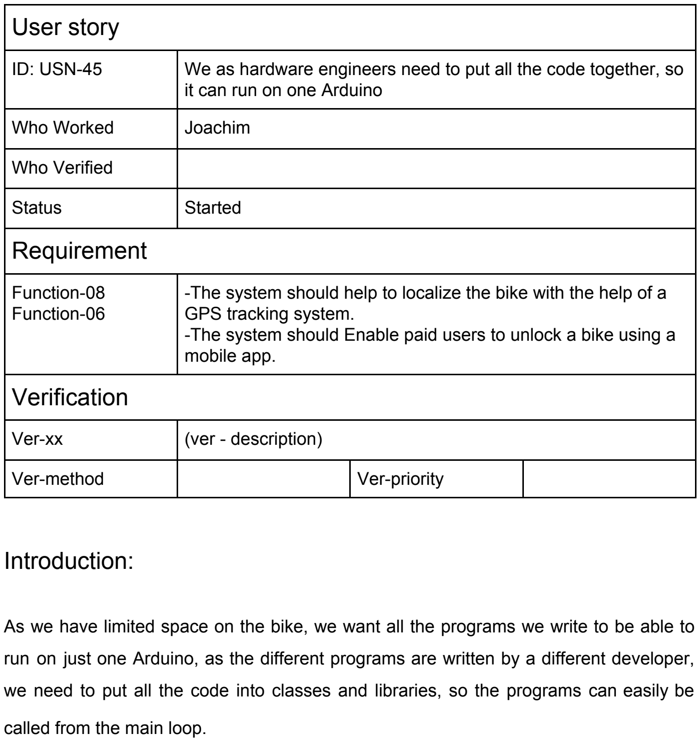
1. https://create.arduino.cc/projecthub/Arduino_Genuino/securely-connecting-a-mkr-gsm-1400-to-google-cloud-iot-core-b8b628?fbclid=IwAR2dHUGnYhRPtCzITWDNDAA-m66k6sm-HV018Z-dtlLd9rrYP0Xo7oul0dg

2. https://create.arduino.cc/projecthub

Appendix:

GCP --Google Cloud Platform

A.5.14 USN-45 Combining all our hardware code



User story			
ID: USN-45	We as hardware engineers need to put all the code together, so it can run on one Arduino		
Who Worked	Joachim		
Who Verified			
Status	Started		
Requirement			
Function-08 Function-06	-The system should help to localize the bike with the help of a GPS tracking system. -The system should Enable paid users to unlock a bike using a mobile app.		
Verification			
Ver-xx	(ver - description)		
Ver-method		Ver-priority	

Introduction:

As we have limited space on the bike, we want all the programs we write to be able to run on just one Arduino, as the different programs are written by a different developer, we need to put all the code into classes and libraries, so the programs can easily be called from the main loop.

Method Used:

To put the code together, it’s important the code comes in the right order. we can divide the program into two parts; gather data and send data, where the gathering of the data happens before the sending of the data. In what order the data is gathered is not important, but the flow chart in Fig1 was followed.

As each program has global variables it used, the program was enclosed in a class, where we can make the global variables local for the class, and therefore there will be no variables that overlap when stitching together the program.

To make the main program easier to read, the individual programs will be put inside a library, so it becomes hidden in the main program, and therefore less to look at

Conclusion:

The get speed and battery functions are already implemented, but because of lack of time, we were not able to implement the get GPS and send data function, as they are not complete yet. But all the groundwork is ready, so once the sub-programs are ready, it should be easy to implement.

Figures:

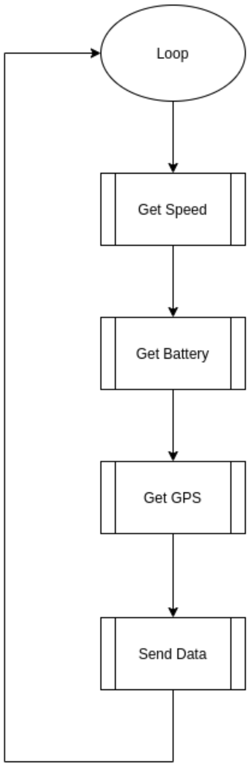
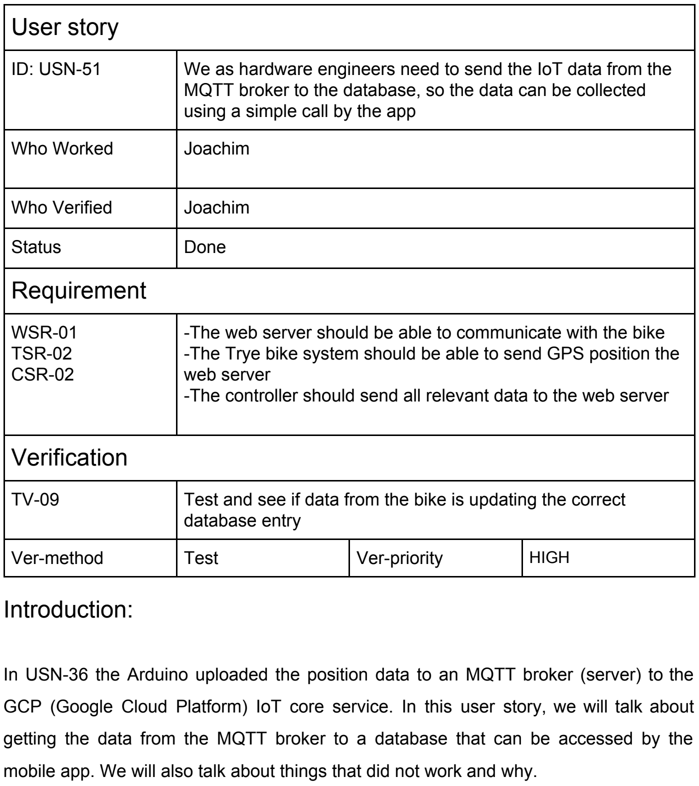


Fig1: shows the order of the execution of the program

A.5.15 USN-51 Sending data to database



User story			
ID: USN-51	We as hardware engineers need to send the IoT data from the MQTT broker to the database, so the data can be collected using a simple call by the app		
Who Worked	Joachim		
Who Verified	Joachim		
Status	Done		
Requirement			
WSR-01 TSR-02 CSR-02	-The web server should be able to communicate with the bike -The Trye bike system should be able to send GPS position the web server -The controller should send all relevant data to the web server		
Verification			
TV-09	Test and see if data from the bike is updating the correct database entry		
Ver-method	Test	Ver-priority	HIGH

Introduction:

In USN-36 the Arduino uploaded the position data to an MQTT broker (server) to the GCP (Google Cloud Platform) IoT core service. In this user story, we will talk about getting the data from the MQTT broker to a database that can be accessed by the mobile app. We will also talk about things that did not work and why.

Method Used:

GCP provides a number of different database storage solution and ways to access them, during the development of the data transfer and storage solution, we tested out most of the different data storage services GCP provides, this includes SQL, BigQuery, and Firestore, where Firestore was the one that worked best for our system.

The problem with the SQL service:

The first service we tested was the SQL database service. To use the SQL service, we require a way to upload the data from the pub/sub (publish/subscribe) service to the SQL service. to do this, it seems the best solution would be the cloud function service, which is a way to program/script what should happen in the cloud with python/node.js or GO. The cloud function service is able to set a trigger in the pub/sub service, which means the program/script will run whenever an event is happening in the pub/sub service. Therefore the function service will receive a data event from the pub/sub, containing the longitude and latitude of the bike, and store it in the SQL database. The problem using SQL in combination with cloud function to upload the data is, it only recently became supported to communicate between them, and therefore lack the required documentation to be able to upload the data.

The problem with BigQuery service:

BigQuery is a NoSQL database in which you are able to run normal SQL commands, because of this, it is able to use another type of google service called dataflow to upload the data from the pub/sub service to BigQuery service. The way this works is the data is uploaded to pub/sub in a JSON format as seen in FIG1, then with the dataflow service you are able to select what information you want to upload from the JSON data to BigQuery with a simple SQL call, (eg: “SELECT bike_id, lat, lon FROM ‘pubSub.tryeBike.json’” and there will automatically be made a table for the information gotten from the JSON data uploaded to the pub/sub service. With this approach we have two problems. The first is the data collected from the pub/sub service is appended to the end of a SQL database and not overwritten existing data. This is a problem because of storage size. During one year each bike will upload 150MB of data which is cumulative, and we will, therefore, waste a lot of storage space. This could be solved by manually deleting the data eg. once a year. The other problem is the cost of these services, after only 3 days of running the price was already at 300 NOK, which gives approximately 3000 NOK/MND. This means with the small dataset we are working on, this is not a viable solution until the TRYE company acquires a lot more bikes.

Solving the problem with Firestore

Firestore is a sub-service of the firebase service which is usually used for mobile developers to use Google's services, like google translate, image recognition, data processing, AI, and so on. Firestore is a NoSQL database usually used for lightweight data storage which can be used to store e.g. user data on mobile devices. This is perfect for us, as the data stored is overwritten when new data comes in, which BigQuery did not. This means we store the data only once per bike, instead of having multiple entries for each bike at different times. The only downside with this is it's more difficult to implement a historical view of the bike's location if that is a feature that's desired in the future.

The way Firestore stores data is in collection and document, where a collection can store multiple documents. To get a deeper tree with sub-collection, you need to start with a collection containing a document, where the document links to one or more collections which contains one or more documents again. In our system, we only need one collection with multiple documents, one for each bike. The hierarchy will then look like:

```
“trye-bike”
        |- “bike-01”
        |- “bike-02”
```

|- “bike-03”
|- “bike-04”
|- ...

Here each bike stores longitude, latitude, bike id, and battery level. Because it’s NoSQL we can easily add more data on the go without changing the structure of the database, or even add special data to only some of the bikes. This can be useful if we want to e.g. add a video stream to bikes with a camera later on and need to store something in the database to get access to the stream.

To send data from the pub/sub service to firebase, we can use the cloud function service. The reason we can use this with firebase and not SQL as previously described, is cloud function has better documentation to connect with firebase then with SQL, as this is something it could do for a longer time. The way cloud function works in this configuration is when an event happens in pub/sub the cloud function triggers a function that was programmed/scripted using python. The python script finds the bike_id, longitude, latitude, and battery level from a JSON file received from the pub/sub event, and using a JSON parser we get the data in an array. When the data is in a simple array, the data can be uploaded to a Firestore database using the python script found in APPX-01.

It’s also possible to use other programming languages for the cloud function service such as GO and Node.js.

Getting the data from Firestore:

The Firestore has a REST API, that can be used with Planet9. To use them we first need to set up a service account in GCP to have access to it outside of the GCP environment. When a service account is configured you get the option to generate a private json OAuth2 key, that can retrieve a temporary token that can be used to access the database. The credentials needed to read the Firestore database is found under “datastore->cloud datastore viewer”

Conclusion:

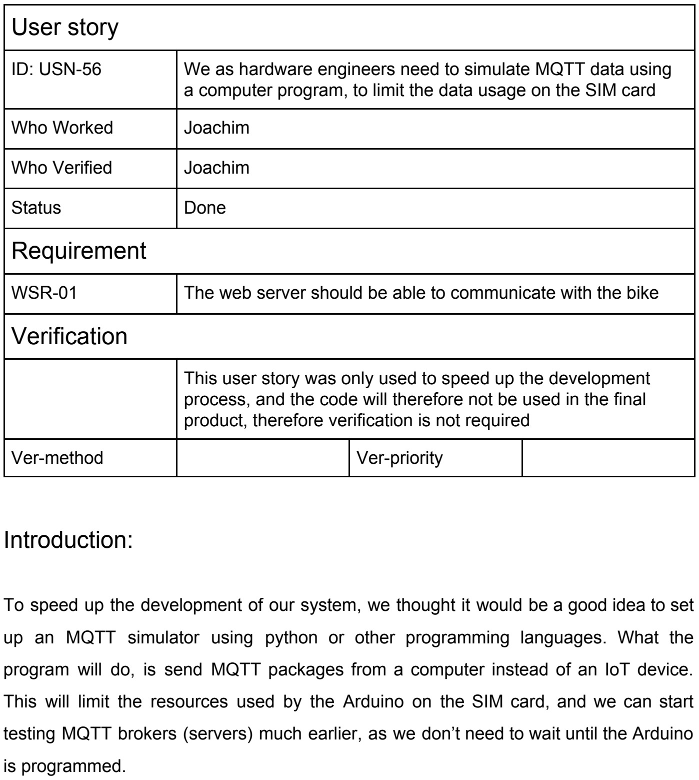
We are now using the Firestore database, which can be accessed with a REST API call with the credentials from a service account. During this user story, we have also looked at different services that ended up not working, like the SQL and BigQuery database.

Appendix:

Code for uploading JSON data from pub/sub to the Firestore database
APPX-01:

```
import base64
import json
import firebase_admin
from firebase_admin import credentials
from firebase_admin import firestore

debug = False

if(debug):
    print("Initilizing global variables")
project_id = "trye-bike-rental"
collection = "trye-bike"
document = "bike-" #add the bikeID to the end before sending

#credentials and establishing connection to the firebase database
cred = credentials.ApplicationDefault()
firebase_admin.initialize_app(cred, {
'projectId': project_id,
})

#the database instance
db = firestore.client()

def WriteToDatabase(bikeID, lon, lat, bat):
    #writing message to the collection and document
    doc_ref = db.collection(collection).document(document+str(bikeID))
    doc_ref.set({
        u'bikeid': bikeID,
        u'lon': lon,
        u'lat': lat,
        u'bat': bat,
    })
    #print debug message
    if(debug):
        print("to collection {} Wrote bikeid:{}, lon:{}, lat:{}, bat:{}
to the database".format(collection, bikeID, lon, lat, bat))


def main(event, context):
    """Triggered from a message on a Cloud Pub/Sub topic.
    Args:
        event (dict): Event payload.
        context (google.cloud.functions.Context): Metadata for the event.
    """
    pubsub_message = base64.b64decode(event['data']).decode('utf-8')


    eventDict = json.loads(pubsub_message)

    if(debug):
        print("event data: {}".format(eventDict))

    if(debug):
        print("debug: writing to database")
    WriteToDatabase(eventDict["bike_id"], eventDict["lon"],
eventDict["lat"], eventDict["bat"])
```

A.5.16 USN-56 Simulate MQTT data to limit data usage



User story			
ID: USN-56	We as hardware engineers need to simulate MQTT data using a computer program, to limit the data usage on the SIM card		
Who Worked	Joachim		
Who Verified	Joachim		
Status	Done		
Requirement			
WSR-01	The web server should be able to communicate with the bike		
Verification			
	This user story was only used to speed up the development process, and the code will therefore not be used in the final product, therefore verification is not required		
Ver-method		Ver-priority	

Introduction:

To speed up the development of our system, we thought it would be a good idea to set up an MQTT simulator using python or other programming languages. What the program will do, is send MQTT packages from a computer instead of an IoT device. This will limit the resources used by the Arduino on the SIM card, and we can start testing MQTT brokers (servers) much earlier, as we don’t need to wait until the Arduino is programmed.

Method Used:

Both Microsoft Azure, and Google Cloud Platform (GCP) has already programmed script which communicates with their services using MQTT. The only thing that has to be done is configured their MQTT services on the platform and then find the endpoint in which the scripts connect to.

Microsoft Azure

To set up the Microsoft Azure IoT service is simple, we first need to create an IoT hub and add a device to that hub. When that is done you can create a shared access key. With the shared access key and the IoT hub name we can make the hostname with the following format:

```
HostName={YourIoTHubName}.azure-devices.net;DeviceId=MyPythonDev
ice;SharedAccessKey={YourSharedAccessKey}
```

This will later be used to access the server from the MQTT python simulator.

To use the python programs we first need to download the library *azure-iot-device* using pip install, when that is done we can download *SimulateDevice.py* and configure it by pasting in the HostName into the file as the connection string. This program will send random temperature and humidity data to the server using MQTT by default. In the Microsoft Azure console, we can see the incoming temperature and humidity data sent from the python script. This can easily be modified to send the data we want to, but we decided to use GCP instead, so we did not modify the program any further. A more in-depth tutorial can be read here [1]

Google Cloud Platform

To connect a simulated MQTT device to GPC, we first need to configure GCP. In GCP there is a service called IoT core. Here we can create a register, which can contain multiple devices, for the simulation we only need one device. Whene a regiseter an a device is made, we can make a hostname with the following format:

```
Hostname=projects/PROJECT_ID/locations/REGION/registries/REGISTRY_ID/devices/DEVICE_ID
```

When GCP is configured we can configure google example python script to send to our server using the hostname, the same as with Microsoft azure. With GCP you can choose where the messages will go to, by ether publishing the messages to:

```
/devices/DEVICE_ID/event
```

Which is what we are going to use in final production, as this sends the message to the service pub/sub. Or we can send it to:

```
/devices/DEVICE_ID/state
```

Which is easier to debug with as it is not dependent on other services google provides, and we can see the messages directly in the IoT service. The full tutorial about configuring an MQTT simulator using GCP can be read [2]

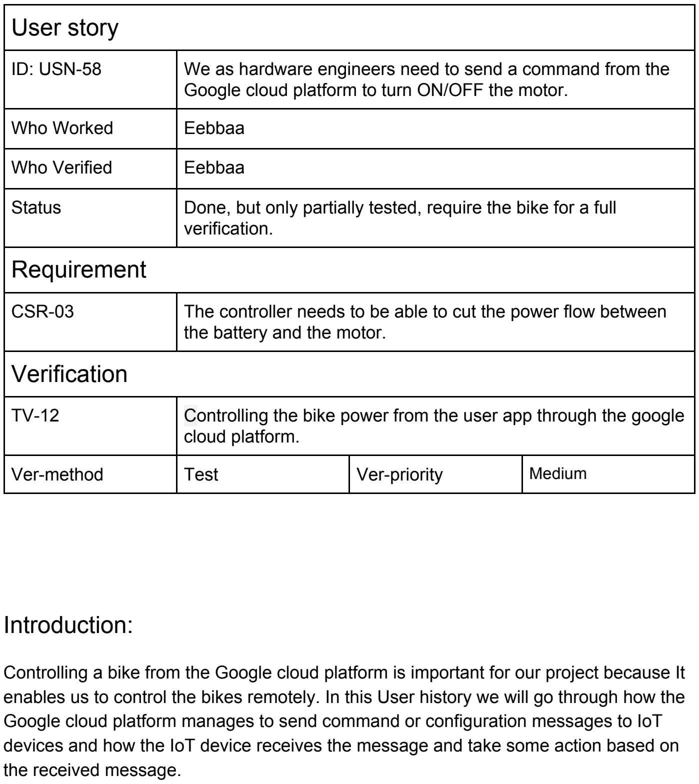
Conclusion:

We now have an MQTT data simulator for both Microsoft Azure, and GCP which helped us understand how the MQTT data message protocol workes. This helped us to set up the GCP platform which we are going to use later on in the development with the Arduino

Literature:

[1] https://docs.microsoft.com/en-us/azure/iot-hub/quickstart-send-telemetry-python
[2] https://cloud.google.com/iot/docs/how-tos/mqtt-bridge#iot-core-mqtt-auth-run-python

A.5.17 USN-58 Finding a way to stop the motor using a microcontroller



User story			
ID: USN-58	We as hardware engineers need to send a command from the Google cloud platform to turn ON/OFF the motor.		
Who Worked	Eebbaa		
Who Verified	Eebbaa		
Status	Done, but only partially tested, require the bike for a full verification.		
Requirement			
CSR-03	The controller needs to be able to cut the power flow between the battery and the motor.		
Verification			
TV-12	Controlling the bike power from the user app through the google cloud platform.		
Ver-method	Test	Ver-priority	Medium

Introduction:

Controlling a bike from the Google cloud platform is important for our project because It enables us to control the bikes remotely. In this User history we will go through how the Google cloud platform manages to send command or configuration messages to IoT devices and how the IoT device receives the message and take some action based on the received message.

Method Used:

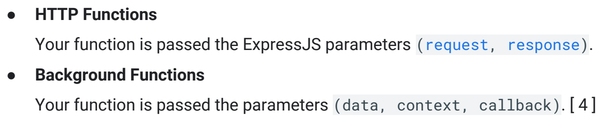
From Google cloud there are 2 types of messages that can be sent to IoT devices registered on the device manager. The cloud platform sends the messages with the help of IoT core Admin SDK or Cloud REST API. The message types are Configuration and command message.

Configuration message:- this is a message sent by the cloud platform through the cloud IoT core for configuring a device. It is user-defined data. The data can be structured and also can be formatted in any format such as arbitrary binary data, text, JSON, or serialized protocol buffers. Configuration messages can be used to update firmware, reboot a device, turn on a feature, or change other properties. Devices using MQTT can subscribe to a special MQTT topic: */device/{device-id}/config* for configuration updates. Devices can receive the latest configuration in a message payload. For example a configuration message can be { "bike-id": "001", “power” : "ON" } or { "bike-id" : "001", “power” : "OFF" }. To verify the configuration message is correctly applied and the devices are in the correct state, each device can report its state whether it is ON or OFF.

Command message:- are commands that can be sent to the IoT devices and they are one-time directive sent to those devices registered to the command topic: /devices/{device-Id}/commands/#". Command messages are much faster than configuration messages and can be sent more frequently. If the device is not connected to the cloud when the command message is sent, the command message will be lost.

There are different ways of sending these messages to the IoT devices. For our project, we will use Google cloud function to send the messages to our IoT device(MKR NB 1500).

Google cloud functions are serverless computing platforms that can be used to execute code in the cloud. Related to cloud functions, we have two concepts called *Events* and *Triggers.* When a change in the state of something happens the Google infrastructure will raise an *event and triggers* will be used to connect raised events with cloud functions. As shown in Figure-1 functions can be invoked either by an HTTP request or indirect event triggers.



- **HTTP Functions**
  Your function is passed the ExpressJS parameters `(request, response)`.
- **Background Functions**
  Your function is passed the parameters `(data, context, callback)`.[4]

In our project we have different events that may occur when users use the TRYE app for renting or booking a bike. For renting a bike a user should register on a user app and fill the necessary fields before sending it to the server. The result of user authentication on the server-side may cause an event where either a user is allowed to rent a bike or not. The events will trigger their own google cloud function which is responsible to run a JSON code for sending a configuration message to a bike for turning ON or OFF the motor.

Since our project has different events that may need to send command or configuration messages to a bike. We need to use a function that triggers based on an event and such function is called background function.

We use background functions when we want to have our Cloud Function invoked indirectly in response to an event, such as a message on a Pub/Sub topic, a change in a Cloud Storage bucket, or a Firebase event. [5]

In this project, Google cloud functions are created and given an appropriate name such as *turn_on_motor* and *turn_off_motor*. The function trigger method will be set to pub/sub and an event that publishes a message to pub/sub triggers its corresponding cloud function. Once a cloud function has triggered it can publish a message by using HTTP publish function. For example, if turn_on_power function is triggered based on a message published on pub/sub topic called user_authenticated as a result the function can publish a message {"power": "ON", "Bike-1"} on a specific topic related to it and the message can be sent to specific bike "Bike-1" if the bike is subscribed for the message on the topic. The IoT board(MKR NB 1500) will be able to get the message. The message can be read and based on the message we can manage to write an Arduino code that can change states on its digital pins, for example, digitalWrite(pin, 0) if the message is "OFF" or digitalWrite(pin, 1) if the message is "ON" considered as a command and the Arduino will take an action based on the message.

Conclusion:

Sending a Configuration message to an Arduino will help us to control devices connected to the digital pins of an Arduino. This approach will help us for example to control the power of a bike which is explained in USN-29, where Arduino sends either '0' or '1' for switching ON and OFF a relay which can turn on or off a bike motor.

Arduino-code:

https://drive.google.com/drive/folders/1I-CeTli1naAQrlLD3CqNlskFulXEoFnk

Figures:

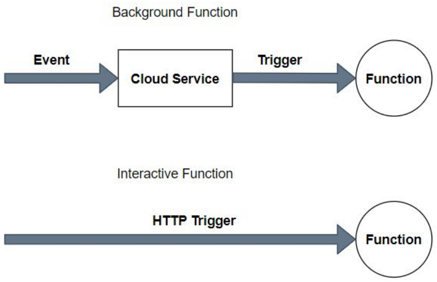


Figure-1: Categories of Cloud Functions based on how they invoked.

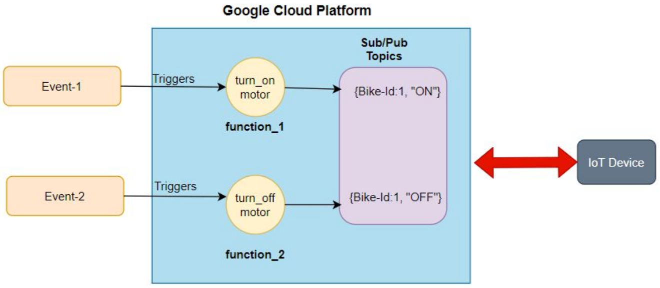
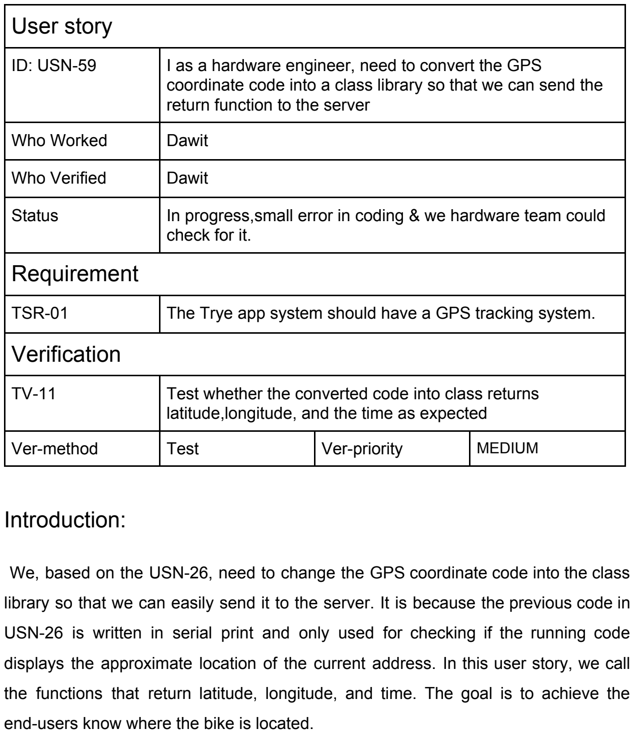


Figure-2: Cloud function created for sending message/command through a pub/sub topic.

Literature:

1. https://cloud.google.com/iot/docs/how-tos/config/configuring-devices
2. https://cloud.google.com/iot/docs/concepts/devices
3. https://cloud.google.com/functions/docs/tutorials/pubsub
4. https://cloud.google.com/functions/docs/concepts/events-triggers
5. https://cloud.google.com/functions/docs/writing/background

A.5.18 USN-59 Changing the GPS coordinate code into a class library



User story			
ID: USN-59	I as a hardware engineer, need to convert the GPS coordinate code into a class library so that we can send the return function to the server		
Who Worked	Dawit		
Who Verified	Dawit		
Status	In progress,small error in coding & we hardware team could check for it.		
Requirement			
TSR-01	The Trye app system should have a GPS tracking system.		
Verification			
TV-11	Test whether the converted code into class returns latitude,longitude, and the time as expected		
Ver-method	Test	Ver-priority	MEDIUM

Introduction:

We, based on the USN-26, need to change the GPS coordinate code into the class library so that we can easily send it to the server. It is because the previous code in USN-26 is written in serial print and only used for checking if the running code displays the approximate location of the current address. In this user story, we call the functions that return latitude, longitude, and time. The goal is to achieve the end-users know where the bike is located.

Method Used:

The need to change the GPS coordinate code into the class is for better code structure and reuse the code for USN-26. Since the code in USN-26 is not large, we have only created the header file and put it in the library. The libraries included in the header file are TinyGPS++, SoftwareSerial, and the Arduino library. The TinyGPS++ and SoftwareSerial provides most of the NMEA GPS functionality and avoids any mandatory floating point dependency and ignores all except the few key GPS fields. But the Arduino library helps us to make it connect to the sensor and module. In the header, we have three functions, such as String get.gps.time(); String get((gps.location.lng(),6)); and String get((gps.location.lat(),6)) where the last two consecutive functions return latitude and longitude, respectively.

Contribution:

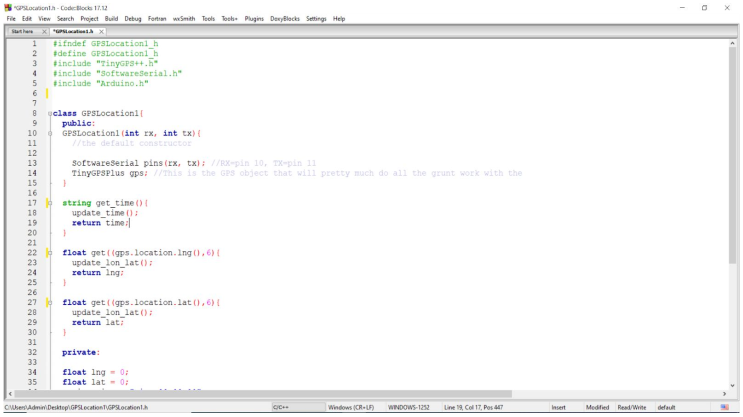
Joachim:

- Helped construct the class, so it was easier to understand and could easily copy past existing code into the class

Conclusion:

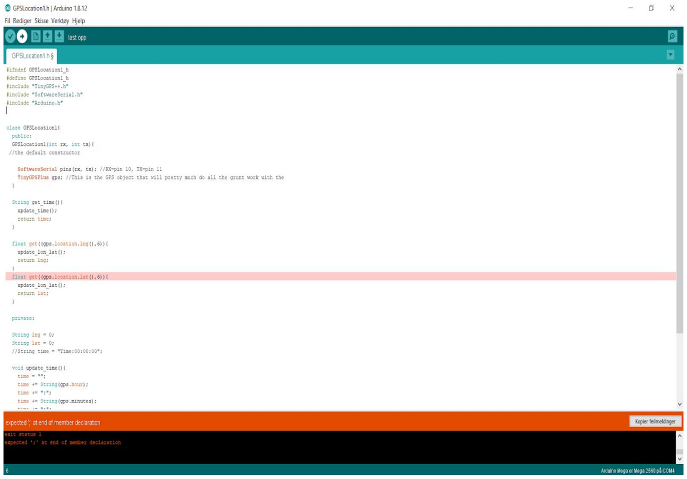
Even though the objective is to reuse the GPS coordinate code in USN-26 and send it to the server, the code is repeatedly showing an error when running. I have tried it many times, but the same thing is happening. So, I hope we, a hardware team, can fix it together.

Figures:


*GPSLocation1.h - Code::Blocks 17.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Start here
*GPSLocation1.h
C:\Users\Admin\Desktop\GPSLocation1\GPSLocation1.h
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 19, Col 17, Pos 447
Insert
Modified
Read/Write
default

```
#ifndef GPSLocation1_h
#define GPSLocation1_h
#include "TinyGPS++.h"
#include "SoftwareSerial.h"
#include "Arduino.h"


class GPSLocation1{
  public:
  GPSLocation1(int rx, int tx){
    //the default constructor

    SoftwareSerial pins(rx, tx); //RX=pin 10, TX=pin 11
    TinyGPSPlus gps; //This is the GPS object that will pretty much do all the grunt work with the
  }

  string get_time(){
    update_time();
    return time;
  }

  float get((gps.location.lng(),6){
    update_lon_lat();
    return lng;
  }

  float get((gps.location.lat(),6){
    update_lon_lat();
    return lat;
  }

  private:

  float lng = 0;
  float lat = 0;
```

Fig 1. The header file


GPSLocation1.h | Arduino 1.8.12
Fil Rediger Skisse Verktøy Hjelp
last opp
GPSLocation1.h §
Kopier feilmeldinger
Arduino Mega or Mega 2560 på COM4

```
#ifndef GPSLocation1_h
#define GPSLocation1_h
#include "TinyGPS++.h"
#include "SoftwareSerial.h"
#include "Arduino.h"


class GPSLocation1{
  public:
  GPSLocation1(int rx, int tx){
 //the default constructor

    SoftwareSerial pins(rx, tx); //RX=pin 10, TX=pin 11
    TinyGPSPlus gps; //This is the GPS object that will pretty much do all the grunt work with the
  }

  String get_time(){
    update_time();
    return time;
  }

  float get((gps.location.lng(),6)){
    update_lon_lat();
    return lng;
  }
  float get((gps.location.lat(),6)){
    update_lon_lat();
    return lat;
  }

  private:

  String lng = 0;
  String lat = 0;
  //String time = "Time:00:00:00";

  void update_time(){
    time = "";
    time += String(gps.hour);
    time += ":";
    time += String(gps.minutes);
```

```
expected ';' at end of member declaration
```

```
exit status 1
expected ';' at end of member declaration
```

```
6
```

Fig 2. The class code when loading

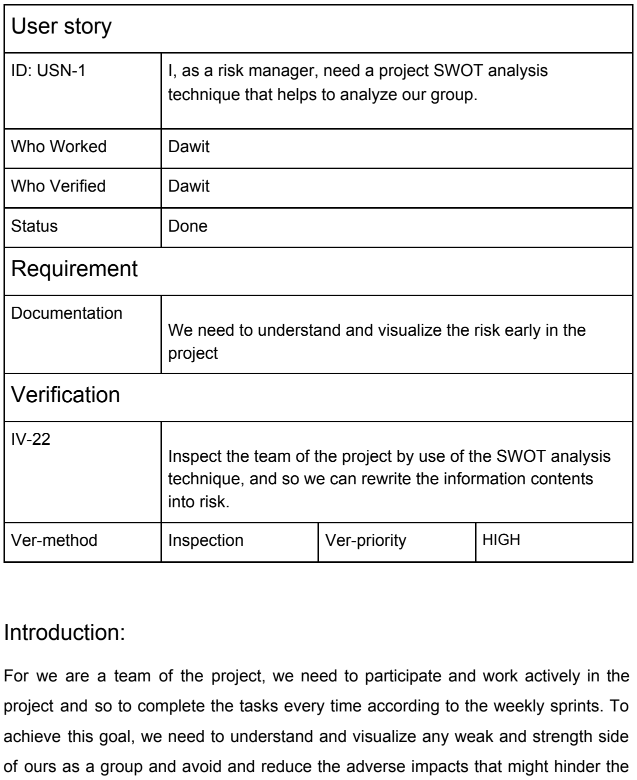
Literature:

[1]https://create.arduino.cc/projecthub/ruchir1674/how-to-interface-gps-module-neo-6m-with-arduino-8f90ad

[2]https://www.arduino.cc/en/Guide/ArduinoUn

A.6 Non technical User Stories

A.6.1 USN-1 I as the risk manager need to perform a project SWOT analysis to help analyze our group



User story			
ID: USN-1	I, as a risk manager, need a project SWOT analysis technique that helps to analyze our group.		
Who Worked	Dawit		
Who Verified	Dawit		
Status	Done		
Requirement			
Documentation	We need to understand and visualize the risk early in the project		
Verification			
IV-22	Inspect the team of the project by use of the SWOT analysis technique, and so we can rewrite the information contents into risk.		
Ver-method	Inspection	Ver-priority	HIGH

Introduction:

For we are a team of the project, we need to participate and work actively in the project and so to complete the tasks every time according to the weekly sprints. To achieve this goal, we need to understand and visualize any weak and strength side of ours as a group and avoid and reduce the adverse impacts that might hinder the success of our project. We also need to visualize the impacts of external factors that affect the objectives of our project.

Method Used:

We, as a group, sit down and discuss the SWOT analysis. SWOT stands for Strengths, Weakness, Threats, and Opportunities. We try to identify all potential uncertainties that might happen due to internal and external factors that go inside and outside of the project. The internal factors refer to the Weakness and Strengths, and which are the merit and demerit side of the group that affects the objectives. The external factors refer to the Threats and Opportunities that occur outside the project and also affects the objectives. After identified all potential uncertainties, we individually decided which of these uncertainties are to accept or reject. These Items we have mostly accepted are their contents rewritten as a risk and taken for further analysis.

Conclusion:

SWOT-analysis helps us to get information that we have already realized early in the project and then rewrite their contents as a risk. In general, it provides us to maximize the opportunities by taking advantage of real risks and reduce and avoid the adverse risk by making a better decision.

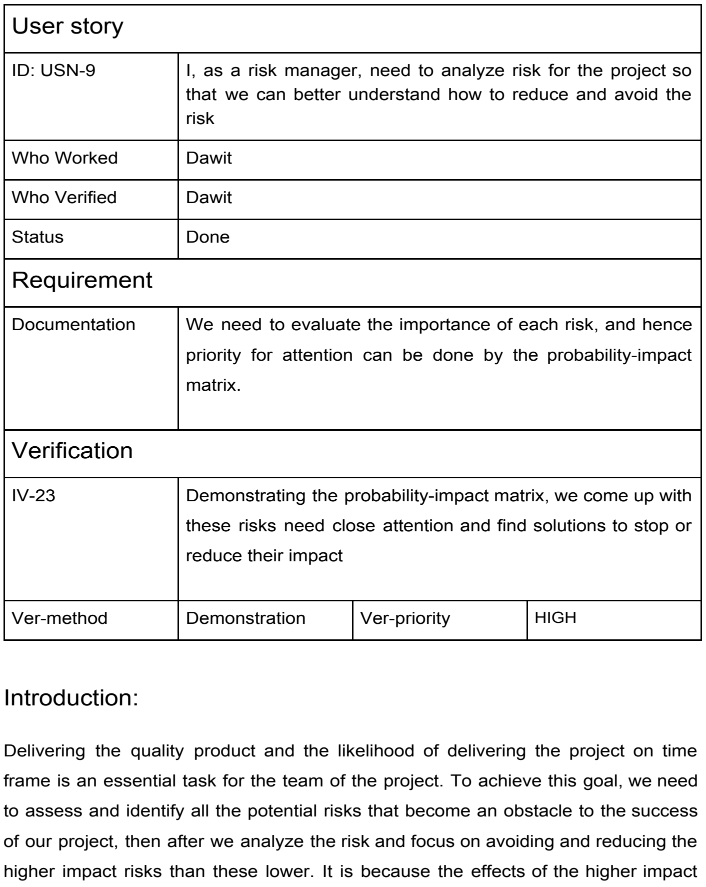
Literature:

[1]https://www.investopedia.com/terms/s/swot.asp

[2]https://articles.bplans.com/how-to-perform-swot-analysis/

[3]https://www.smartdraw.com/swot-analysis/

A.6.2 USN-9 Analyze the risk for our project for a better understanding of how to reduce the risk



User story			
ID: USN-9	I, as a risk manager, need to analyze risk for the project so that we can better understand how to reduce and avoid the risk		
Who Worked	Dawit		
Who Verified	Dawit		
Status	Done		
Requirement			
Documentation	We need to evaluate the importance of each risk, and hence priority for attention can be done by the probability-impact matrix.		
Verification			
IV-23	Demonstrating the probability-impact matrix, we come up with these risks need close attention and find solutions to stop or reduce their impact		
Ver-method	Demonstration	Ver-priority	HIGH

Introduction:

Delivering the quality product and the likelihood of delivering the project on time frame is an essential task for the team of the project. To achieve this goal, we need to assess and identify all the potential risks that become an obstacle to the success of our project, then after we analyze the risk and focus on avoiding and reducing the higher impact risks than these lower. It is because the effects of the higher impact risk on the objectives of the project are serious. The process of checking for risk by the team members takes place every week, depending on the sprints.

Method Used:

We, as the members of the team, sit down and develop a risk management plan together and identifies all potential risks for all parts of the project. These identified risks can make wrong in the project in terms of scope, schedule, budget, and quality. We give each risk an ID number and put them all in the risk register for ease of controlling and monitoring them weekly based on the scrum model we follow. We use the probability-impact matrix to assign the likelihood and impacts of each risk. We make the spreadsheet for calculating the weight for the risk product. We find the risk product by multiplying the assigned value for likelihood times the assigned value for impacts. The members of the team assign value for each risk based on the risk impact affecting the project. The lower the value assigned, the lower the risk impact, whereas the higher the value assigned, the higher the impact it has on the project. We give different colors for each risk product based on the size of the impact. The red color represents the size of the highest impact of the risk, whereas the light green color represents the lower impact of the risk. The color varies between light green and red color and represents the size of the medium impact of the risk. Once we finish with the risk matrix, we need to find the cause for each risk and develop the mitigation techniques to avoid and reduce any adverse of the risk. Since our project is small, we don't need to evaluate the risk in term quantitatively. We sit down as a group every week and look at the new risk coming from each of us. We can also remove these risks that have little effect on the objective of the project. In this approach, we can control and monitor the risk.

Conclusion:

Risk analysis helps us to trace any problem whenever it happens in each part of the project following the ID number for the risk. By using risk analysis, we can manage to avoid any potential risk and reduce the exposure of our project to any risk. Moreover, we can minimize the impact of the risk and increase the success of our project.

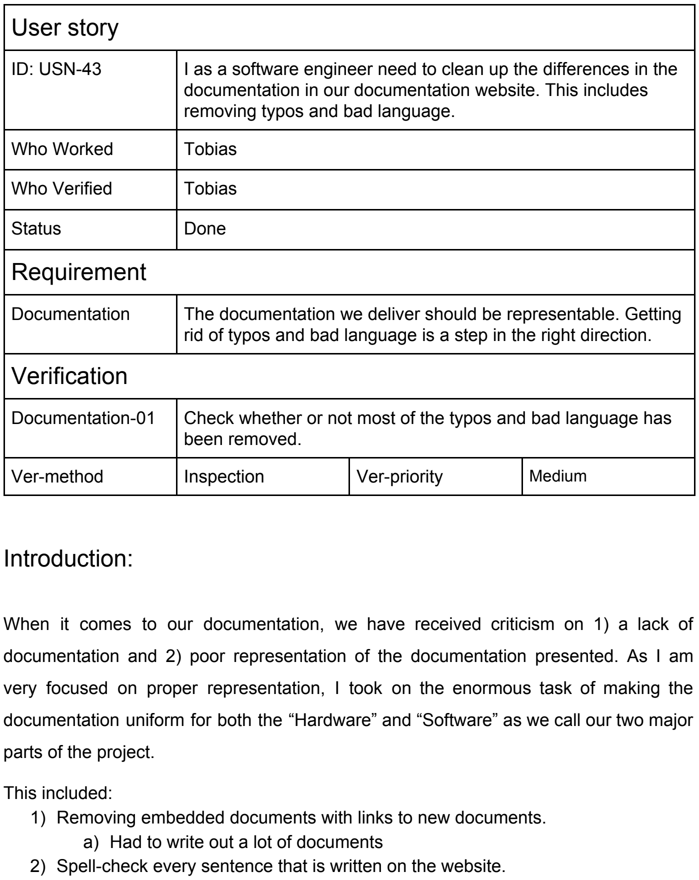
Literature:

[1]https://www.pmi.org/learning/library/risk-analysis-project-management-7070

[2]https://www.sciencedirect.com/topics/earth-and-planetary-sciences/risk-analysis

[3]https://searchsecurity.techtarget.com/definition/risk-analysis

A.6.3 USN-43 Cleaning up differences in the documentation on the website



User story			
ID: USN-43	I as a software engineer need to clean up the differences in the documentation in our documentation website. This includes removing typos and bad language.		
Who Worked	Tobias		
Who Verified	Tobias		
Status	Done		
Requirement			
Documentation	The documentation we deliver should be representable. Getting rid of typos and bad language is a step in the right direction.		
Verification			
Documentation-01	Check whether or not most of the typos and bad language has been removed.		
Ver-method	Inspection	Ver-priority	Medium

Introduction:

When it comes to our documentation, we have received criticism on 1) a lack of documentation and 2) poor representation of the documentation presented. As I am very focused on proper representation, I took on the enormous task of making the documentation uniform for both the “Hardware” and “Software” as we call our two major parts of the project.

This included:

1) Removing embedded documents with links to new documents.
    a) Had to write out a lot of documents
2) Spell-check every sentence that is written on the website.

3) When listing User Stories, I made them look exactly the same under the "Hardware" and "Software" tab.

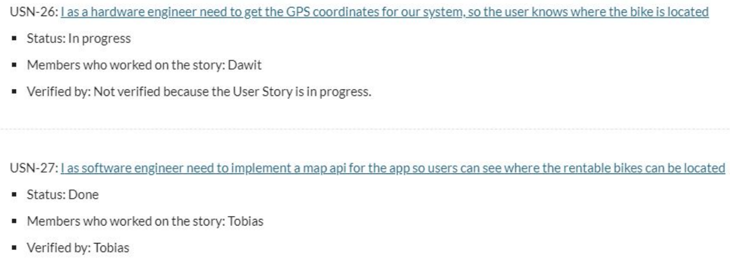
Method Used:

I started from the left and worked my way through all the different sub-menus.

Under the -

- “Requirements” tab I wrote out a handful of embedded documents that lead to new documents with links to new documents. To start off, I wrote out the entire requirement document given to us by Trye in Norwegian, then English, followed by a “Requirement Analysis”, “List of System Requirements and Verification” and “Component Selection of the Hardware Subsystem”.

- “User Stories” tab I wrote out all 43 User Stories we had worked on in a readable fashion, and with a link to the original User Story if one would be interested in the title.



USN-26: I as a hardware engineer need to get the GPS coordinates for our system, so the user knows where the bike is located

- Status: In progress
- Members who worked on the story: Dawit
- Verified by: Not verified because the User Story is in progress.

USN-27: I as software engineer need to implement a map api for the app so users can see where the rentable bikes can be located

- Status: Done
- Members who worked on the story: Tobias
- Verified by: Tobias

Above is a snippet of the 43 User Stories. In addition, I renamed all User Story documents to contain a short description of what the User Story was about as a title. Previously the entire name of a User Story was USN-43, now User Story is named: “USN-43 Cleaning up differences in the documentation on the website”.

- “Risk” tab I wrote out as much as I could of documents leading to documents.

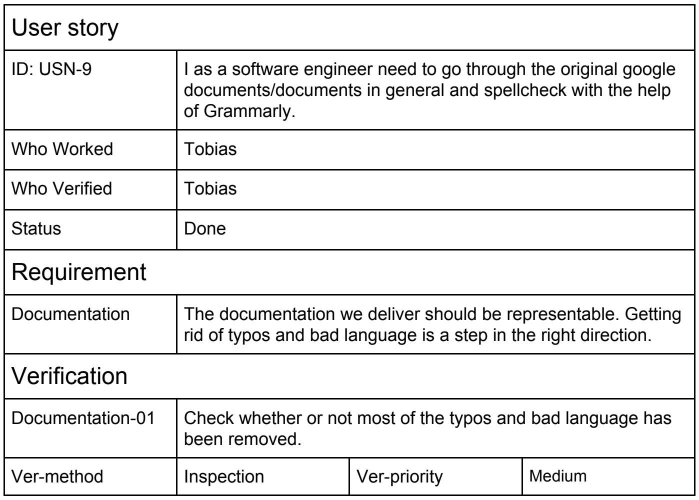
- “Software” tab I copied the way of listing User Stories from the Hardware tab (minor changes) with, 1) User Story number, 2) User Story summary, 3) The entire User Story description, 4) Link to the original document.
- “Hardware” tab I did the least amount of change, barely changing how the User Stories were listed so that they were uniform with the “Software” tab.

Conclusion:

I am certain that the changes done to the presentation of our documentation is positive. Having embedded Google Documents with links to new Google Documents made it easy to get lost and forget what document one was initially reading.

On the other hand, I am aware of user interface being very subjective, so to verify the improvement of readability I asked Jose for his feedback. His opinion was that the change was a positive addition and a step in the right direction, but that we still had work to do. Further work will be done to the website when we are closing in on the deadline for submission.

A.6.4 USN-47 Spellchecking our entire documentation stack with Grammarly



User story			
ID: USN-9	I as a software engineer need to go through the original google documents/documents in general and spellcheck with the help of Grammarly.		
Who Worked	Tobias		
Who Verified	Tobias		
Status	Done		
Requirement			
Documentation	The documentation we deliver should be representable. Getting rid of typos and bad language is a step in the right direction.		
Verification			
Documentation-01	Check whether or not most of the typos and bad language has been removed.		
Ver-method	Inspection	Ver-priority	Medium

Introduction:

As we decided to write our bachelor thesis in English, I expected the language and amount of typos to be quite bad. This combined with the deadline for submission closing in, I found it reasonable to visit every document created since January and get rid of obvious typos. To help me achieve this I bought a subscription for Grammarly that automatically detects typos and bad language in Google Documents. This included document titles, not only it's content.

Note: It is not my intention to bring shame upon any of my group members but as it appears that Google Docs don't automatically spell-check our documentation stack was full of typos and bad language. It was especially a tough challenge to spell-check the Google Sheet (Google's own Excel) documents as Grammarly didn't recognize the text within cells. The only reason I emphasize that a proper round of spell-checking was needed is to make it understandable that it took quite some time to go through every document created since January.

Method Used:

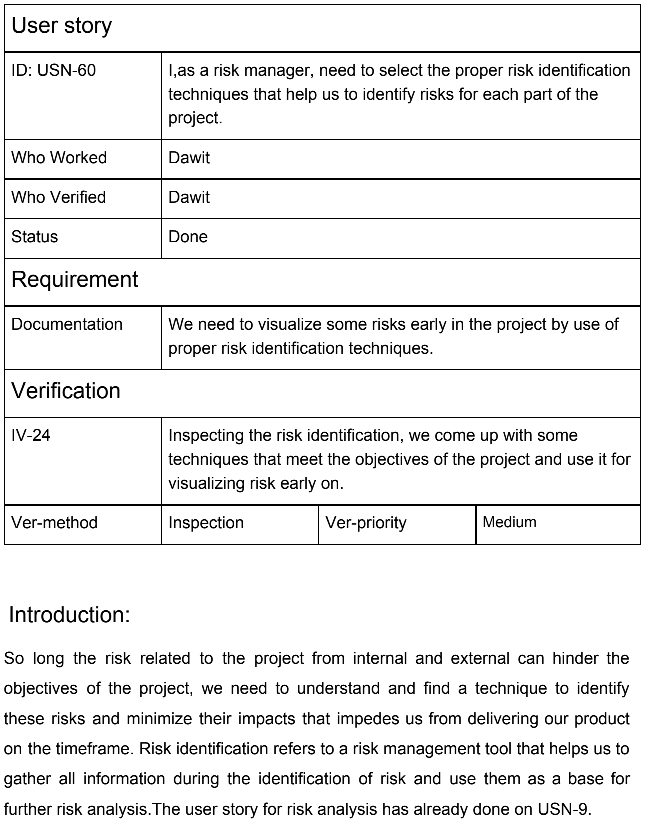
The method used was fairly simple, I opened one Google Document, and scanned it thoroughly from top to bottom. When it came to the Google Sheet documents, I had to reformat entire documents to make them visually appealing. When saying visually appealing I mean making the documents uniform as this was not the case before I started working. After a couple of days and sore eyes, I had finally ploughed my way through every digital document.

Conclusion:

In USN-43 I spell-checked and made the website uniform without touching the "original" documents, here I tackled the challenge of getting rid of typos and obvious bad language in the “original” documents. A challenge I met throughout the work was that some sentences did not make sense at all. I suspect that a mix of Google Translate, fancy words and guessing created impossible to understand sentences. Where obvious, I rewrote sentences to say what I thought they were supposed to say, but as I am not an expert in what the other group members are working on this was difficult.

Because of this, I believe some documents might be rid of “typos” and have the correct “article”, but no obvious meaning. I have reached out to the group and asked them to go over the documents I’ve corrected so they can take a look at the sentences.

A.6.5 USN-60 Need to select the proper risk identification technique to our project



User story			
ID: USN-60	I,as a risk manager, need to select the proper risk identification techniques that help us to identify risks for each part of the project.		
Who Worked	Dawit		
Who Verified	Dawit		
Status	Done		
Requirement			
Documentation	We need to visualize some risks early in the project by use of proper risk identification techniques.		
Verification			
IV-24	Inspecting the risk identification, we come up with some techniques that meet the objectives of the project and use it for visualizing risk early on.		
Ver-method	Inspection	Ver-priority	Medium

Introduction:

So long the risk related to the project from internal and external can hinder the objectives of the project, we need to understand and find a technique to identify these risks and minimize their impacts that impedes us from delivering our product on the timeframe. Risk identification refers to a risk management tool that helps us to gather all information during the identification of risk and use them as a base for further risk analysis.The user story for risk analysis has already done on USN-9.

Method Used:

The first thing we do in risk identification is that we try to understand the area where the risk occurs in the project. The risk occurs and gradually affects the scope, budget, schedule, and quality of the project. Moreover, the risk may affect the expectation of our customers. We sit down and discuss the risk identification in a group to understand which techniques are relevant for our project and help us identify risk effectively. Brainstorming is one of the risk identification techniques that we use in identifying risk. It is because it helps us to create a method to look for risk now and in the future in the project. As we are working on the software and hardware part, everybody comes with risk in mind that related to the parts. Then we document these risks that we can see for a while. It is often our choice or decision on some issues that arise a new risk which we didn`t see in the previous steps. We also use the SWOT analysis technique because we need to analyze ourselves as a group to understand and visualize everything early in the project. We have already done the user story on group analysis in USN-1 in detail

Conclusion:

By using risk identification techniques, we can do the risk analysis in detail and address the most potential risks in the project. It also helps us to understand the scopes of the project early.

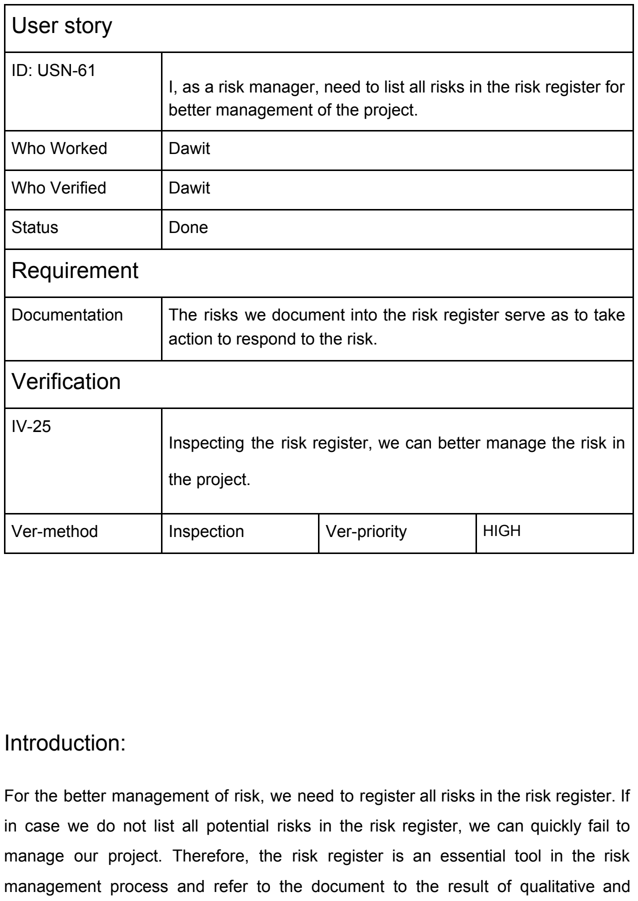
Literature:

[1]https://www.pmi.org/learning/library/risk-identification-life-cycle-tools-7784

[2]https://www.mitre.org/publications/systems-engineering-guide/acquisition-systems-engineering/risk-management/risk-identification

[3]https://www.greycampus.com/opencampus/certified-associate-in-project-management/risk-identification-tools-and-techniques-in-capm

A.6.6 USN-61 We need to list all risks in the risk register for better management of the project



User story			
ID: USN-61	I, as a risk manager, need to list all risks in the risk register for better management of the project.		
Who Worked	Dawit		
Who Verified	Dawit		
Status	Done		
Requirement			
Documentation	The risks we document into the risk register serve as to take action to respond to the risk.		
Verification			
IV-25	Inspecting the risk register, we can better manage the risk in the project.		
Ver-method	Inspection	Ver-priority	HIGH

Introduction:

For the better management of risk, we need to register all risks in the risk register. If in case we do not list all potential risks in the risk register, we can quickly fail to manage our project. Therefore, the risk register is an essential tool in the risk management process and refer to the document to the result of qualitative and quantitative risk analysis and risk response planning. We focus on the qualitative part, which, based on prioritizing the risk and determines the probability and impact of the project. The risk register comprises much information in which each related to individual risk. It contains the Id number of the risk, the description of each risk, the date when the risk identified, the owner of the risk, the causes of the risk, the likelihood of the risk, the impact of the risk, the risk product, the mitigation actions, and response planning. We need the risk register because it helps us describing the risk consistently, enables us to analyze the risk, and it uses as a record for future use.

Method Used:

The first thing, we, as a team of the project, need to identify all potential risks that negatively affect the project. We sit down and write the description for each risk identified. We need to understand and write the causes for each risk. Each risk has given an ID which uses to trace and apply for the future. The responsible person, who keep managing the risk, and the date when the risk identified document the risk register. Then we assess the probability of the risk happening and the impact of the risk if it occurs. We need to use a spreadsheet to calculate the risk product by multiplying probability with the impact of the risk. The higher the risk product, the higher the impact of the risk affecting our project. We need to represent different colors for each risk products. The red color is a representation of a higher risk, whereas the light green color is a representation of lower risk. The color from light yellow to yellow is a representation of medium risk.

We mainly focus on the higher risk product because its effect on the project is tremendously huge. So, we need to find the mitigation actions for each recorded risk products. Then we come up with the response plan and monitor the risk throughout the project. Since we follow the scrum model, we need to evaluate the risk every week with the team of the project. We can reject some of the risks because of its effect on the project is minimum. Some of them we can continue to control and monitor till the lifetime of the project.

Conclusion:

By using a risk register, we can easily manage the risk and increase the likelihood of the success of our project. In general, risk register allow us to analyze the contents of risk and timing. It also creates the opportunity to learn by reviewing the recorded what happened. It provides us to make sure that every risk described at the same level detail.

Literature:

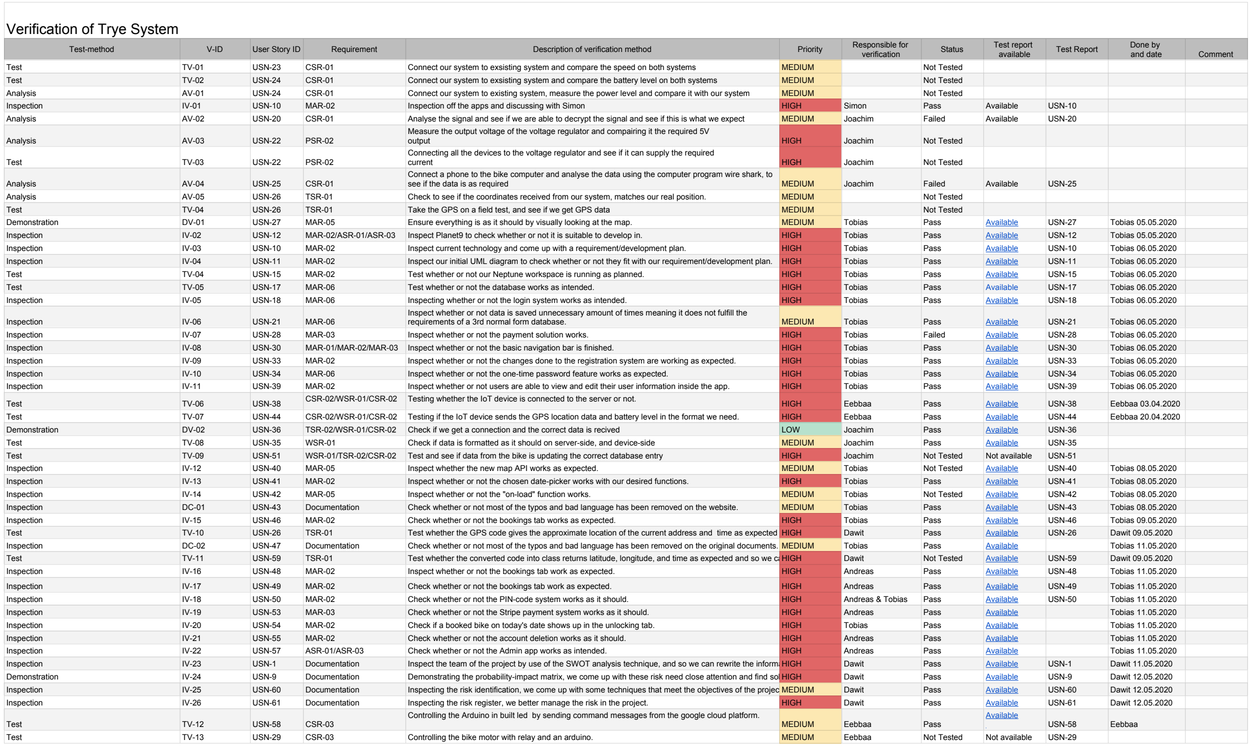
[1]https://www.projectmanager.com/blog/guide-using-risk-register

[2]https://www.sciencedirect.com/topics/engineering/risk-register

[3]https://pmdprostarter.org/risk-register/

A.7 Verification documents

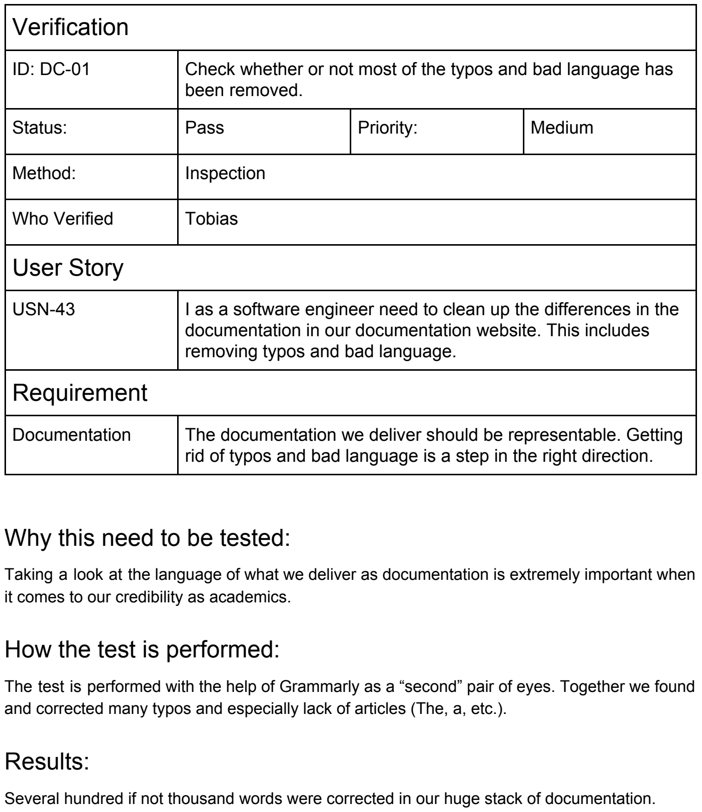
A.7.1 Verification spreadsheet



Verification of Trye System

Test-method	V-ID	User Story ID	Requirement	Description of verification method	Priority	Responsible for verification	Status	Test report available	Test Report	Done by and date	Comment
Test	TV-01	USN-23	CSR-01	Connect our system to exsisting system and compare the speed on both systems	MEDIUM		Not Tested				
Test	TV-02	USN-24	CSR-01	Connect our system to exsisting system and compare the battery level on both systems	MEDIUM		Not Tested				
Analysis	AV-01	USN-24	CSR-01	Connect our system to existing system, measure the power level and compare it with our system	MEDIUM		Not Tested				
Inspection	IV-01	USN-10	MAR-02	Inspection off the apps and discussing with Simon	HIGH	Simon	Pass	Available	USN-10		
Analysis	AV-02	USN-20	CSR-01	Analyse the signal and see if we are able to decrypt the signal and see if this is what we expect	MEDIUM	Joachim	Failed	Available	USN-20		
Analysis	AV-03	USN-22	PSR-02	Measure the output voltage of the voltage regulator and compairing it the required 5V output	HIGH	Joachim	Not Tested				
Test	TV-03	USN-22	PSR-02	Connecting all the devices to the voltage regulator and see if it can supply the required current	HIGH	Joachim	Not Tested				
Analysis	AV-04	USN-25	CSR-01	Connect a phone to the bike computer and analyse the data using the computer program wire shark, to see if the data is as required	MEDIUM	Joachim	Failed	Available	USN-25		
Analysis	AV-05	USN-26	TSR-01	Check to see if the coordinates received from our system, matches our real position.	MEDIUM		Not Tested				
Test	TV-04	USN-26	TSR-01	Take the GPS on a field test, and see if we get GPS data	MEDIUM		Not Tested				
Demonstration	DV-01	USN-27	MAR-05	Ensure everything is as it should by visually looking at the map.	MEDIUM	Tobias	Pass	Available	USN-27	Tobias 05.05.2020	
Inspection	IV-02	USN-12	MAR-02/ASR-01/ASR-03	Inspect Planet9 to check whether or not it is suitable to develop in.	HIGH	Tobias	Pass	Available	USN-12	Tobias 05.05.2020	
Inspection	IV-03	USN-10	MAR-02	Inspect current technology and come up with a requirement/development plan.	HIGH	Tobias	Pass	Available	USN-10	Tobias 06.05.2020	
Inspection	IV-04	USN-11	MAR-02	Inspect our initial UML diagram to check whether or not they fit with our requirement/development plan.	HIGH	Tobias	Pass	Available	USN-11	Tobias 06.05.2020	
Test	TV-04	USN-15	MAR-02	Test whether or not our Neptune workspace is running as planned.	HIGH	Tobias	Pass	Available	USN-15	Tobias 06.05.2020	
Test	TV-05	USN-17	MAR-06	Test whether or not the database works as intended.	HIGH	Tobias	Pass	Available	USN-17	Tobias 06.05.2020	
Inspection	IV-05	USN-18	MAR-06	Inspecting whether or not the login system works as intended.	HIGH	Tobias	Pass	Available	USN-18	Tobias 06.05.2020	
Inspection	IV-06	USN-21	MAR-06	Inspect whether or not data is saved unnecessary amount of times meaning it does not fulfill the requirements of a 3rd normal form database.	MEDIUM	Tobias	Pass	Available	USN-21	Tobias 06.05.2020	
Inspection	IV-07	USN-28	MAR-03	Inspect whether or not the payment solution works.	HIGH	Tobias	Failed	Available	USN-28	Tobias 06.05.2020	
Inspection	IV-08	USN-30	MAR-01/MAR-02/MAR-03	Inspect whether or not the basic navigation bar is finished.	HIGH	Tobias	Pass	Available	USN-30	Tobias 06.05.2020	
Inspection	IV-09	USN-33	MAR-02	Inspect whether or not the changes done to the registration system are working as expected.	HIGH	Tobias	Pass	Available	USN-33	Tobias 06.05.2020	
Inspection	IV-10	USN-34	MAR-06	Inspect whether or not the one-time password feature works as expected.	HIGH	Tobias	Pass	Available	USN-34	Tobias 06.05.2020	
Inspection	IV-11	USN-39	MAR-02	Inspect whether or not users are able to view and edit their user information inside the app.	HIGH	Tobias	Pass	Available	USN-39	Tobias 06.05.2020	
Test	TV-06	USN-38	CSR-02/WSR-01/CSR-02	Testing whether the IoT device is connected to the server or not.	HIGH	Eebbaa	Pass	Available	USN-38	Eebbaa 03.04.2020	
Test	TV-07	USN-44	CSR-02/WSR-01/CSR-02	Testing if the IoT device sends the GPS location data and battery level in the format we need.	HIGH	Eebbaa	Pass	Available	USN-44	Eebbaa 20.04.2020	
Demonstration	DV-02	USN-36	TSR-02/WSR-01/CSR-02	Check if we get a connection and the correct data is recived	LOW	Joachim	Pass	Available	USN-36		
Test	TV-08	USN-35	WSR-01	Check if data is formatted as it should on server-side, and device-side	MEDIUM	Joachim	Pass	Available	USN-35		
Test	TV-09	USN-51	WSR-01/TSR-02/CSR-02	Test and see if data from the bike is updating the correct database entry	HIGH	Joachim	Not Tested	Not available	USN-51		
Inspection	IV-12	USN-40	MAR-05	Inspect whether the new map API works as expected.	MEDIUM	Tobias	Not Tested	Available	USN-40	Tobias 08.05.2020	
Inspection	IV-13	USN-41	MAR-02	Inspect whether or not the chosen date-picker works with our desired functions.	HIGH	Tobias	Pass	Available	USN-41	Tobias 08.05.2020	
Inspection	IV-14	USN-42	MAR-05	Inspect whether or not the "on-load" function works.	MEDIUM	Tobias	Not Tested	Available	USN-42	Tobias 08.05.2020	
Inspection	DC-01	USN-43	Documentation	Check whether or not most of the typos and bad language has been removed on the website.	MEDIUM	Tobias	Pass	Available	USN-43	Tobias 08.05.2020	
Inspection	IV-15	USN-46	MAR-02	Check whether or not the bookings tab works as expected.	HIGH	Tobias	Pass	Available	USN-46	Tobias 09.05.2020	
Test	TV-10	USN-26	TSR-01	Test whether the GPS code gives the approximate location of the current address and time as expected	HIGH	Dawit	Pass	Available	USN-26	Dawit 09.05.2020	
Inspection	DC-02	USN-47	Documentation	Check whether or not most of the typos and bad language has been removed on the original documents.	MEDIUM	Tobias	Pass	Available		Tobias 11.05.2020	
Test	TV-11	USN-59	TSR-01	Test whether the converted code into class returns latitude, longitude, and time as expected and so we ca	HIGH	Dawit	Not Tested	Available	USN-59	Dawit 09.05.2020	
Inspection	IV-16	USN-48	MAR-02	Inspect whether or not the bookings tab work as expected.	HIGH	Andreas	Pass	Available	USN-48	Tobias 11.05.2020	
Inspection	IV-17	USN-49	MAR-02	Check whether or not the bookings tab work as expected.	HIGH	Andreas	Pass	Available	USN-49	Tobias 11.05.2020	
Inspection	IV-18	USN-50	MAR-02	Check whether or not the PIN-code system works as it should.	HIGH	Andreas & Tobias	Pass	Available	USN-50	Tobias 11.05.2020	
Inspection	IV-19	USN-53	MAR-03	Check whether or not the Stripe payment system works as it should.	HIGH	Andreas	Pass	Available		Tobias 11.05.2020	
Inspection	IV-20	USN-54	MAR-02	Check if a booked bike on today's date shows up in the unlocking tab.	HIGH	Tobias	Pass	Available		Tobias 11.05.2020	
Inspection	IV-21	USN-55	MAR-02	Check whether or not the account deletion works as it should.	HIGH	Andreas	Pass	Available		Tobias 11.05.2020	
Inspection	IV-22	USN-57	ASR-01/ASR-03	Check whether or not the Admin app works as intended.	HIGH	Andreas	Pass	Available		Tobias 11.05.2020	
Inspection	IV-23	USN-1	Documentation	Inspect the team of the project by use of the SWOT analysis technique, and so we can rewrite the inform	HIGH	Dawit	Pass	Available	USN-1	Dawit 11.05.2020	
Demonstration	IV-24	USN-9	Documentation	Demonstrating the probability-impact matrix, we come up with these risk need close attention and find so	HIGH	Dawit	Pass	Available	USN-9	Dawit 12.05.2020	
Inspection	IV-25	USN-60	Documentation	Inspecting the risk identification, we come up with some techniques that meet the objectives of the projec	MEDIUM	Dawit	Pass	Available	USN-60	Dawit 12.05.2020	
Inspection	IV-26	USN-61	Documentation	Inspecting the risk register, we better manage the risk in the project.	HIGH	Dawit	Pass	Available	USN-61	Dawit 12.05.2020	
Test	TV-12	USN-58	CSR-03	Controlling the Arduino in built led by sending command messages from the google cloud platform.	MEDIUM	Eebbaa	Pass	Available	USN-58	Eebbaa	
Test	TV-13	USN-29	CSR-03	Controlling the bike motor with relay and an arduino.	MEDIUM	Eebbaa	Not Tested	Not available	USN-29		

A.7.2 DC-01 Verifying USN-43



Verification			
ID: DC-01	Check whether or not most of the typos and bad language has been removed.		
Status:	Pass	Priority:	Medium
Method:	Inspection		
Who Verified	Tobias		
User Story			
USN-43	I as a software engineer need to clean up the differences in the documentation in our documentation website. This includes removing typos and bad language.		
Requirement			
Documentation	The documentation we deliver should be representable. Getting rid of typos and bad language is a step in the right direction.		

Why this need to be tested:

Taking a look at the language of what we deliver as documentation is extremely important when it comes to our credibility as academics.

How the test is performed:

The test is performed with the help of Grammarly as a “second” pair of eyes. Together we found and corrected many typos and especially lack of articles (The, a, etc.).

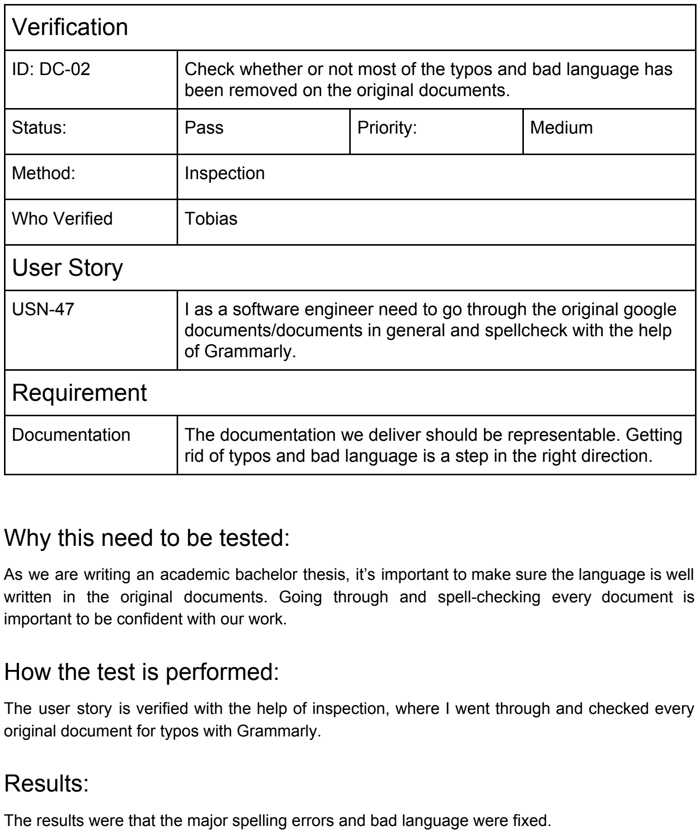
Results:

Several hundred if not thousand words were corrected in our huge stack of documentation.

Conclusion:

As of the date of USN-43, most of the documentation is typo-free. The reason I say most is that it's impossible to be a hundred per cent certain that everything is well written.

A.7.3 DC-02 Verifying USN-47



Verification			
ID: DC-02	Check whether or not most of the typos and bad language has been removed on the original documents.		
Status:	Pass	Priority:	Medium
Method:	Inspection		
Who Verified	Tobias		
User Story			
USN-47	I as a software engineer need to go through the original google documents/documents in general and spellcheck with the help of Grammarly.		
Requirement			
Documentation	The documentation we deliver should be representable. Getting rid of typos and bad language is a step in the right direction.		

Why this need to be tested:

As we are writing an academic bachelor thesis, it's important to make sure the language is well written in the original documents. Going through and spell-checking every document is important to be confident with our work.

How the test is performed:

The user story is verified with the help of inspection, where I went through and checked every original document for typos with Grammarly.

Results:

The results were that the major spelling errors and bad language were fixed.

Conclusion:

As the major spelling errors and bad language was fixed in the entire stack of original documentation, this gives a pass in the verification.

A.7.4 DV-01 Verifying USN-27

Verification			
ID: DV-01	Ensure everything is as it should by visually looking at the map.		
Status:	Pass	Priority:	Medium
Method:	Demonstration		
Who Verified	Tobias		
User Story			
USN-27	I as a software engineer need to implement a map API for the app so users can see where the rentable bikes can be located.		
Requirement			
MAR-01 MAR-02 MAR-03	The mobile app should have a renting system. The mobile app should have a booking system. The mobile app should have a payment app.		

Why this need to be tested:

As our app is made to be as user friendly as possible, having clickable bikes that lead you to the renting/payment solution is an important feature. Trye's mission is first of all to ensure that the users of the app/their service gets the best possible service, but also make money as a company.

How the test is performed:

The test will be done with the help of demonstration. The app will be opened on different mobile phones/pc browsers to ensure that the experience meets our criteria.

Results:

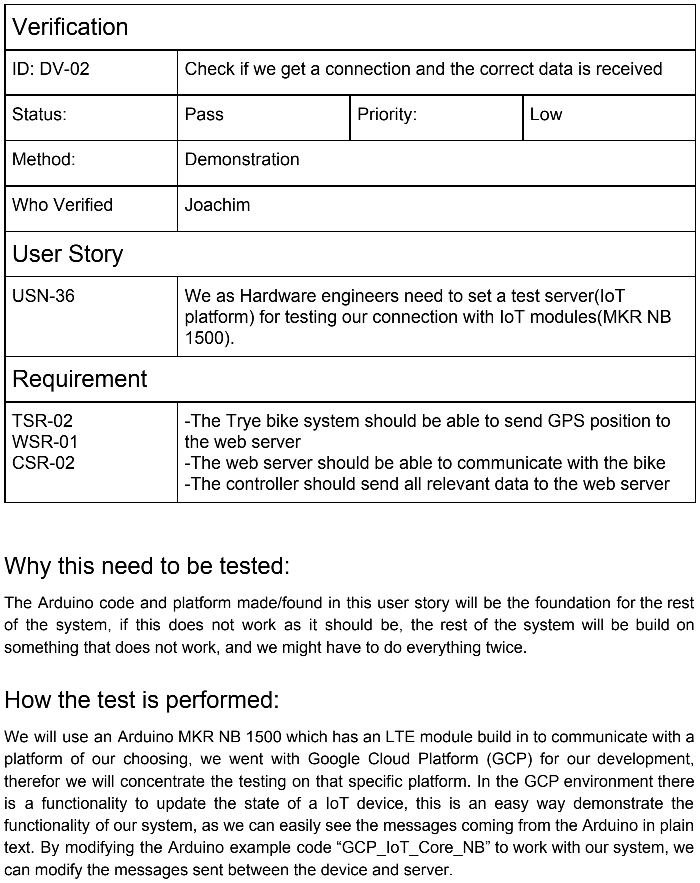
The bikes are successfully being displayed on the map. The visibility is also very good as a white background with a black bike logo is easy to see.

gen
ongsberg
40

Conclusion:

As everything works as expected and it’s easy to spot the bikes this means that the User Story gets to pass the verification.

A.7.5 DV-02 Verifying USN-36



Verification			
ID: DV-02	Check if we get a connection and the correct data is received		
Status:	Pass	Priority:	Low
Method:	Demonstration		
Who Verified	Joachim		
User Story			
USN-36	We as Hardware engineers need to set a test server(IoT platform) for testing our connection with IoT modules(MKR NB 1500).		
Requirement			
TSR-02 WSR-01 CSR-02	-The Trye bike system should be able to send GPS position to the web server -The web server should be able to communicate with the bike -The controller should send all relevant data to the web server		

Why this need to be tested:

The Arduino code and platform made/found in this user story will be the foundation for the rest of the system, if this does not work as it should be, the rest of the system will be build on something that does not work, and we might have to do everything twice.

How the test is performed:

We will use an Arduino MKR NB 1500 which has an LTE module build in to communicate with a platform of our choosing, we went with Google Cloud Platform (GCP) for our development, therefor we will concentrate the testing on that specific platform. In the GCP environment there is a functionality to update the state of a IoT device, this is an easy way demonstrate the functionality of our system, as we can easily see the messages coming from the Arduino in plain text. By modifying the Arduino example code “GCP_IoT_Core_NB” to work with our system, we can modify the messages sent between the device and server.

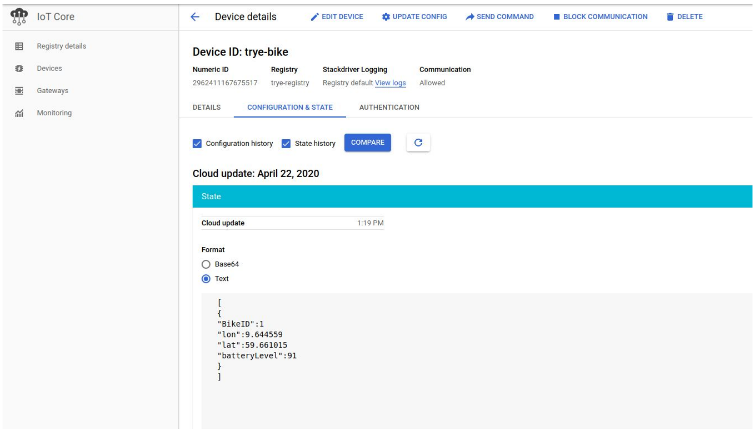
Results:

The message written in the Arduino code ended up as a state in the GCP environment, under the correct device. The message can be viewed in FIG1.

Conclusion:

The sub-system works as it should be, we can now work with relaying the messages to a database for storage, and requested the latest message by the mobile app.

Figure:



Device ID: trye-bike

Numeric ID	Registry	Stackdriver Logging	Communication
2962411167675517	trye-registry	Registry default View logs	Allowed

☑ Configuration history ☑ State history

Cloud update: April 22, 2020

Cloud update 1:19 PM

Format

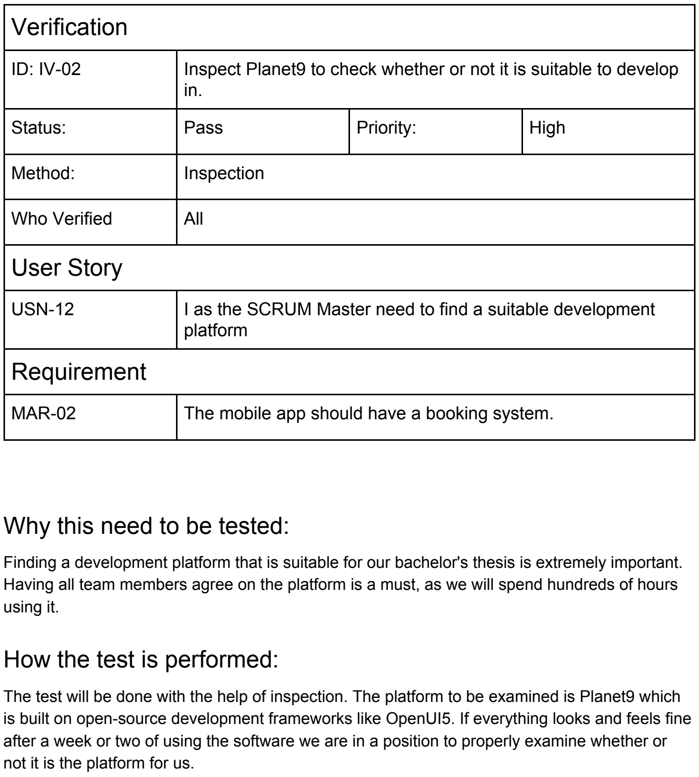
○ Base64

● Text

```
[
{
"BikeID":1
"lon":9.644559
"lat":59.661015
"batteryLevel":91
}
]
```

FIG1: we can see the result we got from the Arduino

A.7.6 IV-02 Verifying USN-12



Verification			
ID: IV-02	Inspect Planet9 to check whether or not it is suitable to develop in.		
Status:	Pass	Priority:	High
Method:	Inspection		
Who Verified	All		
User Story			
USN-12	I as the SCRUM Master need to find a suitable development platform		
Requirement			
MAR-02	The mobile app should have a booking system.		

Why this need to be tested:

Finding a development platform that is suitable for our bachelor's thesis is extremely important. Having all team members agree on the platform is a must, as we will spend hundreds of hours using it.

How the test is performed:

The test will be done with the help of inspection. The platform to be examined is Planet9 which is built on open-source development frameworks like OpenUI5. If everything looks and feels fine after a week or two of using the software we are in a position to properly examine whether or not it is the platform for us.

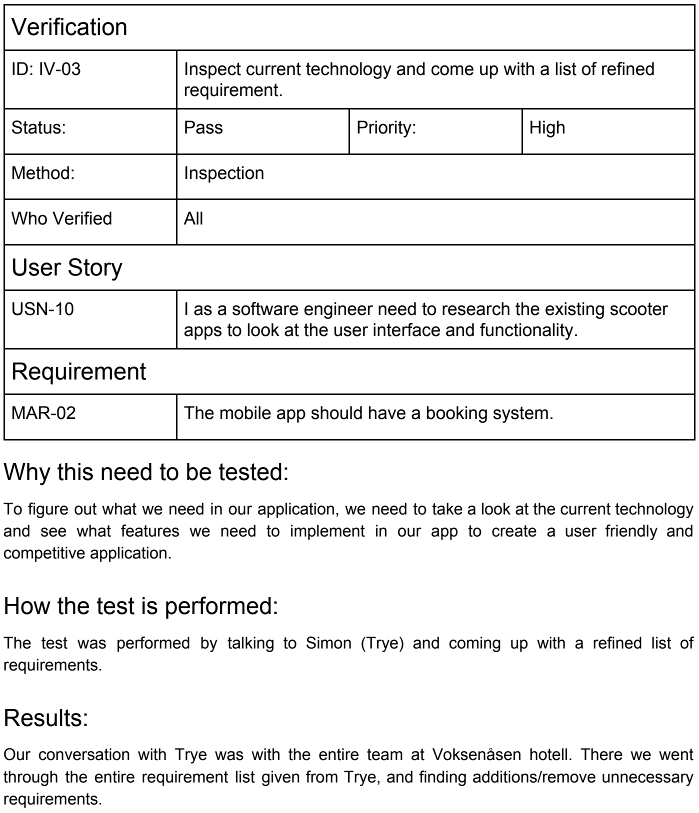
Results:

After a couple of weeks of developing and going through a Planet9 tutorial we are very happy with Planet9 and strongly believe it will become a powerful tool to help us develop a user friendly application.

Conclusion:

Decided on Planet9 to be our go-to development platform.

A.7.7 IV-03 Verifying USN-10



Verification			
ID: IV-03	Inspect current technology and come up with a list of refined requirement.		
Status:	Pass	Priority:	High
Method:	Inspection		
Who Verified	All		
User Story			
USN-10	I as a software engineer need to research the existing scooter apps to look at the user interface and functionality.		
Requirement			
MAR-02	The mobile app should have a booking system.		

Why this need to be tested:

To figure out what we need in our application, we need to take a look at the current technology and see what features we need to implement in our app to create a user friendly and competitive application.

How the test is performed:

The test was performed by talking to Simon (Trye) and coming up with a refined list of requirements.

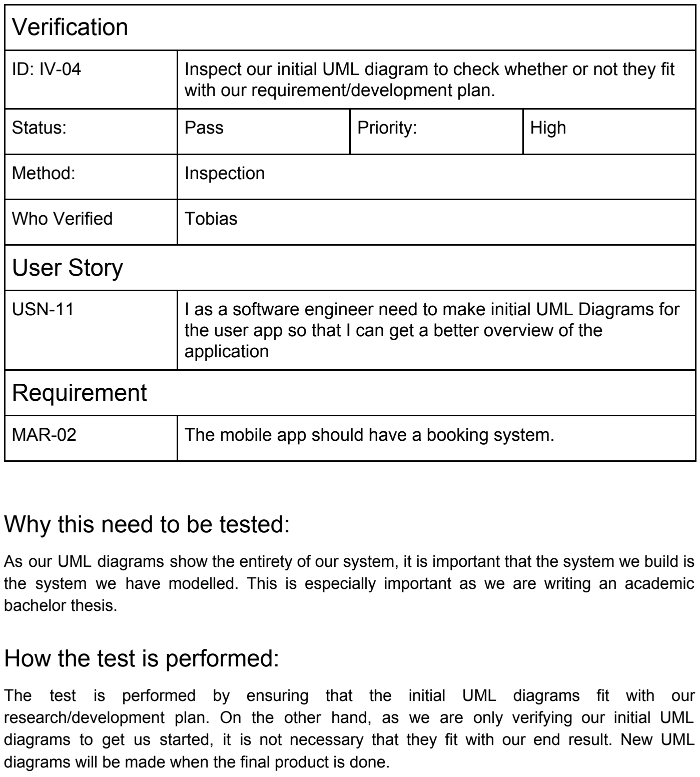
Results:

Our conversation with Trye was with the entire team at Voksenåsen hotell. There we went through the entire requirement list given from Trye, and finding additions/remove unnecessary requirements.

Conclusion:

The talk went as we hoped, and a new refined list of requirements were created. This requirement stack was what we worked with throughout the entire development cycle.

A.7.8 IV-04 Verifying USN-11



Verification			
ID: IV-04	Inspect our initial UML diagram to check whether or not they fit with our requirement/development plan.		
Status:	Pass	Priority:	High
Method:	Inspection		
Who Verified	Tobias		
User Story			
USN-11	I as a software engineer need to make initial UML Diagrams for the user app so that I can get a better overview of the application		
Requirement			
MAR-02	The mobile app should have a booking system.		

Why this need to be tested:

As our UML diagrams show the entirety of our system, it is important that the system we build is the system we have modelled. This is especially important as we are writing an academic bachelor thesis.

How the test is performed:

The test is performed by ensuring that the initial UML diagrams fit with our research/development plan. On the other hand, as we are only verifying our initial UML diagrams to get us started, it is not necessary that they fit with our end result. New UML diagrams will be made when the final product is done.

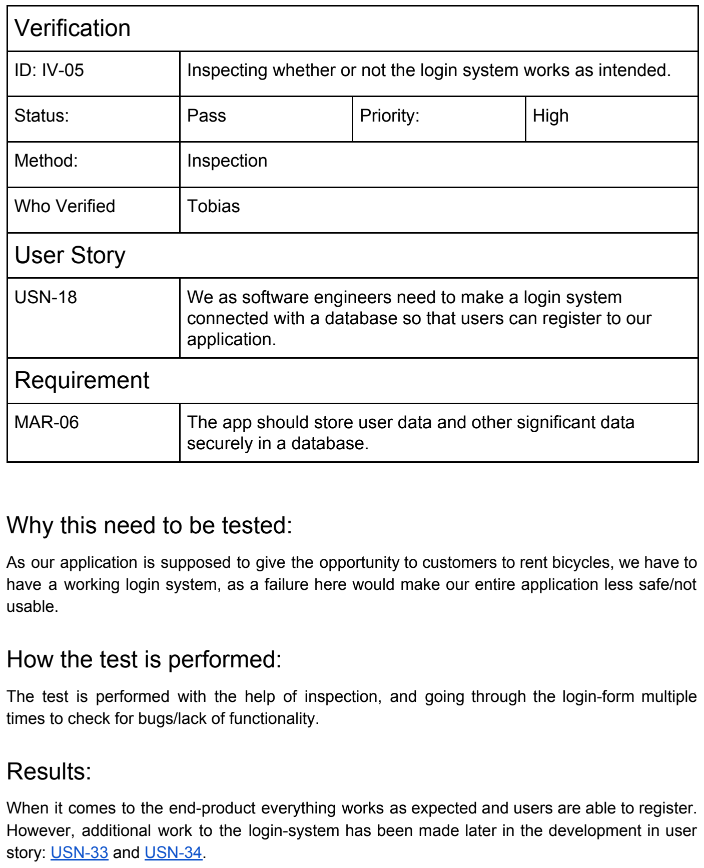
Results:

Our initial diagrams fit with the requirement/development plan made in USN-10, with corresponding verification document IV-03 where we went over the requirement/development plan with Simon.

Conclusion:

As there is a clear correspondence between the initial UML diagrams produced and the requirements/development plan this gives a pass.

A.7.9 IV-05 Verifying USN-18



Verification			
ID: IV-05	Inspecting whether or not the login system works as intended.		
Status:	Pass	Priority:	High
Method:	Inspection		
Who Verified	Tobias		
User Story			
USN-18	We as software engineers need to make a login system connected with a database so that users can register to our application.		
Requirement			
MAR-06	The app should store user data and other significant data securely in a database.		

Why this need to be tested:

As our application is supposed to give the opportunity to customers to rent bicycles, we have to have a working login system, as a failure here would make our entire application less safe/not usable.

How the test is performed:

The test is performed with the help of inspection, and going through the login-form multiple times to check for bugs/lack of functionality.

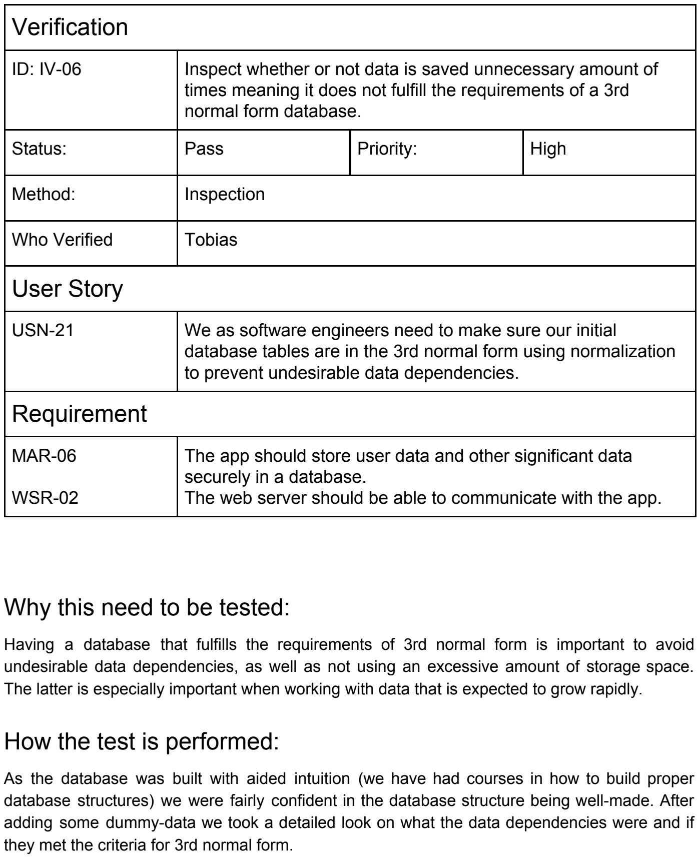
Results:

When it comes to the end-product everything works as expected and users are able to register. However, additional work to the login-system has been made later in the development in user story: USN-33 and USN-34.

Conclusion:

The final result is a working login system that saves user data the way we wanted, and that gives this a pass in the verification.

A.7.10 IV-06 Verifying USN-21



Verification			
ID: IV-06	Inspect whether or not data is saved unnecessary amount of times meaning it does not fulfill the requirements of a 3rd normal form database.		
Status:	Pass	Priority:	High
Method:	Inspection		
Who Verified	Tobias		
User Story			
USN-21	We as software engineers need to make sure our initial database tables are in the 3rd normal form using normalization to prevent undesirable data dependencies.		
Requirement			
MAR-06 WSR-02	The app should store user data and other significant data securely in a database. The web server should be able to communicate with the app.		

Why this need to be tested:

Having a database that fulfills the requirements of 3rd normal form is important to avoid undesirable data dependencies, as well as not using an excessive amount of storage space. The latter is especially important when working with data that is expected to grow rapidly.

How the test is performed:

As the database was built with aided intuition (we have had courses in how to build proper database structures) we were fairly confident in the database structure being well-made. After adding some dummy-data we took a detailed look on what the data dependencies were and if they met the criteria for 3rd normal form.

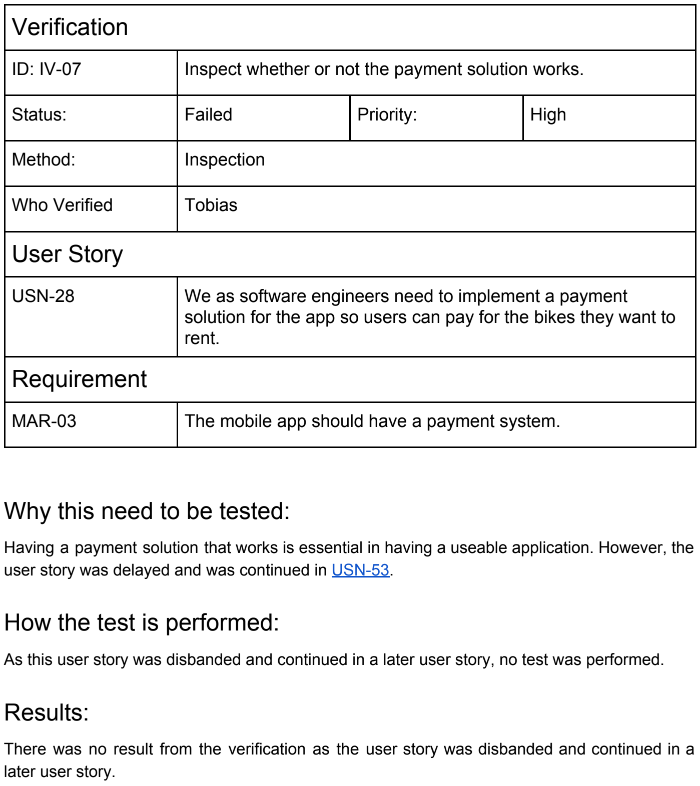
Results:

When examining the database and its data we saw that there were no data dependencies that were not made purposely, meanwhile the database was still usable for all our uses in the application.

Conclusion:

From our examination we concluded with the database structure being properly made and this gives a pass in this verification.

A.7.11 IV-07 Verifying USN-28



Verification			
ID: IV-07	Inspect whether or not the payment solution works.		
Status:	Failed	Priority:	High
Method:	Inspection		
Who Verified	Tobias		
User Story			
USN-28	We as software engineers need to implement a payment solution for the app so users can pay for the bikes they want to rent.		
Requirement			
MAR-03	The mobile app should have a payment system.		

Why this need to be tested:

Having a payment solution that works is essential in having a useable application. However, the user story was delayed and was continued in USN-53.

How the test is performed:

As this user story was disbanded and continued in a later user story, no test was performed.

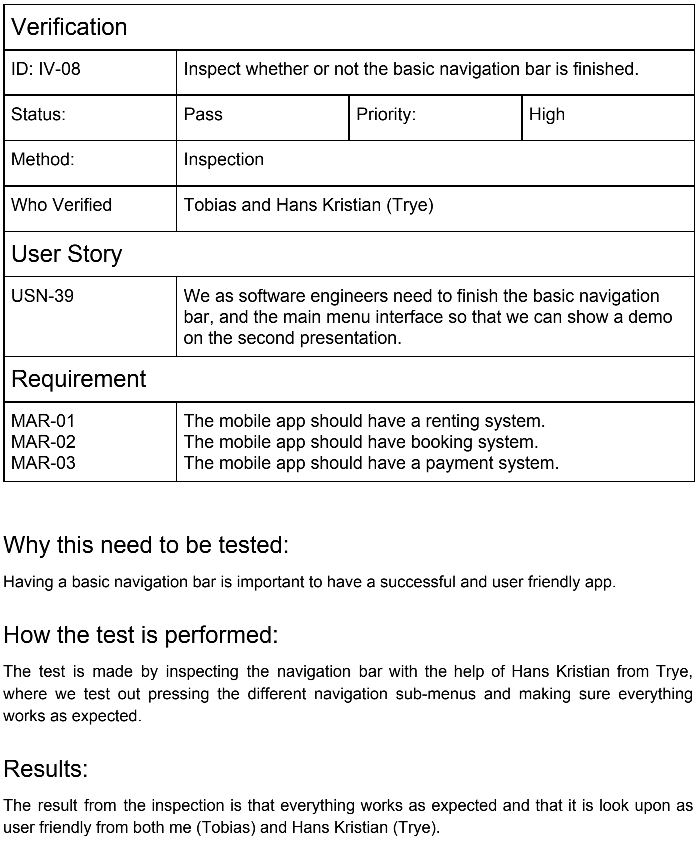
Results:

There was no result from the verification as the user story was disbanded and continued in a later user story.

Conclusion:

Continued in a later user story.

A.7.12 IV-08 Verifying USN-30



Verification			
ID: IV-08	Inspect whether or not the basic navigation bar is finished.		
Status:	Pass	Priority:	High
Method:	Inspection		
Who Verified	Tobias and Hans Kristian (Trye)		
User Story			
USN-39	We as software engineers need to finish the basic navigation bar, and the main menu interface so that we can show a demo on the second presentation.		
Requirement			
MAR-01 MAR-02 MAR-03	The mobile app should have a renting system. The mobile app should have booking system. The mobile app should have a payment system.		

Why this need to be tested:

Having a basic navigation bar is important to have a successful and user friendly app.

How the test is performed:

The test is made by inspecting the navigation bar with the help of Hans Kristian from Trye, where we test out pressing the different navigation sub-menus and making sure everything works as expected.

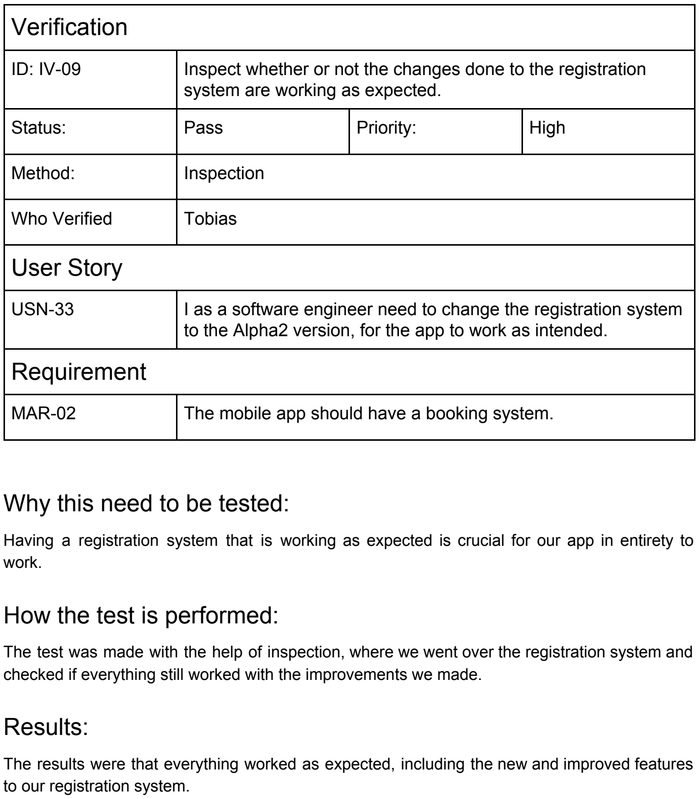
Results:

The result from the inspection is that everything works as expected and that it is look upon as user friendly from both me (Tobias) and Hans Kristian (Trye).

Conclusion:

As everything worked in the inspection we can conclude with the basic navigation bar being done and that gives the verification a pass.

A.7.13 IV-09 Verifying USN-33



Verification			
ID: IV-09	Inspect whether or not the changes done to the registration system are working as expected.		
Status:	Pass	Priority:	High
Method:	Inspection		
Who Verified	Tobias		
User Story			
USN-33	I as a software engineer need to change the registration system to the Alpha2 version, for the app to work as intended.		
Requirement			
MAR-02	The mobile app should have a booking system.		

Why this need to be tested:

Having a registration system that is working as expected is crucial for our app in entirety to work.

How the test is performed:

The test was made with the help of inspection, where we went over the registration system and checked if everything still worked with the improvements we made.

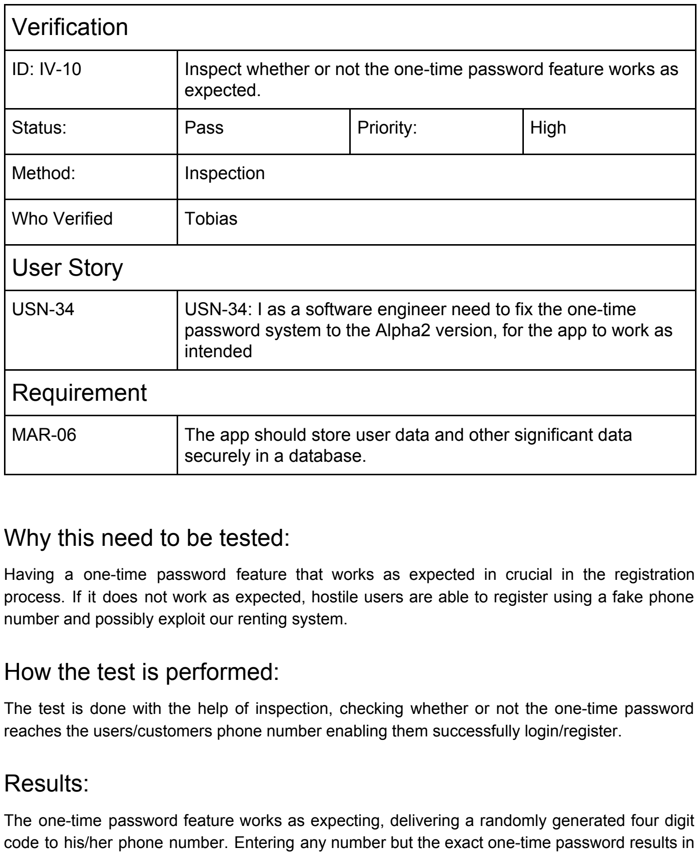
Results:

The results were that everything worked as expected, including the new and improved features to our registration system.

Conclusion:

As everything worked as expected, this gives a pass in the verification.

A.7.14 IV-10 Verifying USN-34



Verification			
ID: IV-10	Inspect whether or not the one-time password feature works as expected.		
Status:	Pass	Priority:	High
Method:	Inspection		
Who Verified	Tobias		
User Story			
USN-34	USN-34: I as a software engineer need to fix the one-time password system to the Alpha2 version, for the app to work as intended		
Requirement			
MAR-06	The app should store user data and other significant data securely in a database.		

Why this need to be tested:

Having a one-time password feature that works as expected in crucial in the registration process. If it does not work as expected, hostile users are able to register using a fake phone number and possibly exploit our renting system.

How the test is performed:

The test is done with the help of inspection, checking whether or not the one-time password reaches the users/customers phone number enabling them successfully login/register.

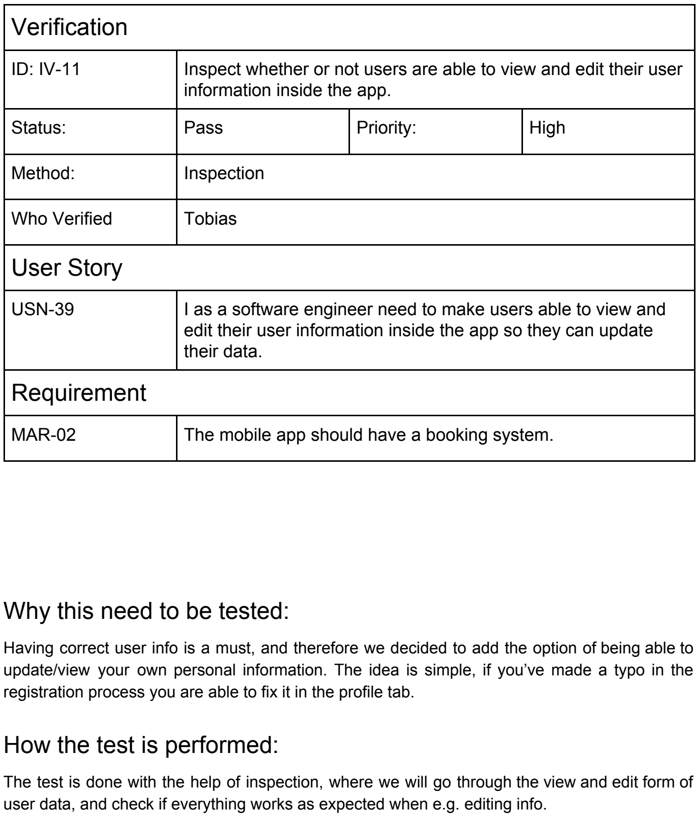
Results:

The one-time password feature works as expecting, delivering a randomly generated four digit code to his/her phone number. Entering any number but the exact one-time password results in a “wrong code” prompt, meanwhile entering the delivered one-time password lets the user register as planned.

Conclusion:

As the one-time password feature works as intended, this gives a pass in the verification.

A.7.15 IV-11 Verifying USN-39



Verification			
ID: IV-11	Inspect whether or not users are able to view and edit their user information inside the app.		
Status:	Pass	Priority:	High
Method:	Inspection		
Who Verified	Tobias		
User Story			
USN-39	I as a software engineer need to make users able to view and edit their user information inside the app so they can update their data.		
Requirement			
MAR-02	The mobile app should have a booking system.		

Why this need to be tested:

Having correct user info is a must, and therefore we decided to add the option of being able to update/view your own personal information. The idea is simple, if you've made a typo in the registration process you are able to fix it in the profile tab.

How the test is performed:

The test is done with the help of inspection, where we will go through the view and edit form of user data, and check if everything works as expected when e.g. editing info.

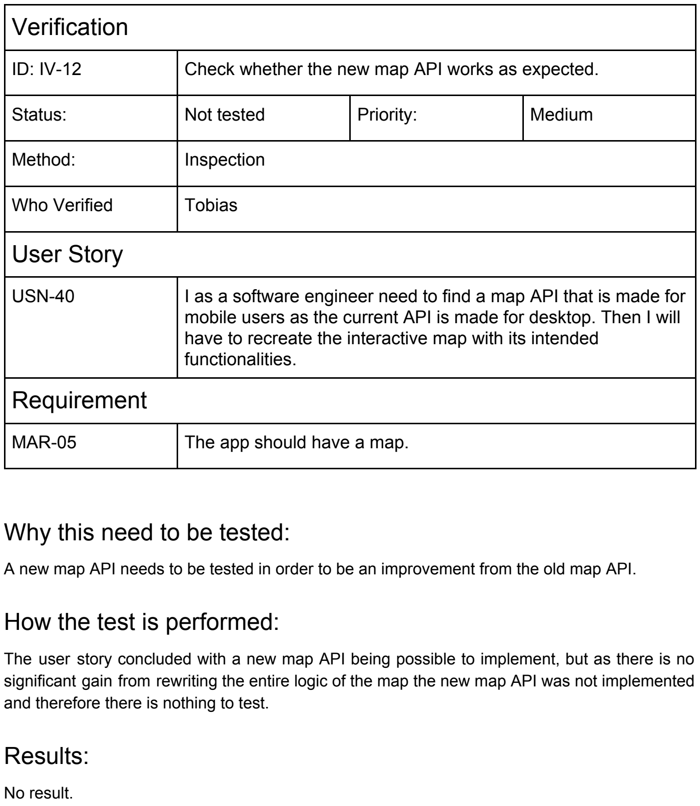
Results:

The results from the test were that everything worked smoothly, and that a user could view and update their information as expected.

Conclusion:

As the user is able to view and edit their personal info, this gives a pass in the verification.

A.7.16 IV-12 Verifying USN-40



Verification			
ID: IV-12	Check whether the new map API works as expected.		
Status:	Not tested	Priority:	Medium
Method:	Inspection		
Who Verified	Tobias		
User Story			
USN-40	I as a software engineer need to find a map API that is made for mobile users as the current API is made for desktop. Then I will have to recreate the interactive map with its intended functionalities.		
Requirement			
MAR-05	The app should have a map.		

Why this need to be tested:

A new map API needs to be tested in order to be an improvement from the old map API.

How the test is performed:

The user story concluded with a new map API being possible to implement, but as there is no significant gain from rewriting the entire logic of the map the new map API was not implemented and therefore there is nothing to test.

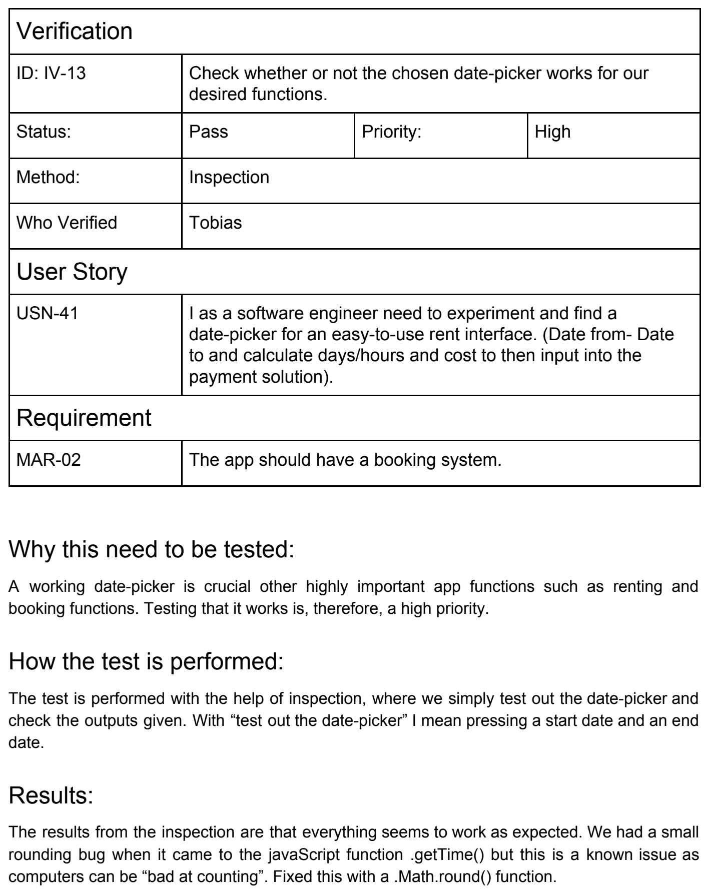
Results:

No result.

Conclusion:

No conclusion.

A.7.17 IV-13 Verifying USN-41



Verification			
ID: IV-13	Check whether or not the chosen date-picker works for our desired functions.		
Status:	Pass	Priority:	High
Method:	Inspection		
Who Verified	Tobias		
User Story			
USN-41	I as a software engineer need to experiment and find a date-picker for an easy-to-use rent interface. (Date from- Date to and calculate days/hours and cost to then input into the payment solution).		
Requirement			
MAR-02	The app should have a booking system.		

Why this need to be tested:

A working date-picker is crucial other highly important app functions such as renting and booking functions. Testing that it works is, therefore, a high priority.

How the test is performed:

The test is performed with the help of inspection, where we simply test out the date-picker and check the outputs given. With “test out the date-picker” I mean pressing a start date and an end date.

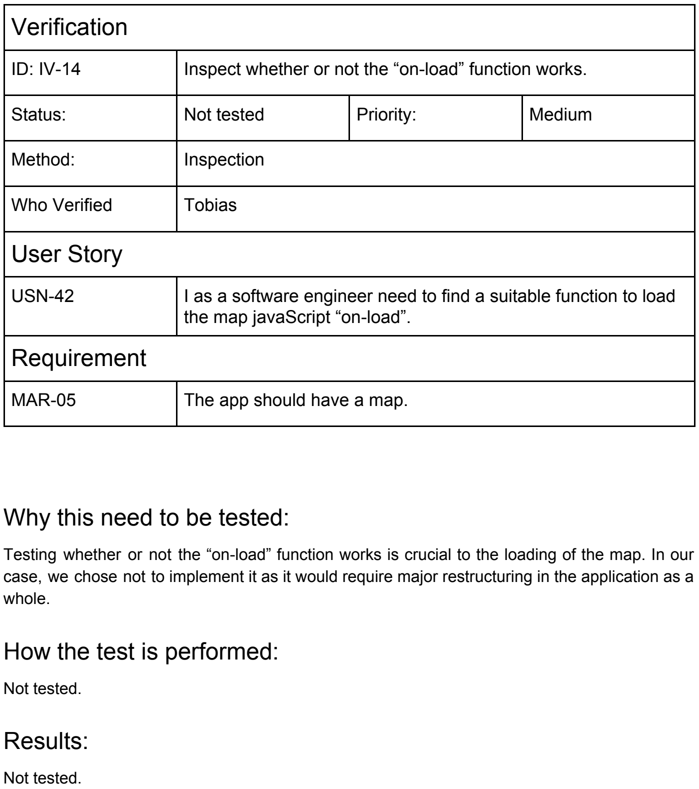
Results:

The results from the inspection are that everything seems to work as expected. We had a small rounding bug when it came to the javaScript function .getTime() but this is a known issue as computers can be “bad at counting”. Fixed this with a .Math.round() function.

Conclusion:

As everything works as expected, this gives a pass in the verification.

A.7.18 IV-14 Verifying USN-42



Verification			
ID: IV-14	Inspect whether or not the “on-load” function works.		
Status:	Not tested	Priority:	Medium
Method:	Inspection		
Who Verified	Tobias		
User Story			
USN-42	I as a software engineer need to find a suitable function to load the map javaScript “on-load”.		
Requirement			
MAR-05	The app should have a map.		

Why this need to be tested:

Testing whether or not the “on-load” function works is crucial to the loading of the map. In our case, we chose not to implement it as it would require major restructuring in the application as a whole.

How the test is performed:

Not tested.

Results:

Not tested.

Conclusion:

Not tested.

A.7.19 IV-15 Verifying USN-46

Verification			
ID: IV-15	Check whether or not the bookings tab work as expected.		
Status:	Pass	Priority:	High
Method:	Inspection		
Who Verified	Tobias		
User Story			
USN-46	I as a software engineer need to display bikes in the bookings tab so they can be selected and put in booking table so users can book bikes.		
Requirement			
Function-04	The system should Enable bike users to book a bike using mobile app.		

Why this need to be tested:

If no bikes are shown in the bookings tab, it is impossible for a user to “select” a bike. This makes it impossible for bookings to be done in the app, so therefore it is extremely important that the bikes are shown.

How the test is performed:

The test will be done with the help of inspection, where we will enter the bookings tab of the application and check if everything works as expected, and the bikes are shown and can be selected.

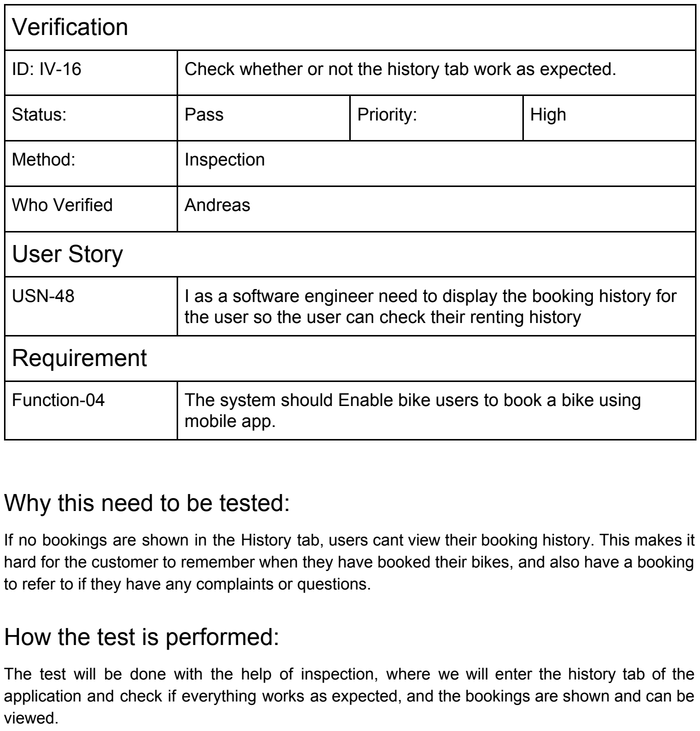
Results:

The bikes are shown and can be selected. When selected a new “your selected bike” pops up, making it easy for a user to see what bike he/she selected.

Conclusion:

As the wanted functionality is fulfilled and the bikes are shown and can be selected this gives a pass in the verification.

A.7.20 IV-16 Verifying USN-48



Verification			
ID: IV-16	Check whether or not the history tab work as expected.		
Status:	Pass	Priority:	High
Method:	Inspection		
Who Verified	Andreas		
User Story			
USN-48	I as a software engineer need to display the booking history for the user so the user can check their renting history		
Requirement			
Function-04	The system should Enable bike users to book a bike using mobile app.		

Why this need to be tested:

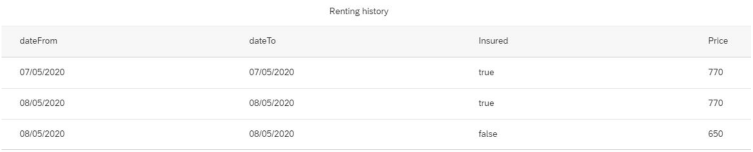
If no bookings are shown in the History tab, users cant view their booking history. This makes it hard for the customer to remember when they have booked their bikes, and also have a booking to refer to if they have any complaints or questions.

How the test is performed:

The test will be done with the help of inspection, where we will enter the history tab of the application and check if everything works as expected, and the bookings are shown and can be viewed.

Results:

The bookings are shown with a booking id and all the needed information related to that user:



Renting history

dateFrom	dateTo	Insured	Price
07/05/2020	07/05/2020	true	770
08/05/2020	08/05/2020	true	770
08/05/2020	08/05/2020	false	650

Conclusion:

As the wanted functionality is fulfilled and the bookings are shown with all needed information which gives a pass in the verification.

A.7.21 IV-17 Verifying USN-49

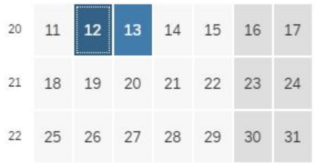
Verification			
ID: IV-17	Check whether or not the bookings tab work as expected.		
Status:	Pass	Priority:	High
Method:	Inspection		
Who Verified	Andreas		
User Story			
USN-49	I as a software engineer need to display the booking history for the user so the user can check their renting history		
Requirement			
Function-04	The system should Enable bike users to book a bike using mobile app.		

Why this need to be tested:

When users book bikes we want to test two main criteria, no overlap and no booking back in time.

How the test is performed:

The test is done by trying to overlap bookings and bookings back in time.

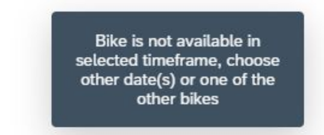


20	11	12	13	14	15	16	17
21	18	19	20	21	22	23	24
22	25	26	27	28	29	30	31

We make a booking long ahead in time and try to book in the same range.

Results:

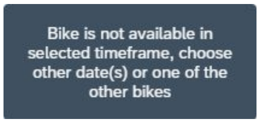
Book the same dates:



Bike is not available in selected timeframe, choose other date(s) or one of the other bikes

Success as trying to book only on the 12th or the 13th gives an error.

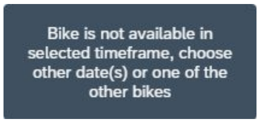
Overlapping bookings:



Bike is not available in selected timeframe, choose other date(s) or one of the other bikes

Success as trying to book 11-12th and 13-14th both fails.

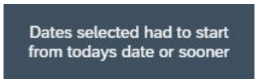
Booking the same timeframe:



Bike is not available in selected timeframe, choose other date(s) or one of the other bikes

Success is trying to book the same timeframe 12-13th fails.

Making a booking back in time:

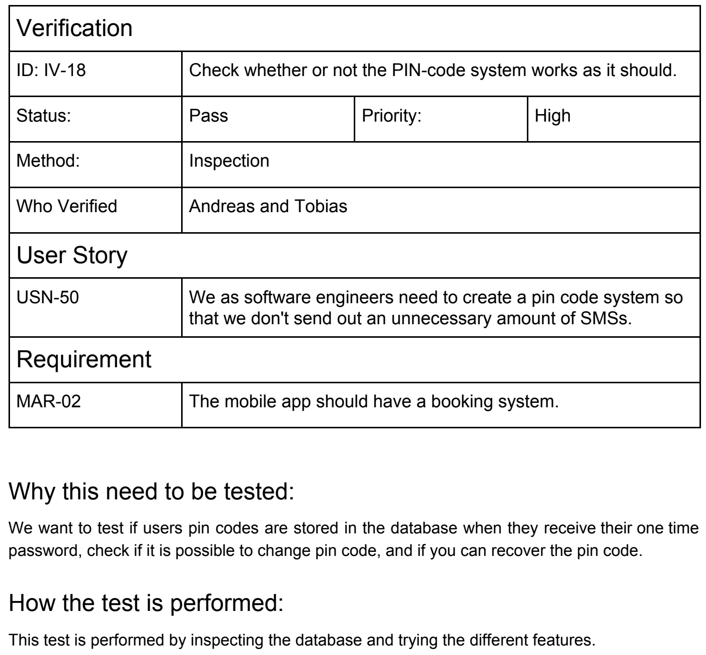

Dates selected had to start from todays date or sooner

Success as trying to book on 1 April 2020 gives an error message since it is date back time.

Conclusion:

As the wanted functionality is fulfilled and no customer can overlap each other's bookings which gives a pass in the verification.

A.7.22 IV-18 Verifying USN-50



Verification			
ID: IV-18	Check whether or not the PIN-code system works as it should.		
Status:	Pass	Priority:	High
Method:	Inspection		
Who Verified	Andreas and Tobias		
User Story			
USN-50	We as software engineers need to create a pin code system so that we don't send out an unnecessary amount of SMSs.		
Requirement			
MAR-02	The mobile app should have a booking system.		

Why this need to be tested:

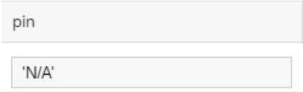
We want to test if users pin codes are stored in the database when they receive their one time password, check if it is possible to change pin code, and if you can recover the pin code.

How the test is performed:

This test is performed by inspecting the database and trying the different features.

Results:

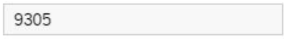
Testing updating the pin to the one time password:
We start by inspecting the database:



pin

'N/A'

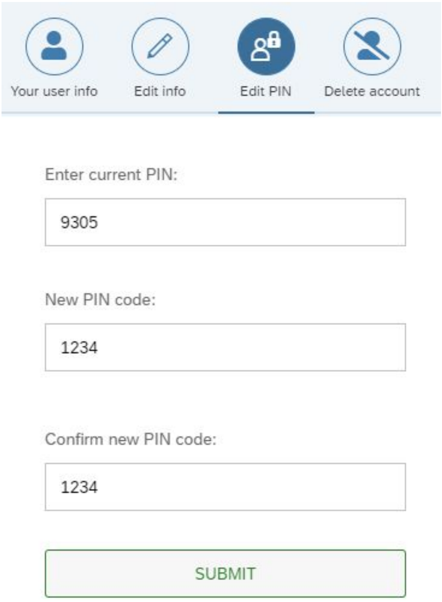
We can see that the pin is the default. When we register and enter the one time password we want it to change to that. The code we got 9305. Now ve verify the code and check the database:



9305

We can now see that the value has changed to the one time password which means this test is passed.

Testing if we can change it inside the app:



Enter current PIN:

9305

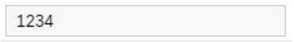
New PIN code:

1234

Confirm new PIN code:

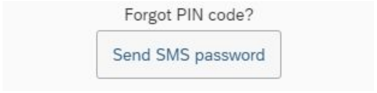
1234

We change the pin code to “1234” to check if it updates in the database:

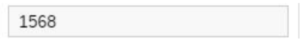

1234

We can see now that the pin code has changed to “1234” in the database which means the test is passed.

Now we try to recover an account if a user has forgotten their password:



“1568” is the code I received. Now to check the database:

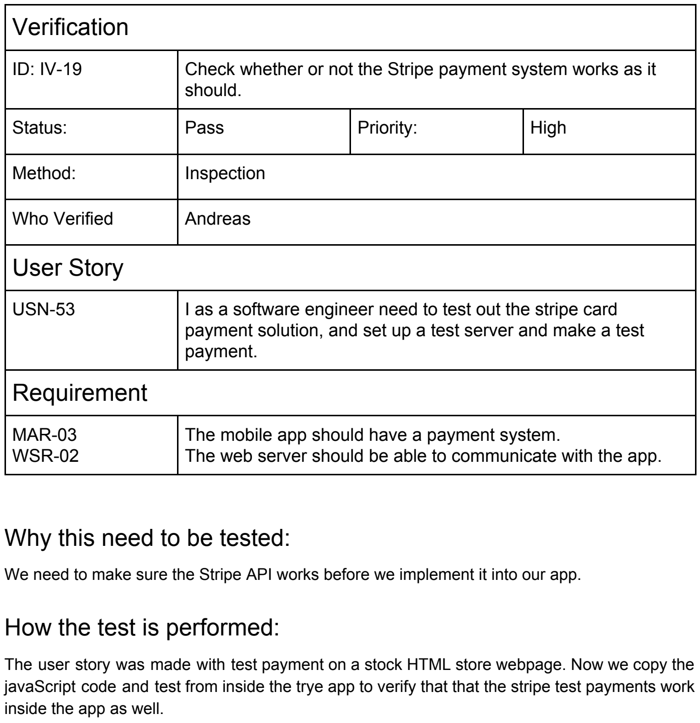

1568

The code has been updated so the recovery is successful.

Conclusion:

The pin code system works as intended which is confirmed by inspection so the test is passed.

A.7.23 IV-19 Verifying USN-53



Verification			
ID: IV-19	Check whether or not the Stripe payment system works as it should.		
Status:	Pass	Priority:	High
Method:	Inspection		
Who Verified	Andreas		
User Story			
USN-53	I as a software engineer need to test out the stripe card payment solution, and set up a test server and make a test payment.		
Requirement			
MAR-03 WSR-02	The mobile app should have a payment system. The web server should be able to communicate with the app.		

Why this need to be tested:

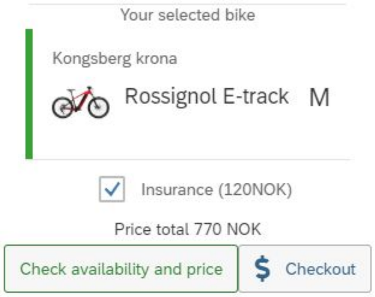
We need to make sure the Stripe API works before we implement it into our app.

How the test is performed:

The user story was made with test payment on a stock HTML store webpage. Now we copy the javaScript code and test from inside the trye app to verify that that the stripe test payments work inside the app as well.

Results:

Testing the integration with a test payment:



Your selected bike

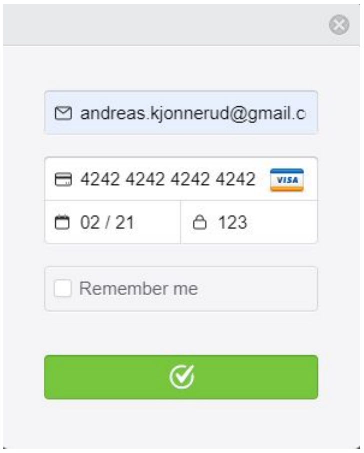
Kongsberg krona

Rossignol E-track M

☑ Insurance (120NOK)

Price total 770 NOK

Now one bike is selected with insurance and we press checkout to pay:



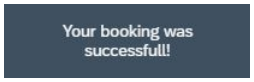
andreas.kjonnerud@gmail.c

4242 4242 4242 4242

02 / 21

123

□ Remember me

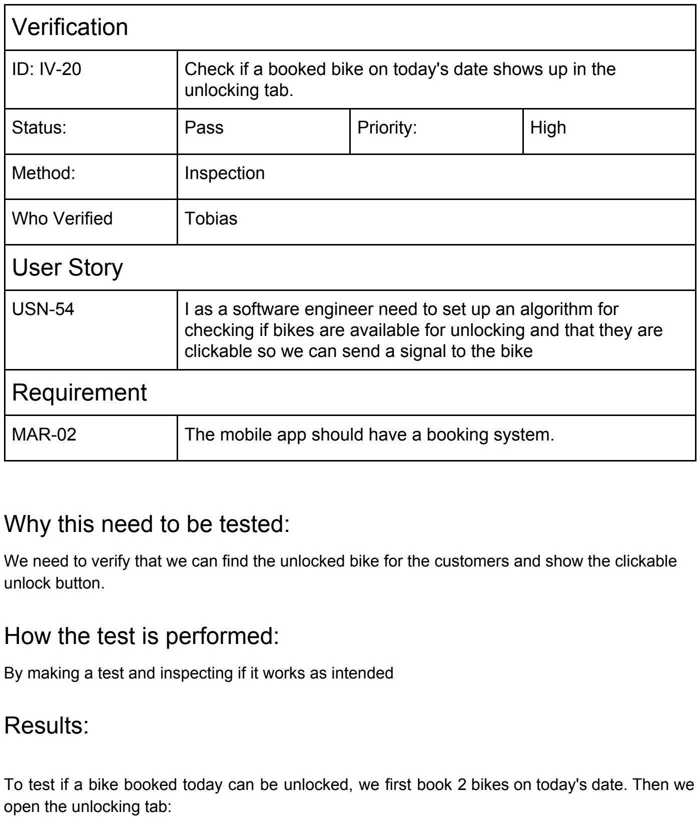

Your booking was successfull!

The booking was successful.

Conclusion:

The implementation worked and the inspection test is passed.

A.7.24 IV-20 Verifying USN-54



Verification			
ID: IV-20	Check if a booked bike on today's date shows up in the unlocking tab.		
Status:	Pass	Priority:	High
Method:	Inspection		
Who Verified	Tobias		
User Story			
USN-54	I as a software engineer need to set up an algorithm for checking if bikes are available for unlocking and that they are clickable so we can send a signal to the bike		
Requirement			
MAR-02	The mobile app should have a booking system.		

Why this need to be tested:

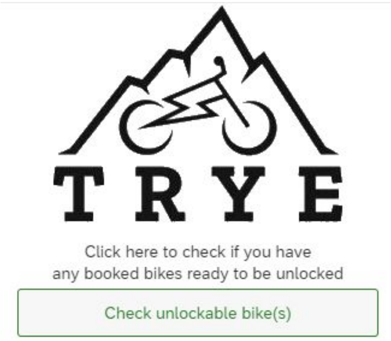
We need to verify that we can find the unlocked bike for the customers and show the clickable unlock button.

How the test is performed:

By making a test and inspecting if it works as intended

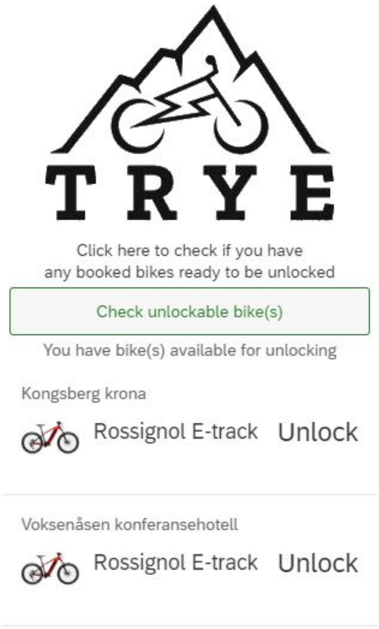
Results:

To test if a bike booked today can be unlocked, we first book 2 bikes on today's date. Then we open the unlocking tab:



Click here to check if you have
any booked bikes ready to be unlocked

Now we click the button the 2 bikes should show up:



Click here to check if you have
any booked bikes ready to be unlocked

You have bike(s) available for unlocking

Kongsberg krona

Rossignol E-track

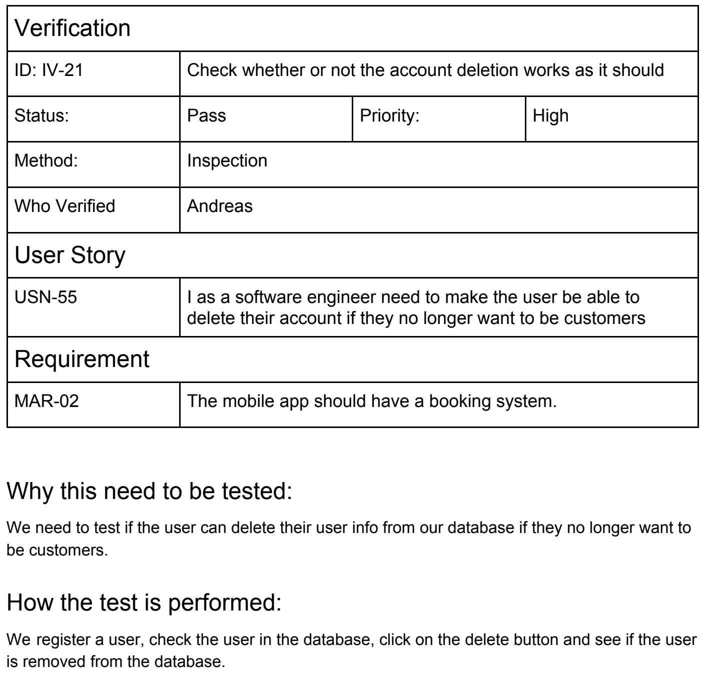
Voksenåsen konferansehotell

Rossignol E-track

Conclusion:

The check for unlockable bikes function works and the test is passed.

A.7.25 IV-21 Verifying USN-55



Verification			
ID: IV-21	Check whether or not the account deletion works as it should		
Status:	Pass	Priority:	High
Method:	Inspection		
Who Verified	Andreas		
User Story			
USN-55	I as a software engineer need to make the user be able to delete their account if they no longer want to be customers		
Requirement			
MAR-02	The mobile app should have a booking system.		

Why this need to be tested:

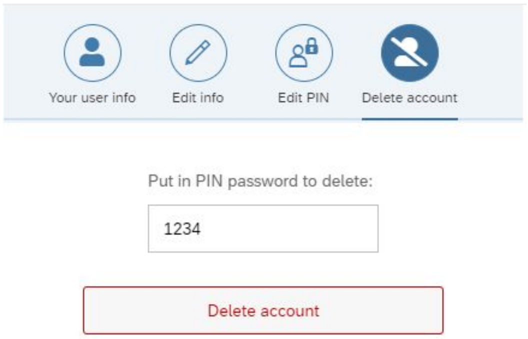
We need to test if the user can delete their user info from our database if they no longer want to be customers.

How the test is performed:

We register a user, check the user in the database, click on the delete button and see if the user is removed from the database.

Results:

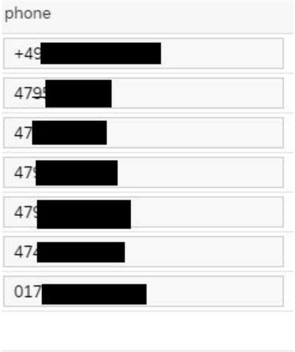
Opening the user tab and entering the pin code and clicking delete:



Put in PIN password to delete:

1234

Now the database browser is opened and “update” is pressed to see if the user has been removed:



phone
+49[redacted]
479[redacted]
47[redacted]
47[redacted]
47[redacted]
47[redacted]
017[redacted]

We can see the user is gone.

Conclusion:

The test was successful and we can conclude the test with a pass.

A.7.26 IV-22 Verifying USN-57

Verification			
ID: IV-22	Check whether or not the Admin app works as intended		
Status:	Pass	Priority:	High
Method:	Inspection		
Who Verified	Andreas		
User Story			
USN-57	I as a software engineer need to make user be able to delete their account if they no longer want to be customers		
Requirement			
ASR-01 ASR-03	The Admin System should be able to control the lock on the bike The Admin System should be able to see who is using the bikes		

Why this need to be tested:

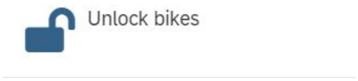
We need to verify that the features in the admin as is working.

How the test is performed:

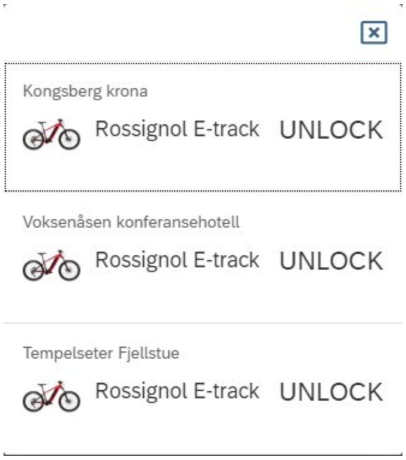
We inspect the different buttons and see if they work as intended.

Results:

Testing the unlocking button:



When we click unlock all the bikes should be listed with a unlock button:



Kongsberg krona

Rossignol E-track UNLOCK

Voksenåsen konferansehotell

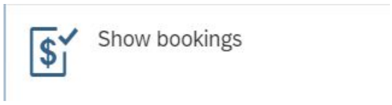
Rossignol E-track UNLOCK

Tempelseter Fjellstue

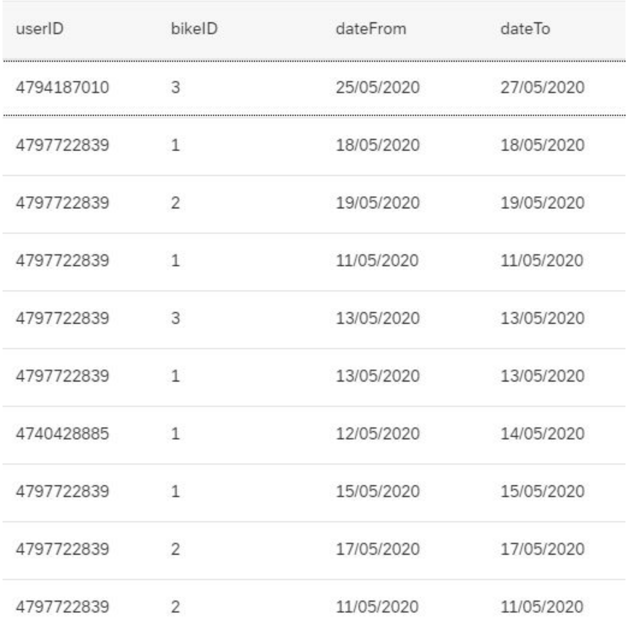
Rossignol E-track UNLOCK

The bikes are listed with the unlock button which means the test was a success.

Testing the bookings button:



When we click a list of all bikes should show:



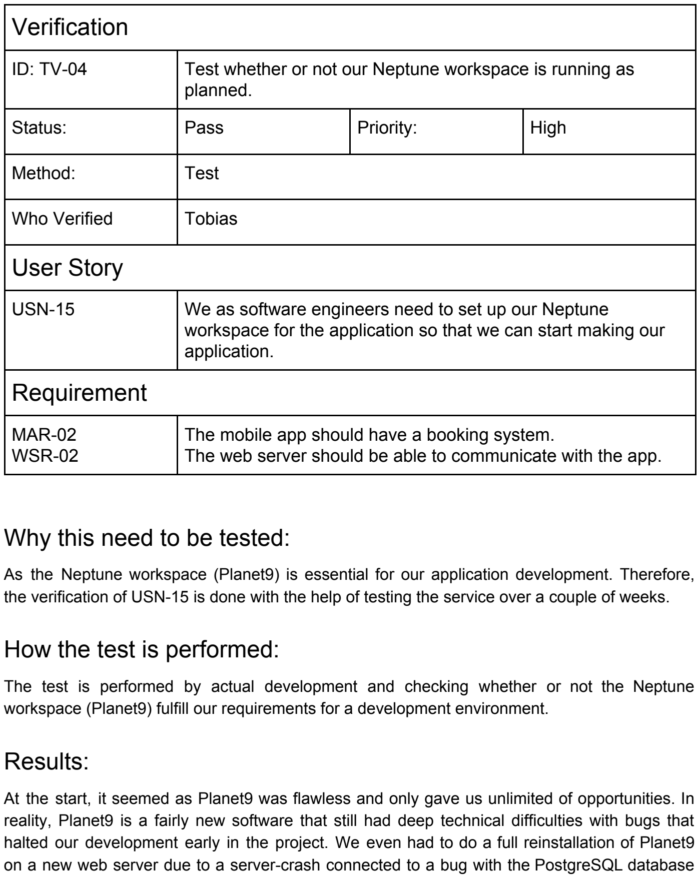
userID	bikeID	dateFrom	dateTo
4794187010	3	25/05/2020	27/05/2020
4797722839	1	18/05/2020	18/05/2020
4797722839	2	19/05/2020	19/05/2020
4797722839	1	11/05/2020	11/05/2020
4797722839	3	13/05/2020	13/05/2020
4797722839	1	13/05/2020	13/05/2020
4740428885	1	12/05/2020	14/05/2020
4797722839	1	15/05/2020	15/05/2020
4797722839	2	17/05/2020	17/05/2020
4797722839	2	11/05/2020	11/05/2020

The list shows so test is passed.

Conclusion:

All tests passed so we can conclude the test as passed.

A.7.27 TV-04 Verifying USN-15



Verification			
ID: TV-04	Test whether or not our Neptune workspace is running as planned.		
Status:	Pass	Priority:	High
Method:	Test		
Who Verified	Tobias		
User Story			
USN-15	We as software engineers need to set up our Neptune workspace for the application so that we can start making our application.		
Requirement			
MAR-02 WSR-02	The mobile app should have a booking system. The web server should be able to communicate with the app.		

Why this need to be tested:

As the Neptune workspace (Planet9) is essential for our application development. Therefore, the verification of USN-15 is done with the help of testing the service over a couple of weeks.

How the test is performed:

The test is performed by actual development and checking whether or not the Neptune workspace (Planet9) fulfill our requirements for a development environment.

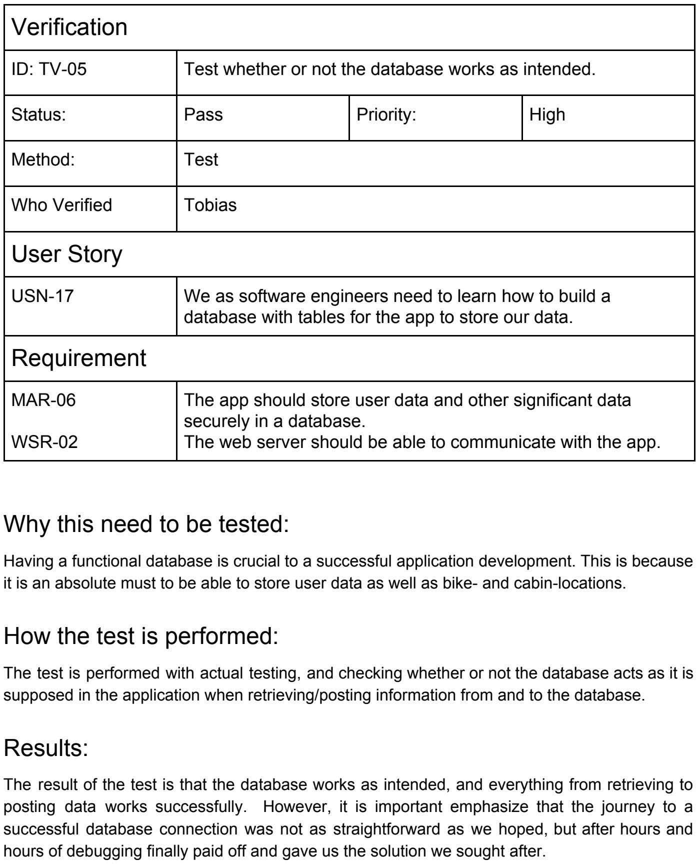
Results:

At the start, it seemed as Planet9 was flawless and only gave us unlimited of opportunities. In reality, Planet9 is a fairly new software that still had deep technical difficulties with bugs that halted our development early in the project. We even had to do a full reinstallation of Planet9 on a new web server due to a server-crash connected to a bug with the PostgreSQL database that caused the entire platform to not work. Luckily, an update of Planet9 came with a more functional way of connecting to databases which resolves the issues we encountered.

Conclusion:

In the end, we were able to set up Planet9 as we wanted and connect it to our PostgreSQL making it possible to store information as well as develop our application. This gives this user story a pass in the verification.

A.7.28 TV-05 Verifying USN-17



Verification			
ID: TV-05	Test whether or not the database works as intended.		
Status:	Pass	Priority:	High
Method:	Test		
Who Verified	Tobias		
User Story			
USN-17	We as software engineers need to learn how to build a database with tables for the app to store our data.		
Requirement			
MAR-06 WSR-02	The app should store user data and other significant data securely in a database. The web server should be able to communicate with the app.		

Why this need to be tested:

Having a functional database is crucial to a successful application development. This is because it is an absolute must to be able to store user data as well as bike- and cabin-locations.

How the test is performed:

The test is performed with actual testing, and checking whether or not the database acts as it is supposed in the application when retrieving/posting information from and to the database.

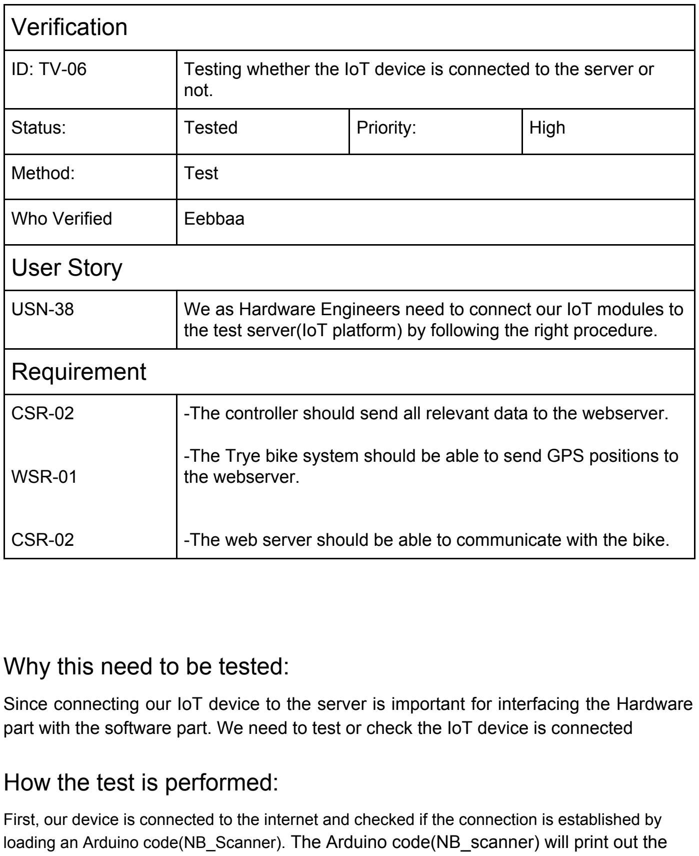
Results:

The result of the test is that the database works as intended, and everything from retrieving to posting data works successfully. However, it is important emphasize that the journey to a successful database connection was not as straightforward as we hoped, but after hours and hours of debugging finally paid off and gave us the solution we sought after.

Conclusion:

The status of the verification is a pass as everything database-related works.

A.7.29 TV-06 Verifying USN-38



Verification			
ID: TV-06	Testing whether the IoT device is connected to the server or not.		
Status:	Tested	Priority:	High
Method:	Test		
Who Verified	Eebbaa		
User Story			
USN-38	We as Hardware Engineers need to connect our IoT modules to the test server(IoT platform) by following the right procedure.		
Requirement			
CSR-02 WSR-01 CSR-02	-The controller should send all relevant data to the webserver. -The Trye bike system should be able to send GPS positions to the webserver. -The web server should be able to communicate with the bike.		

Why this need to be tested:

Since connecting our IoT device to the server is important for interfacing the Hardware part with the software part. We need to test or check the IoT device is connected

How the test is performed:

First, our device is connected to the internet and checked if the connection is established by loading an Arduino code(NB_Scanner). The Arduino code(NB_scanner) will print out the

IMEI number and scans for nearby networks and gives us the signal strength of the network between 0 and 31 where 0 is minimum signal strength while 31 is maximum.

To test the connection between Arduino and the test server, the Arduino code (Azure_IoT_Hub_NB) is loaded to send some data and at the server-side, we check the log file using azure cloud shell to see the message sent from the Arduino board.

Results:

1. The device is connected to the internet and the NB_Scanner runs successfully and prints the result on the serial monitor, the result we got is shown in figure 1.
2. The message is received at the server-side and can be viewed as a log file on the Azure web server. The result for this test is shown in Figure-2

Conclusion:

The device is connected with the internet through the Telenor network and the connection between the IoT device and Server is accomplished.

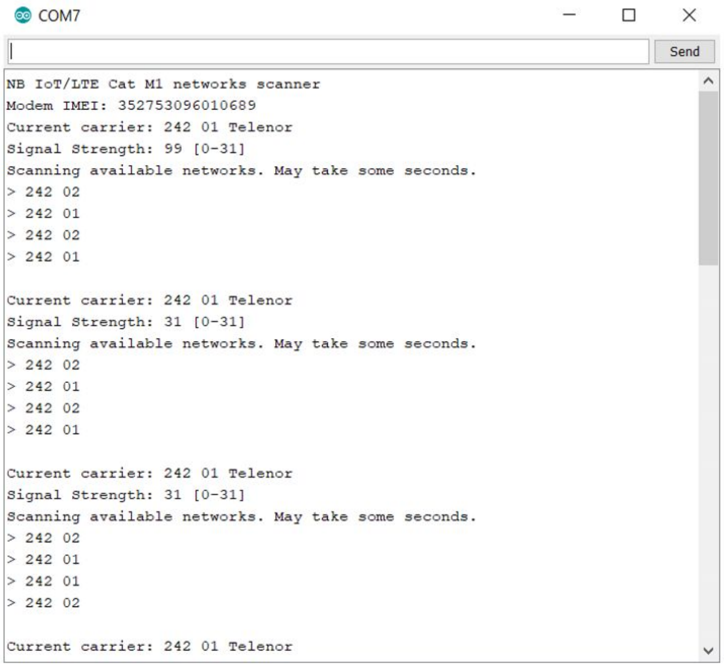
Code used
NB_Scanner

1. https://drive.google.com/drive/folders/1Qd-Z840bCUuJrs0ObDkwxYfx0r_iEniy

Azure_IoT_Hub_NB

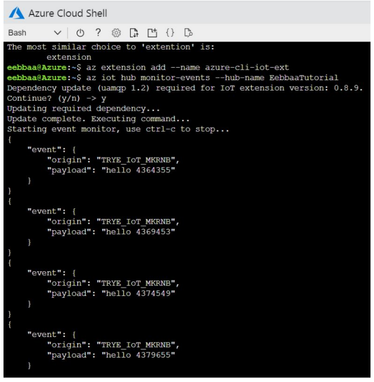
2. https://drive.google.com/drive/folders/1GgG-_XZtXSTbwa60ttNaXqPwTh9BgxPX

Figure:



```
NB IoT/LTE Cat M1 networks scanner
Modem IMEI: 352753096010689
Current carrier: 242 01 Telenor
Signal Strength: 99 [0-31]
Scanning available networks. May take some seconds.
> 242 02
> 242 01
> 242 02
> 242 01

Current carrier: 242 01 Telenor
Signal Strength: 31 [0-31]
Scanning available networks. May take some seconds.
> 242 02
> 242 01
> 242 02
> 242 01

Current carrier: 242 01 Telenor
Signal Strength: 31 [0-31]
Scanning available networks. May take some seconds.
> 242 02
> 242 01
> 242 01
> 242 02

Current carrier: 242 01 Telenor
```

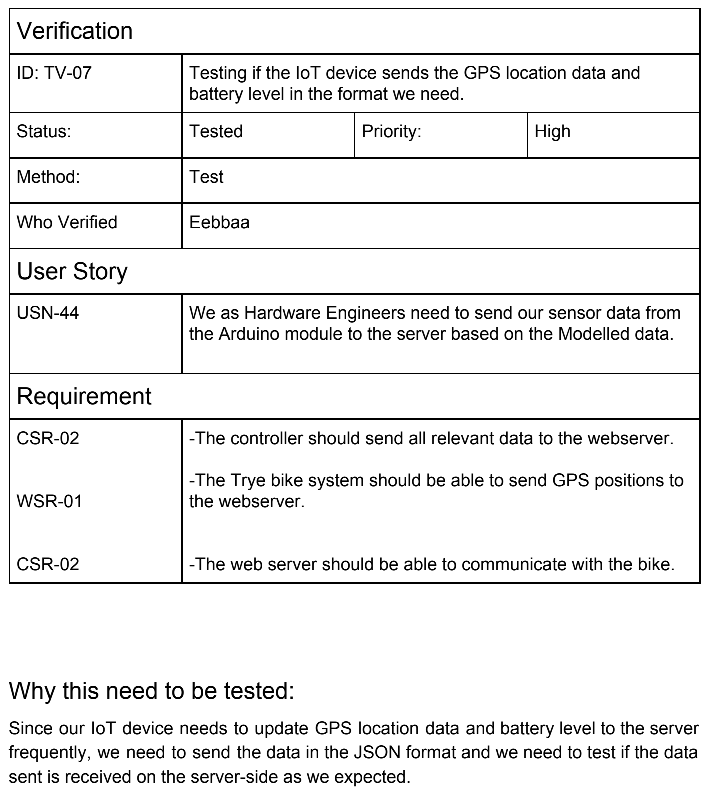
Figure-1: Serial Monitor result by running Arduino code NB_scanner



```
The most similar choice to 'extention' is:
        extension
eebbaa@Azure:~$ az extension add --name azure-cli-iot-ext
eebbaa@Azure:~$ az iot hub monitor-events --hub-name EebbaaTutorial
Dependency update (uamqp 1.2) required for IoT extension version: 0.8.9.
Continue? (y/n) -> y
Updating required dependency...
Update complete. Executing command...
Starting event monitor, use ctrl-c to stop...
{
    "event": {
        "origin": "TRYE_IoT_MKRNB",
        "payload": "hello 4364355"
    }
}
{
    "event": {
        "origin": "TRYE_IoT_MKRNB",
        "payload": "hello 4369453"
    }
}
{
    "event": {
        "origin": "TRYE_IoT_MKRNB",
        "payload": "hello 4374549"
    }
}
{
    "event": {
        "origin": "TRYE_IoT_MKRNB",
        "payload": "hello 4379655"
    }
```

Figure 2: Log file on the Azure cloud server which shows a message sent from an IoT device.

A.7.30 TV-07 Verifying USN-44



Verification			
ID: TV-07	Testing if the IoT device sends the GPS location data and battery level in the format we need.		
Status:	Tested	Priority:	High
Method:	Test		
Who Verified	Eebbaa		
User Story			
USN-44	We as Hardware Engineers need to send our sensor data from the Arduino module to the server based on the Modelled data.		
Requirement			
CSR-02 WSR-01 CSR-02	-The controller should send all relevant data to the webserver. -The Trye bike system should be able to send GPS positions to the webserver. -The web server should be able to communicate with the bike.		

Why this need to be tested:

Since our IoT device needs to update GPS location data and battery level to the server frequently, we need to send the data in the JSON format and we need to test if the data sent is received on the server-side as we expected.

How the test is performed:

By sending the data in the format we need and observe the data received at the server-side in the log file. The function *void publishMessage(**int** bikeID, **String** lon, **String** lat, **int** batteryLevel)* will be used to send the data in the format we need.

Results:

The message is received at the server-side and can be viewed as a log file on the Azure web server. The result of this test is shown in Figure-1.

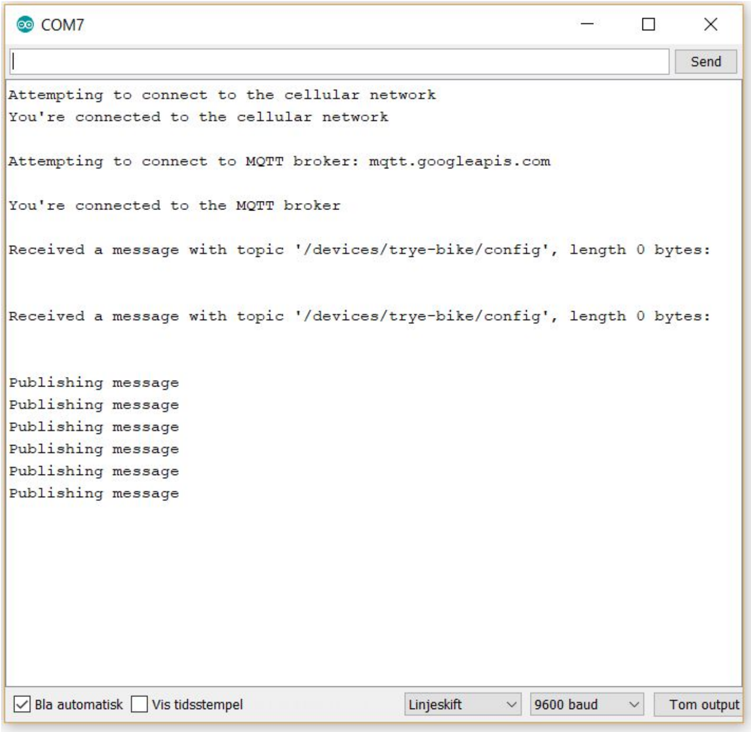
Conclusion:

The data sent from the device in the JSON format will be sent to the MQTT server on the cloud. The JSON format makes the data easier to categorize later in the Database table with their message title.

Code used

1. https://drive.google.com/drive/folders/1kHnc5LB7wuhquW9LOMd-XPQ7TeYaA3Um

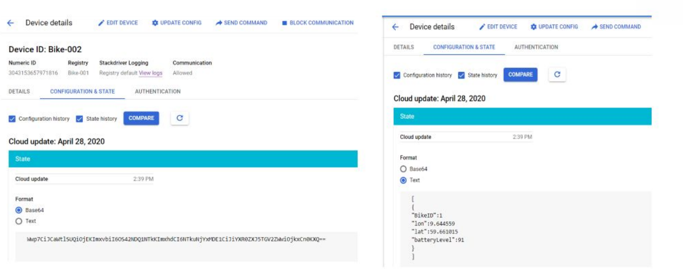
Figures:



```
Attempting to connect to the cellular network
You're connected to the cellular network

Attempting to connect to MQTT broker: mqtt.googleapis.com

You're connected to the MQTT broker

Received a message with topic '/devices/trye-bike/config', length 0 bytes:


Received a message with topic '/devices/trye-bike/config', length 0 bytes:


Publishing message
Publishing message
Publishing message
Publishing message
Publishing message
Publishing message
```

☑ Bla automatisk □ Vis tidsstempel Linjeskift 9600 baud Tom output

Figure-1 Serial Monitor showing the IoT device is publishing Message to the server after connecting to the network.



Device ID: Bike-002

Numeric ID	Registry	Stackdriver Logging	Communication
3043153657971816	Bike-001	Registry default View logs	Allowed

☑ Configuration history ☑ State history

Cloud update: April 28, 2020

Cloud update 2:39 PM

Format
● Base64
○ Text

Wwp7CiJCaWtlSUQiOjEKImxvbiI6OS42NDQ1NTkKImxhdCI6NTkuNjYxMDE1CiJiYXR0ZXJ5TGV2ZWwiOjkxCn0KXQ==

☑ Configuration history ☑ State history

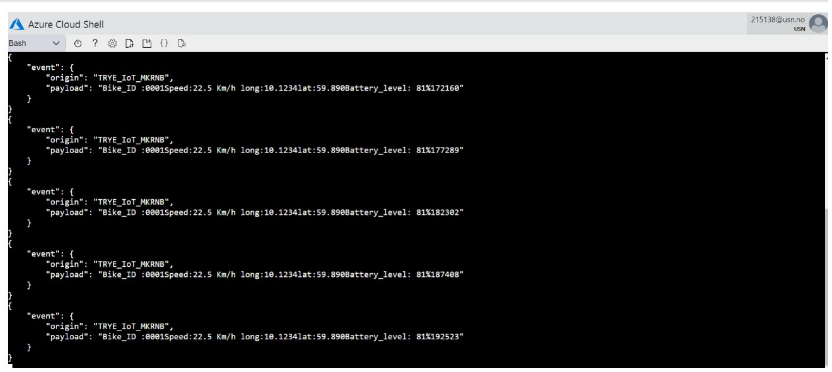
Cloud update: April 28, 2020

Cloud update 2:39 PM

Format
○ Base64
● Text

```
[
{
"BikeID":1
"lon":9.644559
"lat":59.661015
"batteryLevel":91
}
]
```

Figure-2: Message sent from an IoT device is received at the Google cloud platform server.



```
{
    "event": {
        "origin": "TRYE_IoT_MKRNB",
        "payload": "Bike_ID :0001Speed:22.5 Km/h long:10.1234lat:59.890Battery_level: 81%172160"
    }
}
{
    "event": {
        "origin": "TRYE_IoT_MKRNB",
        "payload": "Bike_ID :0001Speed:22.5 Km/h long:10.1234lat:59.890Battery_level: 81%177289"
    }
}
{
    "event": {
        "origin": "TRYE_IoT_MKRNB",
        "payload": "Bike_ID :0001Speed:22.5 Km/h long:10.1234lat:59.890Battery_level: 81%182302"
    }
}
{
    "event": {
        "origin": "TRYE_IoT_MKRNB",
        "payload": "Bike_ID :0001Speed:22.5 Km/h long:10.1234lat:59.890Battery_level: 81%187408"
    }
}
{
    "event": {
        "origin": "TRYE_IoT_MKRNB",
        "payload": "Bike_ID :0001Speed:22.5 Km/h long:10.1234lat:59.890Battery_level: 81%192523"
    }
}
```

Figure-3: Message sent from an IoT device is received at the Azure cloud platform.

A.7.31 TV-08 Verifying USN-35

Verification			
ID: TV-08	Check if data is formatted as it should on server-side, and device-side		
Status:	Pass	Priority:	Medium
Method:	Test		
Who Verified	Joachim		
User Story			
USN-35	We as hardware engineers need to model the data sent from the Arduino to the webserver so we are able to send data in a safe, reliable and efficient way		
Requirement			
WSR-01	The web server should be able to communicate with the bike		

Why this need to be tested:

As we decided to use JSON as a standard format for our communication, it's important to check to see if it's formated as it should be when the server receives the message. This is to avoid miscommunication or loss of data

How the test is performed:

To test the data transferred we used the Arduino to send data to the server, this data was then processed with cloud function, which means we can use python to test the data. Python has a library which parses the JSON data to an array, and if this is done successfully, the data is in the correct format.

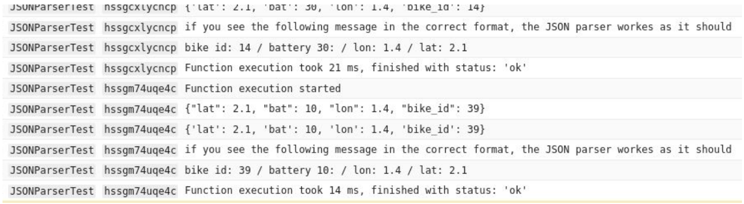
Results:

The result can be seen in FIG-01. here we can see print lines "if you see the following message in the correct format, the JSON parser works as it should" followed by the correctly formated message “`bike id: 39 / battery 10: / lon: 1.4 / lat: 2.1`”. Which is formated as expected, whict means we are able to extract the information from the JSON parser and store it as a dictionary (array)

Conclusion:

We are able to send the data as it should be sent, using JSON as the format

Figure:



```
JSONParserTest  hssgcxlycncp {'lat': 2.1, 'bat': 30, 'lon': 1.4, 'bike_id': 14}
JSONParserTest  hssgcxlycncp if you see the following message in the correct format, the JSON parser workes as it should
JSONParserTest  hssgcxlycncp bike id: 14 / battery 30: / lon: 1.4 / lat: 2.1
JSONParserTest  hssgcxlycncp Function execution took 21 ms, finished with status: 'ok'
JSONParserTest  hssgm74uqe4c Function execution started
JSONParserTest  hssgm74uqe4c {"lat": 2.1, "bat": 10, "lon": 1.4, "bike_id": 39}
JSONParserTest  hssgm74uqe4c {'lat': 2.1, 'bat': 10, 'lon': 1.4, 'bike_id': 39}
JSONParserTest  hssgm74uqe4c if you see the following message in the correct format, the JSON parser workes as it should
JSONParserTest  hssgm74uqe4c bike id: 39 / battery 10: / lon: 1.4 / lat: 2.1
JSONParserTest  hssgm74uqe4c Function execution took 14 ms, finished with status: 'ok'
```

FIG-01: shows the result from the test

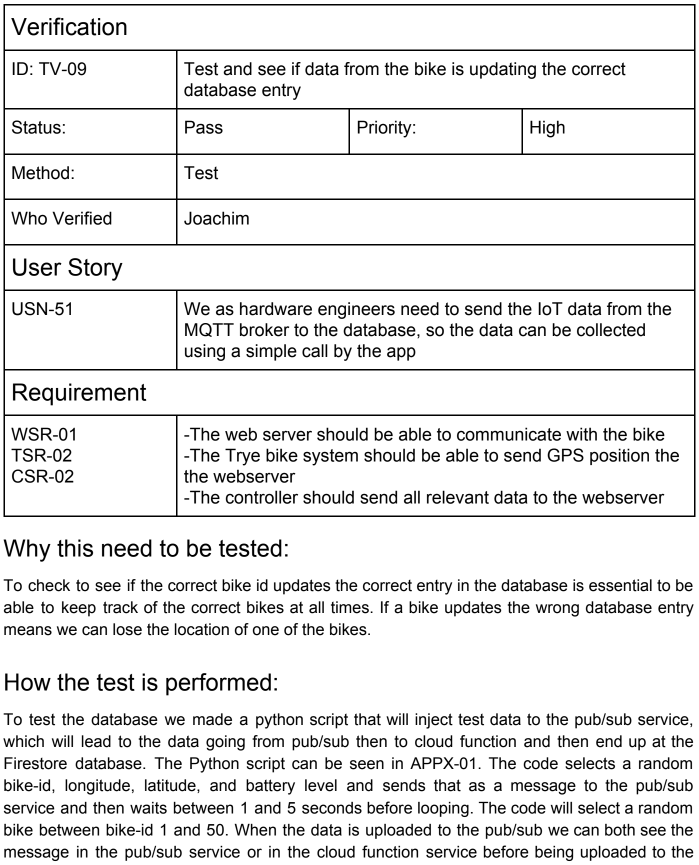
Literatur:

Appendix:

APPX-01: shows the code in GCP, used to perform the test

```
import base64
import json

def JSONParser(event, context):
    """Triggered from a message on a Cloud Pub/Sub topic.
    Args:
         event (dict): Event payload.
         context (google.cloud.functions.Context): Metadata for the event.
    """
    pubsub_message = base64.b64decode(event['data']).decode('utf-8')
    print(pubsub_message)
    eventDict = json.loads(pubsub_message)

    print(eventDict)
    print("if you see the following message in the correct format, the
JSON parser workes as it should")
    print("bike id: {} / battery {}: / lon: {} / lat:
{}".format(eventDict['bike_id'], eventDict['bat'], eventDict['lon'],
eventDict['lat']))
```

A.7.32 TV-09 Verifying USN-51



Verification			
ID: TV-09	Test and see if data from the bike is updating the correct database entry		
Status:	Pass	Priority:	High
Method:	Test		
Who Verified	Joachim		
User Story			
USN-51	We as hardware engineers need to send the IoT data from the MQTT broker to the database, so the data can be collected using a simple call by the app		
Requirement			
WSR-01 TSR-02 CSR-02	-The web server should be able to communicate with the bike -The Trye bike system should be able to send GPS position the the webserver -The controller should send all relevant data to the webserver		

Why this need to be tested:

To check to see if the correct bike id updates the correct entry in the database is essential to be able to keep track of the correct bikes at all times. If a bike updates the wrong database entry means we can lose the location of one of the bikes.

How the test is performed:

To test the database we made a python script that will inject test data to the pub/sub service, which will lead to the data going from pub/sub then to cloud function and then end up at the Firestore database. The Python script can be seen in APPX-01. The code selects a random bike-id, longitude, latitude, and battery level and sends that as a message to the pub/sub service and then waits between 1 and 5 seconds before looping. The code will select a random bike between bike-id 1 and 50. When the data is uploaded to the pub/sub we can both see the message in the pub/sub service or in the cloud function service before being uploaded to the database. By comparing the log file from both the python script, the cloud function service and the result in the database we can conclude if the database updates the correct entry or not.

Results:

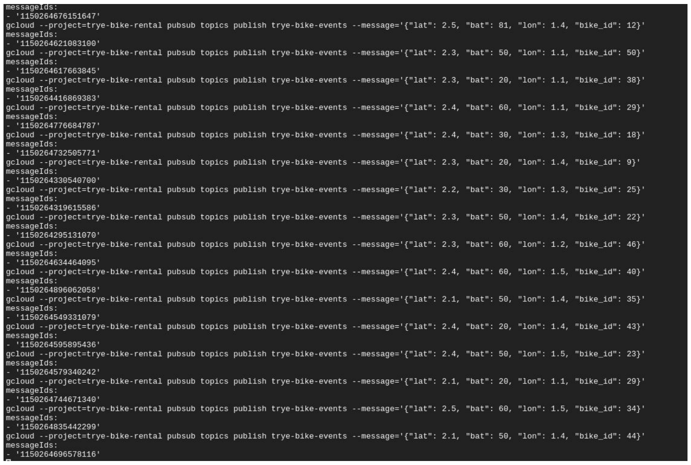
If FIG-01 we can see the data being sent from the python script. In FIG-02 we can see the log file from cloud function. In FIG-03 we can see the Firestore database. By comparing the data from the python log with the database entry we can see if the python scripts update the correct database entry. There is about a 10-20 seconds latency before the data is updated from the data is sent, this can be accounted for by looking at the log file from the cloud function, as there is a much lower latency between the cloud function and the Firestore database

Conclusion:

By comparing the result we concluded with there was no loss of data when the program was up and running as it should be, the only thing to keep in mind is the 10-20 seconds delay before the data is updated, and there is a chance the data is lost when the services have just started, but this should not be a problem in this systems use case.

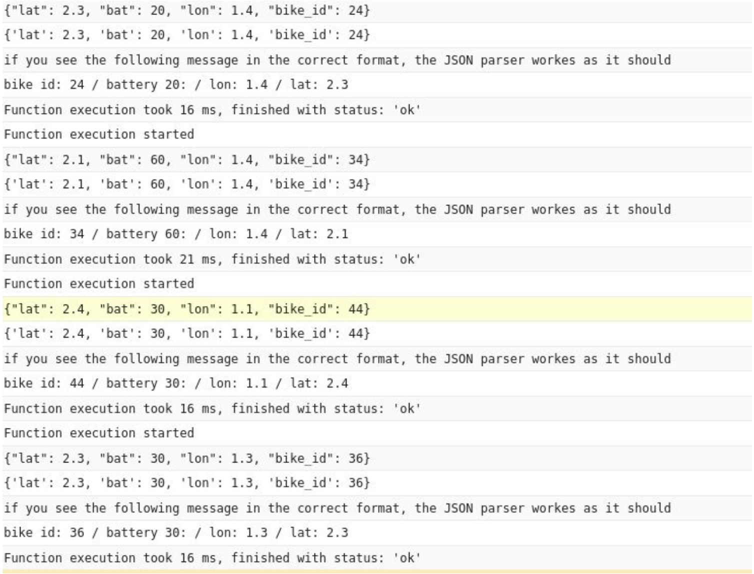
Figure:

FIG-01: the python log



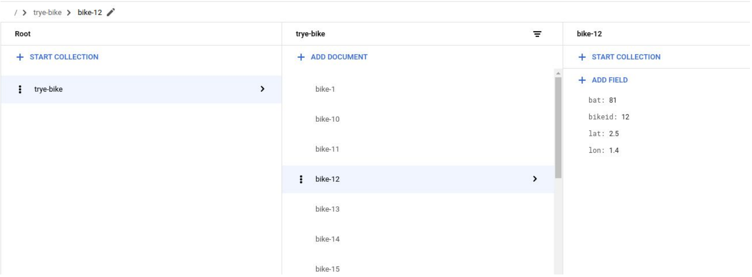
```
messageIds:
- '1150264676151647'
gcloud --project=trye-bike-rental pubsub topics publish trye-bike-events --message='{"lat": 2.5, "bat": 81, "lon": 1.4, "bike_id": 12}'
messageIds:
- '1150264621083100'
gcloud --project=trye-bike-rental pubsub topics publish trye-bike-events --message='{"lat": 2.3, "bat": 50, "lon": 1.1, "bike_id": 50}'
messageIds:
- '1150264617663845'
gcloud --project=trye-bike-rental pubsub topics publish trye-bike-events --message='{"lat": 2.3, "bat": 20, "lon": 1.1, "bike_id": 38}'
messageIds:
- '1150264416869383'
gcloud --project=trye-bike-rental pubsub topics publish trye-bike-events --message='{"lat": 2.4, "bat": 60, "lon": 1.1, "bike_id": 29}'
messageIds:
- '1150264776684787'
gcloud --project=trye-bike-rental pubsub topics publish trye-bike-events --message='{"lat": 2.4, "bat": 30, "lon": 1.3, "bike_id": 18}'
messageIds:
- '1150264732505771'
gcloud --project=trye-bike-rental pubsub topics publish trye-bike-events --message='{"lat": 2.3, "bat": 20, "lon": 1.4, "bike_id": 9}'
messageIds:
- '1150264330540700'
gcloud --project=trye-bike-rental pubsub topics publish trye-bike-events --message='{"lat": 2.2, "bat": 30, "lon": 1.3, "bike_id": 25}'
messageIds:
- '1150264319615586'
gcloud --project=trye-bike-rental pubsub topics publish trye-bike-events --message='{"lat": 2.3, "bat": 50, "lon": 1.4, "bike_id": 22}'
messageIds:
- '1150264295131070'
gcloud --project=trye-bike-rental pubsub topics publish trye-bike-events --message='{"lat": 2.3, "bat": 60, "lon": 1.2, "bike_id": 46}'
messageIds:
- '1150264634464095'
gcloud --project=trye-bike-rental pubsub topics publish trye-bike-events --message='{"lat": 2.4, "bat": 60, "lon": 1.5, "bike_id": 40}'
messageIds:
- '1150264896062058'
gcloud --project=trye-bike-rental pubsub topics publish trye-bike-events --message='{"lat": 2.1, "bat": 50, "lon": 1.4, "bike_id": 35}'
messageIds:
- '1150264549331079'
gcloud --project=trye-bike-rental pubsub topics publish trye-bike-events --message='{"lat": 2.4, "bat": 20, "lon": 1.4, "bike_id": 43}'
messageIds:
- '1150264595895436'
gcloud --project=trye-bike-rental pubsub topics publish trye-bike-events --message='{"lat": 2.4, "bat": 50, "lon": 1.5, "bike_id": 23}'
messageIds:
- '1150264579340242'
gcloud --project=trye-bike-rental pubsub topics publish trye-bike-events --message='{"lat": 2.1, "bat": 20, "lon": 1.1, "bike_id": 29}'
messageIds:
- '1150264744671340'
gcloud --project=trye-bike-rental pubsub topics publish trye-bike-events --message='{"lat": 2.5, "bat": 60, "lon": 1.5, "bike_id": 34}'
messageIds:
- '1150264835442299'
gcloud --project=trye-bike-rental pubsub topics publish trye-bike-events --message='{"lat": 2.1, "bat": 50, "lon": 1.4, "bike_id": 44}'
messageIds:
- '1150264696578116'
```

FIG-02: The cloud function log



```
{"lat": 2.3, "bat": 20, "lon": 1.4, "bike_id": 24}
{'lat': 2.3, 'bat': 20, 'lon': 1.4, 'bike_id': 24}
if you see the following message in the correct format, the JSON parser workes as it should
bike id: 24 / battery 20: / lon: 1.4 / lat: 2.3
Function execution took 16 ms, finished with status: 'ok'
Function execution started
{"lat": 2.1, "bat": 60, "lon": 1.4, "bike_id": 34}
{'lat': 2.1, 'bat': 60, 'lon': 1.4, 'bike_id': 34}
if you see the following message in the correct format, the JSON parser workes as it should
bike id: 34 / battery 60: / lon: 1.4 / lat: 2.1
Function execution took 21 ms, finished with status: 'ok'
Function execution started
{"lat": 2.4, "bat": 30, "lon": 1.1, "bike_id": 44}
{'lat': 2.4, 'bat': 30, 'lon': 1.1, 'bike_id': 44}
if you see the following message in the correct format, the JSON parser workes as it should
bike id: 44 / battery 30: / lon: 1.1 / lat: 2.4
Function execution took 16 ms, finished with status: 'ok'
Function execution started
{"lat": 2.3, "bat": 30, "lon": 1.3, "bike_id": 36}
{'lat': 2.3, 'bat': 30, 'lon': 1.3, 'bike_id': 36}
if you see the following message in the correct format, the JSON parser workes as it should
bike id: 36 / battery 30: / lon: 1.3 / lat: 2.3
Function execution took 16 ms, finished with status: 'ok'
```

FIG-03: The Firestore database



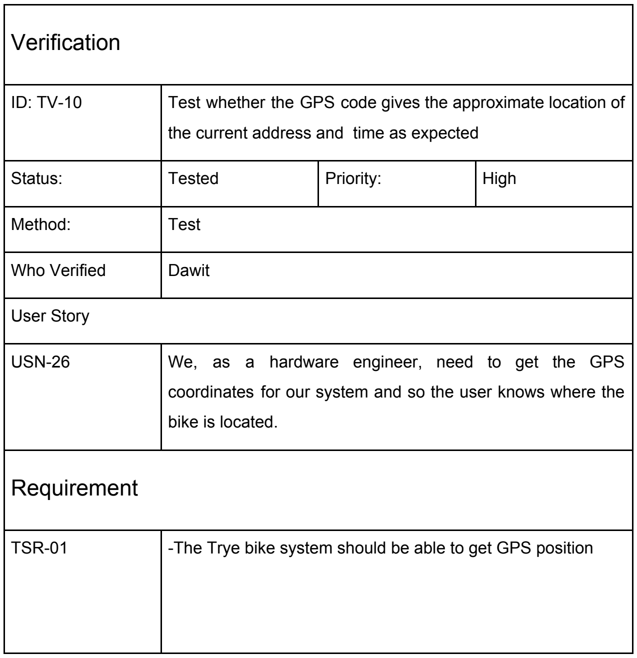
```
bat: 81
bikeid: 12
lat: 2.5
lon: 1.4
```

Appendix:

APPX-01: the python script used to inject random data into the database

```
#!/usr/bin/env python
import datetime, json, os, random, time
# Set the `project` variable to a Google Cloud project ID.
project = 'trye-bike-rental'
topic = 'trye-bike-events'
BIKE_ID = [1,2,3,4,5,6,7,8,9,]
LON = [1.1,1.2,1.3,1.4,1.5,]
LAT = [2.1,2.2,2.3,2.4,2.5,]
BAT = [10,30,20,50,60,81]
while True:
  data = {
      'bike_id': random.randint(1,50),
      'lon': random.choice(LON),
      'lat': random.choice(LAT),
      'bat': random.choice(BAT),
  }
  # For a more complete example on how to publish messages in Pub/Sub.
  #   https://cloud.google.com/pubsub/docs/publisher
  message = json.dumps(data)
  command = "gcloud --project={} pubsub topics publish {} --message='{}'".format(project,topic,
message)
  print(command)
  os.system(command)
  time.sleep(random.randrange(1, 5))
```

A.7.33 TV-10 Verifying USN-26



Verification			
ID: TV-10	Test whether the GPS code gives the approximate location of the current address and time as expected		
Status:	Tested	Priority:	High
Method:	Test		
Who Verified	Dawit		
User Story			
USN-26	We, as a hardware engineer, need to get the GPS coordinates for our system and so the user knows where the bike is located.		
Requirement			
TSR-01	-The Trye bike system should be able to get GPS position		

Why this need to be tested:

We need to test because by receiving the data from the satellites such as latitude, longitude, and time and then converting it to the Google Maps, we need to check if we approximately find the position of the current address and the time which matches Norwegian time.

How the test is performed:

By uploading the GPS coordinate code into the Arduino and interfacing it with the GPS module, we can receive the data from the satellites. The received data such as longitude, latitude, and time converted to Google Maps, and then we approximately find the position of the current address and the current time matches with the Norwegian time.

Results:

By converting the data that we got from satellites into Google Maps, we approximately found the exact position of the current address and the time. The result shows in Figures 1 and 2 below.

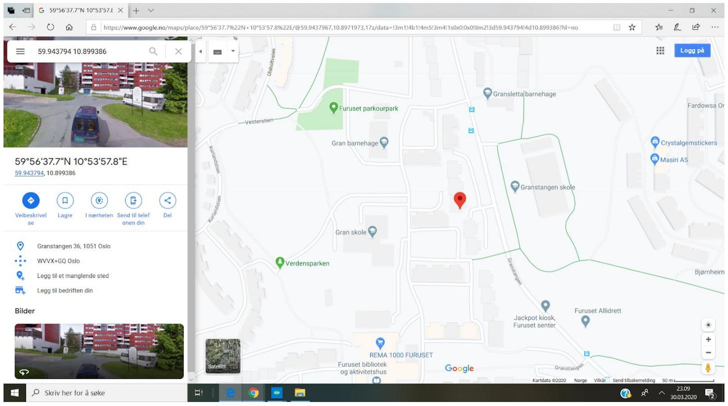
Conclusion:

The result we found in this user story helps us to use it later in USN-?. It means that by changing the GPS coordinate code into the class library, we can easily send it to the server, and so the user quickly knows the location of the bike and also use the real-time. The approximate distance that the GPS displays is about ten to twenty meters away from the targeted address.

Code Used:

1.https://docs.google.com/document/d/1y4OdLxxhsekS3CXoF1z02NFUavV8pxOMHK_kycgQCNU/

Figure:



59°56'37.7"N 10°53'57.8"E

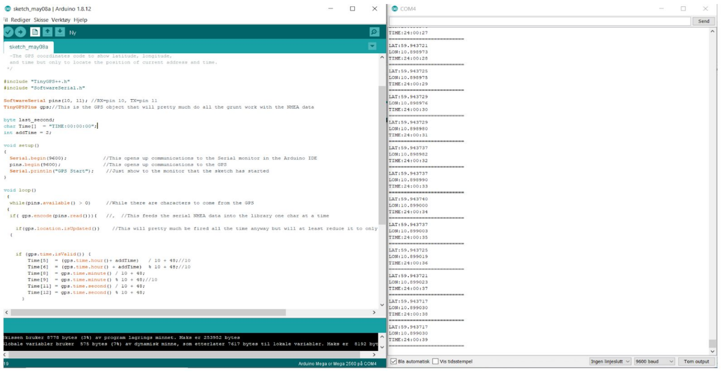
https://www.google.no/maps/place/59°56'37.7%22N+10°53'57.8%22E/@59.9437967,10.8971973,17z/data=!3m1!4b1!4m5!3m4!1s0x0:0x0!8m2!3d59.943794!4d10.899386?hl=no

59.943794 10.899386

59°56'37.7"N 10°53'57.8"E
59.943794, 10.899386

Granstangen 36, 1051 Oslo
WVVX+GQ Oslo
Legg til et manglende sted
Legg til bedriften din

Fig 1. The location of the current address using the data received from Satellites



sketch_may08a | Arduino 1.8.12
Fil Rediger Skisse Verktøy Hjelp
Ny
sketch_may08a
COM4
Send
19
Arduino Mega or Mega 2560 på COM4
☑ Bla automatisk
☐ Vis tidsstempel
Ingen linjeslutt
9600 baud
Tom output

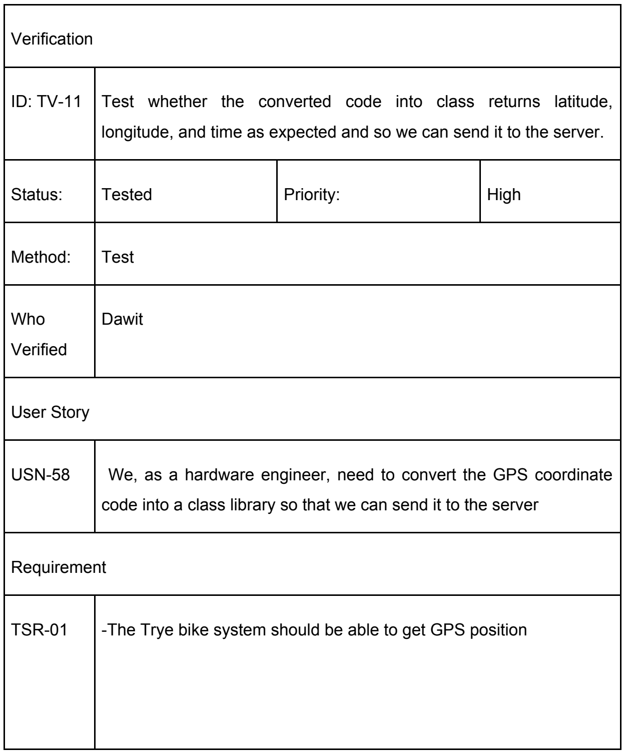
```
 -The GPS coordinates code to show latitude, longitude,
 and time but only to locate the position of current address and time.
 */

#include "TinyGPS++.h"
#include "SoftwareSerial.h"

SoftwareSerial pins(10, 11); //RX=pin 10, TX=pin 11
TinyGPSPlus gps;//This is the GPS object that will pretty much do all the grunt work with the NMEA data

byte last_second;
char Time[]  = "TIME:00:00:00";
int addTime = 2;

void setup()
{
  Serial.begin(9600);            //This opens up communications to the Serial monitor in the Arduino IDE
  pins.begin(9600);              //This opens up communications to the GPS
  Serial.println("GPS Start");    //Just show to the monitor that the sketch has started
}

void loop()
 {
  while(pins.available() > 0)     //While there are characters to come from the GPS
 {
  if( gps.encode(pins.read())){   //,  //This feeds the serial NMEA data into the library one char at a time

    if(gps.location.isUpdated())    //This will pretty much be fired all the time anyway but will at least reduce it to only
   {


    if (gps.time.isValid()) {
        Time[5]  = (gps.time.hour()+ addTime)   / 10 + 48;//10
        Time[6]  = (gps.time.hour() + addTime)  % 10 + 48;//10
        Time[8]  = gps.time.minute() / 10 + 48;
        Time[9]  = gps.time.minute() % 10 + 48;//10
        Time[11] = gps.time.second() / 10 + 48;
        Time[12] = gps.time.second() % 10 + 48;
      }
```

```
Skissen bruker 8778 bytes (3%) av program lagrings minnet. Maks er 253952 bytes
Globale variabler bruker  575 bytes (7%) av dynamisk minne, som etterlater 7617 bytes til lokale variabler. Maks er  8192 byt
```

```
TIME:24:00:27
========================
LAT:59.943721
LON:10.898973
TIME:24:00:28
========================
LAT:59.943725
LON:10.898975
TIME:24:00:29
========================
LAT:59.943729
LON:10.898976
TIME:24:00:30
========================
LAT:59.943729
LON:10.898980
TIME:24:00:31
========================
LAT:59.943737
LON:10.898982
TIME:24:00:32
========================
LAT:59.943737
LON:10.898990
TIME:24:00:33
========================
LAT:59.943740
LON:10.899000
TIME:24:00:34
========================
LAT:59.943737
LON:10.899003
TIME:24:00:35
========================
LAT:59.943725
LON:10.899019
TIME:24:00:36
========================
LAT:59.943721
LON:10.899023
TIME:24:00:37
========================
LAT:59.943717
LON:10.899030
TIME:24:00:38
========================
LAT:59.943717
LON:10.899030
TIME:24:00:39
========================
```

Fig 2. Using GPS coordinate code displays latitude, longitude, and time.

A.7.34 TV-11 Verifying USN-59



Verification			
ID: TV-11	Test whether the converted code into class returns latitude, longitude, and time as expected and so we can send it to the server.		
Status:	Tested	Priority:	High
Method:	Test		
Who Verified	Dawit		
User Story			
USN-58	We, as a hardware engineer, need to convert the GPS coordinate code into a class library so that we can send it to the server		
Requirement			
TSR-01	-The Trye bike system should be able to get GPS position		

Why this need to be tested:

The test needed because we need to check whether the reused GPS coordinate code into class returns the variable we are looking for or not.

How the test is performed:

We create the header file and put it into the Arduino library so that it connects with the GPS module. Since the header file is a class, we need to include different libraries such as TinyGPS++, SoftwareSerial, and Arduino where their use explained in USN-?.Then after we call functions such as String get.gps.time(); String get((gps.location.lng(),6)); and String get((gps.location.lat(),6)), and so to if they return the variables as expected.

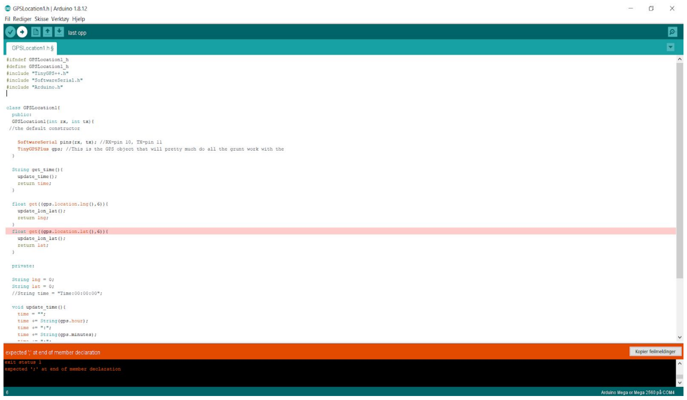
Result and conclusion:

The header file code shows an error when it is loading. I don`t understand well where the problem is. We, as a hardware team, can hopefully fix it together soon. The result shows in Figure 1 below.

Code Used:

1. https://docs.google.com/document/d/1x_QesjmqZjEYHke9Ac44JpN6L7Qq5X0sLInUp8aBjo/

Figure:


GPSLocation1.h | Arduino 1.8.12
Fil Rediger Skisse Verktøy Hjelp
last opp
GPSLocation1.h §
Kopier feilmeldinger
6
Arduino Mega or Mega 2560 på COM4

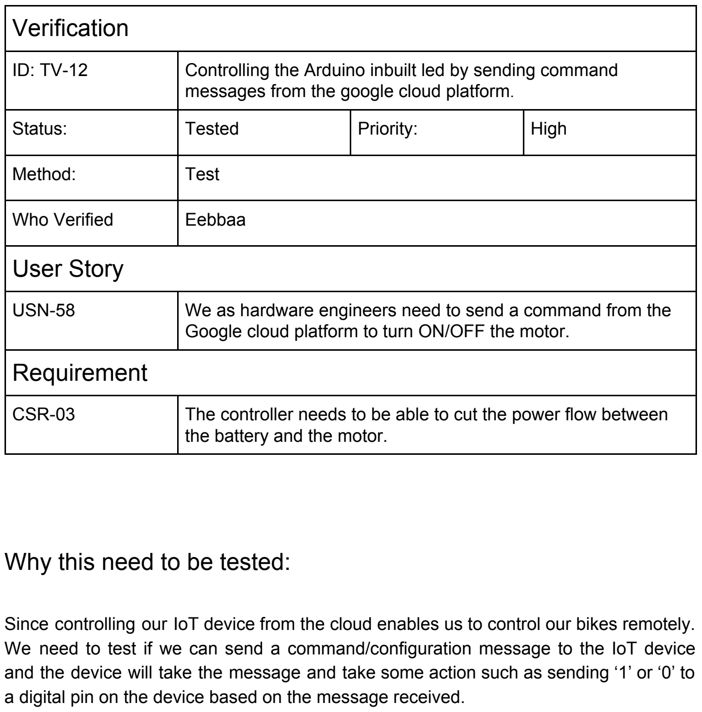
```
#ifndef GPSLocation1_h
#define GPSLocation1_h
#include "TinyGPS++.h"
#include "SoftwareSerial.h"
#include "Arduino.h"


class GPSLocation1{
  public:
  GPSLocation1(int rx, int tx){
 //the default constructor

    SoftwareSerial pins(rx, tx); //RX=pin 10, TX=pin 11
    TinyGPSPlus gps; //This is the GPS object that will pretty much do all the grunt work with the
  }

  String get_time(){
    update_time();
    return time;
  }

  float get((gps.location.lng(),6)){
    update_lon_lat();
    return lng;
  }
  float get((gps.location.lat(),6)){
    update_lon_lat();
    return lat;
  }

  private:

  String lng = 0;
  String lat = 0;
  //String time = "Time:00:00:00";

  void update_time(){
    time = "";
    time += String(gps.hour);
    time += ":";
    time += String(gps.minutes);
```

expected ';' at end of member declaration

```
exit status 1
expected ';' at end of member declaration
```

Fig 1. The class code when loading

A.7.35 TV-12 Verifying USN-58



Verification			
ID: TV-12	Controlling the Arduino inbuilt led by sending command messages from the google cloud platform.		
Status:	Tested	Priority:	High
Method:	Test		
Who Verified	Eebbaa		
User Story			
USN-58	We as hardware engineers need to send a command from the Google cloud platform to turn ON/OFF the motor.		
Requirement			
CSR-03	The controller needs to be able to cut the power flow between the battery and the motor.		

Why this need to be tested:

Since controlling our IoT device from the cloud enables us to control our bikes remotely. We need to test if we can send a command/configuration message to the IoT device and the device will take the message and take some action such as sending ‘1’ or ‘0’ to a digital pin on the device based on the message received.

How the test is performed:

An Arduino code used for sending data to the server and receiving data from the server on USN-44 is modified on its function that accepts the messages *void onMessageReceived (int messageSize)*. In the previous code this function is made just to display the size of the message. We modified the function as it can take the command /configuration message character by character and stores it in a variable and if the content of the message is “ON”, it will turn on the inbuilt led and displays to the serial monitor a text “turn ON the motor”. If the message is OFF will turn off the inbuilt led light and displays to the serial monitor a text “turn OFF the motor”.

Results:

Since our device is registered for both command/configuration topics, any message and commands from the cloud platform will be received. The command message is sent from the Google cloud platform from the project where the device is registered and the device receives the result and takes some action and the result is shown on the serial monitor as shown in Figure-1 and Figure-2.

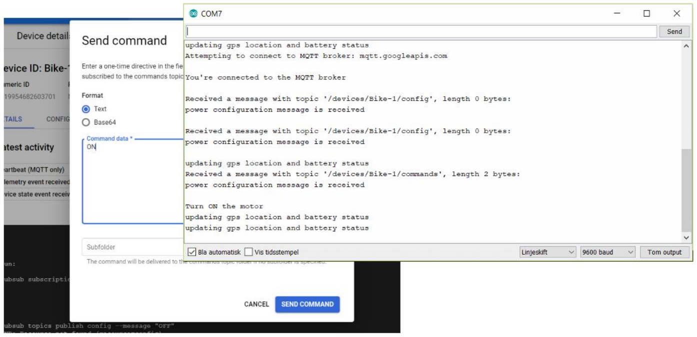
Conclusion:

The device receives The ON/OFF message as a command and based on the command the device turns ON or OFF the inbuilt led on the Arduino for 5 seconds. The main purpose of the test is to control the digital pin by sending either ‘1’ or ‘0’ from the cloud platform which is accomplished in this test. For the final prototype, we need to able to control the device by using the mobile app through the google cloud platform.

Code used

1. https://drive.google.com/drive/folders/1kr5Zj_DHEp9VVgxI-dTMZDsZgemnoEkm

Figure:



Send command

Format

● Text

○ Base64

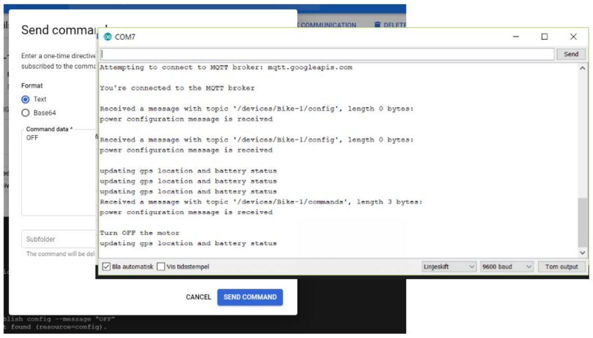
Command data *
ON

```
updating gps location and battery status
Attempting to connect to MQTT broker: mqtt.googleapis.com

You're connected to the MQTT broker

Received a message with topic '/devices/Bike-1/config', length 0 bytes:
power configuration message is received

Received a message with topic '/devices/Bike-1/config', length 0 bytes:
power configuration message is received

updating gps location and battery status
Received a message with topic '/devices/Bike-1/commands', length 2 bytes:
power configuration message is received

Turn ON the motor
updating gps location and battery status
updating gps location and battery status
```

☑ Bla automatisk □ Vis tidsstempel

Linjeskift · 9600 baud

Figure-1 ON message sent from the cloud platform and received by Arduino.



Send command

Format

● Text

○ Base64

Command data *
OFF

```
Attempting to connect to MQTT broker: mqtt.googleapis.com

You're connected to the MQTT broker

Received a message with topic '/devices/Bike-1/config', length 0 bytes:
power configuration message is received

Received a message with topic '/devices/Bike-1/config', length 0 bytes:
power configuration message is received

updating gps location and battery status
updating gps location and battery status
updating gps location and battery status
Received a message with topic '/devices/Bike-1/commands', length 3 bytes:
power configuration message is received

Turn OFF the motor
updating gps location and battery status
```

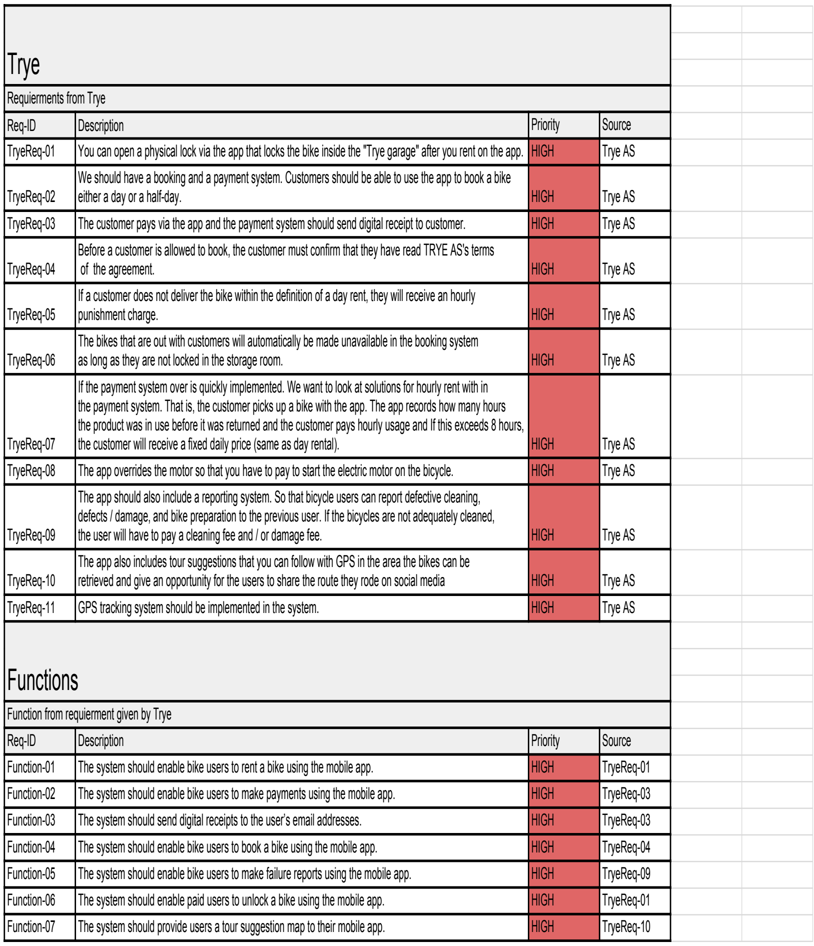
☑ Bla automatisk □ Vis tidsstempel

Linjeskift · 9600 baud

Figure-2: OFF message sent from the cloud platform and received by Arduino.

A.8 Requirements

A.8.1 Full list of requirements



Trye

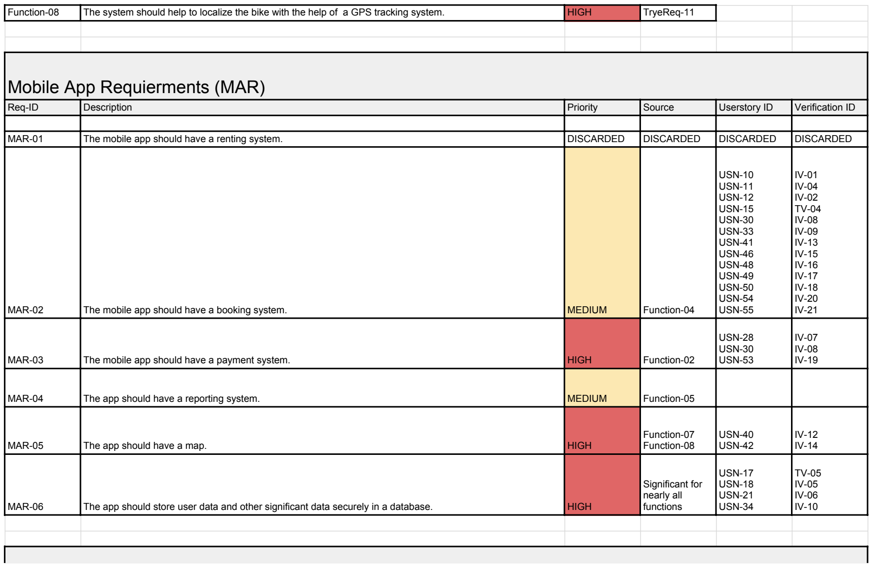
Requierments from Trye

Req-ID	Description	Priority	Source
TryeReq-01	You can open a physical lock via the app that locks the bike inside the "Trye garage" after you rent on the app.	HIGH	Trye AS
TryeReq-02	We should have a booking and a payment system. Customers should be able to use the app to book a bike either a day or a half-day.	HIGH	Trye AS
TryeReq-03	The customer pays via the app and the payment system should send digital receipt to customer.	HIGH	Trye AS
TryeReq-04	Before a customer is allowed to book, the customer must confirm that they have read TRYE AS's terms of the agreement.	HIGH	Trye AS
TryeReq-05	If a customer does not deliver the bike within the definition of a day rent, they will receive an hourly punishment charge.	HIGH	Trye AS
TryeReq-06	The bikes that are out with customers will automatically be made unavailable in the booking system as long as they are not locked in the storage room.	HIGH	Trye AS
TryeReq-07	If the payment system over is quickly implemented. We want to look at solutions for hourly rent with in the payment system. That is, the customer picks up a bike with the app. The app records how many hours the product was in use before it was returned and the customer pays hourly usage and If this exceeds 8 hours, the customer will receive a fixed daily price (same as day rental).	HIGH	Trye AS
TryeReq-08	The app overrides the motor so that you have to pay to start the electric motor on the bicycle.	HIGH	Trye AS
TryeReq-09	The app should also include a reporting system. So that bicycle users can report defective cleaning, defects / damage, and bike preparation to the previous user. If the bicycles are not adequately cleaned, the user will have to pay a cleaning fee and / or damage fee.	HIGH	Trye AS
TryeReq-10	The app also includes tour suggestions that you can follow with GPS in the area the bikes can be retrieved and give an opportunity for the users to share the route they rode on social media	HIGH	Trye AS
TryeReq-11	GPS tracking system should be implemented in the system.	HIGH	Trye AS

Functions

Function from requierment given by Trye

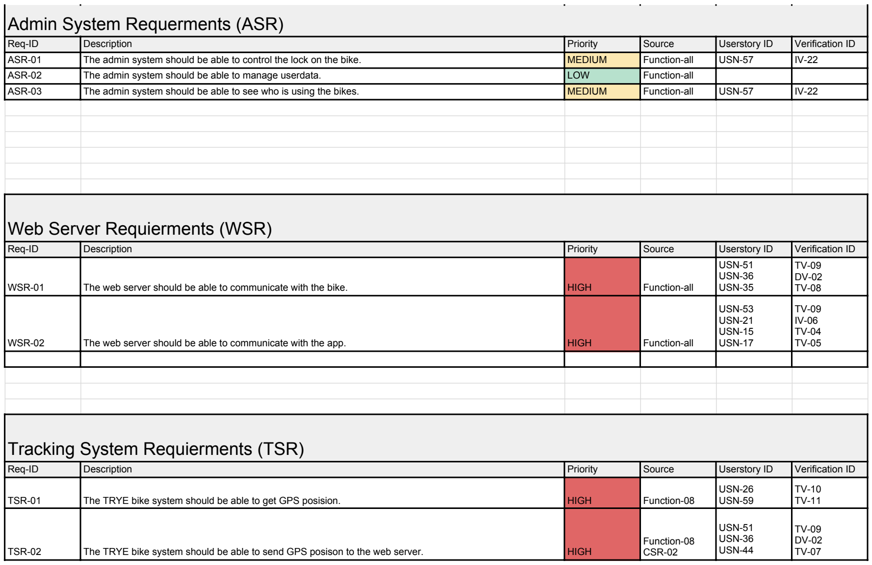
Req-ID	Description	Priority	Source
Function-01	The system should enable bike users to rent a bike using the mobile app.	HIGH	TryeReq-01
Function-02	The system should enable bike users to make payments using the mobile app.	HIGH	TryeReq-03
Function-03	The system should send digital receipts to the user's email addresses.	HIGH	TryeReq-03
Function-04	The system should enable bike users to book a bike using the mobile app.	HIGH	TryeReq-04
Function-05	The system should enable bike users to make failure reports using the mobile app.	HIGH	TryeReq-09
Function-06	The system should enable paid users to unlock a bike using the mobile app.	HIGH	TryeReq-01
Function-07	The system should provide users a tour suggestion map to their mobile app.	HIGH	TryeReq-10



Function-08	The system should help to localize the bike with the help of a GPS tracking system.	HIGH	TryeReq-11

Mobile App Requierments (MAR)

Req-ID	Description	Priority	Source	Userstory ID	Verification ID
MAR-01	The mobile app should have a renting system.	DISCARDED	DISCARDED	DISCARDED	DISCARDED
MAR-02	The mobile app should have a booking system.	MEDIUM	Function-04	USN-10 USN-11 USN-12 USN-15 USN-30 USN-33 USN-41 USN-46 USN-48 USN-49 USN-50 USN-54 USN-55	IV-01 IV-04 IV-02 TV-04 IV-08 IV-09 IV-13 IV-15 IV-16 IV-17 IV-18 IV-20 IV-21
MAR-03	The mobile app should have a payment system.	HIGH	Function-02	USN-28 USN-30 USN-53	IV-07 IV-08 IV-19
MAR-04	The app should have a reporting system.	MEDIUM	Function-05		
MAR-05	The app should have a map.	HIGH	Function-07 Function-08	USN-40 USN-42	IV-12 IV-14
MAR-06	The app should store user data and other significant data securely in a database.	HIGH	Significant for nearly all functions	USN-17 USN-18 USN-21 USN-34	TV-05 IV-05 IV-06 IV-10



Admin System Requerments (ASR)

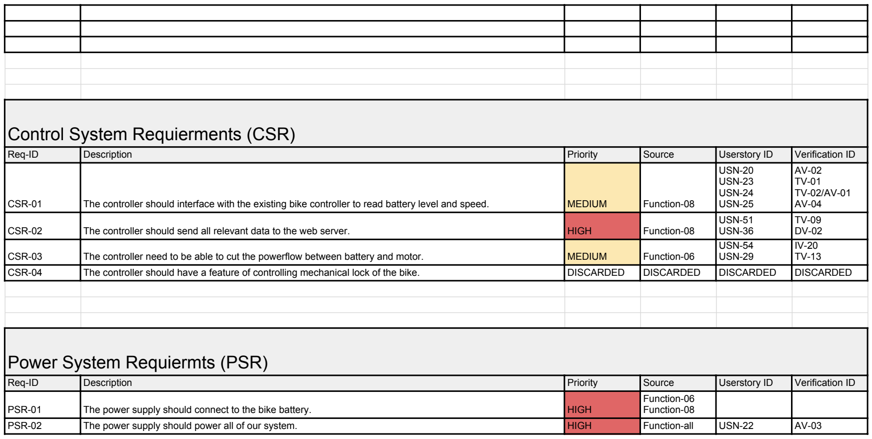
Req-ID	Description	Priority	Source	Userstory ID	Verification ID
ASR-01	The admin system should be able to control the lock on the bike.	MEDIUM	Function-all	USN-57	IV-22
ASR-02	The admin system should be able to manage userdata.	LOW	Function-all		
ASR-03	The admin system should be able to see who is using the bikes.	MEDIUM	Function-all	USN-57	IV-22

Web Server Requierments (WSR)

Req-ID	Description	Priority	Source	Userstory ID	Verification ID
WSR-01	The web server should be able to communicate with the bike.	HIGH	Function-all	USN-51 USN-36 USN-35	TV-09 DV-02 TV-08
WSR-02	The web server should be able to communicate with the app.	HIGH	Function-all	USN-53 USN-21 USN-15 USN-17	TV-09 IV-06 TV-04 TV-05

Tracking System Requierments (TSR)

Req-ID	Description	Priority	Source	Userstory ID	Verification ID
TSR-01	The TRYE bike system should be able to get GPS posision.	HIGH	Function-08	USN-26 USN-59	TV-10 TV-11
TSR-02	The TRYE bike system should be able to send GPS posison to the web server.	HIGH	Function-08 CSR-02	USN-51 USN-36 USN-44	TV-09 DV-02 TV-07



Control System Requierments (CSR)

Req-ID	Description	Priority	Source	Userstory ID	Verification ID
CSR-01	The controller should interface with the existing bike controller to read battery level and speed.	MEDIUM	Function-08	USN-20 USN-23 USN-24 USN-25	AV-02 TV-01 TV-02/AV-01 AV-04
CSR-02	The controller should send all relevant data to the web server.	HIGH	Function-08	USN-51 USN-36	TV-09 DV-02
CSR-03	The controller need to be able to cut the powerflow between battery and motor.	MEDIUM	Function-06	USN-54 USN-29	IV-20 TV-13
CSR-04	The controller should have a feature of controlling mechanical lock of the bike.	DISCARDED	DISCARDED	DISCARDED	DISCARDED

Power System Requiermts (PSR)

Req-ID	Description	Priority	Source	Userstory ID	Verification ID
PSR-01	The power supply should connect to the bike battery.	HIGH	Function-06 Function-08		
PSR-02	The power supply should power all of our system.	HIGH	Function-all	USN-22	AV-03

A.9 Risk tables

A.9.1 Table 1 Risk register

A.9.2 Table 2 Project risk

A.9.3 Table 3 Technical risk

The Descriptions of risk tables document

This documentation includes the project risk register table, project risks table, and technical risks table. The first table is the risk register table and consists of the following items.: The risk description, risk type, risk code (TR-Technical risks, RP-Project risks), code number for risk, risk ID (TR & RP) followed by a number, impact, probability, possible causes, mitigation action, date of risk update, and the owner of the risk. We have registered all types of risks together with the result of risk analysis and the mitigation action. The risk product is the result we found by multiplying the value given for the probability of a risk occurring and the value given for the impact of risk when it happens. We could see different colors specifically on the column of the risk product. The red color represents the higher the impact of the risk on the project, whereas the yellow color represents the medium impact on the risk. The light green color represents the risks with very low impact, whereas the green color is the risk with a low impact.

Each risk in the register has an ID followed by a number. The ID uses to track the risk in the project. The owner of the risk is the person who monitors, follows,and controls the uncertainty in the project. The owner is also updating the risk every time. We can also look at different types of risk in the register. Mainly these risks divided into two: project risk and technical risk. The project risks include personal risk, schedule risk, budget risk, scope risk, and quality risk. The technical risks include only these things connected with technical challenges in the project.

The second and third tables are the project risks table and the technical risks table. These two tables look the same in structure, but the difference is only types of risk. Project risks are a kind of general risk, whereas technical risks are technical kinds of risks. Each risk in both tables has given an ID. The project scope, quality, cost, velocity, and credibility are areas that could be affected by the impact of the risk. The scale of weight is from one to ten, and explain how much each risk affects the objective of the project. The results under the column of total impacts are the sum of weight for each risk, and we call it the sum of products for each risk. The values under the normalized impacts are values that correspond with the probability-impact matrix. The cause for each risk and the mitigation actions to respond to each threat also mentioned in both tables.

Risk Register: Risk Manager: Dawit Abamachu Date: 10/05-2020

Risk description	Risk type	Risk code	Code-number	Risk ID	Impact	Probability	Risk product	Possible cause	Mitigation
Prototype is not yet finished on time.	Technical	TR	1	TR-1	3	3	9	Short timeframe.	Use Planet9 and make sure the group members always have a task.
Some improtant components are not available on the expected time.	Schedule	PR	1	PR-1	4	5	20	Late ordering things needed.	Order components as soon as possible.
One member spends less time on the give work than the expected time.	Personal	PR	2	PR-2	3	2	6	Illness or lack of motivation.	Help each other, be transparent to each other.
One member may quit the group.	Personal	PR	3	PR-3	4	2	8	Illness, personal issues or group conflicts	Help each other, work friendly together and appreciate each other.
Lack of responsibility,	Personal	PR	4	PR-4	4	1	4	Lack of interest, less motivation.	Encourage each other and take a talk with the internal supervisor.
Misinterpretation of the requirements by the member of the group.	Personal	PR	5	PR-5	4	1	4	Lack of experience, lack of knowledge	Discuss the requirements and figure it out well.
Misinterpretation of the requirements by the member of the group.	Personal	PR	5	PR-5	4	1	4	Lack of experience, lack of knowledge	Discuss the requirements and figure it out well.
Several members may loss interest to do the project.	Schedule	PR	6	PR-6	5	2	10	Technical difficulties.	Encourage each other, take advice from the internal supervisor.
Several members does less work than the minimum amount of work expected.	Scope	PR	7	PR-7	5	2	10	Lack of motivation, illness, group conflicts.	Encourage each other, take advice from the internal supervisor.
The system failure.	Budget	PR	8	PR-8	4	2	8	Lack of functionalties, technical defects, incorrect assumptions of the system requirements.	Design and test the hardware and software well, take time with technical difficulties.
The project takes longer time than it is scheduled.	Budget	PR	9	PR-9	3	2	6	The members of the team work less than time expected, not following the plan of the project.	Discuss with the company manager and find a solution.
The requirements doesn't match the product expected.	Schedule	PR	10	PR-10	3	3	9	Not understanding the requirements.	Discuss with the company manager and find a solution

The hardware box doesn’t fit on bike.	Technical	TR	2	TR-2	4	1	4	Parts are too big/in the wrong shape.	Making sure the parts are small enough to fit the bike.
Difficulty of implementation.	Schedule	PR	11	PR-11	4	4	16	Complex technology.	Reading, discussing and understanding on implemetation of the relevant technlogy.
Signals between software and hardware fail during integration phase.	Technical	TR	3	TR-3	4	3	12	The enviorment makes it difficult to send and receive data/Server downtime.	Making sure we have a strong enough antenna, and retrieve error messages if the server is down.
Security incident delays access to the system.	Schedule	PR	12	PR-12	2	2	4	Disruption of normal operation because of hardware or software failure.	Prevent a threat actor from gaining access to the system.
Loss of user interface consistency across the application.	Technical	TR	4	TR-4	3	1	3	Multiple developers on same system, lack of agreement on UI.	Making sure to agree on the looks and feel of the system before creating it.
Some components break during implementation phase.	Technical	TR	5	TR-5	3	3	9	Parts are fragile and breaks easily, misuse of parts.	Making sure to order extra parts if available, and read the proper documentation for each part.
Operational activities might hamper.	Schedule	PR	13	PR-13	3	3	9	Improper implementation or conflicting priorities.	Choosing the right protocol for the hardware and software, and discuss and work intimately in the implementation phase.
Disagreement between the member of the group.	Personal	PR	14	PR-14	3	2	6	Too demanding workload or lack of flexilibity.	Working efficiently when supposed to and being flexible.
One-member loss interest to do the project.	Personal	PR	15	PR-15	3	1	3	Illness or lack of motivation.	Help each other, be transparent to each other.
The product delivery due unanticipated technical hurdles.	Personal	PR	16	PR-16	4	2	8	The software bugs or loss of connection.	Test and verify each requirment in each phase.
Lack of good guidelines.		PR	17	PR-17	2	2	4	Fewer workhours on the project.	Discuss together and work together.
Dependencies on other tasks.	Schedule	PR	18	PR-18	1	1	1	Late errors in the system.	Make sure everyone are aware of potential bottlenecks, and work as a team.
Requirements conflict during integration and coding.	Technical	TR	6	TR-6	2	2	4	Massive quantity of requirments or change in requiments during system development phase.	Talk with the company, and avoid changes to the requirments in the development phase.

Some electrical connections fail when a bicycle get damaged.	Technical	TR	7	TR-7	2	3	6	Loose connection, exposed wire, not enough slack on wire.	Making sure every cable is properly fitted, (not too tight) and is covered from the environment.
Internet coverage may fail because the targeted area is mountainous/highland.	Technical	TR	8	TR-8	2	4	8	Antenna is not big enough.	Make use of analysis to figure out how big of an antenna we need for our environment.
Unavailability of some technical equipments.	Schedule	PR	19	PR-19	2	2	4	Ordering parts late, poor planning on what is important.	Plan together as a team on important equipment.
Hardware doesn't work as expected during operation.	Technical	TR	9	TR-9	5	2	10	Hardware does not communicate as told or loose wires.	Make sure to properly test the system before deployment and TR-07.
Lack of documentation.	Personal	PR	20	PR-20	3	2	6	Neglectance or lack of time.	Document as early as possible.
Project schedule is not clearly defined.	Schedule	PR	21	PR-21	4	3	12	Lack of a good plan, lack of experience in long term projects.	Discuss the schedule with the team members, so that everyone has a common understanding of the plan.
Software doesn't work during production.	Technical	TR	10	TR-10	5	2	10	Error in coding, faulty integration with hardware.	Make sure to properly test the system before deployment and debug the system.
Change of requirements.	Schedule	PR	22	PR-22	3	2	6	The company comes up with new requirements.	During our meeting with the company, we will ask them if any requirements has changed from their side.
COVID-19 pandemic	Schedule	PR	23	PR-23	4	5	20	The project team is not able to meet and work together; as usual, we cannot meet and work on technical tasks together.	We must accept the problem concerning pandemic. Therefore, we must think about alternatives and use different technologies to continue working on our project. Use discord for standup meeting; use Zoom to talk with a supervisor, sensors, and others.

	Weights that affect the obj	Project scope		Project Quality		Project cost		Project velocity		Project credibility		Total Impact	Normalized Impact (1-5)	Possible cause	Probability (1-4)	Risk product	Mitigation action
	Weight(1-10)	8		5		1		7		6							
Risk ID	Project Risk																
RP-01	Some important components are not available in the expected amount of time.	6	48	7	35	0	0	6	42	5	30	155	5	Late oredring things needed.	4	20	Order components as soon as possible.
RP-02	One member spends less time on the given work than expected.	5	40	5	25	0	0	7	49	6	36	150	3	Illness or lack of motivation.	2	6	Help each other, be transparent to each other.
RP-03	One member might quit the group.	6	48	8	40	0	0	7	49	7	42	179	2	Ordering the required hardware too late.	4	8	Help each other, work friendly together and appreciate each other.
RP-04	Lack of responsibility.	7	56	8	40	0	0	8	56	8	48	200	4	Lack of interest, less motivation.	1	4	Encourage each other and take a talk with the internal supervisor.
RP-05	Misinterpretation of the requirements by the member of the group.	4	32	3	15	5	5	4	28	5	30	110	4	Lack of experience, lack of knowledge.	4	16	Discuss the requirements and figure it out well.
RP-06	Several members loses interest in the project.	10	80	10	50	0	0	10	70	8	48	248	5	Technical difficulties.	2	10	Encourage each other, take advice from the internal supervisor.
RP-07	Several members works less than the minimum amount of work expected.	9	72	9	45	0	0	9	63	7	42	222	5	Lack of motivation, illness, group conflicts.	2	10	Encourage each other, take advice from the internal supervisor.
RP-08	The system fails.	7	56	7	35	0	0	7	49	6	36	176	4	Lack of functionalties, technical defects, incorrect assumptions of the system requirements.	2	8	Design and test the hardware and software well, take time with technical difficulties.
RP-09	The project takes more time than scheduled.	5	40	6	30	0	0	5	35	4	24	129	3	The members of the team work less than time expected, not following the plan of the project.	2	6	Discuss with the company manager and find a solution.
RP-10	The original requirements doesn't match the product produced.	4	32	3	15	3	3	4	28	4	24	102	3	Not understanding the requirements.	3	9	Discuss with the company manager and find a solution.
RP-11	Difficulties with the implementation.	7	56	8	40	0	0	0	0	6	36	132	4	Complex technology.	4	16	Read and discuss relevant technlogy.
RP-12	Security incidents prevents access to the system.	3	24	4	20	0	0	3	21	4	24	89	2	Disruption of normal operation because of hardware or software failure.	2	4	Prevent a threat actor from gaining access to the system.
RP-13	Operational activities might get hampered.	4	32	5	25	0	0	6	42	5	30	129	3	Improper implementation or conflicting priorities.	3	9	Choosing the right protocol for the hardware and software, and discuss and work intimately in the implementation phase.
RP-14	Disagreement between the members of the group.	2	16	3	15	0	0		0	4	24	55	3	Too demanding workload or lack of flexibility.	2	6	Working efficiently when supposed to and being flexible.
RP-15	One member loses interest in the project.	5	40	6	30	0	0	7	49	6	36	155	3	Illness or lack of motivation.	1	3	Help each other, be transparent to each other.

RP-16	The product is delayed due to unanticipated technical hurdles.	6	48	5	25	0	0	6	42	4	24	139	4	The software bugs or loss of connection.	2	8	Test and verify each requirment in each phase.
RP-17	Lack of good guidelines.	1	8	1	5	0	0	2	14	1	6	33	2	Fewer workhours on the project.	2	4	Discuss together and work together.
RP-18	Dependencies on other tasks.	1	8	1	5	0	0	2	14	1	6	33	1	Late errors in the system.	1	1	Make sure everyone are aware of potential bottlenecks, and work as a team.
RP-19	Not having the required technical equipment.	1	8	2	10	0	0	2	14	2	12	44	2	Ordering parts late, poor planning on what is important.	2	4	Plan together as a team on important equipment.
RP-20	Lack of documentation.	2	16	3	15	0	0	3	21	2	12	64	3	Neglectance or lack of time.	2	6	Documentation is archived in Google Drive, LaTex or on team members personal computer.
RP-21	The project schedule is not clearly defined.	5	40	4	20	0	0	4	28	2	12	100	4	Lack of a good plan, lack of experience in long term projects.	3	12	Discuss the schedule with the team members, so that everyone has a common understanding of the plan.
RP-22	Change of requirements.	2		2	10	0	0	3	21	2	12	43	3	The company comes up with new requirements.	2	6	During our meeting with the company, we will ask them if any requirements has changed from their side.
RP-23	COVID-19 pandemic	8	64	6	30	3	3	7	49	5	30	176	5	The project team is not able to meet and work together; as usual, we cannot meet and work on technical tasks together.	4	20	We must accept the problem concerning pandemic. Therefore, we must think about alternatives and use different technologies to continue working on our project. Use discord for standup meeting; use Zoom to talk with a supervisor, sensors, and others.

	Weight affect the objectives	Project scope		Project Quality		Project cost		Project velocity		Project credibility		Total Impact	Normalized Impact(1-5)	Possible cause	Pro-bability (1-4)	Risk Product	Mitigation action
	Weight (1-10)	8		5		1		7		6							
Risk ID	Technical Risk																
TR-01	Prototype is not yet finished on time.	2	16	1	5	0	0	2	14	4	24	59	3	Short timeframe.	3	9	Use Planet9 and make sure the group members always have a task.
TR-02	The hardware box doesn't fit on bike.	3	24	7	35	1	1	1	7	3	18	85	4	Parts are too big/in the wrong shape.	1	4	Making sure the parts are small enough to fit the bike.
TR-03	Signals between software and hardware fail during integration phase.	6	12	6	30	0	0	2	14	6	36	92	4	The enviorment makes it difficult to send and receive data/Server downtime.	3	12	Making sure we have a strong enough antenna, and retrieve error messages if the server is down.
TR-04	Signals between software and hardware fail during integration phase.	4	12	5	25	0	0	1	7	5	30	74	3	The enviorment makes it difficult to send and receive data/Server downtime.	1	3	Making sure we have a strong enough antenna, and retrieve error messages if the server is down.
TR-05	Some components break during implementation phase.	1	6	4	20	8	8	3	21	4	24	79	3	Parts are fragile and breaks easily, misuse of parts.	3	9	Making sure to order extra parts if available, and read the proper documentation for each part.
TR-06	Requirements conflict during integration and coding.	1	4	5	25	0	0	2	14	2	12	55	2	Massive quantity of requirments or change in requirnents during system development phase.	2	4	Talk with the company, and avoid changes to the requirments in the development phase.
TR-07	Some electrical connections fail when a bicycle get damaged.	3	3	4	20	0	0	1	7	2	12	42	2	Loose connection, exposed wire, not enough slack on wire.	3	6	Making sure every cable is properly fitted, (not too tight) and is covered from the environment.
TR-08	Internet coverage may fail because the targeted area is mountainous/highland.	4	4	4	20	0	0	1	7	3	18	49	2	Antenna is not big enough.	4	8	Make use of analysis to figure out how big of an antenna we need for our environment.
TR-09	Hardware doesn't work as expected during operation	6	18	4	20	0	0	5	35	8	48	121	5	Hardware does not comunicate as told, Lose wires	2	10	make sure to properly test the system before deployment and TR-07
TR-10	Hardware doesn't work as expected during operation.	6	24	8	40	0	0	6	42	8	48	154	5	Hardware does not communicate as told or loose wires.	2	10	Make sure to properly test the system before deployment and TR-07.

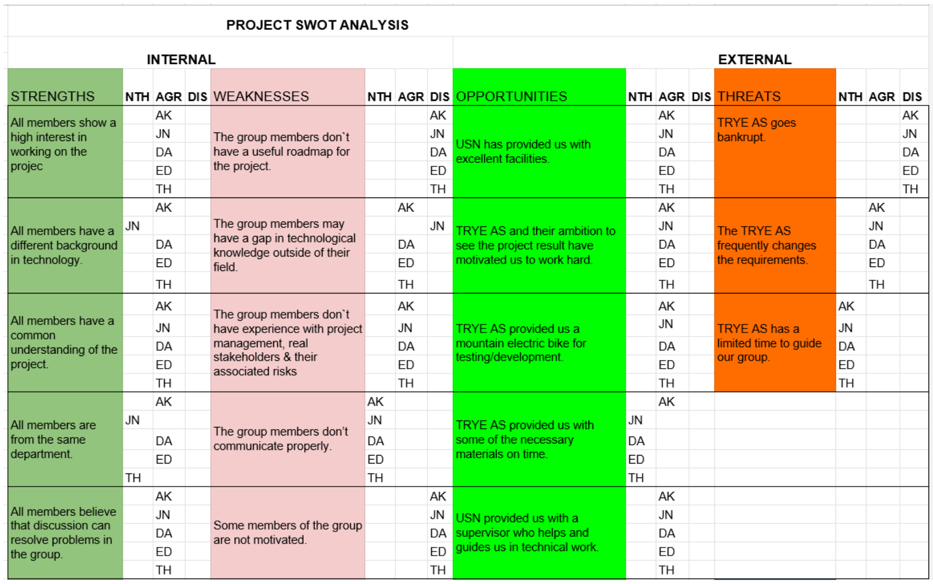
A.9.4 Description of SWOT Analysis for the project

SWOT Analysis is a robust and straightforward technique used for developing Strategic planning and as a risk identification tool. It helps us to understand the surrounding environment from the internal and external perspectives. SWOT stands for Strengths, Weaknesses, Opportunities, and Threats. Strengths and Weaknesses represent the inner qualities of the team members in the project. Those qualities can be positive and negative and have effects on the objective of the project. Opportunities and Threats are external factors that have also impact on the project.

The primary reason to select the SWOT Analysis is, it is a simple, easy, and effective tool to perform the strategic activities. As the risk generally harms the project, the Weaknesses, and the Threats can be considered the critical categories of the SWOT analysis. In contrast, Opportunities and Strengths provide us the advantage to create a strategy that increases the success of the project.

All the information that we have collected, their contents can be rewritten in terms of the risk. Finally, based on the result of risk analysis, we can reduce and avoid the risk impact and increase the chance of project success.[15]

In the SWOT analysis table, the following abbreviations used. These are "NITH" means neither, "AGR" means agree, "DIS" means to disagree. These suggest that items which we have accepted, their contents can be rewritten as a risk and taken for further analysis.

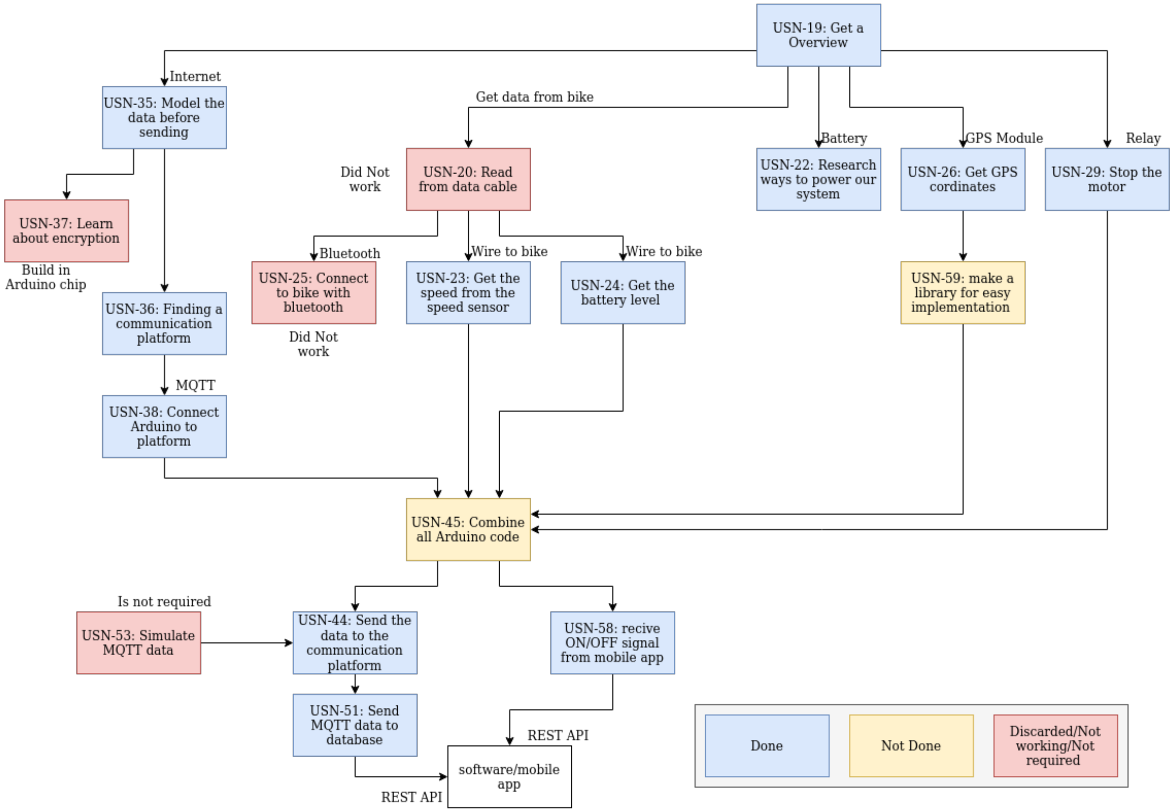


PROJECT SWOT ANALYSIS

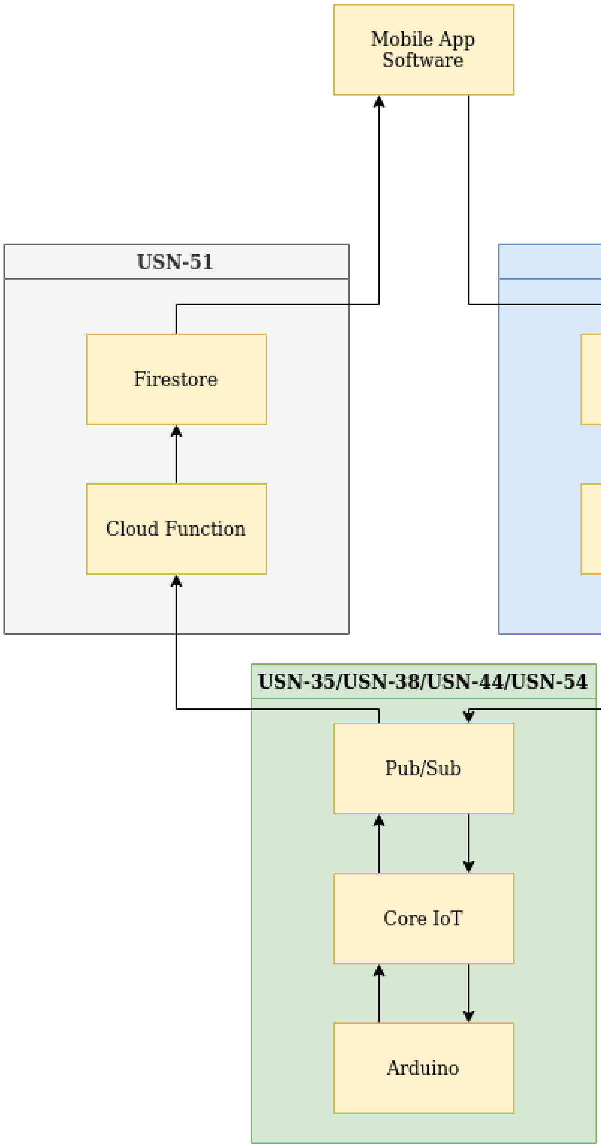
INTERNAL								EXTERNAL							
STRENGTHS	NTH	AGR	DIS	WEAKNESSES	NTH	AGR	DIS	OPPORTUNITIES	NTH	AGR	DIS	THREATS	NTH	AGR	DIS
All members show a high interest in working on the projec		AK JN DA ED TH		The group members don`t have a useful roadmap for the project.			AK JN DA ED TH	USN has provided us with excellent facilities.		AK JN DA ED TH		TRYE AS goes bankrupt.			AK JN DA ED TH
All members have a different background in technology.	JN	AK DA ED TH		The group members may have a gap in technological knowledge outside of their field.		AK DA ED TH	JN	TRYE AS and their ambition to see the project result have motivated us to work hard.		AK JN DA ED TH		The TRYE AS frequently changes the requirements.		AK JN DA ED TH	
All members have a common understanding of the project.		AK JN DA ED TH		The group members don`t have experience with project management, real stakeholders & their associated risks		AK JN DA ED TH		TRYE AS provided us a mountain electric bike for testing/development.		AK JN DA ED TH		TRYE AS has a limited time to guide our group.	AK JN DA ED TH		
All members are from the same department.	JN TH	AK DA ED		The group members don't communicate properly.	AK JN DA ED TH			TRYE AS provided us with some of the necessary materials on time.	JN DA ED TH	AK					
All members believe that discussion can resolve problems in the group.		AK JN DA ED TH		Some members of the group are not motivated.			AK JN DA ED TH	USN provided us with a supervisor who helps and guides us in technical work.		AK JN DA ED TH					

A.10 Hardware Images

A.10.1 hardware development process

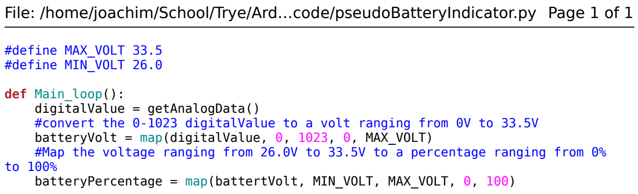

USN-19: Get a Overview
Internet
Get data from bike
Battery
GPS Module
Relay
USN-35: Model the data before sending
USN-20: Read from data cable
Did Not work
USN-22: Research ways to power our system
USN-26: Get GPS cordinates
USN-29: Stop the motor
USN-37: Learn about encryption
Build in Arduino chip
Bluetooth
Wire to bike
Wire to bike
USN-25: Connect to bike with bluetooth
Did Not work
USN-23: Get the speed from the speed sensor
USN-24: Get the battery level
USN-59: make a library for easy implementation
USN-36: Finding a communication platform
MQTT
USN-38: Connect Arduino to platform
USN-45: Combine all Arduino code
Is not required
USN-53: Simulate MQTT data
USN-44: Send the data to the communication platform
USN-58: recive ON/OFF signal from mobile app
USN-51: Send MQTT data to database
REST API
REST API
software/mobile app
Done
Not Done
Discarded/Not working/Not required

A.10.2 hardware development process



A.11 Hardware Code

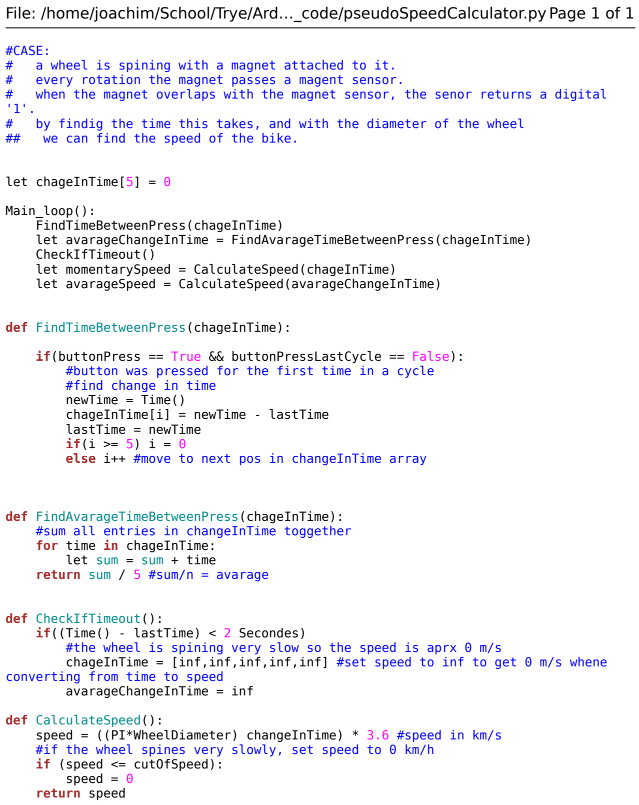
A.11.1 Pseudo code Battery Reader



File: /home/joachim/School/Trye/Ard...code/pseudoBatteryIndicator.py Page 1 of 1

```
#define MAX_VOLT 33.5
#define MIN_VOLT 26.0

def Main_loop():
    digitalValue = getAnalogData()
    #convert the 0-1023 digitalValue to a volt ranging from 0V to 33.5V
    batteryVolt = map(digitalValue, 0, 1023, 0, MAX_VOLT)
    #Map the voltage ranging from 26.0V to 33.5V to a percentage ranging from 0%
to 100%
    batteryPercentage = map(battertVolt, MIN_VOLT, MAX_VOLT, 0, 100)
```

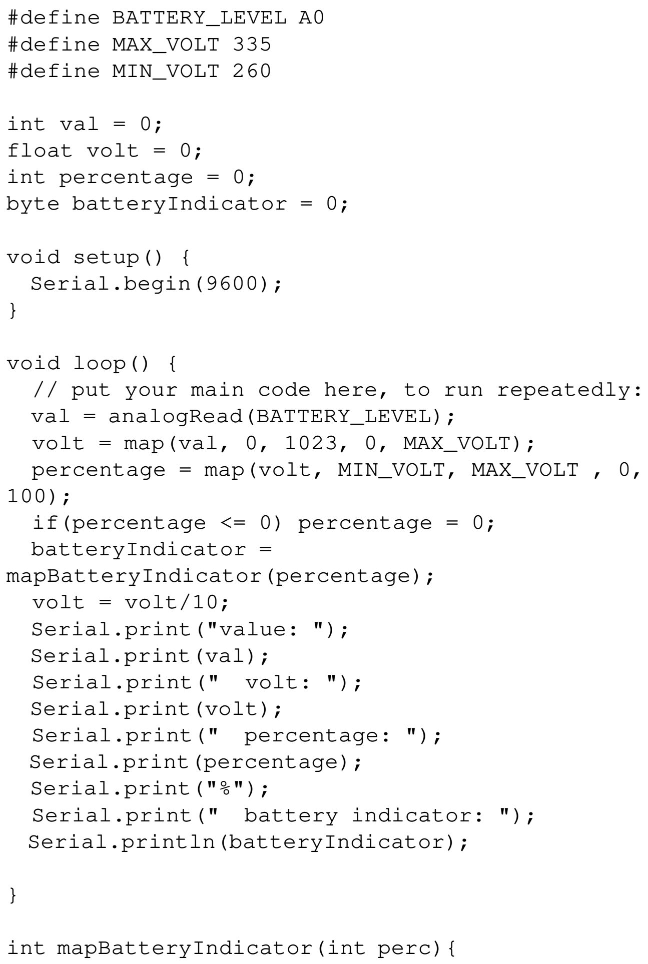
A.11.2 Pseudo code Speed Reader



File: /home/joachim/School/Trye/Ard..._code/pseudoSpeedCalculator.py Page 1 of 1

```
#CASE:
#   a wheel is spining with a magnet attached to it.
#   every rotation the magnet passes a magent sensor.
#   when the magnet overlaps with the magnet sensor, the senor returns a digital
'1'.
#   by findig the time this takes, and with the diameter of the wheel
##   we can find the speed of the bike.


let chageInTime[5] = 0

Main_loop():
    FindTimeBetweenPress(chageInTime)
    let avarageChangeInTime = FindAvarageTimeBetweenPress(chageInTime)
    CheckIfTimeout()
    let momentarySpeed = CalculateSpeed(chageInTime)
    let avarageSpeed = CalculateSpeed(avarageChangeInTime)


def FindTimeBetweenPress(chageInTime):

    if(buttonPress == True && buttonPressLastCycle == False):
        #button was pressed for the first time in a cycle
        #find change in time
        newTime = Time()
        chageInTime[i] = newTime - lastTime
        lastTime = newTime
        if(i >= 5) i = 0
        else i++ #move to next pos in changeInTime array



def FindAvarageTimeBetweenPress(chageInTime):
    #sum all entries in changeInTime toggether
    for time in chageInTime:
        let sum = sum + time
    return sum / 5 #sum/n = avarage


def CheckIfTimeout():
    if((Time() - lastTime) < 2 Secondes)
        #the wheel is spining very slow so the speed is aprx 0 m/s
        chageInTime = [inf,inf,inf,inf,inf] #set speed to inf to get 0 m/s whene
converting from time to speed
        avarageChangeInTime = inf

def CalculateSpeed():
    speed = ((PI*WheelDiameter) changeInTime) * 3.6 #speed in km/s
    #if the wheel spines very slowly, set speed to 0 km/h
    if (speed <= cutOfSpeed):
        speed = 0
    return speed
```

A.11.3 Battery Reader Arduino Code



```
#define BATTERY_LEVEL A0
#define MAX_VOLT 335
#define MIN_VOLT 260

int val = 0;
float volt = 0;
int percentage = 0;
byte batteryIndicator = 0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  val = analogRead(BATTERY_LEVEL);
  volt = map(val, 0, 1023, 0, MAX_VOLT);
  percentage = map(volt, MIN_VOLT, MAX_VOLT , 0,
100);
  if(percentage <= 0) percentage = 0;
  batteryIndicator =
mapBatteryIndicator(percentage);
  volt = volt/10;
  Serial.print("value: ");
  Serial.print(val);
  Serial.print("  volt: ");
  Serial.print(volt);
  Serial.print("  percentage: ");
  Serial.print(percentage);
  Serial.print("%");
  Serial.print("  battery indicator: ");
  Serial.println(batteryIndicator);

}

int mapBatteryIndicator(int perc){
  int level = 0;
  if(perc >= 81) level = 5;
  else if(perc >= 61 && perc <= 80) level = 4;
  else if(perc >= 41 && perc <= 60) level = 3;
  else if(perc >= 21 && perc <= 40) level = 2;
  else if(perc >= 2 && perc <= 20) level = 1;
  else if(perc >= 0 && perc <= 1) level = 0;
  else level = -1;

  return level;
}
```

A.11.4 Speed Reader Arduino Code

```
//---------SETTINGS-----------
#define SWITCH 3  //the pin the magnet switch is
connected to
#define D_TIME_ARR_SIZE 5 //to calculate the
average speed, how many rotation should be
calculated with?
#define WHEEL_DIAMETER 27.5 //the Wheel diameter
on the bike in inches
#define INCH_TO_M 0.0254  //convert wheel diameter
inch to meter
#define SPEED_CUTOF 2  //when you go slower then
this, the micro controller setts the speed to 0
km/h
#define TIMEOUT_TIME 5000 //if the microcontroller
does not recive a signal within this time, it sets
the speed to 0 km/h

//---------VARIABLES-----------

float Speed = 0;
float momentarySpeed = 0;
unsigned long previousMillis = 0;
unsigned long currentMillis = 0;
unsigned long deltaMillis = 0;
unsigned long avgDeltaMillis = 100000;

//set the array to 100000 to get a speed of 0km/h
to start with
unsigned long deltaMillisArr[D_TIME_ARR_SIZE] =
{100000,100000,100000,100000,100000};
byte deltaMillisArrIndex = 0;

bool previousSwitchState = LOW;
bool currentSwitchState = LOW;

//temp
```

```
unsigned long tempSum = 0;

//---------SETUP-----------

void setup() {
  //starting serial communication
  Serial.begin(9600);
  //setting port mode
  pinMode(SWITCH, INPUT);

  previousMillis = millis();
  currentMillis = millis();
}

//---------MAIN-----------

void loop() {
  findDeltaTime();
  findAvgDeltaTime();
  CheckTimeout();
  Speed = calculateSpeed(avgDeltaMillis);
  momentarySpeed = calculateSpeed(deltaMillis);


  printStatus();
  delay(300);
}


//---------FUNCTUIOS-----------


/*
 * CheckTimeout()
 * if the wheel does not sping in a period of
TIMEOUT_TIME (eg. 5000 ms) we will asume the wheel
```

```
is not rotating at all.
 * which means a speed of 0 km/h
 */
void CheckTimeout(){
  if ((millis() - currentMillis) >= TIMEOUT_TIME){
    Serial.print("TIMEOUT after: ");
    Serial.println(millis() - currentMillis);
    for (int i = 0; i < D_TIME_ARR_SIZE; i++){
      deltaMillisArr[i] = TIMEOUT_TIME;
    }
    deltaMillis = TIMEOUT_TIME;
  }
}

float calculateSpeed(unsigned long changeInTime){
  float speedCalc =  ((PI *
WHEEL_DIAMETER*INCH_TO_M * 1000)/changeInTime)*3.6;
  if (speedCalc <= SPEED_CUTOF) speedCalc = 0;
  return speedCalc;
}


/*
 * findDeltaTime()
 * find the time it takes form one button press to
the next press, which is an analogy to the wheel
spinning once
 * also stores the D_TIME_ARR_SIZE (eg. 5) last
results to be able to calculate the average time
 */
void findDeltaTime(){
  //reading the switch
  currentSwitchState = digitalRead(SWITCH);
  if(currentSwitchState == HIGH){
    //SWICH IS PRESED
    //to stope debouncing
    if(currentSwitchState != previousSwitchState){
      //to get the change in time between each
wheel rotation
      previousMillis  = currentMillis;
      currentMillis = millis();
      deltaMillis = currentMillis - previousMillis;
      //to stope debouncing with the above if
statement
      previousSwitchState = currentSwitchState;

      //store delta time in array
      if (deltaMillisArrIndex <= D_TIME_ARR_SIZE){
        deltaMillisArr[deltaMillisArrIndex] =
deltaMillis;
        deltaMillisArrIndex = deltaMillisArrIndex
+ 1;
      }
      else{
        deltaMillisArrIndex = 0;
        deltaMillisArr[deltaMillisArrIndex] =
deltaMillis;
      }


    }
  }
  else{
    //SWITCH IS NOT PRESED
    previousSwitchState = currentSwitchState;
  }
}

/*
 * findAvgDeltaTime()
 * find the average time it takes for the wheel to
spin D_TIME_ARR_SIZE (eg. 5) times
```

```
 */
void findAvgDeltaTime(){
  //find avarage Delta time
  tempSum = 0;
  for (int i = 0; i < D_TIME_ARR_SIZE; i++){
    tempSum = tempSum + deltaMillisArr[i];
  }
  avgDeltaMillis = tempSum/D_TIME_ARR_SIZE;
}


/*
 * printStatus()
 * Print out to Serial port, used for debugging
and testing
 */
void printStatus(){
  Serial.print("Switch: ");
  Serial.println(digitalRead(SWITCH));

  Serial.print("delta time (ms): ");
  Serial.print(deltaMillis);
  Serial.print("  momentary Speed (km/h): ");
  Serial.println(momentarySpeed);


  Serial.print("avg delta time (ms): ");
  Serial.print(avgDeltaMillis);
  Serial.print("  Speed (km/h): ");
  Serial.println(Speed);
}
```

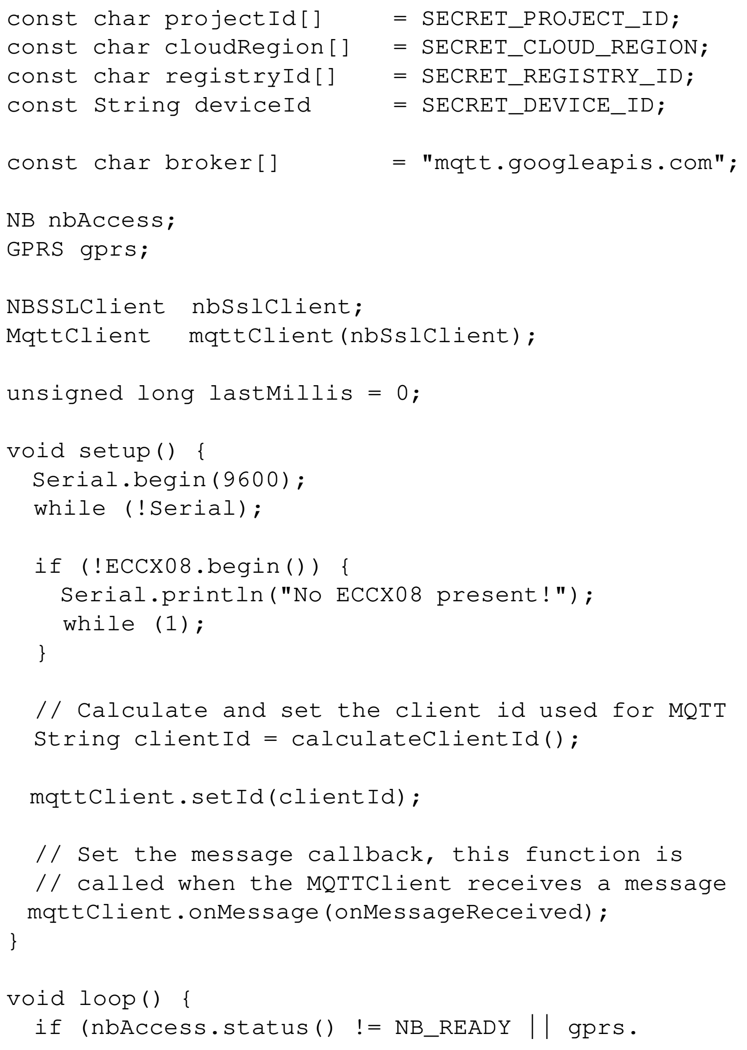
A.11.5 GPS Reader Arduino Code

```
#include "TinyGPS++.h"
#include "SoftwareSerial.h"


/*
 -This is the final GPS cordinates which include
valuable data such as
 latitude, longtiude, altitude in feet, time and
date
 -It shows the position and real-time for our bike
 */

#include "TinyGPS++.h"
#include "SoftwareSerial.h"

SoftwareSerial pins(10, 11); //RX=pin 10, TX=pin 11
TinyGPSPlus gps;//This is the GPS object that will
pretty much do all the grunt work with the NMEA
data

byte last_second;
char Time[]  = "TIME:00:00:00";
//char Date[]  = "DATE:00/00/2000";
int addTime = 2;

void setup()
{
  Serial.begin(9600);            //This opens up
communications to the Serial monitor in the
Arduino IDE
  pins.begin(9600);              //This opens up
communications to the GPS
  Serial.println("GPS Start");    //Just show to
the monitor that the sketch has started
}
```

```
void loop()
 {
  while(pins.available() > 0)     //While there
are characters to come from the GPS
 {
  if( gps.encode(pins.read())){   //,  //This
feeds the serial NMEA data into the library one
char at a time

    if(gps.location.isUpdated())    //This will
pretty much be fired all the time anyway but will
at least reduce it to only after a package of NMEA
data comes in
  {


    if (gps.time.isValid()) {
        Time[5]  = (gps.time.hour()+ addTime)   /
10 + 48;//10
        Time[6]  = (gps.time.hour() + addTime)  %
10 + 48;//10
        Time[8]  = gps.time.minute() / 10 + 48;
        Time[9]  = gps.time.minute() % 10 + 48;//10
        Time[11] = gps.time.second() / 10 + 48;
        Time[12] = gps.time.second() % 10 + 48;
      }
 /*
        if (gps.date.isValid()) {
        Date[5]  = gps.date.day()    / 10 + 48;
        Date[6]  = gps.date.day()    % 10 + 48;
        Date[8]  = gps.date.month()  / 10 + 48;
        Date[9]  = gps.date.month()  % 10 + 48;
        Date[13] =(gps.date.year()   / 10) % 10 +
48;
        Date[14] = gps.date.year()   % 10 + 48;
      }
 */
      if(last_second != gps.time.second()) {
        last_second = gps.time.second();
       // Serial.print(0,0);
        Serial.print("LAT:");
//Display latitude
        Serial.println(gps.location.lat(), 6);

        Serial.print("LON:");
//Display langitude
        Serial.println(gps.location.lng(), 6);
        //Serial.print("ALT Ft:");
        //Serial.println(gps.altitude.feet());
        Serial.println(Time);
//Display time
        //Serial.print(0,1);
        //Serial.
println(Date);                      // Display
calendar
        Serial.
println("=========================");

        }

      }

    }
 }
}
```

A.11.6 Arduino to GCP communication

```
/*
  GCP (Google Cloud Platform) IoT Core NB

  This sketch securely connects to GCP IoT Core
using MQTT over NB IoT/LTE Cat M1.
  It uses a private key stored in the ATECC508A
and a JSON Web Token (JWT) with
  a JSON Web Signature (JWS).

  It publishes a message every 5 seconds to
"/devices/{deviceId}/state" topic
  and subscribes to messages on the
"/devices/{deviceId}/config" and
  "/devices/{deviceId}/commands/#" topics.

  The circuit:
  - MKR NB 1500 board
  - Antenna
  - SIM card with a data plan
  - LiPo battery

  This example code is in the public domain.
*/

#include <ArduinoECCX08.h>
#include <utility/ECCX08JWS.h>
#include <ArduinoMqttClient.h>
#include <Arduino_JSON.h>
#include  <MKRNB.h>

#include "arduino_secrets.h"

/////// Enter your sensitive data in
arduino_secrets.h
const char pinnumber[]     = SECRET_PINNUMBER;
```



```
const char projectId[]     = SECRET_PROJECT_ID;
const char cloudRegion[]   = SECRET_CLOUD_REGION;
const char registryId[]    = SECRET_REGISTRY_ID;
const String deviceId      = SECRET_DEVICE_ID;

const char broker[]        = "mqtt.googleapis.com";

NB nbAccess;
GPRS gprs;

NBSSLClient  nbSslClient;
MqttClient   mqttClient(nbSslClient);

unsigned long lastMillis = 0;

void setup() {
  Serial.begin(9600);
  while (!Serial);

  if (!ECCX08.begin()) {
    Serial.println("No ECCX08 present!");
    while (1);
  }

  // Calculate and set the client id used for MQTT
  String clientId = calculateClientId();

  mqttClient.setId(clientId);

  // Set the message callback, this function is
  // called when the MQTTClient receives a message
  mqttClient.onMessage(onMessageReceived);
}

void loop() {
  if (nbAccess.status() != NB_READY || gprs.
status() != GPRS_READY) {
    connectNB();
  }

  if (!mqttClient.connected()) {
    // MQTT client is disconnected, connect
    connectMQTT();
  }

  // poll for new MQTT messages and send keep
alives
  mqttClient.poll();

  // publish a message roughly every 5 seconds.
  if (millis() - lastMillis > 5000) {
    lastMillis = millis();

   publishMessage("1","1.123123","2.123123","60");
  }
}

unsigned long getTime() {
  // get the current time from the cellular module
  return nbAccess.getTime();
}

void connectNB() {
  Serial.println("Attempting to connect to the
cellular network");

  while ((nbAccess.begin(pinnumber) != NB_READY) ||
         (gprs.attachGPRS() != GPRS_READY)) {
    // failed, retry
    Serial.print(".");
    delay(1000);
  }
  Serial.println("You’re connected to the cellular
network");
  Serial.println();
}

void connectMQTT() {
  Serial.print("Attempting to connect to MQTT
broker: ");
  Serial.print(broker);
  Serial.println(" ");

  while (!mqttClient.connected()) {
    // Calculate the JWT and assign it as the
password
    String jwt = calculateJWT();

    mqttClient.setUsernamePassword("", jwt);

    if (!mqttClient.connect(broker, 8883)) {
      // failed, retry
      Serial.print(".");
      delay(5000);
    }
  }
  Serial.println();

  Serial.println("You’re connected to the MQTT
broker");
  Serial.println();

  // subscribe to topics
  //mqttClient.subscribe("/devices/" + deviceId +
"/config", 1);
  //mqttClient.subscribe("/devices/" + deviceId +
"/commands/#");
```

```
  mqttClient.subscribe("/devices/" + deviceId +
"/topics/trye-bike-events");
  //mqttClient.
subscribe("projects/trye-bike-rental/subscriptions/
trye-sub", 1);

}

String calculateClientId() {
  String clientId;

  // Format:
  //
  //
projects/{project-id}/locations/{cloud-region}/regi
stries/{registry-id}/devices/{device-id}
  //

  clientId += "projects/";
  clientId += projectId;
  clientId += "/locations/";
  clientId += cloudRegion;
  clientId += "/registries/";
  clientId += registryId;
  clientId += "/devices/";
  clientId += deviceId;

  return clientId;
}

String calculateJWT() {
  unsigned long now = getTime();

  // calculate the JWT, based on:
  //   https://cloud.google.
com/iot/docs/how-tos/credentials/jwts
```

```
  JSONVar jwtHeader;
  JSONVar jwtClaim;

  jwtHeader["alg"] = "ES256";
  jwtHeader["typ"] = "JWT";

  jwtClaim["aud"] = projectId;
  jwtClaim["iat"] = now;
  jwtClaim["exp"] = now + (24L * 60L * 60L); //
expires in 24 hours

  return ECCX08JWS.sign(0, JSON.
stringify(jwtHeader), JSON.stringify(jwtClaim));
}

void publishMessage(String bikeId = "1", String
lon = "1.1235", String lat = "1.1235", String bat
= "30") {
  Serial.println("Publishing message");

  String message = "{'bikeid':" + bikeId + ",";
  message = message + "'lon':" + lon + ",";
  message = message + "'lat':" + lat + ",";
  message = message + "'bat':" + bat + ",}";


  // send message, the Print interface can be used
to set the message contents
  mqttClient.beginMessage("/devices/" + deviceId +
"/events");
  mqttClient.print(message);
  mqttClient.endMessage();
}

void onMessageReceived(int messageSize) {
  // we received a message, print out the topic
```

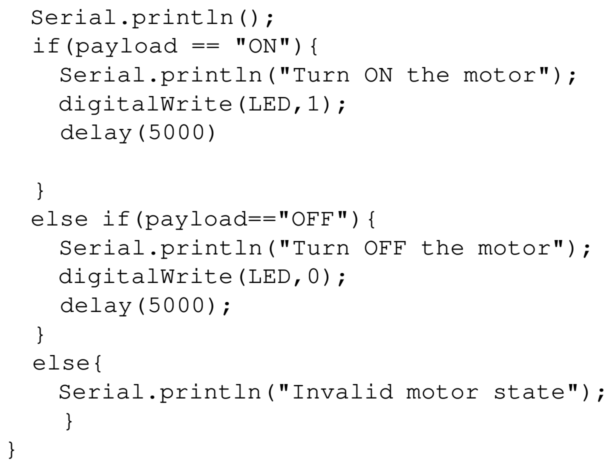
```
and contents
  Serial.print("Received a message with topic ’");
  Serial.print(mqttClient.messageTopic());
  Serial.print("’, length ");
  Serial.print(messageSize);
  Serial.println(" bytes:");

  // use the Stream interface to print the contents
  while (mqttClient.available()) {
    Serial.print((char)mqttClient.read());
  }
  Serial.println();

  Serial.println();
}
```

A.11.7 Motor control with GCP communication

```
#include <ArduinoECCX08.h>
#include <utility/ECCX08JWS.h>
#include <ArduinoMqttClient.h>
#include <Arduino_JSON.h>
#include  <MKRNB.h>
#include <PubSubClient.h>

#include "arduino_secrets.h"

/////// Enter your sensitive data in
arduino_secrets.h
const char pinnumber[]     = SECRET_PINNUMBER;

const char projectId[]     = SECRET_PROJECT_ID;
const char cloudRegion[]   = SECRET_CLOUD_REGION;
const char registryId[]    = SECRET_REGISTRY_ID;
const String deviceId      = SECRET_DEVICE_ID;

const char broker[]        = "mqtt.googleapis.com";

char topic[]= "motor_power";



//void respondToMsg(String msg);
NB nbAccess;
GPRS gprs;
//PubSubClient mqttClient(ethClient);
//EthernetClient ethClient;
//PubSubClient mqttClient(ethClient);
NBSSLClient  nbSslClient;
//PubSubClient  mqttClient(nbSslClient);
MqttClient   mqttClient(nbSslClient);

unsigned long lastMillis = 0;
int LED = 6;
```

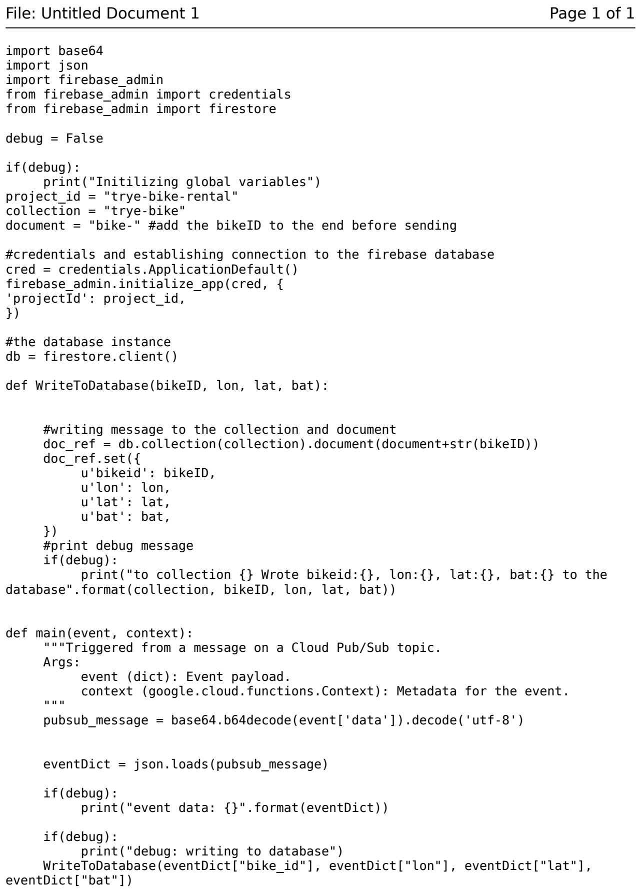
```
void setup() {
  Serial.begin(9600);
  pinMode(LED,OUTPUT);
  while (!Serial);

  if (!ECCX08.begin()) {
    Serial.println("No ECCX08 present!");
    while (1);
  }

  // Calculate and set the client id used for MQTT
  String clientId = calculateClientId();

  mqttClient.setId(clientId);

  // Set the message callback, this function is
  // called when the MQTTClient receives a message
  //callback(char* topic, byte* payload, unsigned
int length)
  //mqttclient.onmessage(callback)
  mqttClient.onMessage(onMessageReceived);
  //mqttClient.setCallback(callback);
  //mqttClient.onMessage(respondToMsg);
}

void loop() {
  if (nbAccess.status() != NB_READY || gprs.
status() != GPRS_READY) {
    connectNB();
  }

  if (!mqttClient.connected()) {
    // MQTT client is disconnected, connect
    connectMQTT();
  }
  // poll for new MQTT messages and send keep
alives
  mqttClient.poll();

  // publish a message roughly every 5 seconds.
  if (millis() - lastMillis > 10000) {
    lastMillis = millis();
      Serial.println("updating gps location and
battery status");
    //publishMessage();
  }
}

unsigned long getTime() {
  // get the current time from the cellular module
  return nbAccess.getTime();
}

void connectNB() {
  Serial.println("Attempting to connect to the
cellular network");

  while ((nbAccess.begin(pinnumber) != NB_READY) ||
         (gprs.attachGPRS() != GPRS_READY)) {
    // failed, retry
    Serial.print(".");
    delay(1000);
  }

  Serial.println("You’re connected to the cellular
network");
  Serial.println();
}

void connectMQTT() {
  Serial.print("Attempting to connect to MQTT
broker: ");
  Serial.print(broker);
  Serial.println(" ");

  while (!mqttClient.connected()) {
    // Calculate the JWT and assign it as the
password
    String jwt = calculateJWT();

    mqttClient.setUsernamePassword("", jwt);

    if (!mqttClient.connect(broker, 8883)) {
      // failed, retry
      Serial.print(".");
      delay(5000);
    }
  }
  Serial.println();

  Serial.println("You’re connected to the MQTT
broker");
  Serial.println();

  // subscribe to topics
  mqttClient.subscribe("/devices/" + deviceId +
"/config", 1);
  mqttClient.subscribe("/devices/" + deviceId +
"/commands/#");
  //mqttClient.subscribe("projects/" + projectId +
"/topics" + "/motor_power");
  mqttClient.subscribe(topic,1);
}

String calculateClientId() {
  String clientId;
  // Format:
  //
  //
projects/{project-id}/locations/{cloud-region}/regi
stries/{registry-id}/devices/{device-id}
  //

  clientId += "projects/";
  clientId += projectId;
  clientId += "/locations/";
  clientId += cloudRegion;
  clientId += "/registries/";
  clientId += registryId;
  clientId += "/devices/";
  clientId += deviceId;

  return clientId;
}

String calculateJWT() {
  unsigned long now = getTime();

  // calculate the JWT, based on:
  //   https://cloud.google.
com/iot/docs/how-tos/credentials/jwts
  JSONVar jwtHeader;
  JSONVar jwtClaim;

  jwtHeader["alg"] = "ES256";
  jwtHeader["typ"] = "JWT";

  jwtClaim["aud"] = projectId;
  jwtClaim["iat"] = now;
  jwtClaim["exp"] = now + (24L * 60L * 60L); //
expires in 24 hours
```

```
  return ECCX08JWS.sign(0, JSON.
stringify(jwtHeader),JSON.stringify(jwtClaim));
}

void publishMessage() {
  Serial.println("Publishing message");

  // send message, the Print interface can be used
to set the message contents
  mqttClient.beginMessage("/devices/" + deviceId +
"/state");
  mqttClient.print("these are your sensor datas ");
  mqttClient.print(millis());
  mqttClient.endMessage();
}

void onMessageReceived(int messageSize) {
  // we received a message, print out the topic
and contents
  Serial.print("Received a message with topic '");
  Serial.print(mqttClient.messageTopic());
  Serial.print("', length ");
  Serial.print(messageSize);
  Serial.println(" bytes:");
   Serial.println("power configuration message is
received");
//  Serial.println(String(payload));
  String payload ="";
  // use the Stream interface to print the contents
  for(int i=0; i< messageSize; i++)
     while (mqttClient.available()) {
    //Serial.print((char)mqttClient.read());
     payload += ((char)mqttClient.read());
  }
```



```
  Serial.println();
  if(payload == "ON"){
    Serial.println("Turn ON the motor");
    digitalWrite(LED,1);
    delay(5000)

  }
  else if(payload=="OFF"){
    Serial.println("Turn OFF the motor");
    digitalWrite(LED,0);
    delay(5000);
  }
  else{
    Serial.println("Invalid motor state");
    }
}
```

A.11.8 Pub/Sub to Firestore database

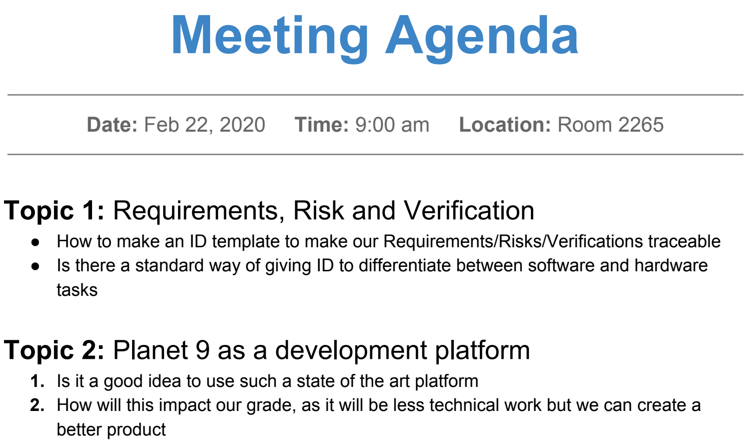


File: Untitled Document 1 Page 1 of 1

```
import base64
import json
import firebase_admin
from firebase_admin import credentials
from firebase_admin import firestore

debug = False

if(debug):
     print("Initilizing global variables")
project_id = "trye-bike-rental"
collection = "trye-bike"
document = "bike-" #add the bikeID to the end before sending

#credentials and establishing connection to the firebase database
cred = credentials.ApplicationDefault()
firebase_admin.initialize_app(cred, {
'projectId': project_id,
})

#the database instance
db = firestore.client()

def WriteToDatabase(bikeID, lon, lat, bat):


     #writing message to the collection and document
     doc_ref = db.collection(collection).document(document+str(bikeID))
     doc_ref.set({
          u'bikeid': bikeID,
          u'lon': lon,
          u'lat': lat,
          u'bat': bat,
     })
     #print debug message
     if(debug):
          print("to collection {} Wrote bikeid:{}, lon:{}, lat:{}, bat:{} to the 
database".format(collection, bikeID, lon, lat, bat))


def main(event, context):
     """Triggered from a message on a Cloud Pub/Sub topic.
     Args:
          event (dict): Event payload.
          context (google.cloud.functions.Context): Metadata for the event.
     """
     pubsub_message = base64.b64decode(event['data']).decode('utf-8')


     eventDict = json.loads(pubsub_message)

     if(debug):
          print("event data: {}".format(eventDict))

     if(debug):
          print("debug: writing to database")
     WriteToDatabase(eventDict["bike_id"], eventDict["lon"], eventDict["lat"], 
eventDict["bat"])
```

A.12 Agendas

A.12.1 24th of January Agenda for Jose



Meeting Agenda

Date: Feb 22, 2020 **Time:** 9:00 am **Location:** Room 2265

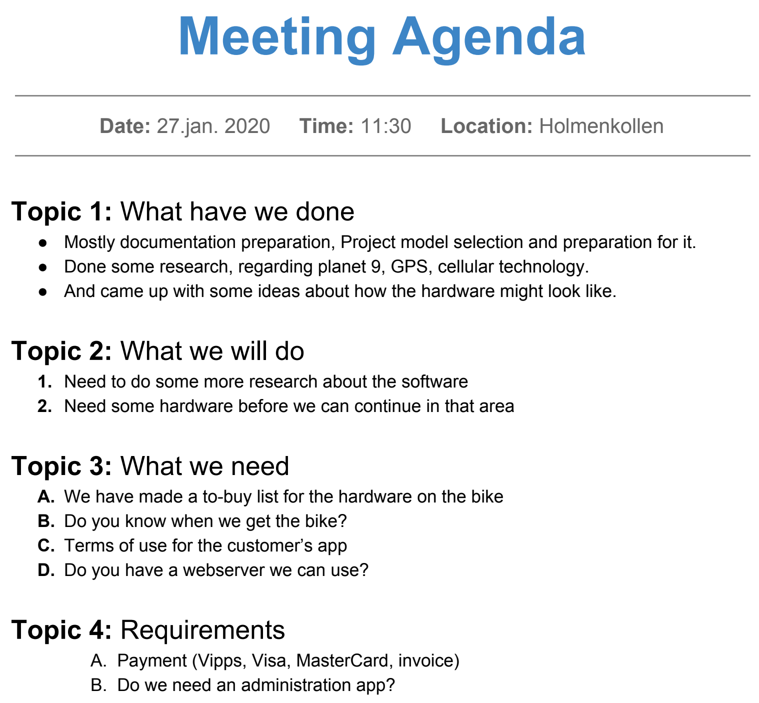
Topic 1: Requirements, Risk and Verification

- How to make an ID template to make our Requirements/Risks/Verifications traceable
- Is there a standard way of giving ID to differentiate between software and hardware tasks

Topic 2: Planet 9 as a development platform

1. Is it a good idea to use such a state of the art platform
2. How will this impact our grade, as it will be less technical work but we can create a better product

A.12.2 27th of January Agenda for Simon



Meeting Agenda

Date: 27.jan. 2020 **Time:** 11:30 **Location:** Holmenkollen

Topic 1: What have we done

- Mostly documentation preparation, Project model selection and preparation for it.
- Done some research, regarding planet 9, GPS, cellular technology.
- And came up with some ideas about how the hardware might look like.

Topic 2: What we will do

1. Need to do some more research about the software
2. Need some hardware before we can continue in that area

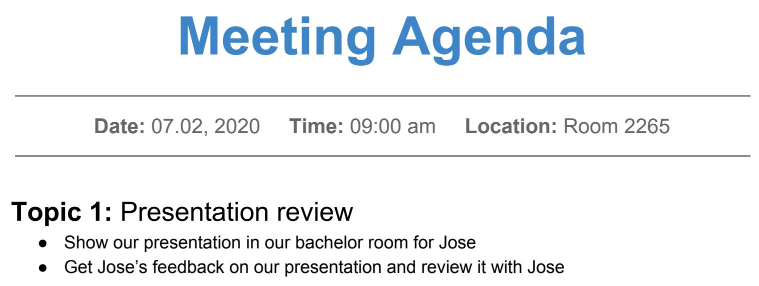
Topic 3: What we need

A. We have made a to-buy list for the hardware on the bike
B. Do you know when we get the bike?
C. Terms of use for the customer's app
D. Do you have a webserver we can use?

Topic 4: Requirements

A. Payment (Vipps, Visa, MasterCard, invoice)
B. Do we need an administration app?

A.12.3 7th of February Agenda for Jose



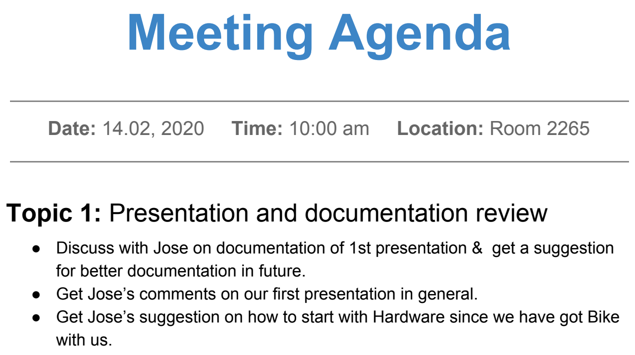
Meeting Agenda

Date: 07.02, 2020 **Time:** 09:00 am **Location:** Room 2265

Topic 1: Presentation review

- Show our presentation in our bachelor room for Jose
- Get Jose's feedback on our presentation and review it with Jose

A.12.4 14th of February Agenda for Jose



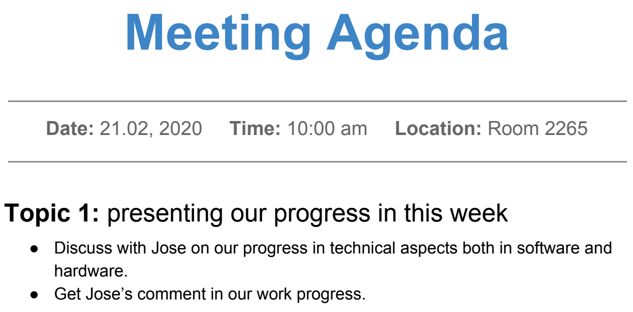
Meeting Agenda

Date: 14.02, 2020 **Time:** 10:00 am **Location:** Room 2265

Topic 1: Presentation and documentation review

- Discuss with Jose on documentation of 1st presentation & get a suggestion for better documentation in future.
- Get Jose’s comments on our first presentation in general.
- Get Jose’s suggestion on how to start with Hardware since we have got Bike with us.

A.12.5 21th of February Agenda for Jose



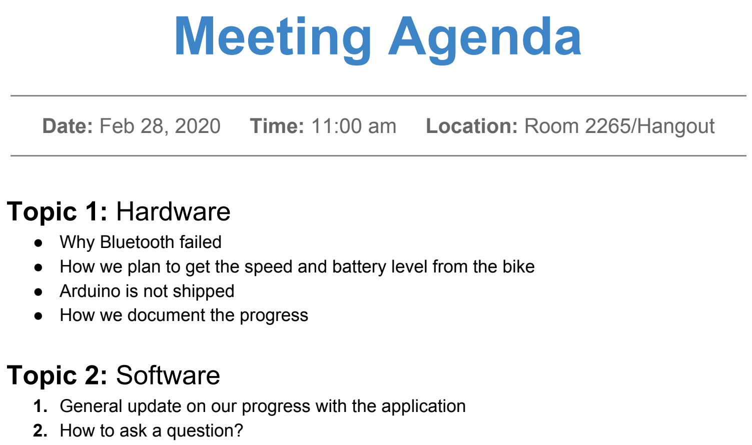
Meeting Agenda

Date: 21.02, 2020 **Time:** 10:00 am **Location:** Room 2265

Topic 1: presenting our progress in this week

- Discuss with Jose on our progress in technical aspects both in software and hardware.
- Get Jose’s comment in our work progress.

A.12.6 28th of February Agenda for Jose



Meeting Agenda

Date: Feb 28, 2020 **Time:** 11:00 am **Location:** Room 2265/Hangout

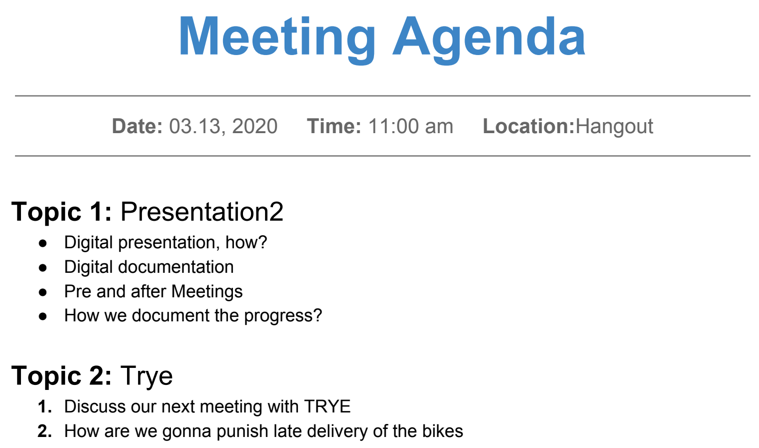
Topic 1: Hardware

- Why Bluetooth failed
- How we plan to get the speed and battery level from the bike
- Arduino is not shipped
- How we document the progress

Topic 2: Software

1. General update on our progress with the application
2. How to ask a question?

A.12.7 13th of March Agenda for Jose



Meeting Agenda

Date: 03.13, 2020 **Time:** 11:00 am **Location:**Hangout

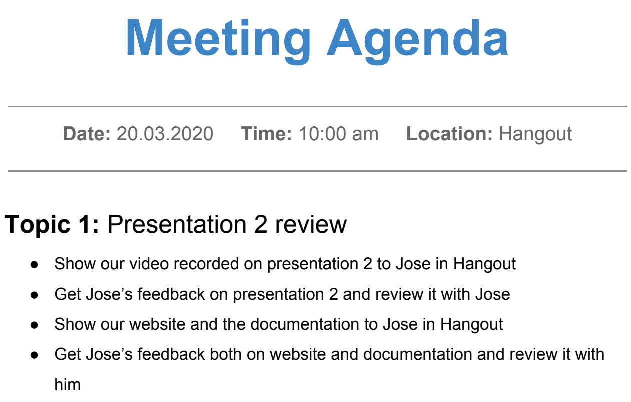
Topic 1: Presentation2

- Digital presentation, how?
- Digital documentation
- Pre and after Meetings
- How we document the progress?

Topic 2: Trye

1. Discuss our next meeting with TRYE
2. How are we gonna punish late delivery of the bikes

A.12.8 18th of March Agenda for Jose



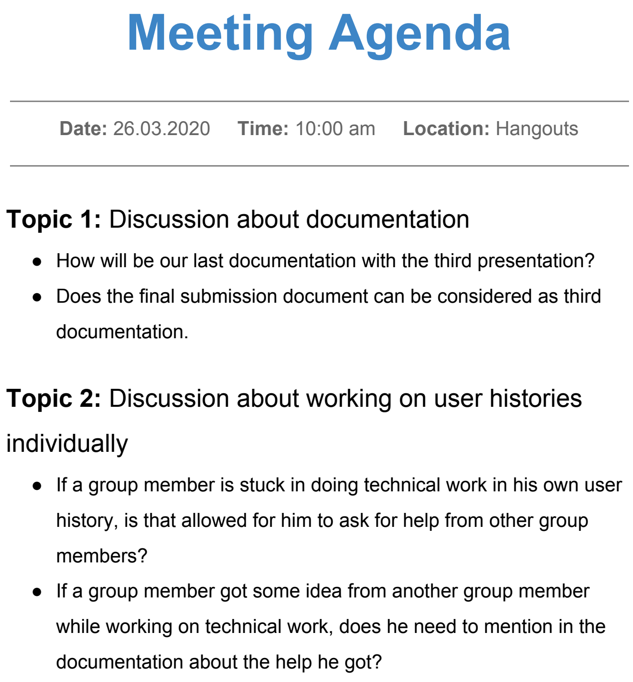
Meeting Agenda

Date: 20.03.2020 **Time:** 10:00 am **Location:** Hangout

Topic 1: Presentation 2 review

- Show our video recorded on presentation 2 to Jose in Hangout
- Get Jose's feedback on presentation 2 and review it with Jose
- Show our website and the documentation to Jose in Hangout
- Get Jose's feedback both on website and documentation and review it with him

A.12.9 26th of March Agenda for Jose



Meeting Agenda

Date: 26.03.2020 **Time:** 10:00 am **Location:** Hangouts

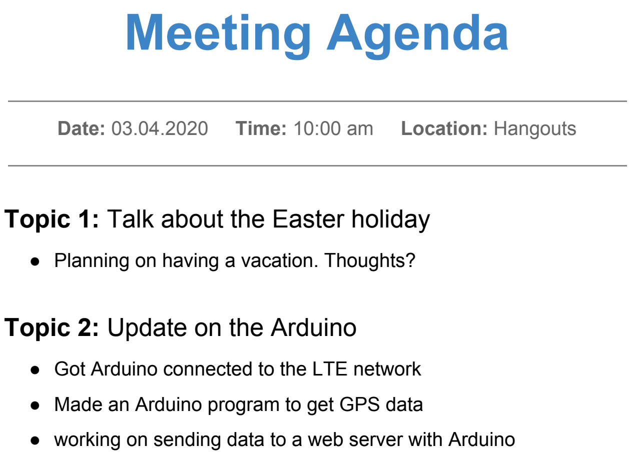
Topic 1: Discussion about documentation

- How will be our last documentation with the third presentation?
- Does the final submission document can be considered as third documentation.

Topic 2: Discussion about working on user histories individually

- If a group member is stuck in doing technical work in his own user history, is that allowed for him to ask for help from other group members?
- If a group member got some idea from another group member while working on technical work, does he need to mention in the documentation about the help he got?

A.12.10 3rd of April Agenda for Jose



Meeting Agenda

Date: 03.04.2020 **Time:** 10:00 am **Location:** Hangouts

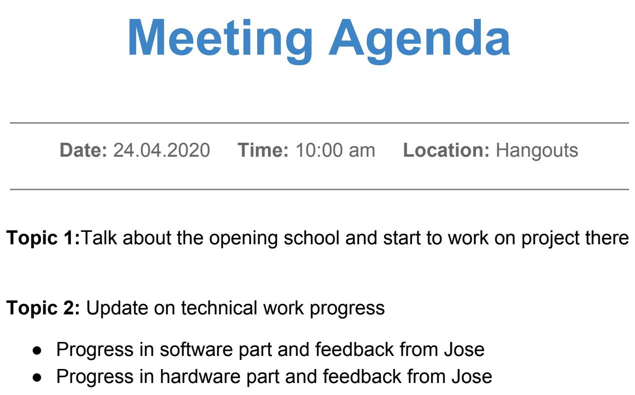
Topic 1: Talk about the Easter holiday

- Planning on having a vacation. Thoughts?

Topic 2: Update on the Arduino

- Got Arduino connected to the LTE network
- Made an Arduino program to get GPS data
- working on sending data to a web server with Arduino

A.12.11 24th of April Agenda for Jose



Meeting Agenda

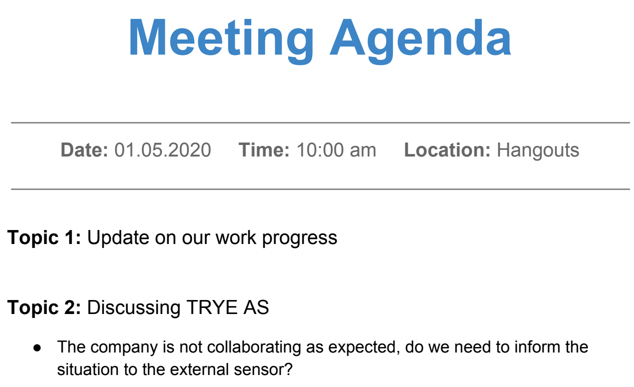
Date: 24.04.2020 **Time:** 10:00 am **Location:** Hangouts

Topic 1: Talk about the opening school and start to work on project there

Topic 2: Update on technical work progress

- Progress in software part and feedback from Jose
- Progress in hardware part and feedback from Jose

A.12.12 1st of May Agenda for Jose



Meeting Agenda

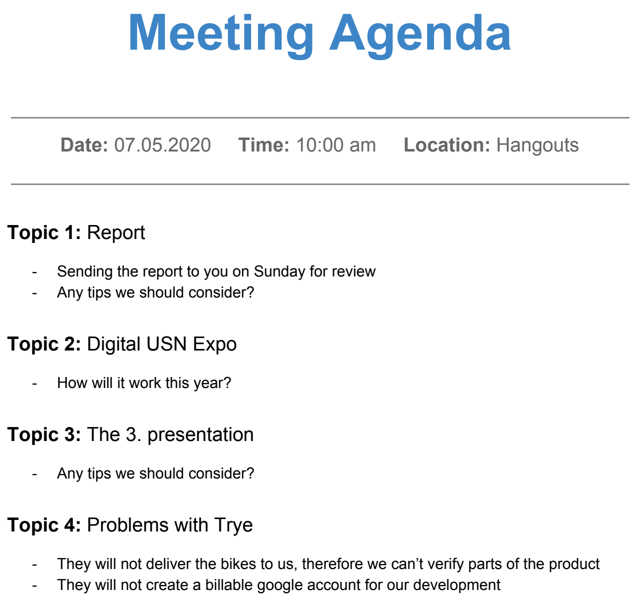
Date: 01.05.2020 **Time:** 10:00 am **Location:** Hangouts

Topic 1: Update on our work progress

Topic 2: Discussing TRYE AS

- The company is not collaborating as expected, do we need to inform the situation to the external sensor?

A.12.13 7th of May Agenda for Jose



Meeting Agenda

Date: 07.05.2020 **Time:** 10:00 am **Location:** Hangouts

Topic 1: Report

- Sending the report to you on Sunday for review
- Any tips we should consider?

Topic 2: Digital USN Expo

- How will it work this year?

Topic 3: The 3. presentation

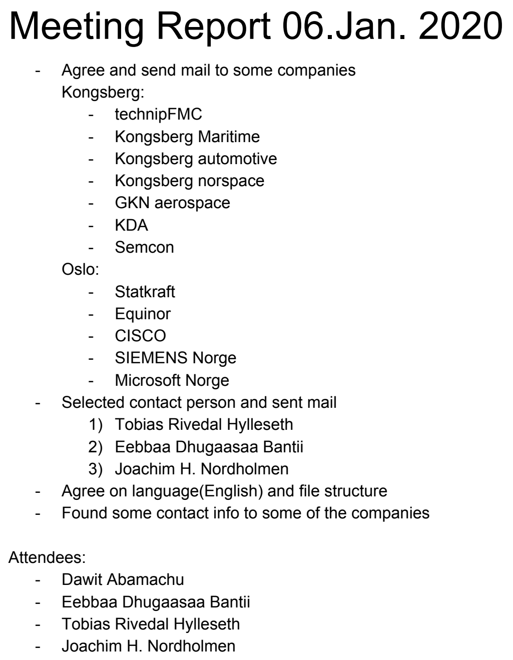
- Any tips we should consider?

Topic 4: Problems with Trye

- They will not deliver the bikes to us, therefore we can't verify parts of the product
- They will not create a billable google account for our development

A.13 Meeting notes

A.13.1 6th of January Group Meeting



Meeting Report 06.Jan. 2020

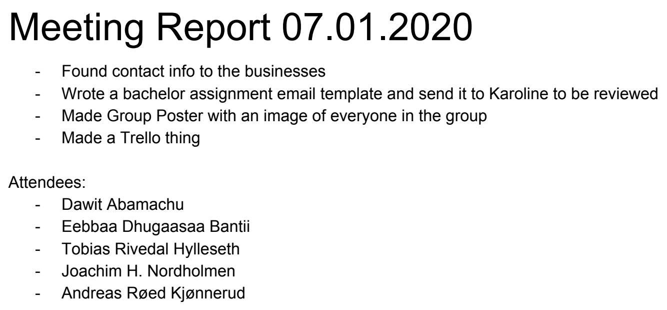
- Agree and send mail to some companies
  Kongsberg:
  - technipFMC
  - Kongsberg Maritime
  - Kongsberg automotive
  - Kongsberg norspace
  - GKN aerospace
  - KDA
  - Semcon

  Oslo:
  - Statkraft
  - Equinor
  - CISCO
  - SIEMENS Norge
  - Microsoft Norge
- Selected contact person and sent mail
  1) Tobias Rivedal Hylleseth
  2) Eebbaa Dhugaasaa Bantii
  3) Joachim H. Nordholmen
- Agree on language(English) and file structure
- Found some contact info to some of the companies

Attendees:

- Dawit Abamachu
- Eebbaa Dhugaasaa Bantii
- Tobias Rivedal Hylleseth
- Joachim H. Nordholmen

A.13.2 7th of January Group Meeting



Meeting Report 07.01.2020

- Found contact info to the businesses
- Wrote a bachelor assignment email template and send it to Karoline to be reviewed
- Made Group Poster with an image of everyone in the group
- Made a Trello thing

Attendees:

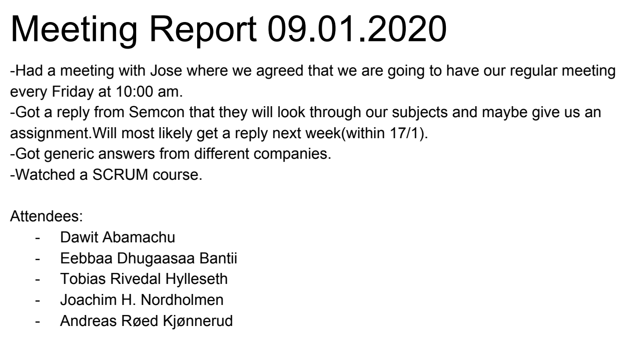
- Dawit Abamachu
- Eebbaa Dhugaasaa Bantii
- Tobias Rivedal Hylleseth
- Joachim H. Nordholmen
- Andreas Røed Kjønnerud

A.13.3 8th of January Group Meeting

Meeting report 08.01.2020

- Sendt mail to TechnipFMC, Kongsberg Maritime and KDA.
- Called semcon and Statkraft Oslo.
- Send mail to Richard to get computer screens

A.13.4 9th of January Group Meeting



Meeting Report 09.01.2020

-Had a meeting with Jose where we agreed that we are going to have our regular meeting every Friday at 10:00 am.
-Got a reply from Semcon that they will look through our subjects and maybe give us an assignment.Will most likely get a reply next week(within 17/1).
-Got generic answers from different companies.
-Watched a SCRUM course.

Attendees:

- Dawit Abamachu
- Eebbaa Dhugaasaa Bantii
- Tobias Rivedal Hylleseth
- Joachim H. Nordholmen
- Andreas Røed Kjønnerud

A.13.5 10th of January Group Meeting

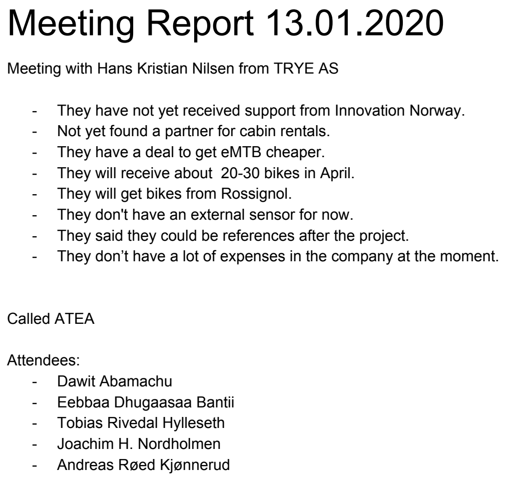
Meeting Report 10.01.2020

We started with discussing and taking roles for the project and started with some documentation work.

Attendees:

- Dawit Abamachu
- Eebbaa Dhugaasaa Bantii
- Tobias Rivedal Hylleseth
- Joachim H. Nordholmen
- Andreas Røed Kjønnerud

A.13.6 13th of January Group Meeting



Meeting Report 13.01.2020

Meeting with Hans Kristian Nilsen from TRYE AS

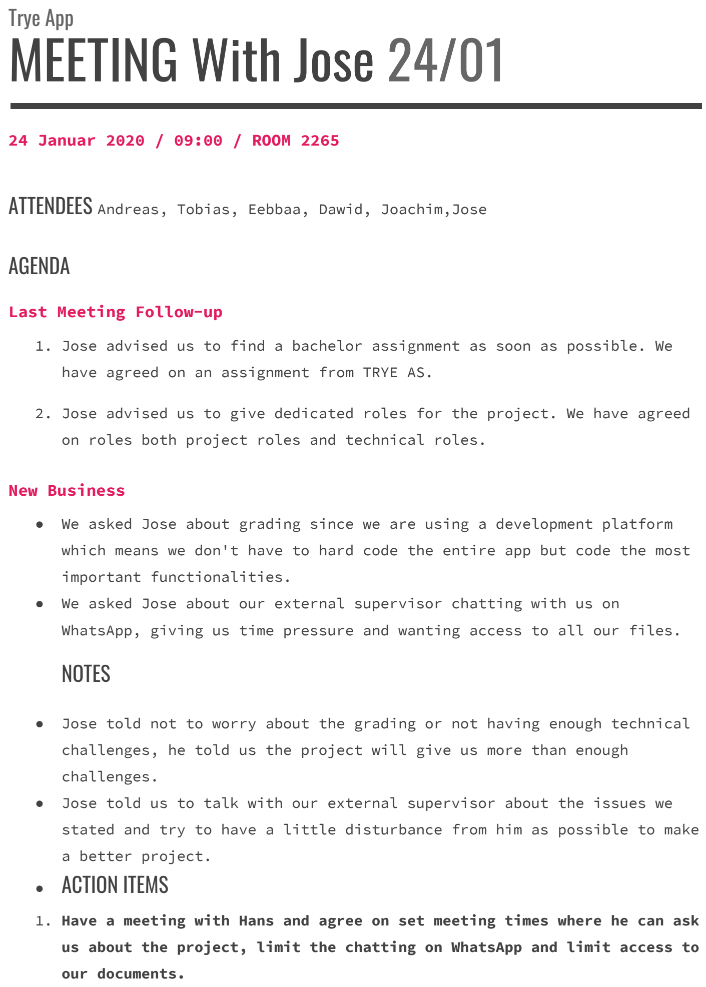
- They have not yet received support from Innovation Norway.
- Not yet found a partner for cabin rentals.
- They have a deal to get eMTB cheaper.
- They will receive about 20-30 bikes in April.
- They will get bikes from Rossignol.
- They don't have an external sensor for now.
- They said they could be references after the project.
- They don’t have a lot of expenses in the company at the moment.

Called ATEA

Attendees:

- Dawit Abamachu
- Eebbaa Dhugaasaa Bantii
- Tobias Rivedal Hylleseth
- Joachim H. Nordholmen
- Andreas Røed Kjønnerud

A.13.7 24th of January Meeting with Jose



Trye App

MEETING With Jose 24/01

24 Januar 2020 / 09:00 / ROOM 2265

ATTENDEES Andreas, Tobias, Eebbaa, Dawid, Joachim,Jose

AGENDA

Last Meeting Follow-up

1. Jose advised us to find a bachelor assignment as soon as possible. We have agreed on an assignment from TRYE AS.
2. Jose advised us to give dedicated roles for the project. We have agreed on roles both project roles and technical roles.

New Business

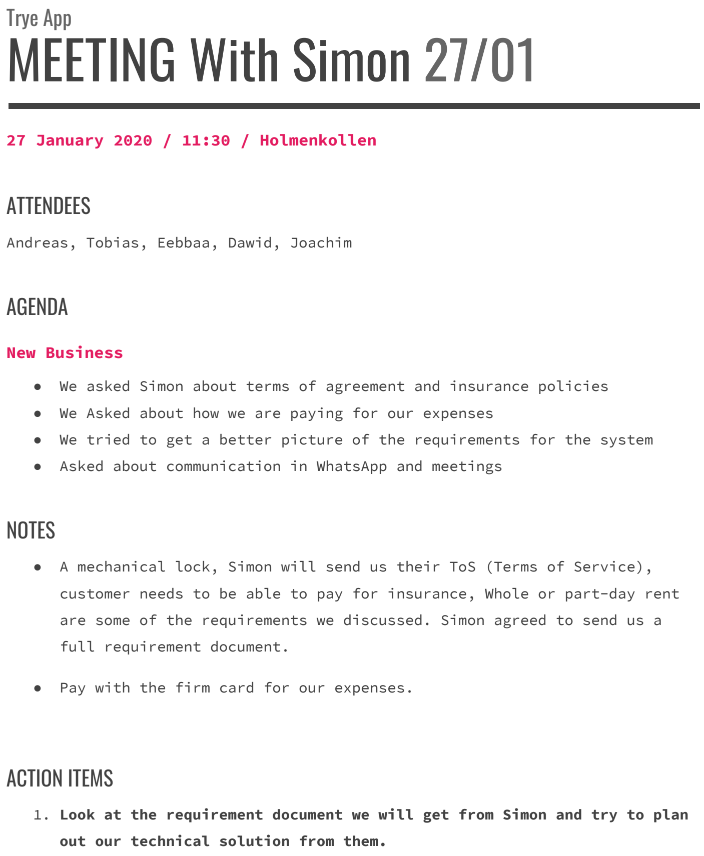
- We asked Jose about grading since we are using a development platform which means we don't have to hard code the entire app but code the most important functionalities.
- We asked Jose about our external supervisor chatting with us on WhatsApp, giving us time pressure and wanting access to all our files.

NOTES

- Jose told not to worry about the grading or not having enough technical challenges, he told us the project will give us more than enough challenges.
- Jose told us to talk with our external supervisor about the issues we stated and try to have a little disturbance from him as possible to make a better project.
- ACTION ITEMS

1. **Have a meeting with Hans and agree on set meeting times where he can ask us about the project, limit the chatting on WhatsApp and limit access to our documents.**

A.13.8 27th of January Meeting with TRYE



Trye App

MEETING With Simon 27/01

27 January 2020 / 11:30 / Holmenkollen

ATTENDEES

Andreas, Tobias, Eebbaa, Dawid, Joachim

AGENDA

New Business

- We asked Simon about terms of agreement and insurance policies
- We Asked about how we are paying for our expenses
- We tried to get a better picture of the requirements for the system
- Asked about communication in WhatsApp and meetings

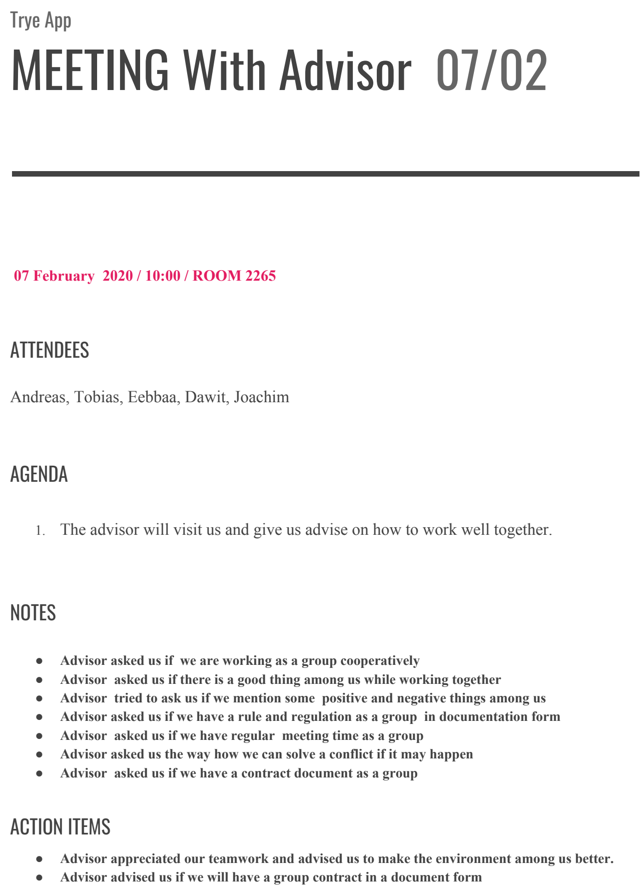
NOTES

- A mechanical lock, Simon will send us their ToS (Terms of Service), customer needs to be able to pay for insurance, Whole or part-day rent are some of the requirements we discussed. Simon agreed to send us a full requirement document.
- Pay with the firm card for our expenses.

ACTION ITEMS

1. **Look at the requirement document we will get from Simon and try to plan out our technical solution from them.**

A.13.9 7th of February Meeting with advisor



Trye App

MEETING With Advisor 07/02

07 February 2020 / 10:00 / ROOM 2265

ATTENDEES

Andreas, Tobias, Eebbaa, Dawit, Joachim

AGENDA

1. The advisor will visit us and give us advise on how to work well together.

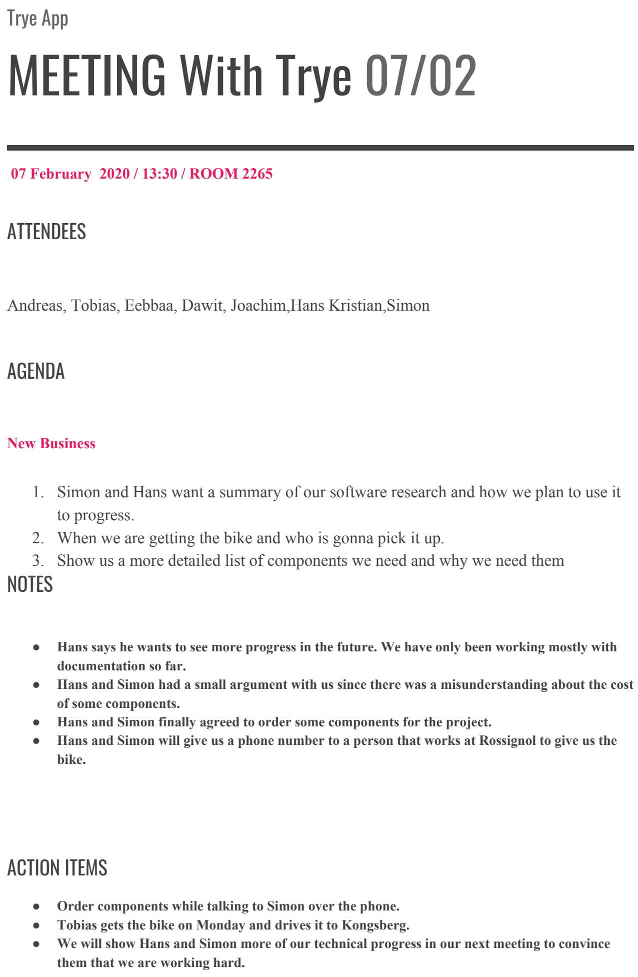
NOTES

- **Advisor asked us if we are working as a group cooperatively**
- **Advisor asked us if there is a good thing among us while working together**
- **Advisor tried to ask us if we mention some positive and negative things among us**
- **Advisor asked us if we have a rule and regulation as a group in documentation form**
- **Advisor asked us if we have regular meeting time as a group**
- **Advisor asked us the way how we can solve a conflict if it may happen**
- **Advisor asked us if we have a contract document as a group**

ACTION ITEMS

- **Advisor appreciated our teamwork and advised us to make the environment among us better.**
- **Advisor advised us if we will have a group contract in a document form**

A.13.10 7th of February Meeting with Hans Kristian



Trye App

MEETING With Trye 07/02

07 February 2020 / 13:30 / ROOM 2265

ATTENDEES

Andreas, Tobias, Eebbaa, Dawit, Joachim,Hans Kristian,Simon

AGENDA

New Business

1. Simon and Hans want a summary of our software research and how we plan to use it to progress.
2. When we are getting the bike and who is gonna pick it up.
3. Show us a more detailed list of components we need and why we need them

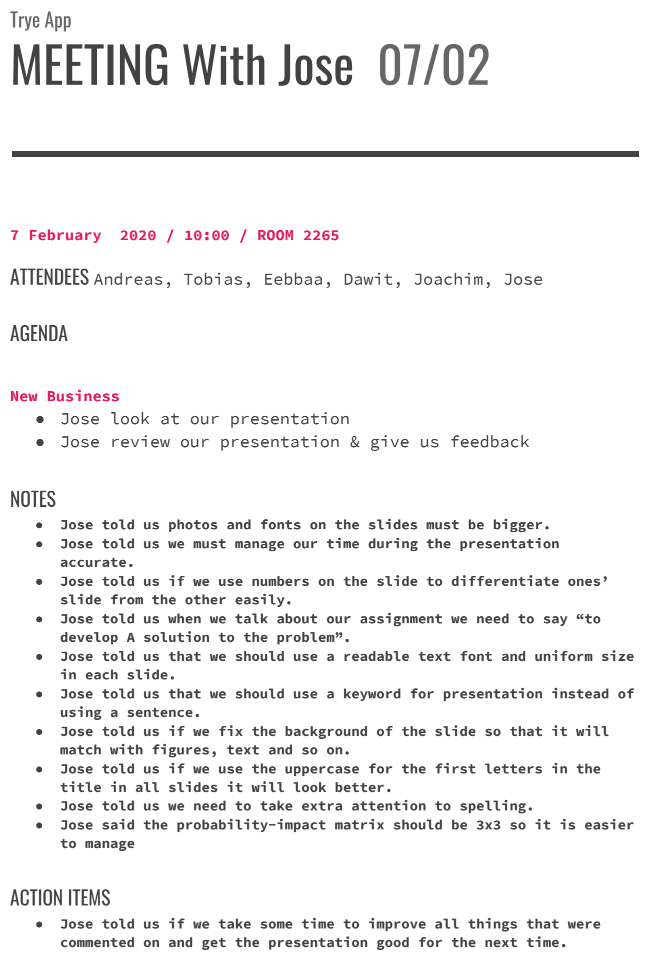
NOTES

- **Hans says he wants to see more progress in the future. We have only been working mostly with documentation so far.**
- **Hans and Simon had a small argument with us since there was a misunderstanding about the cost of some components.**
- **Hans and Simon finally agreed to order some components for the project.**
- **Hans and Simon will give us a phone number to a person that works at Rossignol to give us the bike.**

ACTION ITEMS

- **Order components while talking to Simon over the phone.**
- **Tobias gets the bike on Monday and drives it to Kongsberg.**
- **We will show Hans and Simon more of our technical progress in our next meeting to convince them that we are working hard.**

A.13.11 7th of February Meeting with Jose



Trye App

MEETING With Jose 07/02

7 February 2020 / 10:00 / ROOM 2265

ATTENDEES Andreas, Tobias, Eebbaa, Dawit, Joachim, Jose

AGENDA

New Business

- Jose look at our presentation
- Jose review our presentation & give us feedback

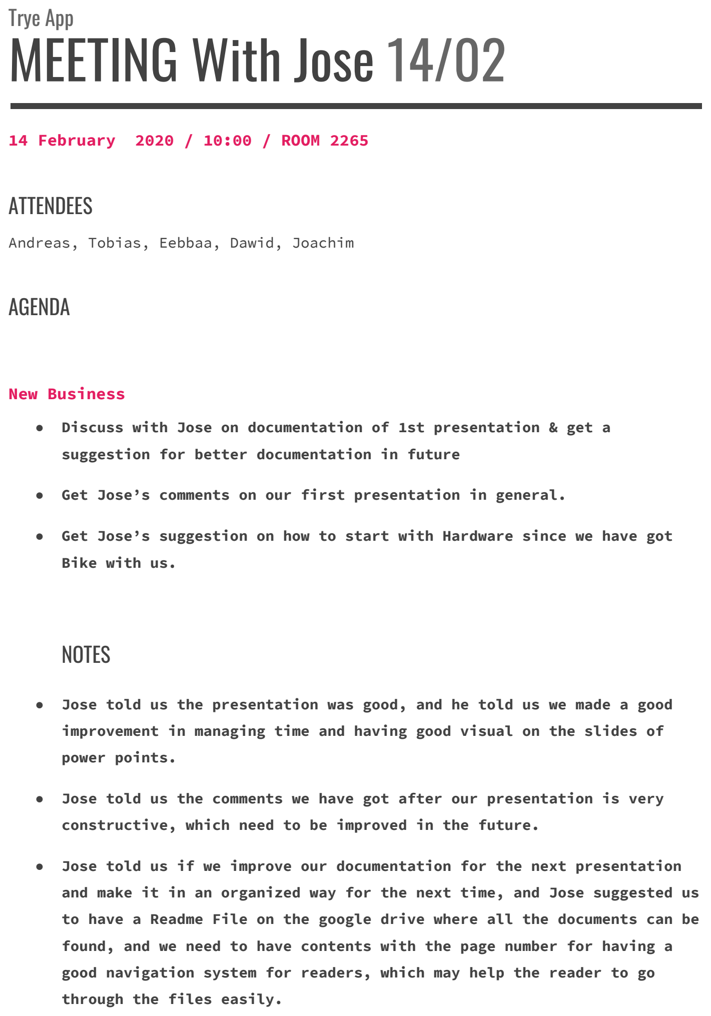
NOTES

- **Jose told us photos and fonts on the slides must be bigger.**
- **Jose told us we must manage our time during the presentation accurate.**
- **Jose told us if we use numbers on the slide to differentiate ones' slide from the other easily.**
- **Jose told us when we talk about our assignment we need to say "to develop A solution to the problem".**
- **Jose told us that we should use a readable text font and uniform size in each slide.**
- **Jose told us that we should use a keyword for presentation instead of using a sentence.**
- **Jose told us if we fix the background of the slide so that it will match with figures, text and so on.**
- **Jose told us if we use the uppercase for the first letters in the title in all slides it will look better.**
- **Jose told us we need to take extra attention to spelling.**
- **Jose said the probability-impact matrix should be 3x3 so it is easier to manage**

ACTION ITEMS

- **Jose told us if we take some time to improve all things that were commented on and get the presentation good for the next time.**

A.13.12 14th of February Meeting with Jose



Trye App

MEETING With Jose 14/02

14 February 2020 / 10:00 / ROOM 2265

ATTENDEES

Andreas, Tobias, Eebbaa, Dawid, Joachim

AGENDA

New Business

- **Discuss with Jose on documentation of 1st presentation & get a suggestion for better documentation in future**
- **Get Jose's comments on our first presentation in general.**
- **Get Jose's suggestion on how to start with Hardware since we have got Bike with us.**

NOTES

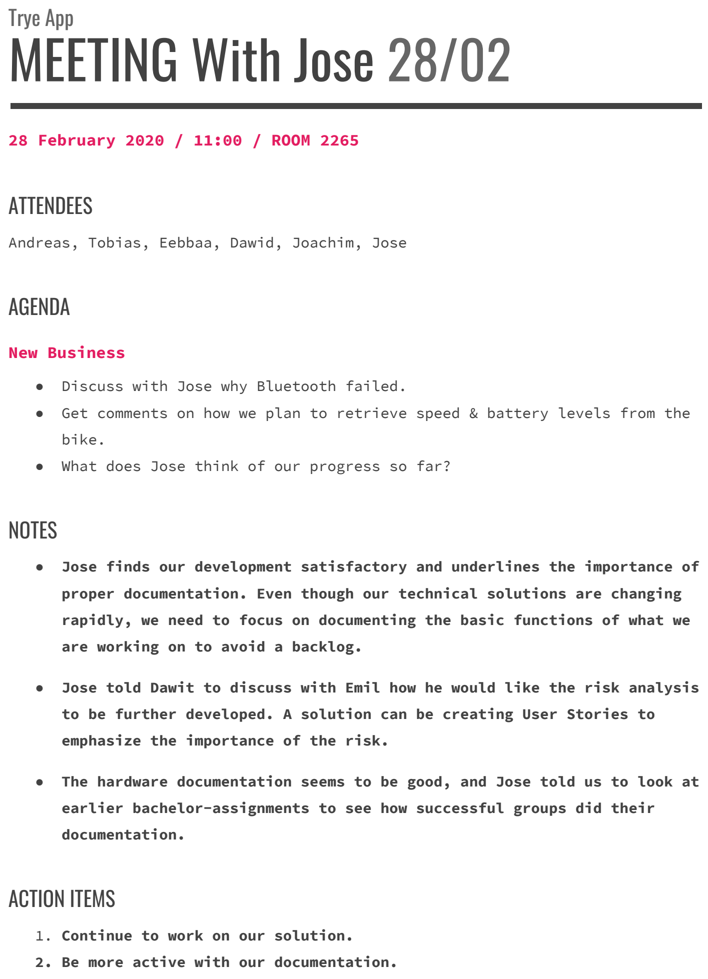
- **Jose told us the presentation was good, and he told us we made a good improvement in managing time and having good visual on the slides of power points.**
- **Jose told us the comments we have got after our presentation is very constructive, which need to be improved in the future.**
- **Jose told us if we improve our documentation for the next presentation and make it in an organized way for the next time, and Jose suggested us to have a Readme File on the google drive where all the documents can be found, and we need to have contents with the page number for having a good navigation system for readers, which may help the reader to go through the files easily.**

- **Jose told us before we deliver the documentation, he can review it and give us comments for improvement, so he wants to send him first before we deliver our documentation.**
- **Jose told us Regarding the Hardware part(Bike) to find out more about the bike system and how the bike computer works.**
- **Jose told us we need to find out how we can extract the data we need from the bike computer system, he told us we need to work more on this part.**
- **Jose told us, we need to think to build a system that can be used for different electric mountain bike brands, not only for one Bike brand this is because as system developer we need to build a system that is modular, instead of designing for only one specific bike.**
- **Jose told us to have a meeting with us at 10:00 on next Friday through google hangout.**

ACTION ITEMS

1. **We need to work on the bike for understanding more about the bike system.**
2. **We need to borrow oscilloscope from the Electro lab for working on the Bike electric system.**
3. **We have decided to contact a company called Shimano which delivers Electric bike components and it is located in Asker. The company has all the different parts for sell, and we can have a visit to see each component of the bike system and talk with them for getting some technical explanation.**

A.13.13 28th of February Meeting with Jose



Trye App

MEETING With Jose 28/02

28 February 2020 / 11:00 / ROOM 2265

ATTENDEES

Andreas, Tobias, Eebbaa, Dawid, Joachim, Jose

AGENDA

New Business

- Discuss with Jose why Bluetooth failed.
- Get comments on how we plan to retrieve speed & battery levels from the bike.
- What does Jose think of our progress so far?

NOTES

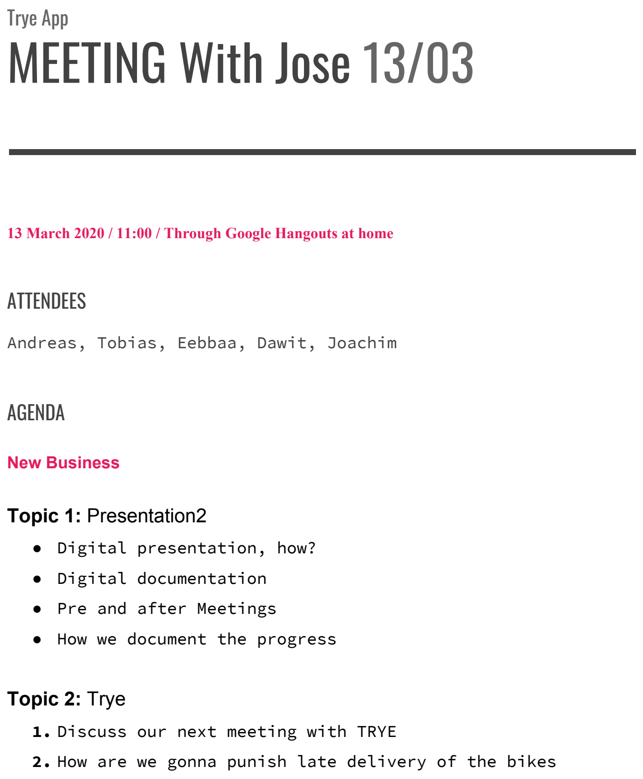
- **Jose finds our development satisfactory and underlines the importance of proper documentation. Even though our technical solutions are changing rapidly, we need to focus on documenting the basic functions of what we are working on to avoid a backlog.**
- **Jose told Dawit to discuss with Emil how he would like the risk analysis to be further developed. A solution can be creating User Stories to emphasize the importance of the risk.**
- **The hardware documentation seems to be good, and Jose told us to look at earlier bachelor-assignments to see how successful groups did their documentation.**

ACTION ITEMS

1. **Continue to work on our solution.**
2. **Be more active with our documentation.**

3. Do not hesitate to ask the guys at Neptune-Software for help when we struggle with their software.
4. Try to find out why our Arduino IoT is in quarantine (possibly because of lack of spare parts from China/Coronavirus?).

A.13.14 13th of March Meeting with Jose



Trye App

MEETING With Jose 13/03

13 March 2020 / 11:00 / Through Google Hangouts at home

ATTENDEES

Andreas, Tobias, Eebbaa, Dawit, Joachim

AGENDA

New Business

Topic 1: Presentation2

- Digital presentation, how?
- Digital documentation
- Pre and after Meetings
- How we document the progress

Topic 2: Trye

1. Discuss our next meeting with TRYE
2. How are we gonna punish late delivery of the bikes

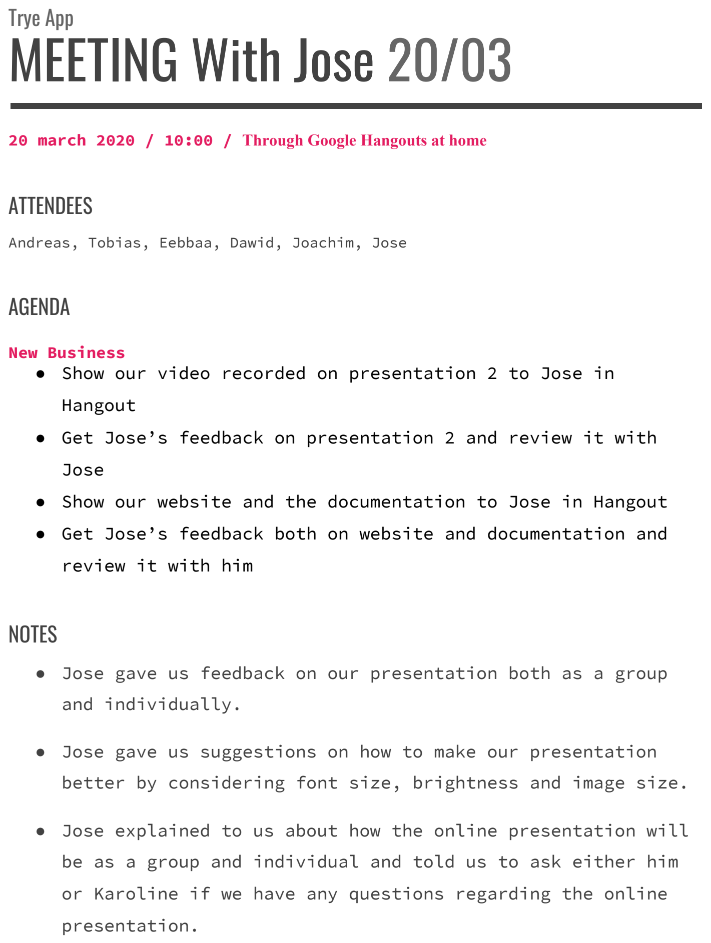
NOTES :

- Jose advised us to make a video for the 2nd presentation which will be held on 24th March 2020
- Jose advised us to organize the online meetings by the help of skype for business
- Jose told us if we send him the 2nd presentation video of the 1st version before our next meetings on Friday so that he will give us feedback or comments
- Jose recommended us to make a website for digital documentation. This is because it helps to navigate and understand the document easily
- Jose recommended us to use WordPress for making a website
- Jose advised and showed us how to organize the meetings using skype for business
- Jose emphasized us to make good documentation, and avoid the spelling errors.

ACTION ITEMS

1. Keep working hard on the technical part and concentrate on documentation
2. Look at different options on a punishing system for those who may be delivering the bike late as a customer

A.13.15 20th of March Meeting with Jose



Trye App

MEETING With Jose 20/03

20 march 2020 / 10:00 / Through Google Hangouts at home

ATTENDEES

Andreas, Tobias, Eebbaa, Dawid, Joachim, Jose

AGENDA

New Business

- Show our video recorded on presentation 2 to Jose in Hangout
- Get Jose's feedback on presentation 2 and review it with Jose
- Show our website and the documentation to Jose in Hangout
- Get Jose's feedback both on website and documentation and review it with him

NOTES

- Jose gave us feedback on our presentation both as a group and individually.
- Jose gave us suggestions on how to make our presentation better by considering font size, brightness and image size.
- Jose explained to us about how the online presentation will be as a group and individual and told us to ask either him or Karoline if we have any questions regarding the online presentation.

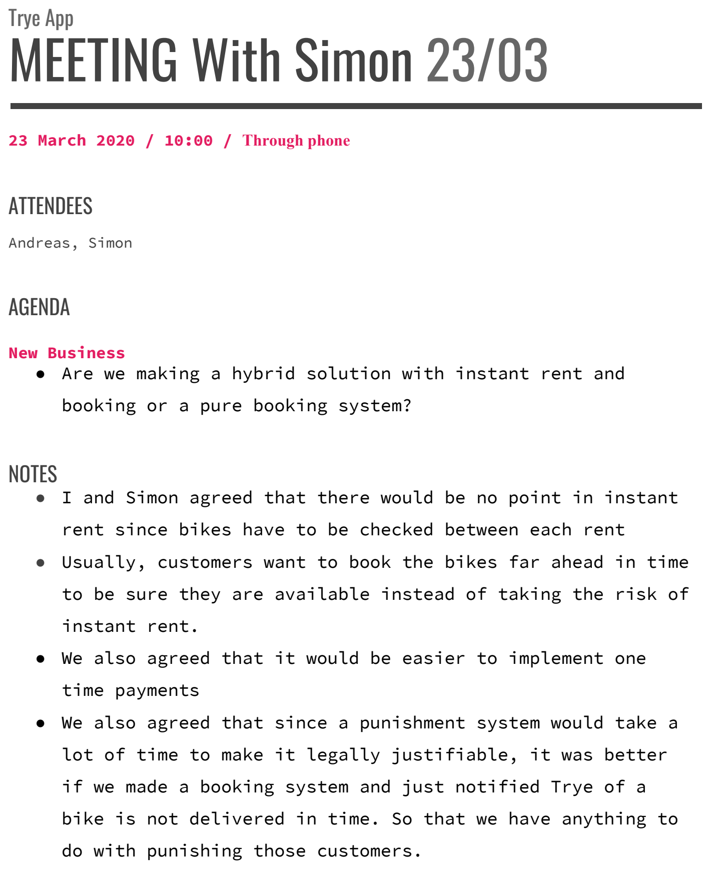
- Jose gave us feedback about our documentation website and how to have better navigation and he showed us an example.

ACTION ITEMS

1. The group decided to make a new video based on the feedback from Jose.

2. The group decides to work on the documentation based on the feedback from Jose.

A.13.16 23th of March Meeting with Simon



Trye App

MEETING With Simon 23/03

23 March 2020 / 10:00 / Through phone

ATTENDEES

Andreas, Simon

AGENDA

New Business

- Are we making a hybrid solution with instant rent and booking or a pure booking system?

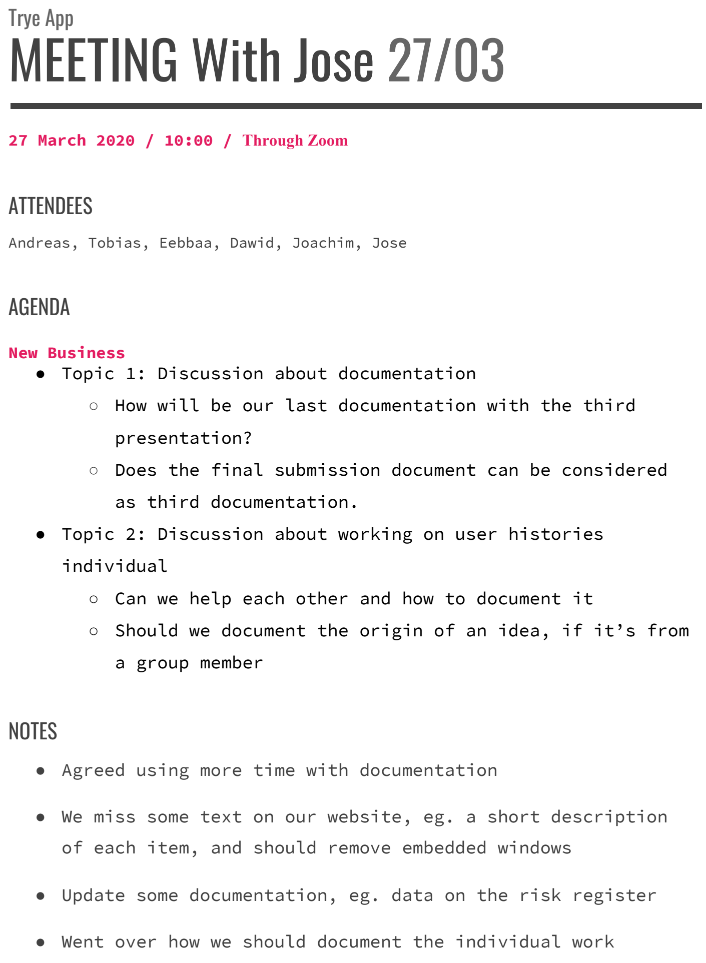
NOTES

- I and Simon agreed that there would be no point in instant rent since bikes have to be checked between each rent
- Usually, customers want to book the bikes far ahead in time to be sure they are available instead of taking the risk of instant rent.
- We also agreed that it would be easier to implement one time payments
- We also agreed that since a punishment system would take a lot of time to make it legally justifiable, it was better if we made a booking system and just notified Trye of a bike is not delivered in time. So that we have anything to do with punishing those customers.

ACTION ITEMS

1) Make a pure booking system
2) Make one time payments
3) Change the map to only show rental places

A.13.17 27th of March Meeting with Jose



Trye App

MEETING With Jose 27/03

27 March 2020 / 10:00 / Through Zoom

ATTENDEES

Andreas, Tobias, Eebbaa, Dawid, Joachim, Jose

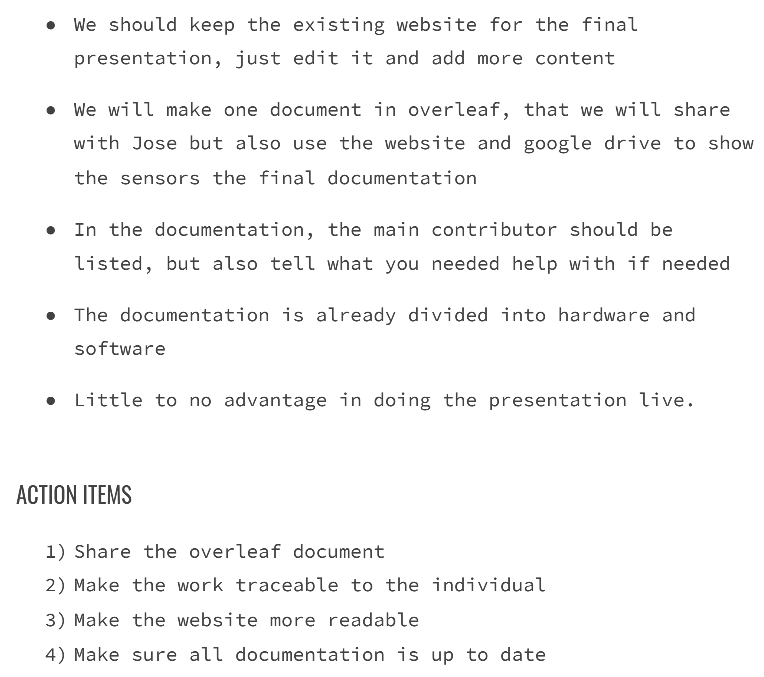
AGENDA

New Business

- Topic 1: Discussion about documentation
  - How will be our last documentation with the third presentation?
  - Does the final submission document can be considered as third documentation.
- Topic 2: Discussion about working on user histories individual
  - Can we help each other and how to document it
  - Should we document the origin of an idea, if it’s from a group member

NOTES

- Agreed using more time with documentation
- We miss some text on our website, eg. a short description of each item, and should remove embedded windows
- Update some documentation, eg. data on the risk register
- Went over how we should document the individual work

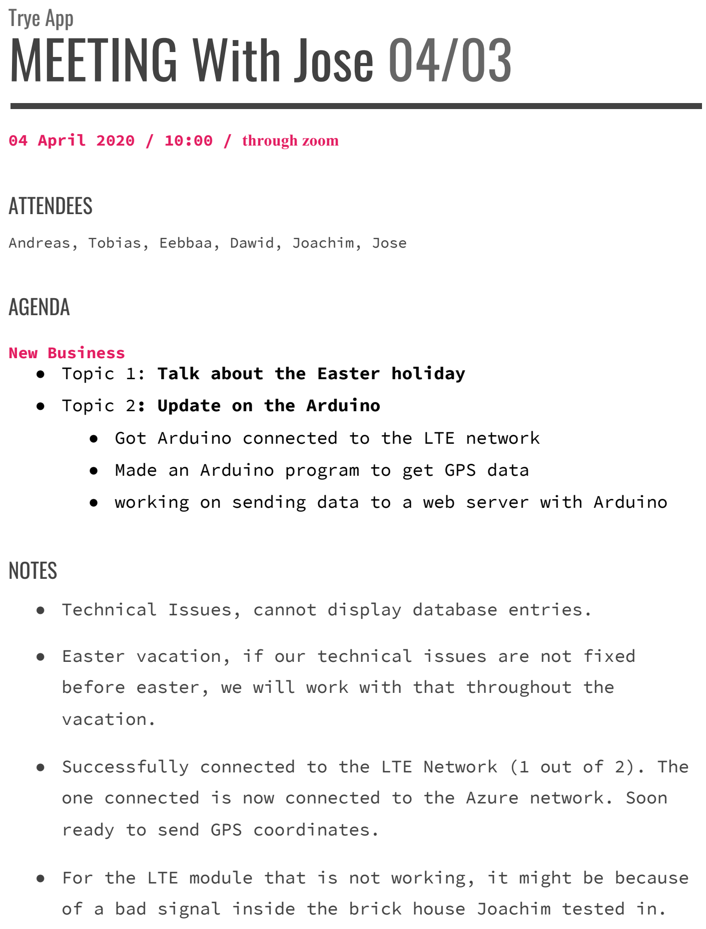


- We should keep the existing website for the final presentation, just edit it and add more content
- We will make one document in overleaf, that we will share with Jose but also use the website and google drive to show the sensors the final documentation
- In the documentation, the main contributor should be listed, but also tell what you needed help with if needed
- The documentation is already divided into hardware and software
- Little to no advantage in doing the presentation live.

ACTION ITEMS

1) Share the overleaf document
2) Make the work traceable to the individual
3) Make the website more readable
4) Make sure all documentation is up to date

A.13.18 4th of April Meeting with Jose



Trye App

MEETING With Jose 04/03

04 April 2020 / 10:00 / through zoom

ATTENDEES

Andreas, Tobias, Eebbaa, Dawid, Joachim, Jose

AGENDA

New Business

- Topic 1: **Talk about the Easter holiday**
- Topic 2**: Update on the Arduino**
  - Got Arduino connected to the LTE network
  - Made an Arduino program to get GPS data
  - working on sending data to a web server with Arduino

NOTES

- Technical Issues, cannot display database entries.
- Easter vacation, if our technical issues are not fixed before easter, we will work with that throughout the vacation.
- Successfully connected to the LTE Network (1 out of 2). The one connected is now connected to the Azure network. Soon ready to send GPS coordinates.
- For the LTE module that is not working, it might be because of a bad signal inside the brick house Joachim tested in.

- Lab:
    - The hardware guys don't need to have access to the lab before late of April. USN might open the lab to some of the students (small possibility). Reply needs to be given by 03.04.2020.
    - They might need some equipment from the Dronesone and the Elektrolab.
- Arduino:
    - Dawit has made an Arduino program to get GPS data.
    - "It's going very good, and I've also already managed to compile it. Already found the correct position of his own apartment.
    - Might add date/time in addition to the already achieved latitude and longitude. (Code is uploaded). Needs to be verified.
    - The GPS program has been tried on an Arduino Mega, not the Arduino LTE. Tested with a GPS module.
    - The GPS module is not a very known module, might have to buy/test a more common one. The Neo6M should be the preferred GPS module.
    - We have ordered a Neo8, which has not come yet. Dawit claims there is not a big difference between the two.
- Testing:
    - Dawit tests it himself. Found his position. The position given by the Arduino was the same as Google gave.

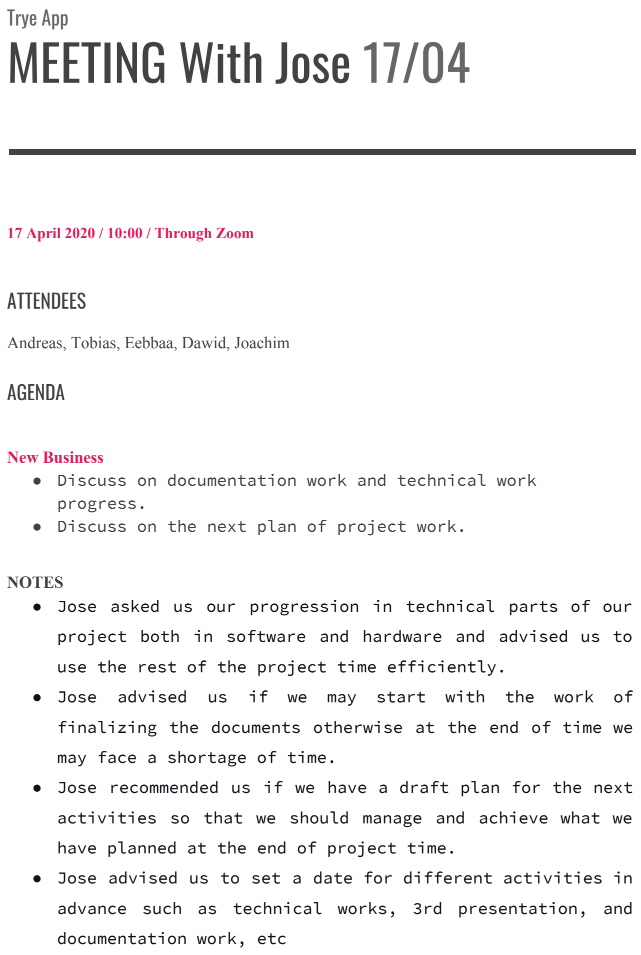
- "Sending GPS data to a web server"
  - Eebbaa is working with that, as noted above he already connected to Azure.
  - "I think it's going well, we've connected the IoT Module to the internet. Successfully connected the Module to the webserver.
  - "Followed the procedures to connect to the Azure IoT platform. The messages are arriving at the server.
- Next step hardware-vice:
  - Need to send the sensor data to the platform, and agree on the protocol for how to send data. (Atm: Date/Time and Lat Long.)
- Grading:
  - Important to be graded individually, but still, work together as a group. Important to not isolate yourself.
- Overleaf:
  - Important to gradually fill the Overleaf document. By gradually filling the Overleaf it is easier to get an overview of the development process.
  - The Overleaf document can and should evolve over time. Important to fill in content progressively. Important to device the structure of the document.
  - Restructuring a document after writing a lot is a long process. Important to agree on a table of contents of the structure of the document.

- Tips for the technical issue:
    - Try fixing the problem meanwhile working parallel on a substitute to avoid a deadlock.
- Dawit: I have fixed my Risk documentation, do you think it is sufficient?
    - Jose: I have not been able to see the progress made as I've not gotten an alert. No longer links to other documents. Looks nicer!
- Documentation work:
    - Make the Software documentation the same as the Hardware documentation.
- Easter vacation:
    - We decided to have an easter vacation.

ACTION ITEMS

1) Resolve technical issues with Planet9 and databases.
2) Verify the GPS sending data program.
3) Gradually fill out the Overleaf document. This process gives us an overview of the project as a whole.
4) Get together as a group and decide on the structure for the Overleaf document. Restructuring because of a lack of communication for the table of content is time-consuming.

A.13.19 17th of April Meeting with Jose



Trye App

MEETING With Jose 17/04

17 April 2020 / 10:00 / Through Zoom

ATTENDEES

Andreas, Tobias, Eebbaa, Dawid, Joachim

AGENDA

New Business

- Discuss on documentation work and technical work progress.
- Discuss on the next plan of project work.

NOTES

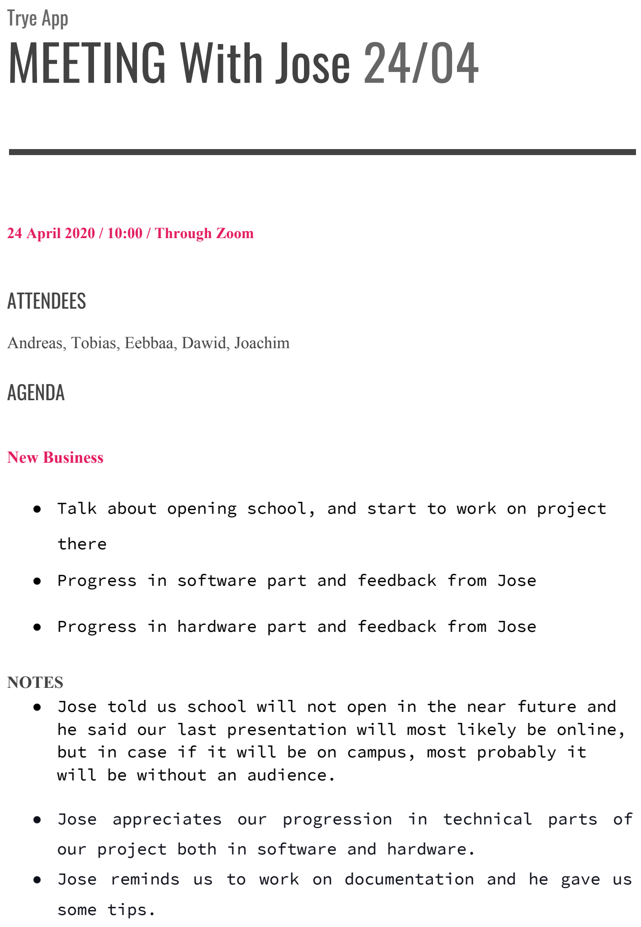
- Jose asked us our progression in technical parts of our project both in software and hardware and advised us to use the rest of the project time efficiently.
- Jose advised us if we may start with the work of finalizing the documents otherwise at the end of time we may face a shortage of time.
- Jose recommended us if we have a draft plan for the next activities so that we should manage and achieve what we have planned at the end of project time.
- Jose advised us to set a date for different activities in advance such as technical works, 3rd presentation, and documentation work, etc

- Jose highly recommended us to focus on planning so that we may not face scarcity of time in the end.
- For the question is asked by Tobias, “Due to coronavirus case, the server that we have provided for free now requested payment, and so we are planned to use Google internet”. Jose advised us if the solution expected from google is the same as from Microsoft internet, then you can go ahead!
- Jose lastly recommended us if we may use Grammarly software for correcting some fails in our English writing, and write better quality documents.

ACTION ITEMS

1. Get together as a group to work on the interface of software and hardware as a final solution.
2. Divide to each of us the parts of the documents and write it for final documentation.

A.13.20 24th of April Meeting with Jose



Trye App

MEETING With Jose 24/04

24 April 2020 / 10:00 / Through Zoom

ATTENDEES

Andreas, Tobias, Eebbaa, Dawid, Joachim

AGENDA

New Business

- Talk about opening school, and start to work on project there
- Progress in software part and feedback from Jose
- Progress in hardware part and feedback from Jose

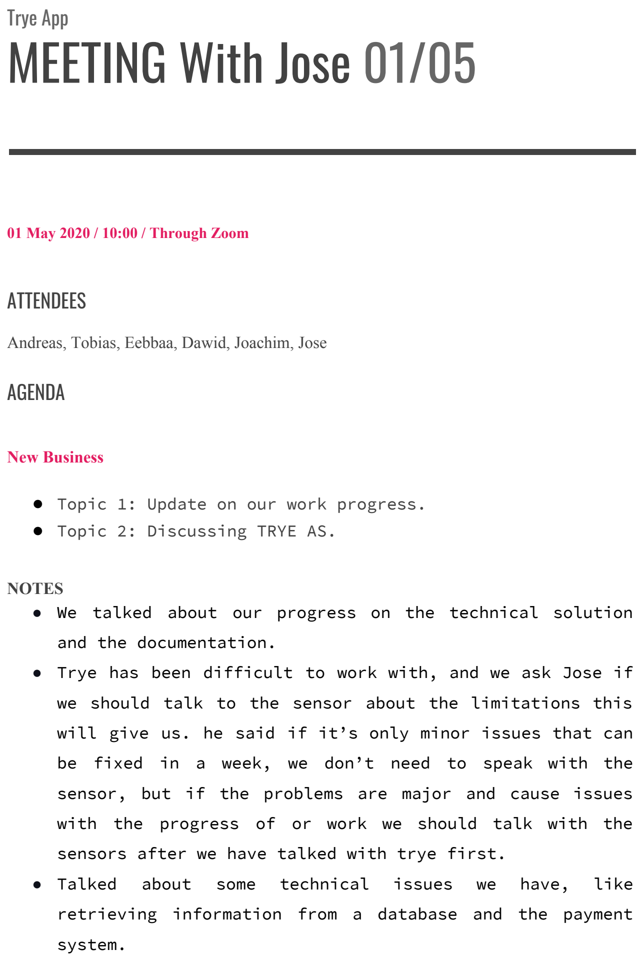
NOTES

- Jose told us school will not open in the near future and he said our last presentation will most likely be online, but in case if it will be on campus, most probably it will be without an audience.
- Jose appreciates our progression in technical parts of our project both in software and hardware.
- Jose reminds us to work on documentation and he gave us some tips.

ACTION ITEMS

1. Working on the documentation part in parallel with technical work to use our time efficiently.

A.13.21 1st of May Meeting with Jose



Trye App

MEETING With Jose 01/05

01 May 2020 / 10:00 / Through Zoom

ATTENDEES

Andreas, Tobias, Eebbaa, Dawid, Joachim, Jose

AGENDA

New Business

- Topic 1: Update on our work progress.
- Topic 2: Discussing TRYE AS.

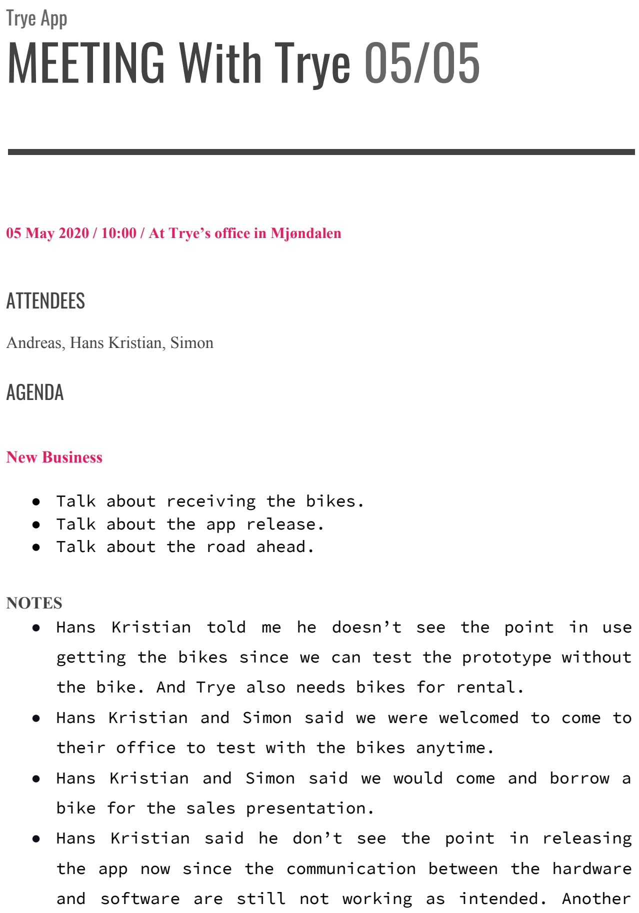
NOTES

- We talked about our progress on the technical solution and the documentation.
- Trye has been difficult to work with, and we ask Jose if we should talk to the sensor about the limitations this will give us. he said if it's only minor issues that can be fixed in a week, we don't need to speak with the sensor, but if the problems are major and cause issues with the progress of or work we should talk with the sensors after we have talked with trye first.
- Talked about some technical issues we have, like retrieving information from a database and the payment system.

ACTION ITEMS

1. Talk with trye, and make sure they understand the limitation they put on us by not providing the information we need.

A.13.22 5th of May Meeting with TRYE



Trye App

MEETING With Trye 05/05

05 May 2020 / 10:00 / At Trye's office in Mjøndalen

ATTENDEES

Andreas, Hans Kristian, Simon

AGENDA

New Business

- Talk about receiving the bikes.
- Talk about the app release.
- Talk about the road ahead.

NOTES

- Hans Kristian told me he doesn't see the point in use getting the bikes since we can test the prototype without the bike. And Trye also needs bikes for rental.
- Hans Kristian and Simon said we were welcomed to come to their office to test with the bikes anytime.
- Hans Kristian and Simon said we would come and borrow a bike for the sales presentation.
- Hans Kristian said he don't see the point in releasing the app now since the communication between the hardware and software are still not working as intended. Another

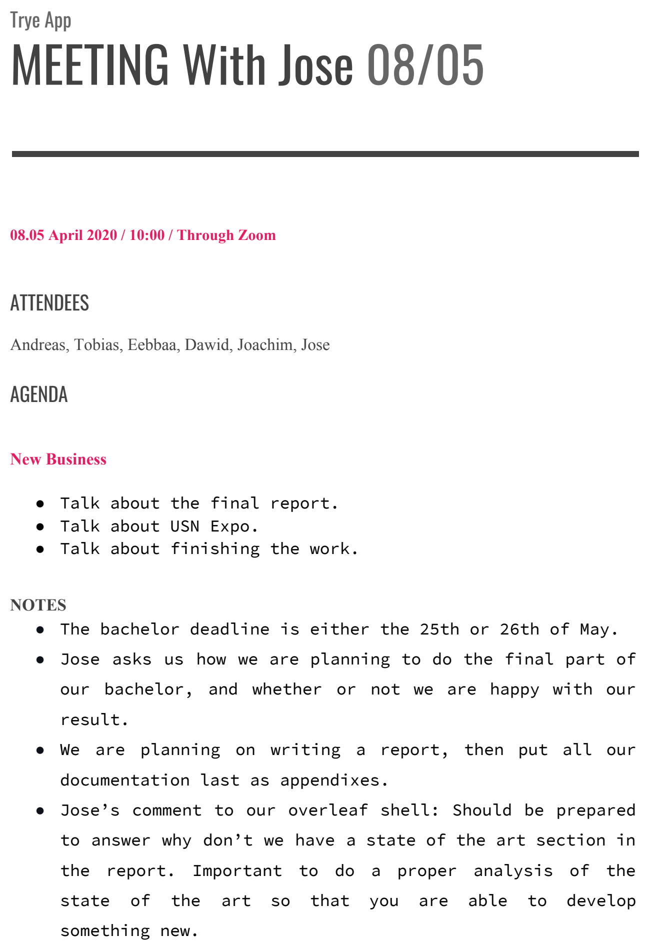
reason is that there are some security features that need to implement.

- Hans Kristian told me that Trye might use consultants to finish our work. They also said they were unhappy that we made our own hardware solution instead of being able to use Shimano’s existing system.
- Hans Kristian told me that they want two different systems so we don't have to interface with each other. That means we are free to make our own payment solution and booking system since our system is only used for mountain lodge rental.
- They might contact us later if they need help with the system we deliver to them.

ACTION ITEMS

1. Get the bike for the presentation
2. Postpone the Beta release
3. Make a prototype without the bike
4. Since Trye wants two different systems we don't have to interface with their system and we can make our own payment system.

A.13.23 8th of May Meeting with Jose



Trye App

MEETING With Jose 08/05

08.05 April 2020 / 10:00 / Through Zoom

ATTENDEES

Andreas, Tobias, Eebbaa, Dawid, Joachim, Jose

AGENDA

New Business

- Talk about the final report.
- Talk about USN Expo.
- Talk about finishing the work.

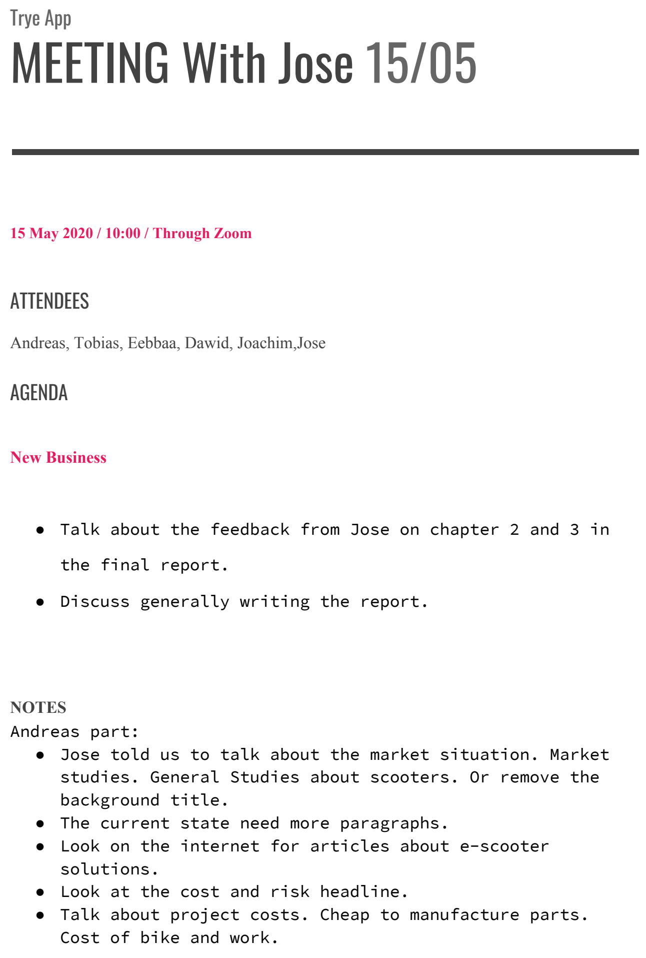
NOTES

- The bachelor deadline is either the 25th or 26th of May.
- Jose asks us how we are planning to do the final part of our bachelor, and whether or not we are happy with our result.
- We are planning on writing a report, then put all our documentation last as appendixes.
- Jose's comment to our overleaf shell: Should be prepared to answer why don't we have a state of the art section in the report. Important to do a proper analysis of the state of the art so that you are able to develop something new.

- Don’t like some of the words we use. “Preparation”, Use “Technical work” instead of “Technical work process”. Would not use the words process.
- Should not use the words “final” in “final beta”. Can say prototype instead.
- Avoid having an unnecessary amount of work.
- Satlites report: Background, state of the art, Intro to start with, Project, Technical work, Result for final prototype and conclusion.
- There’s a difference between conclusion and conclusions. Use conclusion plus a discussion of the report. Not conclusions.
- Might add a part in front of conclusion to discuss what we have done. This includes what could’ve been better, what are we happy with etc. The only reason discussion is drawn out from the conclusion part is if the parts combined gets too large.
- Pay attention to the following: My experience with working with Overleaf is that, Grammarly does not highlight recommendations for typos and improvements. Pay attention to the text, might copy from Overleaf to Google Docs to spell check.
- One frequent problem is that students have excellent technical skills, and excellent technical work, but the report is not up to the same quality. Very irritating if the report is not well written. Mismatches of different sorts.
- “There’s no second opportunity of making a first impression”.
- Don’t just send it to Jose immediately, go through it once again as this will greatly improve the final quality. Have only time to revise it once.

- Karoline just posted the earlier posters of USN Expo. Jose thinks it will be a digital event, better clarify with Karoline. As we will have a virtual Expo we need to create a poster as well. Nice to show all our hard work to the community, so, therefore, it might be a good idea to postpone it to the fall.
- Is the presentation and report delivered the same day? Jose - I don't think that has been decided.
- We will have the presentation Thursday 11th of June 12:30. "I will need your documents latest a day before the presentation starts". This gives us two weeks from delivering the report to making the presentation.
- Would use this as an opportunity to make the presentation richer than a normal presentation.
- Jose likes the idea of travelling to Mjøndalen and showing the app/bicycle would be a great addition.
- Andreas made a user for the billing software Stripe. This enables us to make a working prototype.
- Since you don't have the bike anymore, this gives you the freedom of not being bound to using what's in the bike.
- When testing the unlocking feature, we can simply light an LED when the bike is "unlocked".
- When asked to interface an existing system, but you don't have complete access to it you may or may not do it entirely. Must prove that we have a coherent solution overall and if we have the freedom to install it to a specific bike with the features needed (Could use a normal bike if possible).
- Keep focused on the final task! Keep in mind that your main focus at this moment is to write a proper final report.

A.13.24 15th of May Meeting with Jose



Trye App

MEETING With Jose 15/05

15 May 2020 / 10:00 / Through Zoom

ATTENDEES

Andreas, Tobias, Eebbaa, Dawid, Joachim,Jose

AGENDA

New Business

- Talk about the feedback from Jose on chapter 2 and 3 in the final report.
- Discuss generally writing the report.

NOTES

Andreas part:

- Jose told us to talk about the market situation. Market studies. General Studies about scooters. Or remove the background title.
- The current state need more paragraphs.
- Look on the internet for articles about e-scooter solutions.
- Look at the cost and risk headline.
- Talk about project costs. Cheap to manufacture parts. Cost of bike and work.

- Not so many risks of money since there are not expensive parts.
- Make clear what parts you want to be graded for, which means we should write a paragraph about our main technical responsibilities.
- Change the rent manually rent part.

Tobias part:

- Jose said we don't have to write that we changed stand up a meeting time. But only write about the tools
- Write about Covid-19 in the risk manager in the process part.

Eebbaas part:

- Add some information about the requirements in the start instead of just link to an appendix
- Added some phrases about why stakeholders are there. Explain why they are stakeholders for the project
- Might wish to elaborate further on giving an example of how functions are measurable, traceable, testable and verifiable.

Joachim’s part:

- Jose said the phrase about battery level should be written to show that we are assuming a linear decrease in battery level.

Dawits part:

- Find some more info on the internet related to the project risks.

ACTION ITEMS

1. Fix writing error and grammar.
2. Look into the comments of Jose and maybe rewrite some parts.

A.14 Work hours

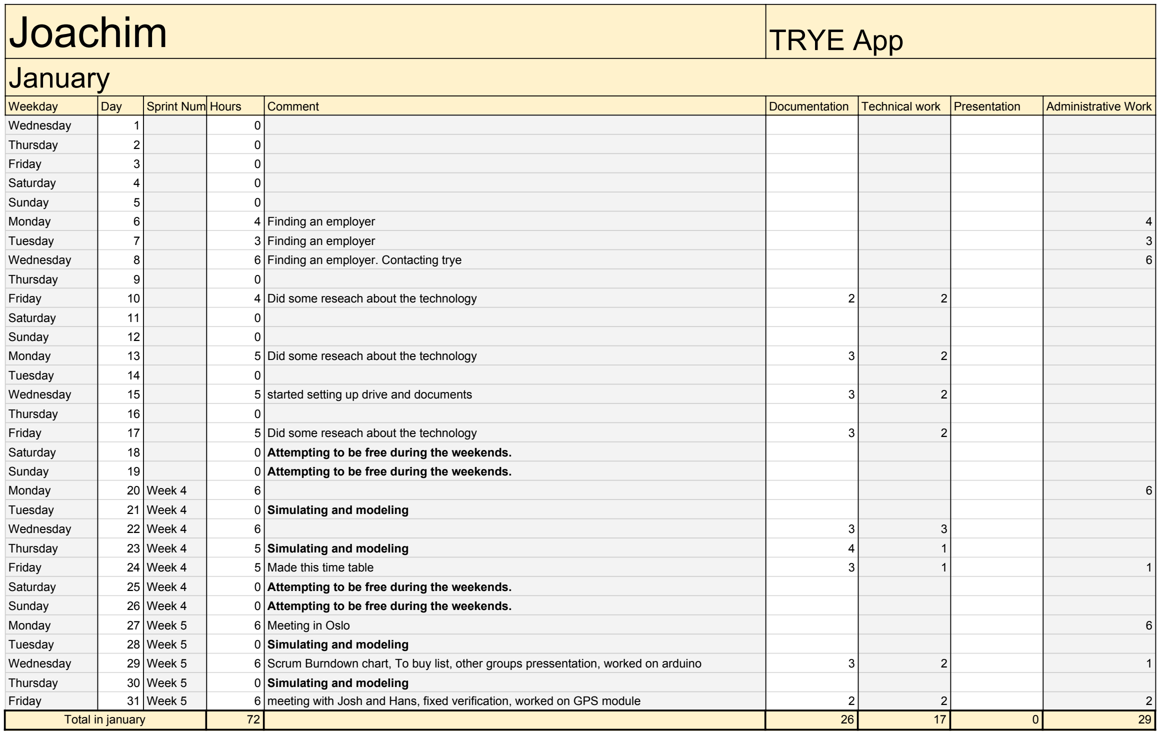
A.14.1 Work hours Joachim

A.14.2 Work hours Andreas

A.14.3 Work hours Tobias

A.14.4 Work hours Eebbaa

A.14.5 Work hours Dawid

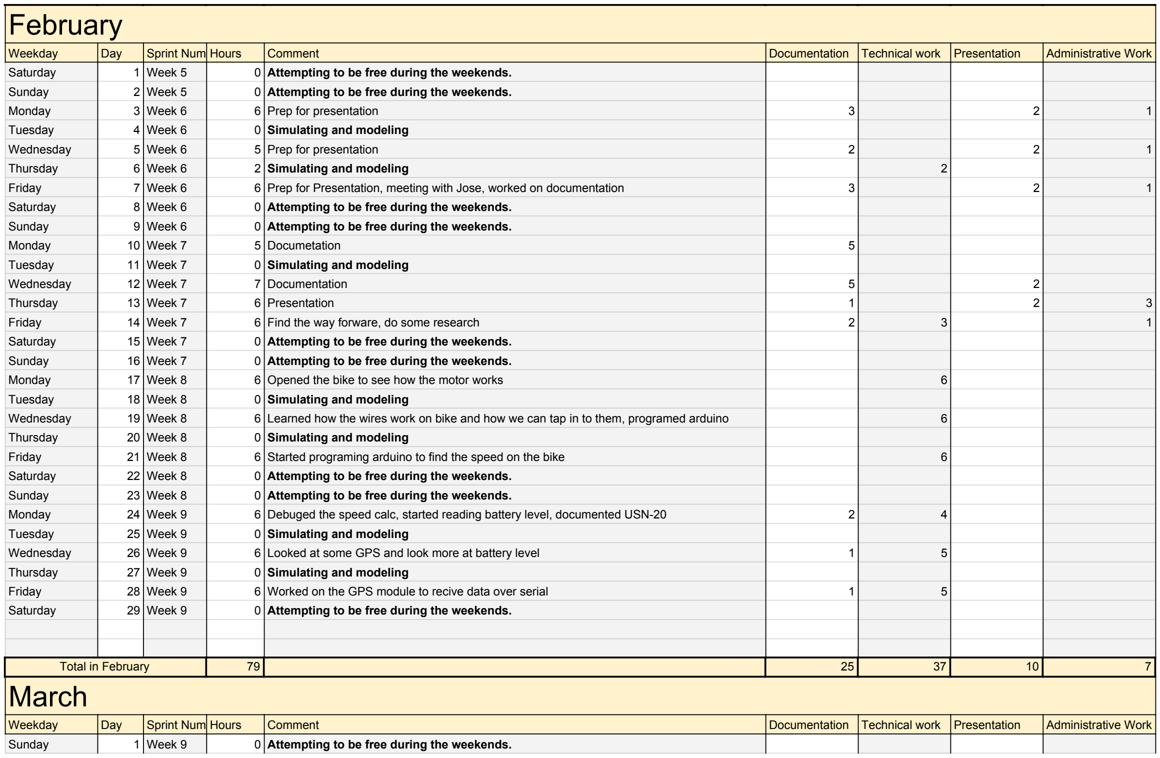


Joachim

TRYE App

January

Weekday	Day	Sprint Num	Hours	Comment	Documentation	Technical work	Presentation	Administrative Work
Wednesday	1		0					
Thursday	2		0					
Friday	3		0					
Saturday	4		0					
Sunday	5		0					
Monday	6		4	Finding an employer				4
Tuesday	7		3	Finding an employer				3
Wednesday	8		6	Finding an employer. Contacting trye				6
Thursday	9		0					
Friday	10		4	Did some reseach about the technology	2	2		
Saturday	11		0					
Sunday	12		0					
Monday	13		5	Did some reseach about the technology	3	2		
Tuesday	14		0					
Wednesday	15		5	started setting up drive and documents	3	2		
Thursday	16		0					
Friday	17		5	Did some reseach about the technology	3	2		
Saturday	18		0	**Attempting to be free during the weekends.**				
Sunday	19		0	**Attempting to be free during the weekends.**				
Monday	20	Week 4	6					6
Tuesday	21	Week 4	0	**Simulating and modeling**				
Wednesday	22	Week 4	6		3	3		
Thursday	23	Week 4	5	**Simulating and modeling**	4	1		
Friday	24	Week 4	5	Made this time table	3	1		1
Saturday	25	Week 4	0	**Attempting to be free during the weekends.**				
Sunday	26	Week 4	0	**Attempting to be free during the weekends.**				
Monday	27	Week 5	6	Meeting in Oslo				6
Tuesday	28	Week 5	0	**Simulating and modeling**				
Wednesday	29	Week 5	6	Scrum Burndown chart, To buy list, other groups pressentation, worked on arduino	3	2		1
Thursday	30	Week 5	0	**Simulating and modeling**				
Friday	31	Week 5	6	meeting with Josh and Hans, fixed verification, worked on GPS module	2	2		2
Total in january			72		26	17	0	29

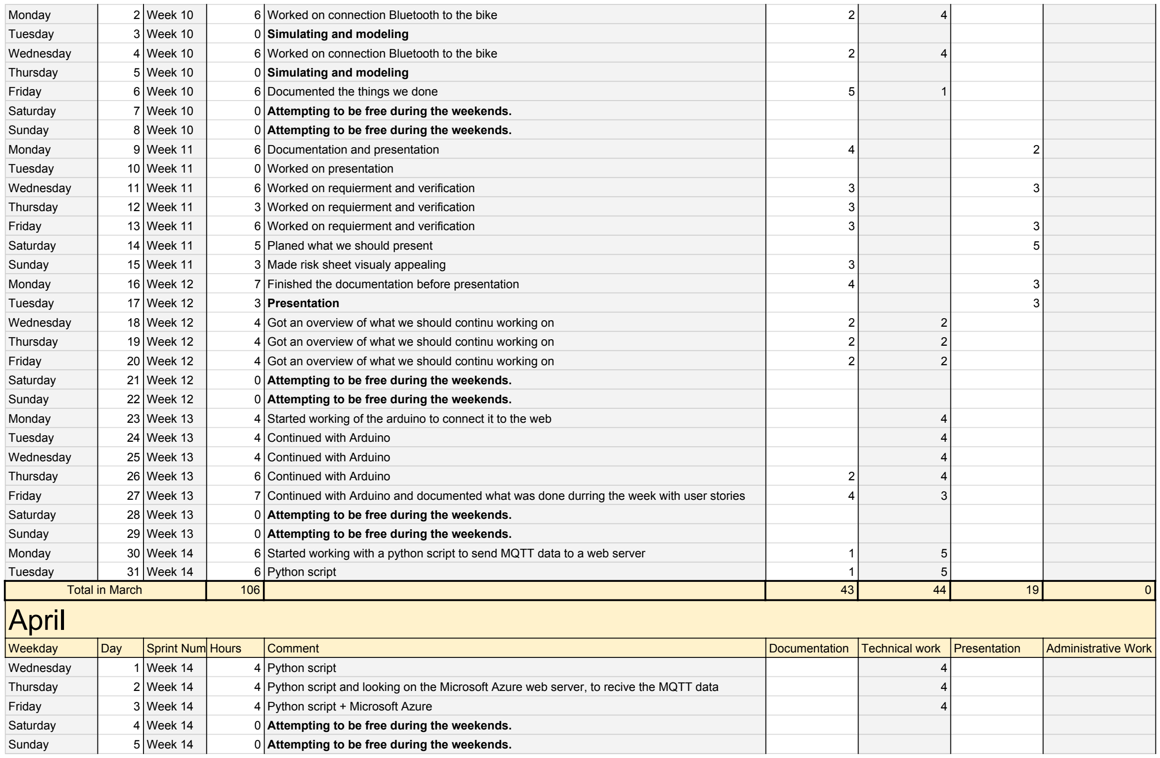


February

Weekday	Day	Sprint Num	Hours	Comment	Documentation	Technical work	Presentation	Administrative Work
Saturday	1	Week 5	0	**Attempting to be free during the weekends.**				
Sunday	2	Week 5	0	**Attempting to be free during the weekends.**				
Monday	3	Week 6	6	Prep for presentation	3		2	1
Tuesday	4	Week 6	0	**Simulating and modeling**				
Wednesday	5	Week 6	5	Prep for presentation	2		2	1
Thursday	6	Week 6	2	**Simulating and modeling**		2		
Friday	7	Week 6	6	Prep for Presentation, meeting with Jose, worked on documentation	3		2	1
Saturday	8	Week 6	0	**Attempting to be free during the weekends.**				
Sunday	9	Week 6	0	**Attempting to be free during the weekends.**				
Monday	10	Week 7	5	Documetation	5			
Tuesday	11	Week 7	0	**Simulating and modeling**				
Wednesday	12	Week 7	7	Documentation	5		2	
Thursday	13	Week 7	6	Presentation	1		2	3
Friday	14	Week 7	6	Find the way forware, do some research	2	3		1
Saturday	15	Week 7	0	**Attempting to be free during the weekends.**				
Sunday	16	Week 7	0	**Attempting to be free during the weekends.**				
Monday	17	Week 8	6	Opened the bike to see how the motor works		6		
Tuesday	18	Week 8	0	**Simulating and modeling**				
Wednesday	19	Week 8	6	Learned how the wires work on bike and how we can tap in to them, programed arduino		6		
Thursday	20	Week 8	0	**Simulating and modeling**				
Friday	21	Week 8	6	Started programing arduino to find the speed on the bike		6		
Saturday	22	Week 8	0	**Attempting to be free during the weekends.**				
Sunday	23	Week 8	0	**Attempting to be free during the weekends.**				
Monday	24	Week 9	6	Debuged the speed calc, started reading battery level, documented USN-20	2	4		
Tuesday	25	Week 9	0	**Simulating and modeling**				
Wednesday	26	Week 9	6	Looked at some GPS and look more at battery level	1	5		
Thursday	27	Week 9	0	**Simulating and modeling**				
Friday	28	Week 9	6	Worked on the GPS module to recive data over serial	1	5		
Saturday	29	Week 9	0	**Attempting to be free during the weekends.**				
Total in February			79		25	37	10	7

March

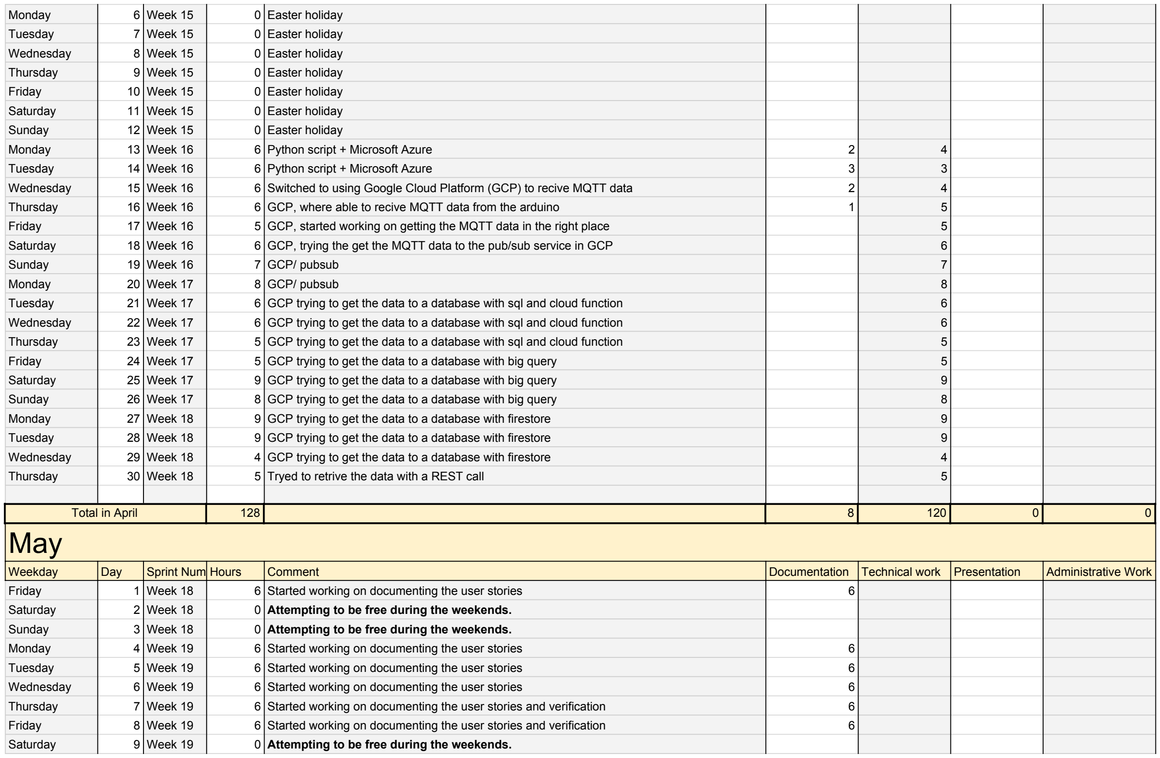
Weekday	Day	Sprint Num	Hours	Comment	Documentation	Technical work	Presentation	Administrative Work
Sunday	1	Week 9	0	**Attempting to be free during the weekends.**				



Monday	2	Week 10	6	Worked on connection Bluetooth to the bike	2	4		
Tuesday	3	Week 10	0	**Simulating and modeling**				
Wednesday	4	Week 10	6	Worked on connection Bluetooth to the bike	2	4		
Thursday	5	Week 10	0	**Simulating and modeling**				
Friday	6	Week 10	6	Documented the things we done	5	1		
Saturday	7	Week 10	0	**Attempting to be free during the weekends.**				
Sunday	8	Week 10	0	**Attempting to be free during the weekends.**				
Monday	9	Week 11	6	Documentation and presentation	4		2	
Tuesday	10	Week 11	0	Worked on presentation				
Wednesday	11	Week 11	6	Worked on requierment and verification	3		3	
Thursday	12	Week 11	3	Worked on requierment and verification	3			
Friday	13	Week 11	6	Worked on requierment and verification	3		3	
Saturday	14	Week 11	5	Planed what we should present			5	
Sunday	15	Week 11	3	Made risk sheet visualy appealing	3			
Monday	16	Week 12	7	Finished the documentation before presentation	4		3	
Tuesday	17	Week 12	3	**Presentation**			3	
Wednesday	18	Week 12	4	Got an overview of what we should continu working on	2	2		
Thursday	19	Week 12	4	Got an overview of what we should continu working on	2	2		
Friday	20	Week 12	4	Got an overview of what we should continu working on	2	2		
Saturday	21	Week 12	0	**Attempting to be free during the weekends.**				
Sunday	22	Week 12	0	**Attempting to be free during the weekends.**				
Monday	23	Week 13	4	Started working of the arduino to connect it to the web		4		
Tuesday	24	Week 13	4	Continued with Arduino		4		
Wednesday	25	Week 13	4	Continued with Arduino		4		
Thursday	26	Week 13	6	Continued with Arduino	2	4		
Friday	27	Week 13	7	Continued with Arduino and documented what was done durring the week with user stories	4	3		
Saturday	28	Week 13	0	**Attempting to be free during the weekends.**				
Sunday	29	Week 13	0	**Attempting to be free during the weekends.**				
Monday	30	Week 14	6	Started working with a python script to send MQTT data to a web server	1	5		
Tuesday	31	Week 14	6	Python script	1	5		
Total in March			106		43	44	19	0

April

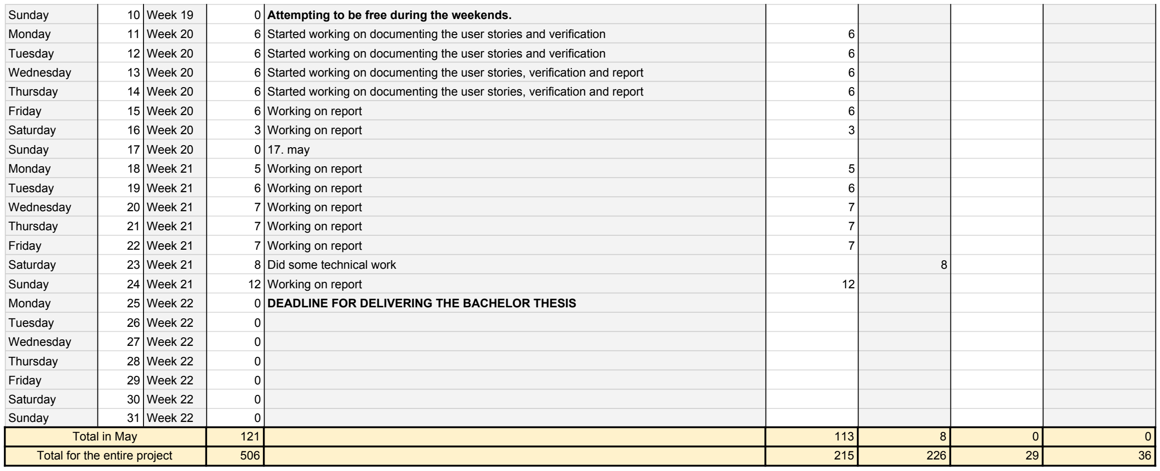
Weekday	Day	Sprint Num	Hours	Comment	Documentation	Technical work	Presentation	Administrative Work
Wednesday	1	Week 14	4	Python script		4		
Thursday	2	Week 14	4	Python script and looking on the Microsoft Azure web server, to recive the MQTT data		4		
Friday	3	Week 14	4	Python script + Microsoft Azure		4		
Saturday	4	Week 14	0	**Attempting to be free during the weekends.**				
Sunday	5	Week 14	0	**Attempting to be free during the weekends.**				



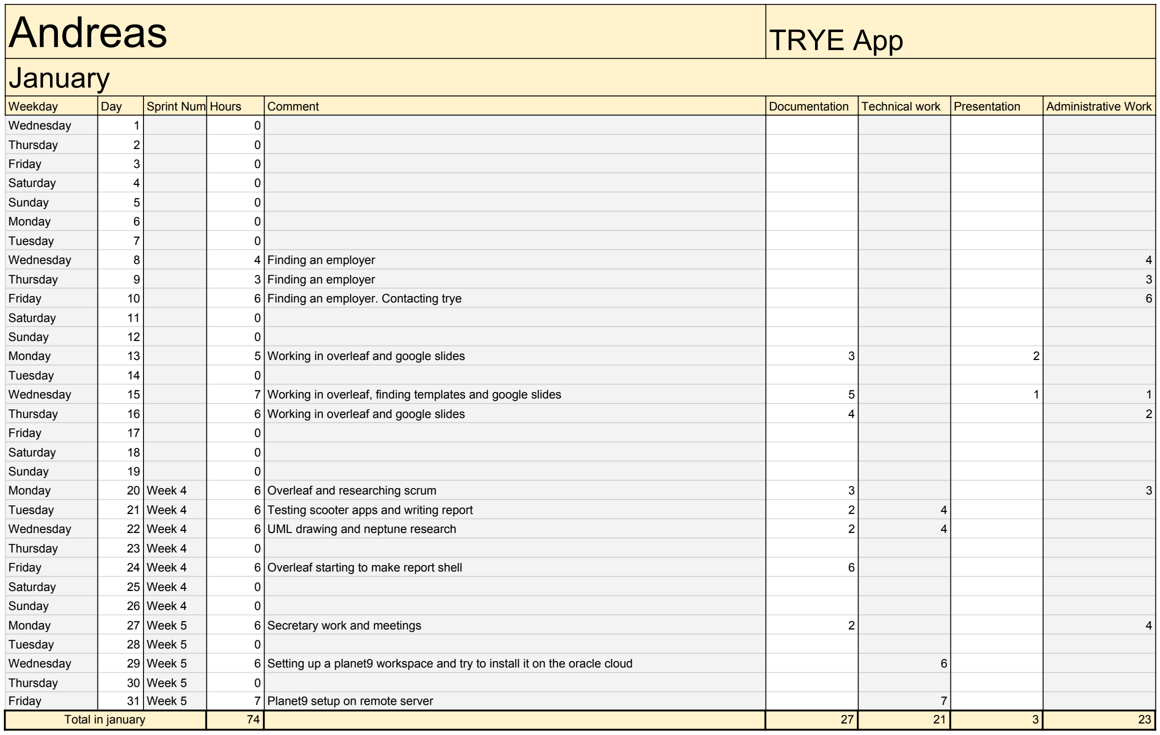
Monday	6	Week 15	0	Easter holiday				
Tuesday	7	Week 15	0	Easter holiday				
Wednesday	8	Week 15	0	Easter holiday				
Thursday	9	Week 15	0	Easter holiday				
Friday	10	Week 15	0	Easter holiday				
Saturday	11	Week 15	0	Easter holiday				
Sunday	12	Week 15	0	Easter holiday				
Monday	13	Week 16	6	Python script + Microsoft Azure	2	4		
Tuesday	14	Week 16	6	Python script + Microsoft Azure	3	3		
Wednesday	15	Week 16	6	Switched to using Google Cloud Platform (GCP) to recive MQTT data	2	4		
Thursday	16	Week 16	6	GCP, where able to recive MQTT data from the arduino	1	5		
Friday	17	Week 16	5	GCP, started working on getting the MQTT data in the right place		5		
Saturday	18	Week 16	6	GCP, trying the get the MQTT data to the pub/sub service in GCP		6		
Sunday	19	Week 16	7	GCP/ pubsub		7		
Monday	20	Week 17	8	GCP/ pubsub		8		
Tuesday	21	Week 17	6	GCP trying to get the data to a database with sql and cloud function		6		
Wednesday	22	Week 17	6	GCP trying to get the data to a database with sql and cloud function		6		
Thursday	23	Week 17	5	GCP trying to get the data to a database with sql and cloud function		5		
Friday	24	Week 17	5	GCP trying to get the data to a database with big query		5		
Saturday	25	Week 17	9	GCP trying to get the data to a database with big query		9		
Sunday	26	Week 17	8	GCP trying to get the data to a database with big query		8		
Monday	27	Week 18	9	GCP trying to get the data to a database with firestore		9		
Tuesday	28	Week 18	9	GCP trying to get the data to a database with firestore		9		
Wednesday	29	Week 18	4	GCP trying to get the data to a database with firestore		4		
Thursday	30	Week 18	5	Tryed to retrive the data with a REST call		5		
Total in April			128		8	120	0	0

May

Weekday	Day	Sprint Num	Hours	Comment	Documentation	Technical work	Presentation	Administrative Work
Friday	1	Week 18	6	Started working on documenting the user stories	6			
Saturday	2	Week 18	0	**Attempting to be free during the weekends.**				
Sunday	3	Week 18	0	**Attempting to be free during the weekends.**				
Monday	4	Week 19	6	Started working on documenting the user stories	6			
Tuesday	5	Week 19	6	Started working on documenting the user stories	6			
Wednesday	6	Week 19	6	Started working on documenting the user stories	6			
Thursday	7	Week 19	6	Started working on documenting the user stories and verification	6			
Friday	8	Week 19	6	Started working on documenting the user stories and verification	6			
Saturday	9	Week 19	0	**Attempting to be free during the weekends.**				



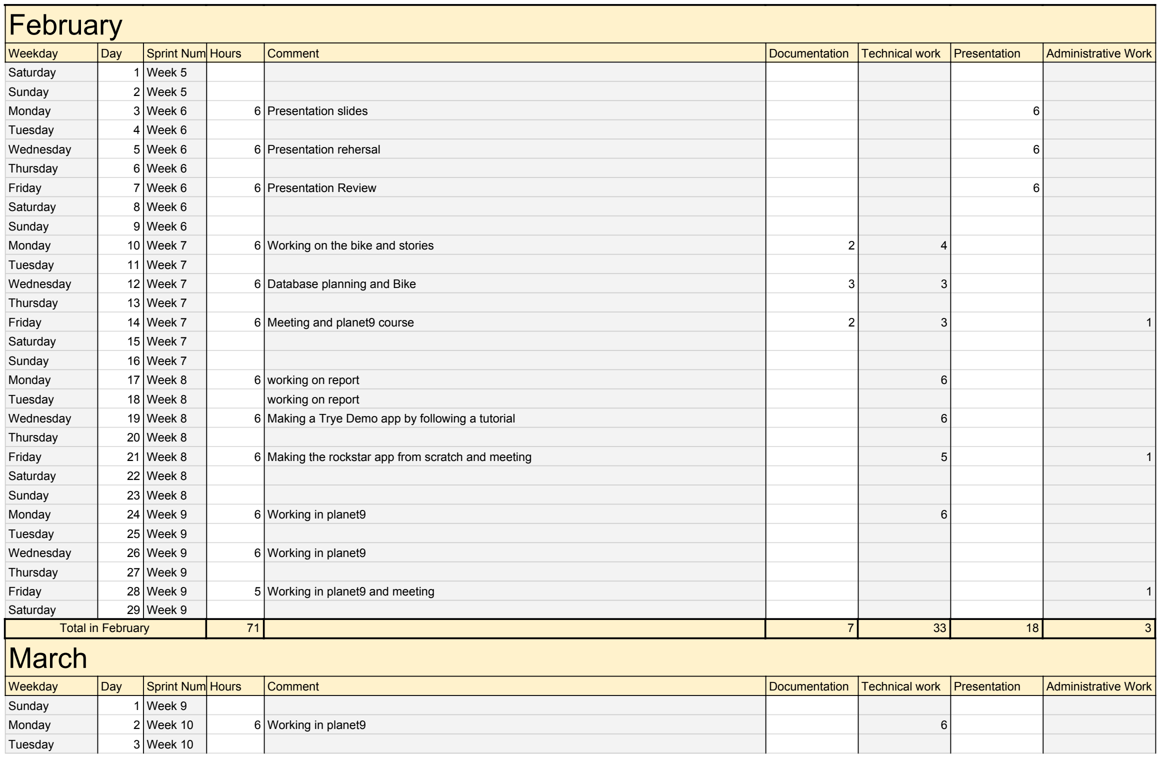
Sunday	10	Week 19	0	**Attempting to be free during the weekends.**				
Monday	11	Week 20	6	Started working on documenting the user stories and verification	6			
Tuesday	12	Week 20	6	Started working on documenting the user stories and verification	6			
Wednesday	13	Week 20	6	Started working on documenting the user stories, verification and report	6			
Thursday	14	Week 20	6	Started working on documenting the user stories, verification and report	6			
Friday	15	Week 20	6	Working on report	6			
Saturday	16	Week 20	3	Working on report	3			
Sunday	17	Week 20	0	17. may				
Monday	18	Week 21	5	Working on report	5			
Tuesday	19	Week 21	6	Working on report	6			
Wednesday	20	Week 21	7	Working on report	7			
Thursday	21	Week 21	7	Working on report	7			
Friday	22	Week 21	7	Working on report	7			
Saturday	23	Week 21	8	Did some technical work		8		
Sunday	24	Week 21	12	Working on report	12			
Monday	25	Week 22	0	**DEADLINE FOR DELIVERING THE BACHELOR THESIS**				
Tuesday	26	Week 22	0					
Wednesday	27	Week 22	0					
Thursday	28	Week 22	0					
Friday	29	Week 22	0					
Saturday	30	Week 22	0					
Sunday	31	Week 22	0					
Total in May			121		113	8	0	0
Total for the entire project			506		215	226	29	36



Andreas | TRYE App

January

Weekday	Day	Sprint Num	Hours	Comment	Documentation	Technical work	Presentation	Administrative Work
Wednesday	1		0					
Thursday	2		0					
Friday	3		0					
Saturday	4		0					
Sunday	5		0					
Monday	6		0					
Tuesday	7		0					
Wednesday	8		4	Finding an employer				4
Thursday	9		3	Finding an employer				3
Friday	10		6	Finding an employer. Contacting trye				6
Saturday	11		0					
Sunday	12		0					
Monday	13		5	Working in overleaf and google slides	3		2	
Tuesday	14		0					
Wednesday	15		7	Working in overleaf, finding templates and google slides	5		1	1
Thursday	16		6	Working in overleaf and google slides	4			2
Friday	17		0					
Saturday	18		0					
Sunday	19		0					
Monday	20	Week 4	6	Overleaf and researching scrum	3			3
Tuesday	21	Week 4	6	Testing scooter apps and writing report	2	4		
Wednesday	22	Week 4	6	UML drawing and neptune research	2	4		
Thursday	23	Week 4	0					
Friday	24	Week 4	6	Overleaf starting to make report shell	6			
Saturday	25	Week 4	0					
Sunday	26	Week 4	0					
Monday	27	Week 5	6	Secretary work and meetings	2			4
Tuesday	28	Week 5	0					
Wednesday	29	Week 5	6	Setting up a planet9 workspace and try to install it on the oracle cloud		6		
Thursday	30	Week 5	0					
Friday	31	Week 5	7	Planet9 setup on remote server		7		
Total in january			74		27	21	3	23

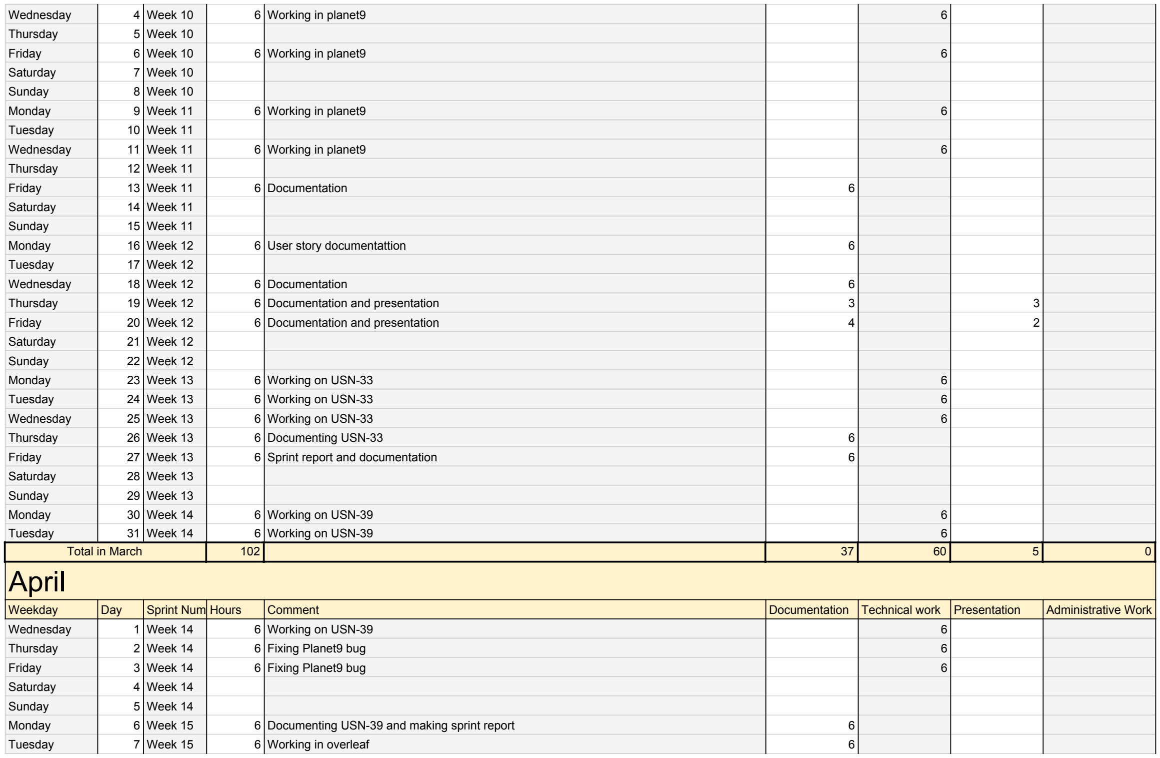


February

Weekday	Day	Sprint Num	Hours	Comment	Documentation	Technical work	Presentation	Administrative Work
Saturday	1	Week 5						
Sunday	2	Week 5						
Monday	3	Week 6	6	Presentation slides			6	
Tuesday	4	Week 6						
Wednesday	5	Week 6	6	Presentation rehersal			6	
Thursday	6	Week 6						
Friday	7	Week 6	6	Presentation Review			6	
Saturday	8	Week 6						
Sunday	9	Week 6						
Monday	10	Week 7	6	Working on the bike and stories	2	4		
Tuesday	11	Week 7						
Wednesday	12	Week 7	6	Database planning and Bike	3	3		
Thursday	13	Week 7						
Friday	14	Week 7	6	Meeting and planet9 course	2	3		1
Saturday	15	Week 7						
Sunday	16	Week 7						
Monday	17	Week 8	6	working on report		6		
Tuesday	18	Week 8		working on report				
Wednesday	19	Week 8	6	Making a Trye Demo app by following a tutorial		6		
Thursday	20	Week 8						
Friday	21	Week 8	6	Making the rockstar app from scratch and meeting		5		1
Saturday	22	Week 8						
Sunday	23	Week 8						
Monday	24	Week 9	6	Working in planet9		6		
Tuesday	25	Week 9						
Wednesday	26	Week 9	6	Working in planet9				
Thursday	27	Week 9						
Friday	28	Week 9	5	Working in planet9 and meeting				1
Saturday	29	Week 9						
Total in February			71		7	33	18	3

March

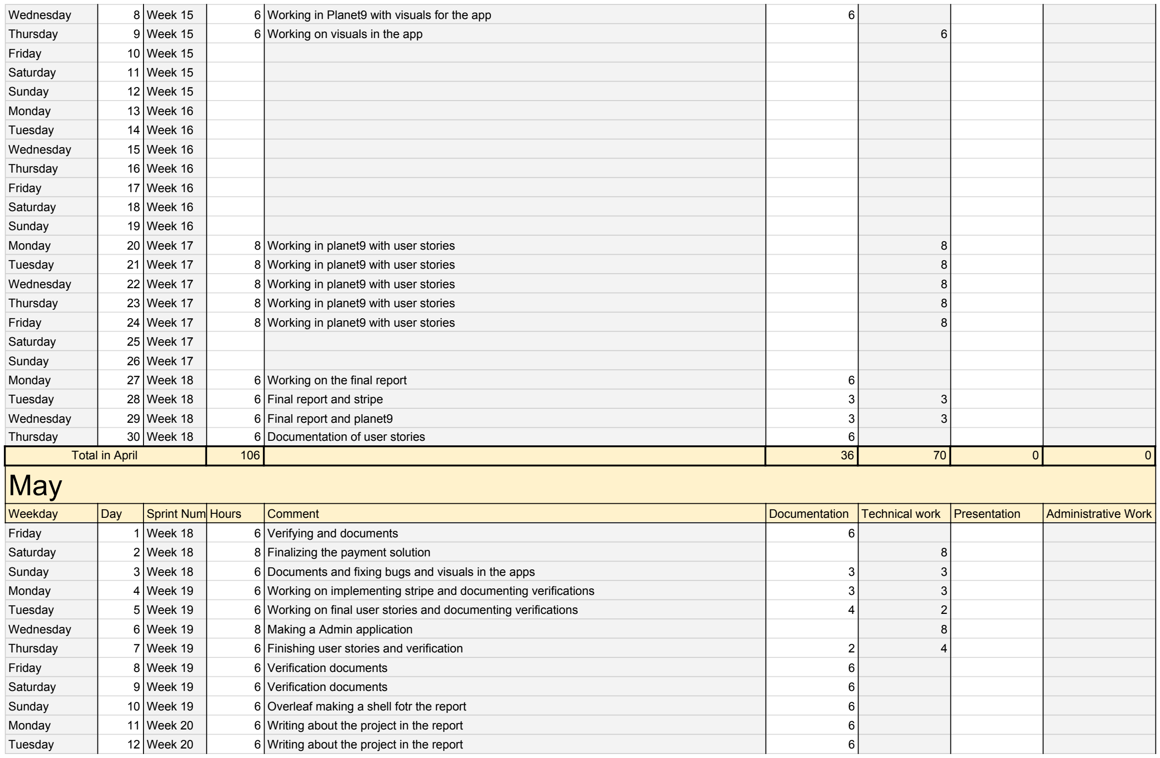
Weekday	Day	Sprint Num	Hours	Comment	Documentation	Technical work	Presentation	Administrative Work
Sunday	1	Week 9						
Monday	2	Week 10	6	Working in planet9		6		
Tuesday	3	Week 10						



Wednesday	4	Week 10	6	Working in planet9		6		
Thursday	5	Week 10						
Friday	6	Week 10	6	Working in planet9		6		
Saturday	7	Week 10						
Sunday	8	Week 10						
Monday	9	Week 11	6	Working in planet9		6		
Tuesday	10	Week 11						
Wednesday	11	Week 11	6	Working in planet9		6		
Thursday	12	Week 11						
Friday	13	Week 11	6	Documentation	6			
Saturday	14	Week 11						
Sunday	15	Week 11						
Monday	16	Week 12	6	User story documentattion	6			
Tuesday	17	Week 12						
Wednesday	18	Week 12	6	Documentation	6			
Thursday	19	Week 12	6	Documentation and presentation	3		3	
Friday	20	Week 12	6	Documentation and presentation	4		2	
Saturday	21	Week 12						
Sunday	22	Week 12						
Monday	23	Week 13	6	Working on USN-33		6		
Tuesday	24	Week 13	6	Working on USN-33		6		
Wednesday	25	Week 13	6	Working on USN-33		6		
Thursday	26	Week 13	6	Documenting USN-33	6			
Friday	27	Week 13	6	Sprint report and documentation	6			
Saturday	28	Week 13						
Sunday	29	Week 13						
Monday	30	Week 14	6	Working on USN-39		6		
Tuesday	31	Week 14	6	Working on USN-39		6		
Total in March			102		37	60	5	0

April

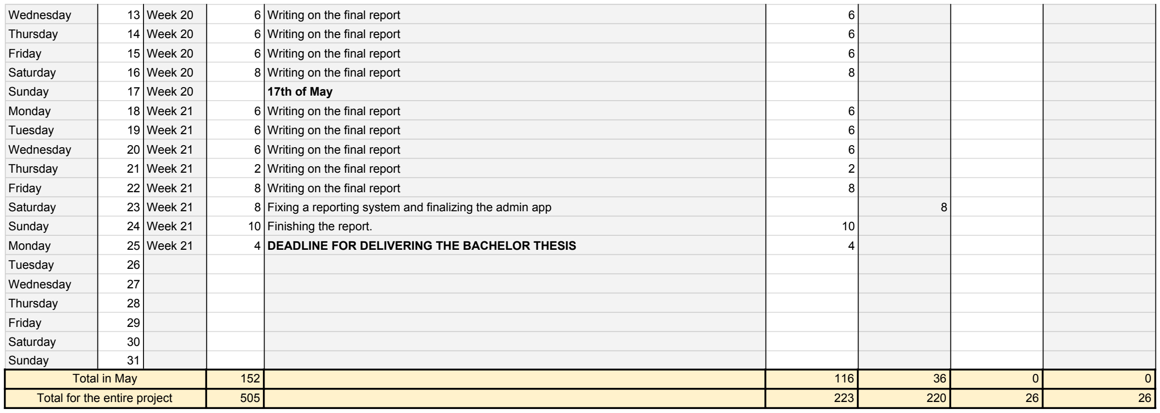
Weekday	Day	Sprint Num	Hours	Comment	Documentation	Technical work	Presentation	Administrative Work
Wednesday	1	Week 14	6	Working on USN-39		6		
Thursday	2	Week 14	6	Fixing Planet9 bug		6		
Friday	3	Week 14	6	Fixing Planet9 bug		6		
Saturday	4	Week 14						
Sunday	5	Week 14						
Monday	6	Week 15	6	Documenting USN-39 and making sprint report	6			
Tuesday	7	Week 15	6	Working in overleaf	6			



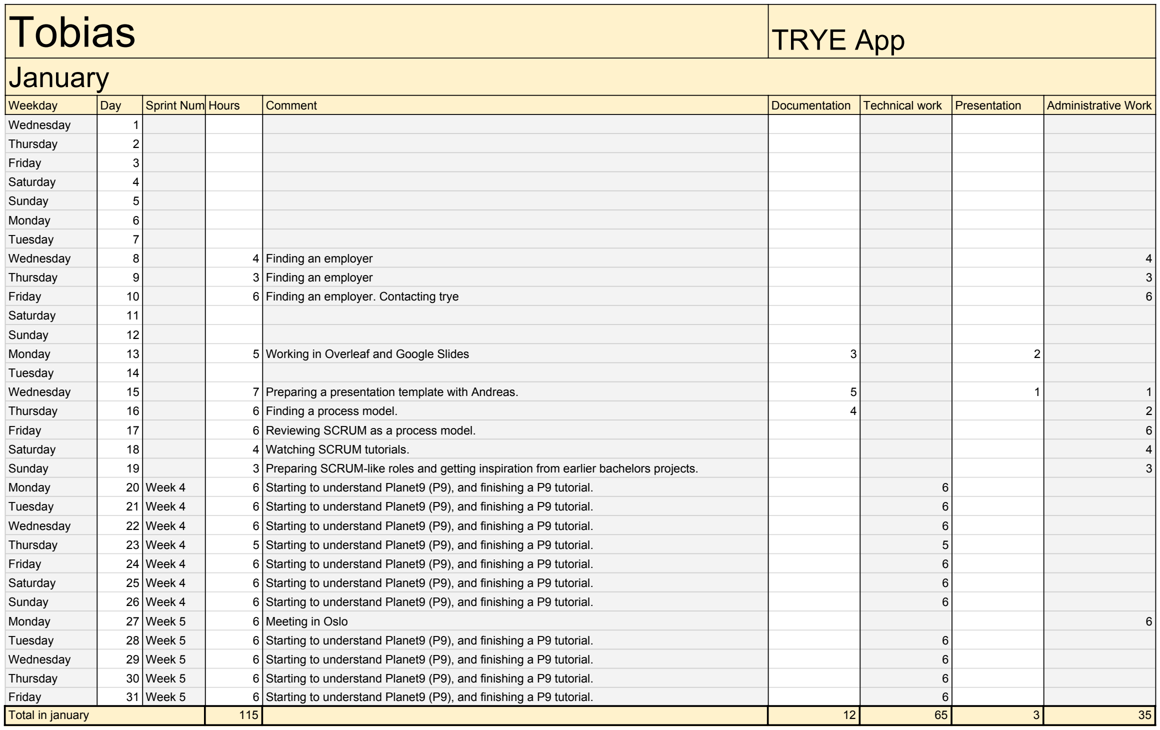
Wednesday	8	Week 15	6	Working in Planet9 with visuals for the app	6			
Thursday	9	Week 15	6	Working on visuals in the app		6		
Friday	10	Week 15						
Saturday	11	Week 15						
Sunday	12	Week 15						
Monday	13	Week 16						
Tuesday	14	Week 16						
Wednesday	15	Week 16						
Thursday	16	Week 16						
Friday	17	Week 16						
Saturday	18	Week 16						
Sunday	19	Week 16						
Monday	20	Week 17	8	Working in planet9 with user stories		8		
Tuesday	21	Week 17	8	Working in planet9 with user stories		8		
Wednesday	22	Week 17	8	Working in planet9 with user stories		8		
Thursday	23	Week 17	8	Working in planet9 with user stories		8		
Friday	24	Week 17	8	Working in planet9 with user stories		8		
Saturday	25	Week 17						
Sunday	26	Week 17						
Monday	27	Week 18	6	Working on the final report	6			
Tuesday	28	Week 18	6	Final report and stripe	3	3		
Wednesday	29	Week 18	6	Final report and planet9	3	3		
Thursday	30	Week 18	6	Documentation of user stories	6			
Total in April			106		36	70	0	0

May

Weekday	Day	Sprint Num	Hours	Comment	Documentation	Technical work	Presentation	Administrative Work
Friday	1	Week 18	6	Verifying and documents	6			
Saturday	2	Week 18	8	Finalizing the payment solution		8		
Sunday	3	Week 18	6	Documents and fixing bugs and visuals in the apps	3	3		
Monday	4	Week 19	6	Working on implementing stripe and documenting verifications	3	3		
Tuesday	5	Week 19	6	Working on final user stories and documenting verifications	4	2		
Wednesday	6	Week 19	8	Making a Admin application		8		
Thursday	7	Week 19	6	Finishing user stories and verification	2	4		
Friday	8	Week 19	6	Verification documents	6			
Saturday	9	Week 19	6	Verification documents	6			
Sunday	10	Week 19	6	Overleaf making a shell fotr the report	6			
Monday	11	Week 20	6	Writing about the project in the report	6			
Tuesday	12	Week 20	6	Writing about the project in the report	6			



Wednesday	13	Week 20	6	Writing on the final report	6			
Thursday	14	Week 20	6	Writing on the final report	6			
Friday	15	Week 20	6	Writing on the final report	6			
Saturday	16	Week 20	8	Writing on the final report	8			
Sunday	17	Week 20		**17th of May**				
Monday	18	Week 21	6	Writing on the final report	6			
Tuesday	19	Week 21	6	Writing on the final report	6			
Wednesday	20	Week 21	6	Writing on the final report	6			
Thursday	21	Week 21	2	Writing on the final report	2			
Friday	22	Week 21	8	Writing on the final report	8			
Saturday	23	Week 21	8	Fixing a reporting system and finalizing the admin app		8		
Sunday	24	Week 21	10	Finishing the report.	10			
Monday	25	Week 21	4	**DEADLINE FOR DELIVERING THE BACHELOR THESIS**	4			
Tuesday	26							
Wednesday	27							
Thursday	28							
Friday	29							
Saturday	30							
Sunday	31							
Total in May			152		116	36	0	0
Total for the entire project			505		223	220	26	26

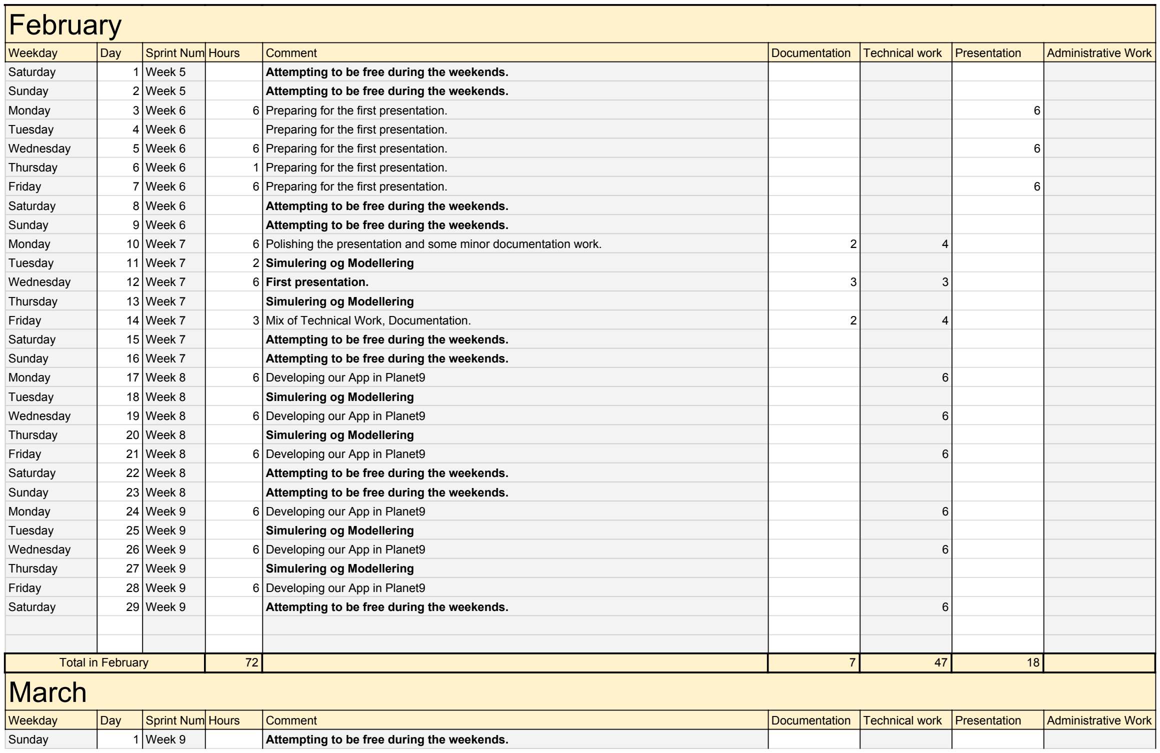


Tobias

TRYE App

January

Weekday	Day	Sprint Num	Hours	Comment	Documentation	Technical work	Presentation	Administrative Work
Wednesday	1							
Thursday	2							
Friday	3							
Saturday	4							
Sunday	5							
Monday	6							
Tuesday	7							
Wednesday	8		4	Finding an employer				4
Thursday	9		3	Finding an employer				3
Friday	10		6	Finding an employer. Contacting trye				6
Saturday	11							
Sunday	12							
Monday	13		5	Working in Overleaf and Google Slides	3		2	
Tuesday	14							
Wednesday	15		7	Preparing a presentation template with Andreas.	5		1	1
Thursday	16		6	Finding a process model.	4			2
Friday	17		6	Reviewing SCRUM as a process model.				6
Saturday	18		4	Watching SCRUM tutorials.				4
Sunday	19		3	Preparing SCRUM-like roles and getting inspiration from earlier bachelors projects.				3
Monday	20	Week 4	6	Starting to understand Planet9 (P9), and finishing a P9 tutorial.		6		
Tuesday	21	Week 4	6	Starting to understand Planet9 (P9), and finishing a P9 tutorial.		6		
Wednesday	22	Week 4	6	Starting to understand Planet9 (P9), and finishing a P9 tutorial.		6		
Thursday	23	Week 4	5	Starting to understand Planet9 (P9), and finishing a P9 tutorial.		5		
Friday	24	Week 4	6	Starting to understand Planet9 (P9), and finishing a P9 tutorial.		6		
Saturday	25	Week 4	6	Starting to understand Planet9 (P9), and finishing a P9 tutorial.		6		
Sunday	26	Week 4	6	Starting to understand Planet9 (P9), and finishing a P9 tutorial.		6		
Monday	27	Week 5	6	Meeting in Oslo				6
Tuesday	28	Week 5	6	Starting to understand Planet9 (P9), and finishing a P9 tutorial.		6		
Wednesday	29	Week 5	6	Starting to understand Planet9 (P9), and finishing a P9 tutorial.		6		
Thursday	30	Week 5	6	Starting to understand Planet9 (P9), and finishing a P9 tutorial.		6		
Friday	31	Week 5	6	Starting to understand Planet9 (P9), and finishing a P9 tutorial.		6		
Total in january			115		12	65	3	35

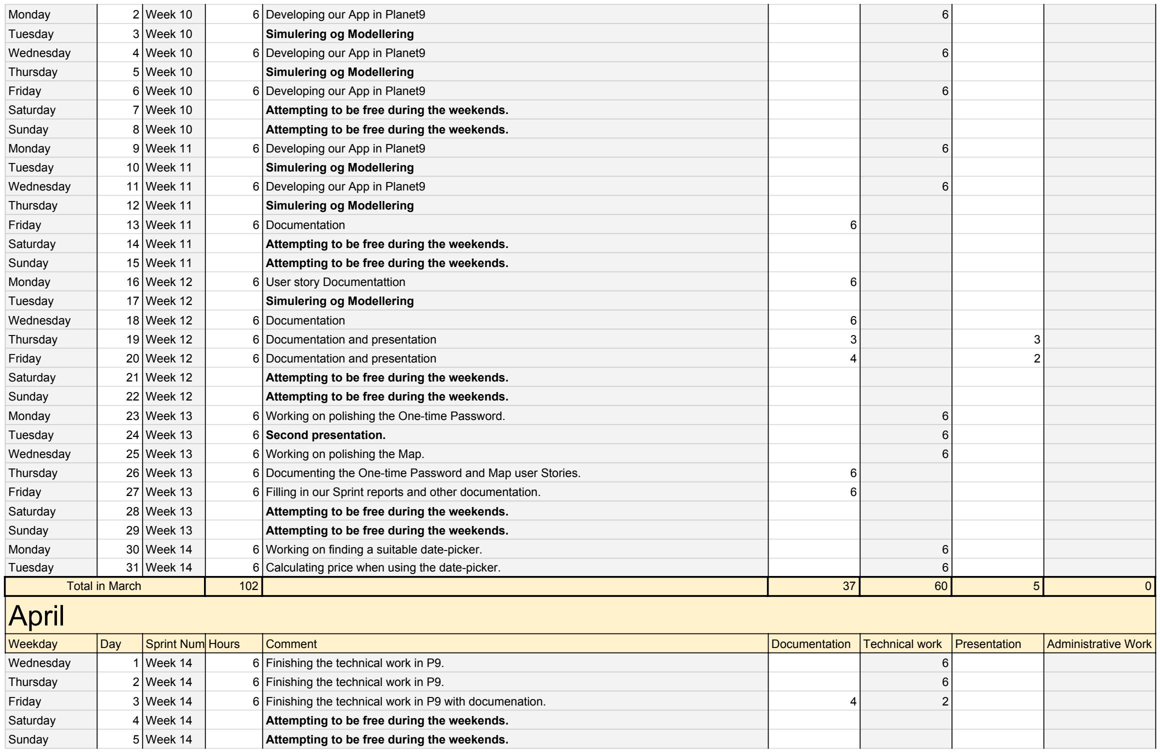


February

Weekday	Day	Sprint Num	Hours	Comment	Documentation	Technical work	Presentation	Administrative Work
Saturday	1	Week 5		**Attempting to be free during the weekends.**				
Sunday	2	Week 5		**Attempting to be free during the weekends.**				
Monday	3	Week 6	6	Preparing for the first presentation.			6	
Tuesday	4	Week 6		Preparing for the first presentation.				
Wednesday	5	Week 6	6	Preparing for the first presentation.			6	
Thursday	6	Week 6	1	Preparing for the first presentation.				
Friday	7	Week 6	6	Preparing for the first presentation.			6	
Saturday	8	Week 6		**Attempting to be free during the weekends.**				
Sunday	9	Week 6		**Attempting to be free during the weekends.**				
Monday	10	Week 7	6	Polishing the presentation and some minor documentation work.	2	4		
Tuesday	11	Week 7	2	**Simulering og Modellering**				
Wednesday	12	Week 7	6	**First presentation.**	3	3		
Thursday	13	Week 7		**Simulering og Modellering**				
Friday	14	Week 7	3	Mix of Technical Work, Documentation.	2	4		
Saturday	15	Week 7		**Attempting to be free during the weekends.**				
Sunday	16	Week 7		**Attempting to be free during the weekends.**				
Monday	17	Week 8	6	Developing our App in Planet9		6		
Tuesday	18	Week 8		**Simulering og Modellering**				
Wednesday	19	Week 8	6	Developing our App in Planet9		6		
Thursday	20	Week 8		**Simulering og Modellering**				
Friday	21	Week 8	6	Developing our App in Planet9		6		
Saturday	22	Week 8		**Attempting to be free during the weekends.**				
Sunday	23	Week 8		**Attempting to be free during the weekends.**				
Monday	24	Week 9	6	Developing our App in Planet9		6		
Tuesday	25	Week 9		**Simulering og Modellering**				
Wednesday	26	Week 9	6	Developing our App in Planet9		6		
Thursday	27	Week 9		**Simulering og Modellering**				
Friday	28	Week 9	6	Developing our App in Planet9				
Saturday	29	Week 9		**Attempting to be free during the weekends.**		6		
Total in February			72		7	47	18	

March

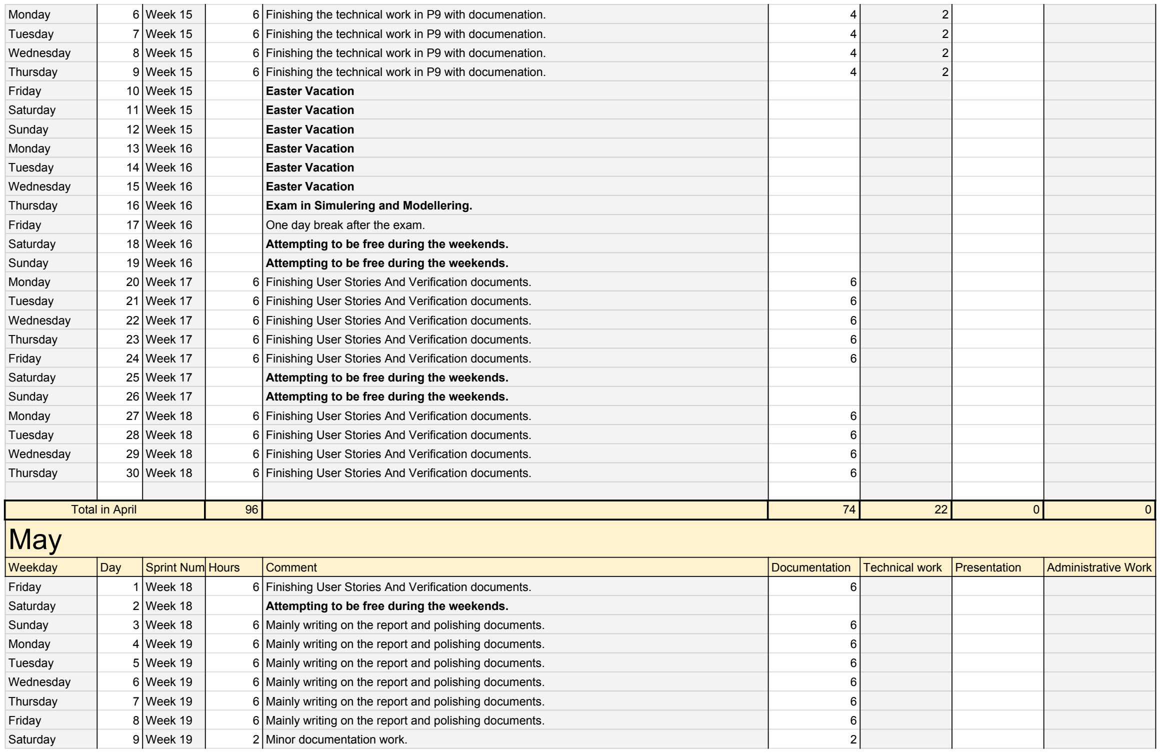
Weekday	Day	Sprint Num	Hours	Comment	Documentation	Technical work	Presentation	Administrative Work
Sunday	1	Week 9		**Attempting to be free during the weekends.**				



Monday	2	Week 10	6	Developing our App in Planet9		6		
Tuesday	3	Week 10		**Simulering og Modellering**				
Wednesday	4	Week 10	6	Developing our App in Planet9		6		
Thursday	5	Week 10		**Simulering og Modellering**				
Friday	6	Week 10	6	Developing our App in Planet9		6		
Saturday	7	Week 10		**Attempting to be free during the weekends.**				
Sunday	8	Week 10		**Attempting to be free during the weekends.**				
Monday	9	Week 11	6	Developing our App in Planet9		6		
Tuesday	10	Week 11		**Simulering og Modellering**				
Wednesday	11	Week 11	6	Developing our App in Planet9		6		
Thursday	12	Week 11		**Simulering og Modellering**				
Friday	13	Week 11	6	Documentation	6			
Saturday	14	Week 11		**Attempting to be free during the weekends.**				
Sunday	15	Week 11		**Attempting to be free during the weekends.**				
Monday	16	Week 12	6	User story Documentattion	6			
Tuesday	17	Week 12		**Simulering og Modellering**				
Wednesday	18	Week 12	6	Documentation	6			
Thursday	19	Week 12	6	Documentation and presentation	3		3	
Friday	20	Week 12	6	Documentation and presentation	4		2	
Saturday	21	Week 12		**Attempting to be free during the weekends.**				
Sunday	22	Week 12		**Attempting to be free during the weekends.**				
Monday	23	Week 13	6	Working on polishing the One-time Password.		6		
Tuesday	24	Week 13	6	**Second presentation.**		6		
Wednesday	25	Week 13	6	Working on polishing the Map.		6		
Thursday	26	Week 13	6	Documenting the One-time Password and Map user Stories.	6			
Friday	27	Week 13	6	Filling in our Sprint reports and other documentation.	6			
Saturday	28	Week 13		**Attempting to be free during the weekends.**				
Sunday	29	Week 13		**Attempting to be free during the weekends.**				
Monday	30	Week 14	6	Working on finding a suitable date-picker.		6		
Tuesday	31	Week 14	6	Calculating price when using the date-picker.		6		
Total in March			102		37	60	5	0

April

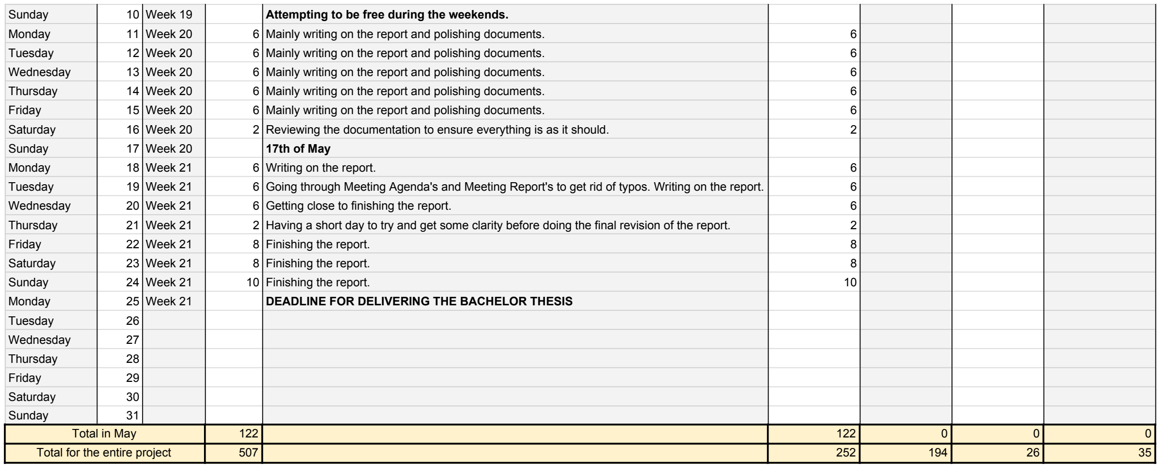
Weekday	Day	Sprint Num	Hours	Comment	Documentation	Technical work	Presentation	Administrative Work
Wednesday	1	Week 14	6	Finishing the technical work in P9.		6		
Thursday	2	Week 14	6	Finishing the technical work in P9.		6		
Friday	3	Week 14	6	Finishing the technical work in P9 with documenation.	4	2		
Saturday	4	Week 14		**Attempting to be free during the weekends.**				
Sunday	5	Week 14		**Attempting to be free during the weekends.**				



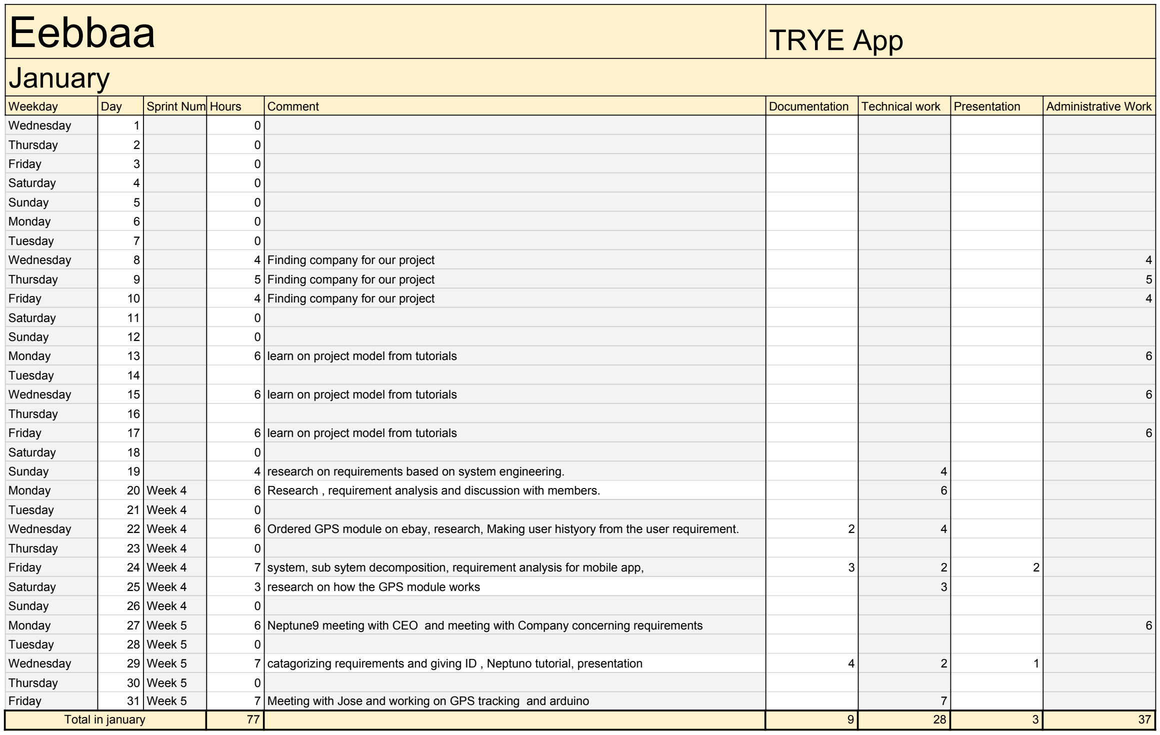
Monday	6	Week 15	6	Finishing the technical work in P9 with documenation.	4	2		
Tuesday	7	Week 15	6	Finishing the technical work in P9 with documenation.	4	2		
Wednesday	8	Week 15	6	Finishing the technical work in P9 with documenation.	4	2		
Thursday	9	Week 15	6	Finishing the technical work in P9 with documenation.	4	2		
Friday	10	Week 15		**Easter Vacation**				
Saturday	11	Week 15		**Easter Vacation**				
Sunday	12	Week 15		**Easter Vacation**				
Monday	13	Week 16		**Easter Vacation**				
Tuesday	14	Week 16		**Easter Vacation**				
Wednesday	15	Week 16		**Easter Vacation**				
Thursday	16	Week 16		**Exam in Simulering and Modellering.**				
Friday	17	Week 16		One day break after the exam.				
Saturday	18	Week 16		**Attempting to be free during the weekends.**				
Sunday	19	Week 16		**Attempting to be free during the weekends.**				
Monday	20	Week 17	6	Finishing User Stories And Verification documents.	6			
Tuesday	21	Week 17	6	Finishing User Stories And Verification documents.	6			
Wednesday	22	Week 17	6	Finishing User Stories And Verification documents.	6			
Thursday	23	Week 17	6	Finishing User Stories And Verification documents.	6			
Friday	24	Week 17	6	Finishing User Stories And Verification documents.	6			
Saturday	25	Week 17		**Attempting to be free during the weekends.**				
Sunday	26	Week 17		**Attempting to be free during the weekends.**				
Monday	27	Week 18	6	Finishing User Stories And Verification documents.	6			
Tuesday	28	Week 18	6	Finishing User Stories And Verification documents.	6			
Wednesday	29	Week 18	6	Finishing User Stories And Verification documents.	6			
Thursday	30	Week 18	6	Finishing User Stories And Verification documents.	6			
Total in April			96		74	22	0	0

May

Weekday	Day	Sprint Num	Hours	Comment	Documentation	Technical work	Presentation	Administrative Work
Friday	1	Week 18	6	Finishing User Stories And Verification documents.	6			
Saturday	2	Week 18		**Attempting to be free during the weekends.**				
Sunday	3	Week 18	6	Mainly writing on the report and polishing documents.	6			
Monday	4	Week 19	6	Mainly writing on the report and polishing documents.	6			
Tuesday	5	Week 19	6	Mainly writing on the report and polishing documents.	6			
Wednesday	6	Week 19	6	Mainly writing on the report and polishing documents.	6			
Thursday	7	Week 19	6	Mainly writing on the report and polishing documents.	6			
Friday	8	Week 19	6	Mainly writing on the report and polishing documents.	6			
Saturday	9	Week 19	2	Minor documentation work.	2			



Sunday	10	Week 19		**Attempting to be free during the weekends.**				
Monday	11	Week 20	6	Mainly writing on the report and polishing documents.	6			
Tuesday	12	Week 20	6	Mainly writing on the report and polishing documents.	6			
Wednesday	13	Week 20	6	Mainly writing on the report and polishing documents.	6			
Thursday	14	Week 20	6	Mainly writing on the report and polishing documents.	6			
Friday	15	Week 20	6	Mainly writing on the report and polishing documents.	6			
Saturday	16	Week 20	2	Reviewing the documentation to ensure everything is as it should.	2			
Sunday	17	Week 20		**17th of May**				
Monday	18	Week 21	6	Writing on the report.	6			
Tuesday	19	Week 21	6	Going through Meeting Agenda's and Meeting Report's to get rid of typos. Writing on the report.	6			
Wednesday	20	Week 21	6	Getting close to finishing the report.	6			
Thursday	21	Week 21	2	Having a short day to try and get some clarity before doing the final revision of the report.	2			
Friday	22	Week 21	8	Finishing the report.	8			
Saturday	23	Week 21	8	Finishing the report.	8			
Sunday	24	Week 21	10	Finishing the report.	10			
Monday	25	Week 21		**DEADLINE FOR DELIVERING THE BACHELOR THESIS**				
Tuesday	26							
Wednesday	27							
Thursday	28							
Friday	29							
Saturday	30							
Sunday	31							
Total in May			122		122	0	0	0
Total for the entire project			507		252	194	26	35

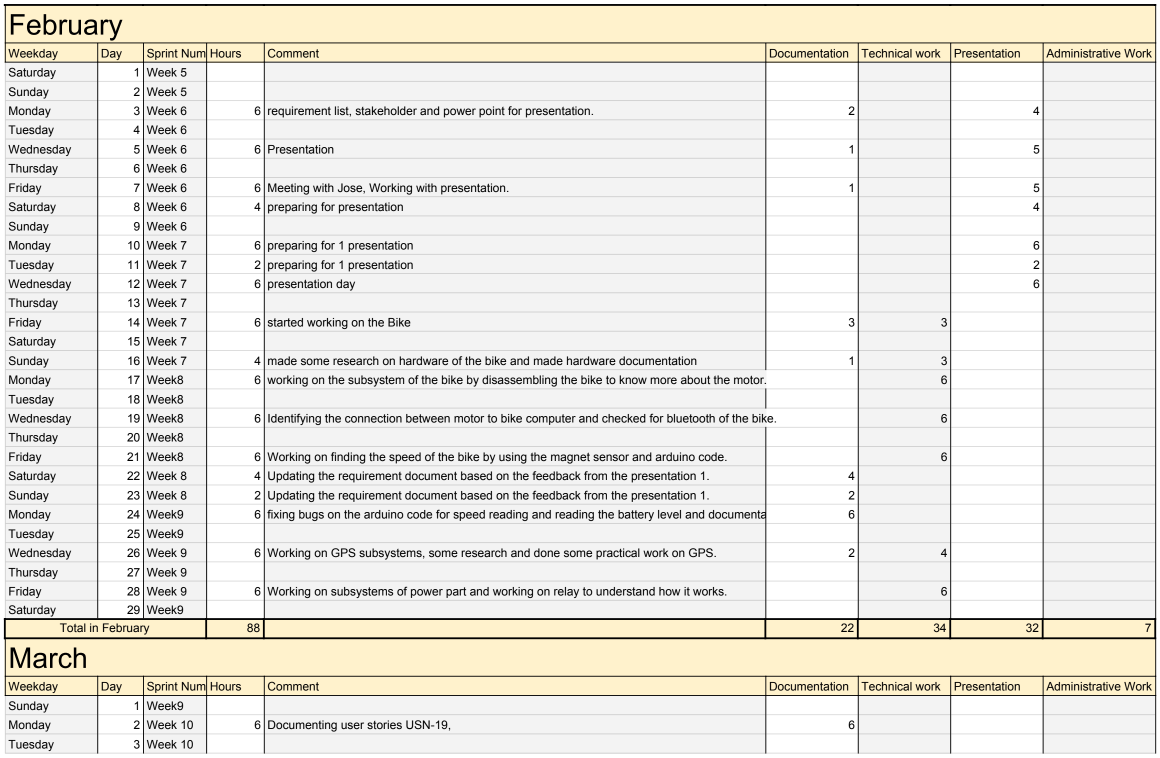


Eebbaa

TRYE App

January

Weekday	Day	Sprint Num	Hours	Comment	Documentation	Technical work	Presentation	Administrative Work
Wednesday	1		0					
Thursday	2		0					
Friday	3		0					
Saturday	4		0					
Sunday	5		0					
Monday	6		0					
Tuesday	7		0					
Wednesday	8		4	Finding company for our project				4
Thursday	9		5	Finding company for our project				5
Friday	10		4	Finding company for our project				4
Saturday	11		0					
Sunday	12		0					
Monday	13		6	learn on project model from tutorials				6
Tuesday	14							
Wednesday	15		6	learn on project model from tutorials				6
Thursday	16							
Friday	17		6	learn on project model from tutorials				6
Saturday	18		0					
Sunday	19		4	research on requirements based on system engineering.		4		
Monday	20	Week 4	6	Research , requirement analysis and discussion with members.		6		
Tuesday	21	Week 4	0					
Wednesday	22	Week 4	6	Ordered GPS module on ebay, research, Making user histyory from the user requirement.	2	4		
Thursday	23	Week 4	0					
Friday	24	Week 4	7	system, sub sytem decomposition, requirement analysis for mobile app,	3	2	2	
Saturday	25	Week 4	3	research on how the GPS module works		3		
Sunday	26	Week 4	0					
Monday	27	Week 5	6	Neptune9 meeting with CEO and meeting with Company concerning requirements				6
Tuesday	28	Week 5	0					
Wednesday	29	Week 5	7	catagorizing requirements and giving ID , Neptuno tutorial, presentation	4	2	1	
Thursday	30	Week 5	0					
Friday	31	Week 5	7	Meeting with Jose and working on GPS tracking and arduino		7		
Total in january			77		9	28	3	37

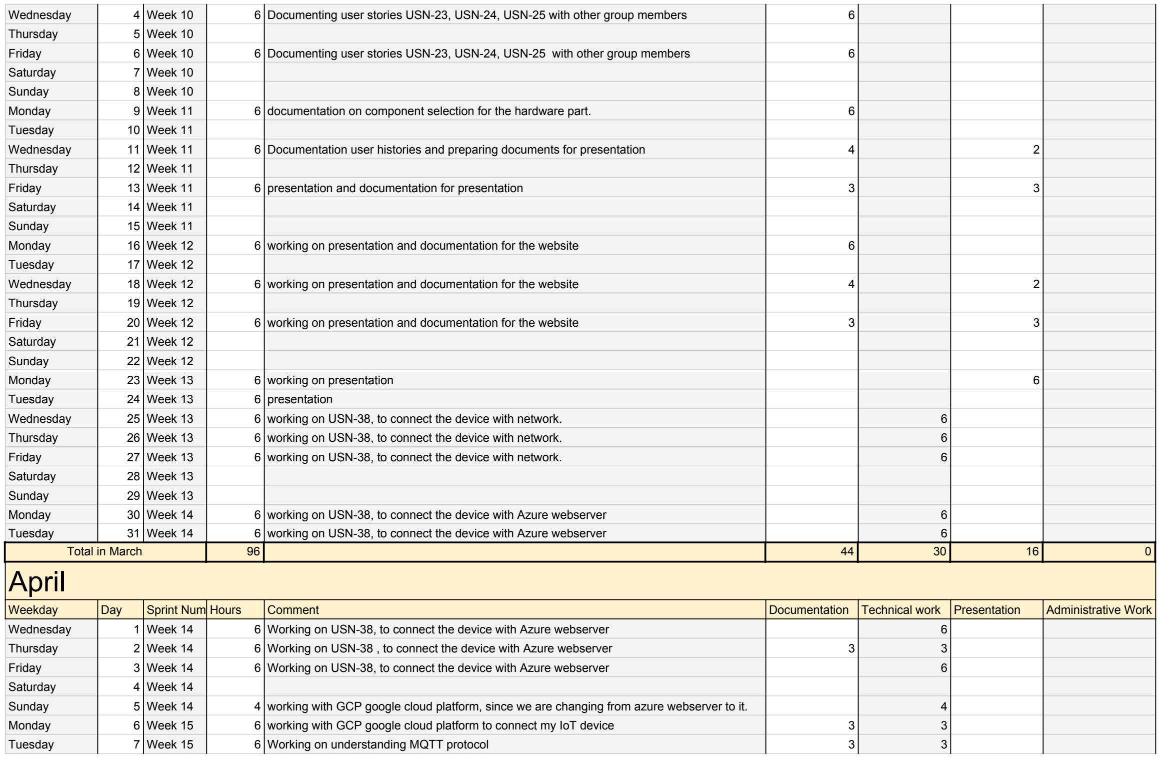


February

Weekday	Day	Sprint Num	Hours	Comment	Documentation	Technical work	Presentation	Administrative Work
Saturday	1	Week 5						
Sunday	2	Week 5						
Monday	3	Week 6	6	requirement list, stakeholder and power point for presentation.	2		4	
Tuesday	4	Week 6						
Wednesday	5	Week 6	6	Presentation	1		5	
Thursday	6	Week 6						
Friday	7	Week 6	6	Meeting with Jose, Working with presentation.	1		5	
Saturday	8	Week 6	4	preparing for presentation			4	
Sunday	9	Week 6						
Monday	10	Week 7	6	preparing for 1 presentation			6	
Tuesday	11	Week 7	2	preparing for 1 presentation			2	
Wednesday	12	Week 7	6	presentation day			6	
Thursday	13	Week 7						
Friday	14	Week 7	6	started working on the Bike	3	3		
Saturday	15	Week 7						
Sunday	16	Week 7	4	made some research on hardware of the bike and made hardware documentation	1	3		
Monday	17	Week8	6	working on the subsystem of the bike by disassembling the bike to know more about the motor.		6		
Tuesday	18	Week8						
Wednesday	19	Week8	6	Identifying the connection between motor to bike computer and checked for bluetooth of the bike.		6		
Thursday	20	Week8						
Friday	21	Week8	6	Working on finding the speed of the bike by using the magnet sensor and arduino code.		6		
Saturday	22	Week 8	4	Updating the requirement document based on the feedback from the presentation 1.	4			
Sunday	23	Week 8	2	Updating the requirement document based on the feedback from the presentation 1.	2			
Monday	24	Week9	6	fixing bugs on the arduino code for speed reading and reading the battery level and documenta	6			
Tuesday	25	Week9						
Wednesday	26	Week 9	6	Working on GPS subsystems, some research and done some practical work on GPS.	2	4		
Thursday	27	Week 9						
Friday	28	Week 9	6	Working on subsystems of power part and working on relay to understand how it works.		6		
Saturday	29	Week9						
Total in February			88		22	34	32	7

March

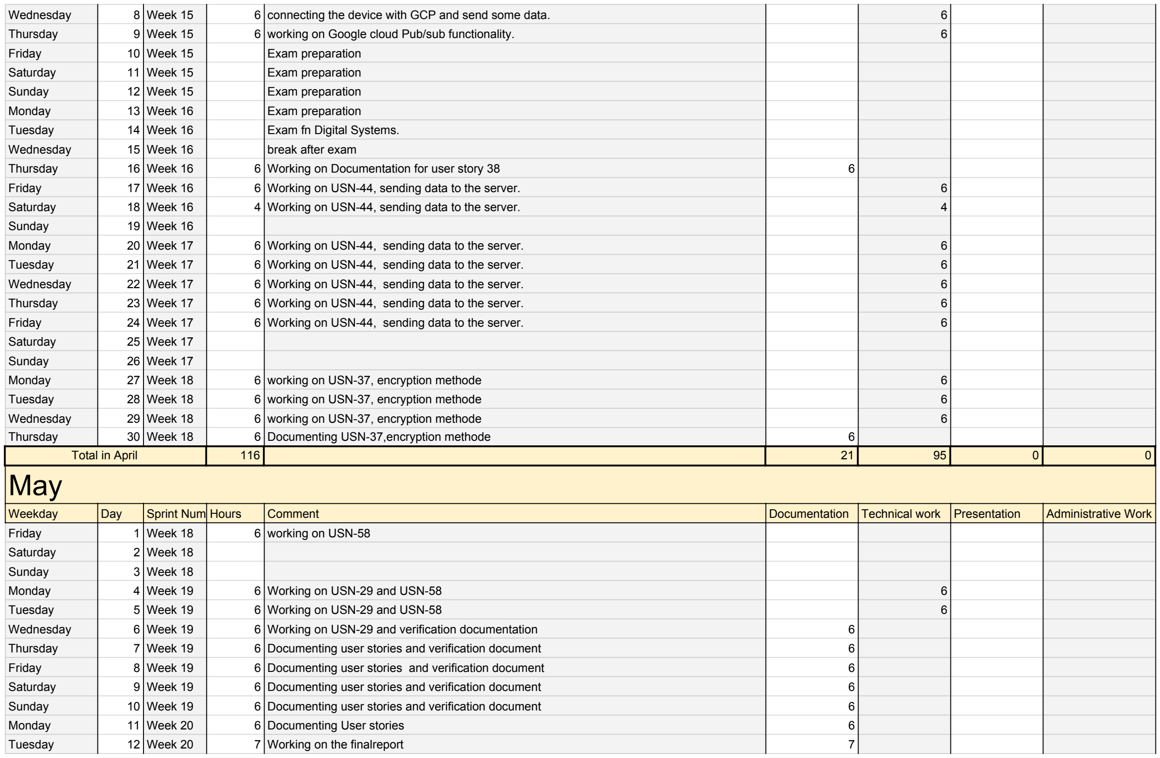
Weekday	Day	Sprint Num	Hours	Comment	Documentation	Technical work	Presentation	Administrative Work
Sunday	1	Week9						
Monday	2	Week 10	6	Documenting user stories USN-19,	6			
Tuesday	3	Week 10						



Wednesday	4	Week 10	6	Documenting user stories USN-23, USN-24, USN-25 with other group members	6			
Thursday	5	Week 10						
Friday	6	Week 10	6	Documenting user stories USN-23, USN-24, USN-25 with other group members	6			
Saturday	7	Week 10						
Sunday	8	Week 10						
Monday	9	Week 11	6	documentation on component selection for the hardware part.	6			
Tuesday	10	Week 11						
Wednesday	11	Week 11	6	Documentation user histories and preparing documents for presentation	4		2	
Thursday	12	Week 11						
Friday	13	Week 11	6	presentation and documentation for presentation	3		3	
Saturday	14	Week 11						
Sunday	15	Week 11						
Monday	16	Week 12	6	working on presentation and documentation for the website	6			
Tuesday	17	Week 12						
Wednesday	18	Week 12	6	working on presentation and documentation for the website	4		2	
Thursday	19	Week 12						
Friday	20	Week 12	6	working on presentation and documentation for the website	3		3	
Saturday	21	Week 12						
Sunday	22	Week 12						
Monday	23	Week 13	6	working on presentation			6	
Tuesday	24	Week 13	6	presentation				
Wednesday	25	Week 13	6	working on USN-38, to connect the device with network.		6		
Thursday	26	Week 13	6	working on USN-38, to connect the device with network.		6		
Friday	27	Week 13	6	working on USN-38, to connect the device with network.		6		
Saturday	28	Week 13						
Sunday	29	Week 13						
Monday	30	Week 14	6	working on USN-38, to connect the device with Azure webserver		6		
Tuesday	31	Week 14	6	working on USN-38, to connect the device with Azure webserver		6		
Total in March			96		44	30	16	0

April

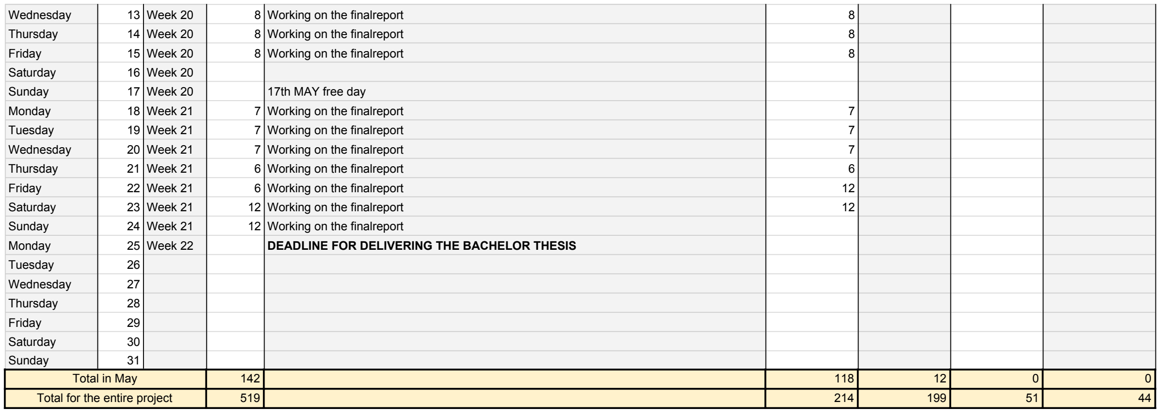
Weekday	Day	Sprint Num	Hours	Comment	Documentation	Technical work	Presentation	Administrative Work
Wednesday	1	Week 14	6	Working on USN-38, to connect the device with Azure webserver		6		
Thursday	2	Week 14	6	Working on USN-38 , to connect the device with Azure webserver	3	3		
Friday	3	Week 14	6	Working on USN-38, to connect the device with Azure webserver		6		
Saturday	4	Week 14						
Sunday	5	Week 14	4	working with GCP google cloud platform, since we are changing from azure webserver to it.		4		
Monday	6	Week 15	6	working with GCP google cloud platform to connect my IoT device	3	3		
Tuesday	7	Week 15	6	Working on understanding MQTT protocol	3	3		



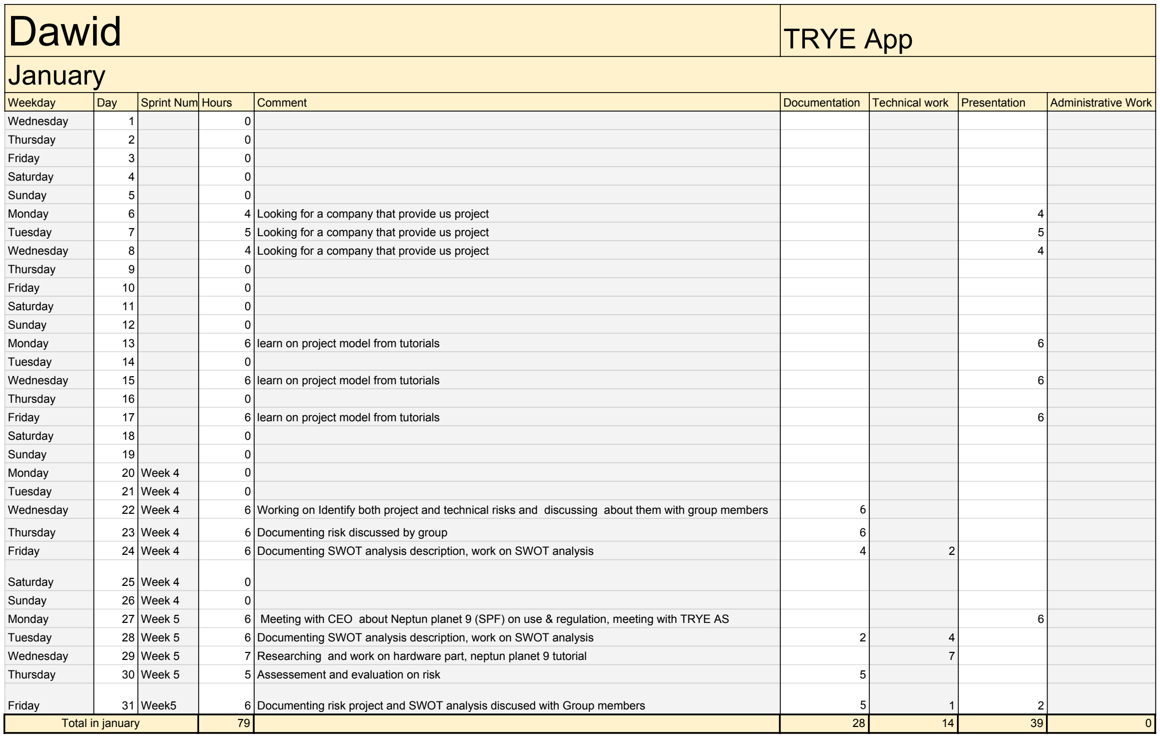
Wednesday	8	Week 15	6	connecting the device with GCP and send some data.		6		
Thursday	9	Week 15	6	working on Google cloud Pub/sub functionality.		6		
Friday	10	Week 15		Exam preparation				
Saturday	11	Week 15		Exam preparation				
Sunday	12	Week 15		Exam preparation				
Monday	13	Week 16		Exam preparation				
Tuesday	14	Week 16		Exam fn Digital Systems.				
Wednesday	15	Week 16		break after exam				
Thursday	16	Week 16	6	Working on Documentation for user story 38	6			
Friday	17	Week 16	6	Working on USN-44, sending data to the server.		6		
Saturday	18	Week 16	4	Working on USN-44, sending data to the server.		4		
Sunday	19	Week 16						
Monday	20	Week 17	6	Working on USN-44, sending data to the server.		6		
Tuesday	21	Week 17	6	Working on USN-44, sending data to the server.		6		
Wednesday	22	Week 17	6	Working on USN-44, sending data to the server.		6		
Thursday	23	Week 17	6	Working on USN-44, sending data to the server.		6		
Friday	24	Week 17	6	Working on USN-44, sending data to the server.		6		
Saturday	25	Week 17						
Sunday	26	Week 17						
Monday	27	Week 18	6	working on USN-37, encryption methode		6		
Tuesday	28	Week 18	6	working on USN-37, encryption methode		6		
Wednesday	29	Week 18	6	working on USN-37, encryption methode		6		
Thursday	30	Week 18	6	Documenting USN-37,encryption methode	6			
Total in April			116		21	95	0	0

May

Weekday	Day	Sprint Num	Hours	Comment	Documentation	Technical work	Presentation	Administrative Work
Friday	1	Week 18	6	working on USN-58				
Saturday	2	Week 18						
Sunday	3	Week 18						
Monday	4	Week 19	6	Working on USN-29 and USN-58		6		
Tuesday	5	Week 19	6	Working on USN-29 and USN-58		6		
Wednesday	6	Week 19	6	Working on USN-29 and verification documentation	6			
Thursday	7	Week 19	6	Documenting user stories and verification document	6			
Friday	8	Week 19	6	Documenting user stories and verification document	6			
Saturday	9	Week 19	6	Documenting user stories and verification document	6			
Sunday	10	Week 19	6	Documenting user stories and verification document	6			
Monday	11	Week 20	6	Documenting User stories	6			
Tuesday	12	Week 20	7	Working on the finalreport	7			



Wednesday	13	Week 20	8	Working on the finalreport	8			
Thursday	14	Week 20	8	Working on the finalreport	8			
Friday	15	Week 20	8	Working on the finalreport	8			
Saturday	16	Week 20						
Sunday	17	Week 20		17th MAY free day				
Monday	18	Week 21	7	Working on the finalreport	7			
Tuesday	19	Week 21	7	Working on the finalreport	7			
Wednesday	20	Week 21	7	Working on the finalreport	7			
Thursday	21	Week 21	6	Working on the finalreport	6			
Friday	22	Week 21	6	Working on the finalreport	12			
Saturday	23	Week 21	12	Working on the finalreport	12			
Sunday	24	Week 21	12	Working on the finalreport				
Monday	25	Week 22		**DEADLINE FOR DELIVERING THE BACHELOR THESIS**				
Tuesday	26							
Wednesday	27							
Thursday	28							
Friday	29							
Saturday	30							
Sunday	31							
Total in May			142		118	12	0	0
Total for the entire project			519		214	199	51	44

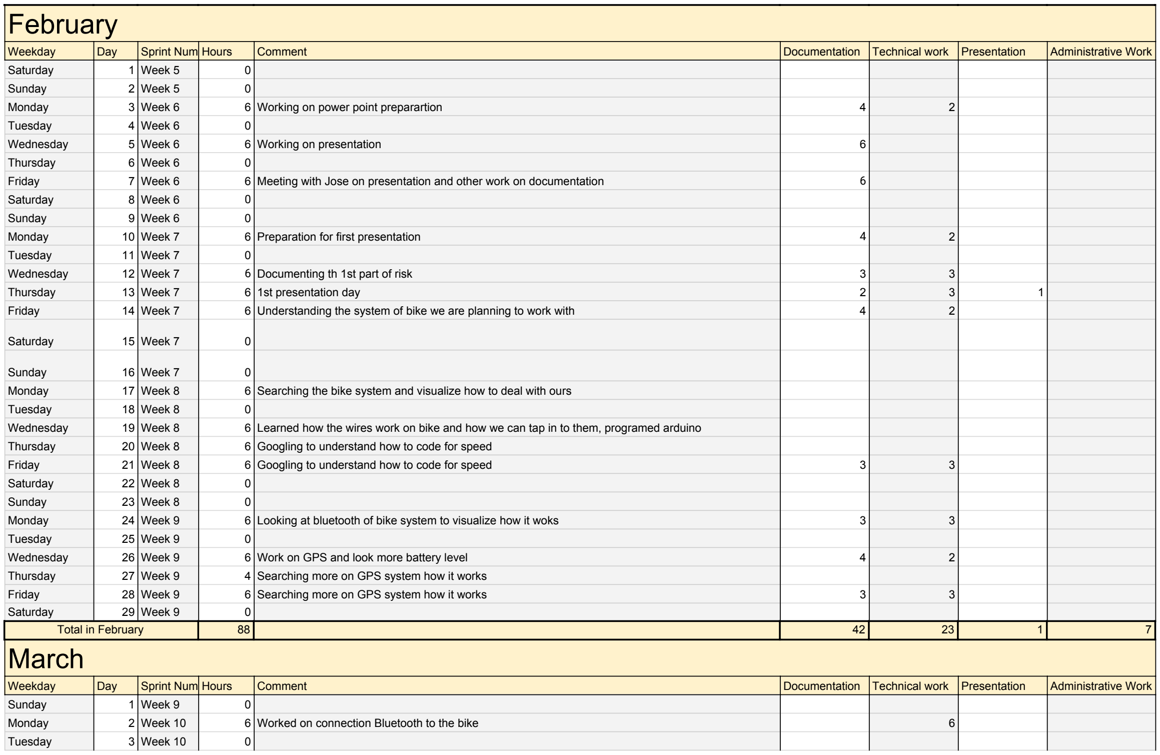


Dawid

TRYE App

January

Weekday	Day	Sprint Num	Hours	Comment	Documentation	Technical work	Presentation	Administrative Work
Wednesday	1		0					
Thursday	2		0					
Friday	3		0					
Saturday	4		0					
Sunday	5		0					
Monday	6		4	Looking for a company that provide us project			4	
Tuesday	7		5	Looking for a company that provide us project			5	
Wednesday	8		4	Looking for a company that provide us project			4	
Thursday	9		0					
Friday	10		0					
Saturday	11		0					
Sunday	12		0					
Monday	13		6	learn on project model from tutorials			6	
Tuesday	14		0					
Wednesday	15		6	learn on project model from tutorials			6	
Thursday	16		0					
Friday	17		6	learn on project model from tutorials			6	
Saturday	18		0					
Sunday	19		0					
Monday	20	Week 4	0					
Tuesday	21	Week 4	0					
Wednesday	22	Week 4	6	Working on Identify both project and technical risks and discussing about them with group members	6			
Thursday	23	Week 4	6	Documenting risk discussed by group	6			
Friday	24	Week 4	6	Documenting SWOT analysis description, work on SWOT analysis	4	2		
Saturday	25	Week 4	0					
Sunday	26	Week 4	0					
Monday	27	Week 5	6	Meeting with CEO about Neptun planet 9 (SPF) on use & regulation, meeting with TRYE AS			6	
Tuesday	28	Week 5	6	Documenting SWOT analysis description, work on SWOT analysis	2	4		
Wednesday	29	Week 5	7	Researching and work on hardware part, neptun planet 9 tutorial		7		
Thursday	30	Week 5	5	Assessement and evaluation on risk	5			
Friday	31	Week5	6	Documenting risk project and SWOT analysis discused with Group members	5	1	2	
Total in january			79		28	14	39	0

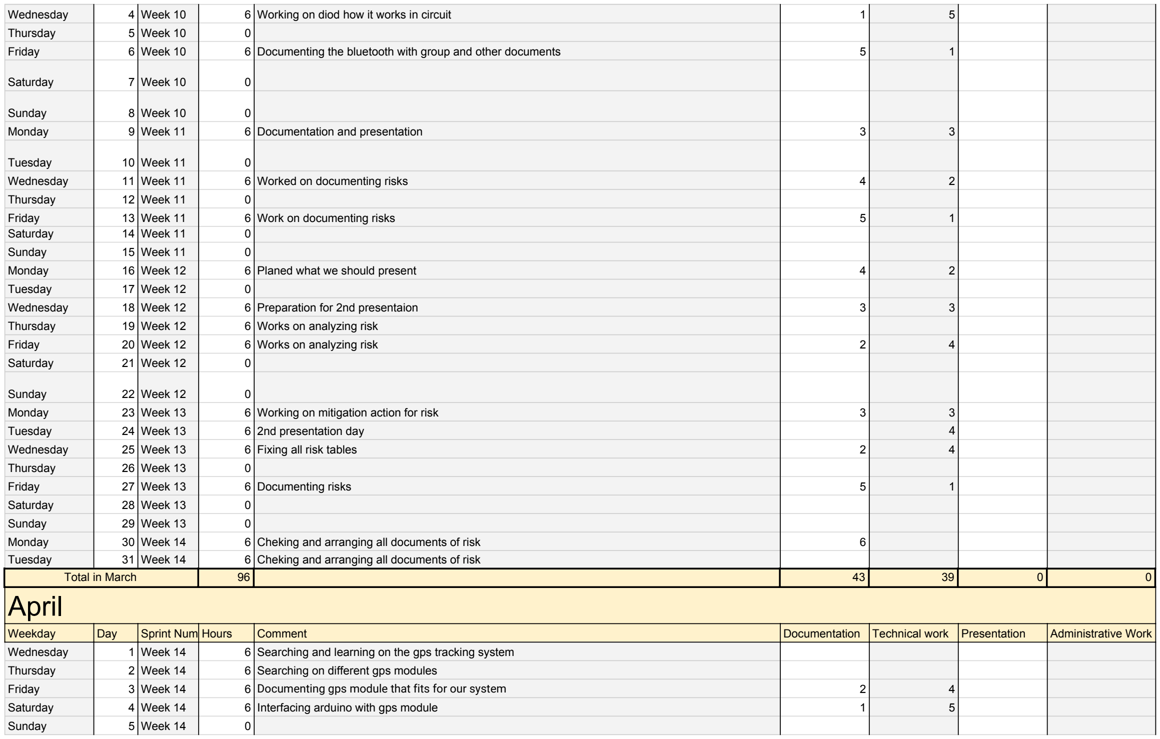


February

Weekday	Day	Sprint Num	Hours	Comment	Documentation	Technical work	Presentation	Administrative Work
Saturday	1	Week 5	0					
Sunday	2	Week 5	0					
Monday	3	Week 6	6	Working on power point preparartion	4	2		
Tuesday	4	Week 6	0					
Wednesday	5	Week 6	6	Working on presentation	6			
Thursday	6	Week 6	0					
Friday	7	Week 6	6	Meeting with Jose on presentation and other work on documentation	6			
Saturday	8	Week 6	0					
Sunday	9	Week 6	0					
Monday	10	Week 7	6	Preparation for first presentation	4	2		
Tuesday	11	Week 7	0					
Wednesday	12	Week 7	6	Documenting th 1st part of risk	3	3		
Thursday	13	Week 7	6	1st presentation day	2	3	1	
Friday	14	Week 7	6	Understanding the system of bike we are planning to work with	4	2		
Saturday	15	Week 7	0					
Sunday	16	Week 7	0					
Monday	17	Week 8	6	Searching the bike system and visualize how to deal with ours				
Tuesday	18	Week 8	0					
Wednesday	19	Week 8	6	Learned how the wires work on bike and how we can tap in to them, programed arduino				
Thursday	20	Week 8	6	Googling to understand how to code for speed				
Friday	21	Week 8	6	Googling to understand how to code for speed	3	3		
Saturday	22	Week 8	0					
Sunday	23	Week 8	0					
Monday	24	Week 9	6	Looking at bluetooth of bike system to visualize how it woks	3	3		
Tuesday	25	Week 9	0					
Wednesday	26	Week 9	6	Work on GPS and look more battery level	4	2		
Thursday	27	Week 9	4	Searching more on GPS system how it works				
Friday	28	Week 9	6	Searching more on GPS system how it works	3	3		
Saturday	29	Week 9	0					
Total in February			88		42	23	1	7

March

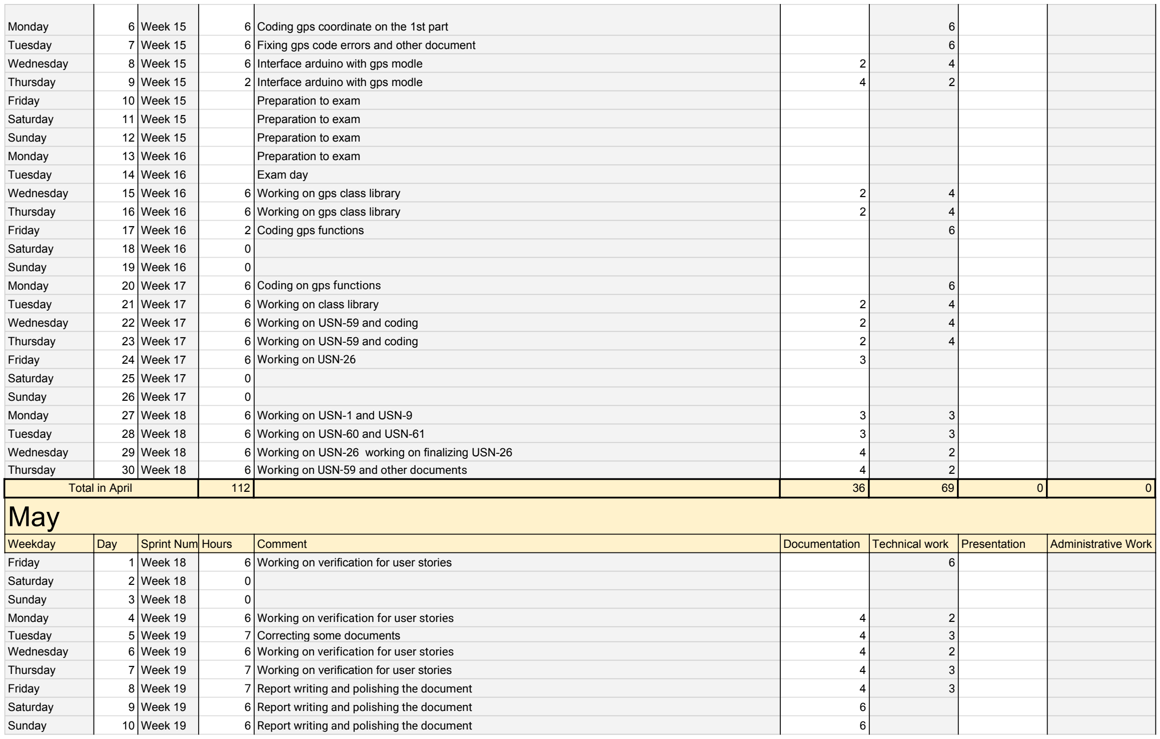
Weekday	Day	Sprint Num	Hours	Comment	Documentation	Technical work	Presentation	Administrative Work
Sunday	1	Week 9	0					
Monday	2	Week 10	6	Worked on connection Bluetooth to the bike		6		
Tuesday	3	Week 10	0					



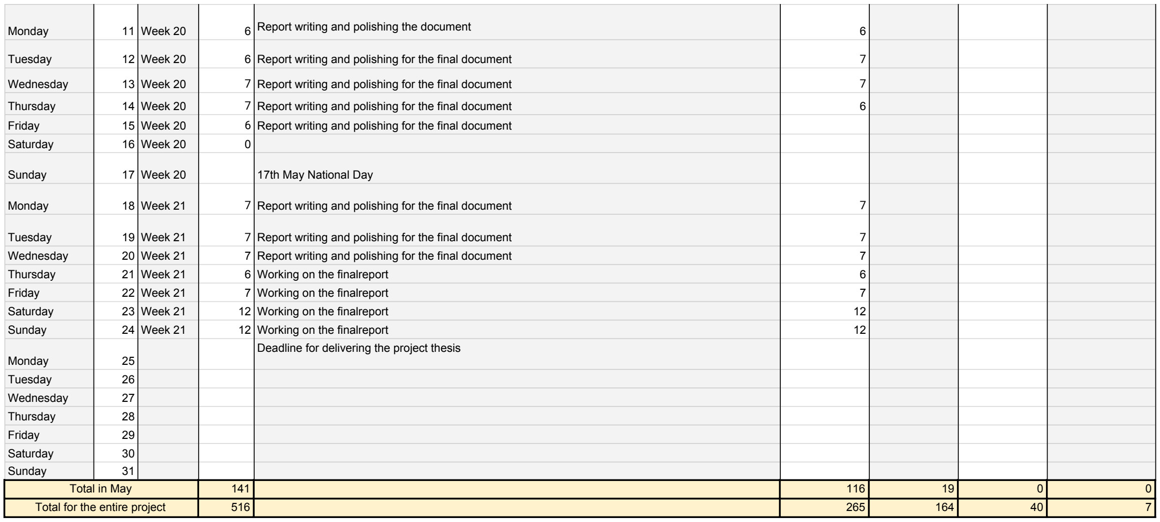
Wednesday	4	Week 10	6	Working on diod how it works in circuit	1	5		
Thursday	5	Week 10	0					
Friday	6	Week 10	6	Documenting the bluetooth with group and other documents	5	1		
Saturday	7	Week 10	0					
Sunday	8	Week 10	0					
Monday	9	Week 11	6	Documentation and presentation	3	3		
Tuesday	10	Week 11	0					
Wednesday	11	Week 11	6	Worked on documenting risks	4	2		
Thursday	12	Week 11	0					
Friday	13	Week 11	6	Work on documenting risks	5	1		
Saturday	14	Week 11	0					
Sunday	15	Week 11	0					
Monday	16	Week 12	6	Planed what we should present	4	2		
Tuesday	17	Week 12	0					
Wednesday	18	Week 12	6	Preparation for 2nd presentaion	3	3		
Thursday	19	Week 12	6	Works on analyzing risk				
Friday	20	Week 12	6	Works on analyzing risk	2	4		
Saturday	21	Week 12	0					
Sunday	22	Week 12	0					
Monday	23	Week 13	6	Working on mitigation action for risk	3	3		
Tuesday	24	Week 13	6	2nd presentation day		4		
Wednesday	25	Week 13	6	Fixing all risk tables	2	4		
Thursday	26	Week 13	0					
Friday	27	Week 13	6	Documenting risks	5	1		
Saturday	28	Week 13	0					
Sunday	29	Week 13	0					
Monday	30	Week 14	6	Cheking and arranging all documents of risk	6			
Tuesday	31	Week 14	6	Cheking and arranging all documents of risk				
Total in March			96		43	39	0	0

April

Weekday	Day	Sprint Num	Hours	Comment	Documentation	Technical work	Presentation	Administrative Work
Wednesday	1	Week 14	6	Searching and learning on the gps tracking system				
Thursday	2	Week 14	6	Searching on different gps modules				
Friday	3	Week 14	6	Documenting gps module that fits for our system	2	4		
Saturday	4	Week 14	6	Interfacing arduino with gps module	1	5		
Sunday	5	Week 14	0					



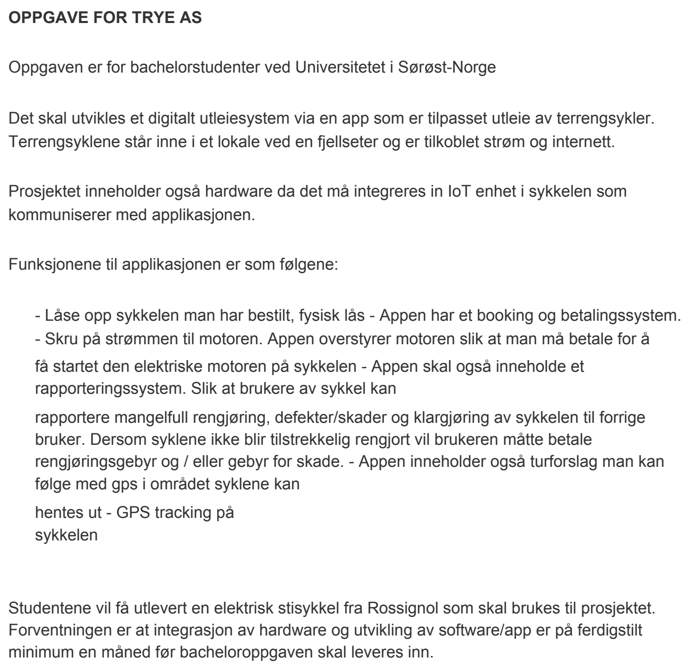
Monday	6	Week 15	6	Coding gps coordinate on the 1st part		6		
Tuesday	7	Week 15	6	Fixing gps code errors and other document		6		
Wednesday	8	Week 15	6	Interface arduino with gps modle	2	4		
Thursday	9	Week 15	2	Interface arduino with gps modle	4	2		
Friday	10	Week 15		Preparation to exam				
Saturday	11	Week 15		Preparation to exam				
Sunday	12	Week 15		Preparation to exam				
Monday	13	Week 16		Preparation to exam				
Tuesday	14	Week 16		Exam day				
Wednesday	15	Week 16	6	Working on gps class library	2	4		
Thursday	16	Week 16	6	Working on gps class library	2	4		
Friday	17	Week 16	2	Coding gps functions		6		
Saturday	18	Week 16	0					
Sunday	19	Week 16	0					
Monday	20	Week 17	6	Coding on gps functions		6		
Tuesday	21	Week 17	6	Working on class library	2	4		
Wednesday	22	Week 17	6	Working on USN-59 and coding	2	4		
Thursday	23	Week 17	6	Working on USN-59 and coding	2	4		
Friday	24	Week 17	6	Working on USN-26	3			
Saturday	25	Week 17	0					
Sunday	26	Week 17	0					
Monday	27	Week 18	6	Working on USN-1 and USN-9	3	3		
Tuesday	28	Week 18	6	Working on USN-60 and USN-61	3	3		
Wednesday	29	Week 18	6	Working on USN-26 working on finalizing USN-26	4	2		
Thursday	30	Week 18	6	Working on USN-59 and other documents	4	2		
Total in April			112		36	69	0	0
May								
Weekday	Day	Sprint Num	Hours	Comment	Documentation	Technical work	Presentation	Administrative Work
Friday	1	Week 18	6	Working on verification for user stories		6		
Saturday	2	Week 18	0					
Sunday	3	Week 18	0					
Monday	4	Week 19	6	Working on verification for user stories	4	2		
Tuesday	5	Week 19	7	Correcting some documents	4	3		
Wednesday	6	Week 19	6	Working on verification for user stories	4	2		
Thursday	7	Week 19	7	Working on verification for user stories	4	3		
Friday	8	Week 19	7	Report writing and polishing the document	4	3		
Saturday	9	Week 19	6	Report writing and polishing the document	6			
Sunday	10	Week 19	6	Report writing and polishing the document	6			



Monday	11	Week 20	6	Report writing and polishing the document	6			
Tuesday	12	Week 20	6	Report writing and polishing for the final document	7			
Wednesday	13	Week 20	7	Report writing and polishing for the final document	7			
Thursday	14	Week 20	7	Report writing and polishing for the final document	6			
Friday	15	Week 20	6	Report writing and polishing for the final document				
Saturday	16	Week 20	0					
Sunday	17	Week 20		17th May National Day				
Monday	18	Week 21	7	Report writing and polishing for the final document	7			
Tuesday	19	Week 21	7	Report writing and polishing for the final document	7			
Wednesday	20	Week 21	7	Report writing and polishing for the final document	7			
Thursday	21	Week 21	6	Working on the finalreport	6			
Friday	22	Week 21	7	Working on the finalreport	7			
Saturday	23	Week 21	12	Working on the finalreport	12			
Sunday	24	Week 21	12	Working on the finalreport	12			
Monday	25			Deadline for delivering the project thesis				
Tuesday	26							
Wednesday	27							
Thursday	28							
Friday	29							
Saturday	30							
Sunday	31							
Total in May			141		116	19	0	0
Total for the entire project			516		265	164	40	7

A.15 Project assignment and terms

A.15.1 Assignment from TRYE



OPPGAVE FOR TRYE AS

Oppgaven er for bachelorstudenter ved Universitetet i Sørøst-Norge

Det skal utvikles et digitalt utleiesystem via en app som er tilpasset utleie av terrengsykler. Terrengsyklene står inne i et lokale ved en fjellseter og er tilkoblet strøm og internett.

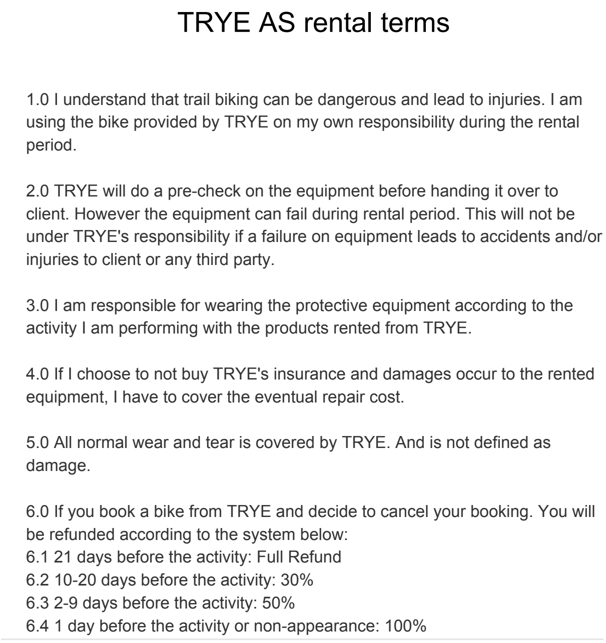
Prosjektet inneholder også hardware da det må integreres in IoT enhet i sykkelen som kommuniserer med applikasjonen.

Funksjonene til applikasjonen er som følgene:

- Låse opp sykkelen man har bestilt, fysisk lås - Appen har et booking og betalingssystem.
- Skru på strømmen til motoren. Appen overstyrer motoren slik at man må betale for å få startet den elektriske motoren på sykkelen - Appen skal også inneholde et rapporteringssystem. Slik at brukere av sykkel kan rapportere mangelfull rengjøring, defekter/skader og klargjøring av sykkelen til forrige bruker. Dersom syklene ikke blir tilstrekkelig rengjort vil brukeren måtte betale rengjøringsgebyr og / eller gebyr for skade. - Appen inneholder også turforslag man kan følge med gps i området syklene kan hentes ut - GPS tracking på sykkelen

Studentene vil få utlevert en elektrisk stisykkel fra Rossignol som skal brukes til prosjektet. Forventningen er at integrasjon av hardware og utvikling av software/app er på ferdigstilt minimum en måned før bacheloroppgaven skal leveres inn.

A.15.2 Rental terms from TRYE



TRYE AS rental terms

1.0 I understand that trail biking can be dangerous and lead to injuries. I am using the bike provided by TRYE on my own responsibility during the rental period.

2.0 TRYE will do a pre-check on the equipment before handing it over to client. However the equipment can fail during rental period. This will not be under TRYE's responsibility if a failure on equipment leads to accidents and/or injuries to client or any third party.

3.0 I am responsible for wearing the protective equipment according to the activity I am performing with the products rented from TRYE.

4.0 If I choose to not buy TRYE's insurance and damages occur to the rented equipment, I have to cover the eventual repair cost.

5.0 All normal wear and tear is covered by TRYE. And is not defined as damage.

6.0 If you book a bike from TRYE and decide to cancel your booking. You will be refunded according to the system below:
6.1 21 days before the activity: Full Refund
6.2 10-20 days before the activity: 30%
6.3 2-9 days before the activity: 50%
6.4 1 day before the activity or non-appearance: 100%