Remote Lab

Anders Kristiansen
Jan Helge Slettebø
Espen Rønningen
Christian Scott

22. mai 2018

Versjon

Versjon	Beskrivelse	Forfatter
0.1.0	Førsteutkast	Christian Scott
0.2.0	Laget disposisjon til sluttdokumentet	Jan Helge Slettebø
0.2.1	La inn introduksjon(dårlig versjon)	Jan Helge Slettebø
0.2.2	Bearbeidet og fornyet introduksjon	Espen Rønningen
0.2.3	La inn risiko	Jan Helge Slettebø
0.2.4	La inn teststrategier	Anders Kristiansen
0.2.5	La inn teknologi	Anders Kristiansen
0.2.6	La inn litt mer teknologi	Jan Helge Slettebø
0.2.7	La inn krav og userstory	Anders Kristiansen
0.2.8	La inn UI under implementasjon	Anders Kristiansen
0.2.9	La inn køsystem under implementasjon	Espen Rønningen
0.3.0	La inn prosessen	Christian Scott
0.3.1	La inn mer om prosessen + backlog	Christian Scott
0.3.2	La inn nettsiden under implementasjon	Jan Helge Slettebø
0.3.3	La inn produktdesign	Christian Scott
0.3.4	La inn gruppemedlemmer i introduksjon	Espen Rønningen
0.3.5	La inn user story tabell og test tabell	Anders Kristiansen
0.3.6	La inn core under implementasjon	Christian Scott
0.3.7	La inn interface og protokoll under implementasjon	Jan Helge Slettebø
0.3.8	La inn nettverk under implementasjon	Jan Helge Slettebø
0.3.9	La inn kravtabeller	Espen Rønningen
0.4.0	La inn Sprint Reviews, Sprint Retrospective og Sprint planning	Espen Rønningen
0.4.1	La inn Møtereferat	Espen Rønningen

Sammendrag

Universitetet i Sørøst-Norge har et ønske om å ta i bruk remote lab eksperimenter. Vår gruppe har fått i oppgave å lage et rammeverk for en remote lab. En remote lab er en metode for å utføre fysiske eksperimenter uten å være fysisk tilstede. Det er annerledes enn simulering, da simuleringer tar hensyn til alle faktorer, mens fysiske eksperimenter gjør det. Et rammeverk er et grunnlag for å lage en fullverdig remote lab. Vår gruppe samarbeider med en annen bachelorgruppe(RELEASE) hvor de lager rammeverk for eksperimenter, mens vi lager rammeverk for systemet som lar deg kjøre eksperimentene når du ikke er tilstede.

Innhold

1 Introduksjon

USN har en gammel og utgått Remote Lab [1] og ønsker en fornyet og moderne løsning som inkluderer et rammeverk og initiell implementering av en Remote Lab løsning. Denne skal gjøre det mulig for andre - både de som er og ikke er tilknyttet USN - å kjøre eksperimenter, uavhengig av økonomi, tidspunkt og hvor de befinner seg.

Eksperimentene kan tilhøre forskjellige fagområder som fysikk, elektronikk, mekanikk, matematikk med fler. Rammeverket skal gjøre det mulig for andre - også de uten særlig kompetanse innen koding og utvikling - å legge til sine egne eksperimenter. Rammeverket skal kunne brukes av undervisere til å sette opp egne løsninger og eksperimenter i forbindelse med undervisning.

I den forbindelse har USN gitt oss i bacheloroppgave å lage et rammeverk for “Remote Lab” som er lett å utvide med nye eksperimenter av alle typer.

Gruppen består av fire datastudenter, og skal lage programvaren som lar brukeren kjøre og analysere eksperimentene. Vi skal gjøre det enkelt å legge til nye eksperimenter uten at eksperimentutvikleren trenger å kunne mye om utvikling av UI og websider.

1.1 Samarbeid

I tillegg til vår bacheloroppgave har USN gitt ut en bahceloroppgave til gruppe 16, og det er gitt to oppgaver til to masterstudenter.

Gruppe 16 består av fire maskiningeniører og to elektroingeniører. De skal lage moduler som gjør det enkelt å måle parametre som hastighet, kraft, lengde og andre vanlige størrelser som måles når man kjører fysiske eksperimenter.

Masterstudent som bruke en FPGA til å kommunisere med løsningen vår.

Masterstudent som skal designe en webside hvor fokuset skal være på brukervennlighet og analyse av brukeratferd.

1.2 Gruppemedlemmer

Christian Scott: Project Lead og Scrum Master

e-post: christian@kreativitet.no

Teknisk ansvarsområde: Core og Linux implementering

Prosjektleder:

- Ansvarlig for å lede prosjektet
  - Delegere oppgaver
  - Sette prioriteringer
  - Motivere teammedlemmene til å holde fokus på prosjektets mål og krav
- Ha oversikt over alle prosesser underveis i prosjektet
- Sikre at prosjektet er i rute til enhver tid

Scrum Master:

- Ansvarlig for å gjøre teamet selvorganisert
- Ansvarlig for at alle hindringer som utviklerteamet er/kan bli utsatt for blir fjernet
- Sikre at teamet følger Scrum rammeverket
- Hjelpe product owner med å vedlikeholde product backlogen

Anders Kristiansen: Testansvarlig

e-post: andersk96@live.no

Teknisk ansvarsområde: Utvikler UI

Testansvarlig:

- Ansvarlig for å identifisere og definere tester
- Ansvarlig for at tester blir utført på korrekt måte
- Sikre at teamet følger test strategier beskrevet i “test strategier” dokumentet

Espen Rønningen: Product Owner

e-post: espenrn@gmail.com

Teknisk ansvarsområde: Køsystem og Filhåndtering

Project Owner:

- Ansvarlig for at grooming av product backlog blir utført
- Sikre at product backlog er definert tydelig
- Definere godkjenningskriterier og verifiser at kriteriene er møtt
- Forstå interessentenes behov og prioriteringer godt nok til å fungere som interessentens “stemme” utad til resten av teamet
- Samarbeide med interessentene
- Samarbeide med development teamet

Jan Helge Slettebø: Dokumentansvarlig

e-post: janhelgeslettebo@gmail.com

Teknisk ansvarsområde: Protokoll og interface, Nettverk og Nettside

Dokumentansvarlig:

- Ansvarlig for teamets dokumentasjon
- Sikre at dokumentasjon er oppdatert og strukturert
- Sikre at teamet dokumenterer arbeidet sitt
- Sikre at teamets webside er oppdatert

1.3 Hva er remote lab

En remote lab gjør det mulig for studenter og undervisere å få tilgang til ekte laboratorie apparater over Internett.

Remote lab fungerer ved at en bruker logger seg inn på en nettside tilknyttet en remote lab. Brukere får ofte tildelt tid for å sette opp eksperimenter, kjøre de og vente på resultater. Brukeren kan velge mellom tilgjengelige eksperimenter og legger deretter inn sine parametre. Websiden sender brukerens input til det fysiske eksperimentet, som foretar nødvendige justeringer, og eksperimentet kjøres. Et resultat blir sendt fra eksperimentet tilbake til websiden og vist til brukeren.

1.3.1 Hva finnes av remote lab i dag?

USN sin remote lab er utdatert og i dag gjennomføres de fleste laber som fysiske eksperimenter på skolens område. Mange av gjennomføringene er begrenset til spesifikke dager i semesteret og det er ofte også et krav om å få godkjent gjennomføringen for å få gå opp til eksamen. Det er altså begrenset åpningstider og ofte går man strømlinjet gjennom eksperimentet en gang og da er man ferdig. Studenter lærer i forskjellige tempo og noen foretrekker repetisjon for å få med seg alle elementene.

Det er kostbart å kjøpe inn utstyr til eksperimentene, derfor kunne man tenke seg en løsning hvor flere skoler deler laboratorie utstyr.

Det finnes flere løsninger på hvordan man kan utføre et eksperiment. Fysisk lab (videre i kapittelet menes fysisk lab at man fysisk er tilstedet ved labben) er kun en av løsningene. Simuleringer, remote lab og pocket lab - for å nevne noen - er andre alternativer. Alle disse løsningene har sine styrker og svakheter.

Fordeler ved fysiske labber er muligheten studentene har til å være med på å koble opp systemer (der dette er mulig) samt erfaringen av å feilsøke når ting ikke går riktig. Ulempen med denne type lab er at studenter kan koble feil eller sette opp systemet feil, noe som kan resultere i ødelagt utstyr eller personskader.

Simuleringer utføres gjerne ved at man bruker et program til å simulere en situasjon/ eksperiment for å oppnå et tilnærmet korrekt resultat. Dette kan være en rimelig løsning sammenlignet med de andre alternativene fordi man trenger ikke fysisk utstyr til eksperimentet. Det vil være - i de fleste tilfeller- en utviklingskostnad. Utstyret man trenger for å utføre denne type eksperimenter har lang levetid og elever har ikke mulighet til å ødelegge eksperimentet ved å gjøre feil. Det vil heller ikke forekomme kostnader i forbindelse med slitasjedeler. En ulempe med denne type eksperiment er at studentene ikke får noen erfaring med oppkobling av eksperimentet, samt hvordan de skal løse eventuelle feil som måtte oppstå. Resultatene man får ved simulering vil kun være forenklinger, sammenlignet med fysiske eksperimenter.

Pocket lab [2]er et “laboratorie” man får plass til i lommen, derav navnet. Enheten inneholder sensorer som kan utføre et utvalg målinger som kan avleses på f.eks en mobil enhet. Dette gir mange fordeler ved at man kan ha den med seg overalt og utføre eksperimenter når og hvor man selv ønsker det. Målet med løsningen er at den skal være så liten og kompakt at man får den med seg og dette vil naturlig nok medføre ulemper. På grunn av pocket lab sin størrelse, så vil det være begrensninger for antall sensorer man får plass til og det samme gjelder lagring. Dette resulterer i at antall eksperimenter man kan utføre per pocket lab er begrenset og ønsker man å utføre flere eksperimenter, må man kjøpe en ny enhet som støtter eksperimentet man ønsker å utføre.

Remote lab er en fysisk lab med tilgang over internett og vil løse en del av problemene med tilgang til systemene. En styrke med denne løsningen er at studenter er fleksible

til å utføre eksperimenter når som helst og fra hvor som helst. Ved bruk av videofeed, vil man få en følelse av at man setter opp eksperimentet, men samtidig vil systemet gi beskjed dersom man gjør noe feil slik at man minimerer risikoen for at utstyr blir ødelagt. Det er i tillegg en fordel at man kan dele tilgang med andre instutisjoner slik at man kan dele på kostnadene (En remote lab løsning kan for eksempel fordeles på alle campus til USN).

En ulempe med denne løsningen vil være kostnadene når det gjelder slitasjedeler og forbruksmateriell, spesielt hvis man deler tilgang til systemet med andre skoler, firmaer eller lignende, slik at pågangen på eksperimentene blir stor. Det vil være utviklingskostnader i startfasen (utvikling av plattform, servere, fysiske eksperimenter som må lages og kobles til o.l).

1.4 Utfordringer ved å lage remote lab

Det er flere utfordringer ved bruk av remote lab. Det er viktig å sørge for at en gjennomkjøring har så like forutsetninger som mulig til et fysisk eksperiment. En remote lab og en simulering vil se likt ut for et menneske (begge vil se en video av en ball som ruller ned en bane og hopper), men resultatene vil være forskjellige ved gjentatte forsøk. Remote lab eksperimentet vil vise varierende resultater (på grunn av slitasje på banen, temperatur i rommet o.l), mens ved simuleringer vil man oppnå samme resultat hver gang.

Om du har et eksperiment med en ball som ruller ned en bane hvor det måles fart, skal ballen tilbake til start slik at nestemann kan kjøre sitt forsøk. Ved et fysisk eksperiment kunne vi løftet opp ballen og satt den i “startbåsen”. På en remote lab må disse åpenbare tingene som vi vanligvis ikke ville tenkt over, håndteres automatisk.

Når du er på skolelaben, og noe uforutsett skjer (en del slutter å virke), så er faglærer rett rundt hjørnet og hjelper deg. Det er ikke like lett på en remote lab, og derfor må utviklerne tenke på mulige scenarioer hvor ting kan gå galt og komme opp med løsninger som håndterer slike situasjoner.

På skolelaben så får man utdelt noe utstyr enten alene eller par/gruppevis. Dette er “ditt” frem til laben er ferdig. En fordel med dette er at man vet hvor mye utstyr man trenger til hver lab. Hvordan løser man dette ved remote lab? Hvor mye tid skal hver bruker få

på å gjennomføre sitt eksperiment? Hvor mange av det samme eksperimentet skal det eventuelt opprettes? Dette er spørsmål de som skal lage en remote lab må ta stilling til. Det er viktig å finne en balanse, hvor både brukerne er fornøyde og kostnadene holdes så lave som mulig.

1.5 Vårt system

I dette kapittelet gir vi innblikk i forskjellige scenarioer fra utvikler kommer med nytt eksperiment og kobler til, til en bruker logger inn og ønsker å kjøre dette eksperimenter helt frem til resultatet er kommet inn og brukeren logger av.

Utvikler logger seg inn på vår webside. Siden brukernavn og passord tilhører en utvikler i databasen vår, vil utvikler få opp en utviklerside som lar han/hun opprette eksperimenter. Utvikler lager ønsket eksperiment og trykker på knappen “Discover” for å opprette kontakt med serveren. Det sendes en konfigurasjonfil til serveren hvor den lagres. Eksperimentet er nå opprettet og klart til bruk.

Utvikleren kan skifte mellom utviklersiden og brukersiden etter ønske. Studenten har ikke denne muligheten og må forholde seg kun til brukerwebsiden.

Studenten logger inn på websiden. Studenten kan bruke en meny for å velge mellom hvilke eksperiment han/hun ønsker å kjøre. Når studenten har valgt et eksperiment vil han/hun få en unik ID som knytter studenten opp mot valgt eksperiment. Studenten kan legge inn ønskede parametere og trykke på knappen “kjør” for å legge parametere i en fil som sendes til serveren og lagres.

Filen(e) som opprettes av studenten(e) legges i en kø. Køen holder på disse filene fram til eksperimentet er ledig. Når eksperimentet er ledig sender køen den første filen som ble lagt til (den “eldste” filen), til eksperimentet. Eksperimentet gjør nødvendige justeringer basert på parameterene i filen fra studenten, og utfører eksperimentet. Når eksperimentet er ferdig sendes resultatet tilbake til brukeren som får dette vist på websiden.

2 Arbeidsmetode

Prosjektet er realisert ved hjelp av Scrum. Dette var ikke tilfelle i første halvdel av prosjektet, og informasjon om hvorfor prosjektmodellen ble endret og andre utfordringer er skissert i vedlegg A.

2.1 Roller

Scrum definerer 3 hovedroller:

- Product Owner
- Scrum Master
- Development Team

En Product Owners oppgave er å fungere som et bindeledd mellom interessenter, kunder og brukere på den ene siden, og utviklingsteamet på den andre. Product Owner må forstå behovene og prioriteringene til kunden og bruke denne informasjonen til å bestemme hva som skal lages, og i hvilken rekkefølge det skal lages.

Scrum Master har ansvar for Scrum prosessen; at den følges og at deltagerne har nok kunnskap til å bruke Scrum og dens prinsipper på en optimal måte.

Development team er alle de som er aktivt involvert i utvikling av produktet. Scrum legger vekt på at teamet skal være selvorganiserende og at grupper dannes etter de aktuelle oppgavene som til enhver tid skal løses.[3]

2.2 Prosessen

Scrum er en iterativ og inkremental prosjektmodell der arbeidet foregår som en rekke repeterende prosesser på vei mot et sluttmål. Sluttmålet kan være et ferdig produkt eller versjon av produktet. Alle utestående utfordringer som må løses for å nå sluttmålet, blir ført opp som en liste som bearbeides kontinuerlig ved at punkter som er ferdige strykes, og punkter som regnes som for store deles opp i mindre og mer håndterlige deler.

Det ideele målet for prosjektet er at listen blir tømt. Prosjektet kan allikevel regnes som

en suksess uten at dette målet blir nådd. Man kan, for eksempel, bestemme seg for at visse funksjoner skal utsettes til en senere versjon av produktet. Andre funksjoner kan ansees som ting vi ønsker å ha med selv om de ikke er absolutt nødvendige. Dermed kan målet for et Scrum prosjektet heller formuleres som: ”Alle de nødvendige funksjonene er gjennomført og fjernet fra listen”.

Listen som ble omtalt i de foregående avsnittene kalles en “Product Backlog” og punktene kalles “Product Backlog Items” eller “PBI”-er. De repeterende prosesser kalles “sprinter”. Disse har en varighet på alt fra 1 til 4 uker og har en fast struktur bestående av:

- Planlegging
- Arbeid
- Sprint Retrospective
- Sprint Review

Hensikten med den faste strukturen er å skape en rytme (engelsk: cadence) som bidrar til å øke følelsen av fremgang. I tillegg, vil gjentagelsen av strukturen bidra til at alle blir fortrolig med prosessen etter noen få sprinter slik at den kan utføres mer effektivt. Dette prosjektet har en sprintlengde på én uke.

2.3 Planlegging

Etter en initiell og overordnet planleggingsfase der hovedrammene for hele prosjektet blir fastsatt, skjer videre planlegging som en kombinasjon av faste møter og en kontinuerlig prosess som kalles “Grooming” (se neste avsnitt).

Hver dag samles teamet til en “Daily Scrum” der alle gir en kort statusoppdatering og sier litt om hva de skal jobbe med den dagen. Dette møtet skal være kort og overholde en streng tidsangivelse som f.eks 15 minutter. Hvis det er tydelig behov for en lengre diskusjon om en bestemt sak, avtaler de relevante partene et eget møte til dette.

I tillegg til Daily Scrum, starter hver sprint med et eget sprintplanleggingsmøte. I dette møtet konverteres de høyest prioriterte PBIer fra product backlog til oppgaver som skal gjennomføres i løpet av sprinten. Tidsbruken til hver oppgave blir estimert og lagt til

sprintplanen fram til summen av estimatene representerer en full sprint. Til slutt defineres det et overordnet mål og fokus for den aktuelle sprinten.

2.3.1 Grooming

"Grooming" betyr bokstavelig talt "pleie" eller "stell" av product backlog.

Et viktig prinsipp i Scrum er at en PBI ikke kan overføres en sprintplan før den er "Ready". Dette prosjektets definisjon av "ready" er at PBIen:

- kan gjennomføres i løpet av én sprint
- har en tydelig verdi for kunden
- er tilstrekkelig detaljert så teamet vet at den er gjennomførbar
- ikke er avhengig av andre PBIer eller forhold utenfor vår kontroll
- ligger innenfor teamets kvalifikasjoner og kapasitet
- er estimert og prioritert
- har tydelige og verifiserbare godkjenningskriterier
- har tydelige og verifiserbare ytelseskrav
- kan demonstreres under sprint review

Under grooming, jobber vi for at de høyest prioriterte PBIer skal passe til definisjonen av "Ready". Dette kan skje i forbindelse med de faste planleggingsmøtene, men også fortløpende etter hvert som nye detaljer oppdages. PBIer som er klare merkes med "R" i product backlog, og "U" ellers.

Estimering av PBIer skjer på to nivåer. Når PBIen er lite detaljert, brukes "T-skjorte størrelser" som S, M, L, XL osv.. Når det er tydeligere hva PBIen innebærer gis den et tall som indikerer antall "Story Points". "Story Points" er et relativt mål for størrelsen på PBIen. En PBI som krever lite arbeid kan, for eksempel, få tallet 1 mens en oppgave som krever dobbelt så mye arbeid kan få tallet 2.

Disse tallene kan bestemmes i en prosess som heter "Poker Planning" der alle i utviklingsteamet tilordner det tallet de mener er riktig til alle PBIene. Etter en periode på eksempelsvis 15 minutter, samles alle for å diskutere deres individuelle estimater. Dersom det

er enighet føres tallene inn i product backlog, ellers diskuteres punktet til man oppnår enighet.

For å gjøre prosessen enklere brukes ofte en modifisert utgave av Fibonacci-sekvensen til å begrense antall muligheter: 1, 2, 3, 5, 8, 13, 20, 40, 100, ∞ og '?'. ∞ betyr at PBIen er for stor til å kunne estimeres, mens '?' betyr at PBIen fortsatt er for uklar til å kunne estimeres ordentlig.

"Story points" kan brukes til å måle hastigheten (engelsk: velocity) til prosjektet, noe som ofte oppgis som story points per sprint. Dette tallet er nyttig i forbindelse med planlegging av videre sprinter.

2.4 Sprint Review og Retrospective

Etter hver sprint utføres en "Sprint Review" der arbeidet som har blitt gjort med produktet gjennomgås og demonstreres slik at PBIen kan fjernes fra product backlog. Videre blir selve sprinten merket som "Done", noe som betyr at:

- Design er gjennomgått
- Koden er komplett og:
  - Refaktorisert
  - I henhold til prosjektstandarden
  - Kommentert tydelig nok til at en utenforstående kan overta
  - Lastet opp til versjonskontrollsystemet
  - Gjennomgått av minst ett annet medlem av teamet
- Koden er testet:
  - Enhetstestet
  - Integrasjonstestet
  - Regresjonstestet
  - Platformtestet
- Ingen kjente feil

- Godkjent av oppdragsgiveren
- Kjører i produksjonsmiljøet
- Nødvendig dokumentasjon er på plass

Disse punktene ansees som det ideelle målet for sprinten, og det er ikke alltid praktisk eller mulig å forvente at alle punktene kan oppfylles ved hver sprint.

2.5 Prosjektets Product Backlog

Utviklingen av prosjektets product backlog kan sees i vedlegg. Den initielle versjonen vises her:

ID	Pri.	Vikt.	Str.	Stat.	Beskrivelse
5	1	A	S	U	Forbered 2. presentasjon
8	1	A	L	U	Implementer Scrum
1	1	A	XL	U	Skriv nødvendig dokumentasjon
2	-	A	XL	U	Som bruker vil jeg kjøre eksperimenter uavhengig av tid og sted
3	-	A	XL	U	Som eksperimentutvikler vil jeg integrere mine eksperimenter med systemet
4	-	A	L	U	Som administrator vil jeg overvåke og vedlikeholde systemet
6	-	A	M	U	Forbered 3. presentasjon
7	-	A	M	U	Avvikle prototypen og overføre til nye endelige prosjekter

Forklaring

ID En unik identifikator som tilordnes alle PBIer.

Prioritet Prioritet som justeres jevnlig i forbindelse med planlegging.

Viktighet Viktigheten av PBIen der A betyr at den skal implementeres, B betyr at den bør implementeres og funksjoner med C betyr at den ikke skal implmenteres i den nåværende versjonen av produktet.

Størrelse og Status Se avsnittet om 2.3.1

Konvensjoner Product Backlog Items (PBI) som begynner på “Som [aktør]...” er user stories. Disse beskriver spesifikke egenskaper og funksjonalitet som en rolle eller et eksternt system ønsker fra Remote Lab. Alle andre PBIer kalles “Technical Stories”. Disse er ikke direkte knyttet mot en aktør, men understøtter en eller flere “User stories”.

2.6 Kommunikasjon med oppdragsgiver

Kommunikasjon med oppdragsgiver har foregått via mail, eller ved møter. Oppdragsgiver har vært behjelpelig og kommet med tips og idéer underveis i prosessen.

2.7 Kommunikasjon med intern veileder

Vi har hatt et bra samarbeid med veileder fra dag en av prosjektet. Det meste av kommunikasjon med veileder har foregått via mail eller møter, men vi har og hatt muligheten til å komme innom kontoret for en prat. Veileder har vært tilgjengelig og engasjert i prosjektet, og guidet oss bra med gode tilbakemeldinger på både det som er bra og dårlig.

2.8 Risiko

Det er viktig å avdekke risiko tidlig i prosjektfasen for å avdekke potensielle farer som vil kunne påvirke prosjektets fremgang. Risikoanalysen gir også en oversikt over sannsynligheten for at risikoen inntreffer, samt hvilken grad av konsekvens det vil ha dersom risikoen oppstår.

2.8.1 Håndtering og kontrollering av risiko

Det er tre måter å håndtere risiko på[4]:

- Akseptere: En bevisst beslutning på å engasjere seg i en aktivitet, da fordelen ved gjennomføring er større enn den potensielle risikoen.
  - Redusere: Jobbe aktivt med å redusere den aksepterte risikoen

  - Reserveplan: Jobbe med å ha en reserveplan dersom risikoen blir en realitet.

- Unngå: En bevisst beslutning på å ikke gjennomføre en bestemt aktivitet som forårsaker risikoen. Dette fordi den potensielle risikoen er større enn fordelen ved gjennomføringen.
- Overføre: Overfør risikoen til en tredjepart, enten gjennom f.eks forsikringsselskap eller annen form for kontrakt som overfører risikoen.

Risiko må kontrolleres og håndteres fortløpende. Vi har allerede lagt planer for hvordan vi håndterer en risiko, dersom den faktisk oppstår er det viktig å iverksette denne planen. Dersom prosjektet utvides må vi avdekke om det her kan oppstå ny risiko som det må tas hensyn til.

2.8.2 Risiko matrise

Vi bruker en risikomatrise for å regne ut risikoen basert på et samlet produkt av sannsynlighet multiplisert med konsekvens. Ved å gjennomføre tiltak vil vi ha mulighet til å redusere sannsynlighet og konsekvens. Matrisen sees på figur 1.

Sannsynlighet	Konsekvens				
	Ubetydelig	Liten	Moderat	Betydelige	Alvorlig
Veldig sannsynlig	5	10	15	20	25
Mer sannsynlig	4	8	12	16	20
Sannsynlig	3	6	9	12	15
Mindre sannsynlig	2	4	6	8	10
Usannsynlig	1	2	3	4	5

Figur 1: Risiko matrise

2.8.3 Risiko eksempler

Samarbeid med gruppe 16					
Risk ID	Sannsynlighet	Konsekvens	Risiko	Redusert	Strategi
2	3	3	5.4	40%	Akseptere
Kommentar: Forskjell i tankesett og fagfelt. Lite overlappende kunnskap					
Tiltak: Lage vår egen demo som kan brukes til presentasjonen					

Korttidsfravær					
Risk ID	Sannsynlighet	Konsekvens	Risiko	Redusert	Strategi
3	3	2	6	0%	Akseptere
Kommentar: Sykdom, eksamen, eller andre familiære årsaker som gjør at en er vekk en dag eller to.					
Tiltak: Sørge for å ha god kommunikasjon dersom en har behov for fravær.					

2.9 Tester

Testing er en metode for å finne og dokumentere feil eller problemer i et produkt og å verifisere at kravene til produktet er oppfylt [5]. Når man oppdager feil kan man korrigere dem og øke kvaliteten til produktet [6]. Hvis man begynner å teste tidlig i produktutviklingen, kan man oppdage problemer i design og feil i implementasjon på et stadie hvor det koster mindre tid og penger å rette opp enn hvis man kun tester det ferdige produktet[5].

Vi bruker fem typer tester til å teste softwaren vår: unit test, integrasjonstest, godkjenningstest, systemtest og regresjonstest.

2.9.1 Unit testing

Unit test er en test som utføres på en enhet med kode for å sjekke at den produserer riktig resultat [6]. Enheten utfører én funksjon, for eksempel, fyller et array med elementer eller utfører en utregning. Unit testing foregår fortløpende når man har laget ferdig en funksjon [5] for å dokumentere at den fungerer som den skal.

Når man unit tester kan man isolere funksjonen, dvs. å finne avhengigheter funksjonen har, både eksterne og interne, og erstatte dem med dummy kode eller statiske data [6]. Hvis testen feiler blir det tydelig at det er funksjonen under test som ikke fungerer, og ikke en eller flere av avhengighetene dens [7]. Å isolere funksjonen kan også gjøre at man oppdager unødvendige avhengigheter [7].

Eksterne avhengigheter er avhengigheter fra andre subsystemer, for eksempel filer på en filserver eller meldinger funksjonen skal ta imot over nettverk.

Interne avhengigheter er avhengigheter funksjonen har til andre funksjoner i klassen/subsystemet, for eksempel at funksjonen bruker verdien returnert fra en annen funksjon.

2.9.2 Integrasjonstesting

Integrasjonstest er en test som utføres på funksjoner som har blitt integrert til en modul. Funksjonene blir unit testet, så satt sammen til en modul og testet på nytt for å sjekke at funksjonene fungerer sammen [6]. Fokuset ved integrasjonstesting er å teste samhandlingen mellom funksjonene [6].

Integrasjon og integrasjonstesting skjer vanligvis i inkrementer, noen funksjoner blir integrert og testet om gangen, i stedet for alle på en gang, fordi dette gjør det lettere å identifisere mellom hvilke moduler eventuelle problemer oppstår [6].

2.9.3 Godkjenningstest

Godkjenningstest er en test som tester et godkjenningskriterie tilhørende en user story. Godkjenningstester brukes til å teste om kriteriene er godkjent og om user storien er fullført.

2.9.4 Systemtesting

Systemtesting tester funksjonen til et helt system. Systemet bør testes i miljøet det skal brukes for å øke sjansen for å finne miljøspesifikke feil. Systemtesting tester at systemet oppfyller kravene til systemet [6].

2.9.5 Regresjonstesting

Når kode blir refaktorert, endret fordi ny funksjonalitet krever tilpassing av funksjonen, eller avhengigheter endrer seg, må testene utført på funksjonen utføres på nytt. Dette gjøres for å sjekke at funksjonen fortsatt gir samme resultat, og at endringene ikke har laget nye feil eller brakt tilbake eldre feil [5].

2.9.6 Når og hvordan utføre test

Unit test Hver gang man lager en funksjon ferdig og tror den fungerer korrekt, utfør en unit test:

- Definer verdier for input variabler.
- Definer forventede verdier for resultat variabler eller returnverdi.
- Isoler funksjonen; finn avhengigheter, eksterne og interne, og erstatt dem med dummy kode eller statiske data.
- Kjør funksjonen med input verdiene og noter resultatene.

Kjør gjerne flere tester med forskjellige input verdier. Man bør teste verdiene man forventer at funksjonen kan få, spesielt ekstremverdier. Man bør også teste utenfor de forventede verdiene, for å teste hvordan funksjonen håndterer feil.

En funksjon er en del av koden som gjør én ting, for eksempel fyller et array eller utfører en utregning. Funksjonsnavnet burde være selvbeskrivende. Hvis det ikke er det, legg til en kommentar med en beskrivelse av funksjonen.

Integrasjonstest Funksjoner som har avhengigheter, testes med avhengighetene i stedet for dummydata. Funksjoner kobles sammen og blir testet sammen, og man definerer

en ny test som tester samhandlingen mellom dem. Integrasjonstest kan gjøres steg for steg (legge til en funksjon, teste, legge til en funksjon til, osv.) eller alle på en gang.

Når nok funksjoner har blitt koblet sammen, utfører de til sammen en funksjon som et godkjenningskravet til en PBI krever. Integrasjonstesten blir da en godkjenningstest for PBIen.

Regresjonstest Når kode blir endret, må unit testene utført på funksjonen utføres og godkjennes på nytt for å sørge for at den fortsatt fungerer som den skal. Hvis en funksjon blir byttet ut med en ny funksjon må alle integrasjonstester som inneholder den gamle funksjonen, utføres på nytt.

2.10 Verktøy

I dette delkapittelet går vi gjennom verktøyene vi har benyttet oss av under prosjektet.

2.10.1 Google Drive

Google drive er en fillagring og synkroniserings tjeneste. Google drive er tilgjengelig på alle plattformer, og vi er ikke avhengige av å sitte på en lokal stasjon for å jobbe.

Vi har delt mappestrukturen opp i to deler:

- Prosjektstyring
- Dokumentasjon

Som figur **??** viser er dokumentasjon igjen delt opp i nye mapper som inneholder materiale til hvert underpunkt som det jobbes med.

Under prosjektstyring har vi undermapper som tar for seg de forskjellige delene av prosjektmodellen vår. Ved et prosjekt av denne størrelsen er god mappestruktur viktig.

2.10.2 Git

Git er et versjonskontrollsystem som holder styr på forandringer i en fil, eller et sett med filer. Git gir brukeren enkel tilgang til å finne tilbake til spesifikke versjoner ved behov, enten det er endringer i en fil, eller hele prosjektet.

Vi valgte å bruke Git da det også gjør det lettere å samarbeide om den samme koden. Vi vurderte ingen andre versjonskontrollsystemer. Alternativet til å bruke versjonskontrollsystem var å sitte med alt lokalt, og lage nye mapper med versjonsnavn som ble delt rundt til nestemann som skulle integrere sin del. Totalt sett ville dette blitt rotete og tidkrevende.

2.10.3 Stilguides

Vi ble tidlig enig om å lage et eget dokument for stilguides som var gjeldende for prosjektet. Når vi er et team, er det bra at alle følger den samme strukturen på utforming av dokumenter, eller kommentering av kode.

Dokumentet finnes i appendix H.

2.10.4 LaTex

LaTex er et formaterings program for ulike typer dokumentproduksjon, og det er velegnet til mellomstore, tekniske eller vitenskapelige dokumenter som, for eksempel, en bachelor eller masteroppgave.

Vi mener LaTex gir oss det beste resultatet for sluttdokumentet og har dermed valgt å overføre dokumentasjonen fra Google Drive inn i LaTex format før innlevering. Vi har fulgt alle krav til dokumentutforming som nevnt i stilguiden, og har laget maler i LaTex som gjør at vi kan overføre direkte uten mye ekstra arbeid.

3 Teknologi

3.1 Webgrensesnittspråk - Server

Webgrensesnittet trenger en server som mottar forespørsler, håndterer dem og sender websider tilbake. Til dette trenger vi et scriptspråk og en server. Vi har sett på to alternativer til dette, Node.js og PHP. Vi har valgt Node.js.

3.1.1 Node.js

Node.js er et språk som blir tolket som Javascript av Googles V8 Javascript motor. Det er et lettvektig og effektivt språk fordi det bruker en asynkron, event-basert, ikke-blokkerende i/o modell [8], som gjør at det er bra til å lage skalerbare webapplikasjoner [9].

At Node.js er asynkron vil si at når den får oppgaver, blir de ikke utført med en gang, men lagt til i en liste over oppgaver som skal utføres. Oppgavene blir utført etterhvert og ingen rekkefølge er garantert. Mellom oppgavene kan Node.js gjøre andre ting, som å håndtere en innkommende forespørsel eller ta imot andre eventer.

Med event-basert menes at Node.js lytter etter spesielle hendelser, og setter igang utførelse av funksjoner når hendelsen inntreffer. For eksempel kan Node.js brukes til å lytte på en nettverksport etter HTTP forespørsler eller at en i/o operasjon er ferdig og har returnert data.

Ikke-blokkerende i/o betyr at Node.js kan gjøre andre ting mens den venter på at i/o operasjoner skal bli ferdig.

Node.js bruker Javascript. Javascript er et relativt nytt programmeringsspråk og drar nytte av forbedringer i design i forhold til eldre serverspråk. Fordi Node.js er asynkron og event-basert, fungerer det godt til sanntids webapplikasjoner. [10]

Javascript kjører på en tråd. Det vil si at den ikke kan gjør flere oppgaver samtidig og dra nytte av multicore prosessorer. Til gjengjeld får den mindre overhead ved å bare kjøre en tråd.

Node.js håndterer lange oppgaver dårlig. Siden den bare kjører på en tråd, får den ikke gjort noe annet mens den holder på med oppgaven. Alle eventer som kommer i mellom-

tiden håndteres etter at oppgaven er ferdig, før neste oppgave starter.

Node.js har en HTTP modul som kan brukes til å sette oppe en webserver som lytter på en nettverksport og tar imot HTTP forespørsler. Hvordan man håndterer forespørslene må man lage selv.

Express.js Express er et minimalistisk og upåståelig webframework til Node.js som håndterer HTTP forespørsler.

Upåståelig betyr at den ikke påstår at det er en "riktig" måte å løse en spesifikk oppgave på, men lar deg velge hvordan problemstillingen skal løses [11]. Dette gjør at Express blir minimalistisk, siden den ikke inneholder funksjoner selv for de mest brukte oppgavene, som å lese dataene som kommer med i en HTTP forespørsel. Express lar deg definere hvordan dette skal håndteres. Andre har laget moduler og bibliotek som håndterer problemer på en spesifikk måte, og du kan velge hvilken du vil bruke eller lage din egen.

Med Express kan man konfigurere en kjede med middleware funksjoner som håndterer forskjellige deler av forespørselen. Middelware kan for eksempel være body-parser som henter dataen sendt med i en HTTP forespørsel og gjør den tilgjengelig for resten av håndteringskjeden.

Express håndterer ruter og HTTP metoder, og sørger for at riktig funksjon blir kalt til å håndtere forespørselen [12].

Express støtter bruk av view template motorer, som gjør at det er enkelt å skille presentasjon av data og håndtering av data [12]. En view template motor er en generator som bruker templates til å generere en visning av dataen den ble gitt.

PUG PUG er en høy ytelse view template motor for Node.js. PUG bruker templates skrevet i et innrykksensitivt språk som blir kompilert til en Javascript funksjon. Denne funksjonen fyller inn verdier i variabler definert i templaten basert på dataen den ble tilsendt, og returnerer en streng med HTML5. [13]

PUG kan brukes til å generere dynamiske HTML-sider basert på dataen den får. Den kan generere lister med forskjellig lengde basert på hvor mange dataelementer den fikk, den kan bruke samme template til å generer to HTML-sider med forskjellig tittel og annet

innhold, og den kan bruke en variabel til å bestemme om den skal utelate eller inkludere en del av siden.

PUG kan også brukes til å initialisere variabler til script som kjører hos klienten.

Valg Vi har valgt å bruke Node.js fordi:

- Node.js er laget for å være skalerbart og det er tenkt at systemt skal kunne brukes av mange personer samtidig.
- Webserveren skal håndtere mange små forespørsler.
- Vi trenger ikke velge webserver ved siden av, siden Node.js kan bruke HTTP modulen til å lage en webserver.
- Vi kan bruke samme språk på server og klient; da slipper vi å bytte fra tankemåten i et språk til tankemåten i et annet.
- Node.js har innebygd støtte til JSON.

Vi har valgt å bruke Express sammen med Node.js fordi:

- Express er det mest brukte frameworket med Node.js.
- Express er upåståelig: Vi får velge selv hvordan vi løser oppgaver og det finnes mange biblioteker som løser de oppgavene for oss som vi kan velge i.
- Den støtter view template motorer, som gjør det enkelt å følge Model-View-Controller design pattern, hvor man skiller data, visning, og arbeid på data.
- Express håndterer HTTP-håndteringskjeden for oss.
- Vi har ikke sett på andre alternativer til Express.

Vi har valgt å bruke PUG sammen med Express og Node.js fordi

- PUG er view template motoren Express har som standard.
- Det er enkelt å generere dynamiske websider.
- Har ikke kikket på andre alternativer.

3.1.2 PHP

PHP er et scriptspråk som er spesielt tilpasset webutvikling. PHP kode kjører på serveren og generer HTML-sider som blir sendt til klienten. [14] Med PHP som scriptspråk må man sette opp en webserver ved siden av som håndterer forespørsler og kaller PHP scriptene.

En server vi kunne brukt er Apache HTTP server. Den er bedre en Node.js på store oppgaver, fordi den bruker en tråd per forespørsel. På en annen side får den mer overhead for å holde styr på alle trådene. Trådene står ofte å venter på i/o operasjoner eller respons fra database, og kan ikke gjøre noe annet mens de venter. Derfor er Apache dårlig til å håndtere mange forespørsler som må vente på data fra eksterne kilder.

Det finnes frameworks som hjelper til å følge Model-View-Controller design pattern i PHP, for eksempel Zend 3. PHP har utvidelser for å gjøre den eventbasert og gi den støtte for JSON.

Valg Det ville ikke gjort store forskjeller å velge PHP over Node.js. Vi har ikke valgt bort PHP av noen spesielle grunner. Både Node.js og PHP klarer de samme operasjonene ved å bruke utvidelser og bibliotek, med tilnærmet lik ytelse.

3.2 Les og skrive filer i Node.js

Webserveren skal bruke informasjon lagret i filer og skrive data til filer. Webserveren skal lese konfigurasjonsfilene til eksperimentene, skrive filer med input mottat fra brukeren, og lese filer med resultater fra en kjøring av eksperimentet.

Til Node.js finnes en modul kalt File System. Den har funksjoner for å lese og skrive filer asynkront og fungerer på flere operativ systemer, inkludert Linux, Windows og Mac. [15]

Valg Vi har valgt å bruke File System modulen til Node.js fordi vi ikke har funnet noen alternativer.

3.3 Filwatching i Node.js

Webserveren skal oppdage at resultatfiler blir laget. Den trenger en funksjon for å lytte etter filer i filsystemet. Til dette har vi sett på flere alternative Node.js moduler - Chokidar, File System, Filewatcher, og Node Sentinel File Watcher - og valgt Chokidar.

3.3.1 Chokidar

Chokidar er en wrapper for File System sin filwatching funksjon, som løser flere av problemene den har. Den har mer presise eventer den kan sende, som beskriver bedre hva endringen på filen er. Den sørger for at en endring bare blir rapportert en gang, og rapporterer alltid filnavnet til filen som ble endret.[16]

Valg Vi har valgt å bruke Chokidar til å lytte etter filer i Node.js. Vi har funnet flere alternativer som gjør det vi trenger, men Chokidar var best dokumentert. I tillegg er det mange som stiller spørsmål om chokidar [17], som tyder på at den er mye brukt.

3.3.2 File System

File System er en modul i Node.js som de fleste bruker når webserveren skal interagere med filsystemet. Den har to funksjoner for filwatching. Dokumentasjonen til File System anbefaler å ikke bruke watchFile funksjonen [18].

Den andre funksjonen heter watch. Den sender eventer når den oppdager endringer i filen eller mappen den lytter på, som Node.js legger til i listen for oppgaver den skal utføre. Når den lytter på mapper rapporterer den ikke alltid filnavnet til filen som ble endret og den er ikke konsekvent på alle platformer. Ofte rapporterer den flere endringer når det bare var en. [19]

Valg Vi har valgt å ikke bruke File System sine funksjoner for filwatching på grunn av problemene de har. Vi har sett etter andre alternativer.

3.3.3 Filewatcher

Filewatcher er en wrapper for File System sin filwatching funksjon, som løser flere av problemene med den. Filewatcher rapporterer alltid filnavnet, sender bare eventer en gang per endring, og fungerer på alle operativsystemer [20].

Valg Filewatcher er dårlig dokumentert. Vi finner ikke en oversikt over hvilke eventer man kan lytte på. Det er få som har stilt spørsmål om Filewatcher, som tyder på at få bruker den. Derfor har vi valgt å ikke bruke Filewatcher.

3.3.4 Node Sentinel File Watcher

Node Sentinel File Watcher er en Node.js modul for filwatching skrevet i C++, som fungerer på Linux, Mac og Windows. Den kjører på en egen tråd, så den har liten effekt på ytelsen til Javascript applikasjoner [21].

Valg Hva funksjonene til NSFW (Node Sentinel File Watcher) gjør og hvordan den fungerer er dårlig dokumentert. Det er få som har stilt spørsmål om NSFW [22], som tyder på at få bruker den. Derfor har vi valgt å ikke bruke Node Sentinel File Watcher.

3.4 Real-time Server-klient kommunikasjon

Eksperimentgresesnittet og webserveren skal sende data til hverandre. Vanligvis kan ikke en HTML-side motta ny informasjon fra webserveren etter at den er sendt. Å sende data fra eksperimentgrensesnittet krever vanligvis at brukeren sender en HTTP POST eller GET forespørsel; da blir websiden lastet inn på nytt.

Vi vil at brukeren ikke skal laste inn siden på nytt når han sender data, og vi vil at eksperimentgrensesnittet skal kunne ta imot data etter at det er sendt til brukeren. Vi har sett på to alternativer, Socket.io og AJAX, og valgt Socket.io.

3.4.1 Socket.io

Socket.io er en Node.js modul som støtter event-basert toveiskommunikasjon mellom server og klient ved bruk av WebSocket. Websocket er en kommunikasjonsprotokoll som gir toveiskommunikasjon over en TCP tilkobling.

At socket.io støtter event-basert kommunikasjon, betyr at man kan definere sine egne type meldinger. På webserveren og i HTML-siden som sendes til brukeren, setter man opp en funksjon som håndterer meldinger som blir motatt for hver type melding man har definert. Man kan også sende meldinger fra webserveren og fra brukeren, ved å oppgi hvilken meldingstype og dataen man vil sende. Man kan sammenligne meldingstyper med ruten i en HTTP forespørsel.

Ved hjelp av Javascript på HTML-siden kan, for eksempel, data fra et skjema leses ut og sendes til webserveren med meldingstypen “form-data”, som webserveren tar imot og kaller funksjonen som håndterer “form-data” meldinger. Urelatert til dette kan webserveren sende en melding med typen “ten-minutes” som klienten tar imot og kaller funksjonen som håndterer “ten-minutes” meldinger ved å lage en popup som sier at brukeren har vært på websiden i ti minutter.

Valg Vi har valgt å bruke Socket.io til å opprette toveiskommunikasjon mellom server og klient fordi WebSocket vil skape mindre last på webserveren enn AJAX.

3.4.2 AJAX

AJAX står for asynchronous Javascript and XML [23]. AJAX er en gruppe med teknologier som sammen brukes til å sende HTTP forespørsler, motta data og legge til dataen på websiden uten å laste inn websiden på nytt [24].

AJAX bruker HTML og CSS til å presentere data, Javascript for å endrer dataen på websiden, XMLHttpRequest objektet i Javascript til å asynkront sende en HTTP forespørsel til og motta en respons med data fra webserver, og XML som format på dataen som blir sendt [24]. I dag brukes ofte JSON i stedet for XML [23].

For å motta data fra serveren må klienten sende en HTTP forespørsel, som vil si at for å få data fra webserveren når ny data er tilgjengelig, må klienten be om dataen hele tiden

(polling), og serveren må enten svare at ingen ny data er tilgjengelig, eller ikke svare før ny data er tilgjengelig (long polling). Serveren kan ikke sende data til klienten uten at klienten har bedt om det.

Valg Vi har valgt å ikke bruke AJAX fordi den bruker en request-response kommunikasjon, som skaper større last på webserveren enn WebSocket, fordi webserveren må håndtere mange hyppige forespørsler fra samme bruker, for alle brukere. Hvis webserveren skal sende ny data til brukeren når den nye dataen kommer, må klienten sende forespørsler om dataen gjentagende frem til dataen er tilgjengelig for å få dataen.

3.5 Format på eksperimentfiler

Webserveren skal lage og lese filer med informasjon som skal brukes av andre subsystemer. Vi må bestemme et filformat og et dataformat til disse filene. Her har vi sett på to alternativer til filformatet, JSON og CSV, og valgt JSON.

3.5.1 JSON

JSON står for JavaScript Object Notation, som vil si at det er en måte å skrive Javascript objekter som tekst. Objektene er skrevet som navn-verdi par, som gjør at JSON er lett for mennesker å lese. I tillegg til objekter kan man skrive tall, tekst, boolean og arrayer som JSON. JSON brukes ofte som formatet på data som blir overført mellom en webserver og en klient. [25]

For å bruke JSON i kode må man ha en JSON parser som kan lese JSON og gjøre det om til objekter, og skrive objekter som tekst. JSON kan brukes med andre språk enn Javascript, men i Javascript trenger man ikke et bibliotek for å parse eller lage JSON, siden dette er innebygd.

Valg Vi har valgt å bruke JSON fordi det er lesbart for mennesker, og det kan inneholde komplekse datatyper som objekter og arrayer. Node.js skrives som Javascript, som har innebygd støtte for parsing og laging av JSON.

3.5.2 CSV

En “Comma Separated Values” fil er en tekstfil med en komma-separert liste med verdier. Hver linje er en post, med samme antall verdier i hver post. Den første linjen i en CSV fil inneholder ofte navnet på kolonnen. [26]

Valg Vi har valgt å ikke bruke CSV fordi vi vil kunne bruke komplekse datatyper som objekter og arrayer, og det blir vanskeligere med CSV enn med JSON.

3.6 Express validator

Express validator er en middleware for express.js som brukes til å validere at brukerinput har rett format, før vi prosesserer data. Å validere data er en sikkerhet som forhindrer at applikasjonen prøver å behandle ukjent data, og risikerer å kræsje.

Vi valgte å bruke express validator fordi den er godt dokumentert[27], og den dekker de behov vi har for validering.

Vi bruker validering som en sikkerhet, men også for å veilede brukeren til å fylle inn riktig data, gjennom å bruke express validator sammen med flash og express message kan vi gi tilbakemeldinger til brukeren dersom det er fylt inn feil data.

3.7 Connect flash

Connect flash er en buffer i brukersession brukt til lagring av meldinger. Meldingene blir skrevet til flash og etter at den er vist til brukeren blir flash tømt. [28]

Vi bruker connect flash sammen med express messages og det er mest vanlig at dette brukes i samhandling med en redirect hvor brukeren enten får et varsel eller en success melding.

3.8 Express messages

Express messages er brukt for å rendre flash varslinger. Vi ønsker å bruke varslinger mot brukeren ved, for eksempel, feil innlogging, feil eller mangler ved registrering og eventu-

elt andre steder hvor brukeren trenger en flash melding fra systemet. [29]

3.9 Passport

Passport er en autentiserings middleware for Node. Dens eneste oppgave er å autentisere forespørsler. I dagens webapplikasjoner kan autentisering skje på mange forskjellige måter. Passport håndterer den tradisjonelle login metoden hvor brukeren sender et brukernavn og et passord, og den støtter OAuth slik at det er mulig å bruke login gjennom Facebook, Google eller Twitter. [30]

Alle mulighetene passport gir oss, gjør den til et naturlig valg for autorisering av brukere.

Den er delt opp i moduler, og dermed trenger vi bare installere akkurat den eller de modulene vi velger for autentisering, og slik slipper vi masse ekstra pakker som aldri blir brukt. [31]

3.10 Bcryptsjs

Bcryptsjs brukes til å beskytte brukerens passord. Den hasher passordet for ekstra sikkerhet, og må brukes for å sammenligne passord ved innlogging. [32]

3.11 MongoDB

MongoDB er en open-source dokumentdatabase med automatisk skalering. Den er en NoSQL database, og bruker JSON format med skjema. Verdifeltene kan inneholde dokumenter, matriser eller matriser av dokumenter[33].

Vi bruker MongoDB som database for brukere. For å kjøre et eksperiment må brukeren være innlogget. Alle resultater en bruker har fått, lagres på den brukeren, slik at vi kan generere en historikk for hver enkelt bruker der bruker kan se på sine tidligere resultat og laste ned resultatfilen.

Vi så ikke på andre alternativer.

3.12 Cookie Session

Vi bruker cookies for å gjøre det enklere å håndtere brukerpreferanser. Cookie session lagrer cookien på klienten. Cookien varer i 24 timer. [34]

3.13 Webgrensesnittspråk - Klient

Vi skal ha dynamisk funksjonalitet på websidene, som å styre grafer og motta data fra serveren uten å oppdatere siden. Dette bruker vi Javascript til. Websidene bruker HTML5 til å presentere data, som er stylet med CSS3 og Bootstrap 4.

3.13.1 Javascript

Javascript er et lettvektig, tolket, objekt-orientert språk [35]. Det er en implementasjon av ECMAscript-262 standarden til Ecma International [36]

Javascript er et scriptspråk som kan kjøre i en nettleser, og brukes til å gi funksjonalitet til HTML-sider hos klienten. Den kan endre innholdet i HTML elementer, fjerne og legge til HTML elementer, sende forespørsler til en server og motta ny data uten å laste inn siden på nytt, og reagere på brukerens handlinger på siden, som for eksempel at brukeren trykker på en knapp eller at musepekeren forlater vinduet. [37].

Valg Siden Javascript er scriptspråket de aller fleste bruker, har vi også valgt å bruke det. Vi har valgt å bruke Node.js til webserveren, som skrives i Javascript. Da slipper vi å bytte fra tankemåten i et språk til tankemåten i et annet. Det finnes mange bibliotek som andre har laget til Javascript, som vi kan bruke til å løse kompliserte oppgaver. Vi har ikke sett på noen alternativer til Javascript.

3.13.2 HTML5

HTML5 er internettets markup språk [38]. Det er brukt til å lage webdokumenter, og beskriver hva innholdet er; for eksempel liste data, paragrafer, headinger, videoer, osv.

Valg Vi har valgt å bruke HTML5 til å lage websidene våre, fordi det er den nyeste versjonen av HTML standarden og alle nettlesere kan lese HTML og gjøre det om til et grensesnitt som brukeren får se. Vi kan HTML fra før, og det er det alle andre websider er bygget opp av. Vi har ikke sett på noen alternativer til HTML.

3.13.3 CSS3

CSS3 er et språk som beskriver hvordan et dokument skal presenteres. CSS er en mekanisme for å style webdokumenter, med fonts, farger, plassering, osv. [39] og fungerer godt med HTML.

Bootstrap 4 Bootstrap er et framework som gjør det enklere å bygge utformingen til websider. Det inkluderer HTML- og CSS-basert design templates for skjemaer, knapper, tabeller, navigasjonsbarer, bildekaruseller, grid-layout og mange andre, og også Javascript plugins som gir funksjonalitet til elementene på websiden. Bootstrap gir mulighet for å lage webdesign som automatisk justerer seg slik at de ser bra ut på flere skjermstørrelser. [40]

Valg Vi har valgt å bruke Bootstrap 4 til å bestemme utseende på websidene fordi den inneholder mange CSS-klasser som vi kan bruke, så vi slipper å lage dem selv. Vi bruker Bootstrap til å legge til funksjonalitet på de visuelle elementene i websidene, for eksempel, å lage en meny som åpner seg når man trykker på den og lukker seg når man trykker utenfor menyen. Vi har brukt Bootstrap 4 før.

Vi har valgt å bruke CSS3 til å gjøre endringer i CSS-klassene Bootstrap 4 har, der de ikke gjør slik som vi vil. Alle nettlesere kan lese CSS3 og vi kan CSS3 fra før. Vi har ikke sett på noen alternativer til CSS3.

3.14 Graf

Eksperimentgresesnittet har som krav at den skal kunne presentere resultater som grafer. Den skal la grafer inneholde flere kurver samtidig, la brukeren endre hvilke kurver som

vises, la brukeren tilpasse aksene, og vise verdien til kurvene når brukeren holder musepekeren over. Vi har også krav om å presentere resultater som kakediagrammer.

Vi har funnet flere Javascript bibliotek som lar oss gjøre dette - Plotly.js, D3.js og Google Charts - og valgt Plotly.js.

3.14.1 Plotly.js

Plotly.js er et Javascript bibliotek for å lage interaktive diagrammer. Den er bygget oppå D3.js [41]. Den støtter mange diagramtyper, inkludert linje diagrammer og kakediagrammer.

Grafene kan ha flere kurver. Brukeren kan gjemme/vise dem, zoome, flytte og resette visningsområdet. Visningsområdet vil automatisk tilpasse seg kurvene når grafen blir laget og når man gjemmer/viser en kurve. Når brukeren flytter musepekeren over en kurve vises verdien til kurven i det punktet. Man kan også laste ned grafen som et bilde i PNG format.

Valg Vi har valgt å bruke Plotly.js fordi den har innebygde funksjoner som dekker kravene vi har til grafer.

3.14.2 D3.js

Data-driven document er et Javascript bibliotek for å visualisere data ved å bruke web standarder. D3.js lar deg binde data til en dokumentmodell og utføre transformasjoner på den. [42]

Støtte for grafer er ikke innebygd, men ved å bruke D3.js sine funksjoner kan man bygge en graf. D3.js har også funksjoner som vil gjøre det mulig å legge til dynamisk funksjonalitet til grafene. [43]

Valg Vi har valgt å ikke bruke D3.js fordi det er et bibliotek for å lage visuel data, ikke spesifikt grafer, som vil si at vi må gjøre mye av jobben selv. Siden det finnes biblioteker som har grafer innebygd, velger vi heller en av de.

3.14.3 Google Charts

Google Charts er et Javascript bibliotek for å lage diagrammer fra data. Den støtter mange diagramtyper, inkludert linje diagrammer og kakediagrammer [44].

Google Charts støtter å ha flere kurver i samme graf, og å vise verdien til kurven der brukeren holder musepekeren. Den kan ikke gjemme kurver, men kan utheve en kurve om gangen og dempe de andre. Brukeren kan ikke endre visningsområdet på noen måte.

Valg Vi har valgt å ikke bruke Google charts fordi den ikke støtter all funksjonaliteten grafene våre må ha.

3.15 Bower

Bower er en packet manager for web, som holder styr på pakkene vi ønsker å bruke. Bower håndterer komponenter som inneholder HTML, CSS og Javascript. [45]

Vi bruker pug templates til vår frontend og bower til å håndtere bootstrap 4 samt jQuery.

Det var og mulig å bruke en npm pakke som heter pug-bootstrap, men denne er relativt utdatert og derfor valgte vi å gå for bower hvor vi fikk tilgang til bootstrap 4[46].

3.16 Font awesome

Font awesome er en ikon verktøykasse basert på CSS. Vi bruker font awesome for våre ikoner på nettsiden. [47]

4 User Stories og Krav

Vi har utarbeidet flere krav til systemet fra samtaler med oppdragsgiver. Kravene definerer funksjoner og egenskaper som vi og oppdragsgiver er enig om at systemet skal ha for at oppdragsgiver skal godkjenne systemet. Vi har også fått krav fra gruppene vi samarbeider med, om hva de trenger fra vårt system for at det skal fungere med deres systemer.

User stories er en beskrivelse av hva som skal implementeres, som omhandler en liten nok del av systemet til at vi vet hvor vanskelig eller lang tid det vil ta å implementere. User storiene omhandler først store features, og blir splittet til flere mindre user stories når vi synes de er for store til å jobbe med. Det betyr at de user storiene som har høy prioritet blir splittet først, og de andre blir splittet etterhvert som de høyt prioriterte storiene er fullført.

User stories er formulet på følgende måte: “Som <aktør> vil jeg at <funksjon> så <fordel>”.

4.1 Sporbarhet

Hvert krav er oppfylt av en eller flere user story. En user story kan oppfylle flere krav.

User storiene har godkjenningskriterier som skal testes for at vi skal kunne si at user storien er fullført. Når alle user storiene som hører til et krav er fullført, er kravet oppfylt.

Kravtabellene har et felt for hvilke user storier som oppfyller kravet. User story tabellene har felt for hvilke krav den oppfyller, og hvilke tester som tester hvert godkjenningskriterie. Testtabellene for godkjenningstester har felt for hvilke user stories den tester.

4.2 Prioritet

Til krav har vi brukt triage for å prioritere. Triage er en metode for hurtigorganisering når tiden er knapp for å prøve å sikre effektiv bruk av ressurser og luke bort tidstyver. Det er en mye brukt metode i medisinske nødsituasjoner. [48]

Hvert krav får en prioritet A, B, eller C, hvor A er det som skal være med, C er det som ikke er så viktig, og B er alt mellom.

Vi bruker triage fordi det er en enkel og god nok måte å prioritere oppgaver på, slik at vi ikke kaster bort mye tid på å diskutere hvilken prioritet et krav skal ha; enten er det viktig, ikke viktig eller midt i mellom.

User storiene er prioritert i Product Backlog etter rekkefølgen vi planlegger å implementer dem.

4.3 Type krav

For å definere kravtyper følger vi FURPS: Functionality, Usability, Reliability, Performance, og Supportability. Kravene som gir funksjonalitet er funksjonelle krav. For kravene som går under Usability, Reliability, Performance, eller Supportability definerer vi ikke hvilken av kravtypene den hører til, men samler dem under typen ikke-funksjonell. Det er fordi vi ikke får noe ut av å spesifisere dypere.

User stories som er for store til å utføre i en sprint, er user story epics. Disse må splittes til flere mindre user stories før de kan implementeres. User stories som omhandler prosjektet og ikke implementasjonen av systemet, er prosjekt epics hvis de er store og må splittes, og prosjekt stories hvis de er små nok. User stories som omhandler features som en aktør ikke direkte bryr seg om er technical stories og er formulert som “Lag <feature>”.

4.4 Krav eksempler

Her er noen eksempler på våre krav. Resten av kravene finnes i vedlegg E

KravID	Dato	Kilde	Status	Prioritet
8	11.02.2018	Oppdragsgiver	Fullført	A
Beskrivelse: Brukergrensesnittet skal bare tillate gyldige valg				
Kravtype	Funksjonell			
User story ID	2.3.3			

KravID	Dato	Kilde	Status	Prioritet
50	11.02.2018	Oppdragsgiver	Fullført	C
Beskrivelse: Systemet skal loggføre brukernes valg av eksperimenter				
Kravtype	Funksjonell			
User story ID	4, 9, 11.5			

4.5 User story eksempler

Her er noen eksempler på våre user- og technical stories. Resten av storiene våre finnes i vedlegg F.

ID	2.1		
Krav	29		
Type	User story		
Beskrivelse	Som bruker vil jeg se en oversikt over alle tilgjengelige eksperimenter		
Godkjennings-kriterier	Det fins en liste over alle eksperimenter	Testet av	1.1
	Linkene i listen fører deg videre til tilhørende eksperiment		2.1
	Eksperimenter blir vist i katalogen under kategorien de hører til		3.1
Kommentar			

KravID	Dato	Kilde	Status	Prioritet
50	11.02.2018	Oppdragsgiver	Fullført	C
Beskrivelse: Systemet skal loggføre brukernes valg av eksperimenter				
Kravtype	Funksjonell			
User story ID	4, 9, 11.5			

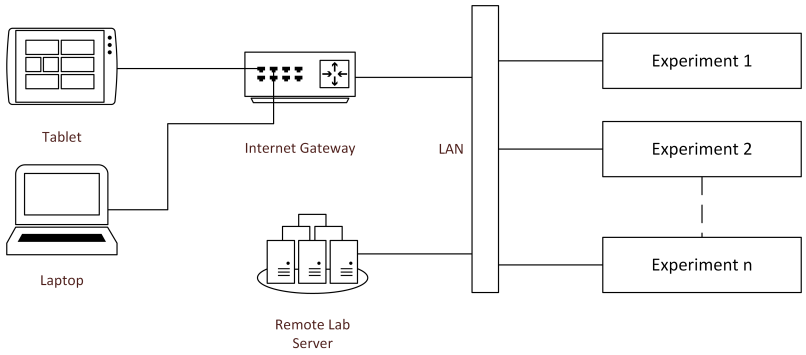
4.6 Godkjenningstest eksempler

Her er noen eksempler på våre godkjenningstester. Resten av testene våre finnes i vedlegg G.

ID	4.1	User story	2.2, 3
Beskrivelse	Når eksperimentutvikler integrerer sitt eksperiment med systemt, inkluderer det å legge til en beskrivelse av eksperimentet		
Avhengigheter			
Ha tilgang til konto med utvikler rettigheter Kontoen har integrert et eksperiment som har fått id 99			
Fremgangsmåte			
1. Logg inn med konto med utvikler rettigheter på nettsiden 2. Gå til ruten /development/setup/99 3. Skriv inn “dette er en beskrivelse” i description feltet 4. Trykk Add info			
Forventet Resultat			
Konfigurasjonsfilen inneholder felt for description med verdi “dette er en beskrivelse”			
Resultat			
Konfigurasjonsfilen inneholder felt for description med verdi “dette er en beskrivelse”			
Godkjent	Ja	Dato	21.5.18

ID	7.1	User story	2.3.1 11.3
Beskrivelse	I brukergrensesnittet til et eksperiment kan bruker legge inn verdier til parameterene til eksperimentet via tekstfelt		
Avhengigheter			
Eksperiment med id 10 er integrert Eksperiment med id 10 har parameter “P1” med type integer Eksperiment med id 10 har parameter “P2” med type floatingpoint			
Fremgangsmåte			
1. Gå til ruten /experiment/10 på nettsiden 2. Tast inn “14” i P1 feltet 3. Tast inn “1.111” i P2 feltet 4. Trykk Submit			
Forventet Resultat			
En fil blir laget i INPUT mappen til eksperimentet med id 10 Filen inneholder data på inputformatet hvor P1 feltet har verdi 14 og P2 feltet har verdi 1.111			
Resultat			
En fil ble laget i INPUT mappen til eksperimentet med id 10 Filen inneholder data på inputformatet hvor P1 feltet har verdi 14 og P2 feltet har verdi 1.111			
Godkjent	Ja	Dato	21.5.18

5 Produktdesign



Figur 2: System Architecture

Figuren over viser en typisk konfigurasjon av prosjektets Remote Lab løsning. I denne konfigurasjon kommuniserer brukeren med systemet via en internettside, og Remote Lab systemet tar seg av kommunikasjonen mellom internettsiden og eksperimentene gjennom et LAN.

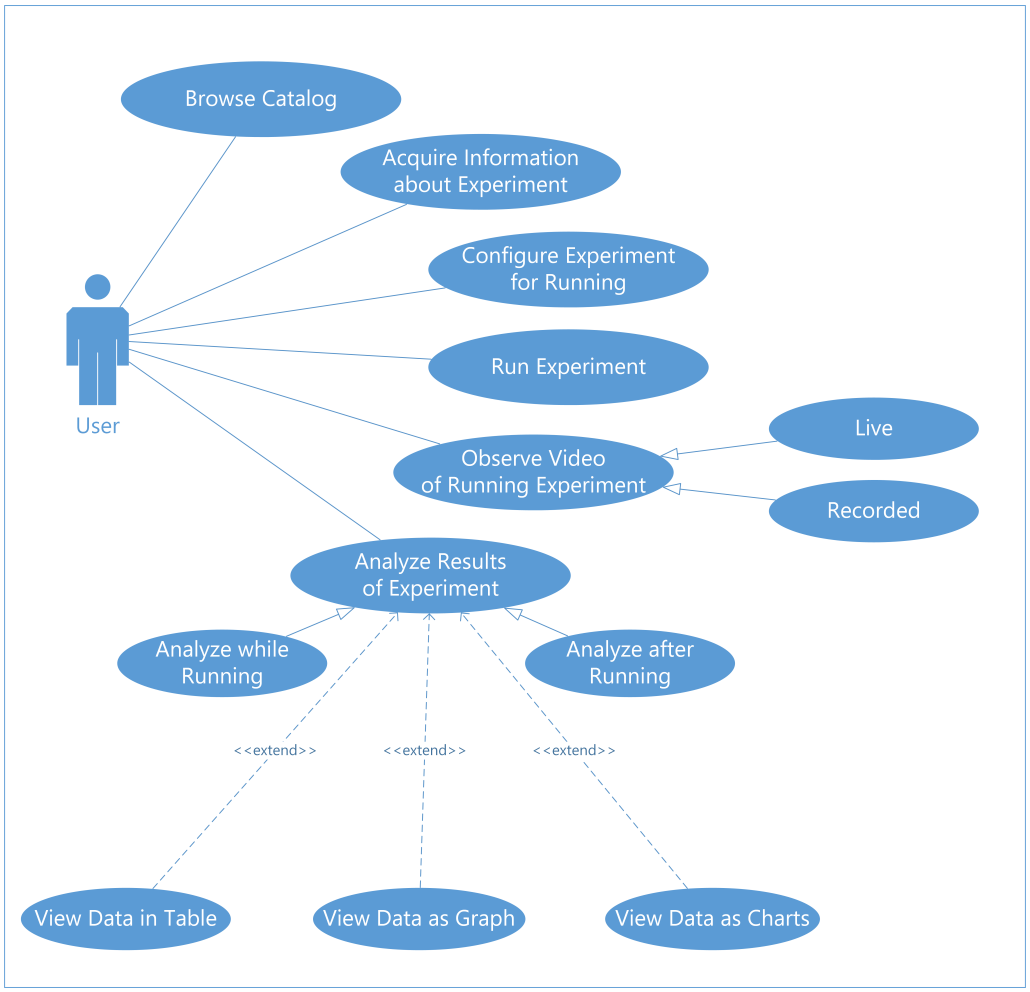
I alternative konfigurasjoner kan eksperimentene kobles direkte til serveren via, for eksempel, Bluetooth eller USB. En annen mulighet er at brukeren laster ned en app som kan være kraftigere og mer funksjonell enn det som kan implementeres i HTML og JavaScript.

5.1 Use Case Modell

UML diagrammene er på engelsk på grunn av deres tilknytning til koden som også er på engelsk

De 3 største aktørene i Remote Lab systemet er:

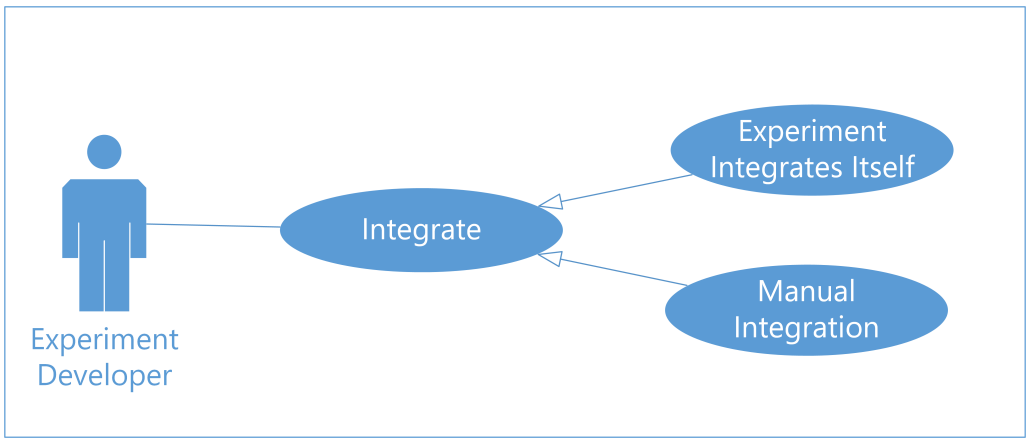
- Brukere
- Eksperiment utviklere
- Administratorer



Figur 3: Use Case - User

5.1.1 Brukere

Figur 3 viser at det er tydelige skiller mellom de forskjellige stegene. Introduksjonen nevner flere eksisterende Remote Lab implementasjoner, og noe som går igjen hos de fleste er at brukeren får tildelt tid med eksperimentet der de får mulighet til å sette opp eksperimentet, kjøre det og vente på resultater. Dette prosjektets tilnærming er annerledes da oppsett, kjøring og resultater ansees som 3 adskilte operasjoner. Dette forbedrer ressursutnyttelsen ved at eksperimentet ikke bindes opp den tiden som brukes til å sette opp og analysere resultatene. Designets modularitet gjør det allikevel mulig å implementere tidsvindu-tilnærmingen om dette er ønskelig.



Figur 4: Use Case - Experiment developer

5.1.2 Eksperiment utviklere

Den typiske eksperimentutvikleren vil være en maskin- eller elektroingeniør. Det betyr at de ikke nødvendigvis har kompetanse til å sette opp et brukergrensesnitt og andre typiske datarelaterte oppgaver. Derfor burde løsningen legge til rette for at eksperimentet kan legges til så enkelt som mulig såfremt utvikleren følger anvisningene i brukermanualen.

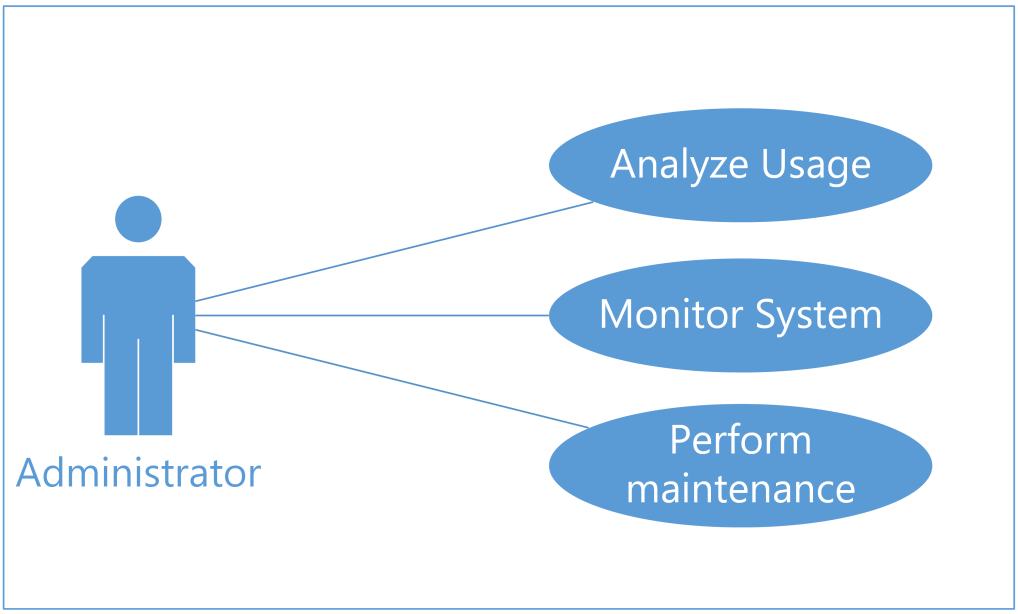
Enklere automatisert integrering krever en del kompromisser. Derfor burde også løsning ta hensyn til de som ønsker å overstyre automatikken og legge til funksjonalitet i form av avanserte input-kontroller og resultatvisning.

5.1.3 Administratorer

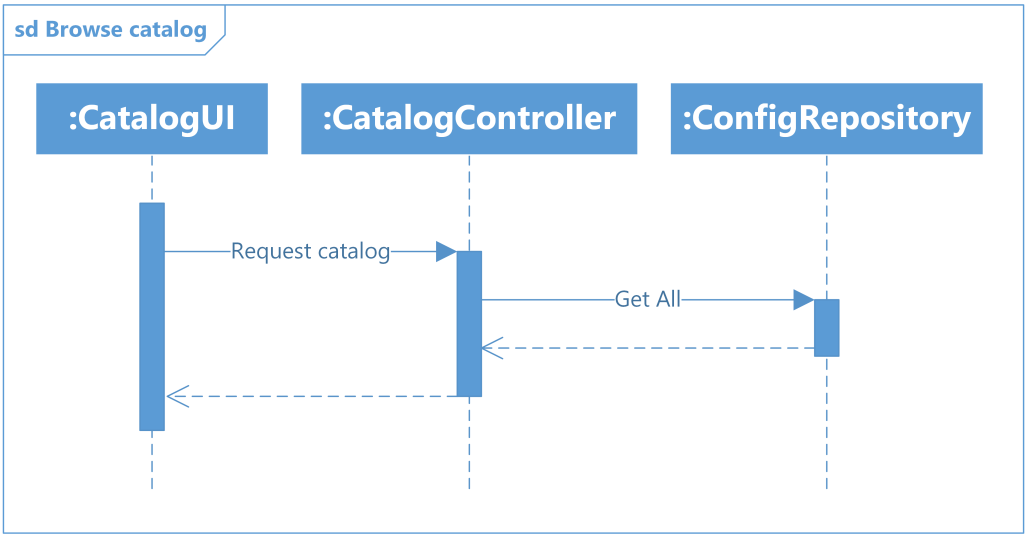
Administrator må ha mulighet til å overvåke og vedlikeholde systemet ved behov. Bruksanalyse vil ikke implementeres i dette prosjektet, men er en funksjon som oppdragsgiveren ønsker på sikt.

5.2 Sekvens Diagrammer

Følgende diagrammer gir overordnet logisk oversikt over hvordan systemet utfører de viktigste oppgavene.



Figur 5: Use Case - Administrator



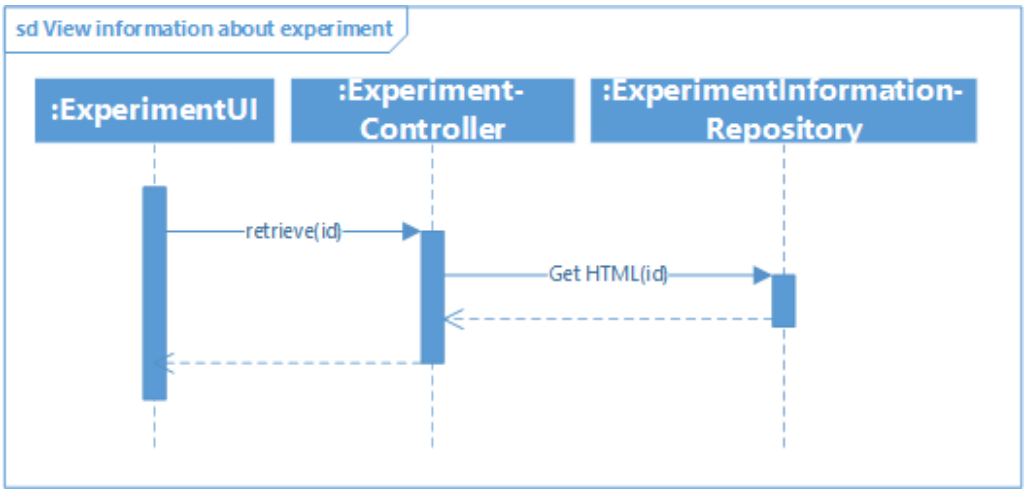
Figur 6: Interaction Sequence Diagram - Browse Catalog

5.2.1 Se eksperimentoversikt

Katalogen bygges ved å gå gjennom alle konfigurasjonsfilene på filserveren for å hente informasjon som eksperimentnavn og kategori. CatalogController bruker denne informasjonen til å generere kategoriserte lister og linker til eksperimentsidene.

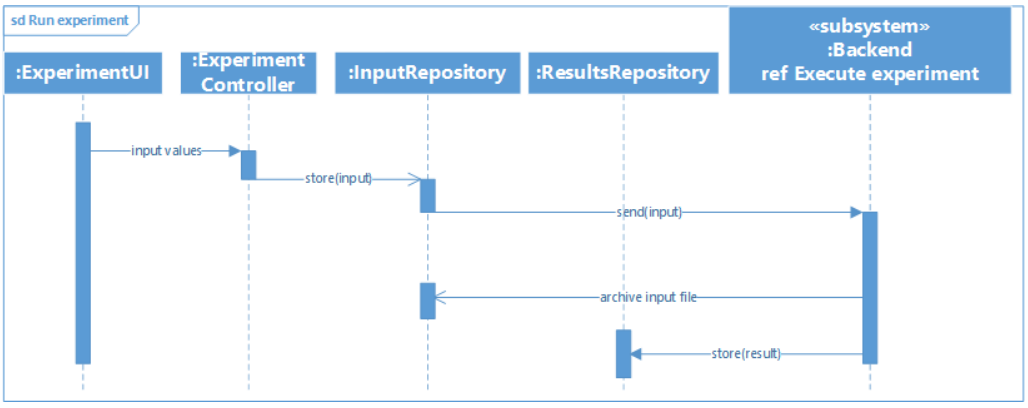
5.2.2 Les informasjon om eksperimentet

Dersom det eksisterer en tilleggsfil på serveren med tekst, bilder og annen media, blir denne vist på eksperimentsiden.



Figur 7: Interaction Sequence Diagram - View information about experiment

5.2.3 Kjør eksperiment - del 1

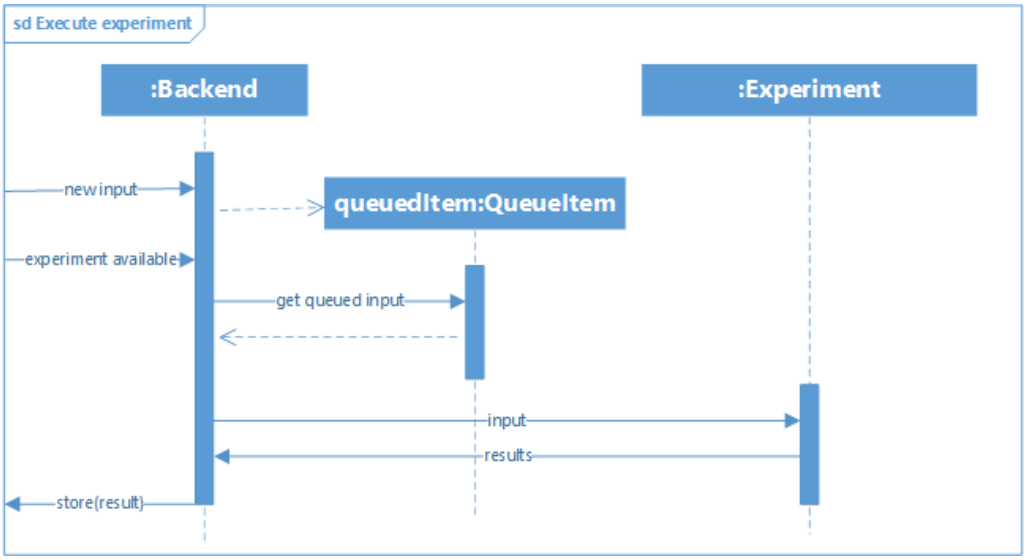


Figur 8: Interaction Sequence Diagram - Run experiment

Ved kjøring av eksperimenter blir input lagret på serveren og venter på å bli plukket opp av en backend tjeneste (se figur 9. Når eksperimentet er ferdig, blir resultatene lagt tilbake på serveren og input-filen arkivert.

5.2.4 Kjør eksperiment - del 2

Backend-tjenesten putter input i en kø, og når eksperimentet blir ledig blir den eldste inputen sendt til eksperimentet. Når eksperimentet er ferdig sendes resultater tilbake til backend tjenesten som lagrer resultatene.



Figur 9: Interaction Sequence Diagram - Execute experiment

5.2.5 Se resultater

Eksperiment grensesnittet venter på resultatene fra eksperimentet og viser disse til brukeren.

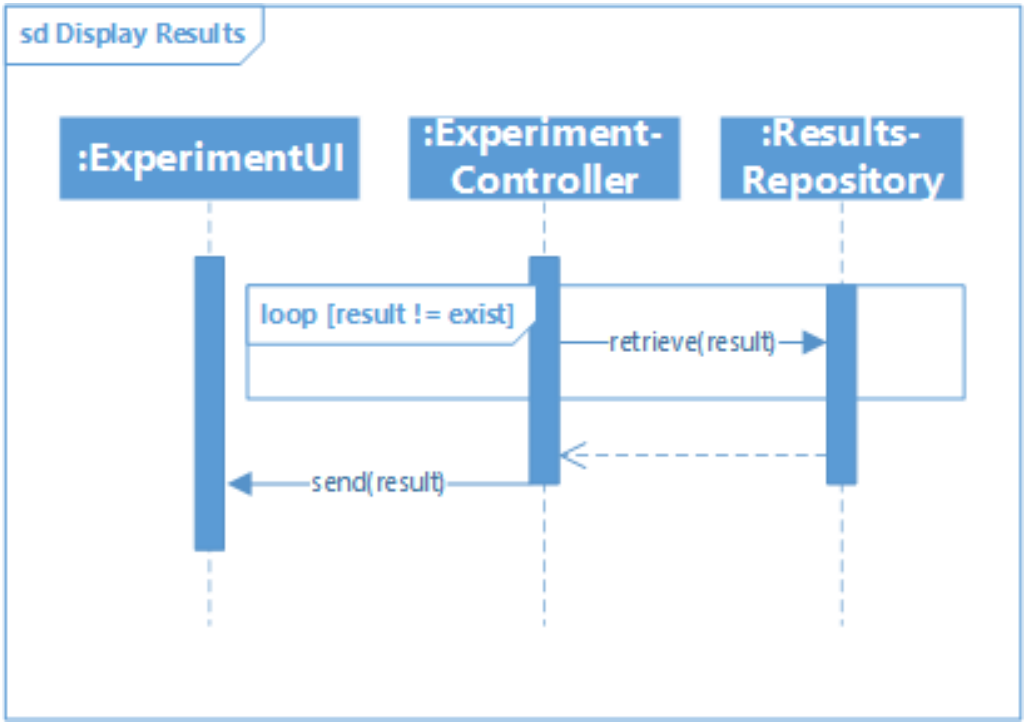
5.2.6 Integrere nye eksperimenter

Eksperimenter kan integreres både manuelt og automatisk. I begge tilfeller blir det produsert en identisk konfigurasjonsfil på serveren. Denne filen inneholder informasjon om eksperimentets navn, inn- og utfelter, enheter som brukes og annen informasjon som brukes til å integrere eksperimentet med brukergrensesnittet, samt identifisere innholdet i pakkene som sendes mellom systemet og eksperimentet.

Ved automatisk konfigurering, trenger man kun å oppgi IP adressen til det nye eksperimentet og backend tjenesten vil sende en forespørsel til det nye eksperimentet og be den om å oppgi navn, parametre og annen informasjon den trenger for å produsere en fullverdig konfigurasjonsfil. Dette forutsetter at eksperimentutvikleren har programmert inn en respons på forespørselen.

Ved manuell konfigurasjon lages konfigurasjonsfilen manuelt. Fordelene med den automatiske metoden er:

- Det er enkelt å legge til flere instanser av samme eksperiment



Figur 10: Interaction Sequence Diagram - Display results

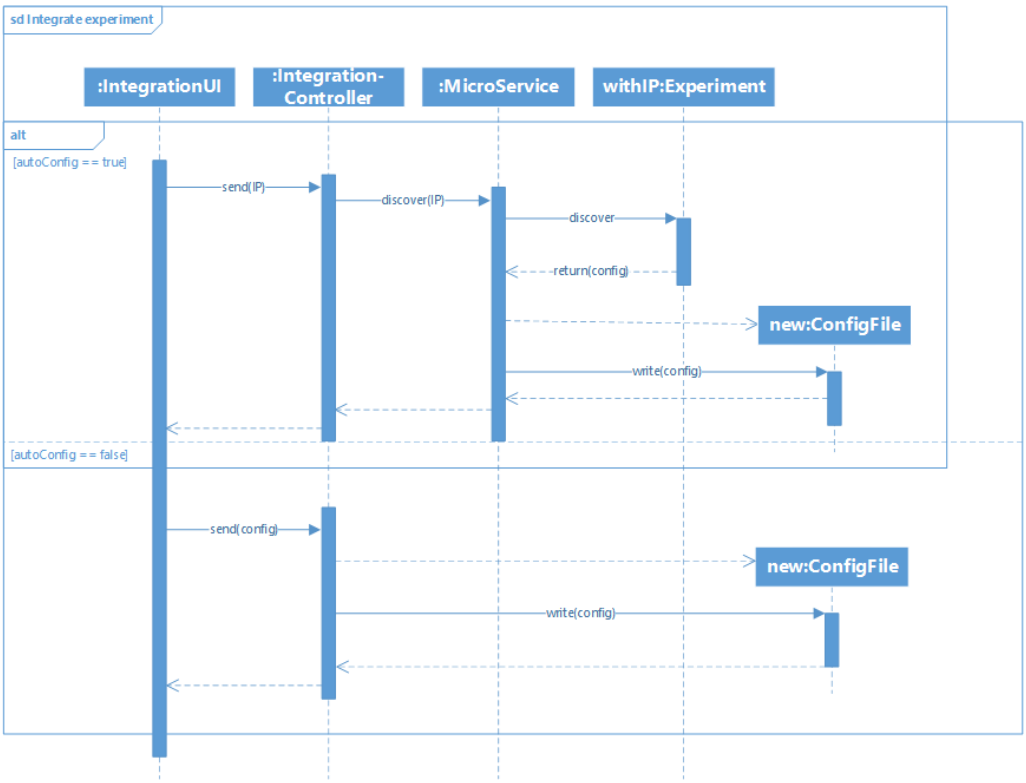
- Man slipper å skrive en konfigurasjonsfil i JSON manuelt
- Det er ikke behov for direkte tilgang til filserveren siden files opprettes automatisk
- Konfigurasjonsfilen produseres av kompilert kode som kan identifisere feil og mangler med en gang eksperimentet blir lagt til

5.3 Komponentdiagram

Figur 12 viser en overordnet logisk oversikt over systemet, og viser et tydelig skille mellom brukergrensesnitt, kontrollere, lagringsenheter og tjenester.

I sekvensdiagrammet kommuniserer integrasjonskontroller direkte med backend. Dette tildels for å forenkle figuren, og tildels fordi det er mulig å konfigurere systemet på denne måten. I dette prosjektet vil kommunikasjonen allikevel foregå via en proxy i form av et lagringselement, noe som vil gjøre skillet mellom de fire forskjellige logiske nivåene komplett.

Det er viktig å merke seg at vårt system anser eksperimentet som en lagringsenhet der input-data lagres og resultater kan hentes ut. Alt som skjer i selve eksperimentet foregår i isolasjon fra vårt system. Dette bidrar til å forenkle interfacet mellom dem.



Figur 11: Interaction Sequence Diagram - Integrate experiment

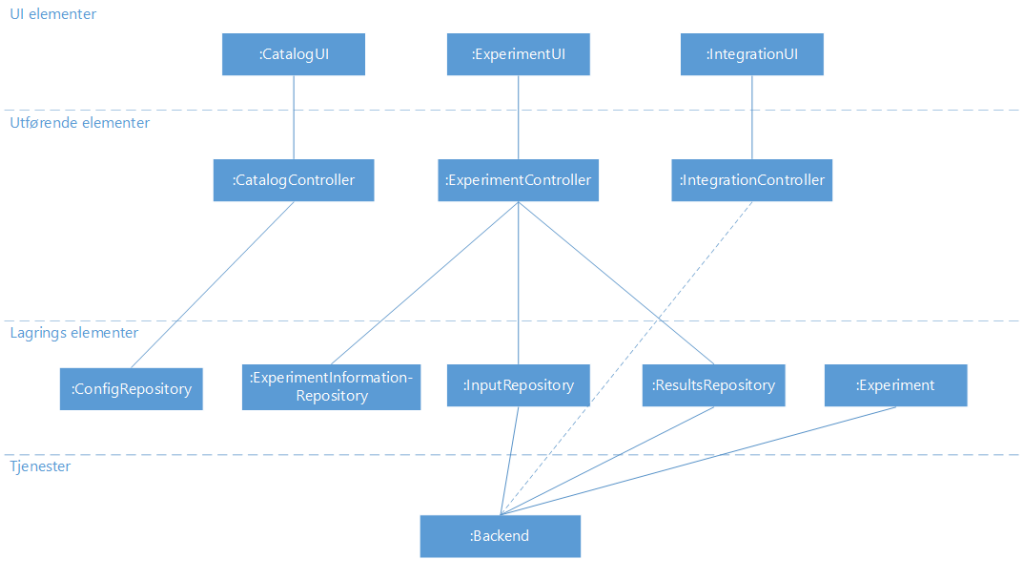
6 Implementasjon

Figur 12 viser fire adskilte nivåer der alle elementene på hvert nivå kan realiseres ved hjelp av samme teknologi som for eksempel:

- UI elementer - HTML/Express
- Utførende elementer - Node.js
- Lagrings elementer - JSON og filsystem
- Tjenester - C++ og TCP/IP

Unntaket fra denne listen er eksperimentene som kan realiseres i en valgfri teknologi såfremt de implementerer interfacet og protokollen på en korrekt måte.

Det er mulig å implementere både hele og deler av disse nivåene med andre teknologier, men listen gjenspeiler hva som er valgt i dette prosjekt. Det å holde antall teknologier til et minimum forenkler arbeidet med prosjektet.



Figur 12: Component Diagram - System overview

6.1 Core

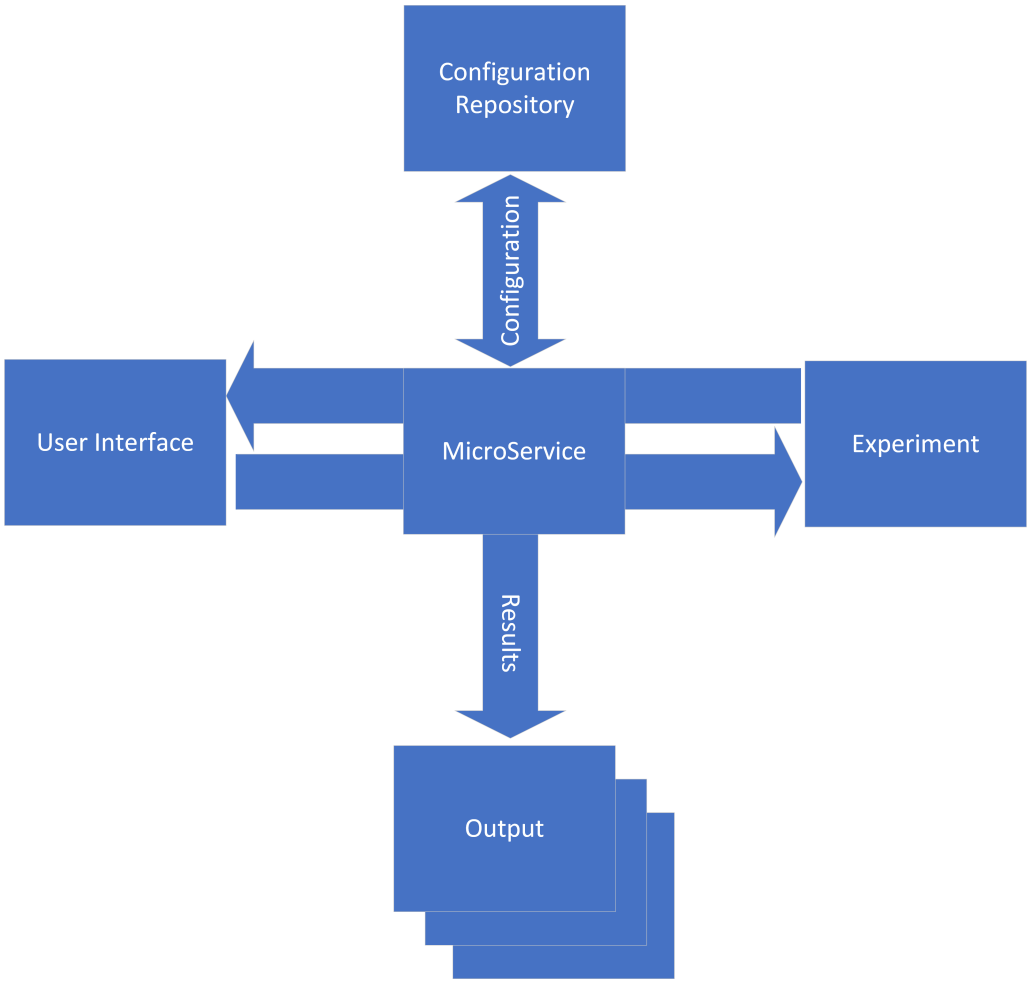
Systemet fremstår som ett system, men består i virkeligheten av ett brukergrensesnitt som binder sammen en rekke mikrotjenester; ett for hvert eksperiment. Dette skjer gjennom en fleksibel komponent som heter “Core”.

Core består av klassen *MicroService*, og grensesnittene som kobler sammen eksperimentet og andre tjenester som henter brukerinput og konfigurasjonsfiler, og skriver ut resultater. MicroService lytter etter nye meldinger fra de andre grensesnittene og videresender de til riktig grensesnitt.

Core er programmert etter SOLID prinsippene til Robert C. Martin, populært kalt Dr. Bob. Selv om Dr. Bob ikke lagde selve akronymet, er prinsippene som han beskrev i 2000 i artikkelen “Design Principles and Design Patterns” [49] veletablerte og anerkjente, og går igjen i mange bøker om Agile utvikling.

SOLID - Single Responsibility Principle Dette prinsippet sier at en klasse kun skal ha ansvar for én oppgave. Det reduserer sannsynligheten for at endringer i forskjellige deler av systemspesifikasjonene påvirker en og samme klasse.

MicroService har ett ansvar: å formidle meldinger mellom de forsjellige grensesnittene. Brukergrensesnittet har også ett ansvar: å formidle meldinger mellom MicroService og brukeren. Dette gjelder også alle de andre systemene som er koblet til Core.



Figur 13: Overview - Core

SOLID - Open/Closed Principle Open/Closed prinsippet sier at en klasse skal være åpen for utvidelser, og lukket for modifikasjoner. Core er laget på en måte som gjør at man kan bytte ut, for eksempel, konfigurasjonsgrensesnittet slik at den kan hente fra XML istedenfor JSON filer. Denne egenskapen er brukt aktivt i forbindelse med testing der de andre delene enn de som blir testet blir byttet ut med "Mocks" eller "Fakes". Mocks og fakes er klasser som kan simulere oppførsel uten å koble til eksterne tjenester som databaser eller filsystemer. Ved å gjøre testen uavhengig av eksterne systemer, utelukker vi muligheten for at det er de eksterne tjenestene som eventuelt får testene til å feile.

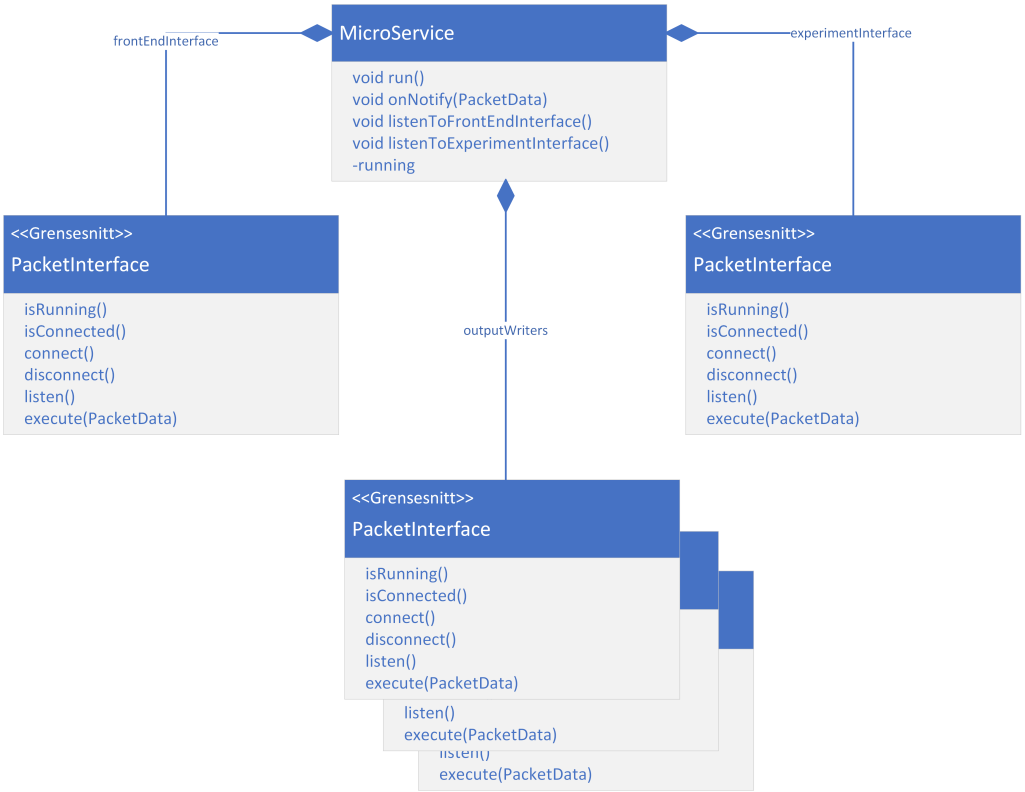
SOLID - Liskov Substition Principle Et objekt av en bestemt klasse skal kunne erstattes av objekter av en subklasse av førstnevnte klasse uten å ødelegge koden. Core inneholder objekter av typen *PacketInterface* som alle grensesnittene rundt arver fra. *PacketInterface* er en abstrakt klasse som erstattes av de konkrete front-end og eksperiment komponentene.

SOLID - Interface Segregation Principle Interface Segregation prinsippet sier at alle interface skal være spisset til én bestemt oppgave, istedenfor å prøve å samle flere oppgaver under én interface. PacketInterface håndterer kun overføring av pakkedata mellom objekter, og ingenting annet.

SOLID - Dependency Inversion principle Dette prinsippet sier at et objekt skal være avhengig av abstraksjoner, ikke konkretiseringer. MicroService kaller metodene til PacketInterface og har ikke kjennskap til de konkrete implementasjonene av PacketInterface.

I tillegg til SOLID prinsippene, implementerer Core noen "patterns". Istedenfor å bruke en dedikert "Inversion of Control" komponent, bruker Core en Factory klasse. Denne inneholder flere metoder som kan brukes til å bygge instanser av MicroService med de ønskete konkretiseringer av de abstrakte grensesnittene.

Den bruker også Observer pattern til å varsle MicroService når et av grensesnittene mottar informasjon utenfra.



Figur 14: Class diagram

Klassediagrammet viser hovedklassene til Core. MicroService kobler til eksperiment, frontend og resultatskrivere gjennom PacketInterface. Disse kan byttes ut mot alternative im-

plementasjoner etter behov som, for eksempel, om man ønsker en egen versjon som fungerer med USB istedenfor TCP/IP, eller Linux istedenfor Windows.

6.2 Interface

Interface er et system for at to eller flere aktører skal kunne kommunisere seg i mellom slik at det kan framstå som ett system. Dette avsnittet omhandler interface mellom eksperiment og mikrotjenesten. Ved å se på operasjonene som skal håndteres er det laget en protokoll som hjelper med realisering av interface og det valgte overføringsmedium er TCP.

6.2.1 Operasjoner

Remote Lab System har tre hoved operasjoner som krever samarbeid mellom flere systemer:

- Discover
- Kjør
- Service

6.2.2 Discover

Discover skjer etter at eksperimentutvikler har koblet sitt eksperiment opp mot mikrotjenesten. Ved å manuelt trykke på “discover request” på nettsiden, sendes det en discovery request til eksperimentet. Microtjenesten forventer å få en discovery response tilbake, som inneholder all informasjon som trengs for å sette opp et nytt eksperiment, blant annet informasjon om hvilken katalog den skal vises i. Appendix C viser en JSON fil av eksperimentet Ohm’s Law.

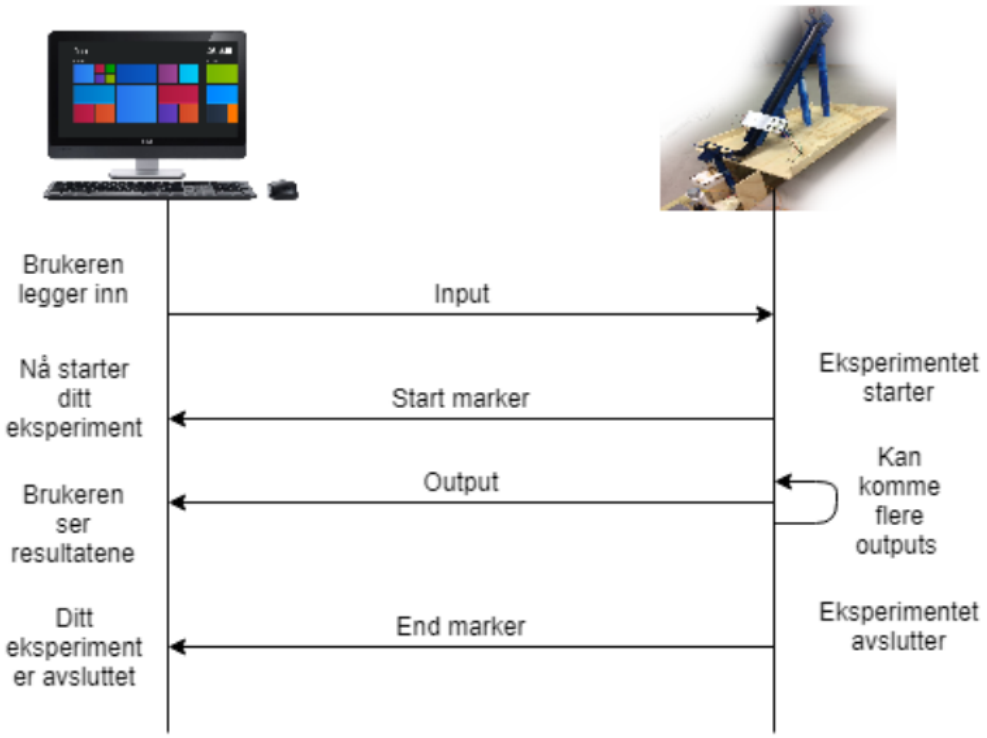
6.2.3 Kjør

Etter at et eksperiment er blitt oppdaget, vil den ligge tilgjengelig i systemet og brukere kan nå kjøre eksperimentet gjennom nettsiden. I kjøringsoperasjonen sendes det input

fra nettsiden til eksperimentet. Når eksperimentet mottar og kjører med disse inputparameterene, sendes det en start marker tilbake. Dette gjøres for å kunne informere en bruker om kø, og når start marker kommer tilbake til mikrotjenesten kan vi oppdatere nettsiden og gi brukeren beskjed om at hans eksperiment er klart til kjøring.

Mens kjøring av eksperiment pågår, eller ved endt kjøring, sendes det output fra eksperimentet til mikrotjenesten som inneholder resultater. Når alle resultater er sendt og eksperimentet er ferdig med å kjøre sendes en end marker for å markere slutt på denne brukerens eksperimentkjøring.

Kjør operasjonen er vist i figur 15



Figur 15: Kjøring av systemet

6.2.4 Service

Service brukes av eksperimentutvikleren til å sende en pakke med en servicekode, for eksempel, for justering av et belte eller re-kalibrering av en heis.

6.3 Protokoll

For at interface skal realiseres trengs det en metode for å kommunisere mellom systemene, dette gjøres ved hjelp av en protokoll. En protokoll er et standardisert sett med regler som bestemmer tilkobling, kommunikasjon og dataoverføring mellom to endepunkter. Det er ikke mulig å sende floats eller int direkte, og en protokoll hjelper ved at den gjør alt om til bytes som lagres i en pakke, som sendes til en mottaker som der oversetter bytes tilbake til int, floats osv.

6.3.1 Remote lab protokoll filosofi

Protokollen er designet slik at en eksperimentutvikler kan kjøre en autodiscover dersom han har fulgt brukermanualen, og dermed blir eksperimentet autoregistrert i systemet. Protokollen skal være "tynn" der output har ingen overhead, for å maksimere kapasiteten. Det var og et ønske om at protokollen skulle være så fleksibel som mulig. Det mener vi at vår protokoll er, ved å støtte alle datatyper og strukturer.

Protokollen er laget slik at hver pakke bruker de første 3 bytes til å lagre packetID (1 byte) og size (2 bytes), med unntak av output.

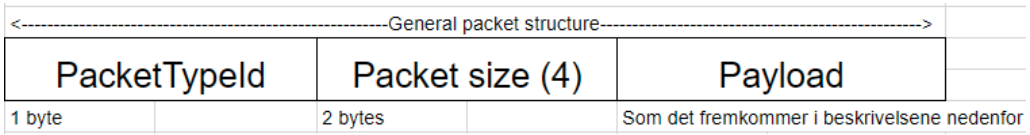
Valget med å sende en packetID og size er for å redusere sjansen for manglende data. Med packetID vet mottaker akkurat hva den skal parse, mens med size vet den hvor mange bytes det er forventet å motta for den pakken.

Dersom et eksperiment har flere output, og en output sendes oftere enn en annen, ønsker vi å unngå å sende tomme bytes. Derfor har vi laget en pakke som heter output with channel som brukes i slike tilfeller. Dette gjøres for å maksimere outputhastigheten, ved å ha ingen overhead. For å finne riktig output, gis parameteren en channelId.

6.3.2 Protokollbeskrivelsen

Protokollen består av flere pakketyper. Disse følger den generelle strukturen som vist i figur 16. PacketTypeId er Id-en til pakketypen som sendes i payload. Dens oppgave er å gi mottaker informasjon om hvilken type pakke som kommer, og på den måten redusere sjansen for feilbehandling av pakken. Liste over pakketypene finnes i tabell 1. PacketSize

er den totale størrelsen på alt som blir sendt. Den inkluderer PacketTypeId og PacketSize, som alltid er 1 og 2 bytes, i tillegg til payload. Payload varierer i størrelse. Mange av pakkene har fast størrelse, mens noen har variert størrelse som defineres av eksperimentutvikleren. For tabell av hver enkelt pakketype se appendix B.



Figur 16: Generell Struktur

Packet Type Id	Packet name
00	Discovery request
01	Discovery response header
02	Discovery response parameter
03	Video sync marker
04	Input
05	Output
06	Output with channel
07	Start marker
08	End marker
09	Service
10	Experiment runID
11-255	For further use

Tabell 1: Pakketyper

Discovery request Blir brukt når en eksperimentutvikler ønsker å kjøre autodiscover for å koble sitt eksperiment til Remote Lab System. Det er en liten pakke, og som en sikkerhet sendes det en streng med pakken for å minske muligheten for at pakken tolkes feil. Når det sendes en discovery request fra microtjenesten til eksperimentet, ventes det å få en discovery respons tilbake.

Discovery response header Brukes når et eksperiment mottar en discovery request. Pakken inneholder informasjon om eksperimentet. Denne informasjonen legges inn av eksperimentutvikler. Pakken har en fohåndsbestemt størrelse, men den sendes også bare en gang, i oppkoblingsfasen av eksperiment mot system. Rett etter discovery response header er det forventet at det kommer en discovery response parameter.

Discovery response parameter Sendes etter discovery response header. Innholder informasjon om parameterene til eksperimentet. Her bestemmer eksperimentutvikler viktige ting som størrelse på input og output verdier. Når denne pakken er mottat og behandlet, blir det opprettet en konfigurasjonsfil, som inneholder all informasjon nødvendig for å få tilgang til eksperimentet gjennom frontend.

Video sync marker Brukes dersom det er video tilknyttet eksperimentet, og denne videoen skal synkroniseres opp mot et resultat. Fra eksperimentet sendes det en "start video marker" som får eksperimentet til å sende en video sync marker pakke. Ved mottak blir tiden registrert, samtidig som alle tidspunkt på resultatene blir registert og dermed er video synkronisert med resultatene.

Input Sender verdiene brukeren velger på frontend til eksperimentet. Antall verdier og størrelsen på pakken varierer fra eksperiment til eksperiment, og er forhåndsdefinert av eksperimentutvikler.

Output Sender resultatet fra eksperimentet så det kan vises på frontend. Output har ikke med PacketSize, for å minske overhead. Størrelsen på output avhenger av konfigurasjonsfilen til eksperimentet. Dette valget er gjort for å sørge for så rask overføring av resultatdata som mulig.

Output with channel Brukes dersom et eksperiment har en eller flere outputs, men de sendes ikke samtidig. Hver resultatparameter vil da få en kanal som brukes for å sende resultatet. Størrelsen på output avhenger av konfigurasjonsfilen til eksperimentet. Denne pakken har ikke med PacketSize for å minske overhead, og ved å bruke kanaler unngår vi å sende tomme bytes. Her brukes samme tankengang som ved vanlig output, at vi skal unngå å sende overhead, for å sørge for rask overføring av resultatdata.

Start marker Pakken sendes fra eksperimentet til mikrotjenesten i det eksperimentet startet på en ny kjøring. Start markeren mottas og brukes for å informere brukere om at eksperimentet er satt i gang.

End marker Pakken sendes fra eksperimentet til mikrotjenesten i det eksperimentet sender den siste resultatpakken fra en kjøring. End marker mottas og brukes for å informere brukere om at eksperimentet er ferdig.

Service Pakken brukes dersom eksperimentutvikleren har laget et system for service på sitt eksperiment. Den sendes til eksperimentet med en ID som indikerer nummeret på den servicen som skal kjøres. F.eks en rekalibrering.

Experiment runID Pakken brukes til å navngi resultatfiler. Den kopierer navnet fra inputfilen og setter denne som navnet på outputfilen.

Ubrukte ID-er Id fra 11-255 er ubrukte og kan brukes til å lage nye pakker dersom det skulle være ønskelig.

6.3.3 Overføringsmedium

Det fins mange overføringsmedium som kan tas i bruk for å sende kommunikasjon mellom eksperiment og mikrotjenesten. Noen av de mer vanlige er:

- USB
  - Avhengig av ledning som er koblet direkte fra eksperiment inn i maskinen hvor mikrotjenesten kjører.
- Bluetooth
  - Begrenset rekkevidde, og et dårlig valg når systemet snakker “1 til mange”.
- TCP
  - Viktigst for oss, fordi eksperimentene og mikrotjenesten vil være på forskjellige lokasjoner. Mer om TCP kan leses under nettverk

6.4 Nettverk

For nettverksprotokoll har vi to valg, enten TCP eller UDP. Begge er protokoller brukt til å sende pakker med data.

TCP

Den mest brukte protokollen på internett. Den sørger for å spore datapakkene, slik at mottaker alltid får alle datapakker den skal ha, og man unngår korrupte pakker. Dette gjøres ved å gi hvert tegn i strømmen et sekvensnummer, som også blir brukt for å forsikre at pakkene blir levert i riktig rekkefølge hos mottakeren. Den bruker bekreftelser, slik at avsender får vite at data har blitt mottatt. Vi har ingen garanti for at data kommer frem, men dersom mottaker ikke får data den forventer, vil den kunne si ifra til avsender, og be om at det sendes en ny pakke med den manglende dataen. [50]

UDP

Sender ut alle pakkene den skal, om noen pakker ikke kommer frem, blir de mistet, da det ikke blir sendt noen nye pakker for å dekke tapet. Med andre ord har UDP ingen leveringsgaranti. UDP er mer vanlig brukt ved live sendinger, som f.eks live tv, eller ved online spill. [51]

Vår protokoll er så tynn at UDP ikke blir vårt valg, og i tillegg ønsker vi ved de fleste tilfeller å ha en error sjekk som sørger for at alle pakkene kommer frem til mottaker.

6.4.1 Server - Klient basert kommunikasjon

Kommunikasjonen foregår mellom server og klient, hvor en klient sender forespørsler til serveren og får svar tilbake. I vårt system blir eksperimentene servere som sitter og lytter etter mikrotjenesten som dermed er en klient.

Socket En nettverkssocket er et endepunkt for sending eller mottak av data i en node i et datanettverk. En socket blir assosiert med en spesifikk socket adresse, som er IP adressen og et port nummer for den lokale noden. I motsatt ende er det en annen node som har en tilsvarende socket adresse med sin IP adresse og port. Denne prosessen kalles binding.

Etter at socket har fått en adresse, begynner den å lytte etter innkommende forespørsler. Dersom det kommer inn en forespørsel til socketen, må denne aksepteres før det oppstår en forbindelse. Det blir nå laget en ny socket tilknyttet denne forbindelsen slik

at den kan fortsette å lytte etter andre forespørsler. All kommunikasjon skjer nå via den nye socketen.

Etter at forespørsel om forbindelse er akseptert, velger begge å koble til hverandre og det har nå oppstått en forbindelse for kommunikasjon. Nå kan begge parter sende og motta forespørsler, og når de er ferdig avsluttes forbindelsen.

6.4.2 Sikkerhet

Core(klient) som kobles opp mot eksperimenter(server) skjer via et privat LAN. Eventuelt kan det opprettes en VPN forbindelse, noe som kan være aktuelt dersom USN sine avdelinger rundt om i landet ønsker å koble opp eksperimenter mot systemet som står i Kongsberg. Utenom dette er det kun frontend som trenger å lagre filer som systemet får tak i, ellers er det ingen åpne kanaler.

6.4.3 Fordeler

Å koble til systemet gjennom tcp/ip er relativt billig, på den måten at du kan få mye overføringshastighet for en lav pris. Dersom ønskelig er det mulig å oppgradere mye på hastigheten, men dette vil da være mer kostbart.

Systemet er koblet opp “1 til mange” og det vil fort oppstå problemer ved bruk av bluetooth, eller mangler på usb-kontakter, ved bruk av USB.

I tillegg anser vi det som naturlig at det vil være avstander mellom systemet og eksperimentene, f.eks ved at systemet står hos USN avd Kongsberg, mens USN avd Porsgrunn lager et vannkrafteksperiment som står i Porsgrunn. De ønsker da å koble seg opp mot systemet som står i Kongsberg.

6.4.4 Ulemper

Det er mer arbeid med å sette opp et system som skal bruke nettverkstilkobling, kontra å bare plugge inn en ledning.

6.5 Kø- og filbehandlingsystem

I dette kapittelet vil vi forklare hvordan vi har implementert kø-/filbehandlingssystemer i vårt prosjekt. Vi går gjennom hvordan filer håndteres fra de blir sendt fra bruker og til resultatene vises på skjermen.

6.5.1 Køsystem

Systemet skal være skalerbart og med hensyn til dette, så er det behov for et system som tar hensyn til at flere brukere ønsker å bruke systemet samtidig.

De fysiske eksperimentene lar kun én bruker kjøre eksperimentet om gangen. Dette medfører at hvis det er flere bruker som ønsker å bruke sammen eksperiment samtidig, må det lages et kø-system som kan håndtere at flere brukere «bestiller» samme eksperiment, men at kun en bruker får tilgang til selve eksperimentet om gangen.

Siden kø funksjonalitet allerede er et standardbibliotek i C++, bruker vi denne løsningen. Oppdragsgiver har gitt gruppen krav som omhandler kø. Dette må vi ta hensyn til når vi velger hvordan kø system vi skal velge.

FIFO (First in-First Out) kø er valget vi har gått for. Valget baserer seg på at dette er et standard bibliotek, slik at vi ikke trenger å laste ned plugins eller lignende. Samtidig vil denne måten å håndtere kø på være den mest rettferdige ovenfor brukerne. En FIFO kø bruker førstemann-til-mølla prinsippet. Den første som bestiller et eksperiment havner først i køen og får dermed tilgang til eksperimentet først når det er klart.

Det er knyttet opp en microtjeneste til hvert eksperiment og det er en FIFO kø per microtjeneste.

Filbehandling

For å lage mappestrukturen bruker vi filesystem biblioteket. Vi definerer først rootpath (heretter kalt root) som filbehandlingssystemet benytter seg av. Root er mappen hvor inputfilene fra brukeren legges når de blir sendt til serveren. Deretter generer vi automatisk undermapper som vi har gitt navnene queue, archive, running, error og result.

Metoden som lytter til root mappen etter endringer er laget slik at det er uvesentlig hvordan type filformat som legges i denne mappen. Kø-/filbehandlingssystemet vil behandle filen uansett type. I løsningen vår har vi valgt å gå for JSON filformatet(**??**).

Biblioteket for kø heter queue og gir oss all funksjonalitet vi er ute etter.

Når det blir oppdaget en, eller flere, filer i root, vil køsystemet legge filene til i køen vi har kalt fileQ. Deretter flyttes filene til queuemappen og samtidig sjekkes det om eksperimentet er opptatt.

Dersom eksperimentet er ledig, flyttes den eldste filen i køen til runningmappen, innholdet i filen konverteres og deretter sendes dataen til eksperimentet. Konverteringen må gjøres for at eksperimentet skal forstå dataen i filen og hvordan den skal behandle den.

Når end-marker (beskrevet her**??**) mottas, vil programmet tolke dette som at eksperimentet er ferdig og filen blir flyttet fra runningmappen til archivemappen. Dette lar neste fil i køen få tilgang til runningmappen og eksperimentet.

Frontend lytter etter resultatfilen som kommer fra eksperimentet, henter resultatene og viser de til brukeren.

Løsninger som vi har jobbet med, men som vi har valgt bort: Vi hadde planlagt å benytte oss av en kode tilgjengelig på nett fra Microsoft [**msdn**] som tok seg av filovervåkning. Dette var utgangspunktet for prototypen og vi ønsket å implementere dette i sluttløsningen også. Vi ønsket kun varsler når det ble lagt til en ny fil, men denne løsningen gav varsel hver gang en fil ble endret, slettet eller lagt til. I tillegg måtte det gjøres endringer på eksempelkoden vi hentet fra nettet for å få dette til å fungere sammen med vårt system. Eksempelkoden var avansert, godt utenfor pensum og siden gruppen ikke forsto alle aspekter av koden, utgikk denne løsningen.

Filbehandling biblioteket ble valgt fordi det var enkelt å forstå og det var et bibliotek som det ble snakket mye om på f.eks Stackoverflow. Det eksisterer andre biblioteker vi kunne brukt, men de ble ikke forsket på siden biblioteket vi har brukt utførte det vi ønsket. I tillegg er filesystem et standard bibliotek i C++, derfor kan det antas at dette er nøye testet før det ble implementert.

6.6 Nettsiden

For en bruker som ønsker å kjøre et eksperiment vil den første brukeropplevelsen være interaksjon med nettsiden. Nettsiden er designet i samarbeid med en masterstudent på USN, hvor masteroppgaven tar for seg blant annet research for utarbeiding av UI design med fokus på brukervennlighet.

Remote lab er tilhørende USN og vi har valgt å hente inspirasjon fra USN sine egne nettsider, samtidig som vi har gjort egne designvalg basert på research gjort av masterstudenten. Masterstudenten har ingen erfaring med UI design eller webprogrammering, vi har tatt forbehold om dette og kommet med innspill og forslag til endringer der vi har følt behov for det.

6.6.1 Design

Vi har valgt å bruke USN sin logo siden dette er et produkt for USN, og har fulgt designmanualen [52] for plassering av logo. Nettsiden bruker farger fra samme designmanual. Bakgrunnsbilder og bilder brukt som thumbnail er bilder tatt av labber eller andre faglige rom/utstyr hos USN. For ikoner brukes font awesome3.16.

6.6.2 Implementering

Som forklart i teknologidokumentet har vi valgt å bruke Node.js for både frontend og backend. Som database brukes MongoDB og PUG danner grunnlaget for view templates.

Det var ingen ønsker om å ha en egen mobilapp til systemet, og derfor har vi sørget for responsivt web design, med fokus på “mobile first” [53]. Dette går ikke på bekostning av andre skjermstørrelser, men sørger for at brukeropplevelsen på mobil er like bra som på en datamaskin. Med disse valgene ble det naturlig å bruke bootstrap 4 som et rammeverk for å lage nettsiden.

6.6.3 Layout template

For de aller fleste nettsider er det helt naturlig å holde en lik struktur på mye av innholdet. Vi har en template for hovedlayouten til nettsiden som inneholder:

- Header tittel
  - Endrer seg automatisk når det genereres en ny html side
- Navigeringsbar
  - Inspirasjon fra USN, ligger som "hamburgermeny" uansett skjermstørrelse
  - Inneholder linker videre til menyer for eksperimenter, kontakt side, login, registrering mm.

Navigeringsbaren er dynamisk slik at når en bruker logger inn, byttes login og signup ut med profile og logout. Dette gjøres enkelt ved en sjekk i cookien om det finnes en bruker. Dersom bruker finnes, hentes brukernavn, og dette navnet vises som profilnavn til brukeren. Figur **??** viser navigeringsbaren før innlogging, mens figur **??** viser navbar med aktiv bruker.

Figur 17: Ingen bruker innlogget

Figur 18: Bruker innlogget

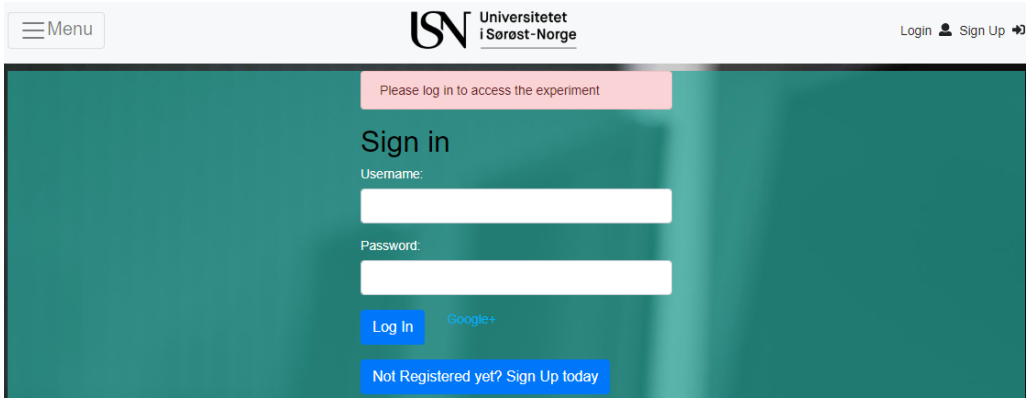
Layouten inneholder alle scripts som er felles samt linker til stylesheet for bootstrap 4, og vår egendefinerte stylesheet, som brukes til å spesialtilpasse CSS'en dersom vi ikke er fornøyd med det bootstrap tilbyr.

Denne layouten blir videre inkludert i alle andre templates ved en enkel kommando på første linje i hver fil.

6.6.4 Innholdet

Den første brukeropplevelsen oppstår på forsiden. Forsiden er designet for å gi brukeren tilstrekkelig med informasjon om hva som finnes, vekke en viss grad av nysgjerrighet, samtidig som den tilbyr vanlige funksjoner som login og registrering.

En bruker kan manøvrere seg rundt på nettsiden, men noen deler er sperret for tilgang dersom brukeren ikke er logget inn. For å tydeliggjøre dette, er det lagt inn et meldingssystem via flash meldinger som vil informere brukeren om viktige ting, som for eksempel, at en side ikke er tilgjengelig. Når brukeren prøver å få tilgang, vil han bli omdirigert til login siden og flash meldingen vil tydelig bli vist frem øverst i innholdsområdet, Som vist i figur 19.

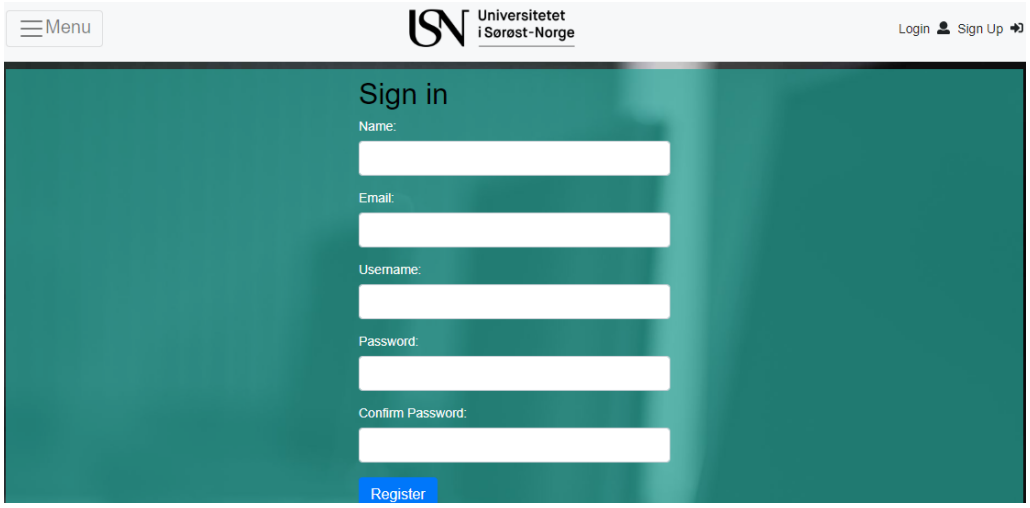


Figur 19: Ingen tilgang til eksperiment uten å være innlogget

6.6.5 Registrering

For å registrere brukere er det laget en model fil kalt user.js. I den filen er det laget et skjema som definerer alle ting vi ber brukeren legge inn ved registrering. Brukere kan registreres ved å legge inn email, passord, brukernavn og navn. Email og brukernavn kontrolleres i databasen, slik at det er umulig å ha to like. Ved feil gis det tilbakemelding til brukeren om hva som er feil, eller hva som mangler. Figur 21 viser registreringssiden.

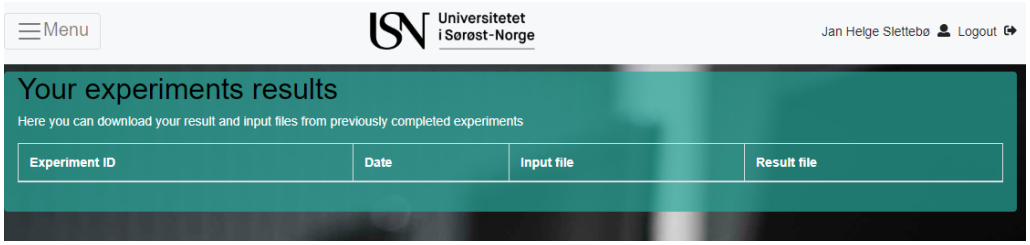
Alle brukere lagres lokalt i en database. Det kan velges mellom to måter å registrere seg på. Ved en lokal strategi, som nevnt over, eller ved å authorisere seg gjennom Google+ via OAuth.



Figur 20: Registreringssiden med tilbakemelding

6.6.6 Profilmeny

Profilmenyen lar brukeren endre passord, og den inneholder en tabell over alle kjørte eksperimenter gjort av brukeren. Denne tabellen vil vise hvilke eksperiment brukeren har kjørt og når de ble kjørt. Det er også en link til resultatfilen som ble produsert da eksperimentet kjørte.

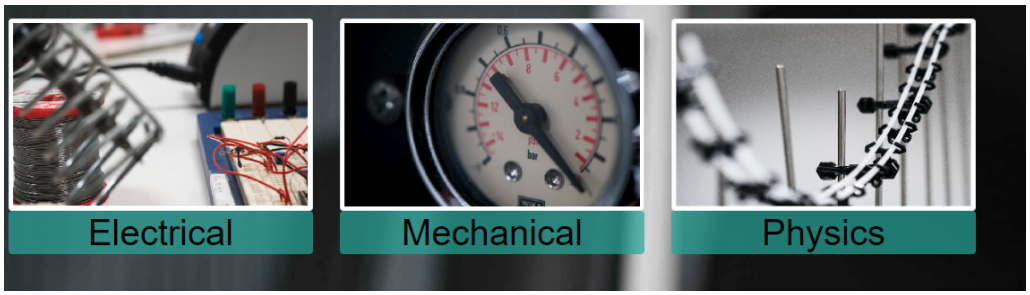


Figur 21: Profilmeny med historikk

6.6.7 Eksperimentkatalog

Eksperimentkatalogen er dynamisk og viser brukeren alle tilgjengelige eksperimenter. Brukeren kan enten navigere seg frem til ønsket eksperiment gjennom navigeringsbaren, eller han kan få opp en nettside som viser en oversikt over tilgjengelige kategorier. Ved å velge ønsket kategori åpnes en ny katalog som viser alle eksperimenter tilhørende den valgte kategorien.

Hvordan vi finner eksperimentene kan leses i 6.7.1



Figur 22: Katalogsiden

6.6.8 Eksperimentsiden

Eksperimentsiden har en "oppgavetekst" øverst, deretter følger et vindu med en tegning eller bilde av eksperimentet. Her vil brukeren også finne inputfelter og en knapp for å kjøre eksperimentet. Mer om inputfeltene kan leses i avsnitt 6.7.3.

Det er et valg for brukeren og vise tips for å løse oppgaven. Disse er kun tilgjengelig dersom eksperimentutvikleren har lagt de inn. Nederst er det et vindu som viser resultatene. Disse kan vises på flere måter, deriblant gjennom graf, kakediagram, tekst osv. Hvordan dette fungerer kan leses i avsnitt 6.7.5. Her er det også lagt til rette for å vise video.

6.6.9 Utviklersiden

Utvikler siden er laget for eksperimentutviklere og kun brukere med en egen administrator rettighet vil ha tilgang. Brukermanualen viser hvordan en administrator kan gi eksperimentutviklere tilgang til utviklersiden. Utviklersiden er bygd opp slik at eksperimentutvikler først får opp en side med alternativer for å legge til sitt eksperiment. Figur 23 viser hvordan det kan velges mellom:

- Laste opp en config fra lokaldisk
- Endre en eksisterende config (dersom eksperimentuvikleren allerede har lagt til et eller flere eksperiment)
- Be om automatisk opprettelse av config fra sitt eksperiment
- Lage en config manuelt via nettsiden

Dersom det legges til et nytt eksperiment videresendes eksperimentutvikleren til en ny side som gir brukeren mulighet til å legge til oppgavebeskrivelse, katalog, og bilde av

eksperimentet.

Mer om utviklersiden kan leses i 6.7.8.

Figur 23: Utklipp av utviklersiden

6.6.10 Cookies

Bruken av cookies er elsket og hatet. På denne nettsiden er det minimalt som lagres i en cookie. Dersom det skulle være ønskelig med en tydeligere logging av brukere, eller brukeres adferd, ville det vært enkelt å legge til data som lagres i brukerens cookie som vi videre kunne hentet ut og lagt inn i en datainnsamling. Dette kan være ting som tids-

punkt for kjøring av eksperimenter, hvilke land folk sitter i når de kjører et eksperiment osv.

6.7 UI

En webserver brukes til å sende og generere et webgrensesnitt til brukeren. Den håndterer registrering av brukere, login, utviklerverktøy, sende input til, og vise resultater fra eksperimenter.

Webserveren er bygget med Node.js og Express. Den tar imot en HTTP forespørsel fra brukeren og håndterer den ved å sende den gjennom flere middleware funksjoner og kaller kontrollerfunksjonen til ruten.

En middleware funksjon er for eksempel body-parser, som henter dataen sendt med i en HTTP forespørsel og gjør den tilgjengelig for kontrollerfunksjonen som et Javascript objekt.

Ruten til HTTP forespørselen er den delen av URLen som kommer etter domenenavnet, men før spørrestrengen. I URLen www.remotelab.no/eksperiment/ohms-lov?R=2&U=1 er www.remotelab.no domenenavnet, /eksperiment/ohms-lov ruten og ?R=2&U=1 spørrestrengen.

Vi definerer en rute til hver side på nettsiden, og en kontrollerfunksjon som håndterer forespørselen og returnerer en HTML-side til brukeren. Deler av ruten kan settes opp til å være variabel, som vil si at de delene kan være hva som helst. Kontrollerfunksjonen leser denne variabelen og bruker den til å velge hvilken HTML-side den skal sende til brukeren. I URL eksempelet over, kunne kontrollerfunksjonen brukt “ohms-lov” til å finne eksperimentsiden til Ohms Lov eksperimentet. Den samme kontrollerfunksjonen kan gjenbrukes til å returnere andre eksperimentsider ved å oppgi navnet på et annet eksperiment i ruten.

Webserveren genererer noen av HTML-sidene dynamisk basert på variabler i ruten, data fra database og data fra konfigurasjonsfilene til eksperimentene. Vi har laget templates som PUG bruker til å generere HTML-sider basert på dataen den får fra kontrollerfunksjonen. Noen ruter har statiske HTML-sider, som for eksempel registering-av-ny-bruker-siden.

6.7.1 Katalog

Nettsiden har en dynamisk generert webside med en katalog over alle tilgjengelige eksperimenter. Katalogen er delt opp i kategorier, med en side per kategori.

For å integrere et eksperiment i systemet, legger eksperimentutvikleren til en konfigurasjonsfil for eksperimentet sitt på serveren gjennom utviklersiden. På utviklersiden kan utvikler definere hvilken kategori eksperimentet hører til. Denne blir føyet til konfigurasjonsfilen.

Ruten "/catalog/category", hvor category er en variabel, kaller en kontrollerfunksjon som henter alle konfigurasjonsfilene, og leser navn, id og kategori fra hver fil. Hvis kategorien oppgitt i ruten er lik kategorien i konfigurasjonsfilen, blir den sendt til PUG. Kontrollerfunkjsonen godtar bare kategoriene electrical, mechanical, physics, mathematics, biology, optometry, uncategorized og all.

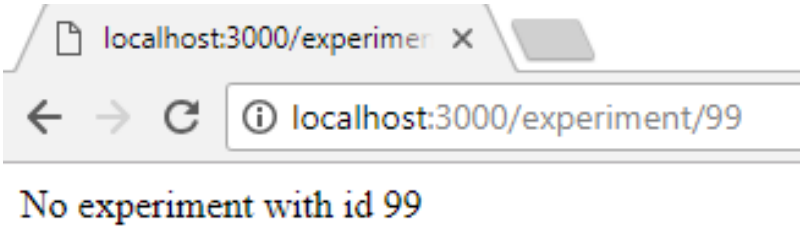
Dataene sendes til PUG som bruker catalog templaten til å lage en liste med alle kategoriene, og en liste med alle eksperimentene som tilhører kategorien. En hyperlink genereres til hvert eksperiment, som linker til eksperimentsiden til eksperimentet.

6.7.2 Automatisk generering av eksperimentbrukergrensesnitt

En måte eksperimentutvikler kan integrere sitt eksperiment med systemet vårt på er å lage en konfigurasjonsfil som følger konfigurasjonsfilformatet, og laste den opp på webserveren via utviklersiden. Webserveren bruker konfigurasjonsfilen til å dynamisk generere et brukergrensesnitt til eksperimentet.

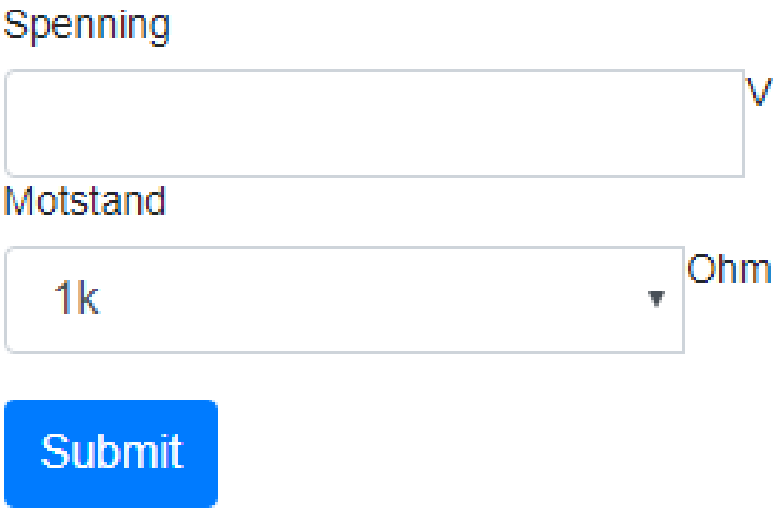
Ruten "/experiment/id", hvor id er en variabel, kaller en kontrollerfunksjon som bruker id til å finne konfigurasjonsfilen til eksperimentet og genererer et enkelt brukergrensesnitt som sendes til brukeren. Hvis konfigurasjonsfilen ikke finnes har brukeren prøvd å gå til et eksperiment som ikke finnes, og mottar en HTML-side som informerer om dette.

Konfigurasjonsfilen beskriver input- og outputparameterne til eksperimentet. Informasjon om hvilken type data parameteren skal ha, hva enheten til parameteren er, og navnet på parameteren hentes fra hver parameter. Denne dataen sendes til PUG som bruker experiment templaten til å generere inputfelter med riktig HTML inputtype, for eksempel tekstfelt, sjekkboks eller dropdown meny, og legger til parameternavn og -enhet. PUG



Figur 24: Brukeren har prøvd å gå til eksperiment 99, som ikke finnes

sender dataen om outputparametere til scriptene inkludert i HTML-siden.



Figur 25: Eksempel inputfelter

En feil kan oppstå mens genereringen pågår. Da mottar brukeren en HTML-side som informerer om at noe gikk galt og at de burde prøve å laste inn siden på nytt. Hvis feilen er så fatal at webserveren kræsjer, vil brukeren motta en 404 melding.

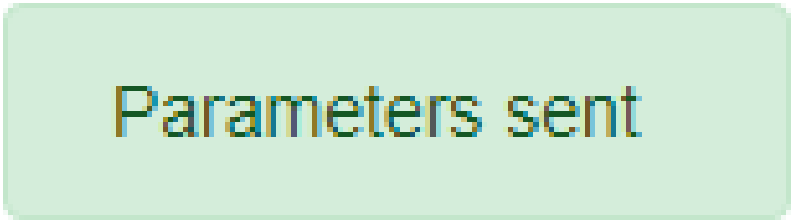
6.7.3 Sende inputverdier

Når brukere har mottat et brukergrensesnitt til et eksperiment, kan de taste inn verdier i inputfeltene og trykke “Submit” for å sende verdiene og sette igang en kjøring av eksperimentet.

Brukergrensesnittet inkluderer et Javascript kalt “socket.io-client” som setter opp en WebSocket-tilkobling til webserveren. Et annet Javascript inkludert i websiden er “submitForm”, som henter ut verdiene i inputfeltene når brukeren trykker på “Submit” og sender de til webserveren via socket.io uten å laste inn siden på nytt.

Siden gir brukeren beskjed om at verdiene er sendt. Verdiene i inputfeltene blir stående

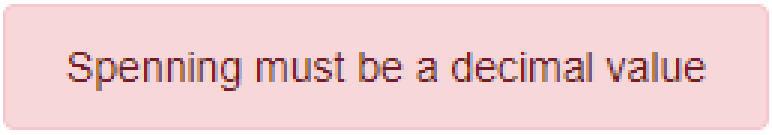
slik at brukeren husker hvilke inputverdier han sendte.



Figur 26: Brukeren får beskjed om at parameterene er sendt

På webserveren blir inputverdiene mottat. En kontrollerfunksjon som håndterer validering av inputverdier og skriver dem til en fil blir kalt.

For å validere inputverdiene, sammenligner den inputverdiene med inputparamterene i konfigurasjonsfilen til eksperimentet. Hvis det ikke finnes en inputparameter som passer til en inputverdi, eller det er færre inputverdier enn inputparametere, har det skjedd noe feil og brukeren får beskjed om dette.



Figur 27: Brukeren får beskjed om at parameteren Spenning må være et desimal tall, og kan ikke være noe annet

Feilen kan være at brukeren har gjort endringer i scriptene hos seg, og klart å sende inputverdier annet enn det inputfeltene tillater. Valideringen tar hånd om dette, slik at det ikke går ann å kjøre et eksperiment med andre parametere enn de som er definert i konfigurasjonsfilen. En annen årsak til feil kan være at dataen har blitt overført feil. Dette oppdages også.

Hvis parameteren har maksimum og minimum verdier oppgitt i konfigurasjonsfilen, blir det sjekket om inputverdien er innenfor disse. Hvis en inputverdi har verdier utenfor grenseverdiene, sendes en beskjed om dette til brukeren. Brukeren får beskjed om hvilke inputverdier som var feil, og hvorfor de var feil.

Hvis inputverdiene er gyldige, blir de skrevet til en JSON fil med navn og verdi, i inputmappen til eksperimentet. Filnavnet blir et løpenummer for kjøringer av dette eksperimentet. Køsystemet skal oppdage filen, og legge den til i køen for kjøring av eksperimenter.

```
[
{ "name":"Spenning", "value":"6" },
{ "name":"Motstand", "value":"2k"}
]
```

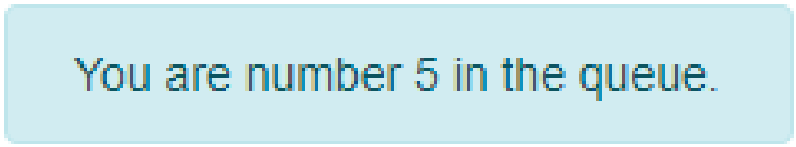
Figur 28: Eksempel inputfil

6.7.4 Køplass rapportering

Flere brukere kan sette igang kjøring av samme eksperiment samtidig. Køsystemet håndterer at de blir kjørt en om gangen(ref køsystem). Webserveren anskaffer informasjon om hvilken plass i køen brukeren har fått når brukeren har satt igang kjøring av et eksperiment og rapporterer dette til brukeren.

Når webserveren har skrevet en inputfil, teller den antall filer i inputmappen, og antar at det er plassen i køen brukeren har fått. Tallet blir sendt til brukeren via socket.io.

Brukergrensesnittet inkluderer et Javascript kalt “reportQueueSpot” som tar imot køplasseringen fra webserveren og viser det til brukeren.



Figur 29: Brukeren får beskjed om sin plass i køen

Når en inputfil blir flyttet til runningmappen til et eksperiment, sender webserveren en oppdatert køplassering til brukeren. Websiden blir oppdatert med den nye køplasseringen.

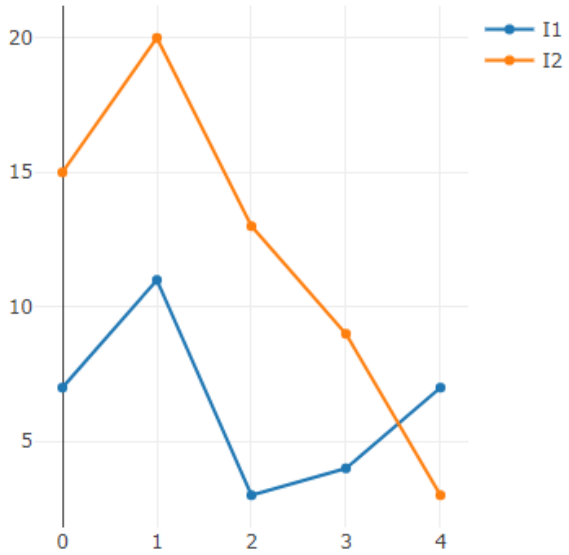
6.7.5 Motta resultater

Når en bruker har brukt brukergrensesnittet til å sette igang kjøring av et eksperiment, venter webserveren på resultater fra eksperimenter, og sender dem til brukeren via socket.io tilkoblingen når de kommer. Brukergrensesnittet tar imot resultatene og viser dem til brukeren.

Når en inputfil blir flyttet fra inputmappen til runningmappen til et eksperiment, antar webserveren at neste fil som dukker opp i resultatmappen til eksperimentet, er resultatet som tilhører den inputfilen. Dataen i resultatfilen er på JSON format, og blir sendt til

```
[
{ "name":"Strøm", "value":"3" }
]
```

Figur 30: Eksempel resultatfil



Figur 31: Eksempel graf

```
I1
1: 7 mA
2: 11 mA
3: 3 mA
4: 4 mA
5: 7 mA

I2
1: 15 mA
2: 20 mA
3: 13 mA
4: 9 mA
5: 3 mA
```

Figur 32: Eksempel liste med resultater

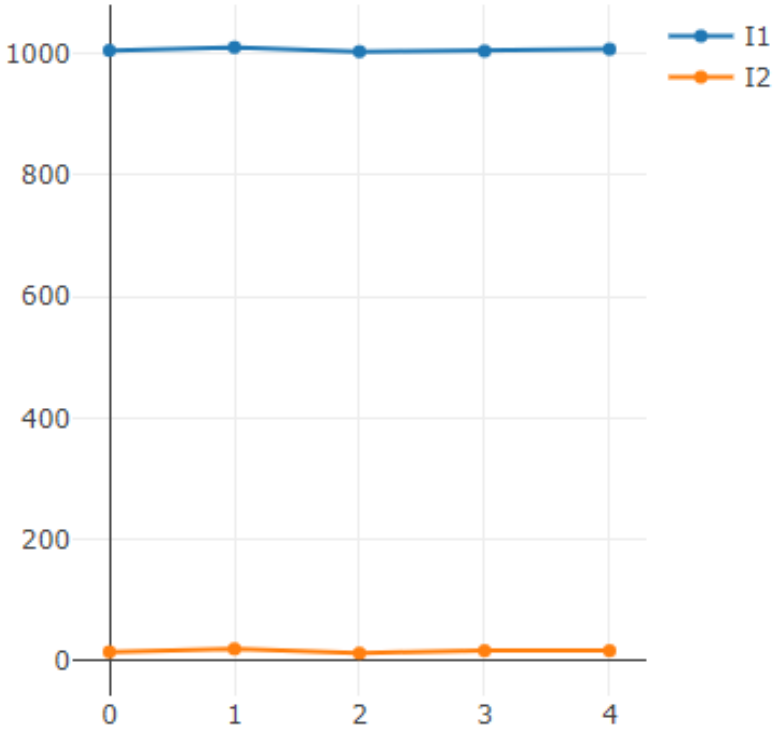
brukeren via socket.io.

Brukergrensesnittet inkluderer et Javascript kalt “showResults” som tar imot resultatdata fra webserveren og viser den til brukeren. Den lager en liste over alle resultatene, fyller inn all data per resultat og legger listen til websiden. Den bruker Plotly.js til å lage en graf med en kurve per resultat.

6.7.6 Tilpasse grafer

Brukere kan tilpasse grafen med resultater på flere måter. Plotly støtter å ha flere datasett i samme koordinatsystem, og å gjemme/vise datasettene. Den støtter å endre visningsområdet, ved å zoome eller dra, og den viser tallverdien til kurven der brukeren holder musepekeren.

Når resultatgrafen blir tegnet, legger “showResults” til to knapper under grafen: “Add graph” og “Remove graph”. “Add graph” knappen lager en kopi av grafen. Brukeren kan lage så mange kopier han vil. “Remove graph” knappen fjerner den nyeste kopien av grafen. Den kan ikke fjerne den orginale grafen.



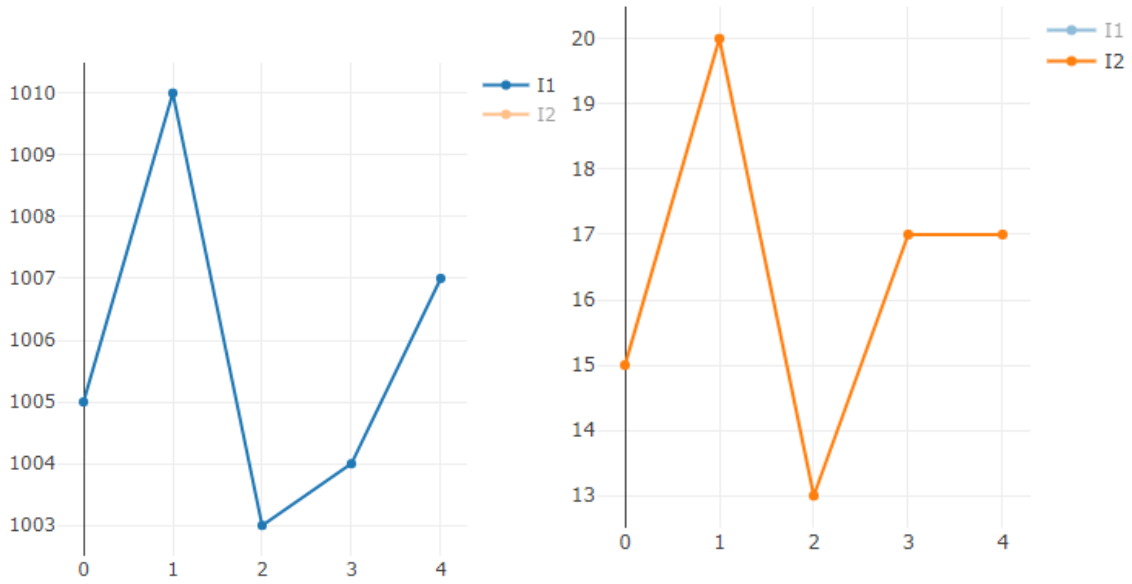
Figur 33: En graf med to kurver som ser flate ut fordi det er stor forskjell i verdiene

Dette er for at brukeren skal kunne ha forskjellig skala på aksene på grafene når han sammenligner grafer. Det er nyttig hvis resultatene er en kurve med verdier som ligger mellom 1000 og 1010, og en kurve som med verdier som ligger mellom 0 og 10. For å ha begge i visningsområdet til grafen, må man zoome ut så langt at begge kurvene ser flate ut. Hvis man har en graf som viser den ene kurven, og en graf som viser den andre kurven, kan begge være zoomet inn nok til at man kan se og sammenligne svingningene i kruvene.

6.7.7 Motta video

Eksperimenter kan ha et kamera som filmer at eksperimentet blir kjørt. Kameraet må strømme videoen til en PC, som strømmer videoen videre til en kanal på Twitch.tv. Hvis denne kanalen er på og strømmer video, blir strømmen inkludert på eksperimentsiden. Hvis kanalen ikke er på, blir videospilleren gjemt.

Denne implementasjonen strømmer direkte video av eksperimentet hele tiden, og alle kan se videoen, enten på eksperimentsiden, eller på twitch kanalen på Twitch.tv.



Figur 34: Eksempel liste med resultater

Figur 35: Kurvene i figur 33 når man zoomer inn

6.7.8 Utviklersiden

Ruten /development kaller en kontrollerfunksjon som leser alle konfigurasjonsfilene som utvikleren har lagt til i systemet, og lager en liste med navn og ID fra hver konfigurasjonsfil, som blir sendt til PUG. PUG bruker development template til å generere en dropdown meny som inneholder navnene på eksperimentene. Development template har felt for å laste opp fil, sette igang autodiscover prosessen, redigere eller slette eksisterende konfigurasjonsfiler ved å velge filen fra dropdown meny og skrive konfigurasjonsfil i websiden.

Utvikler har tre muligheter til å integrere sitt eksperiment fra utviklersiden: lage konfigurasjonsfil selv og laste opp, bruke autodiscover funksjonen, eller skrive en konfigurasjonsfil i nettsiden.

Eksperimentutvikler kan følge konfigurasjonsformatet og lage en konfigurasjonsfil selv. Denne filen kan eksperimentutvikler laste opp til webserveren.

Utvikler kan bruke autodiscover funksjonen, ved å legge inn IP addressen og porten til eksperimentet og sende disse til webserveren. Webserveren lager en fil i discovermappen som inneholder IP addressen og porten, som Core bruker til å send en discovery request pakke til IP addressen. Hvis eksperimentkontrolleren er satt opp til å støtte autodiscover, mottar Core en discovery response pakke med konfigurasjonsdata som blir skrevet til en

```
...
description: "This experiment does this and that.",
category: "Mathematics"
imagePath: "/pictures/experimentImage10.png"
...
```

Figur 36: Utdrag fra en konfigruasjonsfil med beskrivelse, kategori og bilde

fil i config-in mappen, som websiden oppdager og leser.

Den siste muligheten utvikler har er å velge Create Config Online. Utvikleren blir tatt til ruten /development/new, som inneholder en tekstboks hvor utvikleren kan skrive konfigurasjonsfilen og sende den til webserveren.

Unansett hvilken metode utvikler velger, blir en konfigurasjonsfil mottat på webserveren. Webserveren genererer en unik ID til eksperimentet og legger denne til i filen. Filen blir tilslutt lagret i konfigurasjonsmappen med ID-en som filnavn.

Når filen er lagret, blir utvikler tatt til ruten /development/setup/id, hvor id er ID-en konfigurasjonsfilen fikk av webserveren. Denne siden har felt for beskrivelse av eksperimentet, valg av kategori og mulighet til å laste opp et bilde. Eksperimentutvikler kan velge å legge til en eller flere av disse. Når han trykker “Add Info”, blir dataen og bildet sendt til webserveren.

Webserveren legger bilde i public mappen, slik at websidene kan få tak i det, og gir det filnavn “experimentImageID” hvor ID er ID-en konfigurasjonsfilen har fått. Felter for beskrivelse, path til bildet, og kategori blir lagt til i konfigurasjonsfilen med dataen fra utvikler.

På utviklersiden kan utvikler også redigere en konfigurasjonsfil han har lagt til ved å velge eksperimentnavnet fra dropdown menyen og trykke “Edit”. Dette tar utvikler til /development/edit?experimentId=id, hvor id er ID-en til eksperimentet utvikler valgte. Editsiden inneholder en tekstboks som blir fylt med innholdet i konfigurasjonsfilen. Utvikler kan gjøre endringer og trykke “Submit” for å lagre endringene.

Utvikler kan slette konfigurasjonsfiler for å fjerne et eksperiment fra systemet. Han velger eksperimentnavnet i dropdown menyen og trykker “Delete”. Konfigurasjonsfilen blir slettet fra webserveren.

6.8 Logg av kjørte eksperimenter

Når et eksperiment blir kjørt, får kjøringen en ID av webserveren. Navnet på inputfilen og resultatfilen blir denne ID-en. ID-en blir også lagret i en database sammen med ID-en til brukeren.

På brukerprofilsiden blir alle ID-ene til kjøringer brukeren har gjort hentet ut av databasen. Webserveren finner alle input- og resultatfilene og sorterer dem i en liste basert på eksperimentmappen de ligger i.

Denne listen blir sendt til PUG som bruker userprofile templaten til å generere en liste over alle kjøringene brukeren har gjort, per eksperiment. Input- og resultatfilene blir gjort tilgjengelig for nedlastning.

7 Evaluering

7.1 Testing av kode

I kapittel **??** har vi skrevet at vi skal teste all kode, først med unit tester, så med integrasjonstester. Dette har ikke blitt noe av, fordi vi ikke har hatt tid til å gjennomføre slike tester. Mye av koden ble skrevet i innspurten mot innleveringen, hvor vi også har jobbet mye med dokumentasjon.

Vi har gått rett til å utføre godkjenningstester, fordi vi mener at disse er viktigere. Unit tester og integrasjonstester tester funksjoner som er en del av godkjenningstestene, og hvis en funksjon ikke fungerer vil ikke godkjenningstesten bli godkjent. Problemet er at hvis en funksjon ikke fungerer, blir det vanskelig å finne hvilken av funksjonene som feilet.

Vi har heller ikke dokumentert tester som ikke ble godkjent. Det vi heller har gjort er å rette feilen med en gang, og utført testen på nytt, og dokumentert bare den som ble godkjent.

7.2 Arbeidsprosessen

Vi synes prosessen var vanskelig å sette seg inn i, vi slet med å velge den rette prosessen. Vi valgte først UP og brukte den frem til noen uker før 2. presentasjon. Vi følte at vi brukte nesten all tiden på å lure på om det vi planla å gjøre var i tråd med UP. En del av problemet er at vi hadde lite erfaring med å jobbe med en struktert prosessmodell.

Etter total enighet i gruppen bestemte vi oss for å bytte til Scrum. Dette kostes oss tid, fordi vi måtte sette oss inn i en ny modell, men etter en ukes innkjøringsperiode følte vi at vi prosjektet gikk fremover.

Scrum gikk bedre fordi Scrum krever færre prosessdokumenter. Vi kunne planlegge bare det vi skal holde på med i næremeste fremtid på det detaljnivå vi følte vi trengte. Fordi sprintene var en uke lange og i hver sprint defineres et mål, følte vi at det var lettere å lage oppfølgingsdokumenter hver uke.

Å velge en modell vi aldri har jobbet med krever en del tid og energi å sette seg inn i,

ikke bare av en, men alle. Når alle ikke har like mye kunnskap om en prosess oppstår det forvirringer og det er vanskelig for alle å forstå alle delleddene i prosessen.

Hadde vi skulle gjort prosjektet på nytt hadde vi cowboy kodet fra starten, og dokumentert minimalt med prosess, og heller tilbake dokumentert alt i mai.

7.3 Gruppesamarbeid

Programmeringsmessing er det store forskjeller i kunnskapnivå hos medlemmene i gruppen. Dette har ført til diskusjoner og forvirringer rundt designet av systemt, hvor gruppeleder har sett for seg hele systemet til en viss grad, mens resten av gruppen har kanskje sett for seg en del. Gjennom disse diskusjonen kommer gruppen tilslutt frem til en enighet.

Vi har aldri jobbbet med et prosjekt på denne størrelsen eller programmert et så stort system. Vi har måtte vidreutvikle kunnskapsnivået vårt i forhold til programmeringsspråk. Noen gruppemedlemmer lærer fort, andre lærer tregere, mens andre har allerede nok kunnskap.

Vi har pair programmert, og hvis noen på gruppen trenger hjelp eller tips får de det. Alle på gruppen har vært sammen på gruppe i andre prosjekter. Vi kjenner hverandre fra før og vet hverandres styrker.

I påsken brukte noen mye tid på eksamen vi hadde, mens andre brukte tiden på prosjektet i stedet. I ukene før eksamen hadde gruppen vært passive rundt design av prototype til 2. presentasjon. Det ble derfor i perioden før eksamen gjort store endringer på systemet (laget prototype ferdig) som gjorde at det ble vanskelig for resten av gruppen å ta igjen.

7.4 Har vi oppnådd målet

Vi har flere krav som vi ikke har oppnådd:

Krav 5 er ikke mulig for oss å gjennomføre. Det er en del krav som må oppfylles for at vi skal få lov til å bruke feide innlogging. Det er lagt til rette for at USN kan implementere feide dersom de tar i bruk nettsiden.

Krav 18 har ikke blitt implementert fordi vi ikke klarte å finne et eksperiment hvor det ville gitt mening å presentere resultatene som et kakediagram.

Krav 21, 22, 23, 24 og 40 har ikke blitt oppfylt. Å ta imot en strøm av video, synkronisere den, lagre den, og vise den til en bruker er teknisk krevende og vi har ikke hatt nok tid til å sette oss inn i det. Vi har implementert at vi kan vise live video fra et eksperiment.

Krav 42 har ikke blitt oppfylt. Vi har ikke lagt til funksjonalitet for utvikler til å laste opp en video som forklarer eller viser hva eksperimentet er.

Krav 46 er ikke implementert. Systemet kan generere en eksperimentside basert på konfigurasjonsfilen, men utvikler kan ikke overstyre denne genereringen ved å lage sin egen HTML fil og laste opp på serveren.

Krav 49 er ikke implementert. Vi har ikke implementert noen administrator verktøy.

Krav 53 er ikke oppfylt. Vi har ikke implementert noen analyseverktøy.

Vi har designet en brukervennlig og pen nettsiden, og dette var ikke et krav til systemet.

7.5 Samarbeid med de andre gruppene

Vi har sammarbeidet med to masterstudenter, hvor den ene skulle designe en brukervennlig nettside og analysere brukeradferd, og den andre skulle lage en eksperimentkontroller på en FPGA. Vi har også samarbeidet med gruppe 16 som skal lage moduler som kan brukes til å lage eksperimenter.

Vi hadde møte med alle gruppene som samarbeider rett etter 1. presentasjon. Det ble avtalt oppfølgingsmøter hver andre uke - dette ble aldri gjennomført.

Masterstudenten som skulle designe nettsiden flyttet inn på bachelorrommet for å få en tettere kommunikasjon med oss. Vi har også hatt kontinuerlig kommunikasjon med gruppe 16, fordi vi deler rom med dem.

Å dele rom har vært en utfordring i forhold til høyt støynivå og forstyrrelser. Det har hendt at vi har reservert grupperom eller jobber hjemme for å få fred og ro. Det har blitt kontinuerlig avbrytelser og spørsmål fra elektrostudentene i gruppe 16 angående programmering.

Begge masterstudentene har droppet ut av prosjektet. FPGA masterstudenten har vi ikke samarbeidet med i det hele tatt. Vi skulle ha samarbeidet om å lage et grensesnitt mellom FPGAen og microtjenesten. Vi har forsøkt å bruke masterstudentens nettsidedesign så langt det lar seg gjøre.

Samarbeidet mellom vår gruppe og gruppe 16 har foregått via gruppelederene. Vi skulle samarbeide om å lage et grensesnitt mellom våre prosjekter. På grunn av manglende kompetanse hos elektrostudentene i gruppe 16 har gruppen vår designet grensesnittet og protokollen, og den andre gruppen har bare måttet godtatt det vi har laget.

Fordeler med at vi har delt rom med gruppe 16 er at vi har fått erfaring i å være en del av et enda større prosjekt enn vårt eget. KOMMER MER HER

Hvis gruppe 16 mislykkes vil det påvirke vår gruppe ved at vi ikke får vist helheten i produktet. Vi har løst denne avhengigheten ved å lage vårt eget eksperiment.

7.6 Utfordringer

I prosjektet har vi brukt C++, Node og Javascript. Vi har brukt C++ litt før, men Node og Javascript var nye språk for oss. Vi har brukt Git som versjonskontroll, og skrevet dokumentasjonen i LaTeX. Utfordringen har vært at vi har brukt mye tid på å lære oss disse språkene og verktøyene. Vi kunne skrevet dokumentasjonen i Microsoft Office Word eller Google Drive Documents.

Vi har ingen erfaring med å jobbe med prosjekter av denne størrelsen, noe som gjorde at vi brukte lang tid på å planlegge.

Å intergrere delsystemene som forskjellige gruppemedlemmer har laget, skaper utfordringer når medlemmene har forskjellig programmeringsferdigheter. Delsystemet laget av et medlem med gode ferdigheter bruker smarte løsninger, som delsystemet til et medlem med svakere ferdigheter kanskje ikke støtter.

De fleste tilgjengelige Remote Labs bruker LabView som brukergrensesnitt og for å opprette kommunikasjon med eksperimentene. Det finnes lite stoff om hvordan man bygger en Remote Lab fra bunnen av.

Vi har brukt Git til versjonskontroll for delsystemene og for dokumentasjon. Det har opp-

stått utfordringer i forhold til å jobbe med samme fil samtidig. Når vi laster opp endringer, får de andre som har jobbet med filen konflikter som vi ikke har klart å løse.

En utfordring har vært at oppdragsgiver ikke var tilgjenglig på 2. presentasjon. Vi fikk ikke vist hvor langt vi hadde kommet, og fikk ikke tilbakemelding fra oppdragsgive. På grunn av dette, vet vi ikke hva han synes om produktet vårt.

8 Konklusjon

Valget av prosjektmodell var vanskelig, og vi startet med Unified Process. Denne modellen var for omfattende for gruppen. Det ble brukt mye tid på å sette seg inn i processen, og vi følte at vi aldri fikk noe ut av den, og heller ingen framgang på systemet. Derfor ble valget at vi droppet UP, og gikk over til Scrum, noe som ble positivt mottatt av ekstern sensor.

Scrum fungerte bedre for gruppen, selv om vi måtte bruke en ekstra uke midt i prosjektet på å tilrettelegge scrum. Vi fikk raskt framgang, både på prosess og system.

Systemet startet litt trått, og vi brukte tid på å forstå oppgaven. Dette sammen med en feilvalgt prosjektmodell, hindret oss i å komme så langt som vi hadde ønsket med systemet de første månedene. Når det endelig løsnet jobbet vi ”overtidde resterende ukene frem mot prosjektets slutt, og vi så en fremragende progresjon på systemet vårt. Det gjenstår fortsatt uoppfylte krav, og underveis har vi oppdaget flere features vi annser som både nødvendig og nice to have”ved videre arbeid med systemet.

Gruppen startet sløvt, noe som kanskje hang sammen med mangel på forståelse av oppgave, og frustrasjon over en prosjektmodell ingen hadde jobbet med før. Forskjellen på kunnskap og kodeferdigheter, gjorde at flere på gruppen trengte tid til å tilegne seg kunnskap, mens andre jobbet gradvis mot målet. Videre i prosjektet ble alle satt i aktivitet med noe teknisk arbeid, og vi jobbet mot et felles mål.

9 Referanser

[1] Hibu nå USN. *Remote laboratories for Department of Technology - Buskerud University College*. URL: http://rlab.hibu.no/.

[2] *PocketLab*. URL: https://www.thepocketlab.com/.

[3] Kenneth S. Rubin. *Essential Scrum: A Practical Guide to the Most Popular Agile Process (Addison-Wesley Signature Series (Cohn))*. Addison-Wesley Professional, 2012. ISBN: 0137043295. URL: https://www.amazon.com/Essential-Scrum-Practical-Addison-Wesley-Signature/dp/0137043295?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0137043295.

[4] *Avoid, Accept, or Transfer? Understanding and Managing Risk | News | Cowan Insurance Group*. URL: https://www.cowangroup.ca/en/news/article/avoid-accept-or-transfer-understanding-and-managing-risk/.

[5] Philippe Kruchten. *The Rational Unified Process An Introduction Third Edition*. Addison-Wesley Professional, 2003. Kap. 12. ISBN: 0321197704.

[6] International Software Testing Qualifications Board. *Microsoft Word - ISTQB_CTFL_Syll 2011.docx*. 2011. URL: https://www.istqb.org/downloads/send/2-foundation-level-documents/3-foundation-level-syllabus-2011.html%20.

[7] *Unit testing - Wikipedia*. URL: https://en.wikipedia.org/wiki/Unit_testing.

[8] *Node.js*. URL: https://nodejs.org/en/.

[9] *About | Node.js*. URL: https://nodejs.org/en/about/.

[10] *Express/Node introduction - Learn web development | MDN*. URL: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction.

[11] *Express/Node introduction - Learn web development | MDN*. URL: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction%20#Is_Express_opinionated.

[12] *GitHub - expressjs/express: Fast, unopinionated, minimalist web framework for node.* URL: https://github.com/expressjs/express#features.

[13] *GitHub - pugjs/pug: Pug – robust, elegant, feature rich template engine for Node.js*. URL: https://github.com/pugjs/pug.

[14] *PHP: What is PHP? - Manual*. URL: http://php.net/manual/en/intro-whatis.php.

[15] *File System | Node.js v10.1.0 Documentation*. URL: https://nodejs.org/api/fs.html#fs_file_system.

[16] *GitHub - paulmillr/chokidar: A neat wrapper around node.js fs.watch / fs.watchFile / fsevents.* URL: https://github.com/paulmillr/chokidar.

[17] *Issues · paulmillr/chokidar · GitHub*. URL: https://github.com/paulmillr/chokidar/issues.

[18] *File System | Node.js v10.1.0 Documentation*. URL: https://nodejs.org/api/fs.html#fs_fs_watchfile_filename_options_listener.

[19] *File System | Node.js v10.1.0 Documentation*. URL: https://nodejs.org/api/fs.html#fs_fs_watch_filename_options_listener.

[20] *filewatcher - npm*. URL: https://www.npmjs.com/package/filewatcher.

[21] *Introducing Node Sentinel File Watcher*. URL: https://blog.axosoft.com/nsfw/.

[22] *Issues · Axosoft/nsfw · GitHub*. URL: https://github.com/Axosoft/nsfw/issues.

[23] *Ajax - Developer guides | MDN*. URL: https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX.

[24] *IBM Knowledge Center - What is Ajax?* URL: https://www.ibm.com/support/knowledgecenter/SSRTLW_8.5.1/com.ibm.etools.webtoolscore.doc/topics/cajax.html.

[25] *JSON Introduction*. URL: https://www.w3schools.com/js/js_json_intro.asp.

[26] *RFC 4180 - Common Format and MIME Type for Comma-Separated Values (CSV) Files*. URL: https://tools.ietf.org/html/rfc4180.

[27] *GitHub - express-validator/express-validator: An express.js middleware for node-validator.* URL: https://github.com/express-validator/express-validator.

[28] *GitHub - jaredhanson/connect-flash: Flash message middleware for Connect and Express.* URL: https://github.com/jaredhanson/connect-flash.

[29] *GitHub - expressjs/express-messages: Express flash notification message rendering*. URL: https://github.com/expressjs/express-messages.

[30] *Passport.js*. URL: http://www.passportjs.org/.

[31] *passport - npm*. URL: https://www.npmjs.com/package/passport.

[32] *bcryptjs - npm*. URL: https://www.npmjs.com/package/bcryptjs.

[33] *Install MongoDB Community Edition on Windows — MongoDB Manual 3.6*. URL: https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/.

[34] *cookie-session - npm*. URL: https://www.npmjs.com/package/cookie-session.

[35] *About JavaScript - JavaScript | MDN*. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript.

[36] *Standard ECMA-262, 8th Edition / June 2017, ECMAscript 2017 Language Spesification*. URL: https://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf.

[37] *What is JavaScript? - Learn web development | MDN*. URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript.

[38] *HTML 5.2: 1. Introduction*. URL: https://www.w3.org/TR/html5/introduction.html#introduction.

[39] *Cascading Style Sheets*. URL: https://www.w3.org/Style/CSS/.

[40] *Bootstrap Get Started*. URL: https://www.w3schools.com/bootstrap/bootstrap_get_started.asp.

[41] *GitHub - plotly/plotly.js: The open source JavaScript graphing library that powers plotly*. URL: https://github.com/plotly/plotly.js/.

[42] *Home · d3/d3 Wiki · GitHub*. URL: https://github.com/d3/d3/wiki.

[43] *D3.js - Data-Driven Documents*. URL: https://d3js.org/#introduction.

[44] *Using Google Charts | Charts | Google Developers*. URL: https://google-developers.appspot.com/chart/interactive/docs/.

[45] *Bower — a package manager for the web*. URL: https://bower.io/.

[46] *pug-bootstrap - npm*. URL: https://www.npmjs.com/package/pug-bootstrap.

[47] *Font Awesome*. URL: https://fontawesome.com/.

[48] *Triage*. URL: https://en.wikipedia.org/wiki/Triage.

[49] Robert C. Martin. *Design Principles and Design Patterns*. 2000. URL: https://web.archive.org/web/20150906155800/http://www.objectmentor.com/resources/articles/Principles_and_Patterns.pdf.

[50] *Transmission Control Protocol - Wikipedia*. URL: https://en.wikipedia.org/wiki/Transmission_Control_Protocol.

[51] *UDP – Wikipedia*. URL: https://no.wikipedia.org/wiki/UDP.

[52] *HSN_designmanual.pdf*. URL: https://www.usn.no/getfile.php/13439395/usn.no/filer/om_USN/Logo%20og%20grafiske%20elementer/HSN_designmanual.pdf.

[53] *What is Mobile First Design? Why It's Important & How To Make It?* URL: https://www.mockplus.com/blog/post/mobile-first-design.

[54] *Cowboy Coding*. Mai 2018. URL: https://en.wikipedia.org/wiki/Cowboy_coding.

[55] Per Kroll og Philippe Kruchten. *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP: A Practitioner's Guide to the RUP*. Addison-Wesley Professional, 2003. ISBN: 0321166094. URL: https://www.amazon.com/Rational-Unified-Process-Made-Easy/dp/0321166094?SubscriptionId=0JYN1NVW651KCA56C1&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0321166094.

A Prosess utfordringer

Gruppen har jobbet sammen med flere prosjekter tidligere, ofte med gode resultater. Siden mange av prosjektene har inneholdt språk, utviklingsverktøy og rammeverk som har vært nye og ukjente for de fleste i gruppen, har det vært vanskelig å lage gode planer på grunn av for mange ukjente faktorer.

Det har derfor vært jobbet etter det som best kan beskrives som “Cowboy Coding”[54]. Under Cowboy Coding har alle utviklerne frihet til å angripe problemene som de vil ved hjelp av egenlæring og prøving og feiling. En av de store fordelene med metodologien er den aktive læringen og at man ikke kaster bort tid på planlegging basert på - i beste fall - kvalifisert gjetning.

A.1 Hvorfor vi valgte Unified Process først

Cowboy Coding kan knapt betegnes som en prosjektmodell, og siden en stor del av vurderingen av bacheloroppgaven baseres på prosessen måtte vi finne en bedre egnet prosjektmodell.

Gruppen undersøkte en del modeller, men informasjon kommer ofte fra evangelister for sine respektive modeller. Dette gjør det vanskelig å gjøre en objektiv vurdering av de uten å prøve de ut i praksis. Vi kom også fram til at de fleste anerkjente modeller ville fungere uansett, og siden vi ikke hadde spesielt mye erfaring eller kunnskap om én bestemt modell, foreslo gruppeleder for gruppe 4 (software) at begge gruppene (software og hardware) brukte Scrum.

Da gruppeleder presenterte denne idéen for oppdragsgiver, mente han at Unified Process ville være et bedre valg. Siden vi så for oss at Unified Process ville gjøre det enklere å unngå å falle tilbake på Cowboy Coding, tok vi dette rådet til følge og valgte Unified Process. Og på grunn av god tilgang på litteratur, valgte vi Rational Unified Process.

A.2 Hvorfor Rational Unified Process ikke fungerte (for oss)

(Rational) Unified Process baserer seg på strukturen:

- Inception Phase
- Elaboration Phase
- Construction Phase
- Transition Phase

Under Inception Phase, prøver man å forstå omfanget av prosjektet og bygge en "Business Case" for å få en endelig godkjenning fra oppdragsgiver om å fortsette prosjektet.

Elaboration Phase handler om å identifisere og finne måter å redusere store tekniske risikoer og lage en "baselined architecture" som inneholder de største og viktigste delene av produktet. På denne måten får man en forståelse av hva som kreves for å bygge systemet.

Construction Phase er selve byggefasen der man lager den første fungerende versjonen av produktet mens man under Transition Phase lager den endelige versjonen av produktet og overleverer denne til kunden.[55]

Det å forsøke å følge denne modellen bød på flere problemer.

Oppdragsgiver og gruppeleder hadde allerede en intuitiv oversikt av hvordan systemet burde bygges, men både intern sensor og veileder var klare på at det var gruppen som helhet som skulle komme fram til en løsning. Derfor prøvde gruppeleder å simulere en situasjon der vi brukte Rational Unified Process metodologien til å designe en løsning gjennom innhenting av krav og andre aktiviteter som tilhører Inception og Elaboration phase.

Samtidig var det liten vilje og til og med motstand blant de øvrige gruppemedlemmene om å sette seg inn i Unified Process, og dette førte til at simuleringen mislyktes. Gruppen skjønte ikke hvorfor vi gjorde det vi gjorde, og gruppeleder hadde strengt talt ikke behov for å bruke fremgangsmåten i det hele tatt.

UP er også et kompleks prosjektmodell, og ble en belastning i et allerede uoversiktlig prosjekt med mange teknikker og teknologier som måtte læres.

A.3 Byttet til Scrum var heller ikke problemfritt

Teamet valgte å forkaste UP til fordel for Scrum fordi vi følte at det var bedre egnet til prosjektet, og UP fungerte jo dårlig uansett. Men Scrum gikk heller ikke smertefritt.

Vi endte opp med for få iterasjoner i forhold til hvor lenge det var igjen av prosjektet. Dermed fikk vi ingen naturlig innføring i modellen før presset fra innleveringsfristen satte inn for fullt.

En stor del av litteraturen overfokuserer på “User Stories”, og vi klarte ikke å knytte disse direkte mot tekniske oppgaver på serversiden av produktet. Alt gikk bedre når vi tok i bruk noe som heter “Technical Stories” som gjorde at vi stod litt friere til å jobbe med back-end.

I likhet med UP, var det en del motstand i gruppen i forhold til å omfavne prosessen. Mange aktiviteter i Scrum foregår i grupper, og krever at alle har en viss kjenskap til hvordan aktivitetene skal gjennomføres.

B Protokollpakker

Packet Type 00	Discovery request	
Felt navn	Stør. i bytes	Beskrivelse
message	5	“DISCO” (1)

Packet Type 01	Discovery response header	
Felt navn	Stør. i bytes	Beskrivelse
id	2	En unik id som refererer til hvilke eksperiment det er
name	80	Navnet på selve eksperimentet. F.eks ”Ohm’s Law”
ipAdress	6	IPadressen til eksperimentet. Kan være Ipv4 eller Ipv6 adresse
port	2	Porten eksperimentet har
channelCount	1	Antall kanaler, velges av utvikler dersom det trengs
parameterCount	2	Hvor mange parametere som sendes. Brukes for å lage et nytt parameter objekt som lagrer de inkommende parameterpakkene som etterfølger Header pakken.

Packet Type 02	Discovery response parameter	
Felt navn	Stør. i bytes	Beskrivelse
id	2	En unik id som refererer til hvilke parameter det er
direction	1	Bestemmer om parameteren er input eller output
fieldName	40	Navnet på feltet til parameteren. Mer forklart under
type	2	Hva slags type er verdien til parameteren
channelId	1	Dersom det er behov for kanaler settes det en ID på parameteren
enumId	2	Får en ID dersom parametern skal brukes som en enum i frontend
units	16	Enheten til parameteren
size	1	Størrelsen på verdien som skal sendes. F.eks en float er 4 bytes

Packet Type 03	Video sync marker	
Felt navn	Stør. i bytes	Beskrivelse
marker	1	Sender en marker som brukes til å synkronisere video

Packet Type 04	Input	
Felt navn	Stør. i bytes	Beskrivelse
value	(2)	Verdien som sendes som input til eksperimentet

Packet Type 05	Output (3)	
Felt navn	Stør. i bytes	Beskrivelse
value	(2)	Verdien som sendes som output fra resultatet til frontend

Packet Type 06	Output with channel (3)	
Felt navn	Stør. i bytes	Beskrivelse
channelId	1	channelId finner korrekt parameter som skal sendes i pakken
value	(2)	Verdien som sendes som output fra resultatet til frontend

Packet Type 07	Start marker	
Felt navn	Stør. i bytes	Beskrivelse
message	5	“START” (1)

Packet Type 08	End marker	
Felt navn	Stør. i bytes	Beskrivelse
message	4	“STOP” (1)

Packet Type 09	Service	
Felt navn	Stør. i bytes	Beskrivelse
serviceId	1	Id’n til den ønskede servicen som skal kjøres. Se eget punkt under

Packet Type 10	Experiment runId	
Felt navn	Stør. i bytes	Beskrivelse
runID	1	runID skal brukes til å navngi resultatfiler

1. Det sendes med en streng for å redusere muligheten for at pakken tolkes feil.

2. Størrelsen er avhengig av konfigurasjonsfilen til det aktuelle eksperimentet. Input og output kan sende flere value samtidig, og hver value kan ha forskjellig size, avhengig av hva som er definert.

3. Output har ikke med størrelse for å minske overhead. Dette valget er gjort for å sørge for så rask overføring av resultatdata som mulig.

C Konfigurasjonsfil format

Eksperimentutvikler lager en konfigurasjonfil for eksperimentet sitt og laster den opp på serveren for å integrere sitt eksperiment med systemet. Systemet støtter at konfigurasjonsfilen har filformat JSON.

Innholdet i filen skal være:

- id: En unik id til eksperimentet. Denne blir generet av webserveren.
- ip: IP-addressen til eksperimentet.
- port: Porten eksperimentet kommuniserer på.
- name: Navnet til eksperimentet.
- parameterCount: Hvor mange parametere eksperimentet har, både inn og ut. Husk at hver verdi i en enum er en egen parameter.
- channelCount: Hvor mange kanaler eksperimentet bruker.
- parameters: En liste med parametere som hver inneholder:
  - id: Hvilket nummer parameteren er. Starter på null og teller oppover.
  - in_out: Om parameteren er en inn eller ut parameter. Enten I eller O.
  - enumId: Alle parameterne som hører til samme enum må ha samme enumId. Alle andre parametere skal ha enumId null.
  - name: Navnet på parameteren. Hvis parameteren er en enum verdi, står verdien her.
  - parameterType: Datatypen til parameteren. Enten integer, floatingPoint, character, enum eller boolean.
  - size: Størrelsen på datatypen.
  - unit: En liste med enheter som hver inneholder:
    - symbol: Symbolet til enheten.
    - exponent: Eksponenten til enheten.

Filen kan inneholde:

- category: Navn på kategorien eksperimentet tilhører.
- description: En beskrivelse av eksperimentet, hva det gjør og eventuell annen informasjon eksperimentutvikleren mener brukerene burde ha.
- imagePath: Path til bilde eksperimentutvikler laster opp. Blir generert av webserveren når utvikler laster opp bildet.

C.1 JSON fil eksempel

```
{
    "id": 2,
    "ip": "192.168.2.1",
"port": 1337,
    "name": "Ohm's Law",
    "parameterCount": 6,
    "channelCount": 0,
    "parameters": [
        {
            "id": 0,
            "in_out": "I",
"enumId" : 0,
            "name": "Spenning",
            "parameterType": "floatingPoint",
            "size": 4,
            "unit": [
                {
                    "exponent": 1,
                    "symbol": "V"
                }
            ]
        },
        {
            "id": 1,
```

```
            "in_out": "I",
"enumId": 1,
            "name": "Motstand",
            "parameterType": "enum",
            "size": 4,
            "unit": [
                {
                    "exponent": 1,
                    "symbol": "Ohm"
                }
            ]
        },
        {
            "id":2,
            "in_out": "I",
"enumId": 1,
            "name": "1k",
            "parameterType": "floatingPoint",
            "size": 4,
            "unit": [
                {
                    "exponent": 0,
                    "symbol": ""
                }
            ]
        },
{
            "id": 3,
            "in_out": "I",
"enumId": 1,
            "name": "2k",
            "parameterType": "floatingPoint",
            "size": 4,
```

```
            "unit": [
                {
                    "exponent": 0,
                    "symbol": ""
                }
            ]
        },
{
            "id": 4,
            "in_out": "I",
"enumId": 1,
            "name": "10k",
            "parameterType": "floatingPoint",
            "size": 4,
            "unit": [
                {
                    "exponent": 0,
                    "symbol": ""
                }
            ]
        },
{
            "id": 5,
            "in_out": "O",
"enumId": 0,
            "name": "Str m",
            "parameterType": "floatingPoint",
            "size": 4,
            "unit": [
                {
                    "exponent": 1,
                    "symbol": "A"
                }
```

```
            ]
        },
    ],
    "category": ["Electrical"]
"description": "This experiment calculates Ohm's Law",
"image": "/minedirtybilder/Ohmsslaw"
}
```

D Risiko

Mangel på kompetanse hos gruppemedlemmene					
Risk ID	Sannsynlighet	Konsekvens	Risiko	Redusert	Strategi
1	5	4	14	30%	Akseptere
Kommentar: Manglende kompetanse i programmeringsspråk som Javascript, Node, Arduino. Manglende kompetanse på videobehandling (fangst, koding og lagring)					
Tiltak: Bruke tid på å sette seg inn i de nye programmeringsspråkene på forhånd.					

Samarbeid med gruppe 16					
Risk ID	Sannsynlighet	Konsekvens	Risiko	Redusert	Strategi
2	3	3	5.4	40%	Akseptere
Kommentar: Forskjell i tankesett og fagfelt. Lite overlappende kunnskap					
Tiltak: Lage vår egen demo som kan brukes til presentasjonen					

Korttidsfravær					
Risk ID	Sannsynlighet	Konsekvens	Risiko	Redusert	Strategi
3	3	2	6	0%	Akseptere
Kommentar: Sykdom, eksamen, eller andre familiære årsaker som gjør at en er vekk en dag eller to.					
Tiltak: Sørge for å ha god kommunikasjon dersom en har behov for fravær.					

Langtidsfravær					
Risk ID	Sannsynlighet	Konsekvens	Risiko	Redusert	Strategi
4	1	5	5	0%	Akseptere
Kommentar: Større sykdomsfravær, skader, eller at medstudent dropper ut av prosjektet.					
Tiltak: Ha oversikt over hva hvert gruppemedlem jobber med, slik at det er enkelt å ta over dersom det blir nødvendig.					

Tidsbruk					
Risk ID	Sannsynlighet	Konsekvens	Risiko	Redusert	Strategi
5	5	3	15	0%	Akseptere
Kommentar: Bruker lengre tid på deloppgaver i prosjektet enn det som først er planlagt.					
Tiltak:Forenklet prosessen ved å bytte fra UP til Scrum. Aksepterer at vi ikke har rutinen for å vite nøyaktig hvor mye tid hver deloppgave vil ta.					

Skade på hardware					
Risk ID	Sannsynlighet	Konsekvens	Risiko	Redusert	Strategi
6	1	4	4	0%	Akseptere
Kommentar: Skade på hardware som Arduino, eller serveren.					
Tiltak: Bruke ny Arduino. Flytte systemet vårt på ny server.					

Feil valg av software					
Risk ID	Sannsynlighet	Konsekvens	Risiko	Redusert	Strategi
7	3	2	4.8	20%	Akseptere
Kommentar: Velger feil programmeringsspråk til frontend, backend mm.					
Tiltak: Researcher og sammenligner programmeringsspråk slik at vi kan ta den beste avgjørelsen på hvilke språk som passer for vårt system.					

Tap av data					
Risk ID	Sannsynlighet	Konsekvens	Risiko	Redusert	Strategi
8	1	5	0.5	90%	Akseptere
Kommentar: Mister filer/data som omhandler prosjektdokumentasjonen eller systemet.					
Tiltak: Har backup av alle filer på Google drive, lokalt, Git og skytjenester.					

Gruppa blir uvenner					
Risk ID	Sannsynlighet	Konsekvens	Risiko	Redusert	Strategi
9	2	5	6	40%	Akseptere
Kommentar: Store uenigheter i gruppa blåses opp til krangel og sure miner.					
Tiltak: Lufte uenigheter før de blir for alvorlige. Pizzakveld.					

Serveren fungerer ikke					
Risk ID	Sannsynlighet	Konsekvens	Risiko	Redusert	Strategi
10	1	5	5	0%	Overføre
Kommentar: Servere som skal brukes til systemet vårt fungerer ikke.					
Tiltak: Risikoen overføres til skolen da de har hovedansvaret for å stille med server til gruppa.					

Manglende delkomponenter til systemet					
Risk ID	Sannsynlighet	Konsekvens	Risiko	Redusert	Strategi
11	3	5	15	0%	Akseptere
Kommentar: Gruppe 16 får ikke på plass alle delkomponentene som inneholder software.					
Tiltak: Vi hjelper gruppe 16 med software.					

E Krav

I dette vedlegget vises alle krav vi har til systemet. Tabellene som er vist i dette vedlegget er delt inn på følgende måte:

- KravID: Hvert krav er gitt en ID slik at det kan spores videre i prosessen
- Dato: Datoen kravet ble opprettet
- Kilde: Hvem kom kravet fra
  - Oppdragsgiver
  - Gruppe 16
  - Masterstudent (FPGA)
  - Masterstudent (Brukervennlighet, webside)
- Status: Hvilken status har dette kravet
  - Under arbeid: Det jobbes fortsatt med systemet for at kravet skal møtes (Merket med fargen RØD)
  - Fullført: Systemet møter kravet (Merket med fargen GRØNN)
  - Annullert: Kravet er annullert i samsvar med kilde og det kreves ikke arbeid for å imøtekomme dette kravet lenger (Merket med fargen GRÅ)
- Prioritet: Angir viktigheten av at kravet blir møtt (A er viktigst, C er minst viktig)
  - A krav er merket med fargen RØD
  - B krav er merket med fargen GUL
  - C krav er merket med fargen BLÅ
- Beskrivelse: Gir en kort beskrivelse av hva kravet innebærer
- Kravtype: Hvordan type krav er dette (Funksjonell, Ikke funksjonell)
- User Story ID: Angir hvilke(n) User Story kravet oppfyller

KravID	Dato	Kilde	Status	Prioritet
1	11.02.2018	Oppdragsgiver	Fullført	B
Beskrivelse: Brukergrensesnittet skal vise total ventetid når ventetid er lengre enn 10 sekunder				
Kravtype	Funksjonell			
User story ID	2.5			

KravID	Dato	Kilde	Status	Prioritet
2	11.02.2018	Oppdragsgiver	Fullført	B
Beskrivelse: Brukeren skal logge inn for å bruke systemet				
Kravtype	Funksjonell			
User story ID	11.2			

KravID	Dato	Kilde	Status	Prioritet
3	11.02.2018	Oppdragsgiver	Fullført	B
Beskrivelse: Innlogging bør skje via en lokal brukerdatabase				
Kravtype	Funksjonell			
User story ID	11.2			

KravID	Dato	Kilde	Status	Prioritet
4	11.02.2018	Oppdragsgiver	Fullført	C
Beskrivelse: Innlogging bør skje via 3.part				
Kravtype	Funksjonell			
User story ID	11.6			

KravID	Dato	Kilde	Status	Prioritet
5	11.02.2018	Oppdragsgiver	Under arbeid	C
Beskrivelse: Innlogging bør skje via Feide				
Kravtype	Funksjonell			
User story ID	11.6			

KravID	Dato	Kilde	Status	Prioritet
6	11.02.2018	Oppdragsgiver	Fullført	C
Beskrivelse: Innlogging bør skje via OAuth2				
Kravtype	Funksjonell			
User story ID	11.6			

KravID	Dato	Kilde	Status	Prioritet
7	11.02.2018	Oppdragsgiver	Fullført	A
Beskrivelse: Brukergrensesnittet skal ta imot parametre til eksperiment				
Kravtype	Funksjonell			
User story ID	2.3.1, 2.3.2, 11.3			

KravID	Dato	Kilde	Status	Prioritet
8	11.02.2018	Oppdragsgiver	Fullført	A
Beskrivelse: Brukergrensesnittet skal bare tillate gyldige valg				
Kravtype	Funksjonell			
User story ID	2.3.3			

KravID	Dato	Kilde	Status	Prioritet
9	11.02.2018	Oppdragsgiver	Fullført	A
Beskrivelse: Brukergrensesnittet skal varsle om ugyldige valg				
Kravtype	Funksjonell			
User story ID	2.3.3			

KravID	Dato	Kilde	Status	Prioritet
10	11.02.2018	Oppdragsgiver	Annullert	C
Beskrivelse: Brukergrensesnittet skal varsle om tvilsomme valg				
Kravtype	Funksjonell			
User story ID				

KravID	Dato	Kilde	Status	Prioritet
11	11.02.2018	Oppdragsgiver	Fullført	A
Beskrivelse: Brukergrensesnittet skal vise resultat fra et eksperiment				
Kravtype	Funksjonell			
User story ID	11.3			

KravID	Dato	Kilde	Status	Prioritet
12	11.02.2018	Oppdragsgiver	Fullført	A
Beskrivelse: Brukergrensesnittet skal vise resultater som grafer				
Kravtype	Funksjonell			
User story ID	2.4.1			

KravID	Dato	Kilde	Status	Prioritet
13	11.02.2018	Oppdragsgiver	Fullført	A
Beskrivelse: Grafer bør kunne inneholde flere datasett samtidig for sammenligning				
Kravtype	Funksjonell			
User story ID	2.4.1			

KravID	Dato	Kilde	Status	Prioritet
14	11.02.2018	Oppdragsgiver	Fullført	A
Beskrivelse: Grafer bør kunne ha forskjellige skala per datasett				
Kravtype	Funksjonell			
User story ID	2.4.1			

KravID	Dato	Kilde	Status	Prioritet
15	11.02.2018	Oppdragsgiver	Fullført	B
Beskrivelse: Skalaen til grafen bør tilpasse seg datasettet automatisk				
Kravtype	Funksjonell			
User story ID	2.4.1			

KravID	Dato	Kilde	Status	Prioritet
16	11.02.2018	Oppdragsgiver	Fullført	B
Beskrivelse: Skalaen til grafen skal kunne styres manuelt				
Kravtype	Funksjonell			
User story ID	2.4.1			

KravID	Dato	Kilde	Status	Prioritet
17	11.02.2018	Oppdragsgiver	Fullført	C
Beskrivelse: Grafer bør vise tallverdier der brukeren ønsker det				
Kravtype	Funksjonell			
User story ID	2.4.1			

KravID	Dato	Kilde	Status	Prioritet
18	11.02.2018	Oppdragsgiver	Under arbeid	C
Beskrivelse: Brukergrensesnittet bør vise resultater som kakediagrammer				
Kravtype	Funksjonell			
User story ID	2.4.3			

KravID	Dato	Kilde	Status	Prioritet
19	11.02.2018	Oppdragsgiver	Fullført	B
Beskrivelse: Brukergrensesnittet skal vise resultater som tallverdier				
Kravtype	Funksjonell			
User story ID	2.4.2			

KravID	Dato	Kilde	Status	Prioritet
20	11.02.2018	Oppdragsgiver	Fullført	C
Beskrivelse: Systemet bør tillate bruker å endre hvilke elementer i grafene som vises				
Kravtype	Funksjonell			
User story ID	2.4.1			

KravID	Dato	Kilde	Status	Prioritet
21	11.02.2018	Oppdragsgiver	Under arbeid	A
Beskrivelse: Systemet skal ta imot video fra eksperimentet				
Kravtype	Funksjonell			
User story ID	2.6.1			

KravID	Dato	Kilde	Status	Prioritet
22	11.02.2018	Oppdragsgiver	Under arbeid	A
Beskrivelse: Systemet skal synkronisere video med resultatene				
Kravtype	Funksjonell			
User story ID	2.6.3			

KravID	Dato	Kilde	Status	Prioritet
23	11.02.2018	Oppdragsgiver	Under arbeid	A
Beskrivelse: Systemet skal tillate nedlasting av video				
Kravtype	Funksjonell			
User story ID	2.6.4			

KravID	Dato	Kilde	Status	Prioritet
24	11.02.2018	Oppdragsgiver	Under arbeid	C
Beskrivelse: Systemet bør lagre video en periode				
Kravtype	Funksjonell			
User story ID	2.6.4			

KravID	Dato	Kilde	Status	Prioritet
25	11.02.2018	Oppdragsgiver	Fullført	B
Beskrivelse: Systemet skal tilgjengeliggjøre data i filformatet JSON				
Kravtype	Funksjonell			
User story ID	11.5			

KravID	Dato	Kilde	Status	Prioritet
26	11.02.2018	Oppdragsgiver	Under arbeid	A
Beskrivelse: Systemet skal lagre resultatene for brukeren i 1 år				
Kravtype	Funksjonell			
User story ID	11.5			

KravID	Dato	Kilde	Status	Prioritet
28	11.02.2018	Oppdragsgiver	Fullført	C
Beskrivelse: Systemet skal støtte globalisering				
Kravtype	Ikke-Funksjonell			
User story ID	11.8			

KravID	Dato	Kilde	Status	Prioritet
29	11.02.2018	Oppdragsgiver	Fullført	A
Beskrivelse: Brukergrensesnittet skal ha en katalog med oversikt over samtlige tilgjengelige eksperimenter				
Kravtype	Funksjonell			
User story ID	2.1, 11.4			

KravID	Dato	Kilde	Status	Prioritet
30	11.02.2018	Oppdragsgiver	Fullført	C
Beskrivelse: Brukergrensesnittet bør kunne kjøres uten installasjon				
Kravtype	Ikke-Funksjonell			
User story ID	11.3			

KravID	Dato	Kilde	Status	Prioritet
31	11.02.2018	Oppdragsgiver	Fullført	C
Beskrivelse: Brukergrensesnittet bør kunne styres via touch-skjerm				
Kravtype	Funksjonell			
User story ID	11.7			

KravID	Dato	Kilde	Status	Prioritet
32	11.02.2018	Oppdragsgiver	Fullført	A
Beskrivelse: Brukergrensesnittet skal styres vha mus og tastatur				
Kravtype	Funksjonell			
User story ID	11.7			

KravID	Dato	Kilde	Status	Prioritet
33	11.02.2018	Oppdragsgiver	Fullført	C
Beskrivelse: Brukergrensesnittet skal tilpasse seg små skjermstørrelser				
Kravtype	Funksjonell			
User story ID	11.7			

KravID	Dato	Kilde	Status	Prioritet
34	11.02.2018	Oppdragsgiver	Fullført	C
Beskrivelse: Brukergrensesnittet skal tilpasse seg store skjermstørrelser				
Kravtype	Funksjonell			
User story ID	11.7			

KravID	Dato	Kilde	Status	Prioritet
35	11.02.2018	Oppdragsgiver	Fullført	C
Beskrivelse: Systemet skal være i stand til å vise direktevideo				
Kravtype	Funksjonell			
User story ID	2.6.2			

KravID	Dato	Kilde	Status	Prioritet
36	11.02.2018	Oppdragsgiver	Fullført	A
Beskrivelse: Brukeren skal kunne se resultater direkte ved eksperimenter som har varighet lengre enn 2 timer				
Kravtype	Funksjonell			
User story ID	2.4.4			

KravID	Dato	Kilde	Status	Prioritet
37	11.02.2018	Oppdragsgiver	Fullført	A
Beskrivelse: Et eksperiment skal ta mindre enn 10 timer å legge til i systemet				
Kravtype	Ikke-Funksjonell			
User story ID	3			

KravID	Dato	Kilde	Status	Prioritet
38	11.02.2018	Oppdragsgiver	Fullført	A
Beskrivelse: Utvikler skal konfigurere parameterimport				
Kravtype	Funksjonell			
User story ID	3			

KravID	Dato	Kilde	Status	Prioritet
39	11.02.2018	Oppdragsgiver	Fullført	A
Beskrivelse: Utvikler skal konfigurere resultateksport				
Kravtype	Funksjonell			
User story ID	3			

KravID	Dato	Kilde	Status	Prioritet
40	11.02.2018	Oppdragsgiver	Under arbeid	A
Beskrivelse: Utvikler skal konfigurere videoeksport				
Kravtype	Funksjonell			
User story ID	3			

KravID	Dato	Kilde	Status	Prioritet
41	11.02.2018	Oppdragsgiver	Fullført	B
Beskrivelse: Systemet skal støtte import av kretsbilder				
Kravtype	Funksjonell			
User story ID	3			

KravID	Dato	Kilde	Status	Prioritet
42	11.02.2018	Oppdragsgiver	Under arbeid	A
Beskrivelse: Utvikler skal legge til eksperimentvideo				
Kravtype	Funksjonell			
User story ID	3			

KravID	Dato	Kilde	Status	Prioritet
43	11.02.2018	Oppdragsgiver	Fullført	A
Beskrivelse: Systemet skal støtte visning av video formatet MP4				
Kravtype	Funksjonell			
User story ID	11.3			

KravID	Dato	Kilde	Status	Prioritet
44	11.02.2018	Oppdragsgiver	Fullført	A
Beskrivelse: Utvikler skal legge til parameterfelt				
Kravtype	Funksjonell			
User story ID	3			

KravID	Dato	Kilde	Status	Prioritet
45	11.02.2018	Oppdragsgiver	Fullført	A
Beskrivelse: Utvikler skal legge til resultatfelter/ -grafer/ -diagrammer				
Kravtype	Funksjonell			
User story ID	3			

KravID	Dato	Kilde	Status	Prioritet
46	11.02.2018	Oppdragsgiver	Under Arbeid	A
Beskrivelse: Systemet skal ha mal for utbygging av ny eksperimentside				
Kravtype	Ikke-Funksjonell			
User story ID	3			

KravID	Dato	Kilde	Status	Prioritet
47	11.02.2018	Oppdragsgiver	Under arbeid	A

Beskrivelse:

Utvikler skal kunne utvide systemet med flere appservere

Kravtype	Ikke-Funksjonell
User story ID	4

KravID	Dato	Kilde	Status	Prioritet
48	11.02.2018	Oppdragsgiver	Fullført	A

Beskrivelse:

Utvikler skal lage guide (eller link/referanse til læremiddel) for brukeren til sitt eksperiment

Kravtype	Funksjonell
User story ID	2.2

KravID	Dato	Kilde	Status	Prioritet
49	11.02.2018	Oppdragsgiver	Under arbeid	A

Beskrivelse:

Systemet skal melde fra om driftsforstyrrelser

Kravtype	Funksjonell
User story ID	4

KravID	Dato	Kilde	Status	Prioritet
50	11.02.2018	Oppdragsgiver	Fullført	C
Beskrivelse: Systemet skal loggføre brukernes valg av eksperimenter				
Kravtype	Funksjonell			
User story ID	4, 9, 11.5			

KravID	Dato	Kilde	Status	Prioritet
51	11.02.2018	Oppdragsgiver	Fullført	C
Beskrivelse: Systemet skal loggføre brukernes parameterinput til et eksperiment				
Kravtype	Funksjonellt			
User story ID	4, 9, 11.5			

KravID	Dato	Kilde	Status	Prioritet
52	11.02.2018	Oppdragsgiver	Fullført	C
Beskrivelse: Systemet skal loggføre brukernes resultat fra et eksperiment				
Kravtype	Funksjonell			
User story ID	4, 9, 11.5			

KravID	Dato	Kilde	Status	Prioritet
53	11.02.2018	Oppdragsgiver	Under arbeid	C

Beskrivelse:

Systemet skal bruke loggen til å analysere en brukers adferd

Kravtype	Funksjonell
User story ID	4, 9

KravID	Dato	Kilde	Status	Prioritet
54	11.02.2018	Oppdragsgiver	Under Arbeid	A

Beskrivelse:

Opplæring av nye eksperimentutviklere skal ta mindre enn 20 timer

Kravtype	Ikke-Funksjonell
User story ID	3

KravID	Dato	Kilde	Status	Prioritet
55	11.02.2018	Oppdragsgiver	Fullført	A

Beskrivelse:

Systemet skal ha en brukermanual for eksperimentutviklere

Kravtype	Ikke-Funksjonell
User story ID	3

KravID	Dato	Kilde	Status	Prioritet
56	11.02.2018	Oppdragsgiver	Fullført	A
Beskrivelse: Målgruppen for brukermanualen skal være for instruktører/lærere				
Kravtype	Ikke-Funksjonell			
User story ID	3			

KravID	Dato	Kilde	Status	Prioritet
57	11.02.2018	Oppdragsgiver	Fullført	A
Beskrivelse: Systemet skal kjøre på en datamaskin				
Kravtype	Ikke-Funksjonell			
User story ID	4			

KravID	Dato	Kilde	Status	Prioritet
58	11.02.2018	Oppdragsgiver	Under arbeid	A
Beskrivelse: Systemet skal være sikkert				
Kravtype	Funksjonell			
User story ID	4			

KravID	Dato	Kilde	Status	Prioritet
59	11.02.2018	Oppdragsgiver	Fullført	A
Beskrivelse: Systemet skal håndtere flere brukere samtidig				
Kravtype	Funksjonell			
User story ID	12			

KravID	Dato	Kilde	Status	Prioritet
60	11.02.2018	Oppdragsgiver	Fullført	A
Beskrivelse: Systemet skal bygges med utbredt standardspråk				
Kravtype	Ikke-Funksjonell			
User story ID	4			

KravID	Dato	Kilde	Status	Prioritet
61	11.02.2018	Oppdragsgiver	Fullført	A
Beskrivelse: Systemet skal bruke internasjonale standardsymboler etc.				
Kravtype	Ikke-Funksjonell			
User story ID	11.8			

KravID	Dato	Kilde	Status	Prioritet
62	11.02.2018	Oppdragsgiver	Fullført	A

Beskrivelse:

Systemet skal sende data til eksperimentet

Kravtype	Funksjonell
User story ID	10.3

KravID	Dato	Kilde	Status	Prioritet
63	11.02.2018	Oppdragsgiver	Fullført	A

Beskrivelse:

Systemet skal motta data fra eksperimentet

Kravtype	Funksjonell
User story ID	10.3

KravID	Dato	Kilde	Status	Prioritet
65	11.02.2018	Oppdragsgiver	Under arbeid	A

Beskrivelse:

Båndbredden inn/ut av eksperimentet skal være minst 1 MB/s

Kravtype	Ikke-Funksjonell
User story ID	

KravID	Dato	Kilde	Status	Prioritet
66	11.02.2018	Oppdragsgiver	Under arbeid	A
Beskrivelse: Responstiden inn/ut av eksperimentet skal være under x antall sekunder				
Kravtype	Ikke-Funksjonell			
User story ID				

KravID	Dato	Kilde	Status	Prioritet
67	11.02.2018	Oppdragsgiver	Under arbeid	A
Beskrivelse: I/O frekvensen inn/ut av eksperimentet skal være minst x ganger/s				
Kravtype	Ikke-Funksjonell			
User story ID				

KravID	Dato	Kilde	Status	Prioritet
68	11.02.2018	Oppdragsgiver	Fullført	A
Beskrivelse: Alle deler skal bruke samme overføringsprotokoll				
Kravtype	Ikke-Funksjonell			
User story ID	10.1			

KravID	Dato	Kilde	Status	Prioritet
69	11.02.2018	Oppdragsgiver	Fullført	A

Beskrivelse:

Alle deler skal bruke samme lagringsprotokoll

Kravtype	Ikke-Funksjonell
User story ID	

KravID	Dato	Kilde	Status	Prioritet
71	12.02.2018	Oppdragsgiver	Under arbeid	B

Beskrivelse:

Systemet bør importere kretsskjemaer fra CAD-programmer

Kravtype	Funksjonell
User story ID	3

KravID	Dato	Kilde	Status	Prioritet
72	12.02.2018	Oppdragsgiver	Fullført	C

Beskrivelse:

Brukergrensesnittet bør være W3C-Verifisert HTML5

Kravtype	Ikke-Funksjonell
User story ID	11.3

KravID	Dato	Kilde	Status	Prioritet
73	12.02.2018	Oppdragsgiver	Fullført	C
Beskrivelse: Brukergrensesnittet bør støtte nettlesere Google Chrome, Mozilla Firefox og Microsoft Edge				
Kravtype	Funksjonell			
User story ID	11.3			

KravID	Dato	Kilde	Status	Prioritet
74	12.02.2018	Oppdragsgiver	Fullført	A
Beskrivelse: Systemet skal støtte ett eksperiment per installasjon				
Kravtype				
User story ID	10.1			

KravID	Dato	Kilde	Status	Prioritet
75	12.02.2018	Oppdragsgiver	Fullført	C
Beskrivelse: Front-end bør kunne kobles til flere installasjoner				
Kravtype	Interessent			
User story ID	10.1			

KravID	Dato	Kilde	Status	Prioritet
76	12.02.2018	Oppdragsgiver	Under arbeid	B
Beskrivelse: Systemet skal kommunisere med eksperimentet via USB				
Kravtype	Funksjonell			
User story ID	10.6			

KravID	Dato	Kilde	Status	Prioritet
77	12.02.2018	Oppdragsgiver	Fullført	B
Beskrivelse: Systemet skal kommunisere med ekspermentet via IP				
Kravtype	Funksjonell			
User story ID	10.3			

KravID	Dato	Kilde	Status	Prioritet
78	12.02.2018	Oppdragsgiver	Fullført	A
Beskrivelse: Protokollbeskrivelse skal være dokumentert				
Kravtype	Ikke-Funksjonell			
User story ID	1			

KravID	Dato	Kilde	Status	Prioritet
79	12.02.2018	Oppdragsgiver	Fullført	A
Beskrivelse: Systemet skal være på engelsk				
Kravtype	Ikke-Funksjonell			
User story ID	11.8			

KravID	Dato	Kilde	Status	Prioritet
80	12.02.2018	Oppdragsgiver	Annullert	A
Beskrivelse: Systemet skal være på norsk				
Kravtype	Ikke-Funksjonell			
User story ID				

KravID	Dato	Kilde	Status	Prioritet
81	12.02.2018	Oppdragsgiver	Fullført	A
Beskrivelse: Systemet skal bruke punktum som desimalskille				
Kravtype	Ikke-Funksjonell			
User story ID	11.8			

KravID	Dato	Kilde	Status	Prioritet
82	12.02.2018	Oppdragsgiver	Fullført	A

Beskrivelse:

Systemet skal bruke komma som tusenskille

Kravtype	Ikke-Funksjonell
User story ID	11.8

KravID	Dato	Kilde	Status	Prioritet
83	01.03.2018	Master (webside)	Fullført	B

Beskrivelse:

Nettsiden skal følge spesifikasjonen til masterstudent (Brukervennlighet, webside)

Kravtype	Ikke-Funksjonell
User story ID	11.1

KravID	Dato	Kilde	Status	Prioritet
83	01.03.2018	Master (FPGA)	Annullert	B

Beskrivelse:

Systemet skal ha grensesnitt mot FPGA eksperimentkontrolleren til masterstudent (FPGA)

Kravtype	Funksjonell
User story ID	

F User stories

Dette vedlegget inneholder alle user story tabellene. Tabellene inneholder:

- ID: Løpenummer. User stories som er splittet får under-stories som har IDen til over-storien dot nytt løpenummer som ID.
- Krav: IDen til kravene user storien er knyttet til.
- Type: User/Technical/Project story/epic.
- Beskrivelse: Hva er user storien?
- Godkjenningskriterier: Hva systemet må gjøre for at user storien skal være ferdig, med IDene til testene som tester kriteriene.
- Kommentar: Eventuell ekstra informasjon om user storien.

ID	1		
Krav	78		
Type	Project epic		
Beskrivelse	Skrive nødvendig dokumentasjon		
Godkjennings-kriterier	Oppdatert etter tilbakemelding fra veileder	Testet av	
	Godkjent av veileder		
Kommentar	Denne storyen splittes ikke selv om PBIen ikke blir gjennomført i løpet av en sprint. Dokumentasjon er fortløpende arbeid som pågår gjennom hele prosjektet, og er med i Product Backlog som en påminnelse om at vi må huske å jobbe jevnt med dokumentasjon.		

ID	2		
Krav	1, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 29, 35, 36, 48		
Type	User story epic		
Beskrivelse	Som bruker vil jeg kjøre eksperimenter uavhengig av tid og sted		
Godkjennings-kriterier		Testet av	
Kommentar	Erstattet av 2.x		

ID	2.1		
Krav	29		
Type	User story		
Beskrivelse	Som bruker vil jeg se en oversikt over alle tilgjengelige eksperimenter		
Godkjennings-kriterier	Det fins en liste over alle eksperimenter	Testet av	1.1
	Linkene i listen fører deg videre til tilhørende eksperiment		2.1
	Eksperimenter blir vist i katalogen under kategorien de hører til		3.1
Kommentar			

ID	2.2		
Krav	48		
Type	User story		
Beskrivelse	Som bruker ønsker jeg informasjon om de forskjellige eksperimentene så jeg vet hvordan de fungerer og hva de demonstrerer		
Godkjennings-kriterier	Eksperimentutvikler kan legge til en beskrivelse av eksperimentet og annen informasjon	Testet av	4.1
	Eksperimentutvikler kan kan legge til bilde tilhørende eksperimentet sitt		5.1
	Bilde og beskrivelse av eksperiment vises på eksperimentsiden hvis oppgitt		6.1
Kommentar			

ID	2.3		
Krav	7, 8, 9		
Type	User story		
Beskrivelse	Som bruker trenger jeg et brukergrensesnitt så jeg kan legge inn input-parametre for eksperimentet jeg skal kjø-re		
Godkjennings-kriterier		Testet av	
Kommentar	Erstattet av 2.3.x		

ID	2.3.1		
Krav	7, 69		
Type	User story		
Beskrivelse	Som bruker ønsker jeg å legge inn input-parametere ved hjelp av tekstfelter		
Godkjennings-kriterier	Det brukeren legger inn blir lagret i JSON fil	Testet av	7.1
Kommentar			

ID	2.3.2		
Krav	7, 69		
Type	User story		
Beskrivelse	Som bruker ønsker jeg å legge inn input-parametere ved hjelp av dropdown meny		
Godkjennings-kriterier	Det brukeren velger blir lagret i JSON fil	Testet av	8.1
Kommentar			

ID	2.3.3		
Krav	8, 9		
Type	User story		
Beskrivelse	Som bruker ønsker jeg å få varsel hvis jeg sender feil input-parametere		
Godkjennings-kriterier	Bruker får varsel om ugyldige verdier	Testet av	9.1
	Eksperimentet blir ikke kjørt når brukeren sender ugyldige verdier		10.1
Kommentar			

ID	2.4		
Krav	11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 36		
Type	User story		
Beskrivelse	Som bruker vil jeg se resultater fra eksperimentet jeg har kjørt		
Godkjennings-kriterier		Testet av	
Kommentar	Erstattet av 2.4.x		

ID	2.4.1		
Krav	12, 13, 14, 15, 16, 17, 20		
Type	User story		
Beskrivelse	Som bruker vil jeg se resultater fra eksperimentet jeg har kjørt som en graf		
Godkjennings-kriterier	Graf med resultat fra kjøring av eksperiment vises i brukergrensesnittet	Testet av	11.1
Kommentar			

ID	2.4.2		
Krav	19		
Type	User story		
Beskrivelse	Som bruker vil jeg se resultater fra eksperimentet jeg har kjørt som tabeller		
Godkjennings-kriterier	Tabell med resultater fra kjøring av eksperiment vises i brukergrense-snittet	Testet av	12.1
Kommentar			

ID	2.4.3		
Krav	18		
Type	User story		
Beskrivelse	Som bruker vil jeg se resultater fra eksperimentet jeg har kjørt som kakediagrammer		
Godkjennings-kriterier	IKKE IMPLEMENTERT Kakediagram med resultater fra kjøring av eksperiment vises i brukergrensesnittet	Testet av	test id-er
Kommentar			

ID	2.4.4		
Krav	36		
Type	User story		
Beskrivelse	Som bruker vil jeg se resultater fra eksperimentet jeg har kjørt fortløpende		
Godkjennings-kriterier	Resultater som kommer fortløpende vises når de kommer	Testet av	13.1
Kommentar			

ID	2.5		
Krav	1		
Type	User story		
Beskrivelse	Som bruker vil jeg se ventetid når jeg har bestilt kjøring av eksperiment		
Godkjennings-kriterier	Plassen i køen til kjøringen av eksperimentet brukeren har bestilt vises i brukergrensesnittet	Testet av	14.1
Kommentar			

ID	2.6		
Krav	21, 22,23, 24, 35		
Type	User story		
Beskrivelse	Som bruker vil jeg se video fra eksperimentet jeg har kjørt		
Godkjennings-kriterier		Testet av	
Kommentar	Erstattet av 2.6.x		

ID	2.6.1		
Krav	21		
Type	User story		
Beskrivelse	Som bruker vil jeg se video når eksperimentet er ferdig		
Godkjennings-kriterier	Video vises etter at eksperimentet har kjørt	Testet av	
Kommentar			

ID	2.6.2		
Krav	35		
Type	User story		
Beskrivelse	Som bruker vil jeg se video av eksperimentet live		
Godkjennings-kriterier	Live video av eksperimentet vises i brukergrensesnittet	Testet av	15.1
Kommentar			

ID	2.6.3		
Krav	22		
Type	Technical story		
Beskrivelse	Synkroniser resultatene med videoen		
Godkjennings-kriterier	Resultatene på tidsaksen til graf stemmer med aktiviteten på video-en	Testet av	
Kommentar			

ID	2.6.4		
Krav	23, 24		
Type	User story		
Beskrivelse	Som bruker ønsker jeg å laste ned video fra min kjøring av eksperimentet		
Godkjennings-kriterier	Bruker kan laste ned video	Testet av	
	Video blir slettet etter 14 dager		
Kommentar			

ID	3		
Krav	37, 38, 39, 40, 41, 42, 44, 45, 46, 54, 55, 56, 71		
Type	User story epic		
Beskrivelse	Som utvikler vil jeg integrere mine eksperimenter med systemet		
Godkjennings-kriterier	Et integrert eksperiment listes i katalogen automatisk	Testet av	16.1
	Eksperimentutvikler kan kan legge til bilde tilhørende eksperimentet sitt		5.1
	Eksperimentutvikler kan kan legge til beskrivelse av eksperimentet sitt		4.1
	Eksperimentutvikler kan legge til konfigurasjonsfil for sitt eksperiment		17.1, 18.1, 19.1
	Brukergrensesnittet til et integrert eksperiment blir automatisk generert		20.1
	Systemet oppretter kobling til eksperimentet over IP		26.1
	Eksperimentutvikler kan konfigurere systemet til å ta imot video fra eksperimentet		
	Eksperimentutvikler kan overstyre det genererte grensesnittet med sin egen HTML side		
Kommentar	Skal splittes		

ID	4		
Krav	47, 49, 50, 51, 52, 53, 57, 58, 60		
Type	User story epic		
Beskrivelse	Som administrator vil jeg overvåke og vedlikeholde systemet		
Godkjennings-kriterier		Testet av	
Kommentar	Alt som har med vedlikehold, videre utvidelse, sikkerhet, ol		

ID	5		
Krav			
Type	Project epic		
Beskrivelse	Forebered 2. presentasjon		
Godkjennings-kriterier		Testet av	
Kommentar	Erstattet av 5.x		

ID	5.1		
Krav			
Type	Technical story		
Beskrivelse	Fullføre prototypen så den kan vises fram under 2 presentasjon		
Godkjennings-kriterier	Prototypen viser et komplett eksempel fra input til resultat (Ferdig 11.04.18)	Testet av	
Kommentar	Prototypen skal brukes for å finne ut hvordan det endelige systemet burde se ut og fungere. Vårt enkle eksperiment skal ta imot verdier for spenning og motstand fra brukeren og returnere strøm etter Ohms lov.		

ID	5.2		
Krav			
Type	Project story		
Beskrivelse	Lag slides til presentasjonen		
Godkjennings-kriterier	Ferdig 11.4	Testet av	
Kommentar	Presentasjon avholdt		

ID	5.3		
Krav			
Type	Project story		
Beskrivelse	Øve inn selve presentasjonen		
Godkjennings-kriterier	Ferdig 11.4	Testet av	
Kommentar	Presentasjon avholdt		

ID	5.4		
Krav			
Type	Project story		
Beskrivelse	Overlever dokumenter		
Godkjennings-kriterier	Levert inn innen fristen (Ferdig 10.04.18)	Testet av	
Kommentar	Presentasjon avholdt		

ID	6		
Krav			
Type	Project epic		
Beskrivelse	Forbered 3. presentasjon		
Godkjennings-kriterier	Presentasjon er klar til fremføring 5.6	Testet av	
Kommentar			

ID	7		
Krav			
Type	Technical story		
Beskrivelse	Avvikle prototypen og overfør til nye endelige prosjekter		
Godkjennings-kriterier		Testet av	
Kommentar	Erstattet av story 10		

ID	8		
Krav			
Type	Prosjekt story		
Beskrivelse	Sett opp Scrum		
Godkjennings-kriterier	Vi har retningslinjer for hvordan vi skal bruke Scrum (Ferdig 05.04.18)	Testet av	
	Vi har et system for å holde styr på Product Backlog (Ferdig 05.04.18)		
	Vi har et system for å holde styr på Sprint Backlogger (Ferdig 05.04.18)		
	Vi har en definisjon av “Ready” (Ferdig 05.04.18)		
	Vi har en definisjon av “Done” (Ferdig 05.04.18)		
Kommentar			

ID	9		
Krav	50, 51, 52, 53		
Type	User story epic		
Beskrivelse	Som analytiker vil jeg ha tilgang til statistikk om hvordan eksperimentene blir brukt så jeg kan studere brukerad-ferd		
Godkjennings-kriterier		Testet av	
Kommentar	Implementeres ikke i denne versjonen		

ID	10		
Krav	62, 63, 76, 77		
Type	Technical epic		
Beskrivelse	Lag core løsning for microservice		
Godkjennings-kriterier	Microservice fungerer med fakes (console i/o)	Testet av	
Kommentar	Tar utgangspunkt i prototypen, men skal være kompo-nentbasert (interface/arv). Erstattet av 10.x		

ID	10.1		
Krav	68, 69, 74, 75		
Type	Technical story		
Beskrivelse	Lag MicroService klasse som lytter og videreformidler meldinger mellom front-end og eksperimentene		
Godkjennings-kriterier	Tar imot melding fra eksperiment og sender til frontend	Testet av	22.1
	Tar imot melding fra frontend og sender til eksperiment		23.1
Kommentar			

ID	10.2		
Krav			
Type	Technical story		
Beskrivelse	Lag hjelpeklasse som skaper og parser pakker som kan sendes via USB/Ethernet/ol		
Godkjennings-kriterier	Pakker blir laget	Testet av	24.1
	Pakker blir parset		25.1
Kommentar			

ID	10.3		
Krav	62, 63, 77		
Type	Technical story		
Beskrivelse	Lag WinSock PacketInterface som brukes mot eksperimenter som kjører via TCP/IP		
Godkjennings-kriterier	Pakker sendt via TCP/IP blir tatt imot og sendt til frontend Pakker mottat fra frontend sendes videre via TCP/IP	Testet av	26.1 26.1
Kommentar			

ID	10.4		
Krav	59		
Type	Technical story		
Beskrivelse	Lag WinAPI PacketInterface som finner nye input-filer og sender de til systemet		
Godkjennings-kriterier	WinAPI PacketInterface lytter etter inputfiler og sender de videre i systemet	Testet av	27.1
Kommentar			

ID	10.5		
Krav			
Type	Technical story		
Beskrivelse	Gjør nødvendige endringer så Core kompilerer under Linux		
Godkjennings-kriterier	Core kompilerer på Linux	Testet av	28.1
Kommentar	Implementeres ikke i denne versjonen		

ID	10.6		
Krav	76		
Type	Technical story		
Beskrivelse	Lag et PacketInterface som brukes mot eksperimenter som kjører via USB		
Godkjennings-kriterier	Kan sende pakker over USB	Testet av	29.1
Kommentar	Implementeres ikke i denne versjonen		

ID	11		
Krav	2, 3, 4, 5, 6, 7, 11, 26, 28, 30, 31, 32, 33, 34, 50, 51, 52, 59, 61, 72, 73, 79, 81, 82		
Type	Technical story		
Beskrivelse	Lag core løsning for front-end		
Godkjennings-kriterier		Testet av	
Kommentar	Erstattet av 11.x		

ID	11.1		
Krav			
Type	Technical story		
Beskrivelse	Lag en forside til Remote lab i samarbeid med master-student		
Godkjennings-kriterier	Forside er laget i henhold til master-students spesifikasjon	Testet av	30.1
Kommentar			

ID	11.2		
Krav	2, 3		
Type	Technical story		
Beskrivelse	Lag et innloggingsystem som bruker lokal lagring		
Godkjennings-kriterier	Brukere kan registrere seg og bli lagret i database	Testet av	31.1
	Brukere kan logge inn		32.1
	Brukere kan logge ut		33.1
Kommentar			

ID	11.3		
Krav	7, 11, 30, 43, 72, 73		
Type	Technical story		
Beskrivelse	Lag webgrensesnitt for kjøring av eksperimenter		
Godkjennings-kriterier	Webgrensesnitt et kan ta imot parametere	Testet av	7.1, 8.1
	Webgrensesnitt kan vise resultater		11.1, 12.1, 13.1
	Webgrensesnittet fungerer i Google Chrome, Mozilla Firefox og Microsoft Edge		21.1
Kommentar			

ID	11.4		
Krav	29		
Type	Technical story		
Beskrivelse	Lag en katalogside som automatisk viser alle tilgjengelige eksperimenter		
Godkjennings-kriterier	Alle eksperimenter med config fil blir vist	Testet av	1.1
Kommentar			

ID	11.5		
Krav	25, 26, 50, 51, 52		
Type	Technical story		
Beskrivelse	Lag en brukerprofilside med historikk for bruk av eksperimenter		
Godkjennings-kriterier	Kjøringer brukeren har gjort, med input og resultat, vises	Testet av	34.1
	Bruker kan laste ned input og resultat filer i JSON format		35.1
Kommentar			

ID	11.6		
Krav	4, 5, 6		
Type	Technical story		
Beskrivelse	Lag et innloggingsystem som bruker 3.parts innlogging		
Godkjennings-kriterier	Bruker kan logge inn med Feide	Testet av	37.1
	Bruker kan logge inn med OAuth		36.1
Kommentar			

ID	11.7		
Krav	31, 32, 33, 34		
Type	Technical story		
Beskrivelse	Lag layout som tilpasser seg til forskjellige enheter		
Godkjennings-kriterier	Layout tilpasser seg store skjermer	Testet av	38.1
	Layout tilpasser seg små skjermer		39.1
	Layout tilpasser seg mobil/tablet		40.1
Kommentar			

ID	11.8		
Krav	28, 61, 79, 81, 82		
Type	Technical story		
Beskrivelse	Lag websiden så den passer til internasjonalt bruk		
Godkjennings-kriterier	Websiden er på engelsk; tekst, tall og symboler	Testet av	41.1
Kommentar			

ID	12		
Krav	59, 66		
Type	Technical story		
Beskrivelse	Lage køsystem for kjøring av eksperiment		
Godkjennings-kriterier	Flere brukere kan sette igang kjøring av samme eksperiment samtidig, og de blir kjørt etter hverandre	Testet av	42.1
Kommentar			

G Gjennomførte Tester

I dette vedlegget dokumentere vi testene vi har utført. Vi fører unit tester, integrasjonstester og godkjenningstester i forskjellige tabeller.

I tabellene for unit tester fører vi:

- ID: Løpenummer [i].[j], hvor [i] er felles for alle tester og [j] er løpenummer for hver kjøring av testen.
- Funksjonsnavn: Navn på funksjonen som blir testet.
- Subsystem: Hvilken del av systemet hører funksjonen til.
- Fil: I hvilken fil ligger funksjonen. Sammen med funksjonsnavn og subsystem skal det være mulig å finne koden til funksjonen ved å gå til kildekoden til subsytemet, finne filen og finne funksjonen.
- Beskrivelse: En beskrivelse av hva funksjonen skal gjøre.
- Avhengigheter: Hva funksjonen er avhengig av, interne og eksterne, og hvordan de blir håndtert.
- Input: Hvilke input variabler får verdier, og hva er verdien.
- Forventet Resultat: Hvilke output variabler forventes å få verdier, og hva er den forventede verdien.
- Resultat: Hvilke output variabler får verdier, og hva er verdien.
- Godkjent: Ja/Nei. Testen er godkjent hvis forventet resultat er lik resultat. Hvis testen ikke er godkjent må feilen finnes og rettes, og testen må kjøres på nytt.
- Dato: Hvilken dato testen er utført.

I tabellene for integrasjonstester fører vi:

- ID: Løpenummer [i].[j], hvor [i] er løpenummer felles for alle tester og [j] er løpenummer for hver kjøring av testen.
- Beskrivelse: En beskrivelse av hva som testes.
- Enheter under test: Hvilke funksjoner testes i denne testen og hvilken unit test er hver funksjon testet i.

- Avhengigheter: Hva funksjonen er avhengig av, interne og eksterne, og hvordan de blir håndtert.
- Input: Hvilke input variabler får verdier, og hva er verdien.
- Forventet Resultat: Hvilke output variabler forventes å få verdier, og hva er den forventede verdien.
- Resultat: Hvilke output variabler får verdier, og hva er verdien.
- Godkjent: Ja/Nei. Testen er godkjent hvis forventet resultat er lik resultat. Hvis testen ikke er godkjent må feilen finnes og rettes, og testen må kjøres på nytt.
- Dato: Hvilken dato testen er utført.

I tabellene for godkjenningstester fører vi:

- ID: Løpenummer [i].[j], hvor [i] er løpenummer felles for alle tester og [j] er løpenummer for hver kjøring av testen.
- User story: ID til user storiene som har godkjenningskrav som blir testet av testen.
- Beskrivelse: En beskrivelse av hva som testes.
- Avhengigheter: Hva funksjonen er avhengig av, interne og eksterne, og hvordan de blir håndtert.
- Fremgangsmåte: Hvordan man reproduserer testen.
- Forventet Resultat: Hva forventes at skjer
- Resultat: Hva skjedde
- Godkjent: Ja/Nei. Testen er godkjent hvis forventet resultat er lik resultat. Hvis testen ikke er godkjent må feilen finnes og rettes, og testen må kjøres på nytt.
- Dato: Hvilken dato testen er utført.

G.1 Unit tester

G.2 Integrasjonstester

G.3 Godkjenningstester

ID	1.1	User story	2.1, 11.4
Beskrivelse	Ruten /catalog/all på nettsiden inneholder en liste over alle eksperimenter som har blitt integrert og fått konfigurasjonsfil		
Avhengigheter			
1.json fil finnes i CONFIG mappe, med felt for navn: experiment 1 2.json fil finnes i CONFIG mappe, med felt for navn: experiment 2 3.json fil finnes i CONFIG mappe, med felt for navn: experiment 3			
Fremgangsmåte			
1. Gå til /catalog/all ruten på nettsiden			
Forventet Resultat			
Websiden har en liste som inneholder: experiment 1 experiment 2 experiment 3			
Resultat			
Det er en liste på websiden som inneholder experiment 1 experiment 2 experiment 3			
Godkjent	Ja	Dato	20.05.18

ID	2.1	User story	2.1
Beskrivelse	Alle eksperimenter listet i katalogen er en link som linker til sitt eksperimentgrensesnitt		
Avhengigheter			
1.json fil finnes i CONFIG mappe, med felt for navn: experiment 1 2.json fil finnes i CONFIG mappe, med felt for navn: experiment 2 3.json fil finnes i CONFIG mappe, med felt for navn: experiment 3			
Fremgangsmåte			
1. Gå til ruten /catalog/all på nettsiden 2. Klikk på eksperiment 1 i listen 3. Klikk på eksperiment 2 i listen 4. Klikk på eksperiment 3 i listen			
Forventet Resultat			
Å klikke på pilen ved siden av “eksperiment 1” tar deg til ruten /experiment/1 på nettsiden Å klikke på pilen ved siden av “eksperiment 2” tar deg til ruten /experiment/2 på nettsiden Å klikke på pilen ved siden av “eksperiment 3” tar deg til ruten /experiment/3 på nettsiden			
Resultat			
Når man klikker på pilen ved siden av “eksperiment 1” blir man tatt til ruten experiment/1 Når man klikker på pilen ved siden av “eksperiment 2” blir man tatt til ruten experiment/2 Når man klikker på pilen ved siden av “eksperiment 3” blir man tatt til ruten experiment/3			
Godkjent	Ja	Dato	21.5.18

ID	3.1	User story	2.1
Beskrivelse	Å gå til ruten /category/mechanical og få liste over alle eksperimenter med kategorien mechanical		
Avhengigheter			
1.json fil finnes i CONFIG mappe, med felt for navn: experiment 1, og kategori: mechanical 2.json fil finnes i CONFIG mappe, med felt for navn: experiment 2, og kategori: mechanical 3.json fil finnes i CONFIG mappe, med felt for navn: experiment 3, og kategori: mechanical 4.json fil finnes i CONFIG mappe, med felt for navn: experiment 4, og kategori: physics			
Fremgangsmåte			
1. Gå til ruten /category/mechanical på nettsiden			
Forventet Resultat			
Listen inneholder ikke experiment 4 Websiden har liste som inneholder: experiment 1 experiment 2 experimnet 3			
Resultat			
Det er en liste på websiden som inneholder: experiment 1 experiment 2 experiment 3 Ikke experiment 4			
Godkjent	Ja	Dato	21.5.18

ID	4.1	User story	2.2, 3
Beskrivelse	Når eksperimentutvikler integrerer sitt eksperiment med systemt, inkluderer det å legge til en beskrivelse av eksperimentet		
Avhengigheter			
Ha tilgang til konto med utvikler rettigheter Kontoen har integrert et eksperiment som har fått id 99			
Fremgangsmåte			
1. Logg inn med konto med utvikler rettigheter på nettsiden 2. Gå til ruten /development/setup/99 3. Skriv inn “dette er en beskrivelse” i description feltet 4. Trykk Add info			
Forventet Resultat			
Konfigurasjonsfilen inneholder felt for description med verdi “dette er en beskrivelse”			
Resultat			
Konfigurasjonsfilen inneholder felt for description med verdi “dette er en beskrivelse”			
Godkjent	Ja	Dato	21.5.18

ID	5.1	User story	2.2, 3
Beskrivelse	Når eksperimentutvikler integrerer sitt eksperiment med systemt, inkluderer det å legge til et bilde som beskriver eksperimentet		
Avhengigheter			
Ha tilgang til konto med utvikler rettigheter Kontoen har integrert et eksperiment som har fått id 99			
Fremgangsmåte			
1. Logg inn med konto med utvikler rettigheter på nettsiden 2. Gå til ruten /development/setup/99 på nettsiden 3. Trykk “velg fil” ved bilde feltet 4. Velg en bildefil med navn 1234bilde.png 5. Trykk Add info			
Forventet Resultat			
Konfigurasjonsfilen inneholder felt for image med verdi “/pictures/experimentImage99.png”			
Resultat			
Konfigurasjonsfilen inneholder felt for image med verdi “/pictures/experimentImage99.png”			
Godkjent	y/n	Dato	21.5.18

ID	6.1	User story	2.2
Beskrivelse	Hvis eksperimentutvikler lastet opp bilde og/eller skrev beskrivelse til sitt eksperiment, vises dette på eksperimentsiden		
Avhengigheter			
Eksperiment med id 99 er integrert Utvikler la til beskrivelse til eksperiment med id 99: “Dette er beskrivelse” Utivkler lastet opp bilde til eksperiment med id 99: pic.png			
Fremgangsmåte			
1. Gå til ruten /experiment/99 på nettsiden			
Forventet Resultat			
Websiden inneholder tekst som sier “Detter er beskrivelse” Websiden inneholder bilde eksperimentImage99.png			
Resultat			
Websiden inneholder tekst som sier “Detter er beskrivelse” Websiden viser bilde eksperimentImage99.png			
Godkjent	Ja	Dato	21.5.18

ID	7.1	User story	2.3.1 11.3
Beskrivelse	I brukergrensesnittet til et eksperiment kan bruker legge inn verdier til parameterene til eksperimentet via tekstfelt		
Avhengigheter			
Eksperiment med id 10 er integrert Eksperiment med id 10 har parameter “P1” med type integer Eksperiment med id 10 har parameter “P2” med type floatingpoint			
Fremgangsmåte			
1. Gå til ruten /experiment/10 på nettsiden 2. Tast inn “14” i P1 feltet 3. Tast inn “1.111” i P2 feltet 4. Trykk Submit			
Forventet Resultat			
En fil blir laget i INPUT mappen til eksperimentet med id 10 Filen inneholder data på inputformatet hvor P1 feltet har verdi 14 og P2 feltet har verdi 1.111			
Resultat			
En fil ble laget i INPUT mappen til eksperimentet med id 10 Filen inneholder data på inputformatet hvor P1 feltet har verdi 14 og P2 feltet har verdi 1.111			
Godkjent	Ja	Dato	21.5.18

ID	8.1	User story	2.3.2 11.3
Beskrivelse	I brukergrensesnittet til et eksperiment kan bruker velge verdier til parameterene til eksperimentet fra dropdown menyer		
Avhengigheter			
Eksperiment med id 10 er integrert Eksperiment med id 10 har parameter “P1” med type enum, med verdier “valg 1”, “valg 2”, og “valg 3”			
Fremgangsmåte			
1. Gå til ruten /experiment/10 på nettsiden 2. Velg “valg 2” i P1 menyen 3. Trykk Submit			
Forventet Resultat			
En fil blir laget i INPUT mappen til eksperiment med id 10 Filen inneholder data på inputformatet hvor P1 feltet har verdi “valg 2”			
Resultat			
En fil ble laget i INPUT mappen til eksperiment med id 10 Filen inneholder data på inputformatet hvor P1 feltet har verdi “valg 2”			
Godkjent	Ja	Dato	21.5.18

ID	9.1	User story	2.3.3
Beskrivelse	I brukergrensesnittet til et eksperiment får bruker varsel om at han har sendt ugyldige parameterverdier		
Avhengigheter			
Eksperiment med id 10 er integrert Eksperiment med id 10 har parameter “P1” med type floatingpoint			
Fremgangsmåte			
1. Gå til ruten /experiment/10 på nettsiden 2. Tast inn “abc” i P1 feltet 3. Trykk Submit			
Forventet Resultat			
Melding dukker opp i grensesnittet som sier at P1 er feil			
Resultat			
Medling dukker opp som sier “P1 must be a decimal number”			
Godkjent	Ja	Dato	21.5.18

ID	10.1	User story	2.3.3
Beskrivelse	Når bruker sender ugyldige parameter verdier, blir eksperimentet ikke kjørt		
Avhengigheter			
Eksperiment med id 10 er integrert Eksperiment med id 10 har parameter “P1” med type floatingpoint			
Fremgangsmåte			
1. Gå til ruten /experiment/10 på nettsiden 2. Tast inn “abc” i P1 feltet 3. Trykk Submit			
Forventet Resultat			
Ingen fil blir laget i INPUT mappen til eksperiment med id 10			
Resultat			
Ingen fil ble laget i INPUT mappen til eksperimentet med id 10			
Godkjent	Ja	Dato	21.5.18

ID	11.1	User story	2.4.1, 11.3
Beskrivelse	Når en kjøring av et eksperiment er ferdig, vises resultatene til brukeren i brukergrensesnittet som graf		
Avhengigheter			
Eksperiment med id 10 er integrert Fil som følger resultformatet, og inneholder verdiene 1,2,3,4,5 for feltet R1 og 2,4,2,4,2 for feltet R2			
Fremgangsmåte			
1. Gå til ruten /experiment/10 på nettsiden 2. Trykk Submit 3. Flytt inputfilen fra INPUT mappen til RUNNING mappen til eksperimentet med id 10 4. Legg resultat fil i RESULT mappen til eksperimentet med id 10			
Forventet Resultat			
Graf dukker opp i brukergrensesnittet med to kurver, R1 og R2, med verdier 1,2,3,4,5 og 2,4,2,4,2 på y-aksen bortover fra null på x-aksen			
Resultat			
En graf dukker opp i brukergrensesnittet med to kurver, R1 og R2, med verdier 1,2,3,4,5 og 2,4,2,4,2 på y-aksen bortover fra null på x-aksen			
Godkjent	Ja	Dato	21.5.18

ID	12.1	User story	2.4.2, 11.3
Beskrivelse	Når en kjøring av et eksperiment er ferdig, vises resultatene til brukeren i brukergrensesnittet i tabell		
Avhengigheter			
Eksperiment med id 10 er integrert Fil som følger resultformatet, og inneholder verdiene 1,2,3,4,5 for feltet R1 og 2,4,2,4,2 for feltet R2			
Fremgangsmåte			
1. Gå til ruten /experiment/10 på nettsiden 2. Trykk Submit 3. Flytt inputfilen fra INPUT mappen til RUNNING mappen til eksperimentet med id 10 4. Legg resultat fil i RESULT mappen til eksperimentet med id 10			
Forventet Resultat			
Liste med verdier 1,2,3,4,5 under R1 og verdier 2,4,2,4,2 under R2 dukker opp i brukergrensesnittet			
Resultat			
En liste med verdier 1,2,3,4,5 under R1 og verdier 2,4,2,4,2 under R2 dukker opp i brukergrensesnittet			
Godkjent	Ja	Dato	21.5.18

ID	13.1	User story	2.4.4 11.3
Beskrivelse	Når resultat filen oppdateres med ny data, oppdateres visningene av resultatene		
Avhengigheter			
Eksperiment med id 10 er integrert Fil som følger resultformatet, og inneholder verdi 1 for feltet R1 og 2 for feltet R2 Fil som følger resultformatet, og inneholder verdiene 1,2 for feltet R1 og 2,4 for feltet R2 Fil som følger resultformatet, og inneholder verdiene 1,2,3 for feltet R1 og 2,4,2 for feltet R2			
Fremgangsmåte			
1. Gå til ruten /experiment/10 på nettsiden 2. Trykk Submit 3. Flytt inputfilen fra INPUT mappen til RUNNING mappen til eksperimentet med id 10 4. Legg resultat fil 1 i RESULT mappen til eksperimentet med id 10 5. Fjern resultat fil 1 fra RESULT mappen til eksperimentet med id 10 6. Legg resultat fil 2 i RESULT mappen til eksperimentet med id 10 7. Fjern resultat fil 2 fra RESULT mappen til eksperimentet med id 10 8. Legg resultat fil 3 i RESULT mappen til eksperimentet med id 10			
Forventet Resultat			
Visningene av resultatene inneholder verdiene i fil 1 Visningene av resultatene blir oppdatert til å inneholde verdiene i fil 2 Visningene av resultatene blir oppdatert til å inneholde verdiene i fil 3			
Resultat			
Visningene av resultatene inneholder verdiene i fil 1 Visningene av resultatene blir oppdatert til å inneholde verdiene i fil 2 Visningene av resultatene blir oppdatert til å inneholde verdiene i fil 3			
Godkjent	Ja	Dato	21.5.18

ID	14.1	User story	2.5
Beskrivelse	Når flere brukere bestiller kjøringer av et eksperiment, legges de i kø, og plassen i køen vises i brukergrensesnittet		
Avhengigheter			
Eksperiment med id 10 er integrert			
Fremgangsmåte			
1. Gå til ruten /experiment/10 på nettsiden 2. Trykk Submit 3. Åpne ny fane 4. Gå til ruten /experiment/10 på nettsiden 5. Trykk Submit 6. Åpne ny fane 7. Gå til ruten /experiment/10 på nettsiden 8. Trykk Submit			
Forventet Resultat			
Første fane viser at kjøringen har fått plass 1 i køen Andre fane viser at kjøringen har fått plass 2 i køen Tredje fane viser at kjøringen har fått plass 3 i køen			
Resultat			
Første fane viser at kjøringen har fått plass 1 i køen Andre fane viser at kjøringen har fått plass 2 i køen Tredje fane viser at kjøringen har fått plass 3 i køen			
Godkjent	Ja	Dato	21.5.18

ID	15.1	User story	2.6.2
Beskrivelse	Video fra kamera strømmes til en PC som strømmer videoen til Twitch.tv, brukergrensesnittet viser videoen fra Twitch.tv		
Avhengigheter			
Eksperiment med id 10 er integrert Kamera som strømmer til PC PC som strømmer til Twitch.tv Twitch.tv er tilgjengelig			
Fremgangsmåte			
1. Gå til ruten /experiment/10 på nettsiden			
Forventet Resultat			
Live video av eksperiment med id 10 vises i grensesnittet			
Resultat			
Live video av eksperiment med id 10 vises i grensesnittet			
Godkjent	Ja	Dato	21.5.18

ID	16.1	User story	3
Beskrivelse	Når en eksperimentutvikler integrerer sitt eksperiment i systemet, blir eksperimentet automatisk lagt til i katalogen		
Avhengigheter			
Tilgang til en konto med utviklerrettigheter Konfigurasjonsfil for eksperimentet følger konfigurasjonsformatet			
Fremgangsmåte			
1. Legg til konfigurasjonsfil i CONFIG mappen, med navn: nytt eksperiment og kategori: biology 2. Gå til ruten /catalog/all på nettsiden 3. Gå til ruten /catalog/biology på nettsiden			
Forventet Resultat			
Listen i /catalog/all inneholder “nytt eksperiment” Listen i /catalog/biology inneholder “nytt eksperiment”			
Resultat			
Listen i /catalog/all inneholder “nytt eksperiment” Listen i /catalog/biology inneholder “nytt eksperiment”			
Godkjent	Ja	Dato	21.5.18

ID	17.1	User story	3
Beskrivelse	Eksperimentutvikler kan laste opp en konfigurasjonsfil for et nytt eksperiment		
Avhengigheter			
Har tilgang til konto med utviklerrettigheter Konfigurasjonsfilen følger konfigurasjonformatet			
Fremgangsmåte			
1. Gå til ruten /development på nettsiden 2. Velg fil ved “Upload Config file” 3. Trykk Submit			
Forventet Resultat			
En ID blir generert til eksperimentet og lagt til i konfigurasjonsfilen Konfigurasjonsfilen blir lagt til i CONFIG mappen med id-en som navn			
Resultat			
En ID ble generert og lagt til i konfigurasjonsfilen Konfigurasjonsfilen ble lagt til i CONFIG mappen med id-en som navn			
Godkjent	Ja	Dato	21.5.18

ID	18.1	User story	3
Beskrivelse	Eksperimentutvikler kan skrive og sende en konfigrua-sjonsfil for et nytt eksperiment fra nettsiden		
Avhengigheter			
Har tilgang til konto med utviklerrettigheter Konfigurasjonsfilen følger konfigurasjonformatet			
Fremgangsmåte			
1. Gå til ruten /development på nettsiden 2. Velg Create Config File Online 3. Skriv konfigurasjonsfilen i tekstfeltet 4. Trykk Submit			
Forventet Resultat			
En ID blir generert til eksperimentet og lagt til i konfigurasjonsfilen Konfigurasjonsfilen blir lagt til i CONFIG mappen med id-en som navn			
Resultat			
En ID ble generert og lagt til i konfigurasjonsfilen Konfigurasjonsfilen ble lagt til i CONFIG mappen med id-en som navn			
Godkjent	Ja	Dato	21.5.18

ID	19.1	User story	3
Beskrivelse	Eksperimentutvikler kan sette opp eksperimentkontrolleren til eksperimentet sitt til å sende en konfigruasjonsfil for seg selv til serveren, og sette igang denne prosessen		
Avhengigheter			
Har tilgang til konto med utviklerrettigheter Konfigurasjonsfilen følger konfigurasjonformatet Eksperimentet er tilgjengelig over IP Eksperimentet er satt opp til å sende konfigurasjonsfilen når den mottar discovery request pakke			
Fremgangsmåte			
1. Gå til ruten /development på nettsiden 2. Skriv inn IP addressen til eksperimentet under Autodiscover 3. Trykk Submit			
Forventet Resultat			
En ID blir generert til eksperimentet og lagt til i konfigurasjonsfilen Konfigurasjonsfilen blir lagt til i CONFIG mappen med id-en som navn			
Resultat			
En ID ble generert og lagt til i konfigurasjonsfilen Konfigurasjonsfilen ble lagt til i CONFIG mappen med id-en som navn			
Godkjent	Ja	Dato	21.5.18

ID	20.1	User story	3
Beskrivelse	Et brukergrensesnitt med felter for inputparametere til eksperimentet blir automatisk generert basert på konfigurasjonsfilen.		
Avhengigheter			
Konfigurasjonsfil for eksperiment har blitt lagt til Eksperimentet har fått id 10 Eksperimentet har parameter P1 av type integer Eksperimentet har parameter P2 av type floatingPoint Eksperimentet har parameter P3 av type enum, med verdier a,b,c,d,e Eksperimentet har parameter P4 av type boolean			
Fremgangsmåte			
1. Gå til ruten /experiment/10 på nettsiden			
Forventet Resultat			
Websiden inneholder tekstfelt for P1 og P2, dropdown meny for P3 med a,b,c,d og e som valg, og sjekkboks for P4			
Resultat			
Websiden inneholder tekstfelt for P1 Websiden inneholder tekstfelt for P2 Websiden inneholder dropdown meny for P3 Websiden inneholder sjekkboks for P4			
Godkjent	Ja	Dato	21.5.18

ID	21.1	User story	11.3
Beskrivelse	Brukergrensesnittet fungerer i Google Chrome, Mozilla Firefox og Microsoft Edge		
Avhengigheter			
Ha Google Chrome installert			
Ha Mozilla Firefox installert			
Ha Microsoft Edge installert			
Fremgangsmåte			
1. Gå til brukergrensesnittet til et eksperiment i Chrome			
2. Kjør eksperimentet			
3. Gå til brukergrensesnittet til et eksperiment i Firefox			
4. Kjør eksperimentet			
5. Gå til brukergrensesnittet til et eksperiment i Edge			
6. Kjør eksperimentet			
Forventet Resultat			
Tekstfelter, sjekkboks, og dropdown meny fungerer i Chrome			
Tabell og grafer med resultater vises i Chrome			
Tekstfelter, sjekkboks, og dropdown meny fungerer i Firefox			
Tabell og grafer med resultater vises i Firefox			
Tekstfelter, sjekkboks, og dropdown meny fungerer i Edge			
Tabell og grafer med resultater vises i Edge			
Resultat			
Tekstfelter, sjekkboks, og dropdown meny fungerer i Chrome			
Tabell og grafer med resultater ble vist i Chrome			
Tekstfelter, sjekkboks, og dropdown meny fungerer i Firefox			
Tabell og grafer med resultater ble vist i Firefox			
Tekstfelter, sjekkboks, og dropdown meny fungerer i Edge			
Tabell og grafer med resultater ble vist i Edge			
Godkjent	Ja	Dato	21.5.18

ID	22.1	User story	10.1
Beskrivelse	Mikroservice lytter til frontend og backend og viderefor-midler meldinger		
Avhengigheter			
Det er koblet opp en frontend Det er koblet opp et eksperiment Sørg for at det er lagt til en config for et eksperiment.			
Fremgangsmåte			
1. Start mikrotjenesten 2. Motta pakker fra eksperiment og send til frontend			
Forventet Resultat			
Eksperimentet sender en responspakke Eksperimentet mottar input og sender en start marker Eksperiment sender video sync marker Eksperiment sender et resultat Frontend mottar resultat Eksperiment sender en end marker Eksperiment mottar service pakke			
Resultat			
Eksperimentet sender en responspakke Eksperimentet mottar input og sender en start marker Eksperiment sender video sync marker Eksperiment sender et resultat Frontend mottar resultat Eksperiment sender en end marker Frontend sender service pakke Eksperiment mottar service pakke			
Godkjent	Ja	Dato	21.5.18

ID	23.1	User story	10.1
Beskrivelse	Mikroservice lytter til frontend og backend og viderefor-midler meldinger		
Avhengigheter			
Det er koblet opp en frontend Det er koblet opp et eksperiment Sørg for at det er lagt til en config for et eksperiment.			
Fremgangsmåte			
1. Start mikrotjenesten 2. Send input fra frontend			
Forventet Resultat			
Frontend sender input Eksperimentet mottar input Frontend sender service pakke Eksperiment mottar service pakke			
Resultat			
Frontend sender input Eksperimentet mottar inpu Frontend sender service pakke Eksperiment mottar service pakke			
Godkjent	Ja	Dato	21.5.18

ID	24.1	User story	10.2
Beskrivelse	Hjelpeklassen skal skape og parse pakker som skal til og fra frontend/eksperiment		
Avhengigheter			
Trenger en protokoll Trenger en frontend som sender/mottar pakker Trenger et eksperiment som sender/mottar pakker Trenger en config til å lese fra			
Fremgangsmåte			
1. Send pakker 1 til eksperiment 3. Send pakke 4 til eksperiment 5. Send pakke 9 til eksperiment 6. Send pakke 10 til mikrotjenesten			
Forventet Resultat			
Fra pakke 1 sendes til eksperiment skal det fortløpende sendes pakker frem og tilbake til pakke 10 er sendt			
Resultat			
Pakker ble sendt frem og tilbake uten problemer Pakker ble laget			
Godkjent	Ja	Dato	21.5.18

ID	25.1	User story	10.2
Beskrivelse	Hjelpeklassen skal skape og parse pakker som skal til og fra frontend/eksperiment		
Avhengigheter			
Trenger en protokoll Trenger en frontend som sender/mottar pakker Trenger et eksperiment som sender/mottar pakker Trenger en config til å lese fra			
Fremgangsmåte			
1. Motta pakker 2 og 3 fra eksperiment 2. Motta pakker 5-8 fra eksperiment			
Forventet Resultat			
Fra pakke 1 sendes til eksperiment skal det fortløpende sendes pakker frem og tilbake til pakke 10 er sendt			
Resultat			
Pakker ble sendt frem og tilbake uten problemer Pakker ble parset			
Godkjent	Ja	Dato	21.5.18

ID	26.1	User story	10.3, 3
Beskrivelse	Winsock PacketInterface sender og tar imot pakker som overføres via TCP		
Avhengigheter			
Trenger en protokoll Trenger en frontend som sender/mottar pakker Trenger et eksperiment som sender/mottar pakker og kobler seg opp mot winsock Trenger en config for å ha pakker å sende			
Fremgangsmåte			
1. Se om det er en kobling mot eksperiment 2. Koble mot eksperiment dersom det ikke er kobling 3. Videresend pakker dersom det kommer pakker fra frontend til eksperimentet 4. Motta pakker dersom det kommer pakker fra eksperiment, og videresend til mikrotjensten for parsing 5. Send pakke 9 til eksperiment 6. Send pakke 10 til mikrotjenesten			
Forventet Resultat			
Kobler seg til eksperimentet Videresender pakker fra frontend, til eksperiment Mottar og videresender pakker fra eksperiment, til frontend			
Resultat			
Mikrotjenesten og eksperiemnt ble koblet opp Pakker ble videresendt Pakker ble mottat og videresendt			
Godkjent	Ja	Dato	21.5.18

ID	27.1	User story	10.4
Beskrivelse	WinAPI PacketInterface skal lytte etter inputfiler og sende de videre i systemet		
Avhengigheter			
Trenger inputfiler Trenger End Marker pakke			
Fremgangsmåte			
1. Se om det er kommet inn en inputfil 2. Flytte input fra en mappe til en annen 3. Motta end marker når eksperiment er ferdig med å sende resultat 4. Bruke end marker for å initialisere flytting av ny input			
Forventet Resultat			
Lytter etter input, og flytter denne Dersom ny input, legg den i kø, men ikke flytt før end marker er kommet inn Motta end marker og flytt inputen ut av mappe til arkiv, og samtidig flytt ny input inn i kjøringsmappen			
Resultat			
Flytter input, og holder igjen neste, dersom flere Mottar end marker, og flytter input fil til arkiv, og flytter neste input fil til kjøringsmappen Bruker ikke WinAPI, men et annet system som gjør jobben slik vi ønsket			
Godkjent	Ja	Dato	21.5.18

ID	28.1	User story	10.5
Beskrivelse	Endre koden slik at den kan kjøres på Linux		
Avhengigheter			
Linux server Trenger fungerende mikrotjeneste			
Fremgangsmåte			
1. Konverter til Linux 2. Start mikrotjenesten 3. Kjør fakeEksperimentfor å se at alle pakker sendes og mottas som de skal			
Forventet Resultat			
Mikrotjenesten kompilerer på Linux			
Resultat			
Kompilerte ikke, og user storien blir overført til versjon 2.0			
Godkjent	Nei	Dato	21.5.18

ID	29.1	User story	10.6
Beskrivelse	Har en fungerende packetInterface som brukes mot eksperimenter som kjører via USB		
Avhengigheter			
Eksperiment som kjører via USB Protokoll Config som kan brukes til å teste interfacen ved hjelp av protokollen			
Fremgangsmåte			
1. Koble til USB 2. Bruk console til å sende pakker til eksperimentet 3. Bruk console til å se om pakker blir mottatt fra eksperiment			
Forventet Resultat			
Pakker blir sendt frem og tilbake			
Resultat			
Har ingen eksperiment med USB, og derfor er dette ikke blitt testet			
Godkjent	Nei	Dato	21.5.18

ID	30.1	User story	11.1
Beskrivelse	Har en nettside for kjøring av eksperimenter laget i sam-arbeid med masterstudent		
Avhengigheter			
Masterstudenten kommer med design ønsker USN design manual			
Fremgangsmåte			
1. Lag utkast av nettsiden 2. Lag template av det som skal brukes på hver enkelt side 3. Utvid med flere funksjoner etterhvert som masterstudent kommer med ønsker			
Forventet Resultat			
Brukervennlig nettside			
Resultat			
Brukervennlig nettside			
Godkjent	Ja	Dato	21.5.18

ID	31.1	User story	11.2
Beskrivelse	Brukere som registerer seg på nettsiden skal lagres lokalt i en database		
Avhengigheter			
Database for registrering og lagring av brukere Autoriseringsmiddleware som håndterer autorisasjon av brukere.			
Fremgangsmåte			
1. Installer en database 2. Installer og implementer en middleware for autorisering			
Forventet Resultat			
Brukere får ikke tilgang til diverse sider uten registering Brukere registrerer seg og informasjonen lagres i en database			
Resultat			
Brukere får ikke tilgang til diverse sider uten registering Brukere registrerer seg og informasjonen lagres i en database			
Godkjent	Ja	Dato	21.5.18

ID	32.1	User story	11.2
Beskrivelse	Brukere som registerer seg på nettsiden skal lagres lokalt i en database		
Avhengigheter			
Database for registrering og lagring av brukere Autoriseringsmiddleware som håndterer autorisasjon av brukere.			
Fremgangsmåte			
1. Installer en database 2. Installer og implementer en middleware for autorisering			
Forventet Resultat			
Brukere autoriseres når de logger inn, ved å sjekke i databasen om de allerede er registrert			
Resultat			
Brukere autoriseres når de logger inn, ved å sjekke i databasen om de allerede er registrert			
Godkjent	Ja	Dato	21.5.18

ID	33.1	User story	11.2
Beskrivelse	Brukere som registerer seg på nettsiden skal lagres lokalt i en database		
Avhengigheter			
Database for registrering og lagring av brukere Autoriseringsmiddleware som håndterer autorisasjon av brukere.			
Fremgangsmåte			
1. Installer en database 2. Installer og implementer en middleware for autorisering			
Forventet Resultat			
Brukere kan logge ut etter endt økt			
Resultat			
Brukere kan logge ut etter endt økt			
Godkjent	Ja	Dato	21.5.18

ID	34.1	User story	11.5
Beskrivelse	En profilside hvor brukeren kan se og laste ned sine tidligere kjørte eksperimenter		
Avhengigheter			
Nettside med database for lagring av resultat ID mot bruker ID Eksperimenter for å lage resultatfiler som kan lastes ned			
Fremgangsmåte			
1. Logg inn 2. Trykk på brukernavnet 3. Velg resultatfilen som skal lastes ned			
Forventet Resultat			
Brukeren får en tabell med resultater fra tidligere kjørte eksperiment			
Resultat			
Brukeren får en tabell med resultater fra tidligere kjørte eksperiment			
Godkjent	Ja	Dato	21.5.18

ID	35.1	User story	11.5
Beskrivelse	En profilside hvor brukeren kan se og laste ned sine tidligere kjørte eksperimenter		
Avhengigheter			
Nettside med database for lagring av resultat ID mot bruker ID Eksperimenter for å lage resultatfiler som kan lastes ned			
Fremgangsmåte			
1. Logg inn 2. Trykk på brukernavnet 3. Velg resultatfilen som skal lastes ned			
Forventet Resultat			
Brukeren kan laste ned resultatfiler			
Resultat			
Brukeren kan laste ned resultatfiler			
Godkjent	Nei	Dato	21.5.18

ID	36.1	User story	11.6
Beskrivelse	Brukere kan logge inn ved å autorisere seg via 3 part		
Avhengigheter			
Nettside med 3 part innloggingsmuligheter En konto på den valgte 3 partssiden Database for å lagre brukere Godkjent middleware som håndterer OAuth autorisering			
Fremgangsmåte			
1. Gå til logg inn 2. Trykk på ønsket 3 parts tjeneste			
Forventet Resultat			
Brukeren kan logge inn via 3 part			
Resultat			
Brukeren kan logge inn via 3 part Brukeren kan logge inn via Google konto			
Godkjent	Ja	Dato	21.5.18

ID	37.1	User story	11.6
Beskrivelse	Brukere kan logge inn ved å autorisere seg via 3 part		
Avhengigheter			
Nettside med 3 part innloggingsmuligheter En konto på den valgte 3 partssiden Database for å lagre brukere Godkjent middleware som håndterer OAuth autorisering			
Fremgangsmåte			
1. Gå til logg inn 2. Trykk på ønsket 3 parts tjeneste			
Forventet Resultat			
Brukeren kan logge inn via 3 part			
Resultat			
Brukeren kan ikke logge inn via feide pga restriksjoner som gjør at vi ikke fikk implementert det			
Godkjent	Nei	Dato	21.5.18

ID	38.1	User story	11.7
Beskrivelse	Nettsiden skal ha en layout som passer til alle skjermstørrelser		
Avhengigheter			
CSS stylesheet Skjerm/skjermer for å teste forskjellig størrelse			
Fremgangsmåte			
1. Åpne nettsiden på en skjerm med 1200px eller mer 2. Se etter tydelige feil i design			
Forventet Resultat			
Nettsiden holder sitt opprinnelige design			
Resultat			
Nettsiden holder sitt opprinnelige design			
Godkjent	Ja	Dato	21.5.18

ID	39.1	User story	11.7
Beskrivelse	Nettsiden skal ha en layout som passer til alle skjermstørrelser		
Avhengigheter			
CSS stylesheet Skjerm/skjermer for å teste forskjellig størrelse			
Fremgangsmåte			
1. Åpne nettsiden på en skjerm med 992px eller mer 2. Se etter tydelige feil i design			
Forventet Resultat			
Nettsiden holder sitt opprinnelige design			
Resultat			
Nettsiden holder sitt opprinnelige design			
Godkjent	Ja	Dato	21.5.18

ID	40.1	User story	11.7
Beskrivelse	Nettsiden skal ha en layout som passer til alle skjermstørrelser		
Avhengigheter			
CSS stylesheet Skjerm/skjermer for å teste forskjellig størrelse			
Fremgangsmåte			
1. Åpne nettsiden på en skjerm med 768px eller mindre 2. Se etter tydelige feil i design			
Forventet Resultat			
Nettsiden endrer design for å bedre synliggjøre elementer for brukere av mobil eller annen liten skjerm			
Resultat			
Nettsiden endrer design for å bedre synliggjøre elementer for brukere av mobil eller annen liten skjerm			
Godkjent	Ja	Dato	21.5.18

ID	41.1	User story	11.8
Beskrivelse	Nettsiden skal ha innhold på engelsk, og være tilpasset internasjonalt bruk		
Avhengigheter			
Engelsk språk og symboler			
Fremgangsmåte			
1. Gå til nettsiden 2. Bla gjennom nettsiden og se om språk og symboler er internasjonalt			
Forventet Resultat			
Nettsiden har engelsk språk og internasjonale symboler			
Resultat			
Nettsiden har engelsk språk og internasjonale symboler			
Godkjent	Ja	Dato	21.5.18

ID	42.1	User story	12
Beskrivelse	Køsystem som styrer kjøring av eksperiment		
Avhengigheter			
Trenger inputfiler Trenger End Marker pakke			
Fremgangsmåte			
1. Se om det er kommet inn en inputfil 2. Flytte input fra en mappe til en annen 3. Motta end marker når eksperiment er ferdig med å sende resultat 4. Bruke end marker for å initialisere flytting av ny input			
Forventet Resultat			
Lytter etter input, og flytter denne Dersom ny input, legg den i kø, men ikke flytt før end marker er kommet inn Motta end marker og flytt inputen ut av mappe til arkiv, og samtidig flytt ny input inn i kjøringsmappen			
Resultat			
Flytter input, og holder igjen neste, dersom flere Mottar end marker, og flytter input fil til arkiv, og flytter neste input fil til kjøringsmappen			
Godkjent	Ja	Dato	21.5.18

H Dokumentutforming

Dette dokumentet er laget som en guide for gruppen som skal følges når det skrives dokumenter, eller når det lages kode.

H.1 Utseende

Prosjektgruppen har ingen selvlaget logo og der det er behov for logo, som for eksempel på nettsiden som lages i samarbeid med masterstudenten, skal det brukes USN sin egen logo. Ved bruk av denne logoen er det viktig å følge USN sin dokumentasjon for logobruk([52]).

Gruppen følger USN sin mal for oppgaveskriving, hvor de viktigste punktene er:

- Indre marg skal være 4 cm, ytre- og toppmarg 2 cm, og bunnmarg 2,54 cm.
- Skrifttypen som skal brukes er Calibri light med skriftstørrelse 12.
- Linjeavstanden skal være 1,5 cm.

I tillegg er det noen generelle tips hentet fra brukerhåndboken, hvor det er anbefalt:

- Bruk korte setninger.
- Lag korte avsnitt.
- Del stoffet opp i punkter så ofte som mulig.
- Dersom en skal beskrive komplisert stoff, sørg for at forklaringen gjentas på minst to forskjellige måter.
- Forklar de uttrykkene du bruker (f.eks. modul, prosess).

Våre retnignslinjer for språkbruk er:

- “Vi” istedenfor “Jeg”
- Aktivt istedenfor passivt språk

H.2 LaTex

Gruppen ønsker å bruke LaTex som sitt skrivebehandlingsverktøy på sluttinnleveringen. Ved å lage maler til alle tabeller og generelle dokumenter, er håpet at det vil totalt sett være tidsbesparende. Der tabellene er veldig kompliserte, vil det være mulig å gå vekk fra LaTex sin standard “tabular’’ metode, og i stedet forme tabellene i et annet verktøy før det limes inn i LaTex.

H.3 Struktur

Dette avsnittet gir et overordnet innblikk i strukturen på dokumentet som skal lages:

- Forside
- Sammendrag
- Innholdsfortegnelse
- Introduksjon
- Arbeidsmetoder
- Teknologier
- User Stories
- Produktdesign
- Implementering
- Evaluering
- Konklusjon
- Referanser
- Vedlegg

H.4 Dokumentbehandling

Alle skriftlige dokumenter skal lagres i Google Docs gjennom arbeidsperioden. Vi vil overføre dokumentasjonen fra Google Docs til LaTex ved innlevering.

Figurer, tabeller og bilder skal være lagret som PNG. Alle originalfiler skal være lagret i samme mappe som dokumentet filen er limt inn i.

H.5 Versjonskontroll

Det er viktig å sørge for versjonskontroll for å raskt kunne gå tilbake fra endringer dersom det er nødvendig. Det er opprettet versjonstabell i Google Docs som skal brukes ved versjonsendringer. Det skal komme frem i tabellen hvilke dokument som er endret, hvem som har endret det, og hva som er endret. Tabellen skal føre alle endringer som .0x versjon av nåværende. Versjonstabellen vil legges inn som et vedlegg, mens kun hele versjoner, som f.eks 1.2 eller 1.3 føres i versjonstabellen til sluttdokumentet.

H.6 Krav om korrektur

Korrektur gjennomføres ved at minst ett annet gruppemedlem leser over et annet gruppemedlems skriv. Det skal deretter noteres ned når korrektur er gjennomført. Dette noteres med

- Dato
- Hvem sitt arbeid som er sjekket
- Hvem som har sjekket
- Signatur

H.7 Kodestiler

Alle filer skal inneholde en header kommentar. I denne skal det legges ved en MIT lisens, og det skal føres opp navn på personen som har jobbet med koden. Dersom det har vært større bidrag fra en annen på gruppen skal det fremkomme ved at det kommenteres dennes persons navn over f.eks en funksjon laget av den hjelpende part. Der det ikke fremkommer klart og tydelig hva som skjer i koden, skal det lages et lite sammendrag for funksjonen, som forklarer forståelig hva funksjonen løser.

H.7.1 Header kommentar mal

/* MIT License

Copyright (c) 2018 NAVN HER

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE. */

Sprint 1 Review, Retrospective og planning

Sprint Review

Alle

- Lest seg opp på Scrum
- Egenopplæring innen C++
- Egenopplæring innen Node.js og Express
- Avklart definisjonen av Ready
- Avklart definisjonen av Done
- Laget system for product og sprint backlog
- Håndtering av JSON fra C++
- Research på hvordan legge inn metadata i videofil for synkronisering av video med resultat
- Gjennomført research på hvordan vi kan få nye inputfiler til å trigge at eksperiment kjøres.

Christian

- Laget dokumenter for user stories
- Overføring via ethernet

Anders

- Laget katalog over eksperimenter
- Laget autogenerering av UI ved bruk av JSON

Sprint Retrospective

Store deler av uken har gått til å sette opp Scrum og jobbet med prototypen som vi ønsker å ha klart til 2. presentasjon. Uken besto av mye selvstendig jobbing, noe som resulterte

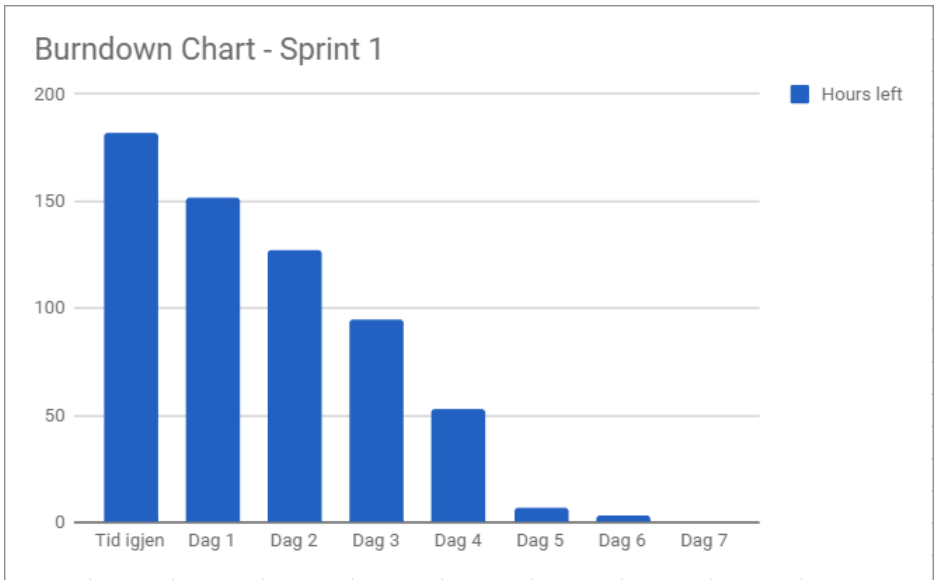
i at vi ikke hadde full oversikt over hva de andre drev med. I neste sprint ønsker vi å bli mer samstemte.

Sprint Planning

1. Alle skal brainstorme til innlevering
2. Alle skal lage slides til presentasjonen
3. Alle skal øve på presentasjonen
4. Alle skal ferdigstille og overlevere dokumenter
5. Alle skal fullføre prototypen så den kan vises til 2.presentasjon
6. Alle skal ha 2.presentasjon og se på Gruppe 16 sin presentasjon

Mesteparten av sprint 2 vil gå til å forberede dokumenter og klargjøre til 2.presentasjon

Burdown chart



Figur 37: Burndown sprint 1

Sprint 2 Review, Retrospective og Planning

Sprint Review

Alle

- Uken gikk til forberedelser og planlegging av 2.presentasjon
- Demoen viste at prototypen fungerte
- Presentasjonen + ettermøte avdekket følgende:
  - Vi trenger tydeligere dokumentasjon på hvem som har gjort hva
  - Vi må ta snapshots av Product Backlog ved hver grooming sesjon
  - Christian må ha et tyderligere teknisk bidrag.
  - Må dokumentere risikoer bedre:
    - Samarbeid mot andre grupper (hva har vi gjort for å redusere og hva bør vi gjøre?)
  - Hoveddokumentet må inneholde det vi føler er viktig, resten må være vedlegg eller tilgjengelig
  - Dokumentet må presentere produkt og teknisk bidrag
  - Må gjøre noe for å vise hva som er våre dokumenter (så de ikke blir stjålet"av andre grupper)
  - Teknologi dokumentet må ha mer om hvordan vi bruker ting:
    - Kode-retningslinjer?
    - Tydeligere på hvordan vi har valgt ting
  - Dokumentet må inneholde en versjonstabell som viser hvem som har gjort hva og når (bedre enn nå)
- Ettermøte med Joakim:
  - Dokumenter må ha bedre struktur med intro som forteller om bakgrunn og enkel forklaring ellers

  - REFERANSER

- Ifølge Emil og co var det en del overraskelser i presentasjonen.
  - Vi hadde egentlig dokumentert alt, men vi må bli tydeligere slik at dette kommer ordentlig fram.
  - Vi leverte mye dokumentasjon slik at ting druknet i mengden (med mye menes informasjon som ikke var nødvendig)
  - Tenk på hva vi skal ha med i presentasjonen når vi skriver og leverer dokuemtene så sensorene ikke blir overrasket
- Joakim skal ha en ukentlig gjennomgang og kopi av alle dokumenter vi har jobbet med

Sprint Retrospective

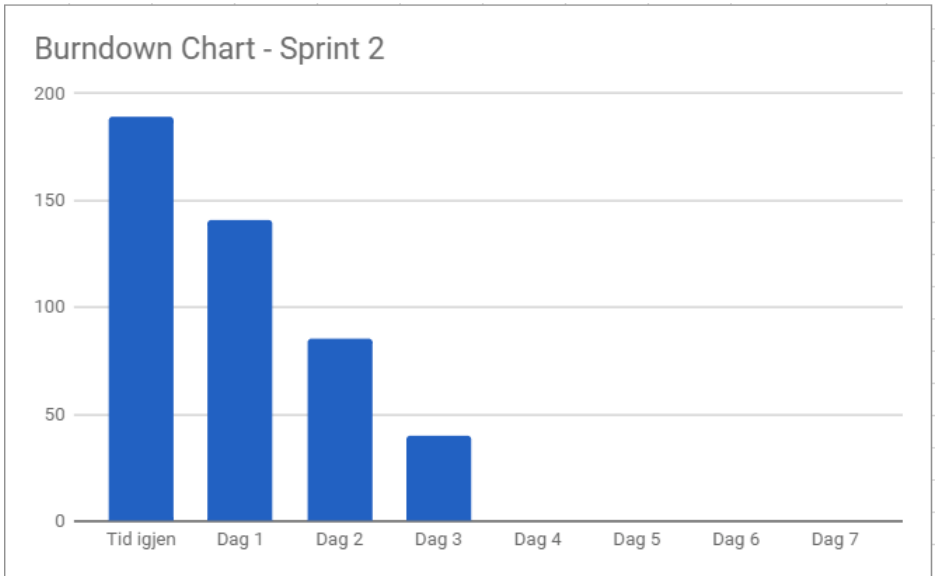
- Få hjelp av Radmila til å prøve å strukturere prosjektet (dette er nytt og kan dokumenteres som en risiko fordi vi gjør det så sent)
- Vi merker at det er vanskelig å koble alle aktiviteter til bestemte user stories, og at vi trenger å ta i bruk technical stories som gjør at vi kan lage en infrastruktur først før vi går videre til å ta tak i user stories direkte

Sprint Planning

1. Jan Helge skal sende mockup av Mathildes UI til henne så fort som mulig (helgen)
2. Christian skal fjerne gamle prosjekter og kode fra Bitbucket så vi får startet på nytt". Brainstorming av prosessen på dette
3. Christian skal lage Core løsning for mikroservice
4. Alle skal finne ut hva som skal være med i hoveddokumentet
5. Alle skal groome Product Backlog på mandag og lage technical stories for å lage en infrastruktur før vi går løs på de enkelte user stories.

6. Espen skal se på tidligere prosjekter mtp hva som er skrevet om bakgrunn for oppgaven (andre remote labs, forskning om nytten av remote labs osv)
7. Anders skal refaktorisere og rydde Express koden (fra prototypen)
8. Christian skal splitte prototypen i deler som kan fordeles på gruppen
9. Espen skal ta egenopplæring i C++

Burdown chart



Figur 38: Burndown sprint 2

Sprint 3 Review, Retrospective og Planning

Christian:

- Pratet med Radmila og har nå klart UML diagrammene
- Lagt til repository til alle delkomponentene
- Begynt arbeidet med nye prosjekter og gammel kode som skal brukes er flyttet over

Jan Helge:

- Laget mockup av websiden og fått tilbakemeldinger fra både studenter og veiledere (Joakim og Dag)
  - Må gjøre visuelle endringer
- Litt bedre tid på oss da masterstudent (webside - brukervennlighet) har utsatt masteroppgaven
- Mockup er lagt inn i versjonskontroll og lastet opp på nett
- Utkast av forslag til samtlige dokumenter som skal leveres er laget
  - Fortsatt rom for forbedringer/endringer
- Dokumentpolicy dokumentet er påbegynt og trenger omskrivning slik at det blir et mer tekstbasert dokument
- Påbegynt ny versjon av protokollen. Den er fortsatt i en tidlig startfase

Anders:

- La til støtte for enums og booleans input i grensesnittet
- Refaktorert koden sin

Espen:

- Påbegynt ny versjon av køsystemet. Den er fortsatt i en tidlig startfase
- Researchet alternative løsninger til køsystemet

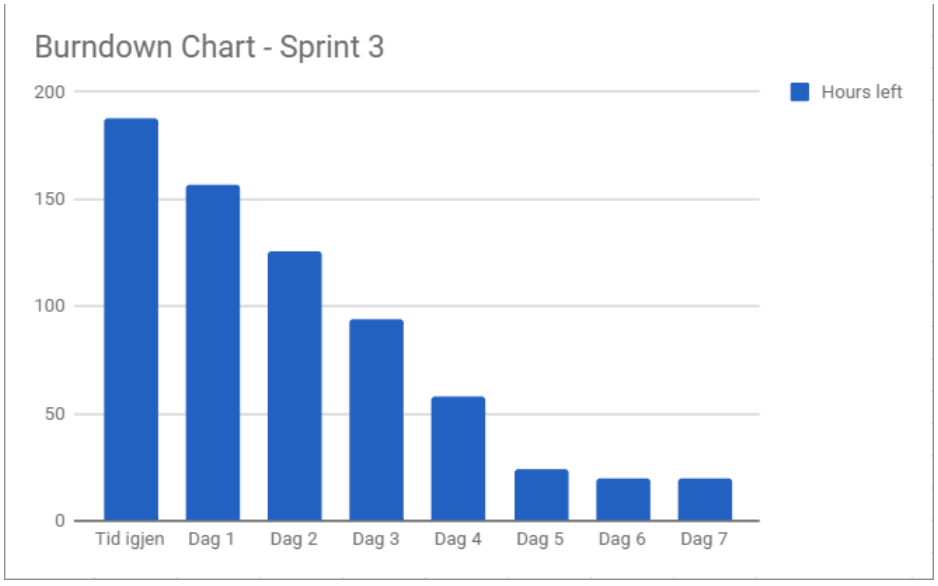
Sprint Retrospective

Sprint goal for denne uken følger ikke definisjonen av Done fordi vi ikke er ordentlig i gang med ”vanligesprinter

Sprint Planning

1. Alle skal dokumentere arbeid sitt fortløpende og skrive om teknologier de bruker.
2. Christian skal jobbe aktivt med core systemet til microservicen
3. Anders skal jobbe med resultatvisning (grafer)
4. Anders skal jobbe med resultatvisning (tabeller)
5. Alle skal groome PB
6. Jan Helge skal oppdatere hoveddokumentet ved behov
7. Espen skal jobbe med køsystemet
8. Jan Helge skal jobbe aktivt med protokollen

Burdown chart



Figur 39: Burndown sprint 3

Spring 4 Review, Retrospective og Planning

Anders:

- Researchet javascript biblioteker som kan tegne grafer
- Researchet Node.js moduler for filewatching
- La til funksjonalitet for å vise resultater som grafer på front-end
- La til funksjonalitet for å vise resultater som tall på front-end
- Dokumentert om implementasjonen av front-end
- Dokumentert om teknologier brukt på front-end

Espen:

- Researchet hvordan man skriver/sletter/flytter filer i C++
- Researchet forskjellige løsninger til implementering av køsystemet
- Gjennomgang av prototype-koden (filewatcher)

Jan Helge:

- Jobbet med protokollen og implementert dette inn i Core
- Jobbet sammen med Christian med å fullføre Core
- Laget disposisjon til hoveddokumentet for prosjektet
- Researchet innloggingssystemer i Node.js mot mongoDB

Christian:

- Laget Core system av løsningen, som resten av gruppen skal implementere sine systemer mot
- Jobbet sammen med Jan Helge for å fullføre Core

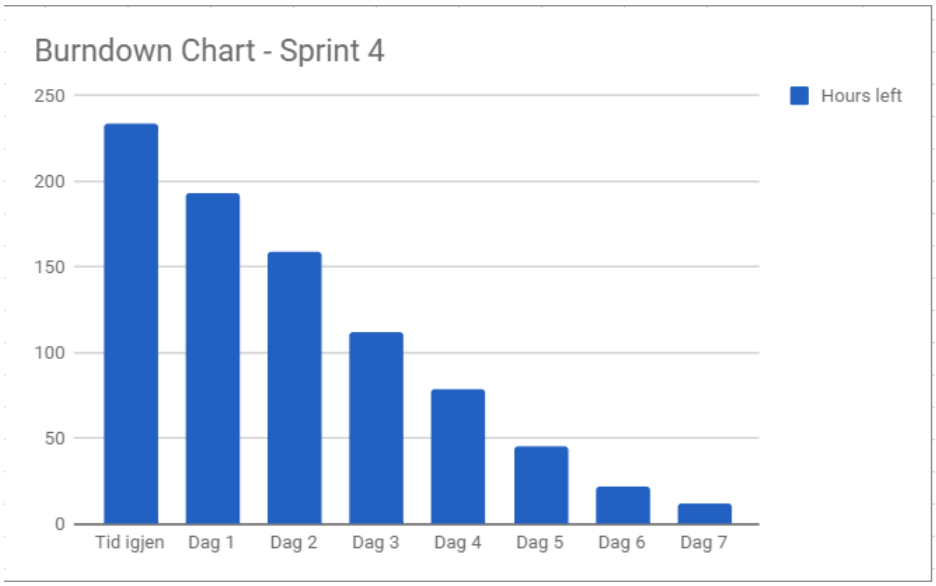
Sprint Retrospective

Vi må skrive hvem som har gjort hva i sprint reviewen

Sprint Planning

1. Christian skal fullføre UML diagrammene (use case diagrammer, sekvens diagrammer og komponent diagrammer)
2. Anders skal skrive om tester
3. Anders skal lage dokument for testrapporter
4. Jan Helge skal jobbe med nettverk
5. Anders skal jobbe med å håndtere flere brukere på front-end (visning av resultat)
6. Christian og Jan Helge skal fullføre Core
7. Espen skal skrive om køsystemet
8. Jan Helge skal jobbe med front-end nettsiden (login og annen funksjonalitet)
9. Jan Helge skal skrive om front-end nettsiden og innloggingsystemet
10. Anders skal skrive om implementasjon av front-end
11. Alle skal dokumentere teknologier vi har brukt
12. Espen skal jobbe med køsystemet
13. Jan Helge skal dokumentere protokoll og interface
14. Christian skal dokumentere implementasjon av mikroservice
15. Christian skal dokumentere prosessen

Burdown chart



Figur 40: Burndown sprint 4

Sprint 5 Review, Retrospective og Planning

Anders:

- Har fikset error: alle brukere ser samme fil
- Har påbegynt dokumentasjon om implementasjonen av front-end
- Har påbegynt dokumentasjon om tester
- Har dokumentert teknologier han har brukt

Espen:

- Har jobbet med køsystemet
- Researchet alternative løsninger til køsystem
- Påbegynt dokumentasjon om køsystem

Jan Helge:

- Dokumentert teknologier han har brukt
- Jobbet med Christian med Core - Debugging og integrering
- Implementert login og registreringssystem til front-end
- Påbegynt dokumentasjon om protokoll og interface

- Påbeyngt dokumentasjon om nettsiden

Christian:

- Jobbet med Jan Helge med Core - Debugging og integrering
- Jobbet med UML diagram
- Påbegynt dokumentasjon om prosessen
- Forbedret core

Sprint Retrospective

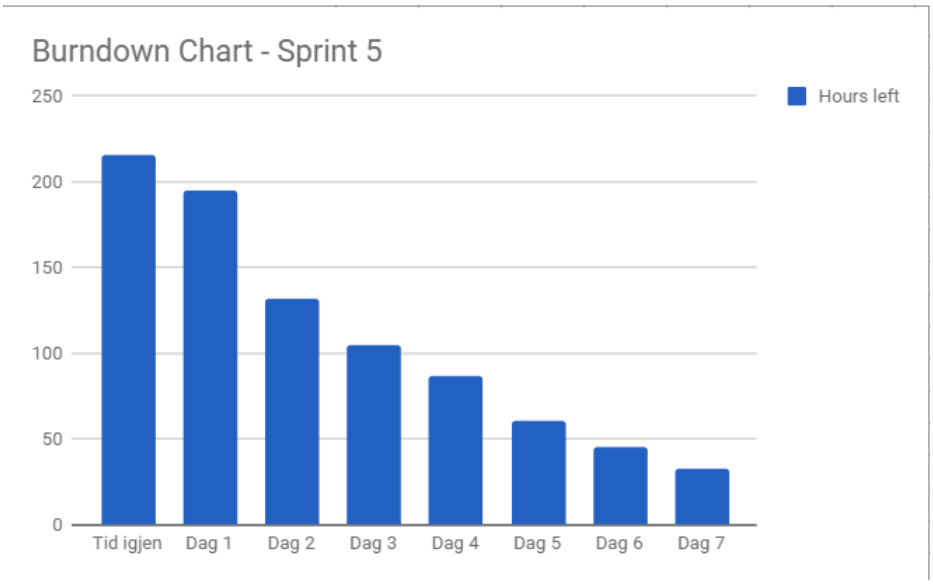
- Må linke planning med sprint ID for sporbarhet
- Timelister i tillegg til burndown chart
- Skrive om hvorfor, hvis vi feilberegner tid

Sprint Planning

1. Anders fullfører dokumentasjon om UI (implementasjon)
2. Anders fullfører dokumentasjon om tester
3. Anders, Christian fullfører dokumentasjon om teknologier
4. Anders fullfører dokumentasjon om user stories
5. Anders, Espen og Jan Helge gjennomfører tester
6. Espen bearbeider og fornyer introduksjon
7. Espen fullfører dokumentasjon om køsystem
8. Espen fullfører dokumentasjon om kravtabeller
9. Espen fullfører dokumentasjon om møtereferat
10. Christian fullfører dokumentasjon om Core
11. Christian fullfører dokumentasjon om prosess

12. Christian fullfører dokumentasjon om UML
13. Christian fullfører dokumentasjon om arkitektur
14. Christian fullfører dokumentasjon om backlog og review
15. Jan Helge fullfører dokumentasjon om nettverk
16. Jan Helge fullfører dokumentasjon om nettsiden
17. Jan Helge fullfører dokumentasjon om protokoll og interface
18. Jan Helge fullfører dokumentasjon om risiko
19. Jan Helge kartlegger hva som gjenstår av dokumentasjon og delegerer oppgaver ut på oppgaven
20. Alle fullfører dokumentasjon om evaluering

Burdown chart



Figur 41: Burndown sprint 5

Spring 6 Review, Retrospective og Planning

Anders:

- Ført inn teknologier i latex
- Ført inn dokumentasjon om UI
- Lagde, utførte og dokumenterte tester
- Skrev dokumentasjon og førte inn user stories med referanse linker

Espen:

- Fornyet og ført inn introduksjon dokumentet
- Ført inn dokumentasjon om køsystem
- Ført inn dokumentasjon om krav
- Ført inn dokumentasjon om møtereferat

Jan Helge:

- Ført inn teknologier i latex
- Ført inn dokumentasjon om nettverk
- Ført inn dokumentasjon om protokoll og interface
- Ført inn dokumentasjon om nettsiden
- Ført inn diverse dokumentasjon (små avsnitt som ikke var delegert)
- Ført inn dokumentasjon om risiko
- Laget idekart over evaluering

Christian:

- Ført inn dokumentasjon om prosess
- Ført inn noe dokumentasjon om UML
- Ført inn dokumentasjon om teknologier

Sprint Retrospective

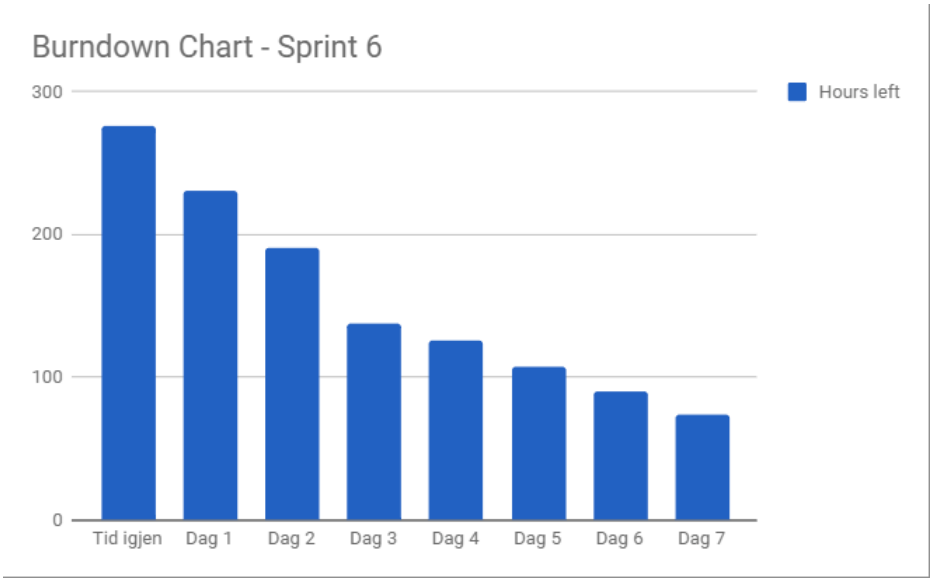
- Det kan oppstå problemer med versjonskontroll verktøyet. Dette hadde vi ikke tatt hensyn til og det kostet oss mange timer
- Mangelfull research kan føre til ekstra arbeid. Som referering av tabeller i latex.
- Dokumentering/innføring av dokumentasjon tar lengre tid enn forventet, og det er vanskelig å forhåndsplanlegge tidsbruk

Sprint Planning

1. Anders skal sørge for validering av ugyldige valg på eksperimentsiden
2. Anders og Jan Helge skal sørge for mulighet til å streame video av et eksperiment
3. Anders skal implementere kakediagram
4. Anders skal lage en utviklernettside som brukes ved oppkobling av nytt eksperiment
5. Anders skal skrive om utviklerverktøy og utviklernettside
6. Anders skal lage et system som viser ventetid til brukeren dersom det er kø for å bruke eksperimentet
7. Espen skal integrere køsystemet i Core
8. Espen skal sørge for at systemet sjekker responstid
9. Espen skal legge inn Backlog og Reviews i latex
10. Christian skal fullføre dokumentasjon om Core
11. Christian skal fullføre UML
12. Christian skal få Core over på linux
13. Christian skal ferdigstille Core
14. Jan Helge skal ferdigstille designet av nettsiden sammen med masterstudent
15. Jan Helge skal ferdigstille protokollen. Legge inn pakker vi mangler

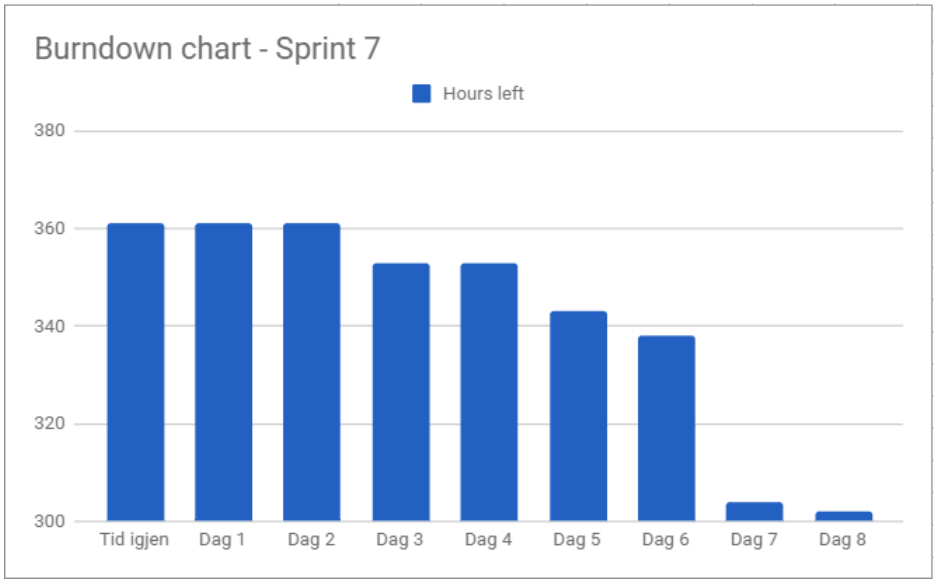
16. Jan Helge skal gjøre nettsiden ”mobile first”
17. Jan Helge og Anders skal lage profilside med historikk
18. Jan Helge skal integrere nettverk med Core
19. Jan Helge skal lage en sjekk som sjekker om bruker finnes ved registrering av ny bruker
20. Alle skal jobbe med evaluering
21. Alle skal gjøre tester
22. Alle skal lage håndbok
23. Alle skal jobbe med konklusjon
24. Alle skal jobbe med tilbakemeldinger fra veileder
25. EXTRA: Drag and drop tegneverktøy
26. EXTRA: OAuth med Facebook/Google

Burdown chart



Figur 42: Burndown sprint 6

Burdown chart



Figur 43: Burndown sprint 7

User manual for USNs Remote Lab

By Christian Scott and Nicolas Ekholdt

V 1.0 – May 22nd 2018

Introduction

This user manual provides a brief description on how to create a remote experiment that can be integrated in USN's Remote Lab system. It includes the following information:

- Suggestions on how to construct a Remote Experiment
- Usage of RELEASE modules to simplify and speed up construction
- How to integrate remote experiments with the software solution

How to Construct a Remote Experiment

A remote experiment can be constructed in several ways. The simplest solution would be to use an existing experiment or a kit. More advanced users might want to 3D-print their own components or use a laser cutter to produce parts in steel or plywood.

There are, however, a few important things to take into consideration.

Remote experiments are unmanned, and this has certain ramifications on how they should be designed. They must reset themselves between experiments, and no-one is necessarily around to correct an abnormal situation such as a part getting stuck or falling outside the experiment area. Experiments that require refilling might also be impractical.

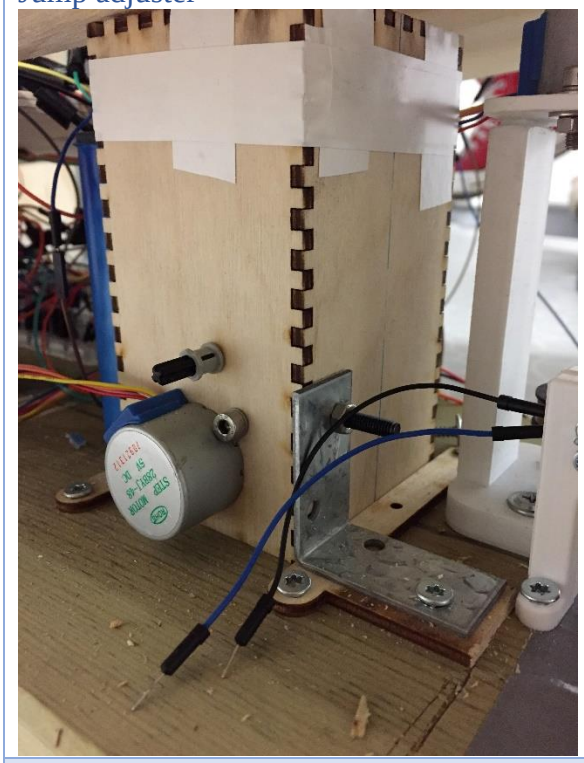
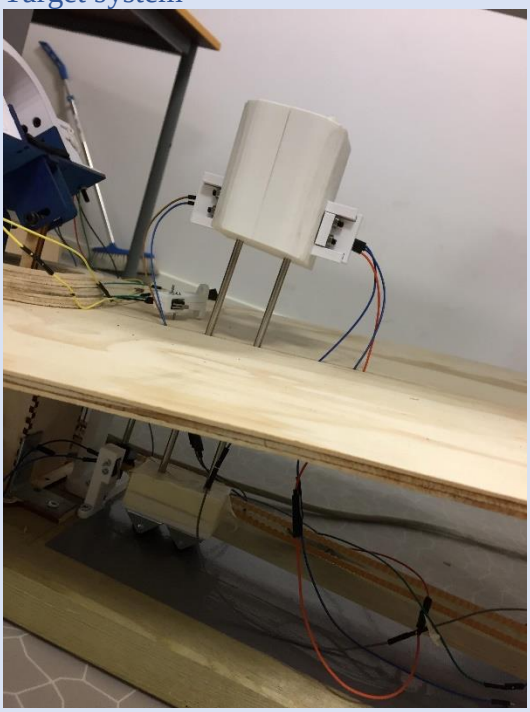
Their availability might mean that they are used more often than a regular experiment. This places higher demands on durability and part quality.

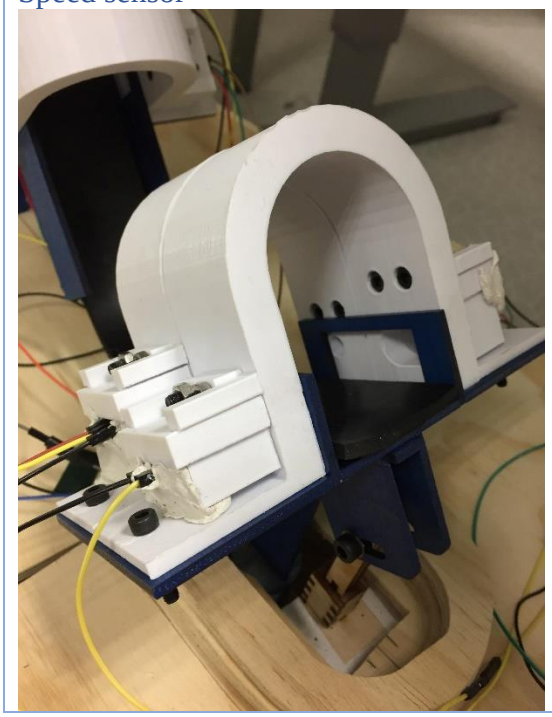
Finally, due to the facts already mentioned and because every setting and control requires some form of automation, remote experiments can be much more complex than their regular counterparts. This requires careful thought into how many capabilities it is practical to include in the experiment.

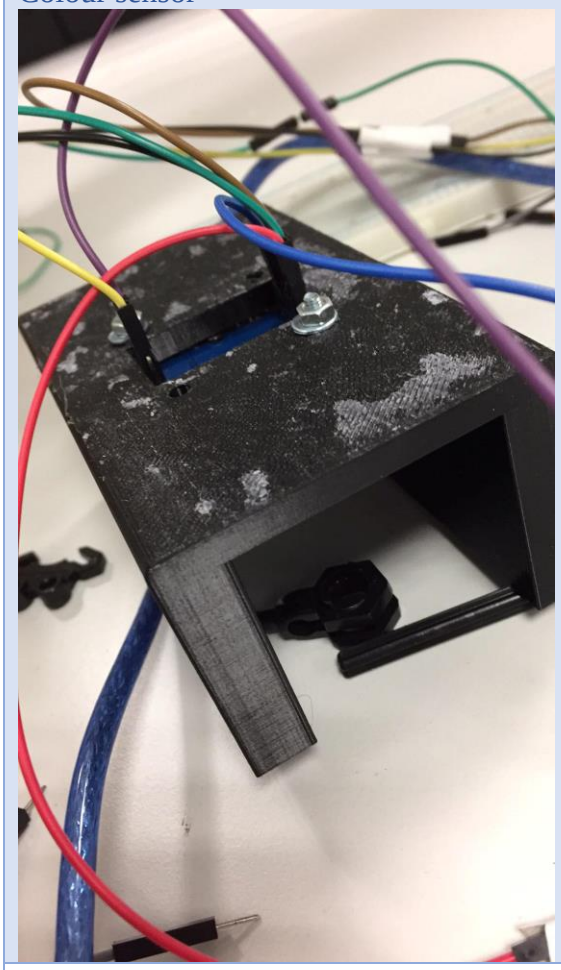
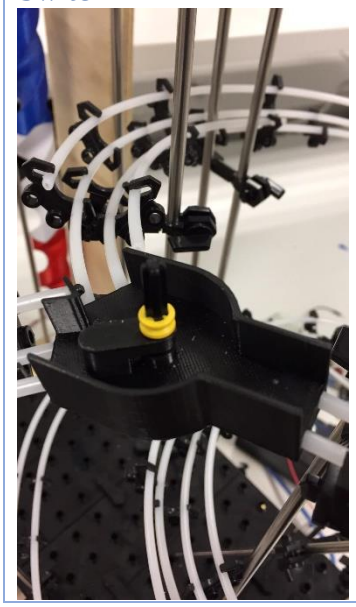
RELEASE Modules

USN has developed several modules to make it easier to construct a remote experiment.

Modules produced	Acquired mechanical purpose
Elevator 	A module able to use the rotational movement from a motor to an elevator able to lift balls.

Jump adjuster 	A module able to convert the rotational movement of a motor to linear motion. Can be used on experiments which requires a linear motion.
Target system 	A module able to convert the rotational movement of a motor to linear motion over a further distance than the jump adjuster module.

Speed sensor 	A module able to calculate speed of an object passing through with two laser and receiver pairs. Can be used in all experiments requiring a measurement of speed.

Colour sensor 	This colour sensor could be used in other specific experiments requiring a sensor capable of telling the difference in colour on an object passing through.
Switch 	The switch module provides a general way of sorting balls or marbles to a desired path.

Integrating Experiments with Remote Lab Software

Experiments connect to the Remote Lab Software via Ethernet and TCP/IP. The controller between the experiments must implement the purpose-designed protocol, and act as a server that can wait for incoming requests from the server.

The general design of the protocol is as follows:

Packet Type ID – 1 byte	Total Packet Size – 2 bytes	Payload – Variable number of bytes

Note that certain packets do not include size or payload. This is documented below.

Packet Types

Packet Type ID	Description
0x00	Discovery Request
0x01	Discovery Response Header
0x02	Discovery Response Parameter
0x03	Video Sync Marker
0x04	Input
0x05	Output
0x06	Output w/Channel
0x07	Start Marker
0x08	End Marker
0x09	Service
0x10	Experiment Run ID

Discovery Request, Discovery Response Header and Parameter

These packets are for an advanced auto-discovery functionality that is not covered in this version of the manual.

Video Sync Marker

If needed, a video sync marker consisting of a single byte (the packet type ID) can be sent be the experiment to the server. This marker will be inserted into the output data stream and can be used to synchronize a video recording with the data. This is an advanced subject and requires software that can insert metadata into the video stream to mark where the data and video should be synchronized.

Input/Output

Input and output to and from the experiment is sent as a payload representing the values. The order, size and type of each value must be described in a JSON configuration file on the server. Note that an output packet does not include the size field. This is to reduce the protocol overhead in situations with large output volumes

Output w/Channel

Channels can be used for an experiment that requires several data output streams. This might be desired if one stream is sent at a higher frequency than the other.

Start Marker

This packet includes the 8-bit string “START”. It is sent by the experiment to signal when the experiment starts. This is useful to notify the user/front-end system when the experiment starts in case of a queue situation.

End Marker

The end marker is sent at the end of the experiment to signal that there is no more output. It includes the 8-bit string “STOP”.

Service Marker

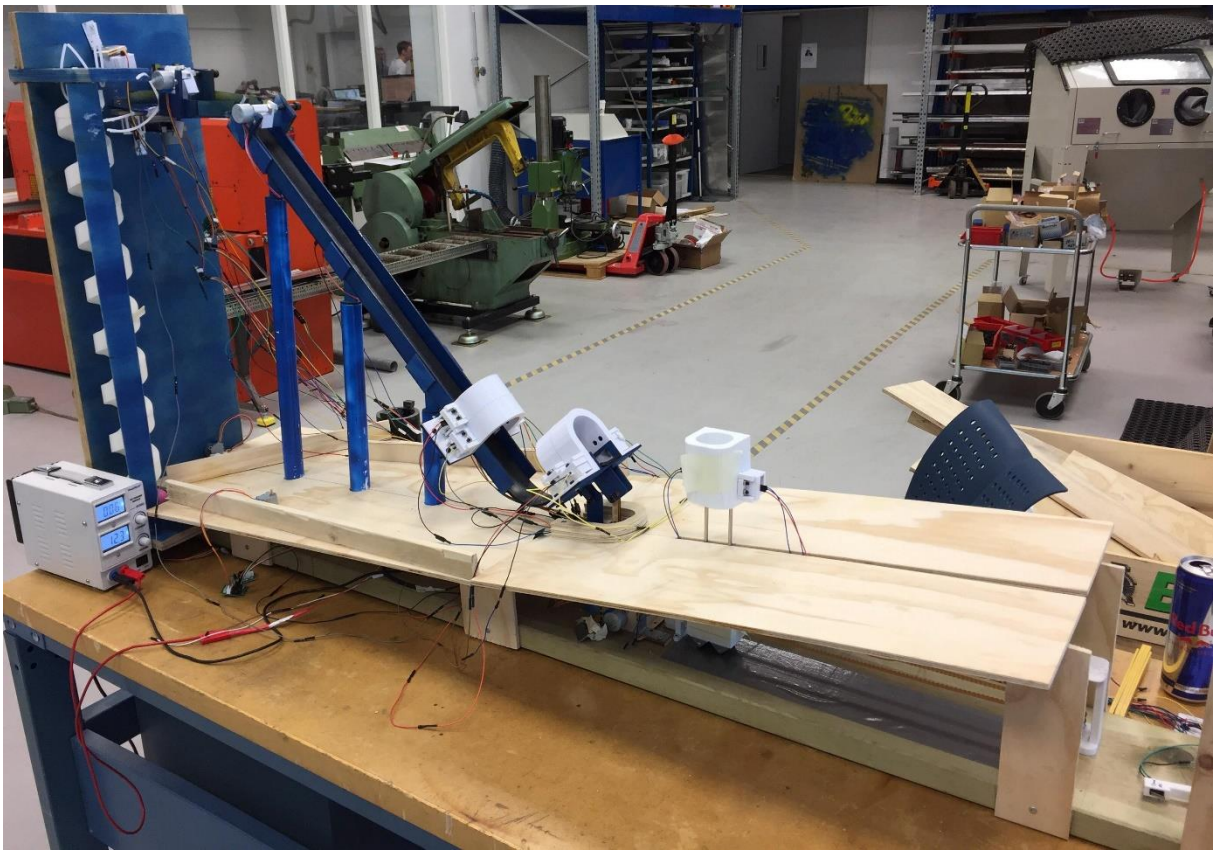
The service marker is sent to the experiment to trigger service events such as calibration or other actions the experiment wants to perform outside regular usage.

Run ID

The run ID isn’t sent to the experiment, and is used internally by the system to associate a certain run to its data.

Constructed Experiments

Experiment 1.



Purpose:

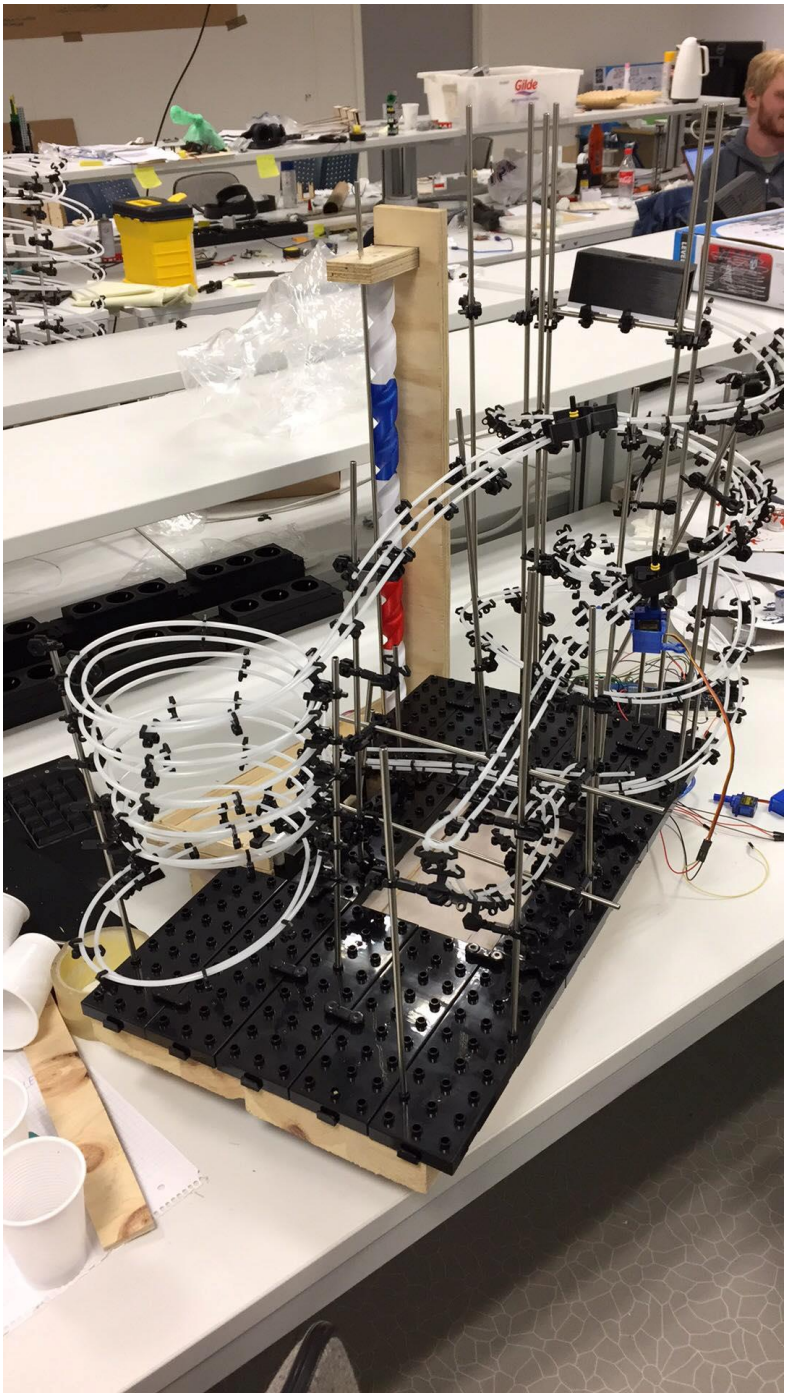
The purpose of Experiment 1 is to calculate how far a ball will be able to go from a jump construction, by knowing the angle of the jump and the type of ball which is launched.

The input and output from Experiment 1 are explained here.

Input parameter	Description
Jump angle.	The angle of the jump can be adjusted, to degrees between 30 and 60 with five degrees intervals in between. This is done to change the distance the ball will be launched.
Target position.	The position of the target can be moved to enable the Remote Lab user to guess or calculate the distance which the ball will be launched.
Ball selection.	The Remote Lab user can choose between two balls with different properties to see the difference in results in Experiment 1.

Measured results	Description
The speed before the curvature.	The Remote lab user will gain the speed of the ball before it reaches the curvature connecting the ramp and jump. This is done to verify that the calculations to this point are correct.
The speed at the jump.	The speed of the object will be measured at the jump. This is the speed at which the ball will be launched into the air with, meaning that the Remote lab user can skip the calculation for speed at the jump as wanted. This speed sensor also enables the user to verify if their calculations to this point are correct or to see the deviations from speed coupled with the result of the first speed sensor.
Registered hit in the target.	The Remote Lab user will be notified if the ball hits the target, confirming that their calculations are correct.

Experiment 2.



Purpose:

The purpose of Experiment 2 is to see how a colour sorting sensor coupled with switches can be used to sort different coloured balls as desired.

The input and output parameters for Experiment 2 are:

Input parameter	Description
Code to sort balls as desired.	The input to this experiment will be a software code able to sort the coloured balls as desired. The switches are in place to ensure usability from the input.

Output result	Description
Colour sensor	A sensor able to detect the colour of the ball passing through.