Bacheloroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

Martinsen, Herman Ryen
Nicolausson, Sander
Rondestvedt, Trond Jacob
Aasvestad, Jørgen

Utvikling av brukergrensesnitt og brukerfunksjoner for Sensorfisk

Bacheloroppgave i Dataingeniør
Veileder: Nilsen, Jan Harald

Mai 2020

NTNU
Kunnskap for en bedre verden

Martinsen, Herman Ryen
Nicolausson, Sander
Rondestvedt, Trond Jacob
Aasvestad, Jørgen

Utvikling av brukergrensesnitt og brukerfunksjoner for Sensorfisk

Bacheloroppgave i Dataingeniør
Veileder: Nilsen, Jan Harald
Mai 2020

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

NTNU
Kunnskap for en bedre verden

Forord

Denne bacheloroppgaven er utarbeidet ved Institutt for Datateknologi og Informatikk ved Norges Teknisk- Naturvitenskapelige Universitet i Trondheim, og er skrevet for SINTEF Ocean AS.

Oppgaven omhandler fiskeri og oppdrett som er en av Norges største næringer, og ble valgt fordi vi ønsket å bidra til å kunne gjøre en forskjell innenfor dette feltet. Store deler av prosjektperioden har blitt annerledes enn vi hadde sett for oss grunnet utbruddet av COVID 19-viruset. Til tross for dette har arbeidet med prosjektet fungert bra, og resultatet har blitt som vi ønsket.

Vi vil gjerne takke veileder Jan Harald Nilsen for god hjelp gjennom hele prosjektperioden, og Sveinung Johan Ohrem, Birger Venås og Elina Willert ved SINTEF Ocean AS for en god dialog under utviklingen. Til slutt vil vi takke alle som tok seg tid til å delta på brukertestene som ble gjennomført i løpet av prosjektperioden.

Herman Ryen Martinsen

Sander Nicolausson

Trond Jacob Rondestvedt

Jørgen Aasvestad

19.05.2020

Trondheim

Oppgavetekst

Oppgaven består av å utvikle et grafisk brukergrensesnitt for behandling og visualisering av data fra SINTEF Ocean AS' Sensorfisk. Brukergrensesnittet skal inneholde brukerfunksjoner som blant annet tillater filtrering av data, sammenligning av forskjellige gjennomkjøringer og eksportering av grafiske filer for bruk i andre program. Alle funksjonelle avhengigheter beskrives i visjonsdokumentet (se vedlegg B, kapittel 5). User stories som omfatter disse funksjonelle avhengighetene er beskrevet i kravdokumentet (se vedlegg C, kapittel 2).

Den initielle oppgavebeskrivelsen inneholdt også en del som omhandlet å undersøke mulighetene for å anvende maskinlæring eller Big Data-analyse på den innsamlede dataen, men etter avtale med oppgavestiller ble denne delen sløyfet.

Sammendrag

I denne bacheloroppgaven har vi utviklet et web-basert brukergrensesnitt med brukerfunksjoner for behandling og visualisering av data fra SINTEFs Sensorfisk. Sensorfisken er et sylinderformet måleinstrument som skal etterligne en oppdrettsfisk, og som samler inn data om fiskens forhold inne i et mekanisk avlusersystem.

Per i dag er det ikke gjort forskning på hvordan transport og behandling i rørsystemer påvirker oppdrettsfiskens velferd, men håpet med Sensorfisken i samspill med vårt produkt er å kunne samle inn et datagrunnlag for denne forskningen. Vårt system kan også brukes av SINTEFs kunder for å sammenligne gjennomkjøringer med forskjellige innstillinger på behandlingsenhetene, og dermed bestemme hvilke innstillinger som er best egnet.

Sluttproduktet effektiviserer prosessen fra Sensorfisk kjøres gjennom en behandlingsenhet til kunden har en ferdig rapport tilgjengelig. Aspekter som påvirker denne effektivitetsøkningen er at hele prosessen er samlet i et system, samt at omgjøring fra rådata til lesbar data går raskere. Produktet inneholder også et brukervennlig grensesnitt som brukeren kan interagere med, og som tillater visualisering og filtrering av data.

Innholdsfortegnelse

Figurliste

Tabelliste

1 Introduksjon og relevans

Fisk har utviklet seg til å bli en av de største eksportvarene i Norge (Statistisk sentralbyrå, 2020). Langs norskekysten finner vi en rekke oppdrettsanlegg som forsyner store deler av verden med fisk. Etter hvert har det også blitt mer og mer fokus på oppdrettsfiskens velferd. I hvor stor grad påvirker de mekaniske behandlingsenhetene fiskens velferd? Blir fisken utsatt for kraftige støt, høye temperaturer eller høyt trykk som kan innvirke negativt på fiskens helse? Per i dag finnes det svært få muligheter for å måle forholdene fisken utsettes for. Derfor kan ny teknologi som gjør dette mulig være svært interessant for aktørene i oppdrettsindustrien.

1.1 Bakgrunn

SINTEF Ocean AS er et norskt forskningsinstitutt som driver med forskning og innovasjon knyttet til havrommet for nasjonal og internasjonal industri. De har gjennom sitt prosjekt KVALISYS forsøkt å lage en standardisert metodikk for kvalifisering av mekaniske avlusingssystemer (Fiskeri- og havbruksnæringens forskningsfond, 2020). I dette prosjektet har de utviklet et produkt som kalles “Sensorfisk”. Dette er et sylinderformet målingsinstrument som består av en mikrokontroller og flere sensorer som registrerer data. Formålet med Sensorfisk er å etterligne en fisk som blir ført gjennom et mekanisk avlusersystem, og dermed påvise hvilken behandling den får på forskjellige steder i systemet. Relevante forhold som fanges opp er blant annet trykk, temperatur og sammenstøt med rørvegger.

Dagens løsning er å benytte Python-scripts som laster inn, behandler og visualiserer data fra Sensorfisk, uten noen form for brukergrensesnitt. SINTEF Ocean AS beskriver denne løsningen som noe tungvint, da det ofte er snakk om store datamengder og kunder med ulike ønsker angående hvilke data som skal presenteres. Derfor blir det mye manuelt arbeid i etterkant av hvert oppdrag med Sensorfisk. SINTEF Ocean AS ønsker derfor å få utviklet et nytt system som tar høyde for problemene med dagens løsning.

1.2 Oppgavebeskrivelse

Oppgaven går ut på å utvikle et nytt system som SINTEF Ocean AS kan bruke til opplasting, analysering og visualisering av dataene fra Sensorfisk. Systemet skal være web-basert, og ha et grafisk brukergrensesnitt som er enkelt og brukervennlig. Informasjon og grafer tilknyttet en gjennomkjøring med Sensorfisk skal være lett tilgjengelig i systemet. Det skal være mulig å filtrere grafene og sammenligne grafer fra ulike gjennomkjøringer. Grafene skal også kunne eksporteres for bruk i andre programmer. Informasjon om kunder og behandlingsenheter skal også lagres i systemet, og det skal være mulig å behandle denne informasjonen gjennom brukergrensesnittet.

Det skal også utvikles en serverbasert, skalerbar database hvor all relevant informasjon i systemet lagres. Denne må kunne håndtere store datamengder, da sensorene på Sensorfisk har høy samplingsfrekvens og genererer mye data.

Den initielle oppgavebeskrivelsen inneholdt også en del som omhandlet å undersøke muligheten for å anvende maskinlæring eller Big Data-analyse på den innsamlede dataen. På oppstartsmøtet ble det bestemt at dette punktet skulle utgå, og at fokuset skulle ligge på utviklingen av systemet.

1.2.1 Avgrensning

I starten av prosjektet ble det bestemt at det web-baserte grensesnittet først og fremst skulle utvikles med tanke på intern bruk for ansatte i SINTEF Ocean AS (se vedlegg E, kapittel 2.1). Funksjonalitet hvor kunder av SINTEF Ocean AS kan logge seg inn i systemet og se egne data var likevel ønskelig, så fremt det lot seg gjøre innenfor tidsrammene til prosjektet. For en gjennomkjøring ønsket oppgavestiller hovedsakelig å ha grafer for trykk, temperatur og sammenstøt med rørvegger, da dette var de mest relevante dataene fra en gjennomkjøring.

1.2.2 Forskningsspørsmål

I denne rapporten ønsker vi å besvare følgende forskningsspørsmål:

1. Hvilken teknologi bør tas i bruk for å lage et system som er mer brukervennlig enn dagens løsning?
2. Hvordan effektivisere prosessen fra Sensorfisk kjøres hos oppdretter til en rapport blir levert til kunden?

3. Hvordan bør sikkerhet implementeres i systemet, slik at uvedkommende ikke får tilgang til sensitiv informasjon?
4. Hvordan få til å visualisere data på en interaktiv men uforstyrrende måte?

1.3 Prosjektmål

Prosjektmålene er satt opp for å gi en konkret beskrivelse av formålet med prosjektet. De er ment som retningslinjer under arbeidet, og gir en felles forståelse av hva oppgaven går ut på. De brukes også til å styre og måle prosjektet underveis, og til å vurdere om de ønskede resultatene ble oppnådd i ettertid.

1.3.1 Effektmål

1. Gjøre behandlingen av data fra Sensorfisk enklere og mer effektiv for de ansatte hos SINTEF Ocean AS.
2. Gjøre det enklere for SINTEF Ocean AS sine kunder å få tilgang til, behandle og visualisere data fra egne anlegg.
3. Gjøre produktet Sensorfisk mer attraktivt for kunder av SINTEF Ocean AS.
4. Bidra til å samle inn nok data til å utvikle metoder for å kartlegge fiskens fysiske forhold i rørtransport.

1.3.2 Resultatmål

1. Utvikle et web-basert brukergrensesnitt for visualisering og behandling av data fra Sensorfisk, som både SINTEF Ocean AS og deres kunder har tilgang til.
2. Utvikle nødvendige brukerfunksjoner for å kunne ta nytte av dataene som ligger lagret i systemet (se vedlegg C, kapittel 2).
3. Utvikle en serverbasert, skalerbar database for håndtering av store datamengder.

1.3.3 Læringsmål

1. Få mer kunnskap og erfaring innen utvikling av store webprosjekter, som kan være relevant i det fremtidige arbeidslivet.
2. Få mer erfaring med bruk av smidige utviklingsmetoder.
3. Få erfaring med hvordan man som utviklerteam skal forholde seg til et stort firma som SINTEF Ocean AS som oppgavestiller.

1.4 Målgruppe

Denne rapporten vil være interessant for SINTEF Ocean AS, som kan ha nytte av å se hvilke valg som har blitt gjort under utviklingen og hvilke utvidelser som kan være aktuelle i fremtiden. Rapporten kan også være interessant for studenter som skal gjennomføre lignende prosjekter, eller andre personer med interesse for teknologi og webutvikling.

1.5 Rapportstruktur

Rapporten er delt inn i 6 hovedkapitler:

1. **Introduksjon og relevans** - Kort introduksjon til oppgaven. Inneholder blant annet oppgavebeskrivelse, forskningsspørsmål, prosjektmål, målgruppe og rapportstruktur.
2. **Teori** - Det teoretiske grunnlaget for arbeidet som har blitt gjort.
3. **Valg av teori og metode** - Begrunnelse av valgene som har blitt gjort i prosjektet. Omhandler blant annet valg av programmeringsspråk, teknologi og utviklingsprosess.
4. **Resultater** - Resultatene vi har kommet frem til gjennom prosjektarbeidet. Disse deles inn i empiriske, ingeniørfaglige og administrative resultater.
5. **Diskusjon** - Drøfting av resultater og arbeid i forhold til et helhetlig systemperspektiv.
6. **Konklusjon og videre arbeid** - Konklusjoner knyttet opp mot forskningsspørsmålene, og beskrivelse av mulige fremtidige utvidelser.

Etter disse hovedkapitlene kommer referanser og vedlegg. Vedlegg A - ordlisten - inneholder forklaringer av ord, begreper, akronymer og forkortelser som forekommer i rapporten og de andre vedleggene, og kan være nyttig for leseren for å få en bedre forståelse av innholdet.

2 Teori

Teorien beskrevet i dette kapittelet legger grunnlaget for arbeidet gjort i prosjektet, samt videre lesing i rapporten.

2.1 Brukervennlighet

I dagens samfunn interagerer mennesker med teknologi på daglig basis. Bussbilletter kjøpes via en app, regninger betales i nettbanken og skattemeldingen leveres digitalt. Det er derfor vesentlig at disse systemene er utformet på en slik måte at de er enkle og intuitive i bruk, og at brukerne føler at de mestrer dem.

2.1.1 Menneske-maskin-interaksjon

Menneske-maskin-interaksjon (MMI) er et fagområde som handler om hvordan mennesker interagerer med datasystemer. Både datateknologi, psykologi, ergonomi, design, ingeniørvitenskap og grafisk design er disipliner som bidrar til MMI (Stone, Jarrett, Woodroffe & Minocha, 2005, s. 3). Et velkjent område innen MMI er bruken av *mentale modeller* (Carroll, 2003, s. 137). Dette innebærer at man tar utgangspunkt i noe som eksisterer i den virkelige verden, og former en funksjonalitet etter dette slik at en bruker intuitivt vil forstå hvordan funksjonen virker. Som eksempel kan vi bruke papirkurven i moderne operativsystemer. Papirkurven er egentlig bare en mappe i filsystemet hvor filer som er slettet av brukeren midlertidig lagres før de slettes permanent. Bruken av et ikon som er utformet som en søppelbøtte gjør det enklere for brukeren av operativsystemet å forstå, da brukeren allerede har en mental modell fra det virkelige liv om hvordan en søppelbøtte fungerer.

2.1.2 Brukergrensesnitt

Det grensesnittet som brukes for å interagere med et system kalles brukergrensesnitt. Tidlige brukergrensesnitt var kommandobaserte, og brukerne av systemet var ofte spesialister på området. Som tidligere nevnt har teknologi blitt en integrert del av dagens samfunn. Dette betyr at brukergrensesnitt på datasystemer også har vært nødt til å tilpasse seg et bredere og mindre datakyndig publikum.

2.1.3 Brukskvalitet

Brukskvaliteten i et system er i følge Jacob Nielsen (Nielsen, 1993, s.26) definert ved følgende fem attributter:

1. **Lett å lære**: Systemet skal være lett å lære slik at en ny bruker tidlig kan få gjort arbeid i systemet.
2. **Effektivt**: Systemet skal være effektivt slik at når en bruker har lært systemet vil det oppstå høy produktivitet.
3. **Lett å huske**: Systemet skal være lett å huske slik at brukere med lav brukshyppighet lett kan komme tilbake til systemet uten å måtte lære seg alt på nytt.
4. **Feilfritt og feiltolerant**: Brukerne av systemet skal ikke ha mulighet til å gjøre mange feil, og feil som er katastrofale for systemet skal ikke forekomme.
5. **Behagelig i bruk**: Systemet skal være behagelig i bruk, slik at brukerne subjektivt kan like å bruke det.

2.1.4 Brukeropplevelse

Brukeropplevelse er den totale opplevelsen som en bruker har med et system. For den enkelte bruker er opplevelsen subjektiv. Dette betyr at brukeropplevelse er vanskelig å måle, men ISO-standarden for brukeropplevelse trekker frem begrepene effektivitet og tilfredshet (International Organization for Standardization, 2018).

2.1.5 Interaksjonsdesign

Interaksjonsdesign handler om å designe interaktive produkter eller systemer. Vi har i dag fem dimensjoner innen interaksjonsdesign. Dimensjoner innen interaksjonsdesign ble først introdusert av Moggridge i boken *Designing Interactions*, hvor Crampton Smith introduserte de fire første (Moggridge, 2007). Silver la senere til en femte dimensjon, oppførsel (Silver, 2007). De fem dimensjonene innen interaksjonsdesign er:

1. **Ord**: Skal være forståelige, og på en slik måte at de kommuniserer med brukeren.
2. **Visuelle representasjoner**: Grafiske elementer som brukeren interagerer med, som bilder eller ikoner. Skal supplementere ordene i kommunikasjonen med brukeren.
3. **Fysiske objekter eller rom**: De fysiske elementene som brukeren interagerer med. Dette kan være et tastatur, en mus eller en touchskjerm.
4. **Tid**: Tiden som brukeren interagerer med de tre første dimensjonene. Dette kan også være innhold i systemet som endrer seg over tid, som lyd, video eller animasjoner.

5. **Oppførsel**: Er reaksjonene og følelsene som en bruker får når brukeren interagerer med systemet. Reaksjonene og følelsene kan være individuelle for hver enkelt bruker.

2.1.6 Universell utforming

The Center For Universal Design ved North Carolina State University lanserte i 1997 sju prinsipper for universell utforming. Ideen bak dette er at samfunnet skal utvikles slik at flest mulig kan ta del uavhengig av funksjonsevne.

1. **Like muligheter for bruk:** Utformingen skal være brukbar og tilgjengelig for personer med ulike evner.
2. **Fleksibel i bruk:** Utformingen skal tjene et vidt spekter av individuelle preferanser og evner.
3. **Enkel og intuitiv i bruk:** Utformingen skal være lett å forstå uten hensyn til erfaringen, kunnskapen, språkevnene eller konsentrasjonsnivået til brukeren.
4. **Forståelig informasjon:** Utformingen skal kommunisere nødvendig informasjon til brukeren på en effektiv måte, uavhengig av forhold knyttet til omgivelsene eller til brukeren sine sensoriske evner.
5. **Toleranse for feil:** Utformingen skal minimalisere farer og skader som kan gi ugunstige konsekvenser, eller minimalisere utilsiktede handlinger.
6. **Lav fysisk utfordring:** Utformingen skal kunne brukes effektivt og hendig med et minimum av byrde.
7. **Størrelse og plass for tilgang og bruk:** Hensiktsmessig størrelse og plass skal gjøre det mulig med tilgang, rekkevidde, betjening og bruk, uavhengig av brukeren sin kroppsstørrelse, kroppsstilling eller mobilitet.

(Digitaliseringsdirektoratet, u.å.).

I Norge stiller diskriminerings- og tilgjengelighetsloven krav om universell utforming med formålet å fremme likestilling og likeverd, sikre like muligheter og rettigheter til samfunnsdeltakelse for alle, uavhengig av funksjonsevne, og hindre diskriminering på grunn av nedsatt funksjonsevne (Diskriminerings- og tilgjengelighetsloven, 2008, §1).

2.2 Representational State Transfer

Representational State Transfer (eller REST) er en programvarearkitektur som setter regler og begrensninger ved utvikling av web-tjenester. Begrepet REST ble først introdusert i 2000 av Roy Fielding i hans doktorgradsavhandling (Fielding, 2000), og i dag omtales webtjenester som følger REST-arkitekturen for RESTful. I avhandlingen er det nevnt seks forskjellige begrensninger som definerer et RESTful system.

2.2.1 Klient-tjener arkitektur

Den første begrensningen som settes i en REST-arkitekturen er at klienten og tjeneren skal være separert (Fielding, 2000, s. 78). Klient-tjener-stilen er den hyppigst brukte arkitekturen ved utvikling av web-baserte tjenester (Fielding, 2000, s. 45). Denne består av en tjener som lytter etter forespørsler og er klar til å tilby sine tjenester, og en klient som ønsker å gjøre noe som dermed sender forespørsler til tjeneren. På denne måten får vi separert bekymringene for brukergrensesnitt fra bekymringene for lagring av data, og det blir også enklere å utvide eller skalere systemet (Fielding, 2000, s. 78).

2.2.2 Tilstandsløshet

En begrensning for kommunikasjonen mellom klient og tjener er at den må være tilstandsløs (Fielding, 2000, s. 78). Dette vil si at hver forespørsel som sendes fra klienten må inneholde nødvendig informasjon, slik at tjeneren forstår forespørselen. Tilstanden til sesjonen lagres da på klientsiden. Dette forbedrer skalerbarhet da tjeneren ikke trenger å lagre tilstanden til klienten mellom hver forespørsel, og kan frigjøre ressurser (Fielding, 2000, s.48).

2.2.3 Cache

REST-arkitekturen tar i bruk det faktum at klienter (nettlesere) kan cache informasjon, altså lagre informasjon i nettleseren. Svar fra tjeneren må da inneholde om det skal caches eller ikke. På denne måten kan vi delvis eller fullstendig eliminere enkelte klient-tjener interaksjoner og dermed forbedre skalerbarhet og ytelse (Fielding, 2000, s.80).

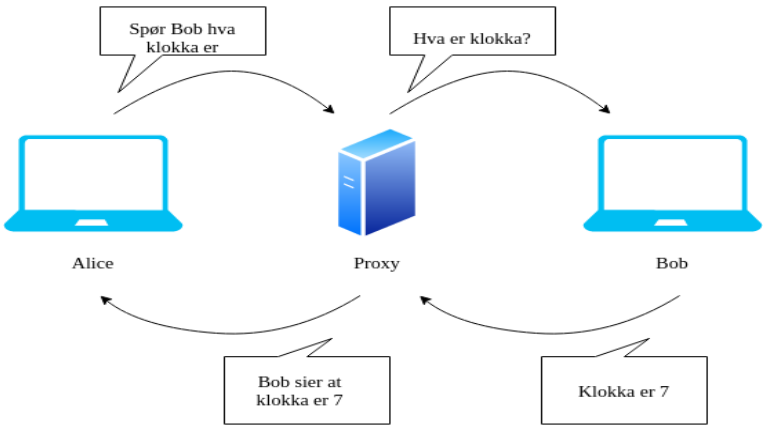
2.2.4 Uniformt grensesnitt

REST legger stor vekt på et uniformt grensesnitt. Dette inneholder fire begrensninger som forenkler arkitekturen og lar hver del utvikles uavhengig av de andre delene (Fielding, 2000, s. 82):

1. Som standard brukes URI-standarden for å identifisere en ressurs. En ressurs kan for eksempel være et web-dokument (nettside).
2. Dersom ressursene skal manipuleres skjer dette ved bruk av HTTP-standarden. Et eksempel på dette er at en GET-forespørsel betyr at klienten vil hente ut data fra tjeneren.
3. Det skal brukes MIME- og RDF-standarder for å gjøre meldinger selvbeskrivende i stedet for å bruke applikasjonsspesifikke meldingsstyper. På denne måten inneholder hver melding nok informasjon om hvordan meldingen skal tolkes, og klienten kan finne data ved å se på semantikken til meldingen.
4. Hyperlenker skal brukes for å endre tilstanden til klienten. Dette vil si at man må trykke på hyperlenker for å bytte fra en spesifikk URI til en annen.

2.2.5 Lagdelt system

Et lagdelt system er organisert hierarkisk, hvor hvert lag tilbyr tjenester til laget over seg og mottar tjenester fra laget under seg. Ved bruk av et lagdelt system kan det eksempelvis plasseres en proxy-tjener (se figur 2.2.5) mellom klienten og tjeneren uten at det må gjøres spesifikke tilpasninger i systemet. På denne måten får vi videre separert bekymringene (se kapittel 2.2.1), og man kan for eksempel lett legge til et lag med sikkerhet mellom klienten og tjeneren.



Figur 2.2.5: Beskrivelse av virkemåte for en proxy-tjener.

2.2.6 Kode ved forespørsel

En klient skal ha mulighet til å utvide sin funksjonalitet ved å laste ned og kjøre Java applets eller scripts. Kode ved forespørsel er den eneste av begrensningene i REST som er valgfri og man må altså ikke benytte seg av denne for å få en RESTful web-tjeneste (Fielding, 2000, s. 85).

2.3 Relasjonsdatabase

Merriam-Webster Dictionary definerer begrepet database som "a usually large collection of data organized especially for rapid search and retrieval (as by a computer)" (Merriam-Webster, u.å.). For å behandle informasjonen som ligger i en database brukes en type programvare som kalles Database Management System (DBMS).

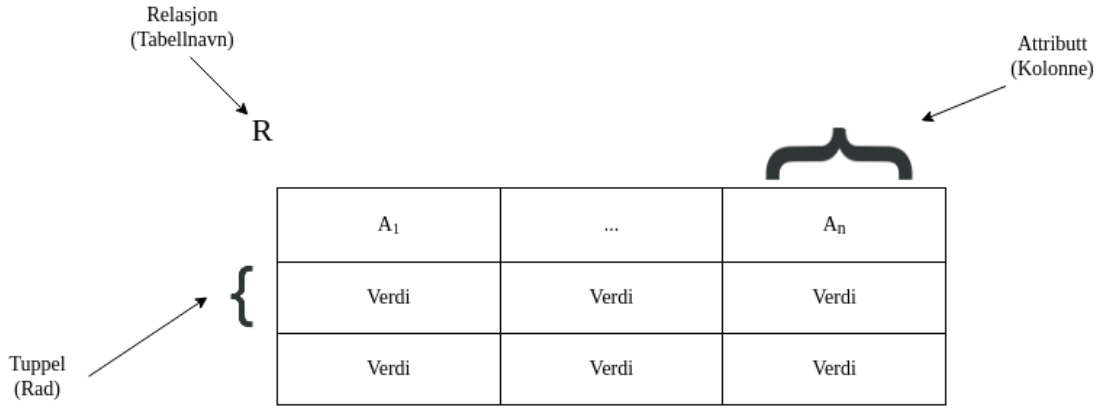
I en relasjonsdatabase lagres informasjonen i form av relasjoner (tabeller). En relasjon består av:

Attributt: En attributt er de egenskapene som definerer en relasjon.

Tuppel: Hver rad i en relasjon kalles for en tuppel. En tuppel i en relasjon skal være unik for å unngå dobbeltlagring (Codd, 1990, s. 89).

Grad: Antallet attributter i en bestemt relasjon definerer relasjonens grad. En relasjon med *n* antall attributter har grad *n.*

Kardinalitet: Antallet tupler i en bestemt relasjon definerer relasjonens kardinalitet. En relasjon med *n* antall tupler har kardinalitet *n*.



Figur 2.3.1: Relasjon lagret i en relasjonsdatabase. Denne består av attributter (kolonner) og tupler (rader).

En veldig viktig egenskap ved relasjonsmodellen er at hvert objekt som er lagret i relasjonen må ha en unik identifikator, og derfor kan skilles fra de andre objektene som er lagret i samme relasjon (Codd, 1990, s.18). Dette gjøres ved at hver relasjon har en *primærnøkkel*. Dette er en

verdi, eller en samling av verdier i en rad som entydig identifiserer denne raden (Codd, 1990, s.23). Hvor enn i databasen man trenger å referere til det gitte objektet, brukes samme verdi. Når en primærnøkkel fra en relasjon brukes i en annen for å referere til et objekt, kalles det en *fremmednøkkel*.

2.4 Utviklingsmetodikk

Utviklingsmetodikk går ut på å dele et utviklingsprosjekt inn i ulike faser. Disse fasene kan blant annet omhandle planlegging, testing, arbeid og ferdigstilling av prosjektet. Den utviklingsmetodikken som velges setter rammer for hvordan prosjektarbeidet skal gjennomføres, og eventuelt hvilke del-leveranser og dokumenter som skal inngå i prosjektet.

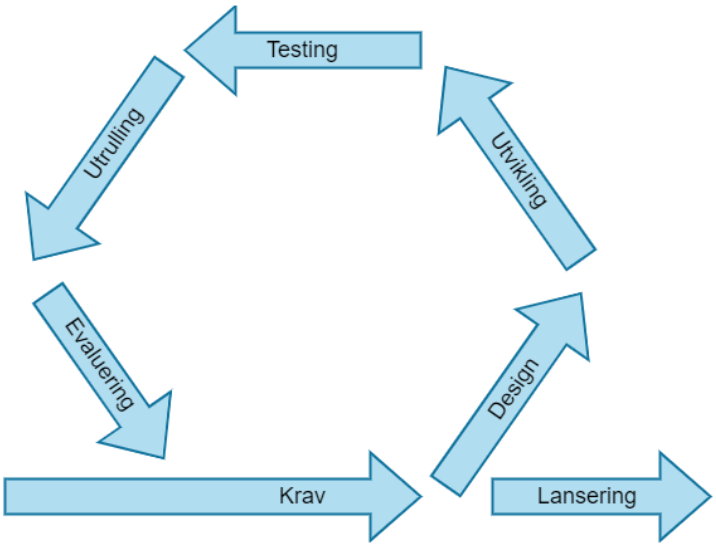
2.4.1 Smidig utvikling

Begrepet "smidige metoder" har blitt brukt i godt over et tiår, og selve konseptene og gjennomføringen assosiert med smidig utvikling har vært tilstede enda lenger (Greer & Hamon, 2011, s. 943). Selv om det ikke er fullstendig enighet om hva smidig utvikling er, så finnes det en god del kjennetegn som går igjen i smidige utviklingsprosjekter. Først og fremst dekker de behovet for å utvikle programvare raskt og effektiv. Dette gjør de ved å ta i bruk iterative metoder, samt å gi hyppige leveranser til kunden underveis i utviklingen. Det er ønskelig å ha et nært samarbeid med kunden gjennom hele prosessen, i tillegg til å kunne respondere raskt på endringer i krav underveis. Andre kjennetegn ved smidig utvikling er at det benyttes små team som organiserer seg selv, samt at ideene om testdrevet utvikling og kontinuerlig integrasjon er sentrale (Greer & Hamon, 2011, s. 943).

Begrepet "smidig" ble for alvor tatt i bruk etter at Manifestet for smidig programvareutvikling (Beck et al., 2001) ble utgitt. Manifestet beskriver verdier som er spesielt viktige i smidig utvikling:

"***Personer og samspill*** *fremfor prosesser og verktøy*
Programvare som virker *fremfor omfattende dokumentasjon*
Samarbeid med kunden *fremfor kontraktsforhandlinger*
Å reagere på endringer *fremfor å følge en plan*"
(Beck et al., 2001).

Manifestet påpeker at selv om punktene som står uthevet til venstre verdsettes høyest, så har også punktene som står til høyre en verdi.



Figur 2.4.1: Smidig utvikling med de ulike fasene. Iterativt arbeid gjør at man kommer tilbake igjen til tidligere faser.

2.4.2 Scrum

En utviklingsmetodikk som er mye brukt innenfor systemutvikling er Scrum. Ifølge Deemer, Benefield, Larman & Vodde (2012, s. 3) er Scrum et utviklingsrammeverk hvor kryssfunksjonelle team utvikler produkter eller prosjekter på en iterativ, inkrementell måte. Under utviklingen deles arbeidet opp i sykluser som kalles sprinter. En sprint kan ha en varighet på opptil en måned, og i løpet av en sprint skal det lages et delprodukt som er mulig å ta i bruk og som potensielt kan lanseres (Schwaber & Sutherland, 2017, s. 9).

Et Scrum-team består av en produkteier, et utviklerteam og en Scrum Master. Produkteieren er en enkeltperson som har ansvar for at det utvikles et produkt med så høy verdi som mulig. Denne personen har også hovedansvaret for produkt-backlogen. Dette er en prioritert liste over funksjonaliteten som skal være med i produktet som utvikles. Utviklerteamet har ansvar for å lage selve produktet ut ifra kravene i produkt-backlogen. Dette teamet er selvorganisert og kryssfunksjonelt, og bør ha en størrelse på 3-9 medlemmer (Schwaber & Sutherland, 2017, s. 7). Scrum Master sørger for at prosjektarbeidet følger rammene for Scrum, og hjelper hele teamet med å forstå prinsippene og teorien som ligger til grunn.

I Scrum brukes det planlagte hendelser for å skape regelmessighet og for å minimere behovet for andre møter (Schwaber & Sutherland, 2017, s. 9). Utenom sprinter har Scrum følgende hendelser:

- **Sprintplanlegging:** Gjennomføres før starten av en sprint. Her planlegges arbeidet som skal gjøres i løpet av sprinten, og det settes opp et sprintmål og en sprint-backlog.

- **Stand-up møte (daily scrum):** Et daglig sprintmøte hvor det planlegges hva som skal gjøres i løpet av de neste 24 timene.
- **Sprint review:** Gjennomføres på slutten av en sprint for å inspisere arbeidet som har blitt gjort, og eventuelt endre produkt-backlogen ved behov.
- **Retrospektiv:** Gjennomføres mellom Sprint review og neste sprintplanlegging. Her går utviklerteamet gjennom hvordan arbeidsprosessen har fungert, og setter opp en plan med forbedringer som skal tas med til neste sprint.

2.5 Brukertester

Brukertester er en systematisk evaluering av et produkt der man observerer brukere for å samle informasjon om hvordan systemet/produktet fungerer i bruk. Det finnes ulike måter å gjennomføre brukertester på, der det ønskede resultatet kan variere, men et sammenfallende hovedmål for slike tester er å øke brukervennligheten (Dumas & Redish, 1999, s. 22). På denne måten kan brukertester være en viktig kilde for å sikre at produktet blir brukervennlig og funksjonelt korrekt for sluttbrukerne. Videre kan også brukertester bli brukt for å øke lønnsomheten for produktet. Dette begrunnes blant annet ved at gode brukertester kan føre til lavere kostnader knyttet til service- og support, økt salg som følge av fornøyde kunder, markedsfordel knyttet til god brukervennlighet, samt at man ved hjelp av brukertestene vil utarbeide kriterier for brukervennlighet som kan brukes i nye og eksisterende produkter (Rubin & Chisnell, 2008, s.23).

2.5.1 Planlegging

En viktig fase ved brukertesting er forarbeidet, altså planleggingen. Hva ønsker man å teste, hvordan skal det testes, når skal det gjennomføres og hvorfor ønsker man å teste, er eksempel på typiske spørsmål som bør besvares i planleggingsfasen. Dette kan gjøres ved hjelp av en testplan. Rubin & Chisnell (2008, s. 66) kommer med flere argumenter for viktigheten av en testplan:

- Den fungerer som mal/oppskrift for selve brukertesten.
- Gir mulighet for økt samarbeid mellom testleder og utviklingsteam.
- Synliggjør nødvendige ressurser for gjennomføring av testen.
- Kan bidra til å fastsette milepæler for utviklingen.

Testplanen legger altså grunnlaget for hvordan testingen blir gjennomført og hva man ønsker å oppnå. Ved å være grundig under utarbeidelsen av testplanen vil man kunne forenkle

gjennomføringen av brukertestene, samt at de ulike aktørene involvert forstår hvilke ressurser som vil være nødvendige (Rubin & Chisnell, 2008, s. 66). Utformingen/disposisjonen for testplanen kan variere for ulike brukertester og vil blant annet være avhengig av ønsket grad av formalitet.

2.5.2 Behandle resultater

Etter brukertestene har blitt gjennomført vil det være nødvendig å gå igjennom informasjonen som er samlet inn. Det vil foregå ved å analysere og behandle de resultatene man har, med utgangspunkt i å finne eventuelle problemer, forbedringer eller endringer for produktet (Rubin & Chisnell, 2008, s. 246). Omfanget av et slikt etterarbeid kan være svært varierende, og det vil dermed være naturlig at det vurderes i forhold til tid og ressurser tilgjengelig. Vanligvis deles etterbehandlingen inn i to deler: innledende analyse og omfattende analyse med rapport (Rubin & Chisnell, 2008, s. 245). Den innledende analysen vil være en raskere og enklere prosess, der hovedmålet vil være å utpeke umiddelbare problemområder, slik at man kan begynne med implementering av løsninger så tidlig som mulig. Dette vil gjerne være kritiske eller tydelige problemområder som kan identifiseres etter en relativt rask analyse. Ved å gjøre dette vil utviklingsteamet kunne utbedre funksjonalitet samtidig som det gjennomføres den omfattende analysen med tilhørende rapport. Denne har et tidsperspektiv på 2-4 uker og vil være grundigere utarbeidet (Rubin & Chisnell, 2008, s. 270). I rapporten skal alle funn fra den innledende analysen inkluderes, og eventuelt oppdateres hvis ny informasjon er avdekket. I tillegg skal den inkludere utvidet behandling og analyse, samt nye funn. I prosessen ved å behandle resultatene er det fire hovedpunkter som blir trekt frem: kompilere og summere data, analysere data, utvikle anbefalinger og produsere rapport (Rubin & Chisnell, 2008, s. 246).

Kompilere og summere data omhandler å sammenfatte data fra brukertestene, som kan ligge i ulike observatørskjema eller lignende, slik at man kan finne trender. Hver brukertest har gjerne et individuelt skjema der testsvar, oppgaveutførelse og generelle notater blir ført for den aktuelle testen. For å forenkle sammenlikning og analyse bør dette tilpasses og plasseres i et felles skjema. Dette bør helst gjøres fortløpende gjennom testperioden slik at man tidlig kan finne mangler i datamaterialet (Rubin & Chisnell, 2008, s.248). Ved **analysering av data** gjelder det å forstå dataen som er innsamlet. Hvilke resultater har man fått og hva betyr disse? Her vil det være vanlig å identifisere oppgaver med lav oppfyllelse av suksesskriterier, brukerfeil eller vanskeligheter samt feilkilder (Rubin & Chisnell, 2008, s. 260). **Utvikling av anbefalinger** blir gjennomført i lys av analysen for å forbedre eller feilrette de ulike

problemene som ble funnet. Disse bør inneholde forslag til hva som kan forbedres og hvorfor. Ved tilstrekkelig forkunnskap vil det også være naturlig å inkludere prioritering eller alvorlighetsgrad på problemene som er funnet (Rubin & Chisnell, 2008. s. 276). Til slutt ønsker man å **produsere en rapport** som oppsummerer brukertestene. Denne vil representere sluttproduktet for testene, som inneholder all data, analyse, resultater, funn og anbefalinger.

2.6 Sikkerhet

I et system med autentisering og kommunikasjon over nett er det viktig at sikkerhet implementeres på en god måte. Data og sensitiv informasjon som sendes mellom tjener og klient skal ikke kunne leses eller manipuleres av andre under transport, og passord må lagres på en sikker måte som tar høyde for eventuelle angrep mot systemet. Dette kapittelet beskriver noen grunnleggende sikkerhetstiltak som bør implementeres i en webapplikasjon.

2.6.1 Hashing av passord

Ved utvikling av systemer med innlogging og autentisering er det viktig å lagre passord på en sikker måte. Passord som lagres i klartekst vil utgjøre en sikkerhetsrisiko ved et eventuelt angrep mot systemet. For å sikre passordene er det vanlig å bruke en hash-funksjon. Dette er en funksjon som tar en tekststreng med vilkårlig lengde og produserer en ny streng med fast lengde, ofte referert til som en hash (Menezes, Katz, Van Oorschot & Vanstone, 1996, s. 321). På denne måten vil ikke det opprinnelige passordet bli kjent ved et eventuelt angrep.

```
hash("passord1") = XPRXbTqQ5Tvr8SBqdg49
hash("passord2") = edZikI2cRFBMmU7VT10n
hash("passord3") = lXexYjMIJbcer14gHepy
```

Figur 2.6.1.1: Pseudokode for en hash-funksjon.

Det er likevel viktig å påpeke at hashing alene ikke er nok til å hindre lekking av passord ved et eventuelt angrep. Det finnes en rekke angrepsmetoder som kan knekke hash-verdier svært raskt. Ifølge Defuse Security (2019) er “dictionary”- og “brute force”-angrep de to vanligste måtene å gjette seg frem til passord på. Et “dictionary”-angrep tar utgangspunkt i en liste med velkjente ord og fraser, mens et “brute force”-angrep prøver alle mulige kombinasjoner av bokstaver opp til en viss lengde. Videre forteller Defuse Security (2019) at “lookup tables” og “rainbow tables” er ekstremt effektive metoder for å knekke mange hasher av samme type. Her beregnes hash-verdien til en rekke passord på forhånd og settes inn i en oppslagstabell sammen med korresponderende passord.

For å unngå angrep ved hjelp av "lookup tables" og "rainbow tables" er det vanlig å bruke et salt. Dette er en tilfeldig generert streng som legges til passordet under hashing. Saltet gjør at hash-verdiene blir unike, selv om passordene som behandles av hash-funksjonen er like.

```
hash("passord1" + salt1) = TcVGPujWTWa80Bfb2aCM
hash("passord1" + salt2) = rnhI8QMPoRjX5fmBsZgw
hash("passord1" + salt3) = UmRY9L1bqOtLRPfSmFR8
```

Figur 2.6.1.2: Pseudokode for en hash-funksjon med salt.

Bruk av salt hindrer ikke angripere fra å kjøre "dictionary"- eller "brute force"-angrep på individuelle hash-verdier. Man kan likevel gjøre disse angrepsmetodene mindre effektive ved å bruke en nøkkelstrekkings-algoritme. Dette er en algoritme som implementeres ved å bruke en CPU-intensiv hash-funksjon, noe som gjør hashing-prosessen såpass treg at "dictionary"- og "brute force"-angrep fungerer dårlig (Defuse Security, 2019). Derfor bør det også brukes en godt implementert nøkkelstrekkings-algoritme under hashing av passord, i tillegg til salt.

2.6.2 Kryptert kommunikasjon og sertifikater

For systemer som kommuniserer over internett er det viktig å tenke over sikkerhetsrisikoen denne kommunikasjonen medfører. Krutz & Nahari (2011, s. 214) definerer de tre grunnleggende prinsippene for informasjonssikkerhet på følgende måte:

- **Konfidensialitet:** Prøve å unngå autorisert avsløring av en meldings innhold, enten dette bevisst eller ubevisst.
- **Integritet:** Sikre at uautoriserte endringer av data ikke forekommer, og at dataen er konsistent både internt og eksternt.
- **Tilgjengelighet:** Sikre pålitelig tilgang til data eller ressurser for passende personell til riktig tid.

For å sikre konfidensialitet ved kommunikasjon over internett er det vanlig å kryptere meldinger som blir sendt. Dette gjør at uvedkommende som lytter til kommunikasjonen mellom to parter ikke har mulighet til å lese selve innholdet i meldingene som sendes. Det finnes to typer kryptering som kan brukes til dette: symmetrisk og asymmetrisk kryptering. Ved symmetrisk kryptering har begge parter tilgang til krypteringsnøkkelen, og denne må holdes hemmelig for resten av verden. Ved asymmetrisk kryptering brukes det to forskjellige nøkler, en offentlig og en privat, hvor kun den private nøkkelen må holdes hemmelig. Ifølge Thomas (2000, s. 27) har hver av disse metodene sine fordeler og ulemper, og den optimale tilnærmingen vil være å bruke en kombinasjon av symmetrisk og asymmetrisk kryptering.

For å sikre integritet ved kommunikasjon over internett er det vanlig å bruke digitale sertifikater. Ifølge Krutz & Nahari (2011, s. 143) er hovedformålet med digitale sertifikater å bekrefte at en persons offentlige nøkkel faktisk tilhører han eller henne. Det digitale sertifikatet forteller altså mottakeren at utstederen er til å stole på, og at den offentlige nøkkelen trygt kan brukes til videre kommunikasjon. Krutz & Nahari (2011, s. 143) forteller videre at et gyldig sertifikat oppnås ved at en tredjeparts sertifikatsutsteder verifiserer at den offentlige nøkkelen tilhører rett person, og deretter digitalt signerer den offentlige nøkkelen og annen tilhørende informasjon.

Tilgjengelighet avhenger av tjenerens oppetid og tilgang til tjenerens ressurser, og er ikke direkte knyttet til kryptert kommunikasjon og sertifikater. Dette vil derfor ikke omhandles i dette kapittelet.

3 Valg av teori og metode

I dette prosjektet utvikles det en webapplikasjon, derfor er det brukt mye teknologi og rammeverk relatert til dette. Oppgavestiller hadde ingen preferanser til hvilke teknologier eller metoder som skulle bli brukt i prosjektet (se vedlegg E, kapittel 2.1), derfor er mange av valgene som er gjort basert på at teamet har hatt gode erfaringer med de forskjellige teknologiene og metodene tidligere. Samtidig har vi sett på hva som faktisk passer bra med prosjektet og fulgt med på hva som har fungert eller ikke fungert. Videre i dette kapittelet skal vi begrunne valgene vi har gjort, og se på positive og negative sider for de forskjellige teknologiene og metodene.

3.1 Valg av programmeringsspråk

Dette kapittelet omhandler programmeringsspråkene som er brukt i prosjektet, og hvorfor disse er valgt.

3.1.1 JavaScript (JS)

JavaScript er et høynivå programmeringsspråk som tilbyr både objektorientering og funksjonell programmering. Det tilbyr også kode som er eksekverbar i en nettleser og blir ofte også betraktet som et scriptspråk (Flanagan, 2006, s.2). Vi valgte å bruke JavaScript som programmeringsspråk både på klient- og tjenersiden da det er et språk som tilbyr mange biblioteker og rammeverk som hjelper oss til å oppnå våre mål for systemet. Bibliotekene og rammeverkene som er brukt i prosjektet blir diskutert i kapittel 3.2.

3.1.2 Cascading Style Sheets (CSS)

Cascading Style Sheets eller CSS er et språk som tillater brukeren å endre utseendet til nettsiden. I de første årene etter verdensveven ble lansert i 1990 hadde utviklere liten til ingen kontroll over hvordan en nettside ble seende ut (Lie & Bos, 2005, s.1). Valget om å bruke CSS falt naturlig, da det i all hovedsak er den eneste måten man kan endre utseendet til en nettside. Selv om det i prosjektet brukes Bootstrap (se kapittel 3.2.2.2) som som har egne CSS filer tilhørende sine komponenter, ble CSS brukt for å endre på disse slik at de stilistisk passet inn i vårt system.

3.2 Valg av teknologi

I hvert av disse underkapitlene vil det bli gjennomgått ulike teknologier som er tatt i bruk, hva hver teknologi brukes til og hvorfor de var hensiktsmessige valg for vårt system. Det vil også bli omtalt eventuelle kompromiss som måtte tas for å få de til å fungere sikkert og optimalt.

3.2.1 Kjøretidsmiljø

Et kjøretidsmiljø er et lag av abstraksjon mellom koden som kjøres og det underliggende operativsystemet, og brukes i programmeringsspråk for å tilby ytterligere tjenester utover standardbiblioteket (Appel, 1989, s. 1). I dette kapittelet beskrives hvilke kjøretidsmiljø som er brukt i prosjektet og hvorfor disse er valgt.

3.2.1.1 Node

Node.js, eller bare Node, er et kjøretidsmiljø lansert i 2009 som er basert på Googles V8 JavaScript motor (Dayley, 2014, s. 2). Dette lar en utvikler skrive kildekode for web-tjeneren (eller hele tjeneren) i JavaScript, og lar V8-motoren kompilere JavaScript-koden til maskinkode (Dayley, 2014, s. 2). Node har også en egen pakkebehandler, Node Package Manager (NPM), som tillater oppdatering og installasjon av tilleggsbiblioteker.

Vi har valgt å bruke Node som kjøretidsmiljø i prosjektet for å sikre et stabilt system som gir en god brukeropplevelse. En annen vesentlig faktor for at Node ble valgt i prosjektet er at utviklerteamet har erfaring med dette fra før, og at det tillater bruk av JavaScript ende-til-ende i prosjektet. Man har da også fordelen av at utviklere som jobber på forskjellige deler av prosjektet snakker samme språk.

3.2.1.2 Babel

Babel er en JavaScript-kompilator som først og fremst er brukt for å konvertere ECMAScript 2015+ kode til en bakoverkompatibel versjon av Javascript (Zhu et al., 2019). I prosjektet har vi brukt Babel for å tolke kildekode skrevet i React og å fjerne typesjekkingen som gjøres med flow.

```
// @flow
function square(n: number): number {
       return n * n;
}
```

Figur 3.2.1.2.1: En funksjon tar inn et tall som parameter og kvadrerer det. I koden brukes det typesjekking med flow.

```
function square(n){
	return n * n;
}
```

Figur 3.2.1.1.2: Samme funksjon som i figur 3.2.1.2.1, kompilert med Babel.

Figur 3.2.1.2.1 viser en funksjon *square* som tar inn et tall som parameter og kvadrerer dette. I funksjonen brukes flow for å sjekke at det faktisk er et tall som tas inn som parameter og at det som returneres fra funksjonen også skal være et tall. Denne typesjekkingen støttes ikke av JavaScript-kompilatoren i nettleseren eller av Node (se kapittel 3.2.1.1), og man bruker derfor Babel for å kompilere til forståelig JavaScript (se figur 3.2.1.2.2).

3.2.2 Brukergrensesnitt

Et av forskningsspørsmålene omhandler hvilken teknologi som bør tas i bruk for å lage et system som er mer brukervennlig enn dagens løsning (se kapittel 1.2.2). Det var derfor viktig å velge teknologi som gav oss muligheten til å oppnå alle aspektene for et brukervennlig system. Teknologiene og begrunnelsen for bruken av disse vil bli beskrevet i dette kapittelet.

3.2.2.1 React

React ble lansert i 2013 av Facebook og er et JavaScript-bibliotek for konstruksjon av brukergrensesnitt (Fedosejev, 2015, s. 2). React bruker en virtuell DOM som er lagret i minnet i stedet for å interagere med den som blir generert av nettleseren, noe som har positiv innvirkning på ytelsen (Aggarwal, 2018, s. 133). Dette er en av hovedårsakene til at vi valgte å bruke React i prosjektet.

Samtlige medlemmer i utviklerteamet hadde også erfaring med React fra tidligere prosjekter. Dette var en faktor som spilte inn i valget av bibliotek, da et annet bibliotek potensielt hadde vært tidkrevende å sette seg inn i. En siste grunn til at valget falt på React var dets bruk av enveis dataflyt. Dette betyr at hver komponent sender data nedover til komponenten under seg. Dersom dataen da blir endret i toppkomponenten, endres den også i alle underkomponenter som bruker denne dataen. Som et resultat av dette viser React seg for å være en av de beste bibliotekene for utvikling av interaktive web-applikasjoner (Aggarwal, 2018, s. 134).

3.2.2.2 React-Bootstrap

I prosjektet er det brukt React-Bootstrap. Bootstrap er et CSS-rammeverk som inneholder maler for diverse komponenter man kan finne på en nettside, og i React-Bootstrap er disse

komponentene bygd med React. Årsaken til at vi valgte å bruke React-Bootstrap i prosjektet er at det forenkler tilpasningen av utseendet til nettsiden, samt at det gir mer oversiktlig kode enn kun bruk av Bootstrap.

3.2.2.3 Grafbibliotek

Da vi skulle bestemme oss for grafbibliotek hadde vi flere forskjellige biblioteker å velge mellom. Vi vurderte disse bibliotekene:

- **Highcharts:** Brukes av store bedrifter som Facebook, Twitter og Yahoo, men koster penger dersom det skal brukes kommersielt. Da prosjektet ikke har avsatt ressurser, ble dette rask avfeid.
- **Plotly:** Gratis og med åpen kildekode. Dette biblioteket testet vi i starten av prosjektet, men vi opplevde minnelekkasjer i Windows 10.
- **Recharts:** Et lettvekts grafbibliotek som er utviklet i React (Recharts Group, 2020) som er gratis og med åpen kildekode.

Vi endte opp med å velge Recharts som grafbibliotek hovedsakelig grunnet dets simplisitet i bruk. De forskjellige grafene brukes i kildekoden som React-komponenter, og dataen kan sendes inn som en egenskap i komponenten.

3.2.3 HTTP-klient

Axios er en lovnadsbasert HTTP-klient som tilbyr muligheten til å sende HTTP-forespørsler til en Node-tjener. Axios brukes i såkalte service-klasser som håndterer kommunikasjonen mellom klient og tjener, noe vi har valgt å bruke for å bedre oversikten i prosjektet. Hver REST-ressurs (se vedlegg D, kapittel 6) har sin egen klasse som lar klienten hente ut, endre eller sende inn den gitte ressursen. Vi valgte å bruke Axios i prosjektet da det er lett å implementere med Node sin pakkebehandler, samt at det er et godt dokumentert bibliotek.

3.2.4 API

Et *Application Programming Interface* (API) skal gjøre all behandling av data mellom klientsiden og databasen. For å gjøre det enklest mulig så har vi valgt å bruke et rammeverk for å bygge APIet vårt. Rammeverket vi bruker heter Express og er et open source rammeverk for webapplikasjoner i Node (Node.js Foundation, 2017). Vi har valgt å implementere autentisering ved å skrive mellomvare for verifisering av access tokens med Express (se vedlegg D, kapittel 7).

3.2.5 ORM

En *Object Relational Mapper* (ORM) brukes av APIet for å kommunisere med databasen. Denne programmeringsteknikken gjør at utviklere unngår å skrive egne SQL spørringer ved at de istedenfor bruker funksjoner fra ORMet som genererer SQL spørringer (Grünwaldt, J. M., 2019, s.3). Dette kan for eksempel gjøre at det unngås flere utviklerfeil i koden og samtidig så kan dette gi en viss beskyttelse mot SQL injection (se vedlegg D, kapittel 7.4). Vi har valgt å bruke en ORM som heter Sequelize, ettersom vi har hatt tidligere erfaring med den.

3.2.6 Testing

Vi bruker enhetstesting for å teste applikasjonen og vi har valgt å bruke Jest som testrammeverk for å sette opp alle testene. Jest støtter de fleste typer prosjekter og har også en funksjon for å sjekke testdekningen (Facebook, 2020). Vi hadde også behov for et par andre pakker for å kunne teste endepunkter og serviceklasser. SuperTest ble brukt for å teste endepunkter på tjenersiden, og *axios-mock-adapter* ble brukt for å teste service-klassene på klientsiden.

Vi bruker CI for å kjøre testene hver gang vi legger ut ny kode i versjonskontrollsystemet. *Continuous integration* (CI) er en programmeringsteknikk som bygger og tester applikasjonen hver gang det legges ut en ny versjon på versjonskontrollsystemet (GitHub, 2020). Vi bruker et verktøy tilbudt av GitHub for å kjøre CI på applikasjonen vår.

3.2.7 Database

Vi har i prosjektet valgt å bruke en relasjonsdatabase da denne tillater fremtidig skalerbarhet. Dette kommer av at relasjoner er lagret i forskjellige tabeller, og lenkes til hverandre via en fremmednøkkel (se kapittel 2.3). Det gjør det da enkelt å legge til flere relasjoner dersom systemet skal utvides. Vi har valgt MySQL som DBMS, da MySQL fungerer på stort sett alle operativsystem. Det er også per juli 2019 det mest brukt DBMS-et i verden som er gratis og med åpen kildekode (Chand, 2019).

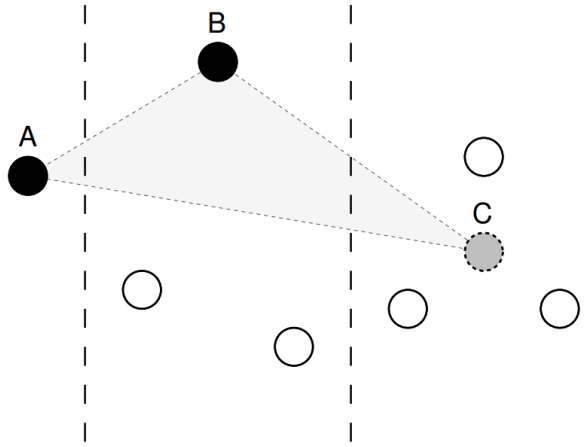
3.3 Nedsamplingsalgoritmer

Vi fant ut tidlig i prosjektet at det ikke gikk å visualisere over 15000 datapunkter på grafen uten problemer. Grafen og nettsiden gikk tregt og det var ikke brukervennlig å ha forsinkelser når man skulle prøve å lese verdier av grafen. Derfor fant vi noen algoritmer for å redusere antall punkter på grafen uten at det skulle påvirke den visualiserte dataen for mye.

3.3.1 Largest Triangle Three Buckets (LTTB)

Largest Triangle Three Buckets er utviklet av Svein Steinarsson i mastergradsavhandlingen hans som heter *Downsampling time series for visual representation* (Steinarsson, S., 2013).

Kort sagt så deler denne algoritmen dataen opp i et visst antall deler der et punkt fra hver del blir valgt for å representere den delen (Steinarsson, S., 2013, s.21). Dette betyr at dersom 40 000 datapunkter skal reduseres ned til 5000 datapunkter vil det bli 8 punkter per bøtte, der et av disse blir valgt til å representere alle 8. Figur 3.3.1 er tatt fra mastergradsavhandlingen og viser hvordan et punkt blir valgt:



Figur 3.3.1: Visualisering av valg av punkt for LTTB. Her er punkt A valgt i forrige bøtte, C et midlertidig gjennomsnittlig punkt i neste bøtte og B det punktet som gir størst trekant mellom de tre. Derfor velges B i nåværende bøtte. Figuren og forklaringen er hentet fra Steinarssons mastergradsavhandling (Steinarsson, S., 2013, s.21-22).

3.3.2 MinMax

Dette er en algoritme inspirert av løsningen som National Instruments bruker i LabVIEW. Mer spesifikt så er den inspirert av deres forklaring som heter *Decimation Algorithm Used to Display Data on a Graph in LabVIEW* (National Instruments, 2018).

Vi har skrevet kode som bare er basert på denne forklaringen, men det er ikke sikkert at vi bruker akkurat samme algoritme. “MinMax”-algoritmen som vi bruker deler opp dataen i flere deler og velger 4 punkter fra hver del. De fire punktene som blir valgt fra en del er startpunktet, maksimumspunktet, minimumspunktet og sluttpunktet. “MinMax”-algoritmen burde brukes når kort kjøretid er veldig viktig, og nøyaktighet ikke er så viktig (National Instruments, 2018).

3.4 Utviklingsmetodikk

Vi baserte valg av utviklingsmetodikk på følgende faktorer:

- Jevnlige møter med oppgavestiller.
- Et relativt lite utviklerteam.
- Potensiale for endring av krav underveis i prosjektet.

Med grunnlag i disse faktorene falt valget av utviklingsmetodikk på Scrum. Dette er både fordi vi er kjent med Scrum som arbeidsmetode, men også fordi det passer til disse faktorene. I starten av prosjektet ble det bestemt at det skulle jobbes i sprinter på 2 uker, hvor hver sprint skulle avsluttes med et møte med oppgavestiller. På grunn av aktivitet i annet fag hadde flere av sprintene 3 ukers varighet. I denne perioden hadde vi god dialog med oppgavestiller, slik at alle parter hadde oversikt over kommende møter.

3.4.1 Møter med oppgavestiller

Prosjektperioden ble innledet med et møte med Sveinung Johan Ohrem og Birger Venås i SINTEF Ocean, hvor oppgaven ble gjennomgått og potensielle løsninger ble drøftet. Det ble videre lagt en plan på at møter skulle forekomme ved slutten av hver sprint i SINTEF sine lokaler. Etter utbruddet av COVID 19-viruset ble alle møter flyttet til plattformen Microsoft Teams. På møtene har vi i utviklerteamet hatt mulighet til å gå gjennom hva som har blitt gjort i arbeidsperioden, og oppgavestillerne har hatt mulighet til å komme med innspill og konstruktiv kritikk. Det ble også organisert et møte under prosjektperioden med en av SINTEFs kunder, hvor vi fikk muligheten til å vise frem produktet slik det var på daværende tidspunkt og fortelle om hvilke visjoner vi hadde. På dette møtet hadde også kunden mulighet til å komme med sine synspunkter på hva de som brukere av et slikt system ville ha av funksjonalitet.

3.4.2 Møter med veileder

Veileder har deltatt på to møter med oppgavestiller, hvorav ett var oppstartsmøte. Utenom dette har vi valgt å ikke ha fastsatte møter med veileder, men å ha møter for tilbakemelding på innleverte dokumenter som visjonsdokument, kravdokumentasjon og systemdokumentasjon.

3.4.3 Møter internt i utviklerteamet

Scrum som utviklingsmetodikk tilbyr flere aktiviteter for å skape regelmessighet og minimere behovet for andre møter (se kapittel 2.4.2). Gjennom prosjektperioden har vi valgt å benytte oss av disse aktivitetene og har dermed hatt både daglige møter, samt møter både ved start og slutt av en arbeidsperiode.

3.5 Arbeids- og rollefordeling

Arbeidsfordelingen kan sees i Gannt-diagrammet i prosjekthåndboka (se vedlegg E, kapittel 1). Rollefordelingen innad i gruppen har vært på følgende måte gjennom prosjektperioden:

- Herman har vært Scrum Master i teamet. Det vil si at han har sørget for at prosjektarbeidet har fulgt rammene for Scrum, og har hjulpet hele teamet med å forstå prinsippene og teorien som ligger til grunn (se kapittel 2.4.2).
- Sander har vært referent og har hatt ansvaret for å skrive møteinnkallinger og referater til møter.
- Trond har hatt rollen som kontaktperson. Dette innebærer utsending av møteinnkallinger og andre henvendelser til oppgavestiller og veileder.
- Jørgen har hatt rollen som teamleder, og har hatt overordnet ansvar for fremdriften til prosjektet.

4 Resultater

Kapittelet beskriver alle resultater vi har kommet frem til i løpet av prosjektperioden. Disse er delt inn i empiriske, ingeniørfaglige og administrative resultater.

4.1 Empiriske resultater

De empiriske resultatene legger grunnlaget for å besvare forskningsspørsmålene satt opp i kapittel 1.2.2. Store deler av behandlingen av data fra Sensorfisk skjer ved opplastning av data. Det er relativt store datamengder som skal behandles, noe som betyr at behandlingen kan være ressurskrevende, både med tanke på tid og datakraft. Vi har derfor valgt å gjøre tidkrevende kalkulasjoner ved opplastning av data, fremfor å gjøre dette hver gang data skal hentes ut og tas i bruk. I dette kapittelet blir det gått gjennom resultatene fra de forskjellige databehandlingene, designet av webapplikasjonen og sikkerhetstiltakene som har blitt implementert i systemet.

4.1.1 Innlesing av rådatafiler

I SINTEF Ocean AS sin tidligere løsning var et av problemene at Python-scriptene de brukte ikke kunne lese inn rådatafilene fra Sensorfisk direkte. De måtte bruke programmet MissionPlanner for å konvertere rådatafilene til Matlab-filer, for deretter å kjøre Python-scriptene på de konverterte filene. Dette førte til at behandlingsprosessen ble mer komplisert og tidkrevende. Derfor var det ønskelig at den nye løsningen skulle kunne lese inn rådatafilene direkte, uten å gå via programmet MissionPlanner.

For å få til direkte innlesing av rådata trengte vi mer kunnskap om hvordan selve rådatafilene var bygget opp. Vi tok derfor utgangspunkt i kildekoden til MissionPlanner som lå åpent ute på nett (ArduPilot, 2020), i tillegg til å utforske innholdet i filene selv. Vi fant ut at rådatafilene er bygget opp av ulike pakker. Disse pakkene består av et pakkehode, i tillegg til selve dataen som ligger lagret. Pakkehodet består av totalt 3 bytes, hvor to av disse brukes til å identifisere starten av pakken, mens den siste forteller hva slags type pakke det er.

Tabell 4.1.1.1: Oppbygningen til en pakke i rådatafilen.

	Pakkehode			Data
	Byte 1	Byte 2	Byte 3	
Beskrivelse	Unik kombinasjon av bytes som viser at dette er starten på en pakke.		Forteller hva slags type pakke dette er.	Selve dataen som er lagret i pakken.
Verdi	163	149	0-255	-

I starten av rådatafilene finnes det en rekke pakker av typen "FMT". Disse er spesielle pakker som inneholder informasjon om de andre pakketypene. En FMT-pakke inneholder opplysninger om pakketype, lengde, navn, lagringsformat og etiketter. MissionPlanner begynner med å lese inn alle FMT-pakkene og lagrer informasjonen de inneholder i en tabell. Når resterende pakker skal leses inn, slår MissionPlanner opp i tabellen for å finne informasjon om den aktuelle pakketypen. På denne måten kan MissionPlanner finne ut akkurat hvor lang pakken som skal leses er, og hvilket format dataen i pakken er lagret på.

Tabell 4.1.1.2: Innholdet i datadelen av en FMT-pakke i rådatafilen.

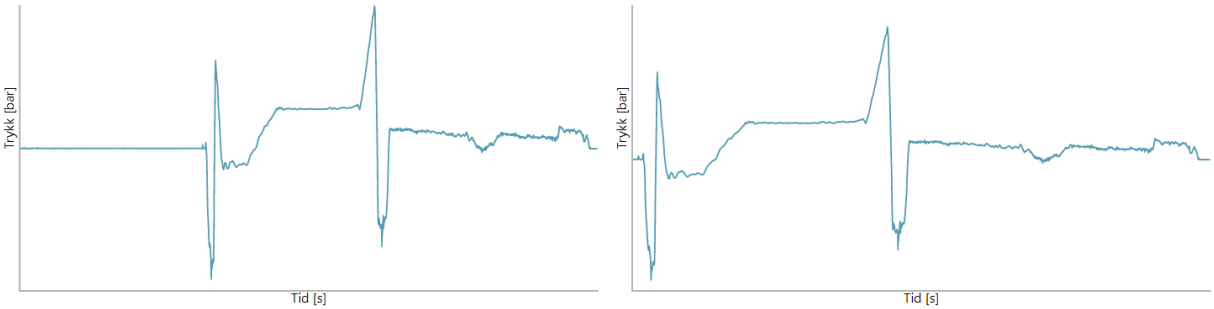
	Beskrivelse	Lengde
Type	Forteller hvilken pakketype FMT-pakken inneholder informasjon om.	1 byte
Lengde	Forteller hvor mange bytes pakker av aktuell pakketype består av.	1 byte
Navn	En tekststreng på maks 4 tegn som gir et unikt navn til pakketypen.	4 bytes
Format	En tekststreng som viser hvilket format som er brukt til å lagre dataen i pakkene. Hver bokstav i tekststrengen angir hvilken datatype en verdi er lagret som.	16 bytes
Etiketter	Tekststreng med etiketter som navngir de ulike verdiene i datadelen av pakkene. Etikettene er atskilt med komma.	64 bytes

Kildekoden til MissionPlanner viser at funksjonaliteten for å lese en rådatafil er veldig generell. Den er laget for å kunne lese alle typer rådatafiler som er lagret på riktig format, uavhengig av hvilke pakker som er med og innholdet i disse pakkene. I tillegg leser programmet ut alle pakkene som ligger lagret i rådatafilen. Vår funksjonalitet behøver derimot ikke å være like generell. Alle rådatafiler fra Sensorfisk er bygget opp likt, og har alltid de samme pakketypene. I tillegg er det kun 4 av pakketypene som SINTEF Ocean AS anser som relevante å bruke (se vedlegg E, kapittel 2.2). På bakgrunn av dette laget vi en mer spesialisert funksjon for å lese inn rådatafiler fra Sensorfisk.

Vår funksjon leser ikke inn FMT-pakkene til å begynne med. Siden rådatafilene til Sensorfisk alltid er bygget opp på samme måte, kan relevant informasjon fra FMT-pakkene implementeres direkte i koden isteden. I tillegg leses kun informasjon fra de 4 aktuelle pakketypene ut, mens resterende pakker hoppes over. Verdiene i pakkene leses ut og konverteres til riktig datatype, før de returneres som tabeller med JSON-objekter. Disse tabellene sendes videre til andre funksjoner som gjør beregninger på dataen, før verdiene lagres i databasen.

4.1.2 Beregning av starttid for gjennomkjøring

Under en gjennomkjøring starter sensorene å måle umiddelbart Sensorfisken blir skrudd på. Det blir derfor ofte litt dødtid mellom Sensorfisk skrus på til den faktisk entrer avluseren. For å fjerne denne dødtiden er det utviklet en funksjon som regner ut starttiden for gjennomkjøringen. Denne funksjonen er basert på kode fra Python-scriptene til SINTEF Ocean AS, og bruker trykkdataen til å regne ut starttiden. Trykkdataen blir delt opp i vinduer på 3 sekunder der den finner den deriverte, altså endringen i trykk. Etter opplysninger gitt fra SINTEF-ansatte i tillegg til utforskning av datamateriale, ser man at avluserne begynner med et trykkfall. Vi kan derfor benytte den deriverte til å oppdage dette trykkfallet for å finne det reelle starttidspunktet. Etter testing med ulike verdier og gjennomkjøringer ble det avdekket at en terskel på omtrent -2500 Pascal/s var en fornuftig grense for å ignorere støy i målingene i tillegg til å fortsatt være lav nok til å oppdage begynnelsen av avluseren. Etter å ha funnet et vindu der trykkfallet er større en den spesifiserte grensen, traverseres det 10 sekunder tilbake, for å sikre at det ikke kuttes bort aktuell data før start. I enkelte gjennomkjøringer ble det registrert en liten trykkøkning i begynnelsen, noe som er grunnen til at det er valgt å gå 10 sekunder tilbake fra det beregnede startpunktet slik at eventuelle trykkøkninger ikke forsvinner.



Figur 4.1.2: Illustrasjon av starttid-funksjon med to grafer av samme gjennomkjøring. Figuren til venstre inneholder originaldata umanipulert, mens i figuren til høyre har starten blitt kuttet ved hjelp av starttid-funksjonen.

4.1.3 Behandling av magnetbånd-data

På ulike steder i avluseren kan det plasseres magnetbånd for å estimere Sensorfiskens posisjon ved hjelp av det innebygde magnetometeret. Utregningen av magnetbånd-posisjonene tar utgangspunkt i tilsvarende funksjon fra Python-kode gitt av SINTEF Ocean AS. Først beregnes den euklidske normen, altså lengden til magnetdata-vektoren, før dataene normaliseres for å tilpasses på et intervall fra 0 til 1. Deretter blir datamaterialet filtrert, slik at forskjeller mellom verdiene blir utjevnet. Dette gjøres for å unngå at det registreres to lokale toppunkt der det egentlig skulle være kun ett, som følge av en støyfull måling. Deretter benyttes dette filtrerte datasettet til å finne lokale toppunkt med verdi over en gitt terskel, slik at det ikke registreres magnetbånd der annen magnetisk støy har gitt utslag. For en gjennomkjøring vil brukerne av systemet ha innsikt i hvor de fysiske magnetbåndene er plassert i avluseren, og dermed vil den grafiske fremstillingen av de kalkulerte dataene gi en omtrentlig posisjonsestimering i avluseren.

4.1.4 Behandling av akselerasjon og gyrometrisk data

Før prosjektet startet hadde allerede SINTEF Ocean AS algoritmer for å regne ut G-krefter fra akselerasjon og gyrometrisk data. Utregning av G-krefter i denne applikasjonen tar utgangspunkt i disse algoritmene og foregår ved å regne ut roll, pitch og yaw-en (RPY) til Sensorfisk ut i fra akselerasjon- og gyrometrisk data. Deretter roteres disse til North East Down (NED) før de kompenseres for gravitasjonen og regner om fra akselerasjon til G-krefter.

Samplingsfrekvensen til akselerometeret og gyroskopet på Sensorfisken er 1000 Hz, men sammen med oppgavestiller så ble det bestemt at vi kunne nedsample denne til 200 Hz. Nedsamplingen ble gjort veldig enkelt ved å ta hvert 5 punkt fra den opprinnelige dataen.

Dette ledet til at tiden applikasjonen bruker på å behandle disse dataene ved opplastning ble redusert. Samtidig reduserte dette lagringsplassen G-kreftene bruker i databasen, noe som førte til at det også går raskere å hente ut dataen i ettertid.

4.1.5 Utregning av støt

Støt mot rørveggen blir kalkulert ved hjelp av dataene for G-krefter. Funksjonen er utarbeidet med utgangspunkt i tilsvarende funksjon gitt ved Python-scriptene fra SINTEF Ocean AS. Metoden fungerer ved at dataen for G-krefter deles opp i vinduer spesifisert ved et antall sekunder og deretter kalkuleres median absolutt avvik (MAD) i disse vinduene. Median absolutt avvik er et mål på hvor spredt et datamateriale er, og brukes gjerne i datasett som inneholder verdier som er ekstremt lave eller store (Rousseeuw & Croux, 1993). MAD blir deretter multiplisert med en selvvalgt kritisk verdi, som tilsier hvor sensitiv kollisjonskalkuleringen skal være. Lave verdier vil her gjøre kalkuleringen mer sensitiv og resultere i flere antall støt. For å finne øvre grense for hva som ikke skal registreres som et støt blir den multipliserte verdien addert med gjennomsnittet for datamaterialet i samme vindu. Ved å justere på vindusstørrelse og kritisk verdi vil man "godta" ulike grenser for hva som skal kjennetegnes som støt, noe som vil gi utslag i antall støt og deres respektive størrelse i G-krefter.

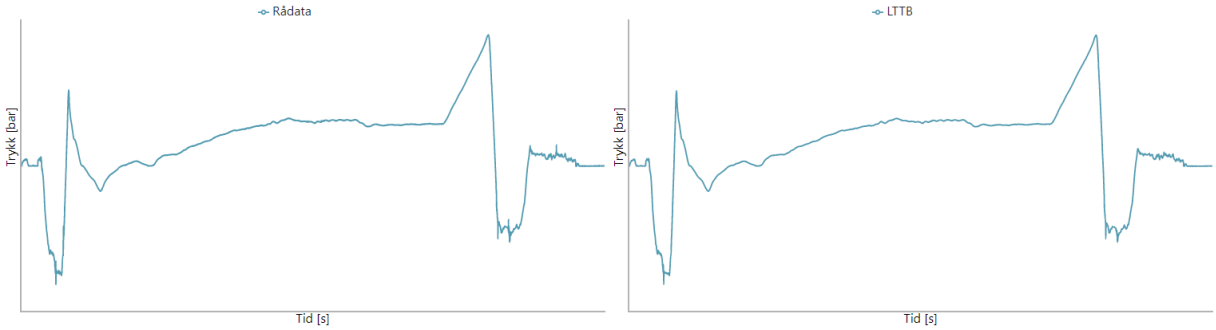
4.1.6 Nedsamplingsalgoritmer

Vi bruker nedsamplingsalgoritmer for å unngå forsinkelser og tregheter ved visualisering av dataen. Gjennom eksperimentering ble det oppdaget at applikasjonen strevde med visualisering over 15000 datapunkter, og det var først ved å reduserte antall datapunkter til 6000 at det ikke oppstod noen forsinkelser eller tregheter. Siden G-krefter grafen inneholder relativt mye støy og det er viktig å beholde flest mulig ekstremalpunkter, valgte vi å beholde 6000 datapunkter på denne grafen. Derimot så vi det ikke som nødvendig å beholde 6000 datapunkter av trykk- og temperaturgrafene, derfor bestemte vi oss for å beholde kun 3000 datapunkter for disse grafene.

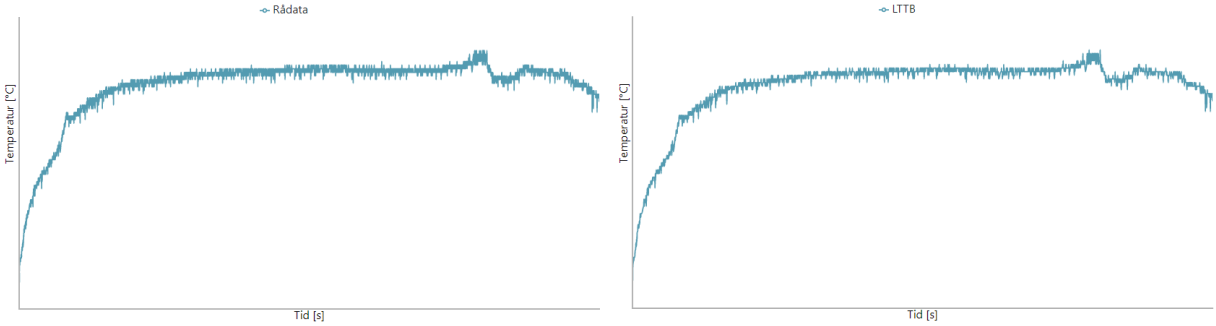
Etter nedsamplingen av dataen så ble grafene vist frem uten merkbare forsinkelser og tregheter, og algoritmene vi bruker til nedsamplingen klarte å beholde den viktigste informasjonen på grafen uansett om vi hadde redusert antall datapunkter på hver graf. Videre i dette delkapittelet forklares det hvilke algoritmer vi brukte for hver av grafene og det vises figurer med grafene før og etter nedsampling.

Trykkgrafen og temperaturgrafen

For trykk- og temperaturgrafen valgte vi å bruke "Largest Triangle Three Buckets" (LTTB) utviklet av Steinarsson i hans mastergradsavhandling (se kapittel 3.3.1). Figur 4.1.6.1 og figur 4.1.6.2 viser henholdsvis trykk- og temperaturgrafen med visualisering av rådataen (10143 datapunkter) til venstre og visualisering med LTTB nedsampling (3000 datapunkter) til høyre.



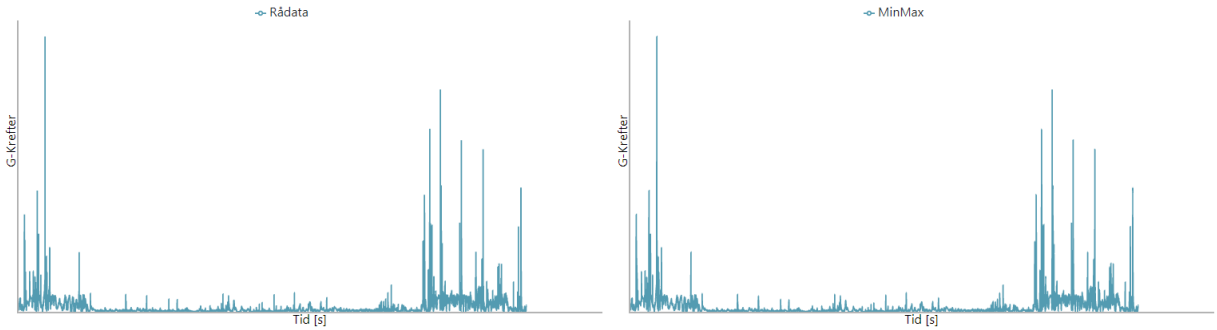
Figur 4.1.6.1: Forskjell mellom rådata og nedsamplet trykkdata. Til venstre vises rådataen med 10143 datapunkter, og til høyre vises nedsampling til 3000 datapunkter med LTTB.



Figur 4.1.6.2: Forskjell mellom rådata og nedsamplet temperaturdata. Til venstre vises rådataen med 10143 datapunkter, og til høyre vises nedsampling til 3000 datapunkter med LTTB.

Grafen av G-krefter

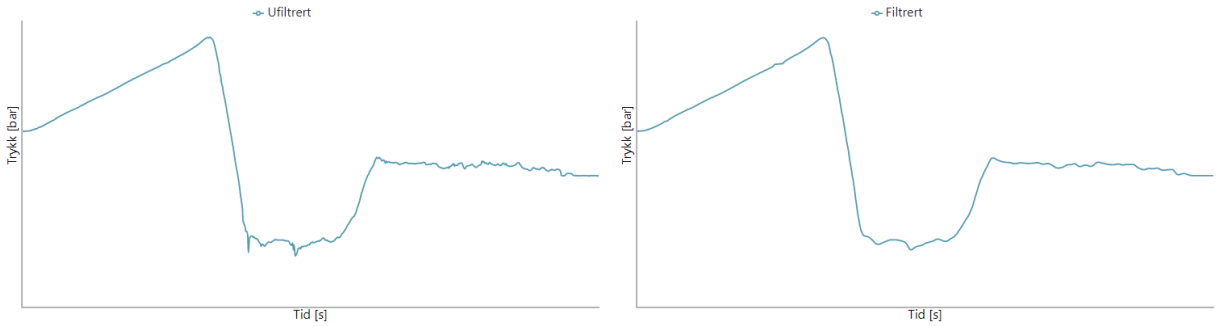
For denne grafen valgte vi å bruke en algoritme som heter MinMax og som er beskrevet av National Instruments (se kapittel 3.3.2). Denne algoritmen prøver å ta vare på maksimums- og minimumspunkter. Figur 4.1.6.3 viser at algoritmen nedsampler fra 39881 datapunkter til 6000 datapunkter og klarer å beholde de fleste maksimums- og minimumspunktene.



Figur 4.1.6.3: Graf for G-krefter med og uten nedsampling. Til venstre vises rådataen med 39881 datapunkter, og til høyre vises nedsampling til 6000 datapunkter med MinMax.

4.1.7 Filtrering av graf

Ved visualisering av datamaterialet vil det som regel være noe "støy" i målingene som representeres ved korte opp- og nedsving i grafen. Dette kan påvirke lesbarheten til grafene, spesielt om det er flere gjennomkjøringer som sammenlignes. For å motvirke dette er det utviklet mulighet for å filtrere datamaterialet. I dette tilfellet er det brukt et eksternt bibliotek kalt *timeseries-analysis* for å fjerne uønsket støy. Ved hjelp av dette biblioteket blir materialet delt opp i ulike perioder, der dataen innenfor disse områdene blir justert slik at den har mindre avvik i forhold til de nærliggende datapunktene. Dermed vil den visualiserte grafen gi inntrykk av at den har blitt "glattet ut". Dette vises på figur 4.1.7 under som sammenligner en filtrert og ufiltrert versjon av samme graf.



Figur 4.1.7: Filtrert og ufiltrert trykkgraf. Figuren er zoomet inn på en viss del av gjennomkjøringen. Ufiltrert graf til venstre og filtrert graf til høyre.

4.1.8 Design av systemet

Et av de viktigste målene som ble satt i starten av prosjektet var at systemet måtte være enkelt i bruk. Systemet skal tas i bruk av ansatte ved SINTEF Ocean AS, men ambisjonen er at det også skal kunne tilbys til kunder som ønsker å se data for sine mekaniske avlusere. Den tekniske kompetansen til brukerne av systemet kan derfor variere mye, og det er viktig at systemet er såpass brukervennlig at det kan brukes av alle. Forskningsspørsmål 1 handler

derfor om hvilken teknologi som bør tas i bruk for å lage et system som er mer brukervennlig enn dagens løsning. I prosjektet er det tatt i bruk brukergrensesnittbiblioteket React (se kapittel 3.2.2.1) og CSS-biblioteket React-Bootstrap (se kapittel 3.2.2.2), samt teorien fra kapittel 2.1 som et forsøk på å oppnå et mest mulig brukervennlig system.

Innloggingsside

Innloggingssiden er den siden brukeren først møter og har felter for utfylling av epost og passord. Knappen for innlogging er deaktivert helt til brukeren har fylt ut begge feltene som vist i figur 4.1.8.1. Dersom brukeren ikke husker passordet sitt og ønsker å endre dette, kan han/hun trykke på knappen "Glemt passord?" for å gå til en side hvor passordet kan tilbakestilles. Innloggingssiden gir også tilbakemelding i form av en rød tekst dersom brukeren ikke har skrevet riktig kombinasjon av epost og passord.

Figur 4.1.8.1: Innloggingssiden med felter for epost og passord. Knappen hvor man trykker logg inn er deaktivert helt til brukeren har fylt ut begge felter.

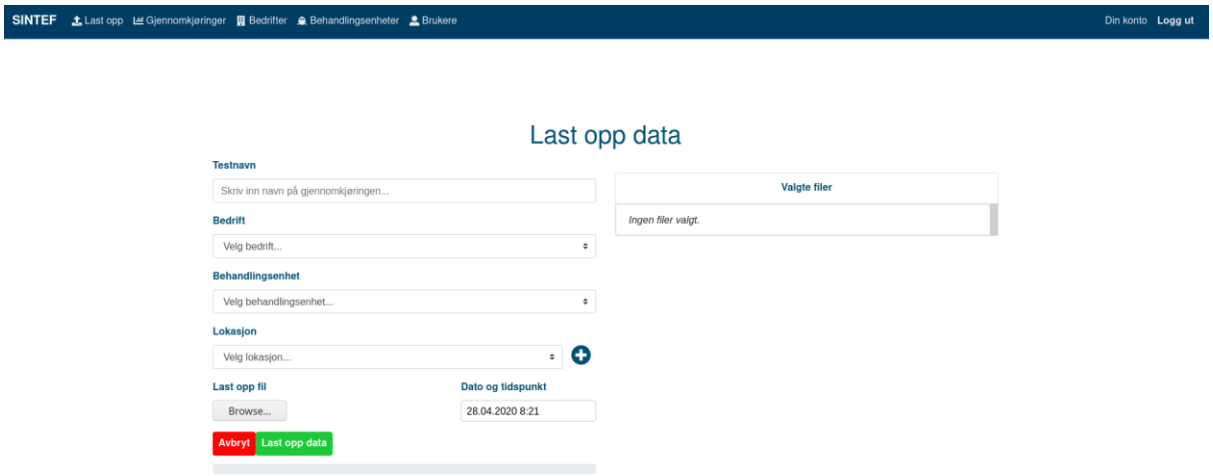
Kontrollpanel

Kontrollpanelet viser oversikten over de forskjellige sidene en administratorbruker kan velge mellom. De forskjellige sidene er representert med et beskrivende ikon og tilhørende tekst. Ikonene brukes som et forsøk på å engasjere brukerens mentale modell (se kapittel 2.1.1). Når administratorbrukeren er logget inn vises også alle muligheter på navigasjonsbaren øverst i vinduet.

Figur 4.1.8.2: Kontrollpanelet i applikasjonen. Kontrollpanelet er minimalistisk og inneholder fem forskjellige klikkbare muligheter for brukeren. Hver mulighet består av en beskrivende tekst og et tilhørende ikon.

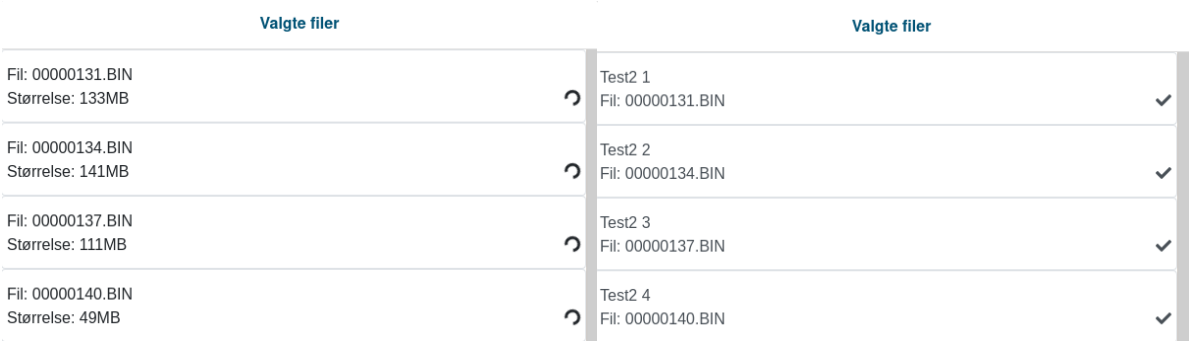
Opplastingsside

På opplastingssiden kan en administratorbruker laste opp én eller flere binærfiler til systemet. Det fylles ut informasjon om gjennomkjøringen til venstre på figur 4.1.8.3, og de valgte filene vises i listen til høyre.



Figur 4.1.8.3: Side for opplasting av data. Denne består av flere felt for utfylling av informasjon til venstre, og en oversikt over valgte filer til høyre.

Listen med valgte filer har et ikon til høyre som gir en indikasjon på om filen lastes opp eller er ferdig lastet opp. Når alle filer er lastet opp blir elementene i listen klikkbare, slik at brukeren kan gå rett til siden for visning av data for en gjennomkjøring. Både listen over opplastede filer samt det faktum at brukeren får tilbakemelding dersom felter ikke er fylt ut i skjemaet gjør systemet mer forståelig i bruk.

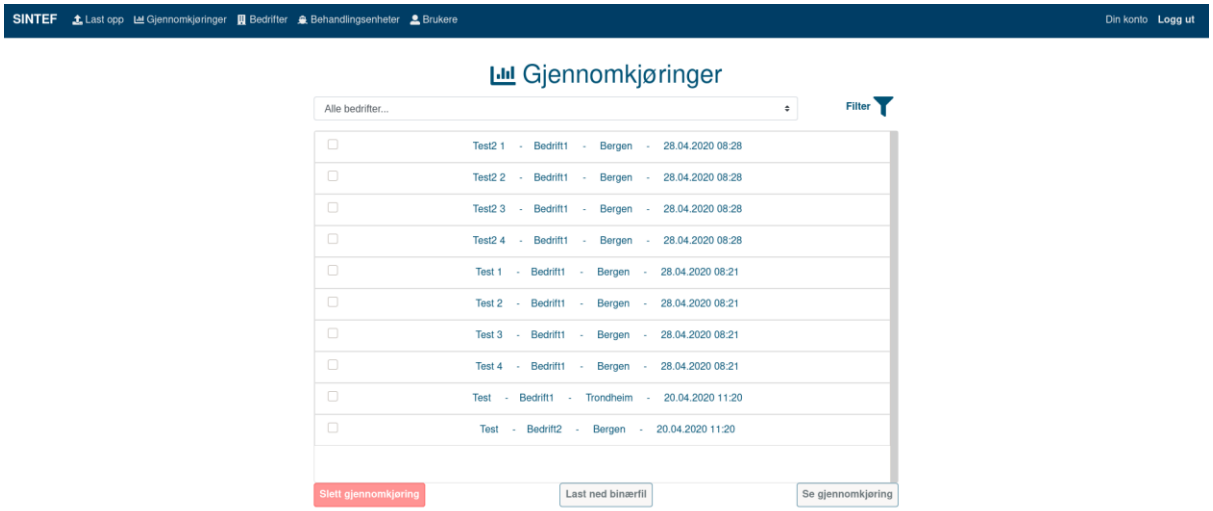


Figur 4.1.8.4: Oversikt over valgte filer. Ikonene viser om filene er i ferd med å bli lastet opp (venstre) eller om opplastingen er ferdig (høyre).

Oversikt over gjennomkjøringer

På siden som viser alle gjennomkjøringer kan en administratorbruker velge hvilke(n) gjennomkjøring vedkommende vil se. Det er mulighet for å trykke på én gjennomkjøring for å se denne, eller så kan én eller flere gjennomkjøringer markeres. Dersom gjennomkjøringer er markert kan disse slettes, sees eller det kan lastes ned tilhørende binærfil(er). Nedtrekksmenyen over listen, samt "Filtrer"-knappen kan brukes for å filtrere hvilke gjennomkjøringer som skal vises i listen. Det er mulighet for å filtrere på hvilken bedrift, hvilken avluser eller hvilket sted gjennomkjøringen tilhører, samt at man kan velge et tidsrom.

Denne siden fungerer også som "hovedside" for en bedriftsbruker, da bedriftsbrukere ikke skal ha tilgang til den samme funksjonaliteten som en administrator. Bedriftsbrukerne skal kun ha tilgang til egne gjennomkjøringer, som da vil vises i listen i figur 4.1.8.5.



Figur 4.1.8.5: Side med oversikt over alle gjennomkjøringer. Lista har mulighet for å velge en enkelt gjennomkjøring eller markere flere. Dersom én eller flere markeres, har man mulighet til å slette, se eller laste ned tilhørende binærfil. Brukeren kan også filtrere hvilke gjennomkjøringer som vises i listen.

Visning av gjennomkjøringsdata

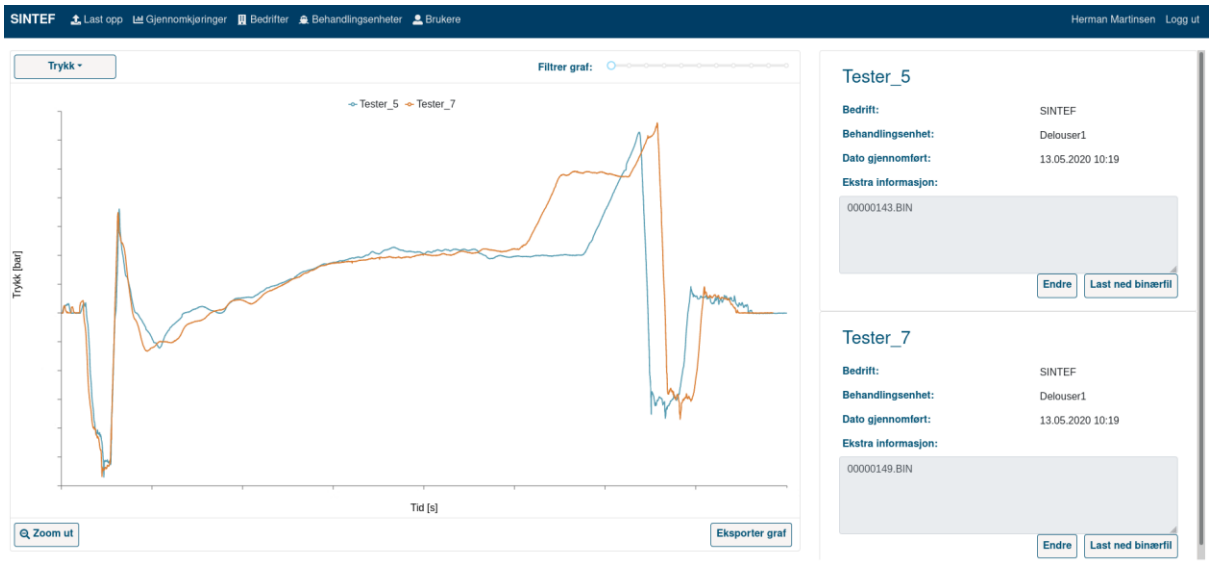
På siden som viser data fra en gjennomkjøring ligger det mye funksjonalitet som var ønsket fra oppgavestiller. På høyre side finner man informasjon om hvilken bedrift gjennomkjøringen tilhører, hvilken behandlingsenhet den ble kjørt gjennom og hvilken dato den ble gjennomført. Brukeren kan legge til ekstra informasjon som er relevant i tekstboksen under, samt endre navnet på gjennomkjøringen ved å trykke på knappen “Endre”. Det er også mulighet for å laste ned binærfilen fra tjeneren dersom brukeren vil ha denne lokalt på sin maskin.

På venstre side finner man selve grafen som fremstiller dataen. I nedtrekksmenyen kan brukeren velge mellom å se data for trykk, temperatur, støt eller g-krefter, og grafen vil endre seg etter behov. Ved siden av nedtrekksmenyen finner man en glidebryter som kan dras for å filtrere bort støy i grafen.

For å zoome inn på grafen klikker man og holder nede musepekeren på ønsket startsted, og drar til ønsket slutt. Mens man holder musepekeren ned vil området markeres, og for å få grafen slik den opprinnelig var brukes “Zoom ut”-knappen under grafen.

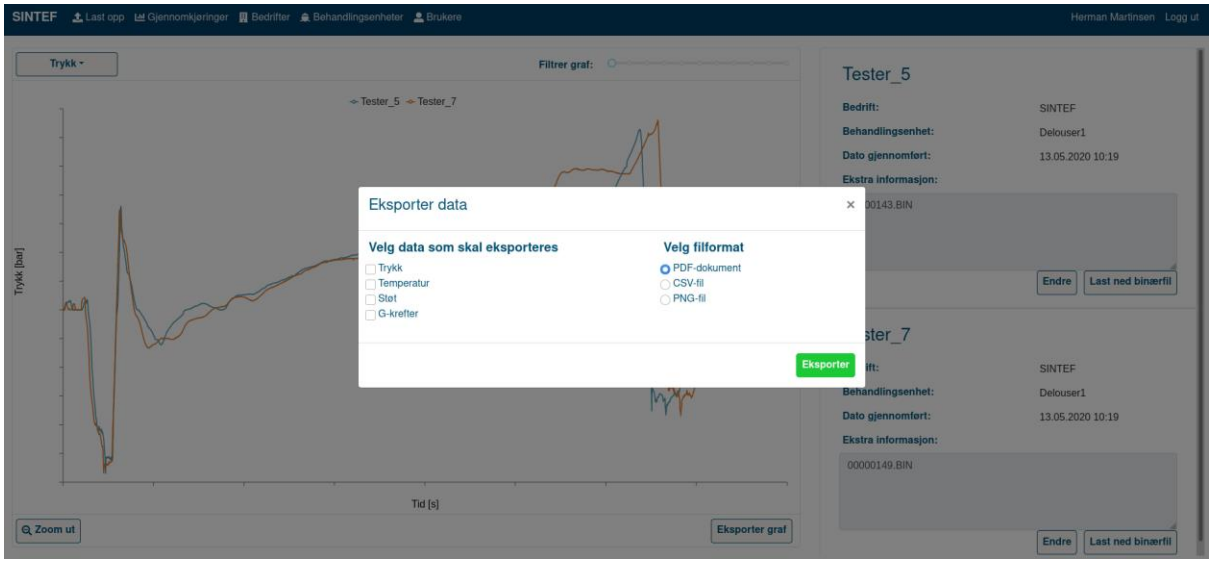
Figur 4.1.8.6: Side for visning av en enkelt gjennomkjøring. Det vises informasjon om gjennomkjøringen til høyre, og grafen (trykk som standard) vises til venstre.

Dersom man trykker på "Sammenlign" under grafen vil man gå tilbake til siden for gjennomkjøringer (se figur 4.1.8.5), og gjennomkjøringen man var inne på vil være markert. Brukeren markerer så hvilke gjennomkjøringer denne gjennomkjøringen skal sammenlignes med. Det kan sammenlignes opp til 6 gjennomkjøringer totalt, og siden vil da vise informasjon om alle gjennomkjøringene, samt vise grafene i samme diagram (se figur 4.1.8.7).



Figur 4.1.8.7: Side for sammenligning av flere gjennomkjøringer. Det vises informasjon om hver gjennomkjøring på høyre side, mens grafene ligger i samme diagram.

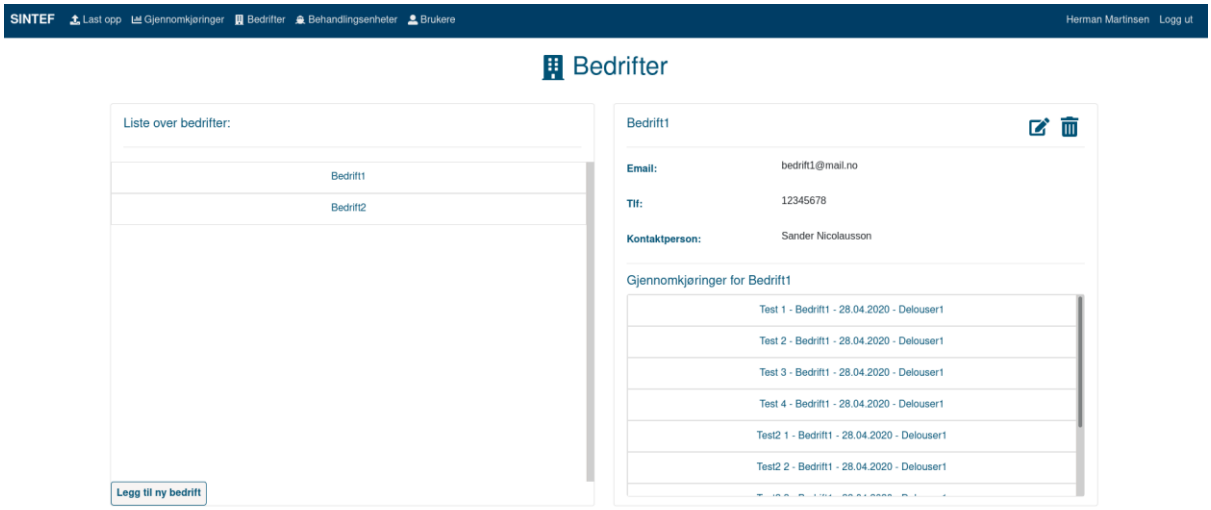
Dersom brukeren ønsker å eksportere dataene som er valgt som en fil på sin datamaskin, kan knappen "Eksporter graf" benyttes. En boks vil da dukke opp som tilbyr valg om hvilke data som skal eksporteres og i hvilket filformat (se figur 4.1.8.8). PDF gir brukeren en .pdf-fil som inneholder grafene med valgt data, og informasjon om gjennomkjøringen(e). CSV gir brukeren en .csv-fil det er mulig å åpne i Microsoft Excel. Denne filen inneholder alle datapunktene for valgte datatyper. Dersom det velges PNG, eksporteres kun grafene som .png-filer.



Figur 4.1.8.8: Modal for eksportering av data. Brukeren gis mulighet til å eksportere de forskjellige typene data for gjennomkjøringen(e) i PDF-, CSV- eller PNG-format.

Administrasjonssider

Administratorbrukeren har tilgang til sider i systemet som tillater diverse administrasjon. Dette inkluderer å legge til, endre og slette bedrifter, behandlingsenheter og brukere. Disse sidene er basert på samme oppsett, og ser derfor nokså like ut. Figur 4.1.8.9 viser siden for administrasjon av bedrifter og gir en representasjon av hvordan alle administrasjonssidene er bygd opp.



Figur 4.1.8.9: Side for administrering av bedrifter. Siden viser en oversikt over hvilke bedrifter som ligger i systemet på venstre side, og en oversikt over valgt bedrift på høyre side.

4.1.9 Sikkerhet

Dette kapittelet gir en kort oversikt over hvordan ulike sikkerhetstiltak har blitt implementert i systemet. For en mer omfattende beskrivelse av sikkerhetstiltakene anbefales det å lese i systemdokumentet (se vedlegg D, kapittel 7).

I systemet hashes passordene til brukerne før de lagres i databasen. Dette gjøres for at passordene skal være bedre sikret ved et eventuelt angrep mot systemet. Ifølge Defuse Security (2019) bør det brukes en godt designet nøkkelstrekkings-algoritme til passord-hashing, som for eksempel PBKDF2, bcrypt eller scrypt. Vi valgte å bruke bcrypt, som er en passord-hashing algoritme som baserer seg på Blowfish chifferet. Denne ble valgt fordi den var relativt enkel å implementere, med et ferdig bibliotek som var tilgjengelig for Node. I tillegg har den en kostfaktor som enkelt kan justeres for å tilpasse beregningstiden for algoritmen. For å finne en optimal kostfaktor for systemet testet vi ulike verdier, og valgte en verdi som ga et godt kompromiss mellom sikkerhet og ytelse. Bcrypt tar seg av salting av passord automatisk, og lagrer saltet som en del av hash-strengen.

For å holde brukere innlogget og sørge for at de ikke må autentisere seg på nytt hele tiden, har vi valgt å bruke access token og refresh token. Disse har blitt implementert ved hjelp av JWT, som er en åpen standard som definerer en kompakt måte å utveksle informasjon som JSON-objekter (Auth0, u.å.). Informasjonen er signert, slik at man kan verifisere at den ikke har blitt endret etter at tokenen ble laget. Både access og refresh token sendes til klienten, og refresh token lagres i tillegg i databasen for å ha mulighet til å fjerne autentisering ved behov.

Et av de sentrale spørsmålene ved bruk av access og refresh token er hvor disse skal lagres hos klienten. Både nettleserens localStorage og cookies kan benyttes til dette. En av farene ved å bruke localStorage er at variablene lagret der potensielt kan aksesseres ved hjelp av XSS. Vi valgte derfor heller å bruke cookies til dette. Cookies kan potensielt være utsatt for CSRF-angrep, men ved å sette de riktige cookie-flaggene kan man beskytte seg mot dette. Cookie-flaggene som blir satt beskrives i systemdokumentet (se vedlegg D, kapittel 7.3).

For å sørge for sikker overføring av data mellom klient og tjener har vi tatt i bruk HTTPS. HTTPS krypterer data som overføres over nettet, og sørger for at uvedkommende ikke kan fange opp sensitiv informasjon som sendes. Vi har implementert HTTPS ved hjelp av et selvsignert sertifikat, altså et sertifikat som ikke har blitt signert av en sertifikatsutsteder.

4.2 Ingeniørfaglige resultater

De ingeniørfaglige resultatene inneholder status for alle prosjektmål (se kapittel 1.3), samt status for systemet og resultat fra ulike tester som er gjennomført.

4.2.1 Status for effektmål

1. Med bakgrunn i effektivitetstesten (se kapittel 4.2.6) kan vi si at effektmål 1 er oppnådd. Systemet viser seg å være mer effektivt både ved opplasting av liten, middels stor og stor binærfil, og ved opplasting av flere filer samtidig. Det er også gjort visuelle og funksjonelle utbedringer basert på tilbakemeldinger og resultater fra brukertester (se kapittel 4.2.5 og vedlegg G, kapittel 3) for å forbedre brukskvaliteten til systemet.
2. Med dagens løsning får SINTEF Ocean AS sine kunder tilsendt en PDF-rapport per e-post som inneholder data fra sine egne anlegg. Dersom produktet blir gjort tilgjengelig for SINTEFs kunder, vil de ha mulighet til å eksportere data fra egne gjennomkjøringer på egenhånd. Dermed vil effektmål 2 være oppnådd.
3. Prosjektet er i et for tidlig stadium til å si om effektmål 3 har blitt oppnådd.
4. Effektmål 4 vil avhenge i stor grad av de andre effektmålene, da mengden data innsamlet vil avhenge av antall kunder som tar i bruk produktet.

4.2.2 Status for resultatmål

1. Vårt sluttprodukt er et web-basert brukergrensesnitt som tillater visualisering og behandling av data fra Sensorfisk. Vi kan dermed si at resultatmål 1 er oppnådd.
2. Brukerfunksjoner som zooming av graf, notater for en gjennomkjøring og eksportering av data er bestemt og utviklet i et tett samarbeid med både SINTEF Ocean AS og deres kunder. Resultatmål 2 er delvis oppnådd, men det hadde vært ønskelig å legge til enda flere brukerfunksjoner i systemet i fremtiden.
3. Det har i prosjektet blitt utviklet en serverbasert, skalerbar database for håndtering av store datamengder. Databasen er en relasjonsdatabase (se kapittel 2.3) opprettet i MySQL. Resultatmål 3 er dermed oppnådd.

4.2.3 Status for læringsmål

1. Prosjektet har gitt oss som utviklerteam god erfaring og lærdom innen utvikling av større web-prosjekter. Dette er erfaring som kan tas med senere til lignende utviklingsprosjekter, noe som betyr at læringsmål 1 er oppnådd.
2. Gjennom hele prosjektperioden har det blitt jobbet iterativt med Scrum. Det har blitt avholdt daglige stand up-møter for å skape en oversikt over hva hele teamet jobber med, og det har blitt gjort retrospektiv etter hver sprint. Vi har dermed fått mer erfaring med bruk av smidige utviklingsmetoder, og kan si at læringsmål 2 er oppnådd.
3. I løpet av prosjektet har vi hatt jevnlig kontakt med SINTEF Ocean AS og har også hatt mulighet til å være med på møter som de har hatt med sine kunder. Vi har derfor fått god erfaring med å forholde oss til en oppgavestiller som har forventninger og krav til produktet som skal leveres. Dette betyr at læringsmål 3 er oppnådd.

4.2.4 Status på systemet ved leveringstidspunkt

Ved leveringstidspunkt leveres et system som tillater opplasting, behandling og visualisering av data fra Sensorfisk. Systemet har også funksjonalitet som lar brukeren zoome inn på ønsket sted på grafen, og eksportere data til andre filtyper. I tillegg har systemet andre administrative funksjonaliteter som å legge til behandlingsenheter, bedrifter og nye brukere. Systemet leveres produksjonsklart med en installasjonsguide i systemdokumentet (se vedlegg D, kapittel 8) og i prosjektets readme.md-fil.

4.2.5 Resultater fra brukertester

Effektmål 2 for prosjektet går ut på å gjøre det enklere for SINTEF Ocean AS sine kunder å få tilgang til, behandle og visualisere data fra egne anlegg. Implisitt betyr dette at systemet også vil føre til forenkling av arbeidet for ansatte i SINTEF Ocean AS. Med utgangspunkt i dette ble det gjennomført brukertester for å vurdere systemet underveis i prosessen. Testdeltakerne ble plukket ut av SINTEF Ocean AS, og besto av fire av deres ansatte i tillegg til to av deres kunder. Brukertestene ble gjennomført ved hjelp av videosamtale gjennom Microsoft Teams. For utfyllende beskrivelses av gjennomføringen henvises det til *Rapport fra brukertester* i vedlegg G. Resultatene ble ført i et felles skjema for å forenkle analyse og sammenligning. Oppsettet kan visualiseres ved å presentere resultatene for oppgave 3 - sammenligning av gjennomkjøringer (se vedlegg G):

Tabell 4.2.5.1: Resultater fra brukertestene for oppgave 3, sammenligning av gjennomkjøringer.

Testdeltaker	Ønsket standard	Suksess-kriterium	Notater
S1	✔	✔	Trykker på sammenlign-knappen. Trykker først direkte inn på en annen gjennomkjøring istedenfor å huke av og sammenligne.
S2	✔	✔	Sammenligner korrekt ved å trykke sammenlign-knappen samt huke av ønsket gjennomkjøring.
S3	✔	✔	Huker av riktig og sammenligner grafene.
S4	X	✔	Trykker først på gjennomkjøringer og deretter huker av to gjennomkjøringer derfra.
K1	✔	✔	Trykker sammenlign-knappen og huker av en ekstra gjennomkjøring.
K2	✔	✔	Trykker sammenlign-knappen og velger korrekt gjennomkjøring.

I dette tilfellet var ønsket standard at deltakeren trykket på “sammenlign”-knapp direkte på grafoversikten og suksesskriteriet var at deltakeren fikk sammenlignet trykk-grafen for gjennomkjøringene. Testdeltakerne er beskrevet med forkortelsene S eller K nummerert, som representerer henholdsvis SINTEF-ansatt og en kunde av SINTEF. I siste kolonne er det

beskrevet ulike notater gjort av testpersonell underveis i løsningen av oppgaven. For å se fullstendige resultater fra brukertestene henvises det til *Rapport fra brukester* i vedlegg G.

Som nevnt tidligere har brukertestene den hensikt å vurdere brukervennligheten samt forbedre brukskvaliteten til systemet. Det vil derfor være naturlig at det gjennom testene vil avdekkes diverse problemområder. Typiske oppdagelser vil for eksempel være hvis man ser at flere av deltakerne har problemer med å gjennomføre visse oppgaver. Fra *Rapport fra brukertester* har vi følgende problemområder med tilhørende løsninger (se vedlegg G, kapittel 3):

Tabell 4.2.5.2: Problemområder og tilhørende foreslåtte løsninger funnet i forbindelse med brukertester.

Problem	**Foreslått løsning**
"Slett gjennomkjøring"-knappen antas å være der man trykker for å se gjennomkjøringen. Deltaker S2, S3 og S4 slettet gjennomkjøringer ved et uhell når de ville se grafene. Ingen bekreftelsesdialog ved sletting, sårbart for feilklikk.	Bytte plass på knappene "Se gjennomkjøring" og "Slett gjennomkjøring". I tillegg anbefales det å legge til en bekreftelsesdialog ved sletting, som spør brukeren om man er sikker på at man ønsker å slette.
Vanskelig å finne slettede gjennomkjøringer. Alle deltakerne brukte lang tid for å finne disse, samt at tre av dem fortalte at den ikke var intuitivt plassert. Enkelte deltakere mente at denne ikke burde vært under "filtrering", men heller vært en egen knapp eller sjekkboks på oversiktsiden.	Synliggjøre slettede gjennomkjøringer, i form av en knapp/sjekkboks på gjennomkjørings-oversikten. Mulighet til å direkte kun se slettede gjennomkjøringer (eventuelt i tillegg til ikke-slettede) og filtrere på disse. Tydeliggjør hvor man finner slettede gjennomkjøringer fra oversikten.
"Last ned"-knappen ble forvekslet som eksportering av graf. Det oppstår noe forvirring når deltakerne skal eksportere graf. Deltaker S2 trykker last-ned knappen og tror denne vil eksportere grafen, mens andre deltakere også uttrykte at de ønsket å trykke last ned.	Klargjøre betydningen av last-ned knappen. Per nå er den litt diffus da det kun står "Last ned" på knappen. En eventuell løsning kan være å endre navnet til "Last ned binærfil" eller noe lignende, slik at det kommer tydelig frem at dette ikke vil være å eksportere graf eller noe lignende.

På små skjermer må man stadig bla seg nedover på sidene. I de fleste tilfellene ble sidene vist på korrekt og naturlig måte, men deltaker K2 hadde en noe mindre skjerm på sin bærbare datamaskin, og måtte derfor bla nedover på sidene for å se all informasjonen. Dette var spesielt problematisk på graf-oversikt siden, da knapper som "eksporter", "sammenlign" og "zoom ut" kom utenfor synsfeltet på skjermen.	Utvidet støtte for diverse skjermstørrelser. Hovedprinsippet er at det vil være ønskelig å få graf-oversikten med tilhørende knapper i ett skjermbilde, uten at det skal være behov for å bla seg nedover på siden. Dette vil i hovedsak gjelde for datamaskiner, da mobiler/tablets sannsynligvis vil behøve noe scrolling.

4.2.6 Sammenligning av effektivitet

Et av målene med prosjektet var å lage en løsning som gjør behandling og visualisering av data fra Sensorfisk mer effektivt. For å finne ut om vi har lyktes med dette bestemte vi oss for å gjennomføre noen enkle tester. Testene gikk ut på å sammenligne tiden det tar å gå fra Sensorfiskens ubehandlede rådatafiler til ferdig visualiserte grafer i de to løsningene. Vi gjennomførte testene på egenhånd, på datamaskinene vi bruker til vanlig. For den nye løsningen ble tjener og klient kjørt på forskjellige nettverk, for at overføringstiden for filene skulle bli så reell som mulig. For at tidene ikke skulle bli for mye påvirket av maskinvaren gjennomførte vi 3 tester på 3 ulike datamaskiner, og regnet deretter ut snittet av tidene.

For begge løsningene startet vi testene med all programvare åpnet på forhånd, for at oppstartstiden til programmene ikke skulle innvirke på resultatene. For web-applikasjonen ble innlogging gjort i forkant av testen. Alle rådata-filene lå plassert på skrivebordet før testene startet, og testene ble avsluttet i det øyeblikket testdeltageren fikk opp en visualisert graf på skjermen. Fremgangsmåten for de to løsningene var følgende:

Tidligere løsning:

1. Bruke programmet MissionPlanner for å konvertere rådatafiler til Matlab-filer.
2. Kjøre Python-scriptene på Matlab-filene for å generere grafer.
3. Åpne en av de genererte grafene etter at Python-scriptene er ferdig med å kjøre.

Ny løsning:

1. Klikke seg inn på siden for opplastning av data.
2. Fylle inn opplysningene for gjennomkjøringen.
3. Velge binærfiler som skal lastes opp og klikke på "last opp"-knappen.
4. Klikke seg inn på gjennomkjøringen når opplastingen er ferdig for å se grafene.

Effektivitetstest med varierende filstørrelse

Først ble testene gjennomført med enkeltfiler som hadde varierende filstørrelse. For hver test ble det gjort behandling av en liten fil på 12,5 MB, en middels stor fil på 39,8 MB, og en stor fil på 128 MB. Resultatene fra testene er presentert i tabellen nedenfor.

Tabell 4.2.6.1: Resultater fra effektivitetstest, varierende filstørrelse.

Oppgave	Test nr.	Tidsbruk tidligere løsning			Tidsbruk ny løsning		
		Liten fil (12,5 MB)	Middels stor fil (39,8 MB)	Stor fil (128 MB)	Liten fil (12,5 MB)	Middels stor fil (39,8 MB)	Stor fil (128 MB)
Fra binærfil til ferdig graf (varierende filstørrelse)	1	01:54	04:28	12:45	00:29	00:53	02:07
	2	01:30	03:25	10:34	00:22	00:50	02:05
	3	01:27	03:52	11:29	00:18	00:37	01:35
	Snitt	**01:37**	**03:55**	**11:36**	**00:23**	**00:46**	**01:55**

Effektivitetstest med varierende antall filer

I neste omgang gjennomførte vi testene med varierende antall filer. For å sørge for at filstørrelse ikke påvirket resultatet, benyttet vi oss av identiske kopier av den samme filen. Filen som ble brukt var på 12,5 MB, og er den samme filen som har blitt betegnet som "liten fil" tidligere. For hver test ble det lastet opp 1, 3 og 5 filer. Resultatene fra testene er presentert i tabellen nedenfor.

Tabell 4.2.6.2: Resultater fra effektivitetstest, varierende antall filer.

Oppgave	Test nr.	Tidsbruk tidligere løsning			Tidsbruk ny løsning		
		1 fil	3 filer	5 filer	1 fil	3 filer	5 filer
Fra binærfil til ferdig graf (varierende antall filer)	1	01:50	04:08	06:34	00:29	00:55	01:10
	2	01:15	03:21	05:21	00:21	00:45	01:07
	3	01:16	03:14	05:11	00:18	00:28	00:36
	Snitt	**01:27**	**03:34**	**05:42**	**00:22**	**00:42**	**00:57**

For den nye løsningen vil også nettverkshastigheten være en faktor som kan spille inn på testresultatene. Derfor ble nettverkshastigheten til testdeltagerne målt i forkant av testene. Vi målte også nettverkshastigheten for datamaskinen som kjørte tjeneren, da dette også kan være relevant informasjon.

Tabell 4.2.6.3: Nettverkshastigheter målt i forkant av effektivitetstest.

Test nr.	Hastighet nedlastning (Mb/s)	Hastighet opplastning (Mb/s)
1	24,66	11,41
2	79,93	10,45
3	26,84	26,70
Tjenermaskin	72,34	75,69

4.3 Administrative resultater

Administrative resultater beskriver hvordan måloppnåelsen ble i forhold til fremdriftsplanen, og inneholder dokumentasjon på valgt utviklingsprosess.

4.3.1 Utviklingsprosess

Gjennom arbeidet med dette prosjektet har vi benyttet utviklingsmetodikken Scrum. En slik smidig utviklingsprosess forenkler justeringer og endringer i krav underveis, da det arbeides iterativt - i form av sprinter (se kapittel 2.4.1). I prosessen har det blitt utarbeidet en rekke dokumenter som muliggjør denne arbeidsformen på korrekt måte. Typiske dokumenter er produkt- og sprintbackloger, sprintplan, burndown-charts og retrospektiver. Videre vil det bli presentert et utvalg av dokumentene som er utarbeidet gjennom prosessen. For en oversikt over all Scrum-dokumentasjon, se vedlegg F. Sprintoversikten gir en oversikt over tidsperspektivet opp mot de ulike oppgavene fra produkt-backlogen. Siste versjon av sprintoversikten er som følgende:

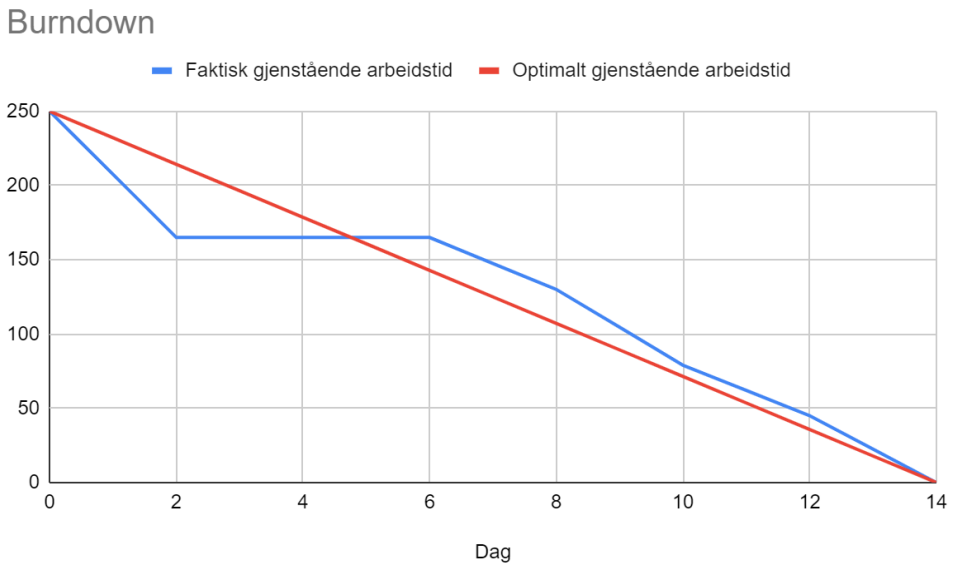
Tabell 4.3.1.1: Sprintoversikt med dato, milepæler fra produkt-backlog og eventuelle merknader.

Sprinter	**Dato**	**Milepæler**	**Merknader**
Sprint 1	27.01.2020 - 07.02.2020 (2 uker)	Opprette database (6) Vise graf for trykk (11)	
Sprint 2	17.02.2020 - 09.03.2020 (3 uker)	Histogram for støt og g-krefter (11) Sammenligne grafer (8) Enkel frontend-prototype Filtrering av graf (5) Velge behandlingsenhet (4)	Ekstra lang grunnet obligatorisk innovasjonscamp og CCD
Sprint 3	10.03.2020 - 30.03.2020 (3 uker)	Mulighet for opplastning av binærfil (10) Eksportere filer (7)	Eksamen 18.03
Sprint 4	31.03.2020 - 20.04.2020 (3 uker)	Opprette ny bruker (1) (12) Logge inn (2) Endre passord (3)	Brukertester 15.04-17.04
Sprint 5	21.04.2020 - 04.05.2020 (2 uker)	Endre informasjon om kunde (13) Slette kunde (14) Slette gjennomkjøringer (9)	

Før hver sprint kreves det planlegging der det blant annet blir fastsatt hva som skal gjøres i løpet av sprinten. Oppgaver fra produkt-backlogen velges ut og deles opp i flere oppgaver, og dette danner samlet grunnlaget for sprint-backlogen. Vi valgte å kombinere sprint-backlogen og burndown-chart i samme Excel-dokument slik at vi får samlet sprint-informasjon og fremgang på ett sted. Her vises eksempel på sprint-backlog med tilhørende burndown-chart fra sprint 3:

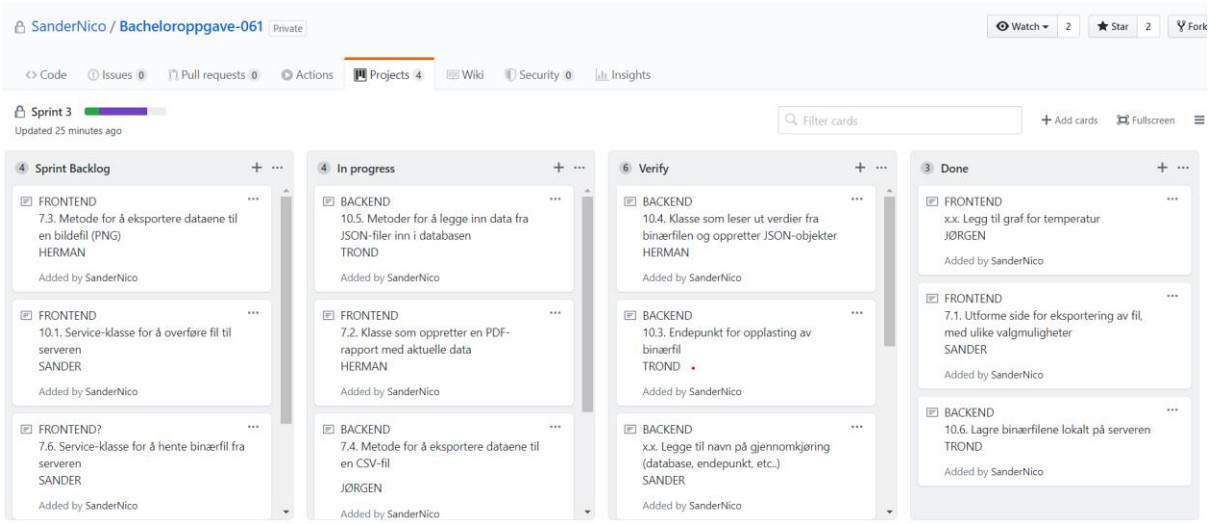
SPRINTMÅL: Opprette mulighet for å laste opp binærfil, samt eksportere graf og datafiler									
Systemkrav fra produkt backlog	**Sprint 3 arbeidsoppgave**	**Originalt estimerte innsats**	**Gjenstående innsats etter dag:**						
			2	**4**	**6**	**8**	**10**	**12**	**14**
Mulighet for opplastning av binærfil (10)	Service-klasse for å overføre fil til serveren	10	10	10	10	2	1	1	0
	Utforme sider for opplastning av gjennomkjøring	10	10	10	10	5	2	2	0
	Endepunkt for opplastning av binærfil	20	5	5	5	2	1	0	0
	Klasse som leser ut verdier fra binærfilen og oppretter JSON-objekter	35	20	20	20	15	5	5	0
	Metoder for å legge inn data fra json filer inn i databasen	15	10	10	10	6	4	2	0
	Lagre binærfilene lokalt på serveren	15	5	5	5	4	3	2	0
	Legge inn utregning av støt og posisjonsestimering av magnetbånd ved opplastning	15	10	10	10	6	5	2	0
Eksportere filer (7)	Utforme side for eksportering av fil, med ulike valgmuligheter	15	0	0	0	0	0	0	0
	Klasse som oppretter en pdf-rapport med aktuelle data	35	30	30	30	30	30	15	0
	Metode for å eksportere dataene til en bildefil (png)	25	20	20	20	20	20	10	0
	Metode for å eksportere dataene til en csv-fil	15	15	15	15	10	7	5	0
	Endepunkt for å hente ut binærfil	30	30	30	30	30	1	1	0
	Service-klasse for å hente binærfil fra serveren	10	10	10	10	10	1	1	0
	Sammenligne 6 gj.kjøringer	5	1	1	1	0	5	4	0
	Graf for temperatur	5	1	1	1	0	5	2	0
	SUM	**250**	**165**	**165**	**165**	**130**	**79**	**45**	**0**

Figur 4.3.1.1: Sprint-backlog for sprint 3 med oversikt over gjenstående innsats.



Figur 4.3.1.2: Burndown-chart for sprint 3. Illustrerer forskjellen mellom faktisk gjenstående arbeid (i blått) og optimalt gjenstående arbeid (i rødt).

Gjennom arbeidet med prosjektet har vi ikke hatt et fast sted å jobbe på grunn av mangel på lokaler og arbeidsrom. Dette har påvirket muligheten for å ha arbeidsoppgavene beskrevet med post-it lapper på en felles tavle, slik at man får visualisert status på de ulike oppgavene. En alternativ løsning vi har valgt er å bruke GitHub Projects som dekker det samme behovet digitalt. Der har vi mulighet til å legge til ulike "oppgavelapper" til forskjellige kolonner for å dokumentere prosessen. Under kommer et skjermbilde tatt i løpet av sprint 3:



Figur 4.3.1.3: GitHub Projects oversikt for sprint 3. Kolonnene tilsier hvilket stadium oppgavene på "lappene" er i. Oppgaver blir valgt fra "Sprint Backlog" og deretter plassert i "In Progress"-kolonnen. Når de er klare for godkjenning flyttes de til "Verify", og etter de har blitt testet/godkjent legges de inn i "Done"-kolonnen.

Etter hver sprint ble det gjennomført retrospektiver. Dette ble også satt opp digitalt slik at vi enkelt senere kunne gå tilbake og se på tidligere notater. Det ble gjennomført anonymt der hvert medlem i utviklingsteamet la inn notater for hva som gikk bra og hva som kunne gått bedre for den aktuelle sprinten. Deretter ble alle notatene gjennomgått og eventuelt utbrodert hvis de var uklare i betydning. Til slutt ble det i samråd vedtatt hva som skulle gjøres til neste sprint for å forbedre det som ikke gikk som det skulle i den nåværende sprinten.

Figur 4.3.1.4: Retrospektiv fra sprint 1. Notatene på venstre side er hva som gikk bra i sprinten, notatene i midten er hva som kunne vært forbedret mens notatene til høyre er hvilke handlinger som skal gjøres for å forbedre videre sprinter.

4.3.2 Fremdrift og timeregnskap

I starten av prosjektet ble det satt opp en fremdriftsplan med tilhørende Gantt-diagram for å planlegge prosjektperioden. Etter tilbakemelding fra veileder og endringer i behov har fremdriftsplanen blitt revidert i flere omganger (se vedlegg E, kapittel 1). Resultatet etter endt prosjektperiode vises i tabell 4.3.2.1.

Tabell 4.3.2.1: Oversikt over aktivitetene satt opp i Gantt-diagrammet med estimert antall timer per aktivitet.

Aktiviteter	**Gjenstående timer**	**Est. Antall timer**
Møter	4	100
Dokumentasjon	11,5	430
Hovedrapport	−18,5	490
Frontend	5,5	500
Backend	26,5	350
Testing	4,5	130
	0	0
	0	0
SUM:	33,5	2000

Tabell 4.3.2.2 viser en samlet timeliste fra siste prosjektuke (uke 21). I figuren ser vi hvor mange timer hvert teammedlem har jobbet i raden *akkumulert hittil i år*, og den totale summen for hver aktivitet fra Gantt-diagrammet i kolonnen *total sum aktivitet*. En fullstendig oversikt over alle uker finnes i prosjekthåndboka (se vedlegg E, kapittel 3).

Tabell 4.3.2.2: Timeliste fra siste uke av prosjektperioden. Raden "akkumulert hittil i år" viser hvor mange timer hvert gruppemedlem har jobbet, og kolonnen "total sum aktivitet" viser antall timer som har blitt brukt på hver av aktivitetene fra Gantt-diagrammet.

Uke 21 (18. mai)						
Aktiviteter	**Herman**	**Jørgen**	**Sander**	**Trond**	**Ukesum aktivitet**	**Total sum aktivitet**
Møter	0	0	0	0	0	96
Dokumentasjon	0	0	0	0	0	418,5
Hovedrapport	20	20	20	20	80	508,5
Frontend	0	0	0	0	0	494,5
Backend	0	0	0	0	0	323,5
Testing	0	0	0	0	0	125,5
	0	0	0	0	0	0
	0	0	0	0	0	0
Ukesum	**20**	**20**	**20**	**20**	**80**	
Akkumulert hittil i år	**475**	**470**	**513**	**508,5**		**1966,5**

.

5 Diskusjon

I dette kapittelet drøftes alle resultater, både empiriske, ingeniørfaglige og administrative. Det diskuteres hvorfor ting endte opp slik de gjorde, spesielt der hvor det ble avvik fra planer og mål i prosjektet.

5.1 Empiriske resultater

I følgende kapittel vil de empiriske resultatene som tidligere er presentert bli drøftet i lys av forskningsspørsmålene. Det vil bli diskutert betydningen av resultatene, hvorfor de ble slik de ble og eventuelle svakheter ved de aktuelle resultatene.

5.1.1 Innlesing av rådatafiler

Som beskrevet i kapittel 4.1.1 har vi laget en spesialisert metode for å lese inn data fra rådatafil, som kun leser inn informasjon fra 4 av pakketypene som ligger lagret i filen. Valget ble gjort på bakgrunn av at alle rådatafiler fra Sensorfisk er bygget opp på samme måte. Dermed ble det vurdert som enklere å implementere en metode som er spesiallaget for dette filformatet. I tillegg virket det høyst sannsynlig at å kun lese inn relevant data fra filene ville redusere innlesingstiden betraktelig sammenlignet med tidligere løsning. Effektivitetstesten som har blitt gjennomført tyder på at denne hypotesen stemmer (se kapittel 4.2.6). Det virker også som at metoden for innlesing av rådatafiler er relativt stabil. Metoden har klart å lese inn alle filer vi har fått tilsendt fra SINTEF Ocean AS, med unntak av noen få som manglet trykk- og temperaturdata.

En mulig ulempe med innlesingsmetoden er at den ikke er laget med tanke på eventuelle endringer. Hvis for eksempel oppbygningen av en rådatafil endres i fremtiden er det ikke sikkert at metoden fortsatt vil fungere. Det kan også hende at SINTEF Ocean AS ved et senere tidspunkt har et ønske om å lese inn mer data fra rådatafilen. Isåfall vil innlesingsmetoden måtte oppdateres og endres. En mer generell innlesingsmetode ville vært mer robust ved slike endringer, men ville også høyst sannsynlig brukt lengre tid på å lese inn rådata. Med forskningsspørsmål 2 i tankene vurderte vi derfor en spesialisert innlesingsmetode som den beste løsningen.

5.1.2 Beregning av starttid

Beregningen av starttid gir mulighet for å kutte vekk "dødtid" for gjennomkjøringen som igjen kan forbedre både effektivitet og brukskvalitet for systemet. For det første vil reduksjon av "dødtiden" redusere antall datapunkter. Som forklart i resultater fra nedsamplings-algoritmer (se kapittel 4.1.6) vil reduksjon av datapunkter gi en raskere responsiv graf og nettside. I tillegg vil visualiseringen bli tydeligere ved at den faktiske gjennomkjøringen blir sentrert i grafen, noe som øker behagelighet ved bruk og effektivitet, som er to punkter knyttet til brukskvalitet (se kapittel 2.1.3). Dette bidrar positivt med tanke på forskningsspørsmål 1, 2 og 4, som omhandler brukervennlighet, effektivitet og mangel på forstyrrelser ved visualisering.

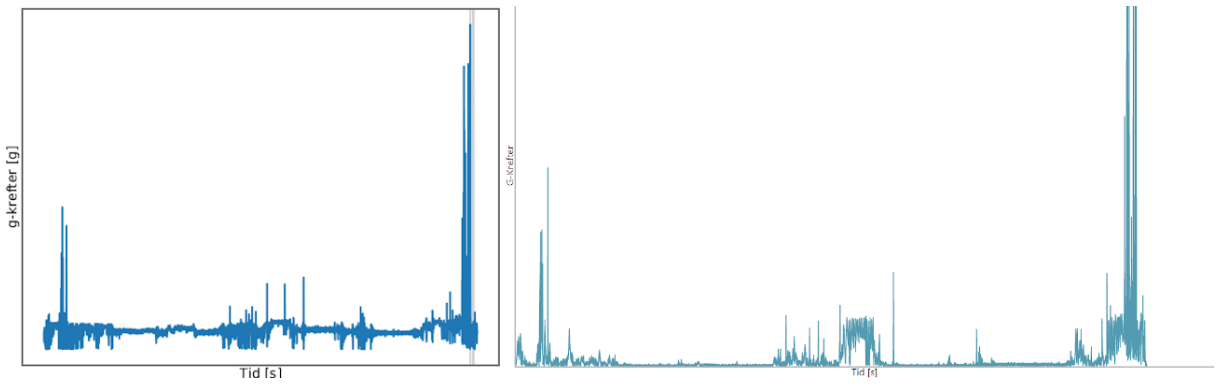
Den nåværende beregningen benytter vinduer på 3 sekunder for å finne den deriverte, mens det tidligere ble satt til et fast antall vinduer uavhengig av gjennomkjøringens størrelse (se kapittel 4.1.2). Dette ga varierende vindusstørrelser som igjen påvirket størrelsen til den deriverte. Dermed måtte terskelen for den deriverte justeres mellom ulike gjennomkjøringer med variasjoner i lengde. Ved å spesifisere vinduene til et gitt tidsintervall vil dette motvirkes ved at den deriverte alltid finnes i det samme tidsintervallet og man kan derfor benytte lik terskel for trykkfall både på lange og korte gjennomkjøringer. En svakhet med dagens kalkulasjoner er derimot at de er utarbeidet med utgangspunkt i kun to ulike behandlingsenheter og deres respektive data. Det kan derfor ikke konkluderes med at nåværende løsning er feilfri, til tross for at den har fungert bra på det aktuelle datasettet. Det er mulig at andre avlusningsmetoder vil gi en større trykkøkning eller eventuelt et mer beskjedent trykkfall i begynnelsen, som kan komme under terskelen for nåværende utregningsmetode. Sensorfisken er et relativt nytt prosjekt og for øyeblikket eneste i sitt slag, og det har derfor ikke vært mulig å innhente tilsvarende trykkdata for andre typer avlusere gjennom litteratursøk. En anbefaling videre vil derfor være å benytte Sensorfisken til å samle data fra flere avlusningsenheter og anvende denne informasjonen til å forbedre robustheten og nøyaktigheten til nåværende metode.

5.1.3 Behandling av magnetbånd-data

Magnetbånd-data blir benyttet for å estimere Sensorfiskens posisjon i avluserenheten (se kapittel 4.1.3). Som tidligere nevnt er disse basert på kode fra SINTEF, og nåværende metode innehar ingen vesentlige forskjeller annet en kodeoversettelse som følge av ulike programmeringsspråk. En automatisk visualisering av magnetbånd-posisjoner vil knyttes opp mot forskningsspørsmål 2 da dette vil kunne effektivisere rapportproduseringen. Magnetbånd-kalkulasjonene og deres respektive verdier blir visualisert som vertikale linjer i utvalgte grafer, og på denne måten vil man kunne benytte denne posisjonsestimeringen direkte ved produsering av rapport til kunden. En videreutvikling vil likevel være å sikre at magnetbåndkalkulasjonene fungerer tilsvarende på ulike avlusere. Som beskrevet ved utregning av starttid er det begrenset informasjon å innhente på dette området ved hjelp av litteratursøk, det vil derfor være naturlig å benytte seg av Sensorfisken til å samle data fra ulike avlusningsenheter for å sikre at utregningen for magnetbånd fungerer for flere enheter.

5.1.4 Behandling av akselerasjon og gyrometrisk data

Algoritmene for utregning av G-krefter er basert på algoritmer som SINTEF utviklet før dette prosjektet. Dette betyr at vi selv ikke har hatt mulighet til å gjøre noen tester på algoritmene og sjekke om de klarer å regne ut G-krefter riktig. Vi baserer oss på at SINTEF sine algoritmer gjorde utregningen korrekt. Det kan ha oppstått noen forskjeller mellom koden til SINTEF (Python) og vår versjon av den (Javascript), men vi har gjort en visuell sammenligning av g-kreftene til en gjennomkjøring i SINTEFs tidligere system og i det nye systemet. Ut ifra dette kan si at de fleste ekstremalpunkter er beholdt fra SINTEFs tidligere løsning til den nye løsningen (se figur 5.1.4).



Figur 5.1.4: Graf for G-krefter fra tidligere og ny løsning. Graf fra SINTEFs tidligere system til venstre og fra vårt system til høyre. Grafene er fra samme gjennomkjøring. Legg merke til at de fleste ekstremalpunktene er til stede på begge grafene.

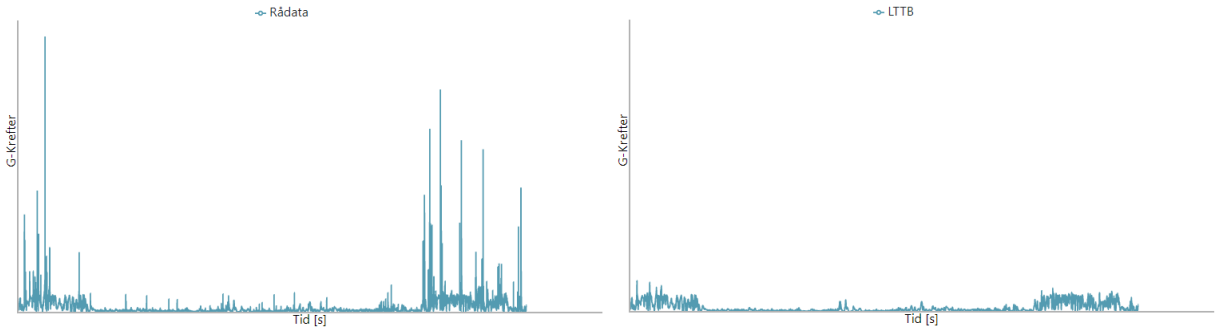
5.1.5 Utregning av støt

Som forklart i kapittel 4.1.5 er støtkalkulasjonene utformet med utgangspunkt i Python-kode gitt av SINTEF. En viktig forskjell fra den oppgitte koden er at nåværende kalkulasjoner benytter faste vindusstørrelser med et gitt tidsintervall slik det også er gjort i beregningen av starttid (se kapittel 5.1.2). Som nevnt tidligere vil dette gi et likt utgangspunkt på beregninger gjort i vinduet, uavhengig av lengden på gjennomkjøringen. Utregningen har som mål å finne lokale toppunkt i datasettet for G-krefter og gjør dette ved å se på median absolutt avvik innenfor et gitt vindu og deretter velge ut punkter etter ønskede terskelverdier (se kapittel 4.1.5). Problemet med slike kalkulasjoner er at det vil være vanskelig å fastslå hva som skal kjennetegnes som et støt eller ikke. Både størrelsen og varigheten på støtet er noe som må tas hensyn til for å sikre gyldige beregninger. Slik det er nå skiller ikke kalkulasjonen vesentlig mellom lange eller korte lokale toppunkt, altså varigheten på støtene. Dette kan være naturlig å implementere ved en videreutvikling av systemet, da varigheten av G-kreftene har en vesentlig betydning for hvilken påvirkning de har (Creer, Smedal & Wingrove, 1960, s. 13).

5.1.6 Nedsamplingsalgoritmer

Tidlig i prosjektet fant vi ut at å vise frem grafer for data med over 15000 punkter førte til forsinkelser og treghet i applikasjonen. Dette ville gjort analysering av data vanskeligere, ved for eksempel zooming eller lesing av datapunkter. For å visualisere data på en uforstyrrende måte, som beskrevet i forskningsspørsmål 4, måtte nedsamplingsalgoritmer tas i bruk. Det var viktig at disse algoritmene klarte å redusere antall datapunkter som vises frem uten å fjerne viktig informasjon fra grafene.

Den første algoritmen vi prøvde var “Largest Triangle Three Buckets” (LTTB) som er utviklet av Svein Steinarsson (se kapittel 3.3.1). Denne algoritmen klarte å beholde formen på grafen dersom den hadde lite støy (se figur 4.1.6.1), men hadde problemer med å bevare alle maksimums- og minimumspunktene. Dette kan tyde på at LTTB er dårlig egnet for grafer med mye støy, hvor det er viktig å beholde flest mulig ekstremalpunkter. I vårt tilfelle gjelder dette grafen av G-krefter som har mye støy i forhold til trykk- og temperaturgrafene. Figur 5.1.6 viser at LTTB ikke fungerte optimalt på grafen for G-krefter.



Figur 5.1.6: Nedsampling av G-krefter med LTTB. Figuren viser rådataen med 39881 datapunkter til venstre og nedsampling til 6000 punkter med LTTB til høyre. Figuren viser at LTTB ikke klarte å beholde nok ekstremalpunkter.

Ettersom LTTB ikke klarte å bevare nok ekstremalpunkter, så vurderte vi et par andre algoritmer for å nedsample G-krefter dataen. En av disse var "Largest Triangle Dynamic" som også var utviklet av Steinarsson i mastergradsavhandlingen hans (Steinarsson, S., 2013), men denne er ganske kompleks og kjøretiden kunne bli økt betydelig hvis vi hadde valgt denne. Av den grunn bestemte vi oss for å teste MinMax som er beskrevet av National Instruments (National Instruments, 2018). MinMax er en algoritme med relativt lite kompleksitet og den har som mål å bevare så mange ekstremalpunkter som mulig. Som figur 4.1.6.3 viser klarer MinMax å beholde de viktigste ekstremalpunktene, og denne ble derfor valgt for å nedsample grafen for G-krefter. Likevel kan det være en idé å implementere "Largest Triangle Dynamic" for å vurdere om denne gir bedre resultater enn MinMax.

5.1.7 Filtrering av graf

Filtrering av graf er sterkt knyttet opp mot forskningsspørsmål 4 som omhandler visualisering på en interaktiv men uforstyrrende måte. Ved hjelp av filtreringen vil visualiseringen av datasettet kunne endre seg i sanntid, noe som er med på å gi brukeren en interaktiv opplevelse. Linjene i grafen vil "glattes" ut, noe som kan bidra til å øke lesbarheten til grafen og fjerne forstyrrelser. Filtreringen er utført ved hjelp av et tredjeparts-bibliotek som deler dataen opp i perioder og justerer datapunkter i forhold til andre nærliggende punkter (se kapittel 4.1.7). Et mulig problem ved å benytte et slikt bibliotek ved databehandling er at det kan redusere kontrollen over hvordan data blir manipulert. En mulighet i fremtiden vil være å finne en algoritme som er enda bedre dokumentert og som har et godt matematisk grunnlag.

5.1.8 Design av systemet

Forskningsspørsmål 1 handler om hvilken teknologi som bør tas i bruk for å få en mer effektiv og brukervennlig løsning enn den løsningen som brukes per i dag. Som nevnt i kapittel 1.1 består dagens løsning av at de ansatte ved SINTEF bruker flere Python-scripts for å visualisere data fra Sensorfisken. Dette betyr at de ikke interagerer med et brukergrensesnitt i dagens løsning i det hele tatt. I prosjektet har vi forsøkt å lage en web-applikasjon med et godt brukergrensesnitt som gir brukeren en god brukeropplevelse. For å oppnå dette har vi tatt i bruk brukergrensesnittbiblioteket React (se kapittel 3.2.2.1) og CSS-biblioteket React-Bootstrap (se kapittel 3.2.2.2). Hvorfor har vi valgt React over andre populære JavaScript-biblioteker som Angular eller Vue?

Angular er et bibliotek med en arkitektur som passer godt til store prosjekter med mulighet for skalerbarhet, men er med mange forskjellige strukturer noe vanskeligere å ta i bruk enn Vue og React (TechMagic, 2020). Vue er et lite og relativt nytt bibliotek som er enkelt å ta i bruk, men på grunn av dets unge alder finnes det mindre dokumentasjon på dette enn de to andre (TechMagic, 2020). React er enkelt å lære, og er på grunn av dets bruk av en virtuell DOM (se kapittel 3.2.2.1) også veldig raskt.

Valget om å bruke React ble tatt med bakgrunn i at vi hadde kjennskap til React fra før, og at det finnes mye dokumentasjon for dette biblioteket. Den negative siden med dette valget er at vi i stedet for å sette oss inn i alle mulighetene biblioteket tilbyr, satte i gang med det vi allerede kunne fra før. Dersom vi hadde valgt Angular eller Vue hadde vi blitt tvunget til å sette oss inn i et helt nytt bibliotek, og kunne da dratt mer nytte av noe av funksjonaliteten som finnes.

Selv om produktet godt kunne blitt utviklet med et annet JavaScript-bibliotek, ble React valgt da vi mente at dette var teknologien som gav oss best grunnlag for å gi et godt svar på forskningsspørsmål 1. Vi vurderte også at en nettside ville være med på å effektivisere prosessen fra da Sensorfisk kjøres hos oppdretter til en rapport blir levert til kunden (se forskningsspørsmål 2) i motsetning til om produktet hadde blitt utviklet som et frittstående program. Ved å ha en web-applikasjon kan ansatte ved SINTEF laste opp gjennomkjøringer på hvilken som helst PC med nettverksforbindelse. Dette gjør at kunden kan få levert en PDF-rapport kort tid etter at gjennomkjøringer er kjørt gjennom den mekaniske avluseren.

5.1.9 Sikkerhet

I dette kapittelet diskuteres sikkerhetstiltakene som har blitt implementert i systemet. Vi vurderer hvor hensiktsmessig tiltakene har vært, og forklarer hvorfor vi valgte gjeldende tiltak fremfor andre alternativer. Kapittelet er sterkt knyttet opp mot forskningsspørsmål 3.

Det viktigste for oss når vi skulle velge algoritme for hashing av passord i systemet var å finne en algoritme som benytter seg av nøkkelstrekking og salting, og som er designet spesielt for passord-hashing. Det finnes en rekke algoritmer som oppfyller disse kravene, hver med sine styrker og svakheter. Valget falt på bcrypt da denne ifølge Defuse Security (2019) regnes som en sikker algoritme, samtidig som den er relativt enkel å implementere og har en innebygget kostfaktor. Den store fordelen med å ha en kostfaktor er at denne enkelt kan justeres i fremtiden, slik at algoritmen fortsatt vil fungere bra selv om den gjennomsnittlige prosessorhastigheten i verden øker. En av ulempene med bcrypt er at den kun er prosessor-krevende. Det finnes også minne-krevende algoritmer, som krever mye minne for gjøre beregninger, og hvor beregningstiden påvirkes negativt hvis det brukes mindre minne (Biryukov, Dinu & Khovratovich, 2016, s. 292). Eksempler på slike er scrypt og Argon2. Vi valgte å ikke bruke noen av disse, da de hadde en del flere parametre som måtte spesifiseres. Flere parametre gjør algoritmene vanskeligere å justere optimalt, spesielt da vi ikke har noen kjennskap til tjenerens maskinvare.

Valget om å bruke tokens til autentisering fremfor sesjoner ble tatt på bakgrunn av at vi ønsket å utvikle en RESTful tjeneste. Siden tokens inneholder all informasjon som trengs for verifisering, slipper vi å lagre disse på tjeneren og oppnår tilstandsløshet. Det blir i tillegg færre kall mot databasen, da det ikke må hentes ut noe data under verifisering. Vi har også valgt å lagre refresh tokens i databasen. Dette bryter strengt talt med prinsippene for en RESTful tjeneste. Valget ble tatt for å ha en måte å fjerne autentisering fra brukere ved behov. Siden refresh token vil være gyldig i en uke etter utsending, så vi på dette som et nødvendig kompromiss for å sikre at uvedkommende skal kunne stenges ute fra systemet relativt raskt ved behov.

Som det ble forklart i kapittel 4.1.9 lagres tokens i cookies for å unngå farene ved XSS, som blant annet localStorage er utsatt for. Videre forklares det at det er satt cookie-flagg for å beskytte mot CSRF-angrep. SameSite er et av disse cookie-flaggene. SameSite cookies støttes av de fleste moderne nettlesere, men i noen eldre nettlesere vil ikke SameSite cookies fungere (Mozilla, 2020). En mulig løsning ville vært å implementere CSRF-tokens i tillegg til

SameSite cookies. På grunn av et begrenset tidsperspektiv fikk vi ikke implementert dette i systemet.

HTTPS har blitt implementert ved hjelp av et selvsignert sertifikat. Grunnen til at vi valgte dette er at webapplikasjonen foreløpig ikke er knyttet opp mot et domenenavn, noe som gjør det vanskelig å få tak i et sertifikat fra en sertifikatsutsteder. Det selvsignerte sertifikatet har fungert godt for å teste webapplikasjonen over en kryptert forbindelse, men før lansering av systemet bør dette byttes ut med et sertifikat fra en sertifikatsutsteder.

Det er viktig å påpeke at sikkerhetstiltakene som diskuteres i dette kapittelet er utarbeidet for akkurat dette systemet, slik som forskningsspørsmål 3 presiserer. Dette betyr at vi ikke nødvendigvis ville gjort de samme valgene ved utvikling av et annet system. For eksempel ville utviklingen av en nettbank ha krevd mye strengere sikkerhetstiltak enn de vi har implementert i vårt system. Implementeringen av sikkerhet er tidkrevende, og derfor må det alltid vurderes hvilke sikkerhetstiltak som er nødvendige for aktuelt system. Vi vil også nevne at brukernes valg vil kunne være med å påvirke sikkerheten. For eksempel vil ikke den implementerte hashing-algoritmen gi økt sikkerhet om brukeren definerer et svakt passord.

5.2 Ingeniørfaglige resultater

I dette kapittelet drøftes de ingeniørfaglige resultatene, noe som innebærer status for de forskjellige prosjektmålene, samt gyldighet og relevans for tester som er utført gjennom prosjektperioden.

5.2.1 Status for effektmål

1. Selv om vi i kapittel 4.2.1 sier at status for effektmål 1 er oppnådd, er dette kun basert på våre egne resultater fra effektivitetstest og brukertester. Gyldigheten og validiteten til disse testene diskuteres nærmere i henholdsvis kapittel 5.2.5 og 5.2.6.
2. Statusen for effektmål 2 vil i stor grad avhenge av om SINTEF ønsker å ta i bruk vårt produkt. Enkelte av SINTEFs kunder har vært en aktiv del av utviklingen av systemet og virker positive til å ta det i bruk. Det er dermed rimelig å anta at ved bruk vil systemet kunne forenkle kunders tilgang til og behandling av data, da dette kan gjøres direkte av kundene selv.
3. Effektmål 3 er et mål som er lite målbart, og som det vil være vanskelig å svare konkret på om er oppnådd eller ikke. Det kan likevel argumenteres for at systemet gir

utvidet funksjonalitet ved bruk av Sensorfisken, noe som kan virke attraktivt for kunder av SINTEF Ocean AS.

4. Som nevnt i kapittel 4.2.1 vil effektmål 4 avhenge av de andre effektmålene, og da spesielt 2 og 3. SINTEF er avhengige av at kunder vil ta i bruk produktet for å kunne samle inn nok data til å forske på denne. Dersom effektmålet blir nådd kan dette potensielt ha mye å si for oppdrettsnæringen, da det legger et grunnlag for kartlegging av fiskens velferd i rørtransporten.

5.2.2 Status for resultatmål

1. På oppstartsmøtet (se vedlegg E, kapittel 2.1) ble det bestemt at det web-baserte grensesnittet i første omgang skulle utvikles for intern bruk i bedriften. Selv om vi underveis i prosjektperioden fikk tid til å implementere kundebrukere i det leveringsklare produktet, vil vi anbefale SINTEF å teste systemet innad i bedriften før deres kunder gis tilgang.
2. Med utgangspunkt i de brukerfunksjonene som ble nevnt i starten av prosjektperioden har vi oppnådd dette målet, men oppgavestiller har naturlig nok kommet med flere ønsker underveis i prosessen. Vi har forsøkt å oppfylle så mange av disse ønskene som mulig, men måtte mot slutten bare sette en strek og si at tiden ikke strakk til. Dette er selvfølgelig beklagelig, men ønskene vil nevnes i kapittel 6.2. Det vil også leveres en rapport til SINTEF med forslag til ny funksjonalitet som kan implementeres i systemet ved et senere tidspunkt.
3. Valget om å utvikle en relasjonsdatabase ble tatt på bakgrunn av at denne er laget for å håndtere store datamengder, og kan søke gjennom og hente ut data svært raskt (se kapittel 2.3). MySQL ble valgt siden det har en syntaks vi kjenner godt fra før, samtidig som det støttes av ORMet Sequelize.

5.2.3 Status for læringsmål

1. Selv om prosjektperioden har bestått av mye dokumentasjon og i perioder har blitt avbrutt av aktivitet i annet fag, føler vi at vi har fått god lærdom innen utvikling av større web-prosjekter.
2. Gjennom hele prosjektperioden har vi brukt Scrum som utviklingsmetode, og har også praktisert alle aktiviteter som fører med dette. Selv med varierende lengde på sprintene og rullerende romplassering har dette gått veldig bra, og vi har fått god

erfaring med bruk av Scrum. For mer utfyllende detaljer om utviklingsprosessen, se kapittel 5.3.1.

3. Vi har hatt jevnlige møter og regelmessig kontakt via e-post med oppgavestillere fra SINTEF gjennom hele prosjektperioden. Mot slutten av prosjektet har vi også blitt mer inkludert i kontakten mellom SINTEF og enkelte av deres kunder som er involvert i Sensorfisk-prosjektet. På denne måten har vi virkelig fått oppleve hvordan man som utvikler skal forholde seg til en kunde som kommer med krav, ønsker og forventninger til produktet.

5.2.4 Status på systemet ved leveringstidspunkt

Selv om systemet leveres produksjonsklart, anbefales SINTEF å teste det ut internt i bedriften før det rulles ut til eventuelle kunder. Hovedårsaken til dette er å sikre at systemet oppfyller de krav og forventninger som en sluttbruker av produktet vil ha. Det vil også være enkelte nødvendigheter som må på plass før produktet kan sies å være klart for å settes ut i produksjon. En av disse vil være å endre fra et selvsignert SSL-sertifikat i HTTPS-tilkoblingen, til et sertifikat som er signert av en autorisert sertifikatutsteder.

5.2.5 Resultater fra brukertester

Brukertestene avdekket ulike problemområder for systemet som påvirket brukskvaliteten og enkelheten i navigeringen (se kapittel 4.2.5). Dette var blant annet knapper som ikke var intuitivt plassert, og enkelte steder der hensikten med knappen ikke var tydeliggjort. I tillegg var det enkelte deler av funksjonaliteten som deltakerne hadde problemer med å finne. Etter å ha implementert de foreslåtte løsningene fra brukertestene vil de avdekkede problemområdene være rettet opp i. Dette gir utslag i at systemet blir enklere og mer effekt i bruk som øker brukervennligheten og brukskvaliteten. En slik påvirkning vil igjen virke positivt inn på effektmål 1 og 2, som omfatter enkelhet og effektivitet ved bruk av systemet, da økt brukskvalitet som nevnt i kapittel 2.1.3 vil kunne bidra til enklere og mer effektiv bruk av systemet for SINTEF-ansatte og deres kunder.

Brukertestene ble gjennomført på fire SINTEF-ansatte samt to av deres kunder (se kapittel 4.2.5). Deltakerne vil alle være reelle sluttbrukere av systemet, noe Rubin & Chisnell (2008, s. 21) anbefaler for å øke nøyaktigheten og gyldigheten av resultatene produsert. Det kan være verdt å merke seg er at det var et relativt lite utvalg deltakere i brukertestene, noe som påvirker reliabiliteten til testene, nettopp fordi individuelle variasjoner vil gi store utslag i

resultatene. Brukertestene ble gjennomført parallelt med systemutviklingen i løpet av en sprint og det var derfor hensiktsmessig å redusere deltakergruppen slik at de enkelte testene kunne bli utført på en korrekt måte. I tillegg argumenteres det for at en gruppe på fire til fem personer som representerer én brukergruppe vil avdekke 80% av manglene på brukervennlighet, og disse vil gjerne være de største problemområdene (Rubin & Chisnell, 2008, s. 72). Hovedmålet med brukertestene var å forbedre brukervennligheten ved å avdekke kritiske og umiddelbare problemområder, og det ble derfor vurdert dithen at en deltakergruppe på seks personer var en fornuftig avveining mellom tidsbruk og ønsket resultatoppnåelse.

5.2.6 Sammenligning av effektivitet

Testene som ble brukt til å sammenligne effektivitet mellom tidligere og ny løsning viser at den nye løsningen er adskillig raskere enn den gamle, både ved behandling av binærfiler med variabel størrelse og ved behandling av flere binærfiler.

Som nevnt i kapittel 4.2.6 ble den gamle løsningen testet på den individuelle datamaskinen til hvert teammedlem. Testing av ny løsning ble utført i sin enkleste form hvor én i teamet brukte port forwarding til å kjøre systemet på sin lokale PC, slik at de andre teammedlemmene dermed kunne koble seg til tjeneren. Faktorer som maskinvare på tjenermaskin, internetthastighet og individuell tidsbruk av sluttbrukere vil kunne spille inn på faktisk effektivitet. Et annet vesentlig poeng er at utviklerteamet har grundig innsikt i hvordan både tidligere og ny løsning fungerer, slik at tidsbruken i effektivitetstestene vil representere tilnærmet optimal bruk.

Denne måten å kjøre testene på vil ikke gi et fasitsvar på forskjellen i effektivitet mellom ny og tidligere løsning, men vil være gyldig nok til å gi en indikasjon på om den nye løsningen er mer effektiv eller ikke. Testene tyder på at systemets evne til å lese inn og behandle binærfiler raskere enn den gamle løsningen vil bidra til å effektivisere prosessen nevnt i forskningsspørsmål 2.

5.3 Administrative resultater

I dette kapittelet drøftes de administrative resultatene. Dette innebærer fremdriften og utviklingsprosessen som i kapittel 4.3, men inneholder også et kapittel hvor gruppearbeidet drøftes.

5.3.1 Utviklingsprosess

Som nevnt i kapittel 4.3.1 har vi brukt Scrum som utviklingsmetodikk i prosjektet. Denne ble valgt da vi forventet hyppige endringer av krav underveis i prosessen. Scrum som utviklingsmetodikk ga oss muligheten til å sette opp konkrete planer for de arbeidsperiodene vi hadde i prosjektet. Bruken av Scrum ga også mulighet til å forbedre arbeidet underveis, da vi under retrospektiv hadde mulighet til å ta tak i ting som hadde gått dårlig, og utbedre disse.

En av ulempene ved å bruke Scrum i prosjektet var at vi ikke hadde en fast lokasjon hvor vi kunne jobbe. Vi måtte da ha sprintbacklog digitalt i stedet for å ha en fysisk tavle med klistrelapper. Den digitale sprintbackloggen ble likevel nyttig, da vi halvveis ut i prosjektet ble nødt til å jobbe hjemmefra grunnet COVID-19. Det gikk også bort en del tid på å lage de nødvendige dokumentene som kreves for en fullverdig Scrum-prosess. Dette er tid vi føler at kunne blitt brukt mer fornuftig, da det ble mye dokumentasjon totalt sett i prosjektet.

Alt i alt føler vi at valget om å bruke Scrum i utviklingen av prosjektet stort sett har hjulpet oss med å komme i mål, men samtidig tatt opp noe tid, slik at vi ikke har hatt mulighet til å utvikle alle funksjonalitetene som vi ønsket i systemet.

5.3.2 Fremdrift og timeregnskap

Vi ser av tabell 4.3.2.1 at vi har overskredet antall timer på hovedrapporten, men har igjen en del timer på backend-utvikling. Fra fremdriftsplanen som ble satt opp i starten av prosjektet (se vedlegg E, kapittel 1.1) ble det gjort flere revideringer underveis i prosjektperioden.
Av tabell 4.3.2.2 vises det at vi i snitt ligger under 501,5 timer per person på prosjektet. Med bedre planlegging i forkant av prosjektet, hadde vi muligens kommet nærmere 2000 timer totalt. Dette var noe vi syntes var vanskelig, da det dukket opp flere aktiviteter i andre fag som tok tid fra jobbingen med bacheloroppgaven.

5.3.3 Gruppearbeid

Gruppearbeidet har gjennom hele prosjektperioden gått veldig bra. Samtlige gruppemedlemmer har lagt ned en god innsats, og det har ikke vært større uenigheter i gruppen. Ved mindre uenigheter angående valg i prosjektet har vi tatt oss tid til å drøfte disse på en saklig måte, og dermed kommet frem til en felles beslutning. Etter utbruddet av COVID-19 måtte vi endre hvordan vi jobbet, da alle måtte jobbe hjemmefra. Vi måtte dermed ta i bruk verktøy som tillot oss å kommunisere over nett, samt å ha møter med veileder og oppgavestillere. På tross av disse utfordringene har ikke dette hatt noen større konsekvenser for avviklingen av prosjektet.

6 Konklusjon og videre arbeid

I følgende kapittel vil forskningsspørsmålene besvares i lys av resultatene produsert i prosjektet og deres tilhørende diskusjon. I tillegg vil det diskuteres aktuelle utgangspunkt for videre arbeid.

6.1 Konklusjon

Gjennom utarbeidelsen av denne avhandlingen har det vært arbeidet med utvikling av et nettbasert brukergrensesnitt og funksjonalitet som kan benyttes i forbindelse med bruk av SINTEFs Sensorfisk. Basert på vår forskning ble det funnet ut at moderne frontend-rammeverk og bibliotek som for eksempel React og React-Bootstrap var hensiktsmessige for å sikre et brukervennlig system. Dette ga et utgangspunkt for oppbygging av blant annet navigasjon og komponenter som sikret en konsekvent utforming for sluttbrukeren. Det finnes sannsynligvis andre rammeverk eller bibliotek som tilbyr den samme funksjonaliteten, og da vil valget gjerne avhenge av personlige preferanser og forkunnskaper. Det er likevel verdt å merke seg at teknologien som ble tatt i bruk bidro til å sikre et brukervennlig system på en tilfredsstillende måte. Ved hjelp av effektivitetstester ble det avdekket en tydelig reduksjon i tidsbruk med det nye systemet, spesielt ved større filer. Dette kan tyde på en effektivisering av prosessen, men det vil likevel være vanskelig å konkludere med at det nye systemet er mer effektivt. Dette vil kreve videre testing av reell bruk over en lengre tidsperiode for å kunne stadfestes.

Vi kan også konkludere med at hashing av passord med en sikker passordshashings-algoritme som bcrypt før lagring, sammen med bruk av access token og refresh token lagret i cookies og kryptert kommunikasjon over HTTPS er gode og hensiktsmessige sikkerhetstiltak for systemet. Gjennom arbeidet med visualisering ble det funnet ut at graf-biblioteket Recharts var et hensiktsmessig verktøy for å sikre en interaktiv men uforstyrrende visualisering av dataen. Det oppsto likevel noen problemer med interaktiviteten ved store datamengder, noe som ble løst ved nedsampling. Dette kunne også vært løst ved å benytte graf-bibliotek som håndterer dette bedre, men av de som ble funnet krevde alle en betalingsløsning for å bli tatt i bruk kommersielt.

I løpet av prosjektet har det blitt gjennomgått ulike teknologier og algoritmer tilknyttet webutvikling og databehandling/visualisering. Dette er informasjon som kan være nyttig for andre med lignende prosjekter eller ansatte ved SINTEF Ocean AS vedrørende bruk og videreutvikling av systemet.

6.2 Videre arbeid

Ett av punktene i oppgaven som ble nedprioritert grunnet tidsrammen, var å undersøke mulighetene for bruk av maskinlæring på den innsamlede dataen. Dette kan være nyttig å utforske ved videre arbeid. Maskinlæring kan potensielt brukes for å automatisk gjenkjenne om en gjennomkjøring er gyldig eller ikke, og om den inneholder "død" data. Det kan også brukes for å finne start- og sluttpunktet for gjennomkjøringen, og for å gjenkjenne både støt og passering av magnetbånd.

Et annet aspekt som kan være interessant å videreutvikle er posisjonsestimeringen. Dagens løsning består av at posisjonen til Sensorfisken estimeres ved at det plasseres ut magnetbånd på forskjellige steder i rørene til behandlingsenheten, og at magnetometeret til Sensorfisken gjør utslag når den passerer disse magnetbåndene. Denne løsningen kan være noe upresis, samt at den krever litt jobb i forkant med tanke på å plassere ut magnetbånd. En forbedret versjon kan innebære implementasjon av Kalmanfilter for å estimere posisjonen til Sensorfisken inne i behandlingsenheten. Kalmanfilter er en rekursiv algoritme som kan brukes for å estimere tidligere, nåværende og fremtidig posisjon (Welch & Bishop, 1995, s. 1).

Fra brukertestene kom det et ønske om å ha mulighet til å tidsforskyve grafer, slik at man kan sammenligne forskjellige interessepunkter. Dersom man kjører Sensorfisken gjennom en brønnbåt, ønsker man gjerne å sammenligne trykkurvene både når fisken suges inn og når den suges ut av båten. Sensorfisken kan da bli liggende ulik lengde i brønnen av brønnbåten, slik at tidspunktet den suges ut av brønnbåten på kan variere fra gjennomkjøring til gjennomkjøring. Dersom denne funksjonaliteten implementeres i systemet vil det bidra til at SINTEFs kunder kan optimalisere innstillinger på forskjellige stadier av prosessen, og dermed bedre fiskens velferd.

Referanser

- Aggarwal, S. (2018). Modern Web-Development using ReactJS. *International Journal of Recent Research Aspects ISSN*, 5(1), 133-137. Hentet 22. april 2020 fra http://ijrra.net/Vol5issue1/IJRRA-05-01-27.pdf
- Appel, A., W. (1989). A Runtime System. *Journal of Lisp and Symbolic Computation, 3, 343-380.* Hentet 23. april 2020 fra https://web.archive.org/web/20131230235941/https://users-cs.au.dk/hosc/local/LaSC-3-4-pp343-380.pdf
- ArduPilot. (2020). Mission Planner Source Code. Hentet 14. februar 2020 fra https://github.com/ArduPilot/MissionPlanner
- Auth0. (u.å.). Get Started with JSON Web Tokens. Hentet 16. april 2020 fra https://auth0.com/learn/json-web-tokens/
- Beck, K., Beedle, M., Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., … Thomas, D. (2001). Manifesto for Agile Software Development. Hentet 26. mars 2020 fra https://agilemanifesto.org/iso/no/manifesto.html
- Biryukov, A., Dinu, D., & Khovratovich, D. (2016). Argon2: new generation of memory-hard functions for password hashing and other applications. *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, 292-302. https://doi.org/10.1109/EuroSP.2016.31
- Carrol, J., M. (2003). *HCI Models, Theories, and Frameworks: Toward a Multidisciplinary Science*. Burlington, MA: Morgan Kaufmann. Hentet 6. mai 2020 fra https://books.google.no/books?hl=no&lr=&id=gGyEOjkdpbYC&oi=fnd&pg=PP1&dq=mental+models+hci&ots=6zvlGopOk2&sig=uJzZkczbH5zHtHm7k4SbfpgQs0w&redir_esc=y#v=onepage&q=mental%20models%20hci&f=false
- Chand, M. (2019, 26. juli). *Most Popular Databases In The World.* Hentet 24. april 2020 fra https://www.c-sharpcorner.com/article/what-is-the-most-popular-database-in-the-world/
- Codd, E., F. (1990). *The Relational Model for Database Management* (2. utgave). Boston, MA: Addison-Wesley Longman Publishing Co. Hentet 1. april 2020 fra https://codeblab.com/wp-content/uploads/2009/12/rmdb-codd.pdf
- Creer, B.Y, Smedal H. A. & Wingrove, R. C. (1960). *Centrifuge study of pilot tolerance to acceleration and the effects of acceleration on pilot performance*

(Technical note D-337). Ames Research Center, California: NASA. Hentet 23. mars 2020 fra: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19980223621.pdf

- Darwen, H. (2009). *An Introduction to Relational Database Theory.* Frederiksberg: Ventus Publishing APS. Hentet 26. mars 2020 fra https://dvikan.no/ntnu-studentserver/kompendier/an-introduction-to-relational-database-theory.pdf
- Dayley, B. (2014). *Node.js, MongoDB, and AngularJS Web Development.* Boston, MA: Addison-Wesley Longman Publishing Co. Hentet 1. april 2020 fra https://books.google.no/books?hl=no&lr=&id=8kTCAwAAQBAJ&oi=fnd&pg=PR6&dq=node.js&ots=Hgami0TAbB&sig=lqeJNR7JW320dzIFx67s2onEYsI&redir_esc=y#v=onepage&q=node.js&f=false
- Deemer, P., Benefield, G., Larman, C. & Vodde, B. (2012). *The Scrum Primer* (2. utgave). Hentet 26. mars 2020 fra https://www.infoq.com/minibooks/Scrum_Primer/
- Defuse Security. (2019, 5. juni). Salted Password Hashing - Doing it Right. Hentet 12. mai 2020 fra https://crackstation.net/hashing-security.htm
- Digitaliseringsdirektoratet. (u.å.). *Kva er universell utforming?.* Hentet 11.03.2020 fra https://uu.difi.no/kva-er-universell-utforming.
- Diskriminerings- og tilgjengelighetsloven. (2008). Lov om forbud mot diskriminering på grunn av nedsatt funksjonsevne (LOV-2008-06-20-42). Hentet 11.03.2020 fra https://lovdata.no/dokument/LTI/lov/2008-06-20-42.
- Dumas, J. S. & Redish, J. C. (1999). *A Practical Guide to Usability Testing* (2. utgave). Portland: Intellect Books.
- Facebook (2020), *Jest.* Hentet den 17. mai 2020 fra https://jestjs.io/en/.
- Fedosejev, A. (2015). *React.js Essentials.* Birmingham: Packt Publishing.
- Fielding, R. (2000). *Architectural Styles and the Design of Network-based Software Architectures.* (Doktoravhandling, University of California). Hentet 24. april 2020 fra https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf
- Fiskeri- og havbruksnæringens forskningsfond. (u.å.). *Standardisert metodikk for kvalifisering av mekaniske avlusingssystemer (KVALISYS).* Hentet 14. januar 2020 fra https://www.fhf.no/prosjekter/prosjektbasen/901397/
- Flanagan, D. (2006). *JavaScript: The Definitive Guide, Fifth Edition.* Sebastopol, CA: O'Reilly Media, Inc.
- GitHub, Inc. (2020). *About*. Hentet 21. april 2020 fra https://github.com/about

- GitHub, Inc. (2020). *About continuous integration.* Henter 17. mai 2020 fra https://help.github.com/en/actions/building-and-testing-code-with-continuous-integration/about-continuous-integration.
- Greer, D. & Hamon, Y. (2011). Agile Software Development. *Software: Practice and Experience*, *41*(9), 943-944. https://doi.org/10.1002/spe.1100
- Grünwaldt, J. M. (2019). *A Comparison of Modern Backend Frameworks Protections against Common Web Vulnerabilities* (Tufts University, Medford MA). Hentet 21. april 2020 fra http://www.cs.tufts.edu/comp/116/archive/fall2019/jgrunwaldt.pdf
- International Organization for Standardization. (2018). *Ergonomics of human-system interaction - part 11: Usability: Definitions and concepts.* (ISO standard 9241-11). Hentet 25. mars 2020 fra https://www.iso.org/obp/ui/#iso:std:iso:9241:-11:ed-2:v1:en
- Krutz, R. L. & Nahari, H. (2011). *Web Commerce Security: Design and Development*. Indianapolis: Wiley.
- Lie, H., W. & Bos, B. (2005). *Cascading Style Sheets: Designing for the Web, Portable Documents* (3. utgave). Boston, MA: Addison-Wesley Longman Publishing Co.
- Menezes, A. J., Katz, J., Van Oorschot, P. C., & Vanstone, S. A. (1996). *Handbook of applied cryptography*. CRC press.
- Merriam-Webster. (u.å.). Database. I *Merriam-Webster.com dictionary*. Hentet 26.03.2020 fra https://www.merriam-webster.com/dictionary/database
- Moggridge, B. (2007). *Designing Interactions*. Cambridge: MIT Press.
- Mozilla. (2020, 26. mars). SameSite cookies. Hentet 15. mai 2020 fra https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie/SameSite
- National Instruments. (2018, 26.november). *Decimation Algorithm Used to Display Data on a Graph in LabVIEW*. Hentet den 6. mai 2020 fra https://knowledge.ni.com/KnowledgeArticleDetails?id=kA00Z0000019YLKSA2&l=en-NO
- Nielsen, J. (1993). *Usability Engineering.* Burlington, MA: Morgan Kaufmann
- Node.js Foundation. (2017). Express 4.17.1 - Fast, unopinionated, minimalist web framework for Node.js. Hentet 8. mai 2020 fra https://expressjs.com/.
- Recharts Group. (2020, 1. april). *Recharts*. Hentet 22. april 2020 fra https://github.com/recharts/recharts

- Rousseeuw, P. J. & Croux, C. (1993) *Alternatives to the Median Absolute Deviation*. Journal of the American Statistical Association. Hentet 14. april 2020 fra https://doi.org/10.1080/01621459.1993.10476408
- Rubin, J. & Chisnell D. (2008). *Handbook of Usability Testing: How to Plan, Design and Conduct Effective Tests* (2. utgave). Indianapolis: Wiley.
- Saleh, H. (2013). *JavaScript Unit Testing : Your Comprehensive and Practical Guide to Efficiently Performing and Automating JavaScript Unit Testing*. Packt Publishing.
- Schwaber, K. & Sutherland, J. (2017). *The Scrum Guide*. Hentet 2. april 2020 fra https://www.scrumguides.org/index.html
- Silver, K. (2007). *What puts the Design in Interaction Design.* Hentet 25.03.2020 fra https://www.uxmatters.com/mt/archives/2007/07/what-puts-the-design-in-interaction-design.php
- Statistisk sentralbyrå. (2020, 15. januar). Utenrikshandel med varer. Hentet 21. april 2020 fra https://www.ssb.no/utenriksokonomi/statistikker/muh/aar
- Steinarsson, S. (2013). *Downsampling time series for visual representation* (Mastergradsavhandling, Universitetet i Island). Hentet 22. april 2020 fra https://skemman.is/bitstream/1946/15343/3/SS_MSthesis.pdf
- Stone, D., Jarrett, C., Woodroffe, M. & Minocha, S. (2005). *User Interface Design and Evaluation.* Burlington, MA: Morgan Kaufmann.
- TechMagic. (2020). *React vs Angular vs Vue.js - What to choose in 2020?*. Hentet 06. mai 2020 fra https://medium.com/@TechMagic/reactjs-vs-angular5-vs-vue-js-what-to-choose-in-2018-b91e028fa91d
- Thomas, S. (2000). *SSL & TLS essentials: Securing the Web*. New York: Wiley.
- Welch, G. & Bishop, G. (1995). *An introduction to the Kalman filter*. Hentet 13. mai 2020 fra http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.336.5576&rep=rep1&type=pdf
- Zhu, H., Ibaj, E., Umamil, A., Vallelonga, J., Chen, S., Gessner, G. (2019, 18. september). *What is Babel?*. Hentet 23. april 2020 fra https://github.com/babel/website/blob/master/docs/index.md

Vedlegg

A Ordliste

Ord / begreper / akronymer / forkortelser	**Forklaring**
Akselerometer	Måler akselerasjonen og tyngdekraften det gjennomgår. Måleenhet: m/s².
Algoritme	En presis beskrivelse av et sett med operasjoner som brukes for å løse en eller flere oppgaver.
Angular	Et åpent kildekode JavaScript-rammeverk for utvikling av brukergrensesnitt laget av Google.
API	Application Programming Interface eller programmeringsgrensesnitt er et sett med regler som spesifiserer hvordan forskjellig programvare kommuniserer med hverandre.
Applet	Et lite Java-program som kan implementeres i en nettside.
Binærfil, BIN	En fil som er lagret på et binært format, og ikke er leselig for mennesker. Også omtalt som rådatafil.
Blowfish chiffer	Et symmetrisk blokkchiffer som brukes i ulike krypteringsalgoritmer.
Cache	Cache er en komponent som brukes for midlertidig lagring av data, og brukes i en nettleser for å lagre tidligere svar fra tjeneren.
CI	Continuous Integration (kontinuerlig integrasjon). Brukes ved versjonskontroll for å kvalitetssikre koden ved hjelp av tester.
CSRF	Forkortelse for Cross Site Request Forgery. Et angrep hvor en bruker tvinges til å gjøre uønskede handlinger på en side hvor

	han/hun er autentisert.
CSV	Comma Separated Value, filformat for å lagre tabellarisk data.
Database	En stor samling av informasjon som er lagret slik at det skal være enkelt å hente ut og oppdatere dataene som ligger der.
DBMS	Database Management System, en type programvare brukt for å håndtere informasjon lagret i en gitt database.
DOM	Document Object Model. En nettleser sin interne representasjon av et HTML- eller XML-dokument.
Domeneekspert	En person med mye kunnskap innenfor et gitt fagområde.
Enhetstesting	Enhetstesting er en metode som sjekker om en del av koden gir forventet resultat. En enhetstest skal være automatisert, repeterbar, lett å forstå, lett å kjøre og kjapp (Saleh, H., 2013, s.8).
Entitet	En ting, en hendelse, et konsept eller et faktum.
Flaskehals	En flaskehals oppstår i et system når hastigheten til systemet begrenses av en enkelt komponent.
Github	Et selskap som leverer tjenester for versjonskontroll.
Glidebryter	En bryter/knapp som kan beveges langs en akse for å oppnå gradvis justering, ofte brukt ved justering av volum eller lysstyrke.
Gyroskop	Måler rotasjonen til instrumentet i forhold til et statisk referansepunkt. Måleenhet: rad/s.
Hashing	En algoritme som utføres på et passord for å gjøre det om til en ugjenkjennelig tekststreng. Prosessen skal være umulig å reversere.
HTML	Hyper Text Markup Language. Et markeringsspråk for formatering av nettsider.

HTTP	HyperText Transfer Protocol er en protokoll som legger grunnlaget for kommunikasjon på verdensveven.
I/O	Input/Output blir brukt når et program på datamaskinen må overføre data til eller fra en annen enhet. Dette kan for eksempel være når programmet må skrive til eller lese fra harddisken.
JSON	En enkel standard for datautveksling, hvor dataen er lagret i et selvbeskrivende, tekstbasert format.
JWT	En åpen standard som definerer en kompakt måte å utveksle informasjon som JSON-objekter (Auth0, u.å.).
Kalmanfilter	Algoritme for å fjerne støy fra målinger, i et forsøk på gi et bedre estimat.
Kryptering	Brukes for å gjøre en melding uleselig under kommunikasjon, slik at en utenforstående tredjepart ikke kan se informasjonen som utveksles.
KVALISYS	Et prosjekt gjennomført av SINTEF Ocean AS, finansiert av fiskeri- og havbruksnæringens forskningsfinansiering (FHF).
Lookup table	En oppslagsstruktur hvor hash-verdien og klarteksten for mange passord ligger lagret, som potensielt kan brukes til å knekke passord ekstremt effektivt (Defuse Security, 2019).
Magnetometer	Måler styrken og retningen til magnetfeltet rundt instrumentet. Måleenhet: tesla.
Man in the middle-angrep	Et angrep hvor en ukjent tredjepart kan lytte til og endre kommunikasjonen mellom to parter.
Matlab	Et matematisk programmeringsspråk.
Mellomvare	Programvare som tilbyr tjenester til applikasjoner utover det som er tilbudt fra operativsystemet.

MIME	Multipurpose Internet Mail Extentions er en standard utviklet for mail, for å tolke andre karaktersett enn ASCII. Har også blitt tatt i bruk i HTTP-protokollen.
MissionPlanner	Programvare brukt for å åpne binærfiler fra blant annet Pixhawk (mikrokontrolleren til Sensorfisk).
NED	North East Down er en standardisert måte å beskrive referanserammen til et koordinatsystem.
Open source	Et prosjekt som er "open source" har kildekoden sin offentlig tilgjengelig.
ORM	Object Relational Mapper. Tar i bruk forskjellige programmeringsteknikker for å konvertere data mellom inkompatible systemer.
OWASP	Open Web Application Security Project. Et fellesskap som baserer seg på å produsere artikler innenfor webapplikasjonssikkerhet.
OWASP A1 (injection)	Sikkerhetsrisiko nummer én ved webapplikasjoner i følge OWASP. Består av at en angriper utnytter seg av et kompromittert inntastingsfelt for å sende inn kommandoer til databasen.
OWASP A7 (cross-site scripting)	Sikkerhetsrisiko ved fremvisning av brukerinput. Unngås ved å filtrere brukerinput slik at html-tags eller annen kode blir fjernet.
PDF	Portable Document Format, filformat for elektroniske dokumenter.
PNG	Portable Network Graphics, filformat for å lagre digitale bilder.
Pool	I informatikken er et pool en mengde ressurser som venter på å bli tatt i bruk.
Port forwarding	Brukes for å kjøre en offentlig tjener på et privat nettverk hvor utenforstående vanligvis ikke har tilgang.
Python	Et høynivå, objektorientert programmeringsspråk.

Rainbow table	En oppslagsstruktur som ligner på lookup tables, men som ofrer hastighet for å sørge for at strukturen tar mindre plass, noe som gjør at flere hash-verdier kan lagres (Defuse Security, 2019).
RDF	Resource Description Framework er et rammeverk som har blitt standard for beskrivelse og modellering av informasjon i webapplikasjoner.
React	Et brukergrensesnitt-bibliotek laget for JavaScript som er utviklet av Facebook.
Repository	I et versjonskontrollsystem er et repository stedet hvor alle filene til prosjektet lagres.
RPY	Roll, Pitch og Yaw beskriver oftest de tre rotasjonsaksene til et fly eller en flyvende gjenstand. I vårt tilfelle så beskriver det rotasjonsaksene til Sensorfisken.
Salt	En tilfeldig generert tekststreng som legges til passordet under hashing for å sørge for at det genereres en unik hash-verdi.
Script	En eksekverbar fil som inneholder en rekke kommandoer.
Sertifikatsutsteder	Et firma som distribuerer digitale sertifikater.
SQL	Structured Query Language. Et spørrespråk som brukes for å kjøre operasjoner mot en database.
SQL Injection	Et type angrep på en SQL database, der en angriper prøver å hente ut eller manipulere data som de ikke skal ha tilgang til.
Storyboard	En form for “tegneserie” brukt for å visualisere en user story.
Tråd	Tråder er de delene av en prosess som eksekverer programkoden, og deler ressursene som er tildelt prosessen.
URI	Uniform Resource Identifier er en tekststreng som identifiserer en ressurs. *http://nettbank.no/bruker/1/konto/1* er et eksempel på en

	URI som entydig identifiserer en gitt konto på en nettbank.
USB	Universal Serial Bus. En databuss som brukes for å koble enheter til en datamaskin.
Vue	Et åpent kildekode JavaScript-rammeverk for utvikling av brukergrensesnitt.
Wireframe	En tegning/modell av en nettside, med den hensikt å lage en skisse av hvordan nettsiden visuelt skal se ut.
XSS	Forkortelse for Cross Site Scripting. Et angrep hvor skadelige scripts sendes inn til en dynamisk nettside. Disse scriptene vil kjøre hvis de ukritisk skrives ut under innlasting av nettsiden.

Utvikling av brukergrensesnitt og brukerfunksjoner for Sensorfisk

Visjonsdokument

Vedlegg B

Forfattere:

Herman Ryen Martinsen

Sander Nicolausson

Trond Jacob Rondestvedt

Jørgen Aasvestad

Versjon 2.1

19.02.2020

Revisjonshistorie

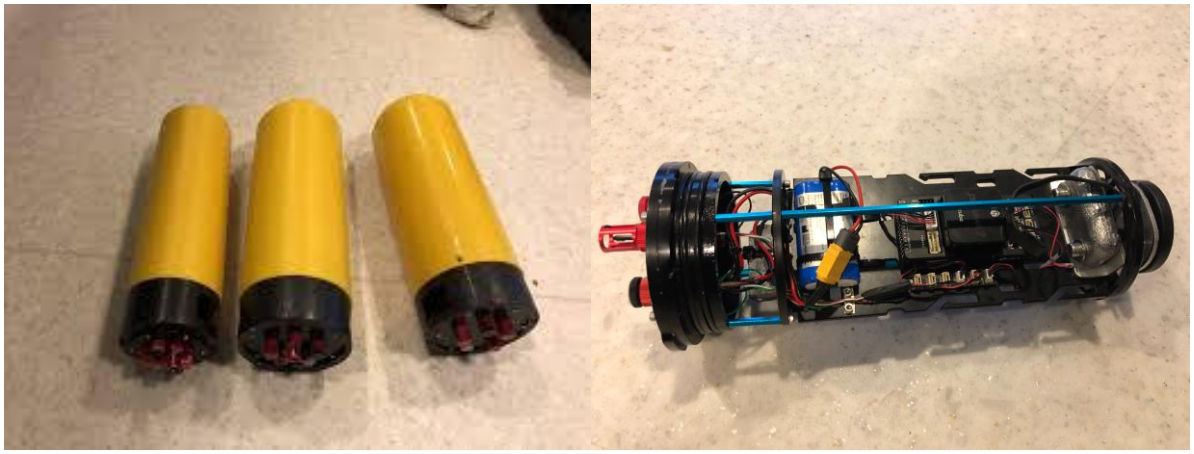
Dato	Versjon	Beskrivelse	Forfatter
14.01.20	1.0	1. utkast visjonsdokument	Herman Ryen Martinsen Sander Nicolausson Trond Jacob Rondestvedt Jørgen Aasvestad
17.01.20	1.1	Finpuss av førsteutkast	Herman Ryen Martinsen Sander Nicolausson Trond Jacob Rondestvedt Jørgen Aasvestad
31.01.20	2.0	2. utkast visjonsdokument	Sander Nicolausson Trond Jacob Rondestvedt Jørgen Aasvestad
19.02.20	2.1	Ferdigstilling visjonsdokument	Herman Ryen Martinsen Trond Jacob Rondestvedt Jørgen Aasvestad Sander Nicolausson

Innholdsfortegnelse

1 Innledning

Hensikten med visjonsdokumentet er å synliggjøre hvem som er interessenter i prosjektet, hvem som er brukerne av produktet og hvilke behov disse har. Det skal også vise produktets funksjonelle- og ikke-funksjonelle avhengigheter. Visjonsdokumentet skal sørge for en felles forståelse mellom prosjektgruppe og oppdragsgiver av hvilke mål og rammer man ønsker for prosjektet (Norges teknisk-naturvitenskapelige universitet, u.å.).

SINTEF Ocean AS har i sitt prosjekt KVALISYS forsøkt å lage en standardisert metodikk for kvalifisering av mekaniske avlusingssystemer (Fiskeri- og havbruksnæringens forskningsfond, 2020). Dette er gjort ved å utvikle et produkt som kalles "Sensorfisk". Sensorfisken består av en mikrokontroller og flere sensorer som registrerer data.



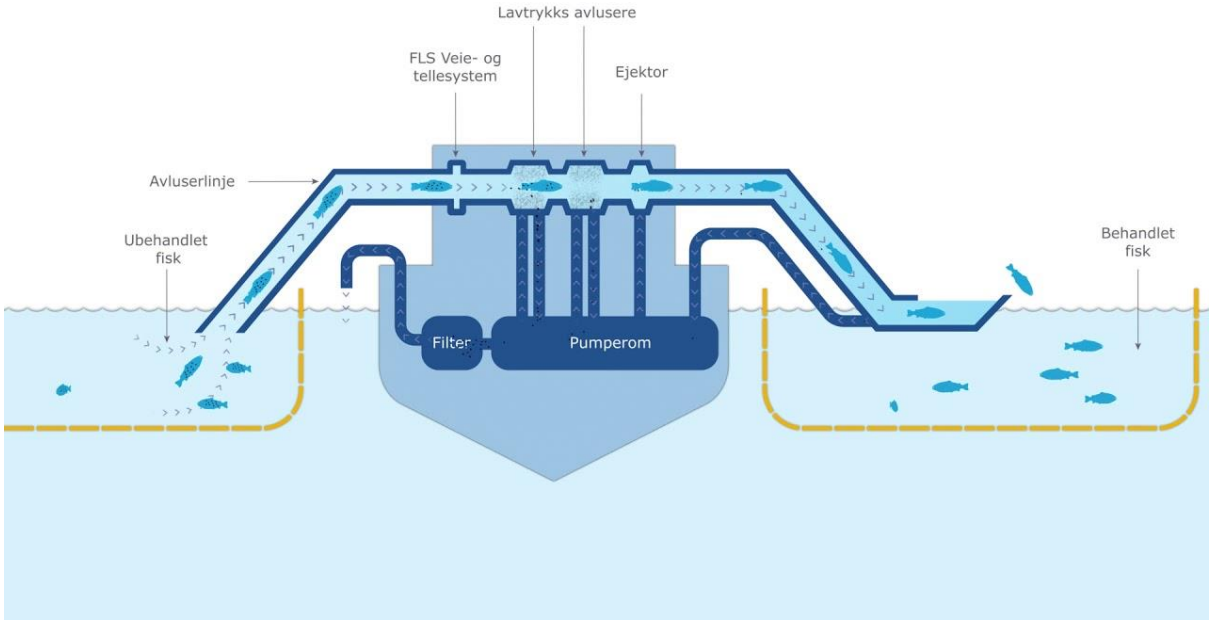
Figur 1.1: SINTEF Ocean AS' produkt Sensorfisk (SINTEF, 2017).

De forskjellige instrumentene og deres tilhørende samplingsfrekvenser er (Caharija et al., 2019, s. 9):

Akselerometer	1000 Hz
Gyroskop	1000 Hz
Magnetometer	50 Hz
Kombinert trykk og temperatur	50 Hz
Temperatur (Fast-response)	20 Hz

Formålet med Sensorfisken er å etterligne en fisk som blir ført gjennom et mekanisk avlusersystem, og dermed påvise hvilken behandling den får på forskjellige steder i systemet. For eksempel kan man se på kraftige endringer i akselerasjon for å finne antall støt mot rørveggen, noe som kan være en stressfaktor for fisken. For å bestemme produktets posisjon i avluseren festes magnetbånd på bestemte punkter i rørene, slik at magnetometeret til

Sensorfisken gir utslag (Caharija et al., 2019, s.8). Sensorfisken blir sendt gjennom avluseren på samme måte som en normal fisk, før den blir fanget opp på slutten slik at man kan hente ut de aktuelle dataene ved å koble Sensorfisken til en PC.



Figur 1.2: Virkemåte for en mekanisk avluser fra Flatsetsund Engineering (Flatsetsund Engineering, u.å.). Ubehandlet fisk blir dratt opp til avluserlinja, der de blir veid og behandlet gjennom en lavtrykks-avluser. Til slutt blir de pumpet ut ved hjelp av ejektoren.

Hensikten med vårt prosjekt er å utvikle et web-basert grafisk brukergrensesnitt som leser inn, analyserer og visualiserer data, og gir brukeren muligheter til å eksportere disse dataene som filer. Etter at Sensorfisken har blitt sendt gjennom avluseren kan man koble den til en PC og deretter laste opp dataen til systemet. Det skal også være mulighet for at historiske data skal være lagret, slik at man kan hente opp disse og sammenligne analyserte verdier fra flere kjøringer av Sensorfisken.

2 Sammendrag problem og produkt

Dette kapittelet gir en overordnet oversikt over hva problemet er, hvilken løsning som er tilgjengelig i dag og hvordan vårt produkt vil løse dette problemet.

2.1 Problemsammendrag

Problemsammendraget beskriver hvordan systemet fungerer i dag, og hvilke problemer som skal løses. Samtidig beskrives det kort hvilke fordeler en vellykket løsning vil ha.

Tabell 2.1: Tabell som kort oppsummerer dagens problem fra oppgavestiller, samt forslag til hva en vellykket løsning kan være.

Problemet med	dagens system er at håndteringen og filtreringen av data gjøres manuelt ved hjelp av Python-scripts
berører	ansatte i SINTEF Ocean AS og kunder som skal behandle data fra Sensorfisken
som resultatet av dette	kan det bli unødvendig tidkrevende å gå fra måling av data til visualiseringen. Det er vanskelig og omfattende å tilpasse forskjellige data i forhold til kundenes ulike ønsker.
En vellykket løsning vil	være å utvikle et brukergrensesnitt der man kan laste opp data fra målingene og automatisk få utlevert nødvendige grafer/visualisering.

2.2 Produktsammendrag

Produktsammendraget har den hensikt å gi en overordnet innsikt i hvilket produkt vi skal lage i tillegg til hvilket system/produkt det skal erstatte.

Tabell 2.2: Tabellen gir et kort innblikk i hvem som er oppdragsgiver, hvilket produkt de ønsker, samt hvilken løsning de har per dags dato.

For	SINTEF Ocean AS
som	trenger et nytt system for behandling av data fra Sensorfisk.
Systemet	er et web-basert brukergrensesnitt
som	leser inn, behandler og visualiserer data fra en Sensorfisk
i motsetning til	Python-scriptene som brukes per i dag
er vårt produkt	mer effektivt, fleksibelt og brukervennlig.

3 Overordnet beskrivelse av interessenter og bruker

Dette kapittelet beskriver hvem som har en interesse av at vårt produkt blir laget, og viser også hvilke behov disse vil ha.

3.1 Oppsummering interessenter

Hensikten med følgende delkapittel er å tydeliggjøre ulike interessenter som vil være knyttet opp mot systemet, samt deres rolle under utviklingen.

Tabell 3.1: Tabellen viser hvem som er interessenter i produktet og hvilken rolle disse vil ha under utviklingen av produktet.

Navn	**Utdypende beskrivelse**	**Rolle under utviklingen**
SINTEF Ocean AS	SINTEF Ocean AS stiller som oppgavestiller og produkteier, og er representert av Sveinung Ohrem og Birger Venås.	Spesifiserer hvilke krav og funksjoner som er ønskelig i sluttproduktet. Følger opp og gir tilbakemeldinger underveis i prosessen.
NTNU	NTNU stiller med Jan Harald Nilsen, som er veileder for bacheloroppgaven.	Ansvarlig for tilbakemeldinger på dokumentasjon og oppfølging på studentenes initiativ.
Utviklerne	Bachelorstudentene som utvikler systemet: Herman Ryen Martinsen, Sander Nicolausson, Trond Jacob Rondestvedt, Jørgen Aasvestad	Ansvarlig for å utvikle produkt og dokumentasjon etter gitte spesifikasjoner. Også ansvarlig for prosjektets framdrift og å levere innen gitte tidsfrister.
Brukere	Ansatte hos SINTEF Ocean AS og firmaets kunder er brukere av systemet.	Påvirker hvilke valg utviklerne tar med tanke på et brukervennlig grafisk grensesnitt.

3.2 Oppsummering brukere

Under følger en oversikt over de aktuelle brukerne av produktet, i tillegg til deres rolle under utviklingen.

Tabell 3.2: Tabellen viser en oversikt over hvem som blir sluttbrukerne av produktet, og hvilken rolle disse har under utviklingen.

Navn	**Utdypende beskrivelse av dagens løsning**	**Rolle under utviklingen**	**Representert av**
Administrator	Innsamlede data er lagret internt i Sensorfiskens mikrokontroller. Administrator leser inn og behandler innsamlede data fra Sensorfisk ved hjelp av Python-scripts. Eksporterer resultater og sender disse tilbake til kunden.	SINTEF Ocean AS er både produkteier og sluttbruker av systemet. Dermed har de definert krav og funksjoner i sluttproduktet, men vil også komme med tilbakemeldinger fra et brukerperspektiv.	Ansatte hos SINTEF Ocean AS.
Kunder	Bestiller en vurdering av sin mekaniske avluser, og får tilsendt en rapport i ettertid.	Påvirker hvilke valg utviklerne tar med tanke på et brukervennlig grafisk grensesnitt.	Kunder av SINTEF Ocean AS som ønsker en vurdering av sin mekaniske avluser.

3.3 Brukermiljøet

Løsningen skal være web-basert og bør dermed fungere på de fleste moderne nettlesere. Den skal være tilgjengelig både på kontoret og ute i felt, gitt at datamaskinen er tilkoblet internett.

3.4 Sammendrag av brukernes behov

Brukerne av systemet vil ha behov for forskjellige funksjonaliteter. Dette kapittelet er et sammendrag av alle behovene brukerne vil ha i systemet, og er utformet i samråd med oppgavestiller. Det beskrives hvordan behovet er løst i dag, og hvordan den foreslåtte løsningen vil være i vårt system.

Tabell 3.3: Tabellen viser en oversikt over brukernes behov, prioriteten til behovet, hva løsningen er i dag og hva som er tiltenkt løsning i vårt produkt.

Behov	**Prioritet**	**Påvirker**	**Dagens løsning**	**Foreslått løsning**
Logge inn i systemet.	Middels.	Innlogging.	Ingen.	Bruker har brukernavn/epost og et passord som brukes til å logge inn.
Velge ulike behandlingsenheter.	Middels.	Dataanalyse.	Ingen.	Brukerne kan velge én eller flere behandlingsenheter for å kun se på relevant data.
Filtrere data (fjerne støy).	Høy.	Systemets effektivitet.	Python-script som filtrerer ved hjelp av matematiske formler.	Videreutvikling av filtreringen, raskere behandlingstid. Mulighet for å justere "hardheten" av filtreringen.
Lagre historiske data.	Høy.	Datalagring.	Lokal lagring av rådata-filer på jobb-PC til ansatte ved SINTEF.	Database som inneholder all ønskelig data. Data aksesserbar via systemet.
Eksportere grafiske filer og datafiler for bruk i andre program.	Høy.	Systemets effektivitet.	Grafplotting ved hjelp av Python-script.	Mulighet for å eksportere de grafiske filene som f.eks. pdf- eller Excel-filer.
Endre passord.	Lav.	Innlogging.	Ingen.	Bruker kan endre passordet via systemet.
Sammenligne data fra ulike gjennomkjøringer.	Høy.	Dataanalyse.	Manuelt med Python-script.	Velge to eller flere gjennomkjøringer for å sammenligne disse, for eksempel ved hjelp av grafer.
Legge inn rådata fra gjennomkjøringer.	Høy.	Datalagring.	Lagret lokalt på jobb-PC til ansatte ved SINTEF.	Grensesnitt for å legge inn rådatafiler, slik at disse blir lagret i databasen.
Slette	Lav.	Datalagring.	Slettes lokalt	Mulighet for å slette

gjennomkjøringer.			på jobb-PC til ansatte ved SINTEF.	gjennomkjøringer, for eksempel ved “død data” eller andre feil.
Registrere kunder.	Middels.	Datalagring.	Ingen.	Kan legge inn kunder i databasen slik at gjennomkjøringer blir koblet til kunden.
Grafisk framstilling av data.	Høy.	Dataanalyse.	Plotting vha. Python-script.	Diverse grafiske framstillinger av ønskelig data, med mulighet for forskjellige typer grafer/diagrammer.
Opprette kundebruker.	Lav.	Kundetilfredshet.	Ingen.	Opprette en bruker i systemet for kunder slik at de selv kan se data angående sine egne gjennomkjøringer.

3.5 Alternativer til vårt produkt

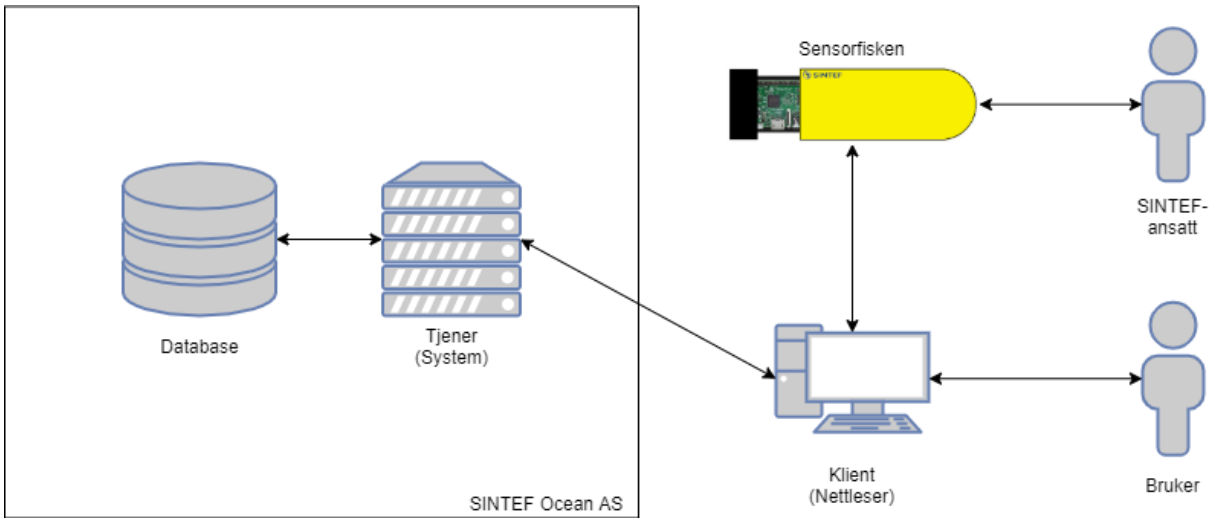
Måten dette gjøres på i dag er at brukerne (ansatte hos SINTEF Ocean AS) bruker programmet MissionPlanner for å omgjøre binærfilene som kommer fra Sensorfisken til Matlab-filer. De henter så ut informasjonen fra Matlab-filene ved hjelp av et Python-script som filtrerer dataene og genererer et plot ut fra dette. Brukerne har da både binærfilene og matlab-filene lagret lokalt på sin egen datamaskin.

4 Produktoversikt

Dette kapittelet viser hvordan produktet er tiltenkt i bruk, og vil gi en ide om hvilke forutsetninger det har for å kunne brukes.

4.1 Produktets rolle i brukermiljøet

Sluttproduktet vil som nevnt tidligere bli en nettside der man kan logge inn for å få visualisert ønsket data. Kommunikasjon med systemet vil foregå via en nettleser, der systemet kommuniserer videre med databasen for å innhente ønskelig data.



Figur 4.1: Enkel skisse av produktets rolle i brukermiljøet. SINTEF-ansatte kobler Sensorfisken til en PC og laster opp dataen ved hjelp av klienten. Klienten sender så dataen til tjeneren, som igjen sender det videre til databasen der det blir lagret. Brukeren kobler seg på systemet via en nettleser, systemet henter så data fra databasen slik at dette kan visualiseres i nettleseren til brukeren.

4.2 Forutsetninger og avhengigheter

For at produktet skal fungere som tenkt, er det visse forutsetninger og avhengigheter som må være oppfylt. Disse kan være gjeldende både under utvikling og etter ferdigstillelse av produktet. Følgende forutsetninger og avhengigheter er spesifisert:

4.2.1 Alle brukere som skal registrere seg i systemet har en e-post.

4.2.2 Vi får tilgang til rådata fra Sensorfisk og tilhørende Python-script.

4.2.3 Sensorfisk-dataene lagres på nåværende format.

4.2.4 SINTEF Ocean AS stiller med tjener og at denne har tilstrekkelig kapasitet.

5 Produktets funksjonelle avhengigheter

I dette kapittelet listes det opp de funksjonaliteter som produktet er avhengig av for at det skal fungere som tiltenkt.

5.1 Bruker skal kunne velge behandlingsenheter.

5.2 Funksjon for å fjerne støy fra målinger. (filtrering av data).

5.3 All ønskelig data skal kunne lagres.

5.4 Bruker skal kunne eksportere grafiske filer og datafiler.

5.5 Passordet skal kunne bli endret av brukeren.

5.6 Bruker skal kunne sammenligne data fra ulike gjennomkjøringer.

5.7 Det skal være mulig å slette gjennomkjøringer.

5.8 Bruker må kunne legge inn rådata fra gjennomkjøringer.

5.9 Det skal være mulig å legge inn nye kunder i systemet.

5.10 Bruker skal få valgt fra diverse visualiseringer av dataen.

5.11 Det skal kunne opprettes brukere koblet til kunder i systemet.

5.12 Funksjon for å la brukere logge inn.

6 Ikke-funksjonelle avhengigheter og krav

Dette kapittelet beskriver hvilke krav som stilles for at produktet skal virke som tiltenkt, men omhandler ikke adferden til programmet.

6.1 En plattformuavhengig nettleser med støtte for en nyere HTML-standard må kunne brukes som klient mot applikasjonen.

6.2 For å kunne legge inn data fra Sensorfisken må man ha en datamaskin med USB-inngang.

6.3 Brukerdata og data fra Sensorfisk skal lagres sikkert i en SQL-database.

6.4 Brukernes passord skal hashes med en sikker algoritme på både klient- og tjenerside, og deretter lagres sikkert i databasen.

6.5 Løsningen skal sjekkes opp mot OWASP A1 (Injection) og A7 (Cross Site Scripting).

6.6 Det skal skrives enhetstester for all kode der dette er relevant, og kjøres kontinuerlig testing ved hjelp av GitHub CI (Continuous Integration).

Referanser

- Caharija, W., Venås, B., Svendsen, E., Schrøder, M. B., Pedersen, M. O., Lie, O. K., … Ohrem, S. J. (2019). *Verktøy for kartlegging av forhold i enheter for føring, håndtering og behandling av laks (Sensorfisk) (SINTEF rapport L4)*. Hentet 14. januar 2020 fra https://www.fhf.no/prosjekter/prosjektbasen/901397/?fileurl=https://fhfno.sharepoint.com/sites/pdb/Publisertedokumenter/307829FHprnr901397_KVALYSIS_faglig_sluttrapport_v3.pdf.PDF&filename=Sluttrapport:%20Verkt%C3%B8y%20for%20kartlegging%20av%20forhold%20i%20enheter%20for%20f%C3%B8ring,%20h%C3%A5ndtering%20og%20behandling%20av%20laks%20(sensorfisk)
- Fiskeri- og havbruksnæringens forskningsfond. (u.å.). *Standardisert metodikk for kvalifisering av mekaniske avlusingssystemer (KVALISYS).* Hentet 14. januar 2020 fra https://www.fhf.no/prosjekter/prosjektbasen/901397/
- Flatsetsund Engineering. (u.å.). *FLS Avluser: For avlusing av laks.* Hentet 18. februar 2020 fra https://www.fls.no/avlusing-av-laks/
- Norges teknisk-naturvitenskapelige universitet. (u.å.). *Visjonsdokumentet*. Hentet 14. januar 2020 fra https://www.ntnu.no/iie/fag/maler-standarder/UP/Visjonsdokumentet_intro.htm
- SINTEF. (2017). *Standardisert metodikk for kvalifisering av mekaniske avlusingssystemer (KVALISYS)*. Hentet 19. februar 2020 fra https://www.sintef.no/prosjekter/standardisert-metodikk-for-kvalifisering-av-mekaniske-avlusingssystemer/
- The OWASP Foundation. (u.å.). *Top 10 Web Application Security Risks.* Hentet 14. januar 2020 fra https://owasp.org/www-project-top-ten/

Utvikling av brukergrensesnitt og brukerfunksjoner for Sensorfisk

Kravdokument

Vedlegg C

Forfattere:

Herman Ryen Martinsen

Sander Nicolausson

Trond Jacob Rondestvedt

Jørgen Aasvestad

Versjon 2.1

23.03.2020

Revisjonshistorie

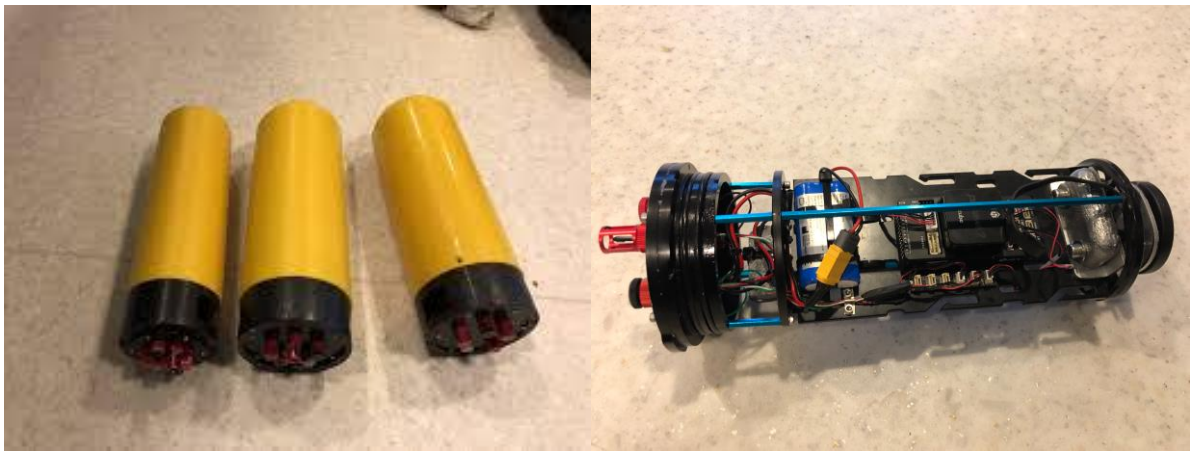
Dato	Versjon	Beskrivelse	Forfatter
19.02.2020	1.0	Førsteutkast	Herman Ryen Martinsen Sander Nicolausson Trond Jacob Rondestvedt Jørgen Aasvestad
26.02.2020	1.1	Finpuss førsteutkast	Sander Nicolausson Trond Jacob Rondestvedt Jørgen Aasvestad
19.03.2020	2.0	Revisjon	Herman Ryen Martinsen Sander Nicolausson Trond Jacob Rondestvedt Jørgen Aasvestad
23.03.2020	2.1	Revisjon	Herman Ryen Martinsen Sander Nicolausson Trond Jacob Rondestvedt Jørgen Aasvestad

Innholdsfortegnelse

1 Introduksjon

Dokumentet er skrevet i forbindelse med bacheloroppgave nr. 061 der SINTEF Ocean AS er oppgavestiller. Kravdokumentet er utarbeidet i den hensikt å gi en oversikt over hvilke krav som stilles til utviklingen av produktet. Dette gjøres ved å beskrive user stories, domenemodell, sekvensdiagrammer og prototyper.

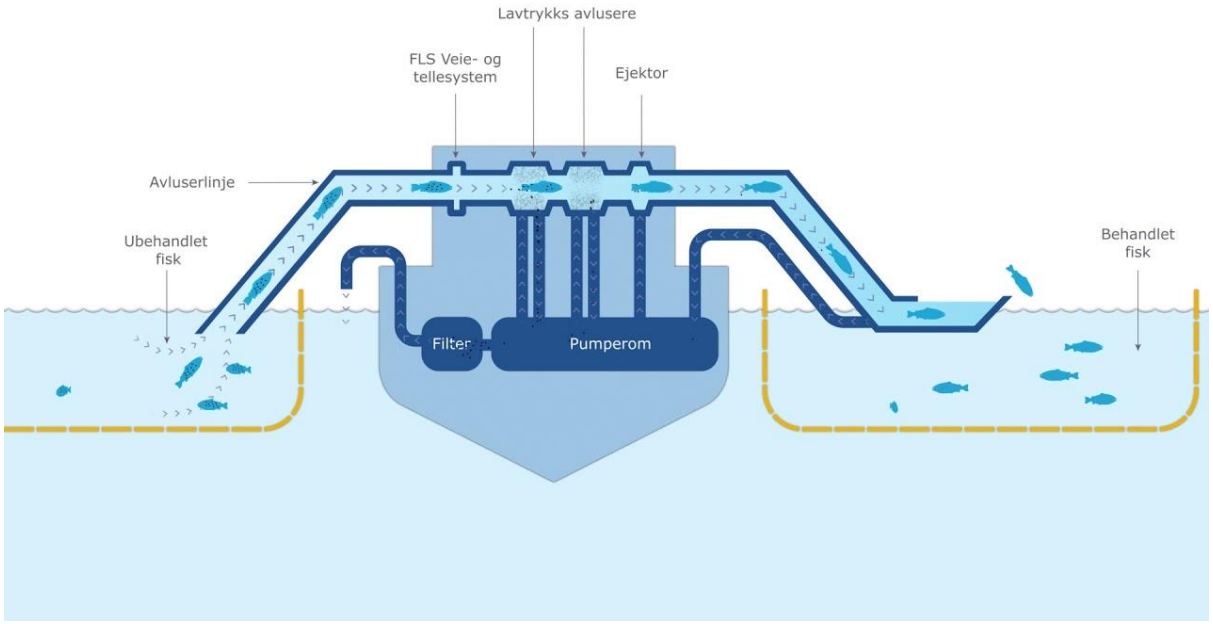
SINTEF Ocean AS har i et tidligere prosjekt utarbeidet et produkt som kalles Sensorfisk. Dette er en sylinderformet gjenstand med sensorer som måler trykk, temperatur, akselerasjon og magnetisme.



Figur 1.1: SINTEF Ocean AS' produkt Sensorfisk (SINTEF, 2017).

Formålet med Sensorfisken er å etterligne en fisk som blir ført gjennom et mekanisk avlusersystem, og dermed påvise hvilken behandling den får på forskjellige steder i systemet.

Man kan eksempelvis se på kraftige endringer i akselerasjon for å finne støt mot rørveggen, noe som kan være en stressfaktor for fisken. For å bestemme produktets posisjon i avluseren festes magnetbånd på bestemte punkter i rørene, slik at magnetometeret til Sensorfisken gir utslag (Caharija et al., 2019, s.8). Sensorfisken blir sendt gjennom avluseren på samme måte som en normal fisk, før den blir fanget opp på slutten slik at man kan hente ut de aktuelle dataene ved å koble Sensorfisken til en PC.



Figur 1.2: Virkemåte for en mekanisk avluser fra Flatsetsund Engineering (Flatsetsund Engineering, u.å.). Ubehandlet fisk blir dratt opp til avluserlinja, der de blir veid og behandlet gjennom en lavtrykks-avluser. Til slutt blir de pumpet ut ved hjelp av ejektoren.

Prosjektet vårt består av å utarbeide et web-basert grafisk brukergrensesnitt for opplasting, analysering og visualisering av dataene fra Sensorfisken. Etter at Sensorfisken har blitt sendt gjennom en mekanisk avluser vil man kunne koble den til en PC, for deretter å gå inn på vår nettside og laste opp binærfilen fra Sensorfisken. Systemet vil da lese ut dataene fra binærfilen og legge disse inn i databasen. Selve binærfilen vil lagres, slik at den består selv om databasetabellene slettes. Etter å ha lagt inn data fra en gjennomkjøring vil man ha mulighet til å se en grafisk fremstilling av denne. Man vil også ha mulighet for å sammenligne en gjennomkjøring med andre, slik at man eksempelvis kan sammenligne data fra to forskjellige avlusere med samme type pumpe.

2 User Stories

En user story brukes i systemutvikling for å forenkle beskrivelsen av en funksjonalitet i systemet. Disse utformes generelt fra produktets funksjonelle avhengigheter, men det er ikke krav om et én-til-én forhold mellom avhengigheter og user stories.

User stories skal dekke all funksjonaliteten til systemet, og vi har valgt å bruke følgende format som vi har hentet fra NTNU (Norges teknisk-naturvitenskaplige universitet, u.å):

Tabell 2: Oppsett for user story

Som	<rolle>
ønsker jeg	<mål>
slik at	<fordel>

Under utarbeidelse av user stories er det ulike roller og begreper nevnt som vil bli definert her. Rollen *administrator* omfatter de SINTEF-ansatte som skal bruke systemet. De vil ha aksess til alle systemets funksjoner og ha tilgang til å laste opp data, opprette behandlingsenheter og nye brukere, både administratorer og kundebrukere. Rollen *kundebruker* vil være knyttet til en bedrift og vil dermed ha begrenset tilgang, og vil for eksempel kun ha tilgang til data fra sine egne gjennomkjøringer. En *behandlingsenhet* er en avluser som brukes under gjennomkjøringene, der disse kan ha forskjellig avluserprinsipp/virkemåte og type pumpe. En *gjennomkjøring* er Sensorfiskens passering gjennom behandlingsenheten og inneholder den dataen som er samlet i ferden gjennom enheten.

2.1 Opprette brukere i systemet

Administratorene for systemet, altså SINTEF-ansatte, skal ha mulighet til å opprette nye brukere. Dette vil være enten nye administratorbrukere eller kundebrukere.

Tabell 2.1: User story for å opprette brukere i systemet.

Som	administrator
ønsker jeg	å opprette brukere i systemet
slik at	jeg kan gi nye brukere tilgang til systemet.

Liste over krav til story:

- Jeg må kunne opprette andre administratorer og kundebrukere.
- Jeg må kunne oppgi e-post, fornavn, etternavn og hvilket firma brukeren er tilknyttet.
- Jeg skal ikke kunne opprette en ny bruker med samme e-post som en eksisterende bruker.
- Jeg ønsker at en e-post skal bli sendt til brukeren med en link til siden hvor de skal opprette passord.

2.2 Innlogging

Denne story-en beskriver kravene til applikasjonen når en bruker skal logge seg inn i systemet. En kundebruker skal bare ha tilgang til gjennomkjøringer til deres tilhørende bedrift og grunnleggende funksjoner. Derimot en administratorbruker skal ha tilgang til alle gjennomkjøringer og alle systemets funksjoner.

Tabell 2.2: User story for å kunne logge inn.

Som	administrator/kundebruker
ønsker jeg	å logge inn
slik at	jeg kan få tilgang til systemets funksjoner.

Liste over krav til story:

- Jeg må kunne oppgi e-post og passord for å logge inn.
- Systemet sjekker om jeg har oppgitt gyldig e-post og passord.

- Dersom jeg har glemt passordet, ønsker jeg å ha en mulighet til å tilbakestille passordet mitt.

2.3 Endre passord

Her beskrives kravene til systemet når en administrator/kundebruker har lyst til å endre passordet sitt til systemet.

Tabell 2.3: User story for å kunne endre passord.

Som	administrator/kundebruker
ønsker jeg	å kunne endre passordet mitt
slik at	slik at jeg kan ta i bruk ønsket passord.

Liste over krav til story:

- Passordet mitt må inneholde minst 8 tegn, inkludere store og små bokstaver og tall.
- For å endre passordet må jeg skrive inn det nåværende passordet, samt skrive inn det nye passordet 2 ganger for å forsikre meg om at jeg har skrevet det riktig.
- Jeg ønsker å motta en mail når passordet har blitt endret.
- Passordet mitt skal hashes med en sikker algoritme og krypteres før det sendes til databasen for lagring.

2.4 Velge behandlingsenheter

Når en bruker skal vise frem gjennomkjøringer fra spesifikke behandlingsenheter, så må man definere hvordan brukeren skal velge hvilke behandlingsenheter de vil se gjennomkjøringer for.

Tabell 2.4: User story for velging av behandlingsenheter.

Som	administrator/kundebruker
ønsker jeg	å velge behandlingsenheter
slik at	jeg kan endre hvilke behandlingsenheter jeg ser en liste med gjennomkjøringer for.

Liste over krav til story:

- Jeg skal bare ha tilgang til behandlingsenheter jeg har rettigheter til.
- Jeg skal kunne velge behandlingsenhet fra en liste.
- Når jeg velger en behandlingsenhet så skal jeg få se gjennomkjøringer fra denne behandlingsenheten.

2.5 Filtrere data

Storyen beskriver at administratorer/kundebrukere kan filtrere grafen som visualiseres for å jevne ut lokale topp/bunnpunkter. Grafen vil dermed bli "glattere" og øke lesbarheten for brukerne. Grafer som skal kunne filtreres vil være trykkgraf, temperaturgraf og graf for g-krefter.

Tabell 2.5: User story for kunne filtrere grafene som vises.

Som	administrator/kundebruker
ønsker jeg	å ha mulighet til å filtrere grafene som visualiseres
slik at	jeg kan fjerne uønsket støy fra målingene.

Liste over krav til story:

- Jeg ønsker å kunne velge filtreringsgraden ("hardheten" av filtreringen) til en graf ved hjelp av en glidebryter.
- Jeg ønsker at grafendringen skjer umiddelbart.
- Jeg ønsker at originaldataen forblir uendret.
- Filtrert graf vil ikke bli tatt vare på i systemet, men kan eksporteres til fil hvis jeg ønsker å ta vare på den (se kapittel 2.7).

2.6 Lagre historiske data

Av historisk data så skal binærfilene fra hver gjennomkjøring lagres på serveren, mens data som blir beregnet ut i fra binærfilene skal lagres i databasen. På denne måten legger vi bare inn dataen som skal visualiseres i databasen.

Tabell 2.6: User story for lagring av historiske data.

Som	administrator/kundebruker
ønsker jeg	at all historiske data lagres
slik at	jeg kan visualisere og vise grafer fra tidligere gjennomkjøringer.

Liste over krav til story:

- Jeg ønsker å kunne velge tidligere gjennomkjøringer av Sensorfisken.
- Jeg ønsker at tidligere data er datostemplet slik at jeg vet når testen ble gjennomført.
- Jeg skal bare kunne se historiske data jeg har rettigheter til.

2.7 Eksportere filer

Brukere skal kunne ha muligheten til å eksportere forskjellige grafer og data. Her skal man kunne eksportere i flere forskjellige format, PDF, CSV og PNG.

I PDFen som eksporteres skal alle grafer være inkludert.

For CSV filene skal det opprettes en fil for hver type data fra databasen. Hvis alle type data er valgt så legges alle CSV filene i en komprimert mappe.

Hvis man ønsker så kan man også eksportere som en PNG, men da eksporteres bare en graf om gangen.

Tabell 2.7: User story for eksportering av filer.

Som	administrator/kundebruker
ønsker jeg	å eksportere grafiske filer og datafiler
slik at	jeg kan bruke de utenfor systemet.

Liste over krav til story:

- Jeg må kunne trykke på en knapp som laster ned valgt data.
- Det må være mulig å velge mellom binærfilen til gjennomkjøringen jeg ser på eller en grafisk fil.
- Jeg ønsker å kunne velge filtreringsgraden på grafene som skal eksporteres (se kapittel 2.5).

- Jeg må kunne velge mellom filformatene PDF, CSV og PNG når jeg laster ned filene.

2.8 Sammenligne gjennomkjøringer

Denne user storyen beskriver kravene ved sammenligning av to eller flere gjennomkjøringer.

Tabell 2.8: User story som beskriver sammenligning av forskjellige gjennomkjøringer.

Som	administrator/kundebruker
ønsker jeg	å ha mulighet til å se to eller flere forskjellige gjennomkjøringer opp mot hverandre tegnet opp i samme figur
slik at	jeg kan sammenligne de forskjellige gjennomkjøringene.

Liste over krav til story:

- Jeg ønsker å kunne velge opptil seks forskjellige gjennomkjøringer og se grafene opp mot hverandre.
- De gjennomkjøringene jeg velger skal vises i en graf hvor en kurve samsvarer med en gjennomkjøring
- Jeg ønsker å ha flere typer data å velge mellom når jeg skal sammenligne gjennomkjøringer.
- Jeg ønsker å ha mulighet til å laste ned en fil med sammenligningen av grafene.

2.9 Slette gjennomkjøringer

Denne storyen beskriver kravene ved sletting av en gjennomkjøring. Det er kun administratorer av systemet som skal ha mulighet til å slette gjennomkjøringer. Når en gjennomkjøring er slettet, skal binærfilen fortsatt bestå i systemet.

Tabell 2.9: User story som beskriver sletting av gjennomkjøringer.

Som	administrator
ønsker jeg	å slette gjennomkjøringer
slik at	jeg kan fjerne unødvendig eller dobbeltlagret data.

Liste over krav til story:

- Når jeg trykker på slett, må jeg bli spurt om jeg er sikker på valget mitt.
- Jeg ønsker at det kun er dataene som er lagt inn i databasen som slettes. Binærfilen skal fortsatt ligge i systemet slik at de ikke er borte for godt.
- Etter at en gjennomkjøring er slettet, skal den være utilgjengelig for både kundebrukere og administratorer.

2.10 Legge inn rådata fra gjennomkjøring

Denne storyen beskriver kravene for opplasting av rådata fra Sensorfisken til systemet. Opplasting av rådata er en av de mest sentrale funksjonalitetene, da det legger grunnlaget for å ha mulighet til å visualisere data fra gjennomkjøringene.

Tabell 2.10: User story som beskriver det å legge inn rådata fra en gjennomkjøring.

Som	administrator
ønsker jeg	å kunne legge inn rådata fra gjennomkjøringer
slik at	dataen lagres og kan brukes senere.

Liste over krav til story:

- Jeg ønsker å ha mulighet til å laste opp en fil fra Sensorfisken.
- Jeg ønsker at systemet sjekker om det faktisk er en binærfil som lastes opp.
- Jeg ønsker å velge hvilken kunde og hvilken behandlingsenhet dataen tilhører, og datoen testen er gjennomført.
- Når filen er lagt inn og tilhørende informasjon er lagt til, vil jeg trykke “lagre”.
- Jeg ønsker at systemet tolker binærfilen og legger til verdier i databasen.
- Jeg ønsker at dataene lagres både som binærfil og med relevante verdier i databasetabellene.

2.11 Se flere grafiske fremstillinger av data

En gjennomkjøring inneholder flere typer data. Når man skal visualisere data fra gjennomkjøringer vil man gjerne at grafene endrer seg ut fra hvilken data som er valgt. Denne storyen beskriver kravene som omhandler visualisering av data.

Tabell 2.11: User story som beskriver visualisering av data.

Som	administrator/kundebruker
ønsker jeg	å ha flere forskjellige grafiske fremstillinger av dataen
slik at	flere aspekter av den mekaniske avluseren kan vurderes.

Liste over krav til story:

- Jeg ønsker å kunne velge å se fremstilling av trykkdata, temperaturdata, akselerasjonsdata og data for støt.
- Jeg ønsker å kun ha mulighet til å se data som jeg har tilgang til.
- Jeg ønsker at når jeg velger en annen fremstilling så skjer denne endringen øyeblikkelig.
- Ved valg av trykk-, temperatur- eller akselerasjonsdata ønsker jeg å se et linjediagram med tid på x-aksen og henholdsvis bar, grader celsius og g-krefter på y-aksen.
- Ved valg av data for støt ønsker jeg et søylediagram med antall g-krefter på x-aksen og antall støt på y-aksen.
- Jeg ønsker at verdiområdet på aksene tilpasser seg datasettets størrelse automatisk.

2.12 Legge til bedrift

Denne storyen beskriver kravene når man legger til en bedrift i systemet. Hensikten med å legge til en bedrift er å ha mulighet til å knytte kundebrukere og gjennomkjøringer opp mot denne bedriften, slik at en kundebruker kun har mulighet til å se de forskjellige gjennomkjøringene som tilhører den bedriften.

Tabell 2.12: User story som beskriver det å legge til en bedrift i systemet.

Som	administrator
ønsker jeg	å ha mulighet til å legge inn nye bedrifter i systemet
slik at	data fra en gjennomkjøring kan knyttes til bedriften den tilhører.

Liste over krav til story:

- Jeg ønsker å ha mulighet til å legge til ny bedrift.
- Jeg ønsker å kunne legge til firmanavn, e-post, kontaktperson og telefonnummer for denne kunden.

2.13 Endre informasjon om bedrift

Denne storyen beskriver kravene ved endring av en bedrift. Man bør ha mulighet til å endre informasjonen i tilfelle informasjonen er feil, eller at den endrer seg etter den er lagt inn.

Tabell 2.13: User story for å endre lagret informasjon om en bedrift.

Som	administrator
ønsker jeg	å kunne endre informasjonen som ligger inne på en bedrift
slik at	den informasjonen som er registrert er oppdatert.

Liste over krav til story:

- Jeg ønsker at jeg kan endre lagret firmanavn, e-post, kontaktperson og telefonnummer.

2.14 Slette kundebruker

Denne storyen beskriver kravene ved sletting av en kundebruker. Dersom administrator sletter brukeren, skal brukeren settes som inaktiv, mens brukerens data skal fortsatt ligge i systemet.

Tabell 2.14: User story for å slette kundebruker.

Som	administrator
ønsker jeg	å kunne slette en kundebruker
slik at	kunden ikke lenger eksisterer i systemet.

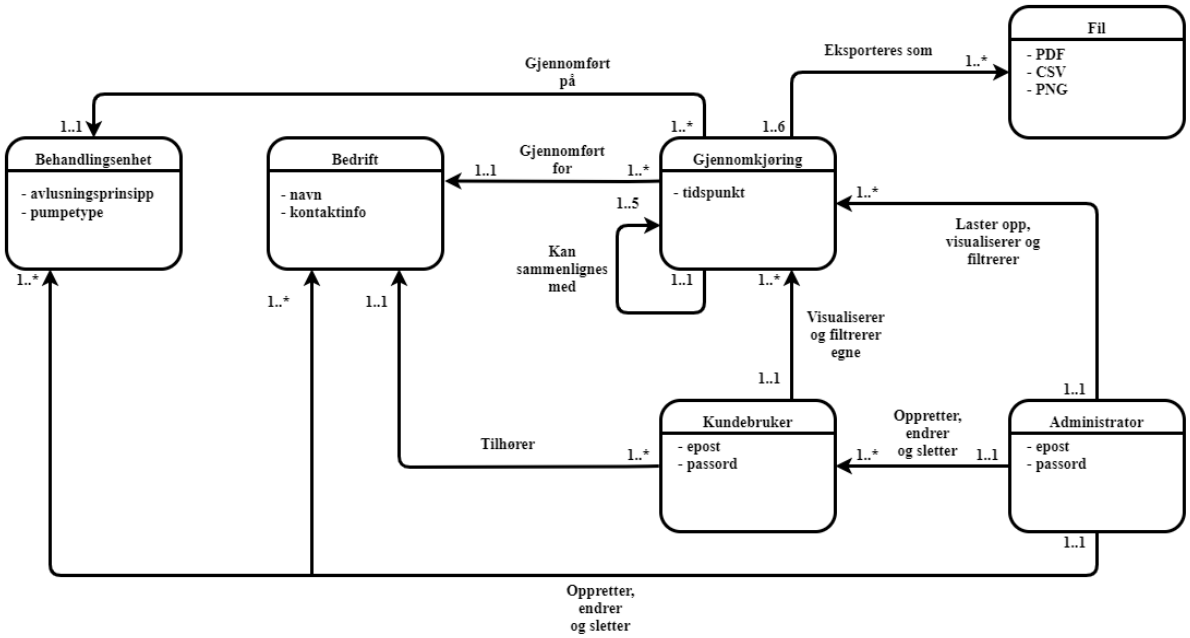
Liste over krav til story:

- Jeg ønsker å ha mulighet for å slette en bestemt bruker.
- Jeg ønsker at dersom jeg trykker på knappen så kommer det opp at jeg må bekrefte valget.
- Jeg ønsker at brukeren ikke slettes fra databasen, men at den settes som inaktiv.
- Jeg ønsker at dataene som tilhører brukeren (testdataene) fortsatt ligger inne i systemet etter at brukeren er slettet.
- Brukeren vil ikke ha tilgang til å logge inn i systemet etter at den har blitt slettet.

3 Domenemodell

En domenemodell er blant de viktigste delene av kravdokumentet i systemutvikling. Den brukes ofte for å få utviklere og domeneeksperter til å få en felles forståelse av problemdomenet. Den brukes også for å oppnå en god kommunikasjon mellom utviklerteamet og oppdragsgiver (Norges teknisk-naturvitenskaplige universitet, u.å).

Vår domenemodell består i grove trekk av entitetene *Administrator*, *Kundebruker*, *Bedrift*, *Gjennomkjøring*, *Behandlingsenhet* og *Fil*. Hver av entitetene har visse attributter som er relevante for systemet og er koblet til minst én annen entitet.

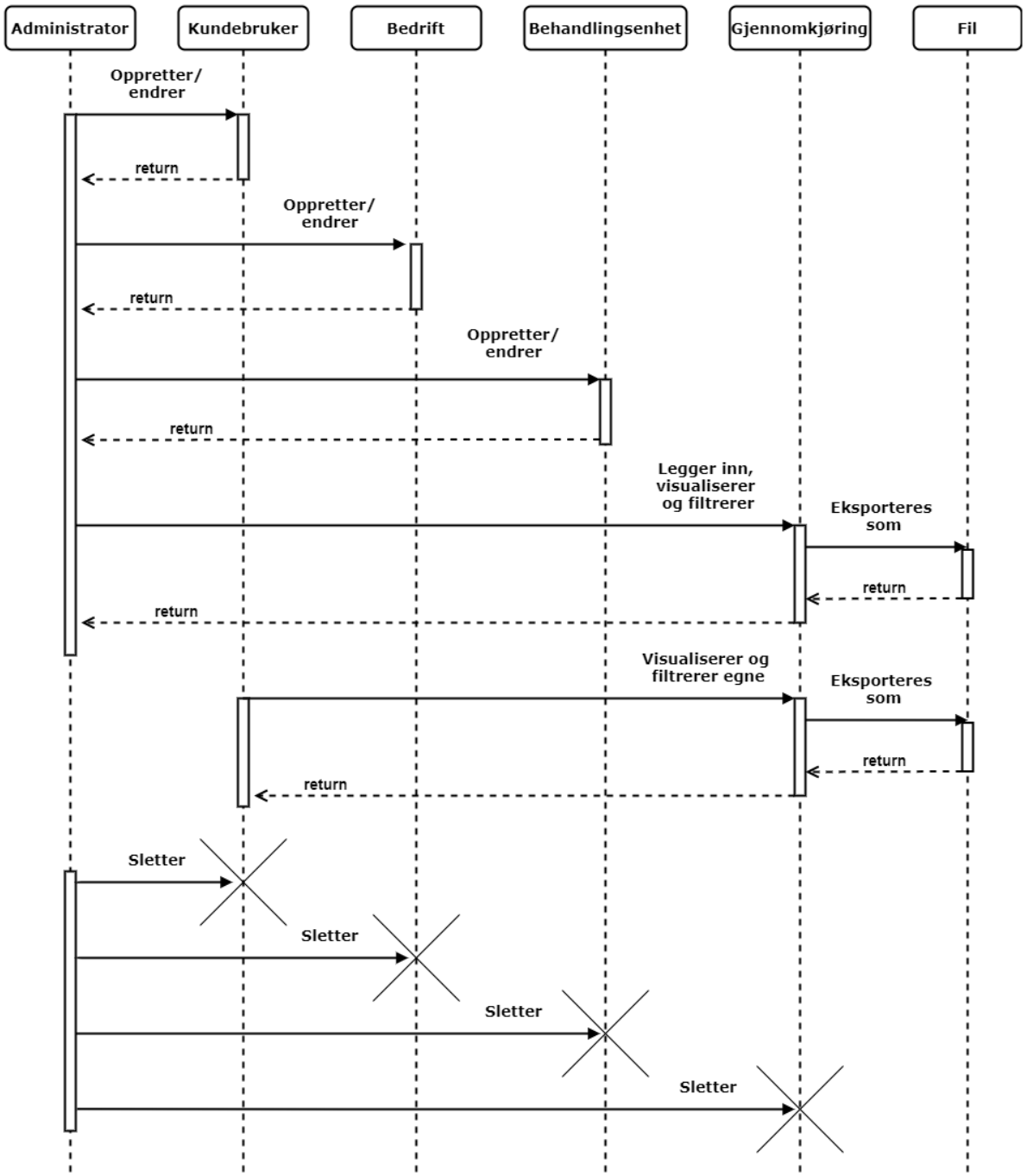


Figur 3.1: Domenemodell som gir en overordnet beskrivelse av problemdomenet. De ulike entitetene er markert i de rektangulære boksene. Pilene og tilhørende tekst beskriver forholdet mellom entitetene og hvordan de agerer med hverandre.

4 Sekvensdiagram

Et sekvensdiagram viser interaksjonen mellom objekter i et system i sekvens, altså i hvilken rekkefølge interaksjonene skjer. De vertikale linjene i et sekvensdiagram viser de forskjellige entitetene, mens de horisontale linjene viser meldingene eller interaksjonene mellom disse (Fowler, 2003).

Sekvensdiagrammet under tar utgangspunkt i domenemodellen (se figur 3.1) for å teste om entitetene man har utarbeidet er hensiktsmessige (Norges teknisk-naturvitenskapelige universitet, u.å.). Under utforming av diagrammet vil man kunne se om det er behov for å legge til/endre entiteter, eller om det er enkelte av entitetene i domenemodellen som er overflødige for bruken av systemet. I vårt sekvensdiagram ser man hvordan *Administrator* samhandler med *Kundebruker*, *Bedrift*, *Behandlingsenhet* og *Gjennomkjøring*. Det blir også beskrevet hele forløpet fra *Administrator/Kundebruker* ser på grafene til en *Gjennomkjøring*, fram til de eventuelt eksporteres til et annet filformat.



Figur 4: Sekvensdiagram som viser samhandling mellom entitetene, som er representert ved de rektangulære boksene øverst i figuren. Diagrammet viser hvordan administrator behandler kundebrukere, bedrifter, behandlingsenheter og gjennomkjøringer, i tillegg til hvordan kundebruker og administrator kan visualisere, filtrere og eksportere gjennomkjøringer.

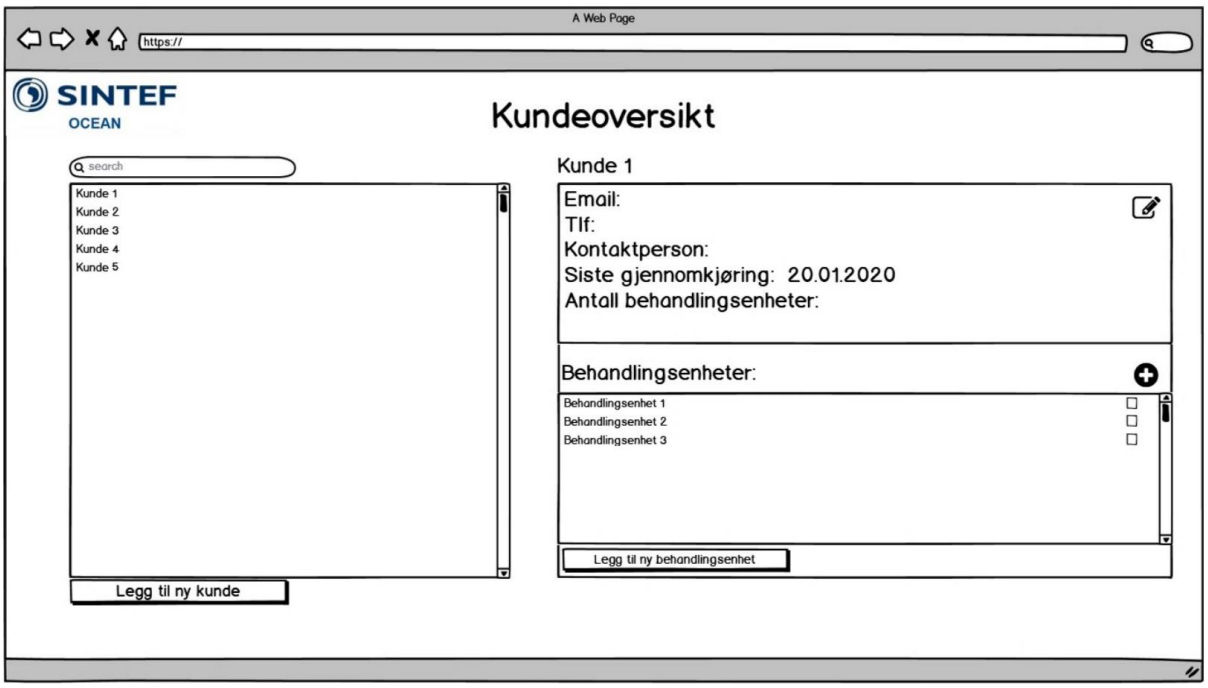
5 Prototype

Prototyper designes slik at utviklerne har et utgangspunkt når de skal starte med utviklingen av nettsiden. De kan også brukes for å bedrive tidlige brukertester på brukere. Vi har i dette prosjektet valgt å bruke wireframes (se kapittel 5.1) ved hjelp av Balsamiq som er et verktøy på nett for å prototype nettsider. I Balsamiq kan man velge mellom forskjellige elementer man kan finne igjen på en nettside, og plassere disse elementene der man ønsker på prototyp-nettsiden. Begrunnelsen for valget av å bruke wireframes til prototyping er at det går raskere å lage i forhold til HTML-prototyper, men man kan fortsatt få tilnærmet lik pek-og-klikk-funksjonalitet. Det er også enklere å gjøre endringer dersom dette er nødvendig underveis i prosessen. Valget om å ikke bruke storyboards kommer først og fremst av at et storyboard er veldig tidkrevende i forhold til hvor lite funksjonalitet som dekkes. Et storyboard tar også mye plass, og da vi ikke har et fast sted å jobbe på kunne det være problematisk å frakte et storyboard med seg hele tiden.

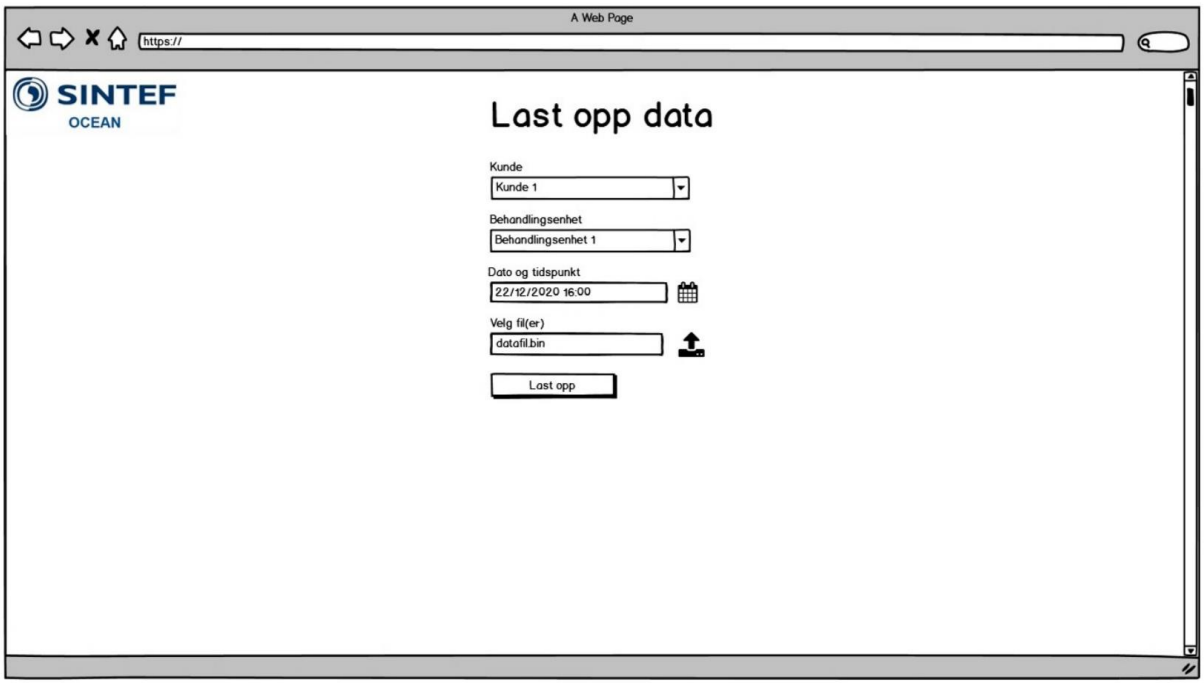
5.1 Wireframes

Wireframes er en type representasjon av hvordan man har tenkt at nettsiden skal se ut, og kan enten tegnes på ark eller utformes ved hjelp av wireframe-verktøy på nett. Med disse verktøyene kan man lage navigerbare wireframes, slik at de blir godt egnet til å benyttes i brukertester (Norges teknisk-naturvitenskaplige universitet, u.å.). I dette kapittelet vises de mest relevante wireframene fra prosjektet.

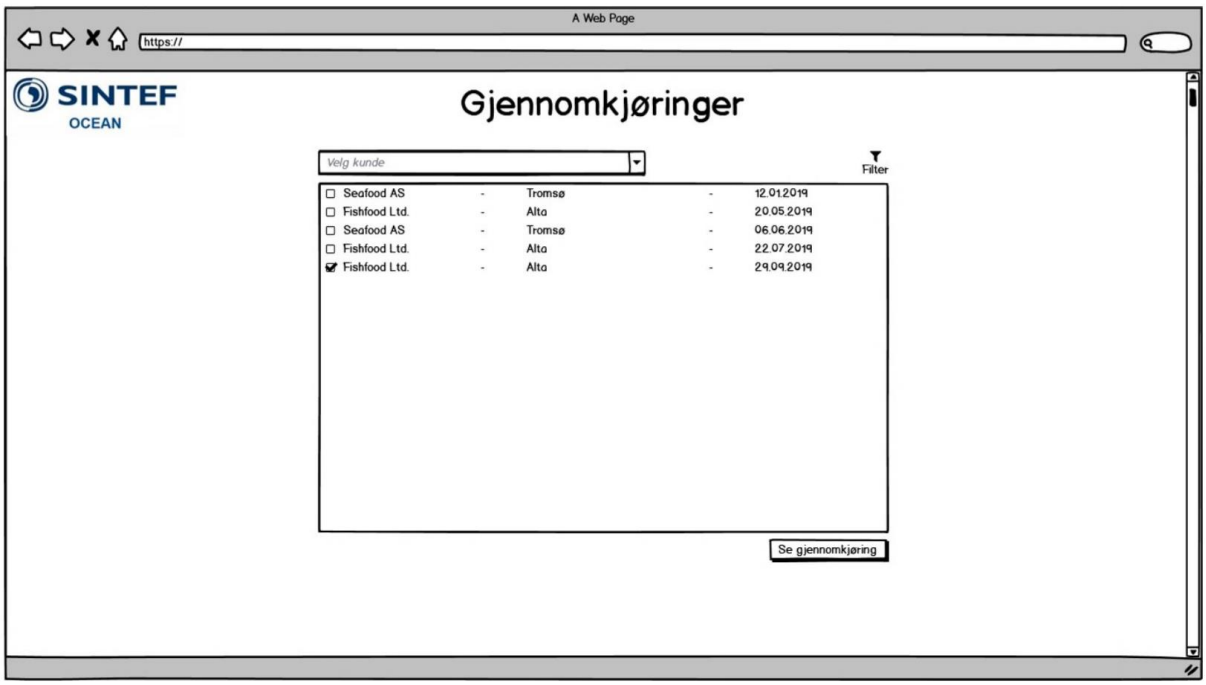
Figur 5.1.1: Fra kontrollpanelet skal det være enkelt for administrator til å få tilgang til alle systemets funksjoner. De tre valgmulighetene er "last opp data", "gjennomkjøringer" og "administrer kunder".



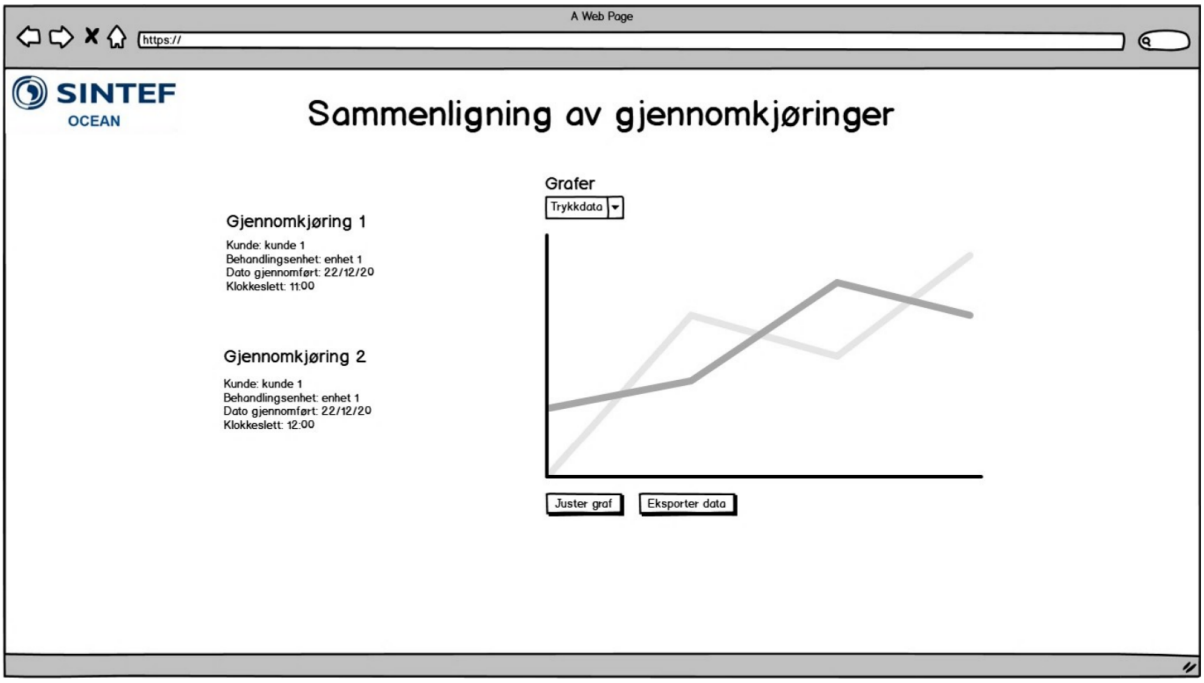
Figur 5.1.2: Kundeoversikten skal bare være tilgjengelig for administrator. Herfra skal det være mulig å legge til nye kundebrukere og behandlingsenheter.



Figur 5.1.3: Side hvor en administrator laster opp data fra Sensorfisken til systemet. Her skal vedkommende kunne velge hvilken kunde disse dataene tilhører, hvilken behandlingsenhet dataene kommer fra og når målingen er gjort.



Figur 5.1.4: Denne siden skal være hovedsiden til kundene, og gir en oversikt over alle gjennomkjøringer for hver enkelt kunde. De skal bare kunne se sine egne gjennomkjøringer, men administrator skal kunne se alle. Herfra skal man også kunne markere en eller flere gjennomkjøringer for å sammenligne de.



Figur 5.1.5: På denne siden vil man kunne få se sammenligning av flere gjennomkjøringer, altså at samme datatype fra flere gjennomkjøringer vises som grafer i samme figur. Det skal være mulig å justere og filtrere grafen, deretter skal man kunne eksportere til ønsket filformat. Filtrering skjer ved at man "drar" på en slider slik at grafen blir "glattere" og øker lesbarheten. På x-aksen vises tiden i sekunder, og på y-aksen vises målingsenheten til den valgte datatypen.

Referanser

- Caharija, W., Venås, B., Svendsen, E., Schrøder, M. B., Pedersen, M. O., Lie, O. K., … Ohrem, S. J. (2019). *Verktøy for kartlegging av forhold i enheter for føring, håndtering og behandling av laks (Sensorfisk) (SINTEF rapport L4)*. Hentet 14. januar 2020 fra https://www.fhf.no/prosjekter/prosjektbasen/901397/?fileurl=https://fhfno.sharepoint.com/sites/pdb/Publisertedokumenter/307829FHprnr901397_KVALYSIS_faglig_sluttrapport_v3.pdf.PDF&filename=Sluttrapport:%20Verkt%C3%B8y%20for%20kartlegging%20av%20forhold%20i%20enheter%20for%20f%C3%B8ring,%20h%C3%A5ndtering%20og%20behandling%20av%20laks%20(sensorfisk)
- Flatsetsund Engineering. (u.å.). *FLS Avluser: For avlusing av laks.* Hentet 18. februar 2020 fra https://www.fls.no/avlusing-av-laks/
- Fowler, Martin. (2003). UML Distilled: A Brief Guide to the Standard Object Modeling Language, Third Edition. Hentet 26. februar 2020 fra https://my.safaribooksonline.com/book/software-engineering-and-development/uml/0321193687/sequence-diagrams/ch04
- Norges teknisk-naturvitenskapelige universitet (NTNU). (u.å.). *Kravdokument*. Hentet 19. februar 2020 fra http://www.aitel.hist.no/fag/hpr/retnLinjerDataingenior.zip
- SINTEF. (2017). *Standardisert metodikk for kvalifisering av mekaniske avlusingssystemer (KVALISYS)*. Hentet 19. februar 2020 fra https://www.sintef.no/prosjekter/standardisert-metodikk-for-kvalifisering-av-mekaniske-avlusingssystemer/

Utvikling av brukergrensesnitt og brukerfunksjoner for Sensorfisk

Systemdokument

Vedlegg D

Forfattere:

Herman Ryen Martinsen

Sander Nicolausson

Trond Jacob Rondestvedt

Jørgen Aasvestad

Versjon 2.1

11.05.2020

Revisjonshistorie

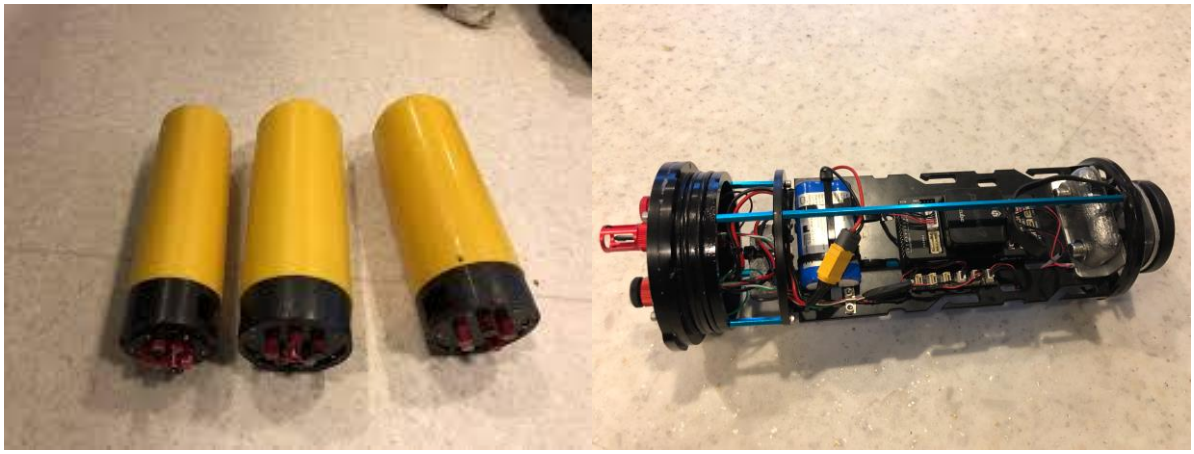
Dato	Versjon	Beskrivelse	Forfatter
02.04.2020	1.0	Førsteutkast	Herman Ryen Martinsen Sander Nicolausson Trond Jacob Rondestvedt Jørgen Aasvestad
08.04.2020	1.1	Revideringer	Herman Ryen Martinsen Sander Nicolausson Trond Jacob Rondestvedt Jørgen Aasvestad
17.04.2020	2.0	Andreutkast	Herman Ryen Martinsen Sander Nicolausson Trond Jacob Rondestvedt Jørgen Aasvestad
11.05.2020	2.1	Revideringer	Herman Ryen Martinsen Sander Nicolausson Trond Jacob Rondestvedt Jørgen Aasvestad

Innholdsfortegnelse

1 Introduksjon

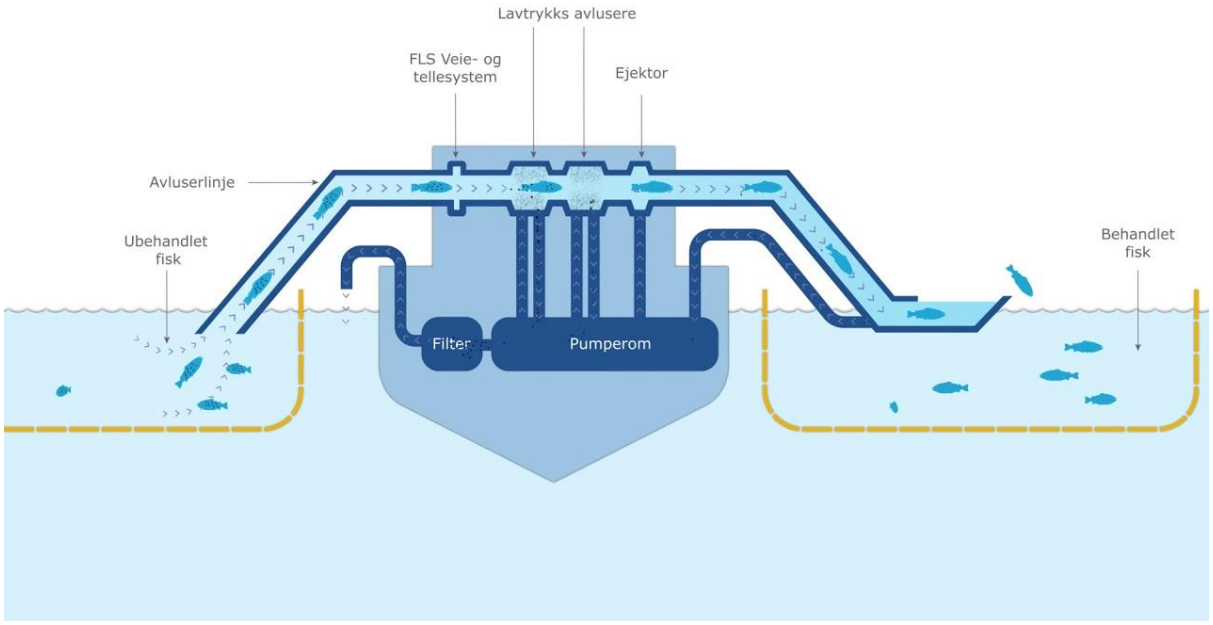
Dette dokumentet er skrevet i forbindelse med bacheloroppgave 061 ved IDI AIT, NTNU for oppdragsgiver SINTEF Ocean AS. Hensikten med dokumentet er å gi en oversikt over hvordan systemet er bygd opp. Dokumentet inneholder en oversikt over arkitekturen og strukturen til prosjektet, samt UML-diagrammer som databasemodell og klassediagram. Det inneholder også en grundig beskrivelse over hvordan sikkerheten i systemet er ivaretatt og hvordan kvaliteten av kildekoden er sikret ved bruk av testing og kontinuerlig integrasjon.

I et tidligere prosjekt har SINTEF Ocean AS utarbeidet et produkt de kaller Sensorfisk. Sensorfisken er en sylinderformet gjenstand som inneholder en mikrokontroller og sensorer som måler trykk, temperatur, akselerasjon og magnetisme.



Figur 1.1: Produktet Sensorfisk som SINTEF Ocean AS har utviklet (SINTEF, 2017).

Formålet med Sensorfisken er at den skal brukes i oppdrettsnæringen for å etterligne en fisk som går gjennom et mekanisk avlusersystem. De forskjellige sensorene registrerer hvilken behandling Sensorfisken får gjennom behandlingsenheten, og man kan for eksempel se på kraftige endringer i akselerasjon som et støt mot rørveggen.
For å bestemme Sensorfiskens posisjon inne i den mekaniske avluseren festes magnetbånd på bestemte punkter i rørene, slik at magnetometeret i Sensorfisken gir utslag ved passering av disse båndene (Caharija et al., 2019, s.8).



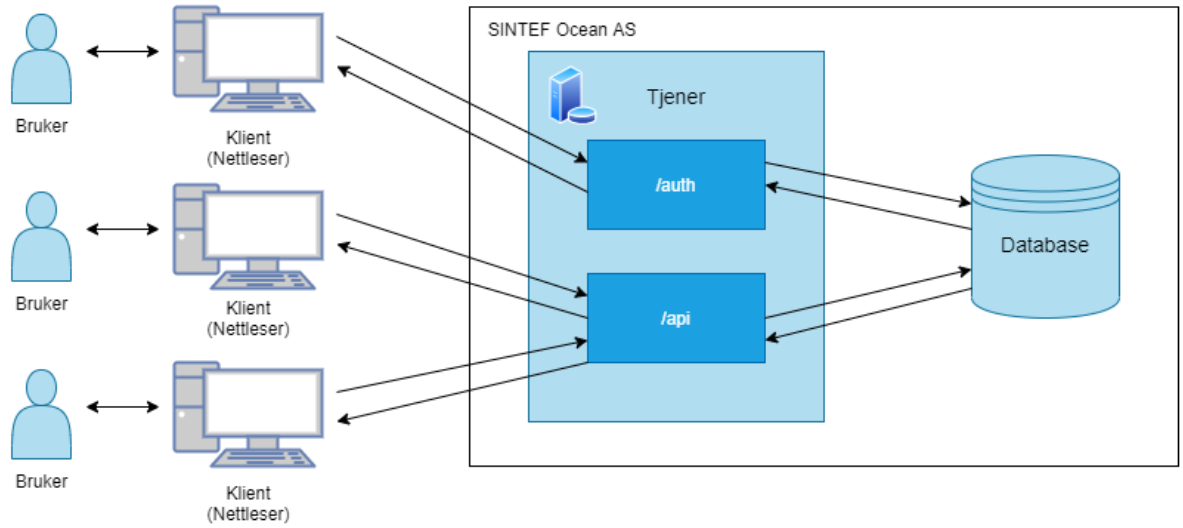
Figur 1.2: Virkemåte for en mekanisk avluser fra Flatsetsund Engineering (Flatsetsund Engineering, u.å.). Ubehandlet fisk blir dratt opp til avluserlinja, der de blir veid og behandlet gjennom en lavtrykks-avluser. Til slutt blir de pumpet ut ved hjelp av ejektoren.

Systemet vi har fått i oppgave å lage er et web-basert grafisk brukergrensesnitt for opplasting, analyse og visualisering av data fra Sensorfisken. Sensorfisken kjøres gjennom en mekanisk avluser (se figur 1.2), hentes opp og kobles deretter til en pc via USB. Ansatte ved SINTEF vil da kunne laste opp binærfilen fra Sensorfisken på vår nettside. Systemet vil da lese ut dataene fra binærfilen og laste disse opp i databasen.

Etter at binærfilen er lastet opp og nødvendig data lagret vil man kunne se grafiske framstillinger av dataen. Man vil også ha mulighet for å sammenligne flere gjennomkjøringer av Sensorfisken slik at man kan se på likheter og ulikheter på flere gjennomkjøringer på samme avluser. Til slutt vil det være en funksjon for eksportering av de grafiske framstillingene av data både som PDF, PNG og CSV.

2 Arkitektur

I komplekse systemer er det viktig å være konsekvent med arkitekturen for å opprettholde oversikten etter hvert som systemet utvikles. Med utgangspunkt i dette har vi valgt å dele systemet i ulike hovedkomponenter, som kan fungere og forbedres uavhengig av hverandre. Figur 2.1 viser de viktigste komponentene som inngår i systemet.



Figur 2.1: Arkitekturskisse med de viktigste komponentene i systemet.

Klienten kjøres i nettleseren på brukerens datamaskin. Brukeren er i dette tilfellet enten en SINTEF-ansatt eller en av deres kunder. Brukeren kommuniserer med systemet gjennom det grafiske brukergrensesnittet til klienten. Klienten gjør så forespørsler mot tjenerens endepunkt på vegne av brukeren, og visualiserer data den får tilbake fra tjeneren.

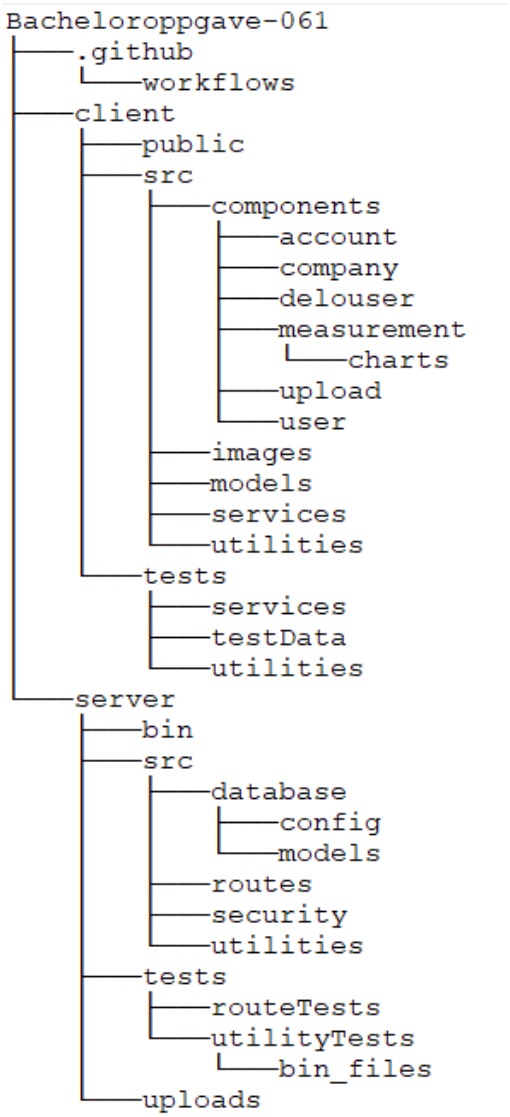
Tjeneren tar imot forespørsler fra klienten og behandler disse. Den utfører videre spørringer mot databasen for å fullføre oppgavene fra klienten. Tjeneren har to hovedkategorier av endepunkt:

> **/auth:** Behandler forespørsler som omhandler autentisering av brukeren. Dette kan blant annet være innlogging, utlogging, fornying av token osv.
>
> **/api:** Behandler forespørsler der brukeren etterspør ressurser i API-et. Får å få tilgang til disse må brukeren være autentisert.

Databasen tar imot spørringer fra tjeneren og henter ut, setter inn eller endrer data som ligger lagret. Her ligger blant annet informasjon om brukere, gjennomkjøringer og bedrifter lagret.

3 Prosjektstruktur

Prosjektstrukturen er hovedsakelig delt inn i to hovedmapper: Én for klienten (client) og én for tjeneren (server). All kildekode som brukes i prosjektet ligger i de respektive src-mappene.



Figur 3.1: Mappestrukturen til prosjektet er i all hovedsak delt inn i en klientdel og en tjenerdel.

3.1 Klient

På klientsiden har vi *components* som er mappa hvor komponentene til de forskjellige nettsidene ligger og *utilities* hvor metodene for å håndtere dataene ligger. I tillegg ligger modellene for de forskjellige objektene slik de ligger i databasen i *models*, mens i *services* ligger metodene som håndterer HTTP-forespørsler. Vi har en egen mappe for tester på klientsiden. Samtidig så har vi en public mappe som inneholder manifest, noen ikoner og *index.html* filen.

3.2 Tjener

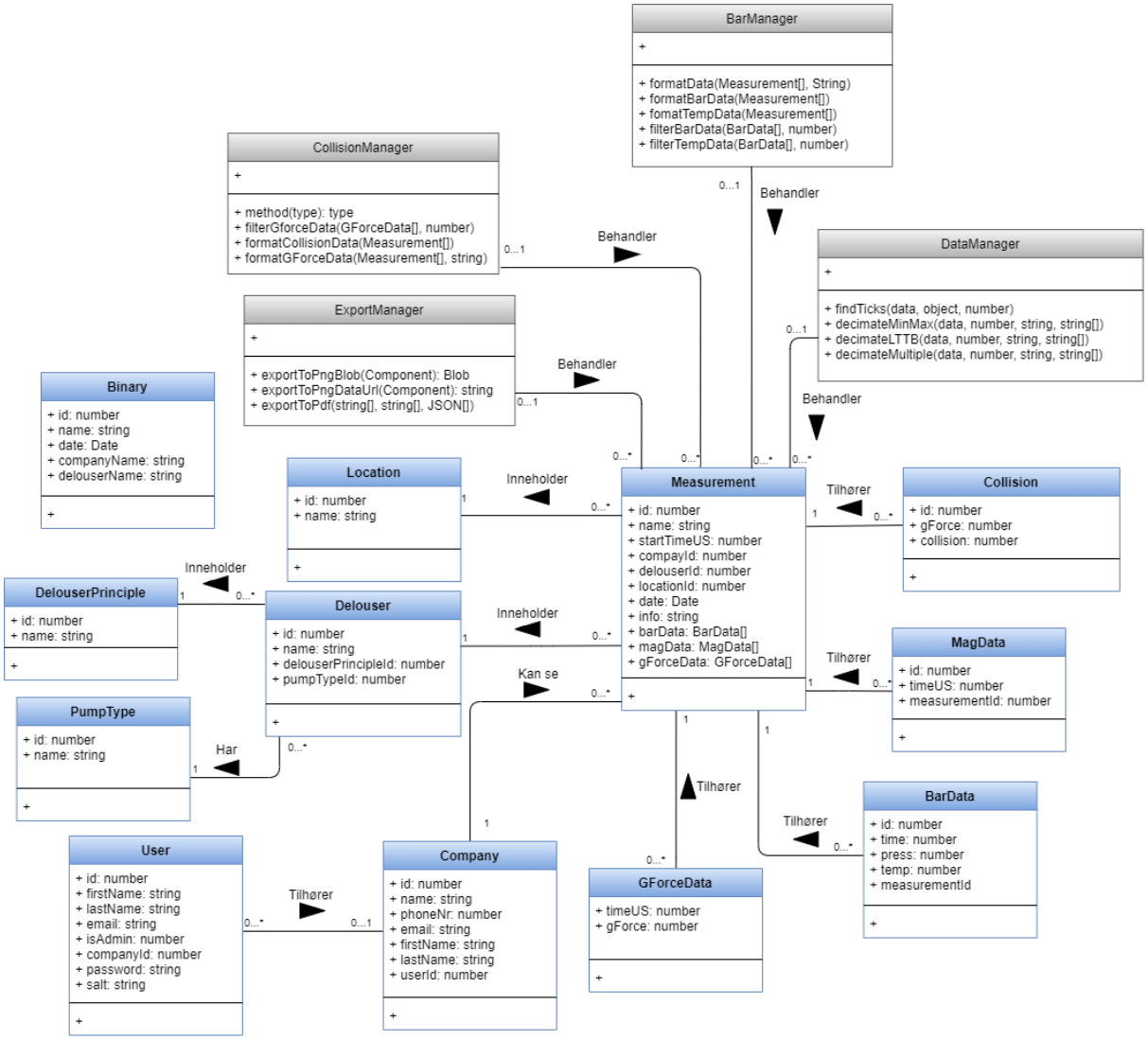
På tjenersiden tar *database*-mappa seg av konfigurasjon og modellering av databasen. Her brukes sequelize for å håndtere spørringer opp mot databasen. I *routes* ligger metodene som tar i mot de forskjellige HTTP-forespørslene fra klienten, mens i *utilities* ligger de større metodene som håndterer dataen før den blir sendt til klientsiden.
I *uploads* lagres binærfilene som lastes opp fra klienten. Vi har også en mappe for tester, og en for testdata på tjenersiden.
Alle filer tilknyttet det selv-signerte SSL-sertifikatet ligger i *security*-mappa.

4 Klassediagram

Som nevnt tidligere er prosjektstrukturen hovedsakelig delt i to deler, en klientside og en tjenerside, noe som også er tilfellet for inndeling av klassene. Ved en slik inndeling får man økt modularitet, altså bestående av flere individuelle deler, noe som kan være hensiktsmessig på flere måter. Blant annet kan det bidra til økt oversikt og lesbarhet for kodebasen, i tillegg til at man kan endre/forbedre ulike moduler uavhengig av hverandre. På klientsiden finner vi klasser som håndterer dataen som kommer fra tjeneren, i tillegg til å behandle og presentere innhold for nettleseren.

4.1 Klient

Figur 4.1 inneholder i hovedsak en oversikt over de ulike modellene samt verktøy-klassene. For å kunne bevare en oversiktlig figur er det fokusert på de mest sentrale klassene på klientsiden, og dermed er enkelte komponenter og tjeneste-klasser ikke inkludert.

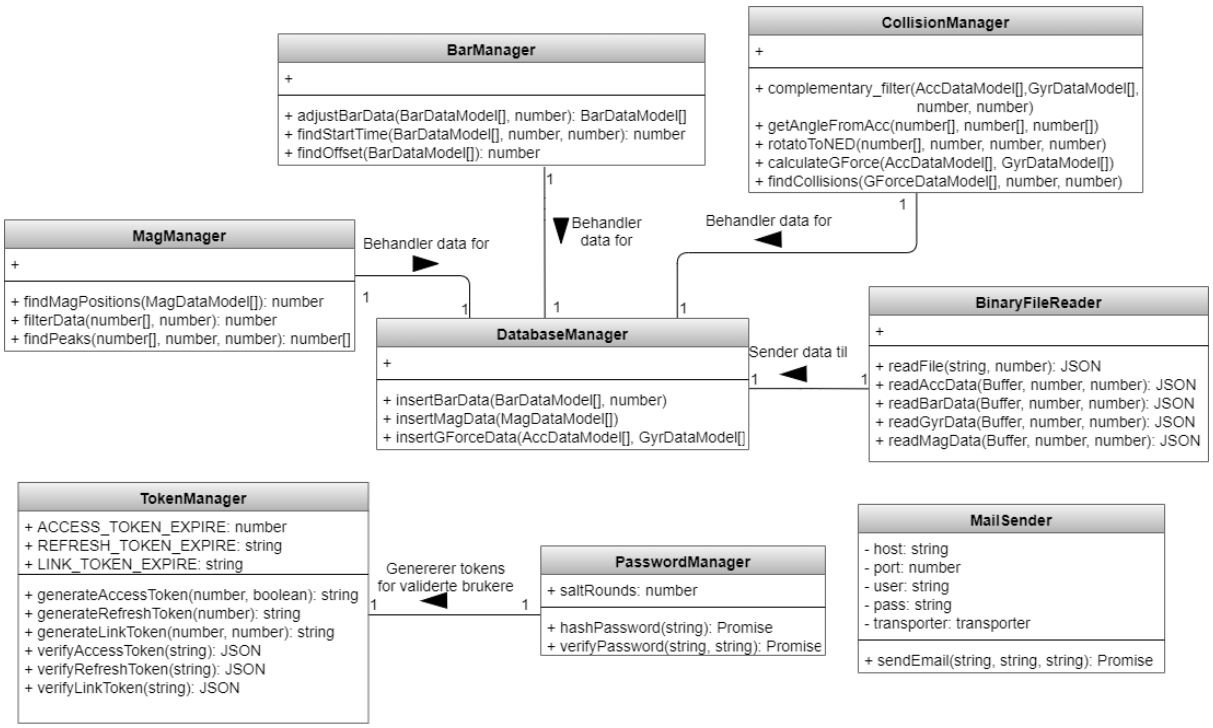


Figur 4.1: Klassediagram for klient-siden. Viser hvordan de ulike modellene - markert i blått - er knyttet opp mot en gjennomkjøring (Measurement), og hvordan verktøy-klassene - markert i grått - behandler data.

Klassediagrammet vil naturligvis være noe likt databasemodellen, hvertfall oppsettet av modell-klassene på klientsiden. Fra figuren ser man hvordan de ulike modellene er knyttet sammen, da gjerne med en gjennomkjøring, Measurement, som bindeledd for de forskjellige modellene som er markert i blått. Øverst i figuren finner man verktøy-klassene som er markert i grått, som har i oppgave å behandle, analysere og formatere data som er knyttet opp mot gjennomkjøringer.

4.2 Tjener

I figur 4.2 ser man klassediagrammet for tjenersiden. Her det lagt hovedvekt på de ulike utility-klassene som blir brukt. Modellklassene er ikke inkludert i følgende klassediagram, da disse er eksakte kopier av modellene i databasemodellen (figur 5.1). På denne måten vil klassediagrammet kunne illustrere en tydeligere oversikt, samt unngå overkompliserende faktorer.



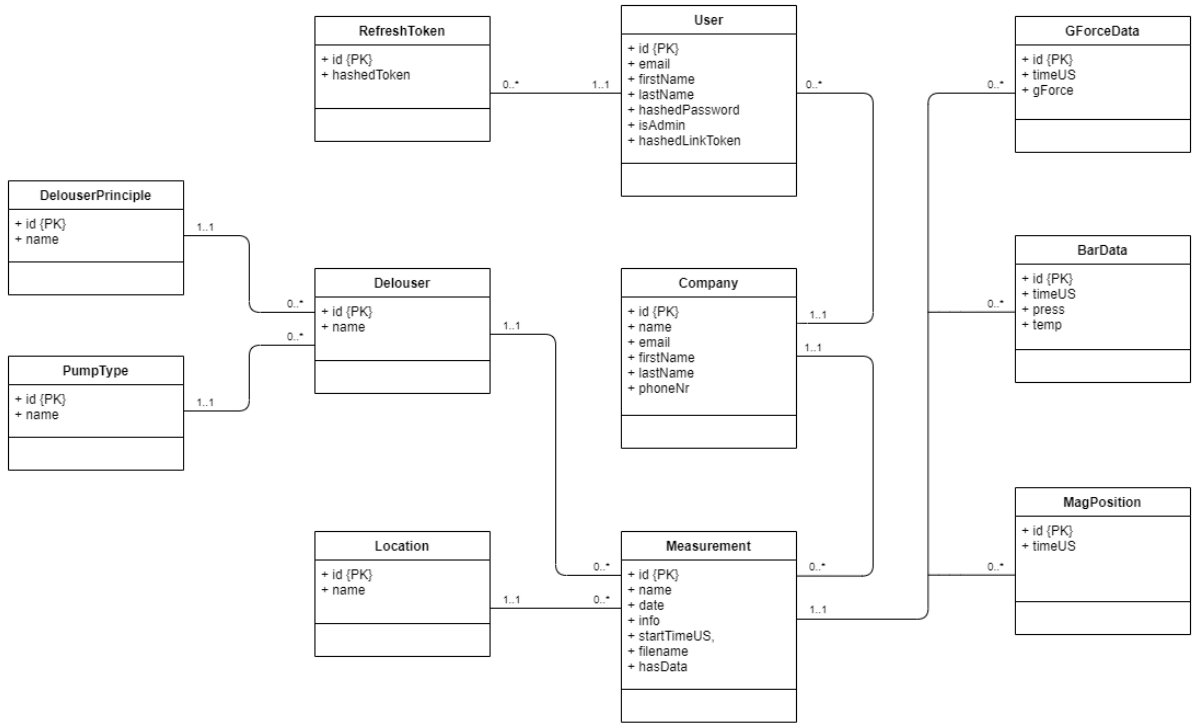
Figur 4.2: Klassediagram for tjener-siden. Den viser samhandling mellom de ulike databehandlingsklassene og hvordan de sender og behandler data for DatabaseManageren. I tillegg vises klassene som håndterer passord- og brukerverifisering, samt klasse for mailutsendelse.

Fra figuren ser man at de ulike Manager-klassene (DatabaseManager, BarManager, et. al.) samhandler for å prosessere, bearbeide og lagre dataen man innhenter. BinaryFileReader brukes for å lese data fra binærfiler produsert av Sensorfisken. Dataen blir så sendt videre til DatabaseManager som håndterer innsetting i databasen. DatabaseManager benytter MagManager, BarManager og CollisionManager for å gjøre kalkulasjoner og bearbeide data før den settes inn i databasen. Med denne framgangsmåten gjøres flesteparten av

kalkulasjonene før data settes inn, slik at det ikke må gjøres hver gang man ønsker å aksessere den.

Tjenersiden inneholder også klasser som for passordhåndtering og tokenhåndtering, henholdvis PasswordManager og TokenManager. Disse er en del av sikkerhetslaget og brukes blant annet for å sørge for at passord blir kryptert samt at innlogging blir foretatt på en sikker måte. I tillegg finner man MailSender som er en klasse for mailutsendelse, slik at ønsket informasjon kan sendes automatisk fra tjeneren.

5 Databasemodell



Figur 5.1: Databasemodell for systemet.

I databasen lagres alt av informasjon om kunder, avlusere og gjennomkjøringer. All sensordata fra gjennomkjøringer vil ikke lagres i databasen, men kun den dataen som er relevant ved visualisering av grafene. Resterende data vil likevel være tilgjengelig ved behov, siden rådatafilen lagres lokalt på tjeneren. Trykk- og temperaturdataen er den eneste dataen som leses inn og lagres direkte. G-krefter og magnetbånd-posisjoner regnes ut ved hjelp av data fra akselerometer, gyroskop og magnetometer, og lagres deretter i databasen.

Kort forklaring av hver tabell i databasemodellen:

- For hvert *Company* lagres en id og et navn. I tillegg lagres det informasjon om en kontaktperson ved denne bedriften. Denne informasjonen består av en e-post, et fornavn, et etternavn og et telefonnummer.
- For hver *User* lagres en id, en e-post, et fornavn, et etternavn, et hashet passord og en variabel som forteller om brukeren har administratortilgang eller ikke. Det lagres også en hashet "linktoken", som brukes til å verifisere brukeren ved førstegangsregistrering.
    - En *User* tilhører et *Company*.

- For hver *DelouserPrinciple* lagres en id og et navn.
- For hver *PumpType* lagres en id og et navn.
- For hver *Delouser* lagres en id og et navn.
    - Hver *Delouser* har et *DelouserPrinciple* og en *Pumptype*.
- For hver *Location* lagres en id og et navn.
- For hver *Measurement* lagres en id, et navn, en dato, en tekst med informasjon om gjennomkjøringen, en starttid, et filnavn og en variabel som forteller om sensordata for gjennomkjøringen ligger lagret i databasen eller ikke.
    - Hver *Measurement* tilhører et *Company*, og har en *Delouser* og en *Location*.
- For hver *GForceData* så lagres det en id, et tidspunkt og en verdi for G-krefter.
    - Hver *GForceData* tilhører en *Measurement*.
- For hver *MagPosition* så lagres det en id og et tidspunkt.
    - Hver *MagPosition* tilhører en *Measurement*.
- For hver *BarData* så lagres det en id, et tidspunkt, en verdi for trykk og en verdi for temperatur.
    - Hver *BarData* tilhører en *Measurement*.
- For hver *RefreshToken* lagres en id og en hashet refresh token. Denne tabellen brukes til å holde oversikt over aktive refresh tokens (se kapittel 7.3).
    - Hver *RefreshToken* tilhører en *User*.

6 Server-tjenester

"A resource is anything that's important enough to be referenced as a thing in itself. If your users might 'want to create a hypertext link to it, make or refute assertions about it, retrieve or cache a representation of it, include all or part of it by reference into another representation, annotate it, or perform other operations on it', then you should make it a resource. (Richardson & Ruby, 2007)". En REST-ressurs har altså ikke en konkret definisjon, men brukes om noe som er viktig i et klient-tjener-system.

I dette kapitlet vil vi gå gjennom de ressursene vi har brukt i vårt system, og disse vil representeres på JSON-format.

6.1 Company

Klienter mottar REST-ressursen Company når det gjøres en forespørsel mot tjener-endepunktet */api/companies*. Denne ressursen beskriver bedriftene som er lagret i systemet.

```
{
	"id": INTEGER,
	"name": STRING,
	"email": STRING,
	"firstName": STRING,
	"lastName": STRING,
	"phoneNr": INTEGER
}
```

Figur 6.1: Ressursen Company representert som JSON med variabler og tilhørende datatype.

6.2 Delouser

Når det gjøres en forespørsel mot tjener-endepunktet */api/delousers* sendes REST-ressursen Delouser. Ressursen beskriver en mekanisk avluser slik den er lagret i systemet.

```
{
	"id": INTEGER,
	"name": STRING,
	"delouserPrincipleId": INTEGER,
	"pumpTypeId": INTEGER
}
```

Figur 6.2: Ressursen Delouser representert som JSON med variabler og tilhørende datatype.

6.3 DelouserPrinciple

Ressursen *DelouserPrinciple* tjenes på endepunktet */api/delouser-principles*. Denne beskriver behandlingsprinsippet som brukes i en mekanisk avluser.

```
{
	"id": INTEGER,
	"name": STRING
}
```

Figur 6.3: Ressursen DelouserPrinciple representert som JSON med variabler og tilhørende datatype.

6.4 Location

Ressursen *Location* beskriver et sted, og tjenes fra endepunktet /*api/locations*. I systemet brukes ressursen for å beskrive hvor en gjennomkjøring har foregått.

```
{
	"id": INTEGER,
	"name": STRING
}
```

Figur 6.4: Ressursen Location representert som JSON med variabler og tilhørende datatype.

6.5 Measurement

Ressursen Measurement tjenes fra endepunktet */api/measurements* og representerer en gjennomkjøring som har blitt gjort med Sensorfisken.

```
{
	"id": INTEGER,
	"name": STRING,
	"date": DATE,
	"info": TEXT,
	"startTimeUS": INTEGER,
	"filename": STRING,
	"hasData": BOOLEAN,
	"companyId": INTEGER,
	"delouserId": INTEGER,
	"locationId": INTEGER,
}
```

Figur 6.5: Ressursen Measurement representert som JSON med variabler og tilhørende datatype.

6.6 PumpType

Ressursen PumpType tjenes fra */api/pump-types* og inneholder informasjon om hvilken type pumpe en mekanisk avluser bruker.

```
{
	"id": INTEGER,
	"name": STRING
}
```

Figur 6.6: Ressursen PumpType representert som JSON med variabler og tilhørende datatype.

6.7 User

En bruker av systemet er representert ved ressursen *User* og kan finnes i endepunktet */api/users*.

```
{
	"id": INTEGER,
	"email": STRING,
	"firstName": STRING,
	"lastName": STRING,
	"isAdmin": BOOLEAN,
	"companyId": INTEGER
}
```

Figur 6.7: Ressursen User representert som JSON med variabler og tilhørende datatype.

6.8 Fil

En binærfil med rådata fra Sensorfisk kan finnes ved hjelp at endepunktet */api/files*. Denne ressursen er ikke representert som et JSON-objekt, siden den lagres som en binærfil på sin opprinnelige form.

6.9 Datatyper

Når det leses ut data fra binærfilene som lastes opp i systemet, lagres kun de relevante verdiene fra målingene. Dette gjøres for å unngå unødvendig lagring i databasen. Ressursen for hver datatype er å finne under endepunktet */api/measurements/:id/<datatype>*.

6.9.1 MagPosition

MagPosition brukes for å beskrive ved hvilke tidspunkt i en gjennomkjøring de forskjellige magnetbåndene (se kapittel 1) er plassert.

```
{
	"id": INTEGER,
	"timeUS": INTEGER,
	"measurementId": INTEGER
}
```

Figur 6.9.1: Ressursen MagPosition representert som JSON med variabler og tilhørende datatype.

6.9.2 BarData

BarData brukes for å beskrive verdier for trykk og temperatur på et gitt tidspunkt under gjennomkjøringen.

```
{
	"id": INTEGER,
	"timeUS": INTEGER,
	"press": INTEGER,
	"temp": DOUBLE,
	"measurementId: INTEGER
}
```

Figur 6.9.2: Ressursen BarData representert som JSON med variabler og tilhørende datatype.

6.9.3 GForceData

GForceData brukes for å beskrive en verdi for G-krefter på et gitt tidspunkt under gjennomkjøringen.

```
{
	"id": INTEGER,
	"timeUS": INTEGER,
	"gForce": DOUBLE,
	"measurementId": INTEGER
}
```

Figur 6.9.3: Ressursen GForceData representert som JSON med variabler og tilhørende datatype.

7 Sikkerhet

Siden systemet som har blitt utviklet er en web-applikasjon, vil alle personer med internettilgang i utgangspunktet kunne aksessere det. Dette åpner opp for en del sikkerhetstrusler. I dette kapittelet beskrives sikkerhetstiltakene som har blitt implementert i systemet for å beskytte mot angripere og uvedkommende.

7.1 Digitale sertifikater og kryptert kommunikasjon

For å få til sikker overføring av data mellom klient og tjener brukes HTTPS, som er en utvidelse av HTTP-protokollen. HTTPS krypterer data som overføres over internett ved hjelp av TLS (Transport Layer Security) eller SSL (Secure Sockets Layer). Dette gjør at uvedkommende ikke kan fange opp sensitiv informasjon som sendes mellom klient og tjener, som for eksempel passord. Til krypteringen brukes et digitalt SSL/TLS sertifikat som installeres på tjeneren og assosieres med domenenavnet (Rodriguez, 2018, s. 131).

I web-applikasjonen har HTTPS blitt implementert med et selvsignert sertifikat. Dette vil si at sertifikatet som benyttes ikke har blitt signert av en sertifikatsutsteder. Et slikt sertifikat krever ikke at web-applikasjonen har et offisielt domenenavn, og kan i tillegg genereres helt gratis. Et selvsignert sertifikat fungerer godt for å få kryptert kommunikasjon under utvikling og testing. Sertifikatet er derimot ikke sikkert nok til å brukes i et ferdig produkt, da det blant annet er sårbart for Man in the middle-angrep. Vi anbefaler derfor at dette sertifikatet byttes ut med et sertifikat signert av en sertifikatsutsteder før lansering.

7.2 Hashing, overføring og lagring av passord

Alle passord som lagres i systemet blir hashet for å beskytte mot eventuelle angrep eller kompromittering. Hashingen foregår på serversiden ved registrering av en bruker. For å beskytte mot Lookup Tables og Rainbow Tables legges det til et salt når passordet hashes. Dette er en tilfeldig generert tekststreng som er ulik for hvert passord, noe som gjør at hver hash blir unik selv om passordene som sendes inn er like.

Ifølge Defuse Security (2019) bør det brukes en godt designet nøkkelstrekkings-algoritme til passord-hashing, som for eksempel PBKDF2, bcrypt eller scrypt. I vårt system brukes bcrypt, som er en passord-hashing algoritme som baserer seg på Blowfish chifferet. Bcrypt er en adaptiv algoritme, og har en kostfaktor som kan økes for at beregningen skal ta lengre tid og Brute Force-angrep skal bli vanskeligere å gjennomføre (Provos & Mazieres, 1999). Denne

kostfaktoren er satt til 14 i vårt system. Ved å teste ulike verdier har vi kommet frem til at dette er et godt kompromiss mellom sikkerhet og ytelse.

Etter at passordet har blitt hashet lagres det i "User"-tabellen i databasen sammen med annen informasjon om brukeren. Tekststrengen som lagres inneholder både hashet passord og salt. Ved pålogging hentes hashet passord ut fra databasen, og det innsendte passordet valideres opp mot hashet passord ved hjelp av en funksjon innebygget i bcrypt-biblioteket.

Overføring av passord fra klient til tjener skjer over HTTPS (se kapittel 7.1). Siden innholdet som sendes er kryptert kan ikke uvedkommende lytte til kommunikasjonen og få tak i passordet.

7.3 Bruk av access token og refresh token

For å holde brukere innlogget og sørge for at de ikke må autentisere seg på nytt hele tiden, bruker vi access token og refresh token:

- **Access Token:** Sendes med alle forespørsler fra klient til tjener for å vise at brukeren er autentisert. Denne har kort levetid.
- **Refresh Token:** Brukes når access token har utløpt for å få tak i en ny token. Sendes til et eget endepunkt som har ansvar for å fornye tokens. Denne har lang levetid.

Vi har brukt JWT (JSON Web Token) for å opprette både access token og refresh token. JWT er en åpen standard som definerer en kompakt måte å utveksle informasjon som JSON-objekter (Auth0, u.å.). Informasjonen er signert, slik at man kan verifisere at den ikke har blitt endret etter at den ble laget.

En JWT består av 3 deler: et tokenhode, en nyttelast og en signatur. I nyttelasten kan det lagres informasjon som kan være nyttig ved autentisering. I vårt system lagres variablene "userId", "isAdmin" og "companyId" i access token, mens det i refresh token kun lagres "userId". En JWT har også en variabel som forteller hvor lenge den skal være gyldig. Denne blir satt til 10 minutter for access token, og 1 uke for refresh token.

Både access token og refresh token lagres i cookies. Dette blir gjort for å unngå faren for XXS ved lagring i for eksempel "localStorage". Cookies åpner imidlertid opp for CSRF-angrep, men dette kan unngås ved å sette de riktige cookie-flaggene. Følgende flagg er satt for cookies som inneholder access token og refresh token:

Tabell 7.3: Flagg som settes for cookies som inneholder access token og refresh token

Flagg	**Forklaring**
httpOnly	Gjør at klienten ikke kan lese innholdet i cookien. Dette beskytter blant annet mot XXS.
secure	Gjør at cookien bare kan overføres over en sikker forbindelse (HTTPS). Beskytter mot "Man In The Middle"-angrep.
SameSite	Denne settes til "strict". Dette gjør at cookien ikke vil bli sendt med forespørsler fra andre domener. Dette beskytter mot CSRF-angrep.

Refresh token lagres også i databasen, og knyttes opp mot en bruker. På denne måten har man muligheten til å fjerne autentisering ved behov, for eksempel hvis refresh token har havnet i gale hender eller brukeren har blitt slettet fra systemet. Før refresh token lagres i databasen hashes den med SHA256.

7.4 Beskyttelse mot SQL-injection og XSS

For å beskytte mot SQL-injection så har vi brukt en ORM (Object Relational Mapper) for å gjøre spørringer til databasen. Ved å bruke en ORM så unngår vi å måtte skrive egne SQL spørringer og bruker heller funksjoner fra ORMet som genererer SQL spørringer (Grünwaldt, J. M., 2019, s.3). Vi har valgt å bruke et ORM som heter Sequelize i denne applikasjonen og vi har fulgt med på sikkerhetsoppdateringer fra blant annet Snyk om forskjellige sårbarheter i Sequelize (Snyk, 2020).

Klientsiden utvikles ved hjelp av React, og det tar seg av XSS (cross-site scripting). React sier selv; "React is safe. We are not generating HTML strings so XSS protection is the default." (Facebook Inc., 2014). Det er derimot noen tilfeller der React ikke klarer å beskytte mot XSS. Disse tilfellene oppstår som oftest hvis man tillater data gitt fra bruker til å endre på komponenter, men kan også for eksempel oppstå hvis man bruker "dangerouslySetInnerHTML" i en komponent (Mueller, 2017). I denne applikasjonen har vi passet på å unngå de tilfellene vi har funnet som kan åpne for XSS angrep.

8 Installasjon og kjøring

Dette kapitlet beskriver hvordan systemet installeres og kjøres, og hvilke avhengigheter som må på plass før systemet kan settes ut i produksjon.

8.1 Eksterne avhengigheter

Dette kapitlet beskriver de eksterne avhengighetene som må være på plass for at systemet kan tas i bruk.

8.1.1 Tjener

For at systemet skal kunne settes ut i produksjon, må systemet kjøre på en tjener. Tjeneren må oppfylle et minimumskrav av maskinvarespesifikasjoner for at systemet skal kunne fungere skikkelig. Vi kommer i dette kapitlet med en anbefaling på hvilke spesifikasjoner tjeneren bør ha. Systemet kan også kjøre på en tjener i skyen, eksempelvis ved bruk av Microsoft Azure.

Prosessor

Tjeneren anbefales å ha en 4-kjerners prosessor med en klokkehastighet på minimum 2,0GHz.

Minne

Tjeneren bør ha minimum 4GB DDR3 eller DDR4 RAM med en hastighet på minst 2400MHz, men anbefales å ha 8GB. Dette kan enkelt byttes ut dersom tjeneren krever mer senere.

Lagring

I systemet lagres både data fra binærfilene i databasen, samt selve binærfilene. Lagringen av binærfilene vil være det som tar mest plass i systemet. Dersom vi tar utgangspunkt i at en binærfil er 100MB, går det ~10000 binærfiler på 1TB. Vi anbefaler derfor at tjeneren har minst 1TB SSD eller HDD lagring, med mulighet for å ekspandere ved behov.

Nettverk

Tjeneren anbefales å støtte gigabit nettverking, slik at systemet ikke opplever en flaskehals på nettverksforbindelsen.

Operativsystem

Systemet kan kjøre både på Windows, Linux og MacOS. Hvert operativsystem har sine fordeler og ulemper, så her går det i all hovedsak på preferanse. Installasjonsguiden i kapittel 8.2 er skrevet for Windows og Linux.

Hot swapping og redundans

Elementer som bør tas i betraktning ved oppsett av en tjener er muligheten for såkalt "hot swapping" og å ha redundans. Hot swapping er et begrep som brukes om muligheten for å legge til eller bytte ut en maskinvarekomponent uten å skru av hele systemet. For at systemet skal ha minst mulig nedetid bør muligheten for hot swapping vurderes. Dersom en komponent feiler, bør tjeneren ha redundans, noe som vil si at en annen komponent tar over for komponenten som feiler. Dette er mest vesentlig at blir implementert på komponentene som tar seg av lagring, slik at data ikke går tapt.

Backup

Tjeneren bør ha en såkalt offsite backup-tjener som regelmessig tar sikkerhetskopier av innholdet på tjeneren. Backup-tjeneren bør ligge på en annen lokasjon enn tjeneren som kjører systemet, i tilfelle brann, vannlekkasjer eller andre trusler.

8.1.2 Andre avhengigheter

8.1.2.1 Node.js

På forhånd må systemet ha Node.js installert. Dette avsnittet beskriver hvordan Node.js installeres på henholdsvis Windows og Linux.

Windows

Node.js installeres ved å følge linken: https://nodejs.org/en/download/

Arch-baserte Linuxdistribusjoner

```
$ sudo pacman -S nodejs
```

Debian-baserte Linuxdistribusjoner

```
$ sudo apt-get install nodejs
```

8.1.2.2 Git

Dette avsnittet beskriver hvordan Git installeres på henholdsvis Windows og Linux.

Windows

Git-bash installeres ved å følge linken: https://git-scm.com/downloads

Arch-baserte Linuxdistribusjoner

```
$ sudo pacman -S git
```

Debian-baserte Linuxdistribusjoner

```
$ sudo apt-get install git
```

8.2 Installasjon

Dette kapitlet beskriver hvordan prosjektet installeres og brukes på lokal maskin eller på en tjener.

8.2.1 Klone prosjektet

Prosjektet klones ved følgende kommando:

```
$ git clone https://github.com/SanderNico/Bacheloroppgave-061
```

8.2.2 Sette opp MySQL / MariaDB

8.2.2.1 Installere MySQL / MariaDB

Windows

Last ned og installer MySQL-server ved å følge denne lenken:

https://dev.mysql.com/downloads/.

Arch-baserte Linuxdistribusjoner

```
$ sudo pacman -S mariadb
```

```
$ sudo mysql_install_db --user=mysql --basedir=/usr --datadir=/var/lib/mysql
```

- MySQL vil sette gjeldende bruker som bruker dersom annet ikke er eksplisitt spesifisert med `--user`-alternativet.
- `--basedir` setter installasjonsmappen.
- `--datadir` setter mappen hvor dataen lagres.

Debian-baserte Linuxdistribusjoner

```
$ apt-get install mysql-server
```

8.2.2.2 Kjøre MySQL / MariaDB

Windows

Som standard starte MySQL-serveren opp ved oppstart og kjører i bakgrunnen. Hvis du likevel har behov for å starte/stoppe serveren manuelt kan dette gjøres ved å kjøre følgende kommandoer i kommandovinduet.

Start:

```
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqld"
```

Stopp:

```
C:\> "C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqladmin" -u root shutdown
```

Arch-baserte Linuxdistribusjoner

Start:

```
$ sudo systemctl start mariadb
```

Stopp:

```
$ sudo systemctl stop mariadb
```

Debian-baserte Linuxdistribusjoner

Start:

```
$ sudo systemctl start mysqld
```

Stopp:

```
$ sudo systemctl stop mysqld
```

8.2.2.3 Bruke MySQL / MariaDB

Windows

Åpne programmet “MySQL 8.0 Command Line Client” og logg inn med passordet som ble definert under installasjon.

Linux

```
$ sudo mysql -u root -p
```

- Brukernavnet er spesifisert med `-u` fulgt av brukernavnet som er *root* som standard.
- Passordet er spesifisert med `-p`-alternativet. En ny linje vil dukke opp hvor passordet må skrives inn.

Opprette nødvendige databaser

```
> CREATE DATABASE db_dev;
```

```
> CREATE DATABASE db_test;
```

```
> CREATE DATABASE db_prod;
```

8.2.3 Generere nye SSL-sertifikater

Dette trenger man kun å gjøre dersom man skal fortsette å bruke selv-signerte SSL-sertifikater og vil generere nye.

8.2.3.1 Installere openssl

Windows

Last ned og installer openssl ved å følge lenken: https://slproweb.com/products/Win32OpenSSL.html. Åpne programmet "Win64 OpenSSL Command Prompt" og bruk dette til generering av nøkkel og sertifikat.

Arch-baserte Linuxdistribusjoner

```
$ sudo pacman -S openssl
```

Debian-baserte Linuxdistribusjoner

```
$ sudo apt-get install openssl
```

8.2.3.2 Generering av nøkkel og sertifikat

Naviger til *security*-mappen

```
$ cd server/src/security
```

HTTPS konfigurasjonsfil

I mappen ligger en konfigurasjonsfil med navnet *req.cnf*.

```
[req]
distinguished_name = req_distinguished_name
x509_extensions = v3_req
prompt = no
[req_distinguished_name]
C =
ST =
L =
O =
OU =
CN =
[v3_req]
keyUsage = critical, digitalSignature, keyAgreement
extendedKeyUsage = serverAuth
subjectAltName = @alt_names
[alt_names]
DNS.1 =
DNS.2 =
DNS.3 =
```

Følgende felter må fylles ut i filen.

- C: Landkode: Dersom Norge - NO.
- ST: Fylke.
- L: Lokasjon.
- O: Organisasjon.
- OU: Enhet i organisasjonen
- CN: “Common name” - www.eksempel.com

- DNS.1: www.eksempel.com
- DNS.2: eksempel.com
- DNS.3: eksempel

Generere nøkkel og sertifikat

```
$ openssl req -x509 -nodes -days 365 -newkey rsa:2048 keyout- cert.key -out
cert.pem -config req.cnf -sha256
```

8.2.4 Opprette .env filer

Klient

```
$ touch /server/.env
```

Åpne så filen i ønsket teksteditor og lim inn følgende:

```
SKIP_PREFLIGHT_CHECK=true

# HTTPS settings
HTTPS=true
SSL_CRT_FILE=../server/src/security/cert.pem
SSL_KEY_FILE=../server/src/security/cert.key

# Domain name
REACT_APP_SERVER_DOMAIN=
```

Tjener

```
$ touch /server/.env
```

Åpne så filen i ønsket teksteditor og lim inn følgende:

```
# Database settings
DB_USER=
DB_PASS=
DB_HOST=

# Different databases
DB_DEV=
DB_TEST=
DB_PROD=

# HTTPS settings
HTTPS=true
SSL_CRT_FILE=./server/src/security/cert.pem
SSL_KEY_FILE=./server/src/security/cert.key
HTTPPORT=
HTTPSPORT=

# Token secrets
ACCESS_TOKEN_SECRET=
REFRESH_TOKEN_SECRET=
LINK_TOKEN_SECRET=

SKIP_PREFLIGHT_CHECK=true

# Email settings for testing
EMAIL_DEV_HOST=
EMAIL_DEV_PORT=
EMAIL_DEV_USER=
EMAIL_DEV_PASS=

# Email settings for testing
EMAIL_TEST_HOST=
EMAIL_TEST_PORT=
EMAIL_TEST_USER=
EMAIL_TEST_PASS=
```

```
# Email settings for production
EMAIL_PROD_HOST=
EMAIL_PROD_PORT=
EMAIL_PROD_USER=
EMAIL_PROD_PASS=

# Domain name
PROD_DOMAIN=
DEV_DOMAIN=
```

Fyll deretter inn de tomme feltene

- HTTPPORT og HTTPSPORT settes til hhv 80 og 443 dersom de ikke fylles ut.
- En epostkonto må opprettes for å kunne bruke eposttjenesten i systemet.
- Token secrets må genereres på nytt (se kapittel 8.2.4.1).
- Prosjektet bruker et selv-signert SSL-sertifikat som nå ligger i */server/src/security*.

8.2.4.1 Generere token secrets

Åpne Node.js i terminalen

```
$ node
```

Generer deretter token secrets ved å kjøre følgende kommando tre ganger. Lim inn stringen som genereres av kommandoen (uten apostrofer) på anvist plass i */server/.env*-filen.

```
> require('crypto').randomBytes(256).toString('base64');
```

8.2.5 Installasjon av avhengigheter

```
$ npm install
```

```
$ cd client && npm install
```

```
$ cd server && npm install
```

8.2.6 Kjøring

8.2.6.1 Kjøring i et utviklingsmiljø

```
$ npm start
```

--ELLER--

Kjør klient og tjener i separate terminaler:

```
$ cd client && npm start
```

```
$ cd server && npm start
```

8.2.6.2 Klargjøring for produksjon

Bygge prosjektet

```
$ npm run build
```

Installere serve globalt

```
$ sudo npm install -g serve
```

Kjøre tjeneren i produksjon

```
$ npm run production
```

Kjøre klienten i produksjon

```
$ npm run serve
```

9 Dokumentasjon av kildekode

I prosjektet er det brukt JSDoc 3 som er et API for dokumentasjon av kildekode i JavaScript. Dette gjøres ved at JSDoc installeres i prosjektet ved hjelp av Node Package Manager. Videre oppretter man en konfigurasjonsfil i prosjektet som forteller API-et i hvilke filer som skal sjekkes. Det settes opp et script i *package.json* filen til prosjektet slik at man kan generere dokumentasjonen ved hjelp av en enkel kommando (se kapittel 9.1). JSDoc scanner da kildekoden og genererer HTML-filer slik at man kan opprette en nettside over dokumentasjonen (JSDoc 3, 2017). Videre brukes GitHub Pages for å tjene nettsiden som viser dokumentasjonen av kildekoden.

9.1 Generering av dokumentasjon

For å generere dokumentasjonen med JSDoc kjøres følgende kommando fra kommandovinduet (i prosjektmappa):

```
npm run createdoc
```

9.2 Lenke til dokumentasjon

Etter ønske fra oppgavestiller publiseres ikke dokumentasjonen av kildekode. Dersom dokumentasjon av kildekode ønskes, bes leser ta kontakt med SINTEF Ocean AS.

10 Kontinuerlig integrasjon og testing

Kontinuerlig integrasjon (CI) og testing er viktig i et systemutviklingsprosjekt for å sikre kvaliteten til koden som utvikles. Dette gjøres ved at testene skrives samtidig som kildekoden, og at de kjøres ved hjelp av kontinuerlig integrasjon når de lagres i versjonskontrollsystemet.

10.1 Kontinuerlig integrasjon

I prosjektet har vi benyttet oss av GitHub Actions. Dette er en tjeneste hvor Github tilbyr virtuelle maskiner hvor kontinuerlig integrasjon blir kjørt når kode pushes til et gitt repository. De virtuelle maskinene i GitHub Actions tilbys via Microsoft Azure skytjenester, og man kan velge mellom å kjøre den kontinuerlige integrasjonen på Linux, Windows og MacOS. Disse virtuelle maskinene har følgende maskinvare tilgjengelig (Github Inc, 2020):

- 2-kjerners prosessor
- 7 GB RAM
- 14 GB SSD

I vår implementasjon av kontinuerlig integrasjon har vi valgt å bruke den seneste versjonen av Ubuntu (i skrivende stund 18.04).

```
name: Node.js CI

on:
  push:
    branches: [ master, develop ]
  pull_request:
    branches: [ master, develop ]

jobs:
  build:

    runs-on: ubuntu-latest

    strategy:
      matrix:
        node-version: [12.x]
```

```
    services:
      mysql:
        image: mysql:5.7
        env:
          MYSQL_USER: root
          MYSQL_DATABASE: db_test
          MYSQL_ROOT_PASSWORD: password
        ports:
          - 32574:3306
        options: --health-cmd="mysqladmin ping" --health-interval=10s --health-
timeout=5s --health-retries=3

    steps:

    - name: Verify MySQL connection from host
      run: |
        sudo apt-get install -y mysql-client
        mysql --host 127.0.0.1 --port 32574 -uroot -ppassword -e  "SHOW
DATABASES"

    - uses: actions/checkout@v2
    - name: Use Node.js ${{ matrix.node-version }}
      uses: actions/setup-node@v1
      with:
        node-version: ${{ matrix.node-version }}
    - run: npm ci
    - run: npm run build --if-present
    - run: cd server && npm install && npm test
      env:
        CI: true
    - run: cd client && npm install && npm test
```

Figur 10.1: Vår versjon av nodejs.yml. Kontinuerlig integrasjon skjer på master- og develop-branch, bruker nyeste versjon av Ubuntu og setter opp en MySQL-database. Deretter kjøres npm install for å installere alle avhengigheter fra Node Package Manager, og tester kjøres både på klient- og tjenerside.

Som vi ser av nodejs.yml-filen (se figur 10.1) som er filen som bestemmer hvordan CI i prosjektet er satt opp velges det først at CI skal kjøres når det pushes til master- eller develop-branchen. Videre settes det at den virtuelle maskinen som skal brukes kjører seneste versjon av Ubuntu, nyeste versjon av Node og MySQL versjon 5.7.
Variablene for tilkobling til databasen settes opp, og deretter installeres avhengigheter fra Node Package Manager som brukes i prosjektet. Til slutt kjøres testene både på klient- og tjenersiden.

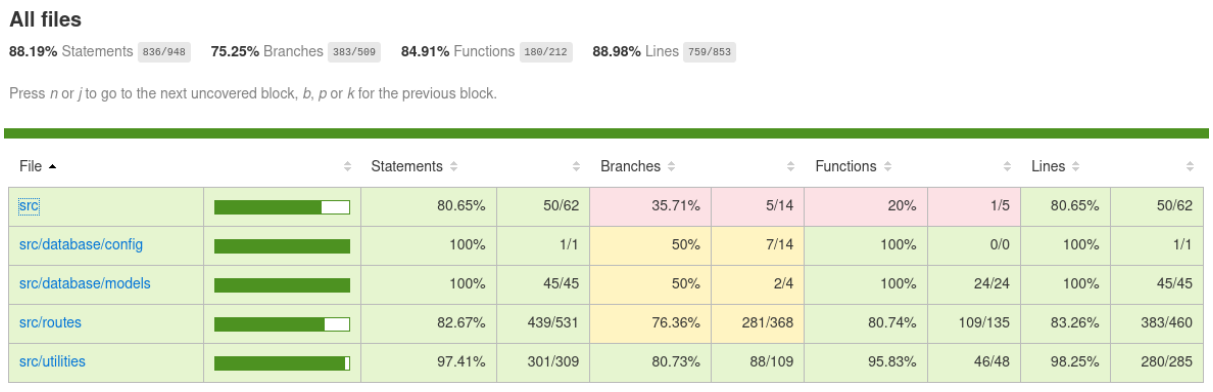
10.2 Tester

Testene som er brukt i prosjektet er i all hovedsak enhetstester, og har blitt skrevet kontinuerlig i løpet av utviklingsprosessen. Ved bruk av enhetstester tar man en enkel funksjonalitet av koden (en enhet) og skriver en test på denne. Dersom det gjøres en endring i koden, kjøres testene på nytt og man får verifisert at funksjonen fortsatt fungerer som tiltenkt. Fordelen med å bruke enhetstester under utviklingen av et system er at man raskt kan fange opp hva som gjør at koden feiler (Software Testing Fundamentals, 2019).

Tester kjøres ved følgende kommandoer (fra prosjektmappa):

```
// Går inn i mappa til tjeneren og kjører testene
cd server && npm test
// Går tilbake til prosjektmappa
cd ..
// Går inn i mappa til klienten og kjører testene
cd client && npm test
```

Figur 10.2.1: Instruksjoner på hvordan man kjører testene.



File		Statements		Branches		Functions		Lines	
src		80.65%	50/62	35.71%	5/14	20%	1/5	80.65%	50/62
src/database/config		100%	1/1	50%	7/14	100%	0/0	100%	1/1
src/database/models		100%	45/45	50%	2/4	100%	24/24	100%	45/45
src/routes		82.67%	439/531	76.36%	281/368	80.74%	109/135	83.26%	383/460
src/utilities		97.41%	301/309	80.73%	88/109	95.83%	46/48	98.25%	280/285

Figur 10.2.2: Figuren viser dekningen av tester i de forskjellige mappene på tjenersiden.

Ovenfor (se figur 10.2.2) ser vi en oversikt over dekningen som testene som er skrevet på tjenersiden. Dette vil si hvor mye av kildekoden som er dekket av testene. I kolonnen *File* ser vi i hvilke mapper på tjenersiden de forskjellige filene ligger. I dette prosjektet er det mest hensiktsmessig å se på filer som ligger i mappene *src/database/models, src/database/models/datamodels, src/routes* og *src/utilites* da disse ikke er "standardfiler" for et NodeJS-prosjekt.

- I *src/database/models* og *src/database/models/dataModels* ligger modellene for hvordan de forskjellige REST-ressursene (se kapittel 6) er lagret i databasen.
- *Src/routes* inneholder de forskjellige tjener-endepunktene i API-et.
- Filene i *src/utilites* tar seg av omgjøring fra binærfil til forståelig data, og gjør videre utregninger på denne dataen før den legges inn i databasen.

Kolonnen *Statements* viser dekningsprosenten av deklarasjoner, *Branches* viser om alle variable utførelser av koden er dekket (altså alle "hvis dette, gjør dét") og *Functions* viser dekningen av alle funksjoner.
Lines viser den totale dekningen av alle eksekverbare linjer i koden.

All files

70.8% Statements 308/435 **53.16%** Branches 42/79 **73.08%** Functions 76/104 **69.69%** Lines 292/419

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

File		Statements		Branches		Functions		Lines	
models		84.48%	49/58	100%	0/0	83.33%	10/12	84.48%	49/58
services		64.1%	50/78	0%	0/12	72.73%	40/55	64.1%	50/78
utilities		69.9%	209/299	62.69%	42/67	70.27%	26/37	68.2%	193/283

Figur 10.2.3: Figuren viser dekningen av tester i de forskjellige mappene på klientsiden.

Figuren ovenfor (se figur 10.2.3) viser en oversikt over testdekningen av de forskjellige mappene som er hensiktsmessige på klientsiden. *Models* er objektklassene til de forskjellige REST-ressursene (se kapittel 6), mens *services* er klassene som håndterer HTTP-forespørslene til tjeneren. Mappa *utilities* inneholder klassene som tar seg av datahåndtering på klientsiden.

Referanser

- Auth0. (u.å.). Get Started with JSON Web Tokens. Hentet 16. april 2020 fra https://auth0.com/learn/json-web-tokens/
- Caharija, W., Venås, B., Svendsen, E., Schrøder, M. B., Pedersen, M. O., Lie, O. K., … Ohrem, S. J. (2019). Verktøy for kartlegging av forhold i enheter for føring, håndtering og behandling av laks (Sensorfisk) (SINTEF rapport L4). Hentet 14. januar 2020 fra https://www.fhf.no/prosjekter/prosjektbasen/901397/?fileurl=https://fhfno.sharepoint.com/sites/pdb/Publisertedokumenter/307829FHprnr901397_KVALYSIS_faglig_sluttrapport_v3.pdf.PDF&filename=Sluttrapport:%20Verkt%C3%B8y%20for%20kartlegging%20av%20forhold%20i%20enheter%20for%20f%C3%B8ring,%20h%C3%A5ndtering%20og%20behandling%20av%20laks%20(sensorfisk)
- Defuse Security. (2019, 5. juni). Salted Password Hashing - Doing it Right. Hentet 16. april 2020 fra https://crackstation.net/hashing-security.htm
- Github Inc. (2020). *Virtual environment for GitHub-hosted runners.* Hentet 02. april 2020 fra https://help.github.com/en/actions/reference/virtual-environments-for-github-hosted-runners
- Grünwaldt, J. M. (2019). *A Comparison of Modern Backend Frameworks Protections against Common Web Vulnerabilities.* Tufts University, Medford MA.
- Facebook Inc. (2014). Tutorial. Hentet 02. april 2020 fra https://shripadk.github.io/react/docs/tutorial.html
- Flatsetsund Engineering. (u.å.). *FLS Avluser: For avlusing av laks.* Hentet 18. februar 2020 fra https://www.fls.no/avlusing-av-laks/
- JSDoc 3. (2017). *Getting started with JSDoc 3*. Hentet 17. april 2020 fra https://jsdoc.app/about-getting-started.html
- Provos, N. & Mazieres, D. (1999, 28. april). A Future-Adaptable Password Scheme. Hentet 16. april 2020 fra https://www.usenix.org/legacy/events/usenix99/provos/provos_html/node1.html
- Richardson, L. & Ruby, S. (2007). *RESTful Web Services.* Sebastopol, CA: O'Reilly Media Inc.
- Rodriguez, M. (2018). HTTPS Everywhere: Industry Trends and the Need for Encryption. *Serials Review, 44*(2), 131-137. https://doi.org/10.1080/00987913.2018.1472478

- SINTEF. (2017). *Standardisert metodikk for kvalifisering av mekaniske avlusingssystemer (KVALISYS)*. Hentet 19. februar 2020 fra https://www.sintef.no/prosjekter/standardisert-metodikk-for-kvalifisering-av-mekaniske-avlusingssystemer/
- Snyk. (2020, 23. januar). *Sequelize Vulnerabilities*. Hentet 17. april 2020 fra https://snyk.io/vuln/npm:sequelize
- Software Testing Fundamentals. (2019). *Unit testing*. Hentet 08. april 2020 fra http://softwaretestingfundamentals.com/unit-testing/
- Mueller, B. (2017, 2. august). Exploiting Script Injection Flaws in ReactJS Apps. Hentet 02. april 2020 fra https://medium.com/dailyjs/exploiting-script-injection-flaws-in-reactjs-883fb1fe36c1

Utvikling av brukergrensesnitt og brukerfunksjoner for Sensorfisk

Prosjekthåndbok

Vedlegg E

Forfattere:

Herman Ryen Martinsen

Sander Nicolausson

Trond Jacob Rondestvedt

Jørgen Aasvestad

19.05.2020

Innholdsfortegnelse

1 Framdriftsplan – Gannt-diagram

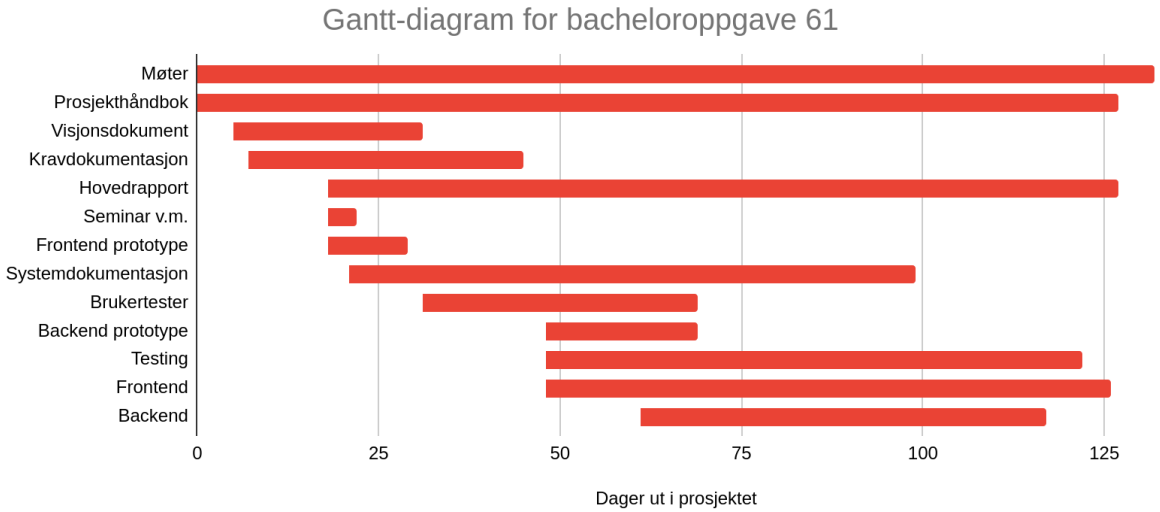
Gantt-diagram blir brukt i prosjektet for å illustrere tidsplanen som er lagt. Da vi har jobbet iterativt med scrum endres krav underveis, noe som krever at Gantt-diagrammet også oppdateres. Prosjektperioden regnes fra oppstartsdato 9. januar til innleveringsdato 20.mai.

1.1 Første oppsett - 14.01.2020

Første oppsett av fremdriftsplanen ble gjort dagen etter oppstartsmøte mellom bachelorstudenter, veileder og oppgavestiller. Det ble tatt utgangspunkt i at vi startet prosjektperioden 9. januar da møteinnkallingen til oppstartsmøtet ble skrevet.

Tabell 1.1: Oversikten over prosjektperioden slik den ble satt opp i starten av prosjektet. Hver aktivitet har planlagt start- og sluttdato, et anslag på antall timer og en ansvarlig person. De to siste kolonnene trengs for å lage selve Gantt-diagrammet (Figur 1.1).

Aktivitet	**Planlagt start**	**Planlagt slutt**	**Antall timer**	**Ansvarlig**	**Relativ startdag**	**Lengde (dager)**
Møter	9. januar	20. mai	40	Trond	0	132
Prosjekthåndbok	9. januar	15. mai	60	Sander	0	127
Visjonsdokument	14. januar	9. februar	40	Herman	5	26
Kravdokumentasjon	16. januar	23. februar	100	Herman	7	38
Hovedrapport	27. januar	15. mai	400	Sander	18	109
Seminar v.m.	27. januar	31. januar	40	Alle	18	4
Frontend prototype	27. januar	7. februar	50	Jørgen	18	11
Systemdokumentasjon	30. januar	17. april	150	Sander	21	78
Brukertester	10. februar	19. mars	20	Jørgen	31	38
Backend prototype	17. februar	9. mars	50	Trond	48	21
Testing	17. februar	1. mai	150	Herman	48	74
Frontend	17. februar	5 mai	500	Jørgen	48	78
Backend	10. mars	5. mai	400	Trond	61	56
SUM			**2000**			



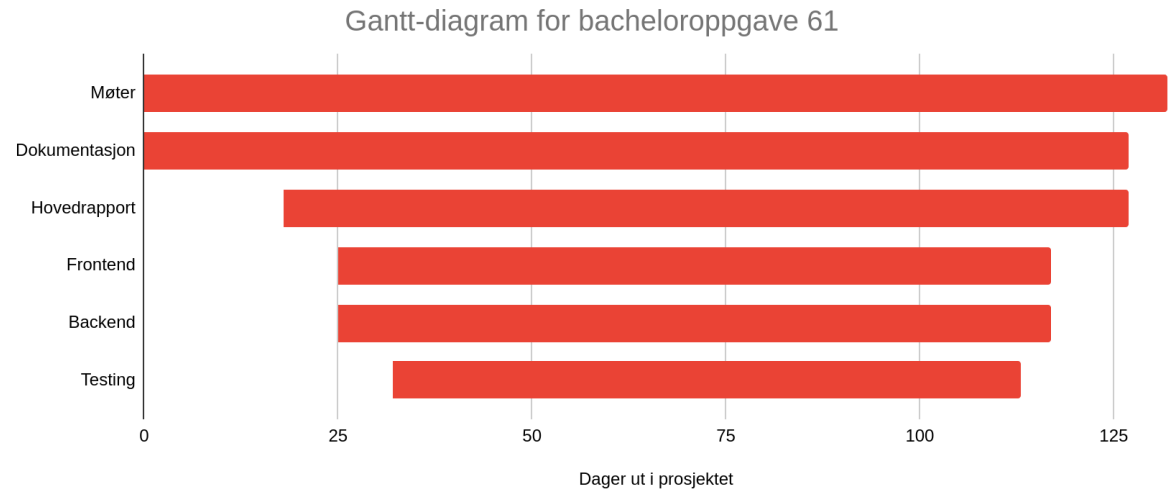
Figur 1.1: Gantt-diagrammet som viser framdrift gjennom prosjektperioden med utgangspunkt i Tabell 1.1.

1.2 Revidert versjon - 31.01.2020

Revisjonen av fremdriftsplanen ble gjort etter første veiledningsmøte, da veileder ønsket å ha mer generaliserte aktiviteter. Dette ble ønsket da det ville forbedre oversikt og forenkle føring av timer basert på aktiviteter.

Tabell 1.2: Oversikten over prosjektperioden etter første revisjon av Gantt-diagrammet. Etter tilbakemelding fra veileder har aktivitetene blitt mer generalisert. Hver aktivitet har planlagt start- og sluttdato, et anslag på antall timer og ansvarlig person. De to siste kolonnene trengs for å lage selve Gantt-diagrammet (Figur 1.2).

Aktivitet	**Planlagt start**	**Planlagt slutt**	**Antall timer**	**Ansvarlig**	**Relativ startdag**	**Lengde**
Møter	9. januar	20. mai	60	Trond	0	132
Dokumentasjon	9. januar	15. mai	350		0	127
Hovedrapport	27. januar	15. mai	440	Sander	18	109
Frontend	3. februar	5 mai	530	Jørgen	25	92
Backend	3. februar	5. mai	450	Trond	25	92
Testing	10. februar	1. mai	170	Herman	32	81
SUM			**2000**			



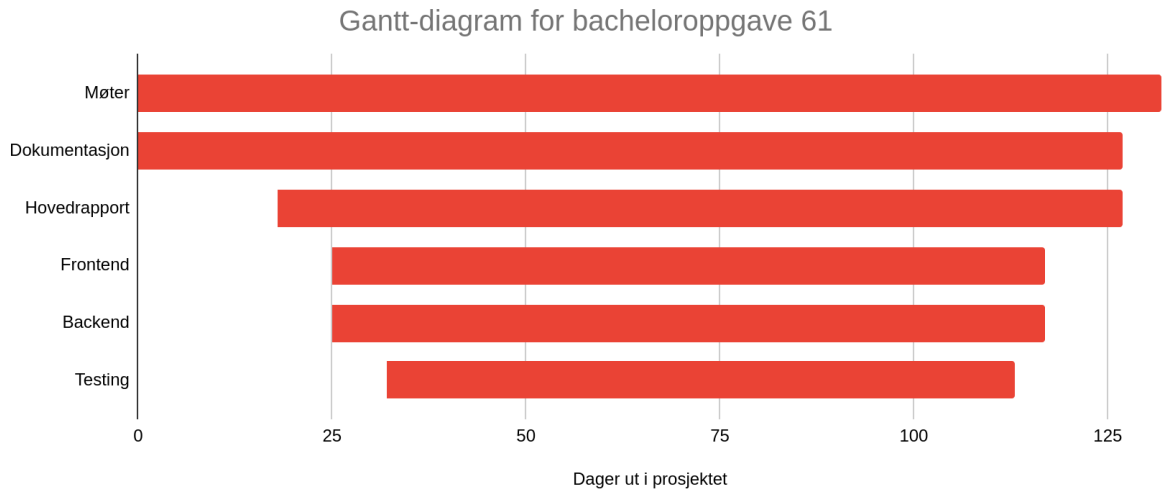
Figur 1.2: Gantt-diagrammet som viser framdrift gjennom prosjektperioden med utgangspunkt i Tabell 1.2.

1.3 Revidert versjon - 02.03.2020

Andre revisjon av fremdriftsplanen ble gjort på bakgrunn av at antall timer som var satt av til aktiviteten *møter* ble overskredet. Det så også ut som at vi behøvde noe mer tid enn originalt avsatt til hovedrapport og dokumentasjon, og noe mindre til frontend, backend og testing.

Tabell 1.3: Oversikten over prosjektperioden etter andre revisjon av Gantt-diagrammet. Vi hadde overskredet antall timer som var satt av til møter, og endret derfor verdiene i denne kolonnen. Hver aktivitet har planlagt start- og sluttdato, et anslag på antall timer og ansvarlig person. De to siste kolonnene trengs for å lage selve Gantt-diagrammet (Figur 1.3).

Aktivitet	**Planlagt start**	**Planlagt slutt**	**Antall timer**	**Ansvarlig**	**Relativ startdag**	**Lengde**
Møter	9. januar	20. mai	100	Trond	0	132
Dokumentasjon	9. januar	15. mai	400		0	127
Hovedrapport	27. januar	15. mai	470	Sander	18	109
Frontend	3. februar	5 mai	500	Jørgen	25	92
Backend	3. februar	5. mai	400	Trond	25	92
Testing	10. februar	1. mai	130	Herman	32	81
SUM			**2000**			



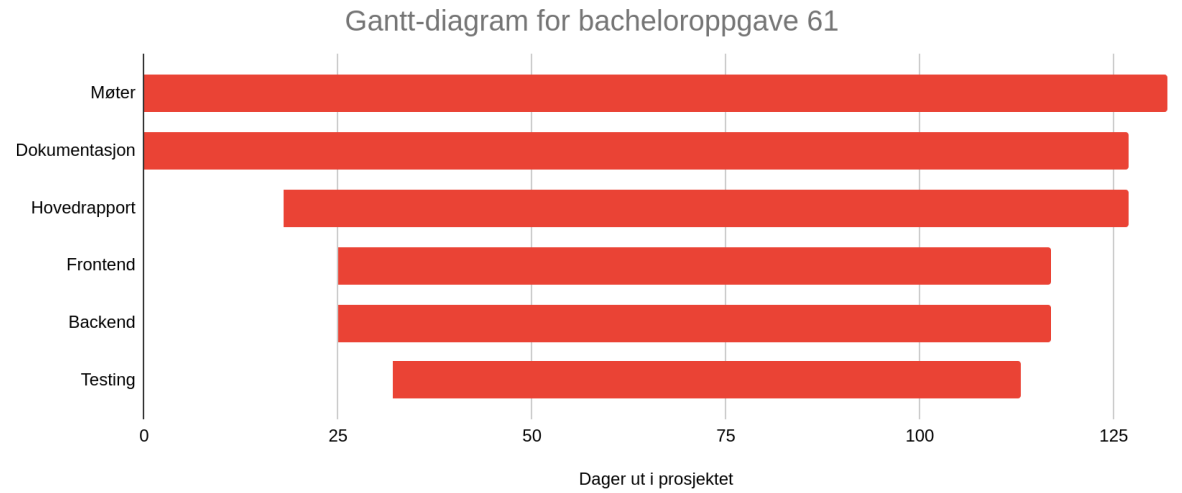
Figur 1.3: Gantt-diagrammet som viser framdrift gjennom prosjektperioden med utgangspunkt i Tabell 1.3.

1.4 Revidert versjon - 01.05.2020

Siste revisjon av framdriftsplanen ble gjort mot slutten av prosjektperioden, med en økning i antall timer på dokumentasjon og hovedrapport, og en reduksjon av backend-utvikling. Dette ble gjort med bakgrunn i at produktet er tilnærmet ferdig, og en god del gjenstår for å få ferdig hovedrapporten og vedleggene til denne.

Tabell 1.4: Oversikten over prosjektperioden etter siste revisjon av Gantt-diagrammet. Vi begynte å nærme oss overskridelse av dokumentasjonen og så at vi hadde estimert for mange timer på backend.

Aktivitet	**Planlagt start**	**Planlagt slutt**	**Antall timer**	**Ansvarlig**	**Relativ startdag**	**Lengde**
Møter	9. januar	20. mai	100	Trond	0	132
Dokumentasjon	9. januar	15. mai	430		0	127
Hovedrapport	27. januar	15. mai	490	Sander	18	109
Frontend	3. februar	5 mai	500	Jørgen	25	92
Backend	3. februar	5. mai	350	Trond	25	92
Testing	10. februar	1. mai	130	Herman	32	81
SUM			**2000**			



Figur 1.4: Gantt-diagrammet som viser framdrift gjennom prosjektperioden med utgangspunkt i Tabell 1.4.

2 Møteinnkallinger med referat

2.1 Oppstartsmøte

Innkalling til oppstartsmøte: Bacheloroppgave 061

Tidspunkt/sted: Mandag 13.01.2020 kl 14.30 – 15.30, Rom OCE Sealab G. O. Sars, Brattørkaia 17C

Følgende personer innkalles:

Sveinung Johan Ohrem (Oppgavestiller)

Birger Venås (Oppgavestiller)

Jan Harald Nilsen (Veileder)

Trond Jacob Rondestvedt

Jørgen Aasvestad

Herman Ryen Martinsen

Sander Nicolausson

Agenda:

Sak 01/2020: Bli kjent

Sak 02/2020: SINTEF informerer om sensorfisken, prosjektet og hvordan de behandler data i dag

Sak 03/2020: Drøfting av oppgaven og framdrift

Sak 04/2020: Gjennomgang av prosjekthåndboka

Sak 05/2020: Gjennomgang av retningslinjene for vurdering

Sak 06/2020: Bevisstgjøring av ambisjonsnivå

Sak 07/2020: Orientering om NTNU sine krav til arbeid og dokumentasjon

Sak 08/2020: Valg av språk

Sak 09/2020: Avklaring av hvordan dokumenter skal distribueres

Sak 10/2020: Kontraktsignering og orientering om opphavsrettigheter, tilgjengeliggjøring og skolens ansvarsfraskrivelse

Ta kontakt med undertegnede dersom du ikke har anledning til å komme.

Mvh

Trond Jacob Rondestvedt Trondheim 09.01.2020

Referat fra oppstartsmøte: Bacheloroppgave 061

Tidspunkt/sted: Mandag 13.01.2020 kl 14.30 – 15.30, Rom OCE Sealab G. O. Sars, Brattørkaia 17C

Til stede: Sveinung Ohrem (oppgavestiller), Jan Harald Nilsen (veileder), Herman Ryen Martinsen, Trond Jacob Rondestvedt, Jørgen Aasvestad og Sander Nicolausson

Frafall: Birger Venås (oppgavestiller)

Ordstyrer: Herman Ryen Martinsen

Sak 01/2020: Bli kjent

Alle møtedeltakere presenterer seg, hva de heter, hvor de er fra og hva de driver med.

Sak 02/2020: SINTEF informerer om sensorfisken, prosjektet og hvordan de behandler data i dag

Oppdragsgiver presenterer sensorfisken. Hovedpunkter fra dette:

- Prosjektet går ut på å utvikle objektive velferdsindikatorer for laks.
- Mangel i dag: Har ikke et verktøy for å fortelle om laksen har det bra eller ikke
- Målet med sensorfisken: Samle data om trykk og støt som laksen utsettes for i en mekanisk avluser.
- Måten dette gjøres på i dag:
    - Kjører sensorfisk ute hos en bedrift
    - Tar med tilbake til kontoret
    - Kobler til PC, og kjører Python-script som leser inn data, filtrerer og genererer plot.
    - Sender plot tilbake til kunden i form av pdf.
- Ønsker et grafisk brukergrensesnitt til sensorfisk som
    - Leser inn data
    - Behandler data
    - Visualiserer data
    - Gir mulighet til å eksporterer filer
    - Lagre data i programmet
- Ønsket leveranse fra oss:

- GUI-applikasjon
- Brukerveiledning for SINTEF Ocean og sluttbrukere
- Sluttrapport

Sak 03/2020: Drøfting av oppgaven og framdrift

Bestemmer at en web-basert løsning er best, da det kan være mulighet for at man har med PC ut når man kjører sensorfisken gjennom mekanisk avluser. Slipper da å vente til man kommer tilbake til kontoret for å se dataene.
Oppdragsgiver har ingen preferanse når det gjelder teknologi som skal benyttes. Det bestemmes at vi skal ta utgangspunkt i de eksisterende Python-scriptene for å hente ut verdiene til sensorfisken, og at den web-baserte løsningen skal utvikles i JavaScript.
Det fastsettes at møte med oppdragsgiver skal forekomme hver 2. uke. Oppdragsgiver stiller med lokale.

Sak 04/2020: Gjennomgang av prosjekthåndboka

Malen for prosjekthåndboka vises fram til oppdragsgiver.

Sak 05/2020: Gjennomgang av retningslinjer for vurdering

Vurderingskriteriene for bacheloroppgaven gjennomgås. Oppdragsgiver har en grei oversikt over disse, da de tidligere har hatt bachelorstudenter hos dem.

Sak 06/2020: Bevisstgjøring av ambisjonsnivå

Oppdragsgiver lurer på hvilket ambisjonsnivå vi har, og det fastslås at det skal utvikles et produkt som bedriften er fornøyd med.
Det bestemmes også at det web-baserte grensesnittet i første omgang skal lages for intern bruk i bedriften, med mulighet til å tilføre ekstra løsninger slik at kunder av bedriften skal kunne ha sin egen bruker for å gå inn å se data som omhandler dem.

Sak 07/2020: Orientering om NTNU sine krav til arbeid og dokumentasjon

Det orienteres om NTNU sine krav til arbeid og dokumentasjon.

Sak 08/2020: Valg av språk

Oppdragsgiver har ingen spesiell preferanse til språk, så det vedtas at kode skal skrives på engelsk, mens dokumentasjon og hovedrapport skrives på norsk.

Sak 09/2020: Avklaring av hvordan dokumenter skal distribueres

Det bestemmes at github brukes til behandling av kildekode, og at Google drive brukes for å dele dokumenter. Oppdragsgiver og veileder gis tilgang slik at de har mulighet til å følge fremgangen på prosjektet.

Sak 10/2020: Kontraktsignering og orientering om opphavsrettigheter, tilgjengeliggjøring og skolens ansvarsfraskrivelse

Det orienteres om opphavsrettigheter, tilgjengeliggjøring og skolens ansvarsfraskrivelse, og oppdragsgiver, veileder og studenter skriver under NTNUs "Avtale mellom partene". Her bestemmes det at oppdragsgiveren har rettighetene til produktet, og kan utnytte dette kommersielt. Se kontrakt for mer spesifikk informasjon.
Oppdragsgiver informerer om at studentene er pålagt å signere en taushetserklæring, men har ikke denne for hånden. Taushetserklæringen vil signeres ved et senere tidspunkt.

13.01.2020, Sander Nicolausson

2.2 Møte 1

Innkalling til møte 1: Bacheloroppgave 061

Tidspunkt/sted: Mandag 27.01.2020 kl 15:00 – 16.00, Rom OCE Sealab G. O. Sars, Brattørkaia 17C

Følgende personer innkalles:

Sveinung Johan Ohrem (Oppgavestiller)

Birger Venås (Oppgavestiller)

Trond Jacob Rondestvedt

Jørgen Aasvestad

Herman Ryen Martinsen

Sander Nicolausson

Agenda:

Sak 01/2020: Gjennomgang av visjonsdokument

Sak 02/2020: Kort forklaring av scrum

Sak 03/2020: Framvisning av wireframes og domenemodell

Sak 04/2020: Oppklaring av spørsmål

Sak 05/2020: Framdrift mot neste møte

Sak 06/2020: Eventuelt

Ta kontakt med undertegnede dersom du ikke har anledning til å komme

Mvh

Trond Jacob Rondestvedt Trondheim 21.01.2020

Referat fra møte 1: Bacheloroppgave 061

Tidspunkt/sted: Mandag 27.01.2020 kl 15.00 – 16.00, Rom OCE Sealab G. O. Sars, Brattørkaia 17C

Til stede: Sveinung Ohrem (oppgavestiller), Birger Venås (oppgavestiller), Herman Ryen Martinsen, Trond Jacob Rondestvedt, Jørgen Aasvestad og Sander Nicolausson

Ordstyrer: Trond Jacob Rondestvedt

Sak 01/2020: Gjennomgang av visjonsdokument
Det gås gjennom hovedpunktene i visjonsdokumentet. Sveinung J. Ohrem ved SINTEF Ocean AS har sett igjennom på forhånd og føler at dokumentet dekker det bedriften har tenkt angående produktet.

Sak 02/2020: Kort forklaring av Scrum
Bachelorstudentene forteller kort hva Scrum er, og hvordan de skal bruke Scrum som arbeidsmetode under utviklingen av produktet.

Sak 03/2020: Fremvisning av wireframes og domenemodell
Wireframes av produktet vises fram til oppgavestillere. De får mulighet til å gi innspill til- og oppklare spørsmål som finnes rundt løsningen. Hovedpunkter fra gjennomgangen:

- En kunde har som regel flere behandlingsenheter.
- En behandlingsenhet er på en båt. Det vil da være relevant å lagre:
    - Navn på båten
    - Behandlingsprinsipp
    - Type pumpe
- Til en gjennomkjøring skal det lagres:
    - Kunde
    - Behandlingsenhet
    - Dato og tidspunkt
    - Sted
- Oppgavestillere ønsker at en gjennomkjøring kan eksporteres som både pdf, excel-fil og png.

- For å velge én/flere gjennomkjøringer er det mest relevant å kunne filtrere på kunde, behandlingsenhet og type pumpe.
- Oppgavestillere ønsker å ha mulighet til å legge til en offset ved sammenligning av to grafer, slik at de stemmer overens i tid. De ønsker også at bachelorstudentene (ut fra eksisterende python-script) prøver å finne en bedre metode for å bestemme når sensorfisken entrer systemet.

Sak 04/2020: Oppklaring av spørsmål

Det bestemmes at alle data lagres, også "dødtiden" før sensorfisken entrer systemet. Dersom en kunde slettes fra systemet, settes kunden til inaktiv. Data fra kunden eksisterer fortsatt.
Det bestemmes at alle ved SINTEF Ocean AS er "superbruker", og det skal altså ikke være noen brukere ved bedriften som har mindre restriksjoner enn andre. Det er også kun SINTEF-brukere som kan laste opp data til systemet.
Det bestemmes at følgende instrumenter skal fokuseres på i første omgang:

- ACC2: Akselerometer. Det kan eventuelt brukes en snittverdi mellom ACC1, ACC2 og ACC3.
- BAR3: Barometer. Er det instrumentet som gir den korrekte verdien for trykket. Måles i pascal, og er ønskelig at konverteres til bar.
- MAG: Magnetometer.
- GYR: Gyroskop. Måles i rad/s, og er ønskelig at konverteres til grader.

Sak 05/2020: Framdrift mot neste møte

Bachelorstudentene orienterer om hvilke sprintmål de har satt seg, som skal jobbes med fram mot neste møte.

Sak 06/2020: Eventuelt

Det bestemmes at neste møte flyttes fra mandag 10. februar kl. 12.30 - 13.30 til fredag 7. februar kl. 14.30 - 15.30.

27.01.2020, Sander Nicolausson

2.3 Møte 2

Innkalling til møte 2: Bacheloroppgave 061

Tidspunkt/sted:

Fredag 07.02.2020 kl 14:30 – 15:30

Rom OCE Sealab G. O. Sars, Brattørkaia 17C

Følgende personer innkalles:

Sveinung Johan Ohrem (Oppgavestiller)

Birger Venås (Oppgavestiller)

Trond Jacob Rondestvedt

Jørgen Aasvestad

Herman Ryen Martinsen

Sander Nicolausson

Agenda:

Sak 01/2020: Sprint review

Sak 02/2020: Planlegging av neste sprint

Sak 03/2020: Eventuelt

Ta kontakt med undertegnede dersom du ikke har anledning til å komme

Mvh

Trond Jacob Rondestvedt Trondheim 04.02.2020

Referat fra møte 1: Bacheloroppgave 061

Tidspunkt/sted: Fredag 07.02.2020 kl 14.30 – 15.30, Rom OCE Sealab G. O. Sars, Brattørkaia 17C

Til stede: Sveinung Ohrem (oppgavestiller), Elina Willert (SINTEF OCEAN, via Skype), Herman Ryen Martinsen, Trond Jacob Rondestvedt, Jørgen Aasvestad og Sander Nicolausson

Ikke møtt: Birger Venås (meldt fra)

Ordstyrer: Herman Ryen Martinsen

Sak 00/2020: Brukertester med kunde

Elina Willert hadde ikke mulighet til å delta på hele møtet, og begynte dermed med første sak. SINTEF OCEAN hadde nettopp vært på oppdrag med sensorfisken hos en kunde der de hadde kjørt en del gjennomkjøringer. Elina lurte på om studentene ønsket samarbeide med kunden med tanke på utformingen av grensesnittet for web-applikasjonen. På denne måten kan studentene gjennomføre brukertester på kunden under utviklingen av grensesnittet, for å få en bedre pekepinn på hvilken funksjonalitet og utforming kunden ønsker. Både studentene selv og Sveinung Ohrem var positive til dette, men det ble enighet om at man måtte ha noe konkret å vise fram til kunden som kunne vurderes. Studentene skal sende en mail til Elina med informasjon om diverse datoer og milepæler for når man tenker at man både kan og ønsker å gjennomføre brukertestene. Elina vil så videreformidle dette til kunden for å avklare om de ønsker å delta i dette samarbeidet.

Sak 01/2020: Sprint review

Det gjennomføres en sprint-review der det gjennomgås hva som ble gjort i løpet av sprinten. Følgende ble gjennomgått:

- Sprint-backlog
- Burndown-chart
- Databasemodell
- Nettside med trykk-graf

Det fortelles at visualisering av støt-graf ikke var gjennomført på grunn av tidsmangel. Visualisering av trykk-graf og annen backend var klart som planlagt i henhold til backlog. Det bemerkes at trykket både før og etter gjennomkjøringen skal være 1 bar

(atmosfæretrykk), dataene er lagret som litt over 1, dette ønskes justert til 1 i visualiseringen. I tillegg ønskes det at man bytter navn fra delouser til boat (båt) i databasemodellen for å spesifisere hva det er snakk om. En båt har alltid samme prinsipp.

Det blir også tatt opp samplingsfrekvensen, og om denne skal kuttes. En løsning som blir nevnt er å slå sammen 2 samplinger til 1, men det blir ikke enighet om hvorvidt det skal gjøres.

Sak 02/2020: Planlegging av neste sprint

Av ønskelig funksjonalitet i løpet av neste sprint nevnes følgende:

- Utvidet versjon av grafen:
    - Data fra flere gjennomkjøringer
    - Støt
    - Akselerasjoner
    - Histogram over G-krefter (studentene kan sjekke Python-scripts for hvordan dette gjøres per nå)
- Ønskelig med ned navigere mellom sider, noe interaktivt slik at man kan trykke litt rundt på siden. Spesielt med tanke på brukertester hos kunde.

Det bestemmes at det neste planlagte møtet, altså 24.02.2020 blir avlyst, pga. obligatorisk innovasjonscamp for studentene som stjeler mye av arbeidstiden for sprinten. Neste møtet med oppgavestiller blir derfor 09.03.2020, klokken 12:30-13:30 i henhold til møteplan.

Sak 03/2020: Eventuelt

Sveinung nevner en tidligere kunde som ønsket innholdet som excel-fil. Ønskelig å ha mulighet til å eksportere som CSV-fil. Det blir også tatt opp om det er noen annen data enn det som er inkludert i databasemodellen (ACC, MAG, BAR (temp), GYR) som vil bli brukt. Sveinung sier at det sannsynligvis ikke vil bli brukt noen annen data, men at det per nå er ønskelig at det lagres i databasen.

08.02.2020 Jørgen Aasvestad

2.4 Møte 3

Innkalling til møte 3: Bacheloroppgave 061

Tidspunkt/sted:

Mandag 09.03.2020 kl 12:30 – 13:30

Rom OCE Sealab G. O. Sars, Brattørkaia 17C

Følgende personer innkalles:

Sveinung Johan Ohrem (Oppgavestiller)

Birger Venås (Oppgavestiller)

Herman Ryen Martinsen

Sander Nicolausson

Jørgen Aasvestad

Trond Jacob Rondestvedt

Agenda:

Sak 01/2020: Gjennomgang av produktet så langt

Sak 02/2020: Planlegging av neste sprint

Sak 03/2020: Planlegging av tur til Frøy

Sak 04/2020: Eventuelt

Ta kontakt med undertegnede dersom du ikke har anledning til å komme

Mvh

Trond Jacob Rondestvedt Trondheim 04.03.2020

Referat fra møte 3: Bacheloroppgave 061

Tidspunkt/sted: Mandag 09.03.2020 kl 12.30 – 14.00, Rom OCE Sealab G. O. Sars, Brattørkaia 17C

Til stede: Sveinung Ohrem (oppgavestiller), Birger Venås (oppgavestiller), Herman Biørn Amundsen (SINTEF Ocean), Elina Willert (SINTEF Ocean, via Skype), Herman Ryen Martinsen, Trond Jacob Rondestvedt, Jørgen Aasvestad og Sander Nicolausson

Ordstyrer: Herman Ryen Martinsen

Sak 01/2020: Gjennomgang av produktet så langt

Representanter fra SINTEF Ocean AS er overrasket over og fornøyd med fremgangen i prosjektet. Det blir enighet om at følgende endringer/tillegg i funksjonalitet:

- Det skal legges til en graf for temperatur, i tillegg til eksisterende grafer for trykk, støt og g-krefter.
- Det ønskes å se en forhåndsvisning av trykkgrafen til en gjennomkjøring før dataen lastes opp, slik at personen som legger inn dataene kan bekrefte at det ikke er en "død" gjennomkjøring.
- Når man laster opp en gjennomkjøring skal man ha mulighet for å skrive inn et navn som kjennetegner den spesifikke gjennomkjøringen.
- Studentene skal se på muligheter for å legge til en funksjon som tillater brukeren å zoome inn på en del av grafen. Dette er spesielt tilfellet for grafen som fremstiller g-krefter.
- Det bestemmes at man skal ha mulighet til å sammenligne opp til 6 gjennomkjøringer samtidig.
- Det er ønskelig å legge til filer som vedlegg til en gjennomkjøring, og at disse skal vises på samme side som grafen vises.

Sak 02/2020: Planlegging av neste sprint

Det bestemmes at neste sprint forlenges med én uke grunnet eksamen, slik at neste møte nå blir 30.03.2020. Studentene skal fokusere på opplasting av binærfil og eksportering av grafer, samt tilbakemeldingene som ble gitt på dette møtet.

Sak 03/2020: Planlegging av tur til Frøy

SINTEF Ocean AS ordner transport ut til Frøya, men vil se an utviklingen med tanke på COVID-19 viruset. Studentene vil fortsette arbeidet med produktet fram mot turen, mens SINTEF vil jobbe med hvordan leveransen til kunde skal bli best mulig. Hensikten med turen er å bedre verktøy/metodikk for å optimalisere avlusningsprosessen og å kunne si hvor de uheldige punktene i eksisterende avluser er.
Det vil bli en omvisning på båten (eksisterende avluser) og studentene vil presentere produktet.

Sak 04/2020: Eventuelt

Når en gjennomkjøring slettes fra systemet skal verdier fra databasen slettes, mens binærfilen skal fortsatt bestå.
Studentene lurer på om illustrasjonen som er brukt i visjonsdokumentet er en god representasjon av hvordan avluserprosessen er, og oppgavestillere bekrefter at det kan brukes.
Python-scripts for uthenting av støt er ikke fasit. Oppgavestillere ønsker en slider med horisontal linje som oppdaterer nedre grense for hva et støt er.
Det er kun datapunkter fra graf som skal eksporteres som CSV-fil.

09.03.2020 Sander Nicolausson

2.5 Møte 4

Innkalling til møte 4: Bacheloroppgave 061

Tidspunkt/sted:

Mandag 30.03.2020 kl 12:30 – 13:30, Skype

Følgende personer innkalles:

Sveinung Johan Ohrem (Oppgavestiller)

Birger Venås (Oppgavestiller)

Herman Ryen Martinsen

Sander Nicolausson

Jørgen Aasvestad

Trond Jacob Rondestvedt

Agenda:

Sak 01/2020: Gjennomgang av produktet så langt

Sak 02/2020: Gå gjennom presentasjonen for Frøy møtet

Sak 03/2020: Informasjon om brukertester

Sak 04/2020: Plan for neste sprint

Sak 05/2020: Eventuelt

Ta kontakt med undertegnede dersom du ikke har anledning til å komme.

Mvh

Trond Jacob Rondestvedt Trondheim 26.03.2020

Referat fra møte 4: Bacheloroppgave 061

Tidspunkt/sted: Mandag 30.03.2020 kl 12.30 – 13.30, Microsoft Teams

Til stede: Sveinung Ohrem (oppgavestiller), Birger Venås (oppgavestiller), Herman Ryen Martinsen, Trond Jacob Rondestvedt, Jørgen Aasvestad og Sander Nicolausson

Ordstyrer: Trond Jacob Rondestvedt

Sak 01/2020: Gjennomgang av produktet så langt

Tilbakemelding fra oppgavestiller på endringer som ønskes:

- Ønske om at navn endres til testnavn v/opplasting, slik at brukere ikke skriver inn sitt eget navn.
- Ved opplasting av flere filer ønskes det at hver gjennomkjøring skilles på ved at det legges til et tall på hvert navn (unik identifikator). Dersom det blir tid ønskes det også at man får en oversikt på opplastingssiden over hvilke filer som er valgt (og om de er lastet opp).
    - ✔ test1 - 40 MB
    - ✔ test2 - 40 MB
    - ✔ test3 - 40 MB
    - ✔ test4 - 40 MB
    - ✔ test5 - 40 MB
- På aksene til de forskjellige grafene ønskes det at det står "Tid [s]" i stedet for "Tid i sekunder" og "Trykk [bar]" i stedet for "Trykk i bar".
- Det er fortsatt ønske om å ha en zoom-funksjon på grafene. Her er det først og fremst zooming i tid, altså å endre verdiene på x-aksen. Også mulighet for justering av øvre og nedre verdi på y-aksen.
- Det er ønskelig å kunne legge til notater på en gjennomkjøring. I første omgang så kan dette være en tekstboks.

Sak 02/2020: Gå gjennom presentasjonen før Frøy møtet

Det informeres om at representanter fra Frøy ikke nødvendigvis har teknologisk innsikt, så det ønskes derfor at presentasjonen skjer i et rolig tempo og med lite fagterminologi.

Sak 03/2020: Informasjon om brukertester

Sveinung ønsker et lite skriv om hvordan brukertestene skal foregå. Han skal spørre andre ansatte v/SINTEF, og mener at det ikke skal være problemer å skaffe 5 deltakere. Brukertester skjer første uken etter påske.

Sak 04/2020: Plan for neste sprint

Hovedfokus i kommende sprint er sikkerhet i systemet. Det skal legges til et lag med sikkerhet slik at brukere skal autentiseres.

Sak 05/2020: Eventuelt

Ingen saker.

30.03.2020 Sander Nicolausson

2.6 Møte 5

Innkalling til møte 5: Bacheloroppgave 061

Tidspunkt/sted:

Mandag 20.04.2020 kl 14.00 – 15:00, Teams

Følgende personer innkalles:

Sveinung Johan Ohrem (Oppgavestiller)

Birger Venås (Oppgavestiller)

Jan Harald Nilsen (Veileder)

Herman Ryen Martinsen

Sander Nicolausson

Jørgen Aasvestad

Trond Jacob Rondestvedt

Agenda:

Sak 01/2020: Gjennomgang av produktet så langt

Sak 02/2020: Resultat fra brukertester

Sak 03/2020: Plan for neste sprint

Sak 04/2020: Eventuelt

Ta kontakt med undertegnede dersom du ikke har anledning til å komme.

Mvh

Trond Jacob Rondestvedt Trondheim 17.04.2020

Referat fra møte 5: Bacheloroppgave 061

Tidspunkt/sted: Mandag 20.04.2020 kl 14.00 – 15.00, Microsoft Teams

Til stede: Sveinung Ohrem (oppgavestiller), Birger Venås (oppgavestiller), Jan Harald Nilsen (veileder), Herman Ryen Martinsen, Trond Jacob Rondestvedt, Jørgen Aasvestad og Sander Nicolausson

Ordstyrer: Herman Ryen Martinsen

Sak 01/2020: Gjennomgang av produktet så langt
Utviklergruppen viser frem de nye funksjonalitetene i programmet og får følgende tilbakemeldinger:

- Oppgavestillere vil gjerne unnlate downsampling av grafen som viser g-krefter.
- Det bør komme opp et varsel når man sletter gjennomkjøringer på oversiktssiden.
- Oppgavestillere vil gjerne at følgende funksjonaliteter prioriteres dersom det blir tid videre:
    - Tidsforskyve grafer ved sammenligning slik at de kan sammenlignes på forskjellige punkter.
    - Markere et spesifikt punkt på grafen og feste et notat til dette punktet.

Sak 02/2020: Resultat fra brukertester
Siden forrige møte har det blitt gjennomført brukertester med 5 ansatte ved SINTEF og 2 ansatte hos Frøygruppen. De tilbakemeldingene som kom etter brukertestene var i meget stor grad positive, hvor alle syntes at nettsiden var “simplistisk” og med det enkel i bruk. De aller fleste oppgavene som ble gitt til deltakerne ble også gjennomført. Det ble gjort noen observasjoner basert på brukertestene som bør tas til vurdering:

- Knappen som sletter gjennomkjøringer blir ofte trykket på når en bruker har valgt én eller flere gjennomkjøringer på oversiktssiden, da brukeren instinktivt tenker at dette er posisjonen til “Se gjennomkjøring”-knappen.
- Knappen som lar brukeren redigere tekstboksen for notater om en gjennomkjøring bør muligens endre navn og flyttes, da det kan være uklart hva denne gjør.

- Knappen hvor det står last ned bør også endre navn, og det bør potensielt komme opp et varsel når den trykkes på at brukeren er i ferd med å laste ned binærfilen for gjennomkjøringen. Enkelte testdeltakere tenkte at denne knappen ville gi dem mulighet til å eksportere dataen til gjennomkjøringen som en fil.

Sak 03/2020: Plan for neste sprint

I neste sprint/periode vil utviklergruppen først og fremst fortsette å utbedre/fikse feil ved funksjonalitet som allerede eksisterer i systemet. Dette er av den grunn at det er mye dokumentasjon som gjenstår før dato for leveranse.

Sak 04/2020: Eventuelt

Det stilles spørsmål om det er greit å dokumentere kildekode ved hjelp av jsdocs, og deretter legge dette ut på GitHub Pages selv om det er offentlig. Seere vil da ikke ha tilgang til kildekoden men til de kommentarene som er lagt til klasser og funksjoner i koden. Oppgavestiller ønsker ikke dette.

Oppgavestiller skal sjekke med IT-ansvarlige ved SINTEF angående hvilken tjener som skal tas i bruk, og kjøp av domene.

Det er ønskelig at det skal skrives en enkel brukerveiledning til applikasjonen.

Oppgavestiller ønsker også at dersom det er funksjonalitet som utviklergruppen ikke rekker å implementere, at dette dokumenteres. Det er planlagt et eget kapittel i hovedrapporten som omhandler dette.

20.04.2020 Sander Nicolausson

2.7 Møte 6

Innkalling til møte 6: Bacheloroppgave 061

Tidspunkt/sted:

Mandag 04.05.2020 kl 12.30 – 13:30, Microsoft Teams

Følgende personer innkalles:

Sveinung Johan Ohrem (Oppgavestiller)

Birger Venås (Oppgavestiller)

Herman Ryen Martinsen

Sander Nicolausson

Jørgen Aasvestad

Trond Jacob Rondestvedt

Agenda:

Sak 01/2020: Gjennomgang av produktet

Sak 02/2020: Gjennomgang av fremtidig funksjonalitet og utvidelser

Sak 03/2020: Eventuelt

Ta kontakt med undertegnede dersom du ikke har anledning til å komme.

Mvh

Trond Jacob Rondestvedt Trondheim 01.05.2020

Referat fra møte 6: Bacheloroppgave 061

Tidspunkt/sted: Tirsdag 05.05.2020 kl 12.30 – 13.30, Microsoft Teams

Til stede: Elina Willert (v/SINTEF Ocean), Herman Ryen Martinsen, Trond Jacob Rondestvedt, Jørgen Aasvestad og Sander Nicolausson

Ordstyrer: Herman Ryen Martinsen

Fraværende: Sveinung Ohrem (oppgavestiller) og Birger Venås (oppgavestiller)

Sak 01/2020: Gjennomgang av produktet

Siden ingen av oppgavestillerne er tilstede, utgår gjennomgangen av produktet.

Sak 02/2020: Gjennomgang av fremtidig funksjonalitet og utvidelser

Det blir gått gjennom hvilken funksjonalitet som kan bli implementert etter levering av produktet, og blir diskutert at man mistenker at store trykkendringer over kort tid er negativt for fisken. Det er dog ingen forskning på dette per dags dato så det kan ikke brukes til å "rangere" avlusere.
Det blir bestemt at Elina skal sende en e-post til IT-avdelingen til SINTEF for å oppklare hvem som tar eierskap over prosjektet innad i bedriften. Dette for å sikre at noen tar ansvar for vedlikehold og videreutvikling.
Ved skriving i rapporten angående fremtidige utvidelser ønskes det at disse skal være presise og kobles opp mot et behov som kunden har. Dette for å gjøre utvidelsene mer forståelige for ansatte ved SINTEF.

Sak 03/2020: Eventuelt

Det blir bestemt at Elina hører med Frøy om hvilke gjennomkjøringer de ønsker å få sammenlignet, og ved hvilke punkter. Dette for at Frøy som har vært en bidragsyter i utviklingen av produktet skal få litt tilbake for arbeidet de har lagt ned.

05.05.2020 Sander Nicolausson

2.8 Møte 7

Innkalling til møte 7: Bacheloroppgave 061

Tidspunkt/sted:

Mandag 11.05.2020 kl 12.00 – 12:30, Microsoft Teams

Følgende personer innkalles:

Sveinung Johan Ohrem (Oppgavestiller)

Birger Venås (Oppgavestiller)

Herman Ryen Martinsen

Sander Nicolausson

Jørgen Aasvestad

Trond Jacob Rondestvedt

Agenda:

Sak 01/2020: Gjennomgang av det endelige produktet

Sak 02/2020: Spørsmål angående prosjektets effektmål

Sak 03/2020: Gjennomgang av systemdokumentasjon

Sak 04/2020: Eventuelt

Ta kontakt med undertegnede dersom du ikke har anledning til å komme.

Mvh

Trond Jacob Rondestvedt Trondheim 11.05.2020

Referat fra møte 7: Bacheloroppgave 061

Tidspunkt/sted: Tirsdag 11.05.2020 kl 12.00 – 12.30, Microsoft Teams

Til stede: Sveinung Ohrem (oppgavestiller), Birger Venås (oppgavestiller), Herman Ryen Martinsen, Trond Jacob Rondestvedt, Jørgen Aasvestad og Sander Nicolausson

Ordstyrer: Herman Ryen Martinsen

Sak 01/2020: Gjennomgang av endelig produktet

Sluttproduktet gjennomgås og det snakkes om hvilke småting som kan settes som "Fremtidige utvidelser". Oppgavestillere mener at siste punkt på grafen har vel mange desimaler, og ønsker videre mulighet for å skru på og av et rutenett i grafen.

Sak 02/2020: Spørsmål angående prosjektets effektmål

Det snakkes om hvilke effektmål SINTEF ønsker at vårt produkt skal bidra til i Sensorfisk-prosjektet. De ønsker å øke verdien av leveranse til kunde og å gjøre "produktet" Sensorfisk mer attraktivt. I det lange løp kan også disse to bidra til å samle inn nok data til å utvikle metoder for å kartlegge fiskens fysiske forhold i rørtransport.

Sak 03/2020: Gjennomgang av systemdokumentasjon

Systemdokumentasjon gjennomgås og SINTEF mener at de ikke behøver mer enn det som står der per i dag. De lurer også på om de kan bruke systemdokumentasjonen i en senere rapport om deres pågående prosjekt.

Sak 04/2020: Eventuelt

Det bestemmes at overlevering av kode skjer etter innlevering til NTNU, altså etter 20. mai.

11.05.2020 Sander Nicolausson

2.9 Veiledningsmøte 1

Innkalling til veiledningsmøte 1: Bacheloroppgave 061

Tidspunkt/sted: Torsdag 30.01.2020 kl 12.00 – 12.30, NTNU Gløshaugen

Følgende personer innkalles:

Jan Harald Nilsen (veileder)

Trond Jacob Rondestvedt

Jørgen Aasvestad

Herman Ryen Martinsen

Sander Nicolausson

Agenda:

Sak 01/2020: Gjennomgang av- og tilbakemelding på visjonsdokument

Sak 02/2020: Eventuelt

Ta kontakt med undertegnede dersom du ikke har anledning til å komme

Mvh

Sander Nicolausson Trondheim 29.01.2020

Referat fra veiledningsmøte 1: Bacheloroppgave 061

Tidspunkt/sted: Torsdag 30.01.2020 kl 12.00 – 12.30, Rom 201, IT-bygget, Gløshaugen

Til stede: Jan Harald Nilsen (veileder), Trond Jacob Rondestvedt, Jørgen Aasvestad og Sander Nicolausson

Fraværende: Herman Ryen Martinsen

Ordstyrer: Trond Jacob Rondestvedt

Sak 01/2020: Gjennomgang av- og tilbakemelding på visjonsdokument

Veileder gir tilbakemelding på følgende punkter i visjonsdokumentet:

- Forsiden av visjonsdokumentet bør være estetisk penere. Navn på gruppemedlemmene og dato for ferdigstilling bør også stå her.
- Det bør brukes illustrasjoner/figurer som viser hvordan sensorfisken ser ut, og hvordan denne brukes i praksis.
- Det bør opprettes en ordliste som forklarer tekniske ord som blir brukt, og det bør henvises til denne i teksten. Denne ordlisten tas med videre til hovedrapporten.
    - Ordlisten brukes for å gjøre teksten mer tilpasset til lesere som ikke er inneforstått med akronymer og bransjesjargong.
- Det bør brukes flere referanser i teksten. Veileder bruker SINTEFs prosjektnavn KVALISYS som eksempel, da det her bør refereres til en lenke til prosjektbeskrivelsen på SINTEFs hjemmeside.
    - Det bør også henvises til dokumentet som er brukt som mal for dette dokumentet.
- I innledningen forklares det at sensorfisken samler inn data for trykk, temperatur og akselerasjon, men senere refereres det til et gyroskop. De forskjellige målingene og instrumentene bør samsvare, og instrumentene bør også forklares.
- Tabellene i visjonsdokumentet bør ha en tabelltekst som forklarer hensikten med tabellen og hvilken informasjon den inneholder. De bør også være nummerert (eks. “Figur 4.1”) slik at de enkelt kan refereres til.

- Underkapitlene kan endres fra enkeltord (som det står i malen) til korte, mer forklarende setninger.
- Delkapittel 3.5 - Alternativer til vårt produkt bør endres, da det beskrives at det ikke finnes eksisterende alternativer til vårt produkt. Alternativet som finnes er måten det gjøres på i dag.
- Figuren som viser produktets rolle i brukermiljøet bør ha en mer utfyllende tekst som innledning, og en bedre beskrivelse. Beskrivelsen bør være så god at leseren forstår produktets rolle i brukermiljøet selv om figuren ikke er tilstede.
- Forutsetninger og avhengigheter, og produktets funksjonelle avhengigheter bør nummereres. Det bør også være noen innledende setninger som beskriver disse.

Sak 02/2020: Eventuelt

Det gis veiledning på hvordan problemstillingen bør utformes. På et systemutviklingsprosjekt er det vanskelig å utforme en god vitenskapelig problemstilling, og det bestemmes at denne bør rettes inn mot hvilken teknologi eller hvilket rammeverk som er brukt i utviklingen og hvorfor. Det kan også settes i sammenheng med at det skal være et godt brukergrensesnitt for personer uten særlig teknisk innsikt.
Vitenskapelig metode som brukes her er Design Science Research.

Veileder ønsker at punktene på Gantt-diagrammet blir mer generelle, og at disse samsvarer med aktivitetene i timelistene.

30.01.2020, Sander Nicolausson

2.10 Veiledningsmøte 2

Innkalling til veiledningsmøte 2: Bacheloroppgave 061

Tidspunkt/sted: Fredag 20.03.2020 kl 10.00 – 11.00, Skype

Følgende personer innkalles:

Jan Harald Nilsen (veileder)

Trond Jacob Rondestvedt

Jørgen Aasvestad

Herman Ryen Martinsen

Sander Nicolausson

Agenda:

Sak 01/2020: Gjennomgang av- og tilbakemelding på kravdokument

Sak 02/2020: Spørsmål til hovedrapport

Sak 03/2020: Eventuelt

Ta kontakt med undertegnede dersom du ikke har anledning til å møte.

Mvh

Sander Nicolausson Trondheim 19.03.2020

Referat fra veiledningsmøte 2: Bacheloroppgave 061

Tidspunkt/sted: Torsdag 20.03.2020 kl 10.00 – 11.00, Skype

Til stede: Jan Harald Nilsen (veileder), Herman Ryen Martinsen, Trond Jacob Rondestvedt, Jørgen Aasvestad og Sander Nicolausson

Ordstyrer: Herman Ryen Martinsen

Sak 01/2020: Gjennomgang av- og tilbakemelding på kravdokumentasjon
Veileder gir tilbakemelding på følgende punkter i kravdokumentet:

- Introduksjonen behøver ikke være like utfyllende som i visjonsdokumentet, men bør være utfyllende nok til at den kan "stå på egne ben".
- Tabellene bør ha samme nummerering som underkapitlene. Dersom man har flere tabeller i samme kapittel/underkapittel tas det i bruk enda en "serie".
- Uttrykket "hashing" eller "hashing med salt" bør tas med i ordlisten.
- Referanser bør være mer presis enn (Norges teknisk-naturvitenskapelige universitet, u.å). Det foreslås at man eventuelt forklarer hvilket dokument man refererer til i teksten for å få en konsistens i refereringen.

Sak 02/2020 Spørsmål til hovedrapport
Bachelorgruppen har spørsmål til utforming av hovedrapporten. Det bestemmes at kapitlet "problemstilling" endres til "forskningsspørsmål". Det kan også være lurt å relatere forskningsspørsmålet til det som har blitt utviklet (eller det man ønsker å oppnå med utviklingen). Her kan man være konkrete. Det bør også være en sammenheng mellom forskningsspørsmålet og systemet som utvikles.

Veileder ønsker et delkapittel om litteraturstudie under kapittel 1. Her oppsummeres utvikling fram til nå, og det bør dras fram tilsvarende arbeid som er gjort av andre.

Innledningen bør inneholde bachelorgruppens hovedbidrag.

Teoridelen bør ikke bli for omfattende, men man bør fokusere mer på hva som gjør at man kommer til resultatet i rapporten.

Valg av rammeverk bør knyttes nært opp mot teorien.

Selv om brukertesting har blitt rammet av COVID-19, bør det vurderes brukertesting over nett. Det bør også nevnes hvordan situasjonen med viruset har påvirket prosessen (skrives om i innledning og utviklingsmetode).

Alle resultater bachelorgruppen har kommet fram til som er relatert til forskningsspørsmålet skal være med. Det skal "reklameres" for det som har blitt gjort.

Dersom bachelorgruppen får tid til å se på mulighetene for implementering av maskinlæring bør dette være med i kapittelet "konklusjon og videre arbeid". Det kan også være et lite underkapittel i teoridelen som forklarer forskjellig maskinlæringsteori (random forest, cross-validation).

Sak 03/2020: Eventuelt

Veileder vil bli med på et møte med oppdragsgiver etter påske.

20.03.2020, Sander Nicolausson

2.11 Veiledningsmøte 3

Innkalling til veiledningsmøte 3: Bacheloroppgave 061

Tidspunkt/sted: Tirsdag 05.05.2020 kl 14.30 – 15.15, Teams

Følgende personer innkalles:

Jan Harald Nilsen (veileder)

Jørgen Aasvestad

Herman Ryen Martinsen

Sander Nicolausson

Trond Jacob Rondestvedt

Agenda:

Sak 01/2020: Gjennomgang av hovedrapporten

Sak 02/2020: Eventuelt

Ta kontakt med undertegnede dersom du ikke har anledning til å møte.

Mvh

Trond Rondestvedt Oslo 04.05.2020

Referat fra veiledningsmøte 3: Bacheloroppgave 061

Tidspunkt/sted: Torsdag 05.05.2020 kl 14.30 – 15.30, Skype

Til stede: Jan Harald Nilsen (veileder), Herman Ryen Martinsen, Trond Jacob Rondestvedt, og Sander Nicolausson

Ordstyrer: Herman Ryen Martinsen

Fraværende: Jørgen Aasvestad

Sak 01/2020: Gjennomgang av hovedrapporten

Møtet innledes med at bachelorstudentene har noen praktiske spørsmål angående hovedrapporten. Forskningsspørsmålene diskuteres, og veileder er enig i at det bør heller være flere konkrete forskningsspørsmål enn ett stort forskningsspørsmål som er vanskelig å svare på. Videre blir det gått gjennom resten av kapittel 1, og det blir diskutert at målbare mål er viktig.

Bachelorstudentene synes kanskje at teorikapitlet har blitt i lengste laget, og veileder sier seg enig. Det bestemmes at noen av underkapitlene kortes litt ned, og at det legges til teori om sikkerhet.

Det blir spurt om det er nødvendig å ha med all teknologi som er brukt, og veileder mener at dette kan være fornuftig med tanke på at oppgavestiller sikkert ønsker å vite dette da de skal ta over produktet etter prosjektslutt. Det påpekes igjen at gjennom hele rapporten er det vesentlig å være forsiktig med å utgi noe man ikke har kommet opp med selv som sitt eget.

I resultatkapitlet kan det være lurt å prøve å relatere resultatene til forskningsspørsmålene. Dersom noe av resultatene er beskrevet et annet sted, er det greit å dra frem hovedpunktene fra dette, og deretter henvise til gitt dokument.

Sak 02/2020: Eventuelt

Da oppgavestiller ikke vil at dokumentasjonen av kildekoden (i systemdokumentasjonen) skal ligge offentlig, bestemmes det at man her skal henvise til oppgavestiller i dokumentet.

06.05.2020, Sander Nicolausson

3 Timelister med statusrapporter

Timelistene i dette kapitlet viser timeforbruket på hver aktivitet per person per uke, timeforbruket totalt per aktivitet, total ukesum for hver person og den totale akkumulerte summen hittil. Under hver timeliste er det også en kort beskrivelse av hva hver enkelt i gruppa har jobbet med i uka som har gått, samt en felles overordnet vurdering av hva som har blitt oppnådd.

3.1 Uke 2

Uke 2 (6. januar)						
Aktiviteter	**Herman**	**Jørgen**	**Sander**	**Trond**	**Ukesum aktivitet**	**Total sum aktivitet**
Møter	1,5	1,5	1,5	1,5	6	6
Dokumentasjon	0	0	0	0	0	0
Hovedrapport	0	0	0	0	0	0
Frontend	0	0	0	0	0	0
Backend	0	0	0	0	0	0
Testing	0	0	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0
Ukesum	**1,5**	**1,5**	**1,5**	**1,5**	**6**	
Akkumulert hittil i år	**1,5**	**1,5**	**1,5**	**1,5**		**6**

Statusrapport Herman

Skrev møteinnkalling til oppstartsmøte sammen med gruppa, og gjorde forberedelser til møtet.

Statusrapport Jørgen

Skrev møteinnkalling til oppstartsmøte i fellesskap med gruppe, og gjorde forberedelser til møtet.

Statusrapport Sander

Skrev møteinnkalling til oppstartsmøte sammen med gruppen og forberedte oss til møtet.

Statusrapport Trond

Forberedelser til oppstartsmøte og skrev møteinnkalling sammen med gruppen.

Samlet statusrapport

Uka ble brukt til å skrive møteinnkalling og gjøre forberedelser til oppstartsmøtet i neste uke.

3.2 Uke 3

Uke 3 (13. januar)						
Aktiviteter	**Herman**	**Jørgen**	**Sander**	**Trond**	**Ukesum aktivitet**	**Total sum aktivitet**
Møter	2	2	2	2	8	14
Dokumentasjon	15	15	14	14	58	58
Hovedrapport	0	0	0	0	0	0
Frontend	0	0	0	0	0	0
Backend	0	0	0	0	0	0
Testing	0	0	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0
Ukesum	**17**	**17**	**16**	**16**	**66**	
Akkumulert hittil i år	**18,5**	**18,5**	**17,5**	**17,5**		**72**

Statusrapport Herman

På mandag var det oppstartsmøte med SINTEF Ocean AS. Resten av uka ble brukt til å utarbeide visjonsdokumentet sammen med resten av gruppa.

Statusrapport Jørgen

Oppstartsmøte i begynnelsen av uka, de resterende dagene ble benyttet til å arbeide med visjonsdokumentet i samråd med gruppa.

Statusrapport Sander

Var på mandag på oppstartsmøte i SINTEFs lokaler, og brukte noe tid etter møtet til å skrive møtereferat. Resten av uka ble brukt på å skrive visjonsdokument.

Statusrapport Trond

Deltok på oppstartsmøte mandag den 13. januar. Jeg opprettet en Google Drive mappe for alt av dokumentasjon, og vi startet å skrive visjonsdokument og annen dokumentasjon.

Samlet statusrapport

Vi hadde på mandag (13. januar) oppstartsmøte med Sveinung Johan Ohrem v/SINTEF Ocean AS og veileder Jan Harald Nilsen fra NTNU. Møtet ble holdt i SINTEFs lokaler på Brattørkaia.
Videre ble det jobbet med visjonsdokumentet og annen dokumentasjon.

3.3 Uke 4

Uke 4 (20. januar)						
Aktiviteter	**Herman**	**Jørgen**	**Sander**	**Trond**	**Ukesum aktivitet**	**Total sum aktivitet**
Møter	0	0	0	0	0	14
Dokumentasjon	9	15	21	20	65	123
Hovedrapport	0	0	0	0	0	0
Frontend	2,5	3,5	2	0	8	8
Backend	10	3	0	2	15	15
Testing	0	0	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0
Ukesum	**21,5**	**21,5**	**23**	**22**	**88**	
Akkumulert hittil i år	**40**	**40**	**40,5**	**39,5**		**160**

Statusrapport Herman

Utarbeidet wireframes for brukergrensesnittet og ferdigstilte førsteutkastet av visjonsdokumentet sammen med gruppa. Satt opp prosjektet i Node.js og laget en grov databasemodell. Bidro også til å få utarbeidet produkt-backlog og sprint-backlog for sprint 1.

Statusrapport Jørgen

Arbeidet med wireframes og visjonsdokument i fellesskap med gruppa. Hjalp også med å utforme produkt-backlog og sprint-backlog for sprint 1. Begynte med å opprette enkelte Service-klasser og deres oppsett for axios.

Statusrapport Sander

Jobbet litt videre med visjonsdokumentet, samt utformet wireframes for nettsiden. Bidrog også med planlegging av første sprint. Dette innebar å sette opp produktbacklog for produktet basert på oppstartsmøtet og sprintbacklog for første sprint.

Statusrapport Trond

Vi lagde førsteutkast av wireframes og fortsatte med å skrive dokumentasjon. Samtidig startet vi å planlegge første sprint og lagde produkt- og sprintbacklog. I slutten av uken startet jeg å arbeide litt med endepunkter på serversiden.

Samlet statusrapport

Denne uka ble det jobbet videre med visjonsdokumentet. Det ble også utformet wireframes med forslag til hvordan nettsiden skal bli seende ut, og noen av oss begynte å se på hvordan prosjektoppsettet skulle være. Det ble opprettet et Node.js-prosjekt. Vi brukte også noe tid på å planlegge sprint 1.

3.4 Uke 5

Uke 5 (27. januar)						
Aktiviteter	**Herman**	**Jørgen**	**Sander**	**Trond**	**Ukesum aktivitet**	**Total sum aktivitet**
Møter	1	2	2	2	7	21
Dokumentasjon	0	5	6	5	16	139
Hovedrapport	10	10	10	10	40	40
Frontend	0	3	7	0	10	18
Backend	10	0	0	4	14	29
Testing	0	4	0	3	7	7
	0	0	0	0	0	0
	0	0	0	0	0	0
Ukesum	**21**	**24**	**25**	**24**	**94**	
Akkumulert hittil i år	**61**	**64**	**65,5**	**63,5**		**254**

Statusrapport Herman

På mandag var det møte med SINTEF Ocean AS. Resten av uka har blitt brukt til å jobbe med oppgaver fra sprint-backlogen. Det har blitt satt opp en MySQL-database ved hjelp av Sequelize, og opprettet testdata til denne databasen. Deler av mandag, onsdag og torsdag gikk med til obligatorisk seminar i bachelorfaget.

Statusrapport Jørgen

Deltok i møte med SINTEF på mandag, ble påbegynt med noen tester for Service-klasser. Deler av uken gikk med til obligatorisk seminar i bachelorfaget. Slutten av uken ble det arbeidet med revisjon av visjonsdokumentet etter tilbakemelding fra veileder.

Statusrapport Sander

Hadde møte med SINTEF på mandag, og brukte noe tid etter møtet på å skrive møtereferat. Jobbet med å opprette en navigerbar struktur med utgangspunkt i wireframes, slik at vi har oversikt over hvordan nettsiden skal bli seende ut. Hadde også møte med veileder på torsdag for tilbakemelding på visjonsdokumentet.

Statusrapport Trond

Jeg var med på møte med SINTEF på mandag, der vi viste frem wireframes og gikk kjapt igjennom visjonsdokumentet. Videre utover uken arbeidet jeg videre med litt dokumentasjon og fortsatte med endepunkter. Vi hadde også presentasjon av førsteutkastet til problemstillingen denne uken.

Samlet statusrapport

Vi hadde på mandag (27. januar) et nytt møte med SINTEF hvor vi viste frem visjonsdokumentet og wireframes for nettsiden. Vi startet også den første sprinten denne uken. Det ble jobbet med utforming av de forskjellige sidene på frontend, og det ble opprettet database og relevante endepunkter på backend. Deler av mandag, onsdag og torsdag gikk bort til obligatorisk seminar i bachelorfaget. Møte med veileder angående visjonsdokument på torsdag.

3.5 Uke 6

Uke 6 (3. februar)						
Aktiviteter	**Herman**	**Jørgen**	**Sander**	**Trond**	**Ukesum aktivitet**	**Total sum aktivitet**
Møter	1	2	1	1	5	26
Dokumentasjon	3	1	3	3,5	10,5	149,5
Hovedrapport	0	0	0	0	0	40
Frontend	3	3	18	1	25	43
Backend	16	0	0	7	23	52
Testing	0	3	0	10	13	20
	0	0	0	0	0	0
	0	0	0	0	0	0
Ukesum	**23**	**9**	**22**	**22,5**	**76,5**	
Akkumulert hittil i år	**84**	**73**	**87,5**	**86**		**330,5**

Statusrapport Herman

Fant ut at database-oppsettet med Sequelize måtte endres en god del, så mye av uka har gått med på dette. Har jobbet med å få lest inn dataen vi har fått fra SINTEF Ocean AS og lagt den inn i databasen. Har i tillegg gjort revideringer på visjonsdokumentet etter tilbakemelding fra veileder i forrige uke. Møte med SINTEF Ocean AS på fredag.

Statusrapport Jørgen

Var syk i store deler av denne uken så ble noe begrenset arbeidstid. Fortsatte utvikling av modeller til klient-siden samt nødvendige tester. Deltok i møte med SINTEF på fredag.

Statusrapport Sander

Har brukt størsteparten av uken på å utforme siden som skal vise fram data fra sensorfisken i grafform, samt grafen for trykk. Dette innebar å lese seg opp på- og forstå dokumentasjonen til grafbiblioteket. Brukte også noe tid på å utbedre visjonsdokumentet basert på tilbakemeldinger fra veileder.

Statusrapport Trond

Jobbet med å lage alle endepunkter vi trengte, deretter så startet jeg å skrive tester for alle endepunktene. Jeg skrev også møteinnkalling for møte med SINTEF på denne fredagen, og lagde en møteplan av planlagte møter. Jeg var med på møtet på fredagen.

Samlet statusrapport

Denne uka jobbet vi videre med utforming av de ulike sidene på frontend, samt hvordan graf for trykk skal vises frem. Det ble også jobbet videre med oppsett av database og endepunkter på backend. Møtet som egentlig skulle vært mandag i neste uke (10. februar) ble framskyndet til fredag denne uken (7. februar) på grunn av innovasjonscamp i annet emne. Dette gjorde at vi ikke kom helt i mål med trykkgrafen, samt at noen tester for endepunkter mangler.

3.6 Uke 7

Uke 7 (10. februar)						
Aktiviteter	**Herman**	**Jørgen**	**Sander**	**Trond**	**Ukesum aktivitet**	**Total sum aktivitet**
Møter	0	0	0	0	0	26
Dokumentasjon	3	3,5	4	4	14,5	164
Hovedrapport	0	0	0	0	0	40
Frontend	0	0	6	0	6	49
Backend	6	3	0	0	9	61
Testing	0	0	0	0	0	20
	0	0	0	0	0	0
	0	0	0	0	0	0
Ukesum	**9**	**6,5**	**10**	**4**	**29,5**	
Akkumulert hittil i år	**93**	**79,5**	**97,5**	**90**		**360**

Statusrapport Herman

Deltok på obligatorisk innovasjonscamp fra mandag til onsdag, så det ble ikke jobbet noe med prosjektet disse dagene. Utenom dette ble det forsket litt på hvordan rådatafilene fra Sensorfisk er bygget opp, og andreutkastet av visjonsdokumentet ble ferdigstilt.

Statusrapport Jørgen

Obligatorisk innovasjonscamp fra mandag til onsdag. Ble satt opp flere service-metoder i tillegg til at testing for service-klasser ble restrukturert med egen setup-fil. Ferdigstillelse av visjonsdokument.

Statusrapport Sander

Hadde innovasjonscamp fra mandag til onsdag. Brukte resten av uka på å jobbe med de tilbakemeldingene vi hadde fått på visjonsdokumentet, samt siden hvor en bruker kan velge gjennomkjøringer. Hadde litt problemer med hvordan man skal få en checkbox til å være markert fra starten av men det ordnet seg til slutt.

Statusrapport Trond

Vi jobbet ikke med bacheloroppgaven fra mandag til onsdag grunnet innovasjonscamp i et annet emne. På torsdag så var jeg med på å planlegge neste sprint som skulle starte neste uke. Jeg var en tur ned til Oslo denne helgen så jeg jobbet ikke denne fredagen.

Samlet statusrapport

Det gikk bort tre dager til obligatorisk innovasjonscamp denne uka. Utenom dette fikk vi jobbet en del med visjonsdokumentet og ferdigstilt et andreutkast av dokumentet. I tillegg fikk alle jobbet noen timer med sine respektive deler av koden.

3.7 Uke 8

Uke 8 (17. februar)						
Aktiviteter	**Herman**	**Jørgen**	**Sander**	**Trond**	**Ukesum aktivitet**	**Total sum aktivitet**
Møter	0	0	0	0	0	26
Dokumentasjon	4,5	7,5	4,5	5	21,5	185,5
Hovedrapport	0	0	0	0	0	40
Frontend	1	15,5	26	12	54,5	103,5
Backend	27,5	7	1	16	51,5	112,5
Testing	0	0	0	0	0	20
	0	0	0	0	0	0
	0	0	0	0	0	0
Ukesum	**33**	**30**	**31,5**	**33**	**127,5**	
Akkumulert hittil i år	**126**	**109,5**	**129**	**123**		**487,5**

Statusrapport Herman

Har utarbeidet en funksjon som regner ut posisjonen til magnetbåndene, samt gjort noen små endringer i database og endepunkter. Har også begynt å skrive på kravdokumentet sammen med resten av gruppa.

Statusrapport Jørgen

Utarbeidet funksjon som finner ut når en gjennomkjøring har startet ved å se på trykkfall. La inn testdata fra binærfil for ekstra gjennomkjøring i databasen. Metode for å sammenligne 5 gjennomkjøringers trykk-graf. Metoder som henter ut nødvendig data fra flere gjennomkjøringer. Arbeid med kravdokumentet.

Statusrapport Sander

Fortsatte hovedsakelig på siden som viser en oversikt over alle gjennomkjøringer for en bruker. Lagt til logikk for hvordan flere gjennomkjøringer kan velges for å sammenlignes. Hadde igjen litt problemer med å få implementert checkboxer i en underkomponent, når man skal ha tak i verdien på checkboxen i toppkomponenten. Begynte også å skrive på kravdokumentet sammen med resten av gruppa.

Statusrapport Trond

Denne uken arbeidet jeg med å filtrere grafer/ fjerne støy fra grafer. Jeg fant en pakke i node som kunne gjøre dette greit og enkelt, deretter så startet jeg å konvertere noe av den tidligere koden som SINTEF hadde brukt til å regne ut g-krefter fra python til javascript. Dette jobbet jeg videre med resten av uken og fikk samtidig satt av litt tid til å restrukturere litt på serveren. På fredagen så hadde jeg en fungerende prototype av utregningen av g-krefter så jeg prøvde også å lage en kjapp og enkel funksjon for beregning av støt. Jeg fikk også satt av tid til å jobbe med dokumentasjon i løpet av uken.

Samlet statusrapport

Denne uka startet vi på sprint 2. Det ble hovedsakelig jobbet med hvordan de forskjellige grafene skal framstilles for brukeren. Hovedfokus her var filtrering av trykkgrafen og plassering av magnetbånd på grafen. Det ble også lagt til funksjonalitet for at brukeren skal kunne sammenligne flere gjennomkjøringer med hverandre. Hele gruppa startet også på arbeidet med kravdokumentet.

3.8 Uke 9

Uke 9 (24. februar)						
Aktiviteter	**Herman**	**Jørgen**	**Sander**	**Trond**	**Ukesum aktivitet**	**Total sum aktivitet**
Møter	0	0	0	0	0	26
Dokumentasjon	0	3,5	4,5	3	11	196,5
Hovedrapport	0	0	0	0	0	40
Frontend	0	16	27	0	43	146,5
Backend	0	11	2	17	30	142,5
Testing	0	0	0	11	11	31
	0	0	0	0	0	0
	0	0	0	0	0	0
Ukesum	**0**	**30,5**	**33,5**	**31**	**95**	
Akkumulert hittil i år	**126**	**140**	**162,5**	**154**		**582,5**

Statusrapport Herman

Bortreist på treningssamling i Frankrike med begrenset internettilgang, så ble ikke jobbet noe med prosjektet denne uka.

Statusrapport Jørgen

Oppdaterte trykk-graf slik at den har mulighet til å vise flere gjennomkjøringer samtidig. Funksjonalitet for å filtrere flere gjennomkjøringer grafisk. Påbegynte videreutvikling av funksjon som finner støt mot rørveggen. La til nytt diagram (stolpediagram) som illustrerer støtene. Oppdaterte endepunkt og og frontend slik at støtene ble hentet ut.

Statusrapport Sander

Brukte denne uka på å opprette sider hvor en administratorbruker ser oversikten over alle bedrifter som ligger inn i systemet. Lagde også en side for hvor administratorbrukeren kan legge til nye bedrifter. Litt av uken ble og brukt på å jobbe mot innlevering av førsteutkastet av kravdokumentasjonen.

Statusrapport Trond

Jeg startet denne uken med å restrukturere databasen for å kunne ha en konfigurasjonsfil der man har to forskjellige databaser for utvikling og testing, la også til en database for produksjon. Når dette var gjort var det mye enklere å sette opp synkronisering av databasen før og etter alle testene. Samtidig skrev jeg noen flere tester for endepunkter. I slutten av denne uken så lagde jeg et endepunkt for opplastning av binærfiler. I løpet av uken jobbet jeg også med å fullføre kravdokumentasjonen.

Samlet statusrapport

Fortsatte denne uka å jobbe med å legge til flere grafer i systemet, herunder støtgraf og graf for g-krefter. Det ble også jobbet på serversiden med hvordan disse dataene skal regnes ut (utregning av støt basert på g-krefter). Førsteutkastet av kravdokumentet ble ferdigstilt.

3.9 Uke 10

Uke 10 (2. mars)						
Aktiviteter	**Herman**	**Jørgen**	**Sander**	**Trond**	**Ukesum aktivitet**	**Total sum aktivitet**
Møter	0,5	0	0	0	0,5	26,5
Dokumentasjon	0	0,5	0	0,5	1	197,5
Hovedrapport	0	0	0	0	0	40
Frontend	0	11	21	0	32	178,5
Backend	16	7	0	15	38	180,5
Testing	0	0	0	4	4	35
	0	0	0	0	0	0
	0	0	0	0	0	0
Ukesum	**16,5**	**18,5**	**21**	**19,5**	**75,5**	
Akkumulert hittil i år	**142,5**	**158,5**	**183,5**	**173,5**		**658**

Statusrapport Herman

Kom hjem fra Frankrike på tirsdag, og onsdagen gikk med til obligatorisk CCD-sesjon. Resten av uka ble brukt til å fullføre sprint-backlogen, samt å utforske koden til MissionPlanner for å finne ut hvordan de leser inn rådatafiler fra Sensorfisk. Fikk også sendt førsteutkast av kravdokument til veileder, og gjort litt forberedelser til møtet neste mandag.

Statusrapport Jørgen

Feilrettinger til g-krefter grafen. Henter kun ut data etter starttidspunkt. Endringer i metoden for utregning av støt. Ferdigstilte kravdokumentet sammen med gruppa og fikk sendt dette til veileder. La inn støtte for at kollisjonsdiagrammet kan vise støt for flere gjennomkjøringer samtidig. Flere tester utarbeidet.

Statusrapport Sander

Denne uken ble i all hovedsak brukt på å opprette siden hvor en administratorbruker ser oversikten over alle behandlingsenheter i systemet, og dermed også siden hvor nye behandlingsenheter kan registreres.

Statusrapport Trond

Denne uken brukte jeg på å bugfikse serversiden og skrive tester. Jeg satte også opp testdekning på serversiden for å få en oversikt over hva som blir testet. Annet enn det så skrev jeg innkalling til møte i neste uke og dokumentasjon.

Samlet statusrapport

Onsdagen gikk med til obligatorisk CCD-sesjon i annet fag. Ble ferdig med siste del av målene som ble satt for sprinten, og er klare for møte med SINTEF kommende mandag. Førsteutkastet av kravdokumentet ble sendt til veileder for tilbakemelding.

3.10 Uke 11

Uke 11 (9. mars)						
Aktiviteter	**Herman**	**Jørgen**	**Sander**	**Trond**	**Ukesum aktivitet**	**Total sum aktivitet**
Møter	3	3	2	2	10	36,5
Dokumentasjon	3	6	3	4	16	213,5
Hovedrapport	13	8	14	6	41	81
Frontend	0	3	11,5	0	14,5	193
Backend	12	2	0	15	29	209,5
Testing	0	7	0	0	7	42
	0	0	0	0	0	0
	0	0	0	0	0	0
Ukesum	**31**	**29**	**30,5**	**27**	**117,5**	
Akkumulert hittil i år	**173,5**	**187,5**	**214**	**200,5**		**775,5**

Statusrapport Herman

Møte med SINTEF Ocean AS på mandag. Startet opp med sprint 3, og jobbet hovedsakelig med å lage en funksjon som kan lese inn rådatafiler fra Sensorfisk. Fikk satt opp en disposisjon for hovedrapporten sammen med de andre på gruppa, og begynte å jobbe med teori-delen av rapporten.

Statusrapport Jørgen

Utvidet støtte for å sammenligne opptil 6 gjennomkjøringer etter ønske fra Sintef. Feilrettinger på enkelte grafer. Arbeidet med hovedrapport teori-del. Tester for klasse som håndterer formatering og behandling av trykk-data og kollisjonsdata.

Statusrapport Sander

Brukte uka på å opprette en side hvor det er mulighet for en bruker å velge hvilket filformat han/hun vil eksportere en gjennomkjøring som. Det gikk bedre å implementere checkboxer i underkomponenter denne gangen, da jeg hadde gjort det tidligere på siden som viser alle gjennomkjøringene. La også til muligheten for at brukeren kan gi et navn til en gjennomkjøring ved opplasting.

Statusrapport Trond

Deltok på møtet med SINTEF på mandag og planla hva vi hadde tenkt til å gjøre i denne sprinten. Vi begynte også å arbeide mye mer med hovedrapporten. Jeg oppdaterte upload endepunktet til å faktisk ta imot data og legge det inn i databasen. For å legge det inn i databasen lagde jeg en databasemanager som hadde metoder for å legge inn de ulike typene data. Resten av tiden denne uken gikk til fikse bugs som jeg fant i koden.

Samlet statusrapport

Hadde møte med SINTEF på mandag, og startet på sprint 3 etter dette. Arbeidet i sprinten omhandler opplasting av binærfil til nettsiden, samt eksportering av grafene til andre filtyper. Hovedfokus denne uken var å lage et endepunkt for opplasting av rådatafiler, metoder for å legge inn data fra JSON-filer i databasen, mulighet for å lagre rådatafilene lokalt på tjeneren, og en side for å velge filformat for eksporteringen. Det ble også lagt til en graf for temperatur da dette var ønsket av oppgavestiller.

3.11 Uke 12

Uke 12 (16. mars)						
Aktiviteter	**Herman**	**Jørgen**	**Sander**	**Trond**	**Ukesum aktivitet**	**Total sum aktivitet**
Møter	3	1	1	1	6	42,5
Dokumentasjon	4	5	5	5	19	232,5
Hovedrapport	2	0	0	0	2	83
Frontend	0	8,5	5,5	0	14	207
Backend	3,5	0	3	8,5	15	224,5
Testing	6,5	0	0	0	6,5	48,5
	0	0	0	0	0	0
	0	0	0	0	0	0
Ukesum	**19**	**14,5**	**14,5**	**14,5**	**62,5**	
Akkumulert hittil i år	**192,5**	**202**	**228,5**	**215**		**838**

Statusrapport Herman

Mandag til onsdag gikk bort til eksamen. Funksjonen for å lese inn rådatafiler hadde noen bugs som gjorde at den ikke klarte å lese alle filer. Resten av uka gikk dermed med til å rette opp feilene i metoden, samt å skrive enhetstester for den. Gjorde også endringer i kravdokumentet etter tilbakemelding fra veileder.

Statusrapport Jørgen

Mandag-onsdag gikk bort til eksamen. Restrukturerte eksempel-data brukt i tester for å forenkle endringer. Arbeidet med eksportering av CSV-fil for de ulike dataene. Mye tid tapt på feilsøking, da den nyeste versjonen av en pakke inneholdt feil.

Statusrapport Sander

Mandag til onsdag gikk bort grunnet eksamen i Systemtenking med økonomi. Resten av uka ble brukt til å se på hvordan man skulle opprette og laste ned en pdf-fil. Hadde noe trøbbel med dette, da det var uvisst hvordan man skulle kunne hente komponenter som ikke var rendret i systemet (altså de grafene som brukeren har mulighet til å velge, men ikke vises på siden).

Statusrapport Trond

Starten av uken brukte vi til å øve til eksamen i et annet emne. Torsdag og fredagen jobbet jeg med kravdokumentasjonen og hovedrapport. Innimellom så drev jeg også med å fikse noen feil i upload og eksempel dataen.

Samlet statusrapport

Mandag, tirsdag og onsdag denne uken gikk bort grunnet lesing til, og gjennomføring av eksamen i *TDAT3002 Systemtenking med økonomi.* Resten av uken ble det jobbet med hvordan de forskjellige grafene skulle eksporteres som henholdsvis PDF, CSV og PNG-filer. Hadde også møte med veileder på fredag angående hovedrapport og kravdokument.

3.12 Uke 13

Uke 13 (23. mars)						
Aktiviteter	**Herman**	**Jørgen**	**Sander**	**Trond**	**Ukesum aktivitet**	**Total sum aktivitet**
Møter	3	2	2	2	9	51,5
Dokumentasjon	2	3	2,5	2	9,5	242
Hovedrapport	7	6	12,5	6	31,5	114,5
Frontend	20,5	16	10,5	24	71	278
Backend	0	0	8	1	9	233,5
Testing	5	2	1	5	13	61,5
	0	0	0	0	0	0
	0	0	0	0	0	0
Ukesum	**37,5**	**29**	**36,5**	**40**	**143**	
Akkumulert hittil i år	**230**	**231**	**265**	**255**		**981**

Statusrapport Herman

Mesteparten av uka har blitt brukt til å fikse eksportering av grafer til PDF og PNG. Mye trøbbel for å få eksportering til PNG til å fungere, men klarte til slutt å få det til. Har også jobbet litt med teoridelen av hovedrapporten og kravdokumentet.

Statusrapport Jørgen

Feilrettinger for nedlasting av CSV-filer samt sammenligning av flere gjennomkjøringer. Noe galt med Babel-kompilatoren der tid gikk med på feilsøking. Sammenligning av gjennomkjøringer har nå støtte for ubegrenset antall gjennomkjøringer, men begrensning fysisk er satt til 6 etter godkjennelse fra Sintef. Arbeid med hovedrapport og kravdokument.

Statusrapport Sander

Brukte uka på å legge til mulighet for å laste ned en binærfil fra serveren. Dette innebærer endepunkt på server (med tester), samt serviceklasse og logikk på frontend. Har også brukt noe tid til å skrive på kravdokumentet og teoridelen av hovedrapporten.

Statusrapport Trond

Denne uken jobbet jeg med å implementere en metode for å redusere antall punkter som skal visualiseres på grafen. Med en slik metode kunne redusere antall punkter fra for eksempel 10000 til 3000 uten å miste formen på grafen. Testet noen forskjellige måter å gjøre dette på før jeg fant en algoritme som passet bra. Skrev også møteinnkalling for møtet i neste uke, og forberedte en presentasjon av produktet og videre arbeid for en kunde av SINTEF.

Samlet statusrapport

Innlesing av data fra en opplastet rådatafil ble fullført, og grafene henter nå sin data fra de opplastede filene. Vi ble også ferdige med eksportering av grafer til andre filformater (PDF, CSV og PNG), og har klart sprintmålet for sprint 3. Dette var "Opprette mulighet for å laste opp binærfil, samt eksportere graf og datafilter". Det ble også jobbet med teoridelen av hovedrapporten.

3.13 Uke 14

Uke 14 (30. mars)						
Aktiviteter	**Herman**	**Jørgen**	**Sander**	**Trond**	**Ukesum aktivitet**	**Total sum aktivitet**
Møter	5,5	5,5	5,5	5,5	22	73,5
Dokumentasjon	5	5	6	8	24	266
Hovedrapport	10,5	10,5	9,5	7,5	38	152,5
Frontend	0	5	8,5	13	26,5	304,5
Backend	18,5	6,5	5,5	0	30,5	264
Testing	0	0	0	0	0	61,5
	0	0	0	0	0	0
	0	0	0	0	0	0
Ukesum	**39,5**	**32,5**	**35**	**34**	**141**	
Akkumulert hittil i år	**269,5**	**263,5**	**300**	**289**		**1122**

Statusrapport Herman

Møte med SINTEF Ocean AS på mandag. Har jobbet med å implementere sikkerhet på backend. Dette innebar å lage funksjoner for hashing av passord, generering av tokens, og endepunkt for innlogging. Jobbet også videre med teoridelen av hovedrapporten, og startet på førsteutkastet av systemdokumentasjonen.

Statusrapport Jørgen

Opprettet e-post tjeneste med funksjon for å sende ut epost som kan tas i bruk ved for eksempel registrering av nye brukere eller tilbakestilling av passord. Arbeidet med systemdokumentasjon. I tillegg utvidet funksjonalitet ved opplasting av filer, slik at man kan gå direkte til en av de opplastede gjennomkjøringene.

Statusrapport Sander

Opprettet en side hvor administrator har mulighet til å opprette, slette og generelt ha oversikt over eksisterende brukere i systemet. Har også lagt til logikk for oppretting og sletting av brukere.

Statusrapport Trond

Utenom møtene denne uken så jobbet jeg med å opprette muligheten for å opprette nytt passord for en ny bruker, og det å kunne endre passord for innlogget bruker. Jeg jobbet også en god del med hovedrapport og systemdokumentasjon.

Samlet statusrapport

Denne uken hadde vi to møter. Det første møtet var på mandagen med SINTEF og det andre møtet var på onsdagen med SINTEF og en kunde fra SINTEF. Møtet på mandag involverte det vanlige sprint reviewet, mens møtet på onsdagen handlet om å presentere produktet så langt for å få litt tilbakemelding på det. Ellers så ble denne uken brukt på å opprette en side hvor en administratorbruker har mulighet til å opprette, endre og slette brukere. Det ble også opprettet en epost-tjeneste som sender ut epost til opprettede brukere. Videre ble det implementert sikkerhet på nettsiden, herunder hashing av passord, innlogging og utsending av tokens. Vi startet også på førsteutkastet av systemdokumentasjonen.

3.14 Uke 15

Uke 15 (6. april)						
Aktiviteter	**Herman**	**Jørgen**	**Sander**	**Trond**	**Ukesum aktivitet**	**Total sum aktivitet**
Møter	0	0	0	0	0	73,5
Dokumentasjon	0	5	6,5	0	11,5	277,5
Hovedrapport	0	0	0	0	0	152,5
Frontend	0	15	11,5	12	38,5	343
Backend	10,5	0	1	1	12,5	276,5
Testing	10	0	0	0	10	71,5
	0	0	0	0	0	0
	0	0	0	0	0	0
Ukesum	**20,5**	**20**	**19**	**13**	**72,5**	
Akkumulert hittil i år	**290**	**283,5**	**319**	**302**		**1194,5**

Statusrapport Herman

Jobbet videre med sikkerhet på backend. Har opprettet noen nye endepunkter som trengtes i forbindelse med innlogging/utlogging, og lagt til autentisering på mange av endepunktene. Tok påskeferie torsdag til søndag.

Statusrapport Jørgen

Påskeferie torsdag til søndag denne uken. Forbedringer og feilrettinger ved filopplastning, i tillegg til å arbeide med hovedrapport teori-del.

Statusrapport Sander

Jobbet kun mandag til onsdag denne uken på grunn av påskeferie. Har lagt til mulighet for å zoome inn- og ut på en graf, samt jobbet med innloggingssiden. Opprettet også en axios interceptor som sjekker om en bruker har gyldig access token, og hvis ikke - om brukeren har gyldig refresh token.

Statusrapport Trond

Dette var en kort uke. På mandag la jeg til funksjonalitet for å legge til notater for ulike gjennomkjøringer. Tirsdag så fikset jeg noen algoritmer for visualisering på klientsiden og endret litt på strukturen i DataView. Onsdag jobbet jeg med systemdokumentasjonen.

Samlet statusrapport

Denne uken ble det kun jobbet mandag-onsdag grunnet påskeferie. Vi fortsatte denne uken med implementering av sikkerhet på nettsiden, herunder sjekking av hvilken bruker som er innlogget slik at en bruker ikke kan hente ut data den ikke skal ha tilgang til. Det ble også implementert zooming av graf, oversikt over opplastede filer, og mulighet for å endre en bruker.

3.15 Uke 16

Uke 16 (13. april)						
Aktiviteter	**Herman**	**Jørgen**	**Sander**	**Trond**	**Ukesum aktivitet**	**Total sum aktivitet**
Møter	0	0	0	0	0	73,5
Dokumentasjon	14,5	14,5	21	10	60	337,5
Hovedrapport	0	0	0	0	0	152,5
Frontend	7	1	7,5	16	31,5	374,5
Backend	5	0	1	0	6	282,5
Testing	2	10,5	0	7	19,5	91
	0	0	0	0	0	0
	0	0	0	0	0	0
Ukesum	**28,5**	**26**	**29,5**	**33**	**117**	
Akkumulert hittil i år	**318,5**	**309,5**	**348,5**	**335**		**1311,5**

Statusrapport Herman

Har nå implementert autentisering på alle endepunkter på backend. La også inn funksjonalitet for å kunne registrere passord ved oppretting av bruker, og tilbakestille passordet ved glemt passord. De siste dagene ble brukt til å fullføre førsteutkastet av systemdokumentasjonen og sende det til veileder.

Statusrapport Jørgen

Arbeidet med dokumentasjon størsteparten av uka. Ble også gjennomført 6 brukertester. Analysere og kompilere data fra brukertester.

Statusrapport Sander

Mesteparten av uken gikk med på å implementere jsdoc i prosjektet slik at kildekode kunne dokumenteres. Måtte da også gå over kommentarene til alle klassene og de mest relevante funksjonene og gjøre om disse til jsdoc-kompatible kommentarer. Noe av uken gikk også med til skriving på systemdokumentet.

Statusrapport Trond

Denne uken hadde vi bestemt oss for å gjennomføre brukertester så jeg var med på å utføre 6 brukertester i løpet av uken. Annet enn det så begynte jeg med å rydde litt opp på frontend, endret navn på noen komponenter og fikset slik at de fleste komponentene ble responsive. Samtidig så fikset jeg logg ut logikken og skrev innkalling til møtet i neste uke.

Samlet statusrapport

Denne uken var hovedfokus å få ferdig et førsteutkast av systemdokumentasjonen og sende dette til veileder. Det ble også jobbet med gjenstående oppgaver fra sprint-backlogen. Dette omhandlet blant annet å sikre at kun innloggede brukere har tilgang til nettsiden, og legge til mulighet for endring av passord. Det ble også gjennomført brukertester på ansatte ved SINTEF Ocean AS og en av deres kunder.

3.16 Uke 17

Uke 17 (20. april)						
Aktiviteter	**Herman**	**Jørgen**	**Sander**	**Trond**	**Ukesum aktivitet**	**Total sum aktivitet**
Møter	2	2	2	1	7	80,5
Dokumentasjon	1	16	2	0	19	356,5
Hovedrapport	30	0	27,5	8	65,5	218
Frontend	1	1	2,5	24	28,5	403
Backend	0	2	1	0	3	285,5
Testing	0	13,5	0,5	1	15	106
	0	0	0	0	0	0
	0	0	0	0	0	0
Ukesum	**34**	**34,5**	**35,5**	**34**	**138**	
Akkumulert hittil i år	**352,5**	**344**	**384**	**369**		**1449,5**

Statusrapport Herman

Møte med SINTEF Ocean AS på mandag. Uka har stort sett gått med til å jobbe på hovedrapport, hovedsakelig innledning og resultater. Lite jobbing med kode denne uka, kun fiksing av noen små feil på frontend.

Statusrapport Jørgen

Møte med Sintef på mandag. Noen feilrettinger i kode, ellers dokumentasjon i hovedfokus. Arbeidet med å tolke data fra brukertestene, i tillegg til å generere en rapport fra brukertestene.

Statusrapport Sander

Hadde møte på mandag denne uken. Utenom dette har mesteparten av uken blitt brukt til skriving av hovedrapport. Brukte noe tid på prosjekthåndboka, hvor jeg la inn alt vi hadde så langt av møteinnkallinger, referater fra møter, timelister og statusrapporter.

Statusrapport Trond

Vi hadde et møte på mandag med SINTEF og veileder. Videre i uken jobbet jeg med logikk på frontend for å begrense tilgang til forskjellige sider på frontend. Samtidig restrukturerte jeg frontend og jobbet videre med å sjekke om alt var responsivt og og ting fungerte som de skulle.

Samlet statusrapport

Møte med SINTEF Ocean AS på mandag. Hovedfokus denne uken har vært hovedrapport og systemdokumentasjon. Det har også blitt utbedret enkelte feil i systemet, og gjort noen endringer på utforming.

3.17 Uke 18

Uke 18 (27. april)						
Aktiviteter	**Herman**	**Jørgen**	**Sander**	**Trond**	**Ukesum aktivitet**	**Total sum aktivitet**
Møter	1,5	0	1,5	1,5	4,5	85
Dokumentasjon	12,5	5,5	7,5	0	25,5	375
Hovedrapport	14	21	18,5	0	53,5	278,5
Frontend	0	0	0	39	39	442
Backend	3	0	5	0	8	293,5
Testing	0	7	0,5	0	7,5	113,5
	0	0	0	0	0	0
	0	0	0	0	0	0
Ukesum	**31**	**33,5**	**33**	**40,5**	**138**	
Akkumulert hittil i år	**383,5**	**377,5**	**417**	**409,5**		**1587,5**

Statusrapport Herman

Denne uka har det hovedsakelig blitt jobbet med resultatdelen av hovedrapporten, i tillegg til litt systemdokumentasjon. Det har også blitt gjort noen små endringer på backend for å rette opp i feil. Deltok på møtet med IT-avdelingen til SINTEF Ocean AS på tirsdag.

Statusrapport Jørgen

Feilrettinger i funksjonen som finner starttid for en gjennomkjøring, der den var for følsom på registrering av trykkfall. Arbeidet med hovedrapport resultatdel, både for brukertester, databehandling og utviklingsmetodikk.

Statusrapport Sander

Har i all hovedsak jobbet med resultatdelen av hovedrapporten, men har også fortsatt med å legge inn ting i prosjekthåndboka. Deltok på møtet med IT-avdelingen til SINTEF Ocean AS på tirsdag. Etter dette ble vi enige om å bruke en HTTPS-tjener med selvsignerte sertifikater for å teste at det faktisk virker, så implementerte dette i systemet på onsdag og torsdag.

Statusrapport Trond

Denne uken fortsatte jeg med å sjekke om alt fungerte som de skulle og fikset responsivness. Annet enn det så ryddet jeg opp litt på alle sider på frontend og fikset de fleste feilene jeg fant mens jeg holdt på i de komponentene. Jeg var også med på møtet på tirsdag med IT ved SINTEF for å diskutere litt om overleveringen av prosjektet, samt en kjapp gjennomgang av hvordan vi hadde bygget og prosjektet. I slutten av uken sendte jeg en innkalling til møte i neste uke.

Samlet statusrapport

På tirsdag hadde vi møte med IT-avdelingen til SINTEF Ocean AS. Her ble det blant annet diskutert hva som skal gjøres med koden etter prosjektets slutt, og hvilken informasjon som skal være med under “installasjon og kjøring” i systemdokumentasjonen. Resten av uka ble brukt til å jobbe med hovedrapporten og rette opp i små feil som har blitt funnet på nettsiden.

3.18 Uke 19

Uke 19 (4. mai)						
Aktiviteter	**Herman**	**Jørgen**	**Sander**	**Trond**	**Ukesum aktivitet**	**Total sum aktivitet**
Møter	2	1	2	2	7	92
Dokumentasjon	8,5	9,5	3	10	31	406
Hovedrapport	4	3	17	2	26	304,5
Frontend	6,5	13,5	11	20,5	51,5	493,5
Backend	11	8,5	6	3,5	29	322,5
Testing	2	0	0	0	2	115,5
	0	0	0	0	0	0
	0	0	0	0	0	0
Ukesum	**34**	**35,5**	**39**	**38**	**146,5**	
Akkumulert hittil i år	**417,5**	**413**	**456**	**447,5**		**1734**

Statusrapport Herman

Tirsdagen gikk stort sett med til møter, både med veileder og SINTEF Ocean AS. Resten av uka har stort sett blitt brukt til å rette opp i diverse feil i koden på frontend og backend, samt å få alle tester til å fungere. Det har også blitt jobbet med å oppdatere systemdokumentasjon, og skrevet litt på hovedrapport. Fredagen ble brukt til å builde og deploye kildekoden.

Statusrapport Jørgen

I begynnelsen av uka ble det arbeidet med dokumentasjon og prosjekthåndbok, i tillegg til møte med SINTEF. La også til funksjonalitet for å skjule gjennomkjøringer som mangler data i tillegg til diverse visning- og rettighetsendringer når man er innlogget som bedriftsbruker. Deretter ble det arbeidet med feilretting diverse steder og forberedelser til å deploye kildekoden.

Statusrapport Sander

Startet denne uken med å skrive på resultat- og diskusjonskapitlet på mandag og tirsdag. På tirsdag deltok jeg også på møtet med SINTEF. Onsdag og torsdag ble brukt til å fikse feil i systemet. Fikset småting som å gi brukeren tilbakemelding ved feil innfylling av skjema på flere forskjellige sider. Brukte nesten hele fredag på å finne ut hvordan man builder og deployer prosjektet når det er ferdig.

Statusrapport Trond

I starten av denne uken gjorde jeg meg ferdig med den generelle feilfiksingen og skrev et dokument om framtidige utvidelser i forberedelse til møtet på tirsdag. Videre i uken har jeg holdt på med dokumentasjon og hovedrapport. Torsdag og fredag ble brukt på å gjøre klar til å bygge prosjektet, og litt generell rydding i koden.

Samlet statusrapport

Tirsdagen hadde hele gruppa møte med SINTEF Ocean AS, og deretter veileder. Resten av uka har hovedsakelig blitt brukt til å jobbe med hovedrapport og dokumentasjon, samt å gjøre kildekoden klar for levering. Fredagen ble brukt til å builde og deploye hele prosjektet

3.19 Uke 20

Uke 20 (11. mai)						
Aktiviteter	**Herman**	**Jørgen**	**Sander**	**Trond**	**Ukesum aktivitet**	**Total sum aktivitet**
Møter	1	1	1	1	4	96
Dokumentasjon	0	0	9,5	3	12,5	418,5
Hovedrapport	32,5	33	26,5	32	124	428,5
Frontend	0	0	0	1	1	494,5
Backend	0	0	0	1	1	323,5
Testing	4	3	0	3	10	125,5
	0	0	0	0	0	0
	0	0	0	0	0	0
Ukesum	**37,5**	**37**	**37**	**41**	**152,5**	
Akkumulert hittil i år	**455**	**450**	**493**	**488,5**		**1886,5**

Statusrapport Herman

Møte med SINTEF Ocean AS sammen med resten av gruppa. Resten av uka gikk stort sett med til å skrive ferdig innholdet i hovedrapporten. Noen timer gikk med til å fikse tester på frontend.

Statusrapport Jørgen

Møte med SINTEF Ocean AS sammen med resten av gruppa. Resten av uka gikk med til arbeid på hovedrapport, i hovedsak diskusjon og konklusjon.

Statusrapport Sander

Hadde møte med SINTEF og resten av gruppa på mandag. Resten av uka ble brukt på å gjøre ferdig innholdet i hovedrapporten. Her har jeg skrevet mest på kapittel 4.3 - administrative resultater og 5.3 diskusjon om disse. Har også blant annet skrevet forord og sammendrag.

Statusrapport Trond

Denne uken startet med et møte med SINTEF for å vise frem det produktet slik det kom til å bli levert. Resten av uken gikk til å skrive ferdig resultater og diskusjon i hovedrapporten og litt annen dokumentasjon.

Samlet statusrapport

Vi har denne uka hatt et siste møte med SINTEF før innlevering av prosjektet. Vi har også jobbet videre mot ferdigstilling av hovedrapporten. Det som har stått igjen er hovedsakelig kapitlene om diskusjon og konklusjon, men det var også noe som gjenstod på resultatdelen.

3.20 Uke 21

Uke 21 (18. mai)						
Aktiviteter	**Herman**	**Jørgen**	**Sander**	**Trond**	**Ukesum aktivitet**	**Total sum aktivitet**
Møter	0	0	0	0	0	96
Dokumentasjon	0	0	0	0	0	418,5
Hovedrapport	20	20	20	20	80	508,5
Frontend	0	0	0	0	0	494,5
Backend	0	0	0	0	0	323,5
Testing	0	0	0	0	0	125,5
	0	0	0	0	0	0
	0	0	0	0	0	0
Ukesum	**20**	**20**	**20**	**20**	**80**	
Akkumulert hittil i år	**475**	**470**	**513**	**508,5**		**1966,5**

Statusrapport Herman

Uka har gått med til å ferdigstille hovedrapport og all vedlagt dokumentasjon, og forberede innlevering av prosjektet.

Statusrapport Jørgen

Denne uken gikk med på å ferdigstille rapporten og klargjøre for innlevering. Få vedlagte dokumenter inn i hovedrapport på en korrekt måte.

Statusrapport Sander

Har denne uken jobbet med å få satt sammen alle dokumenter og gjøre alt klart til innlevering

Statusrapport Trond

Vi hadde bare 3 dager denne uken før vi måtte levere bacheloroppgaven. Derfor brukte vi disse dagene til å klargjøre alle filene vi skulle levere inn.

Samlet statusrapport

Ferdigstillelse av hovedrapport slik at den ble klargjort for innlevering.

Utvikling av brukergrensesnitt og brukerfunksjoner for Sensorfisk

Scrum-dokumentasjon

Vedlegg F

Forfattere:

Herman Ryen Martinsen

Sander Nicolausson

Trond Jacob Rondestvedt

Jørgen Aasvestad

19.05.2020

Innholdsfortegnelse

1 Innledning

Gjennom arbeidet med prosjektet er det blitt benyttet utviklingsmetoden Scrum. Dette er en agil prosess som baserer seg på iterativt arbeid og begrensninger i dokumentasjonsomfang. Til tross for reduksjon i dokumentasjonomfang, er det enkelte dokumenter som er kritiske og nødvendige i en Scrum-prosess. I følgende dokument vil det bli presentert de ulike dokumentene som er utarbeidet i forbindelse med utviklingsprosessen. Det vil først bli presentert produkt-backlogen som inneholder all ønskelig funksjonalitet i systemet. Deretter vil de ulike sprintene bli presentert med relevant dokumentasjon i forbindelse med dette. Underveis i en Scrum-prosess vil det også arbeides iterativt med å utarbeide og forbedre dokumenter, og de som blir presentert videre er de siste reviderte utgavene.

2 Product-backlog

Tabell 2.1: Product-backlog for prosjektet. Inneholder systemkrav, deres prioritet og estimert antall timebruk.

Nr	Systemkrav	Prioritet	Estimert Timer
1	Som administrator ønsker jeg å opprette brukere for å gi nye brukere tilgang til systemet	Lav	50
2	Som bruker ønsker jeg å logge inn slik at jeg kan få tilgang til systemets funksjoner	Middels	60
3	Som bruker ønsker jeg å kunne endre passordet mitt slik at sikkerheten forbedres	Lav	30
4	Som bruker ønsker jeg å velge behandlingsenheter slik at jeg kan bytte data som skal visualiseres	Middels	30
5	Som bruker ønsker jeg å ha mulighet til å filtrere dataene slik at jeg kan fjerne uønsket støy fra målingene	Høy	150
6	Som bruker ønsker jeg at all historiske data lagres slik at jeg kan se gamle gjennomkjøringer	Høy	60
7	Som bruker ønsker jeg å eksportere grafiske filer og datafiler slik at jeg kan bruke de utenfor systemet	Middels	150
8	Som bruker ønsker jeg å ha mulighet til å se to forskjellige gjennomkjøringer opp mot hverandre slik at jeg kan sammenligne de forskjellige gjennomkjøringene	Middels	80
9	Som bruker ønsker jeg å slette gjennomkjøringer slik at jeg kan fjerne unødvendig eller dobbeltlagret data	Lav	30
10	Som administrator ønsker jeg å kunne legge inn rådata fra gjennomkjøringer slik at dataen lagres og kan brukes senere	Høy	200

11	Som bruker ønsker jeg å ha flere grafiske fremstillinger av dataen slik at flere aspekter av den mekaniske avluseren kan vurderes	Høy	120
12	Som administrator ønsker jeg å ha mulighet til å legge inn nye kunder i systemet slik at data fra en gjennomkjøring kan knyttes til kunden det tilhører	Middels	60
13	Som administrator ønsker jeg å kunne endre informasjonen som ligger inne på en kunde slik at den informasjonen som er registrert er oppdatert.	Lav	30
14	Som administrator ønsker jeg å kunne slette en kunde slik at kunden ikke lenger eksisterer i systemet	Lav	20
15	Som bruker ønsker jeg at posisjonen til sensorfisken estimeres vha. magnetbånd/Kalmanfilter slik at jeg har mulighet til å se hvor i prosessen forskjellige hendelser skjer	Lav	50

3 Sprint 1

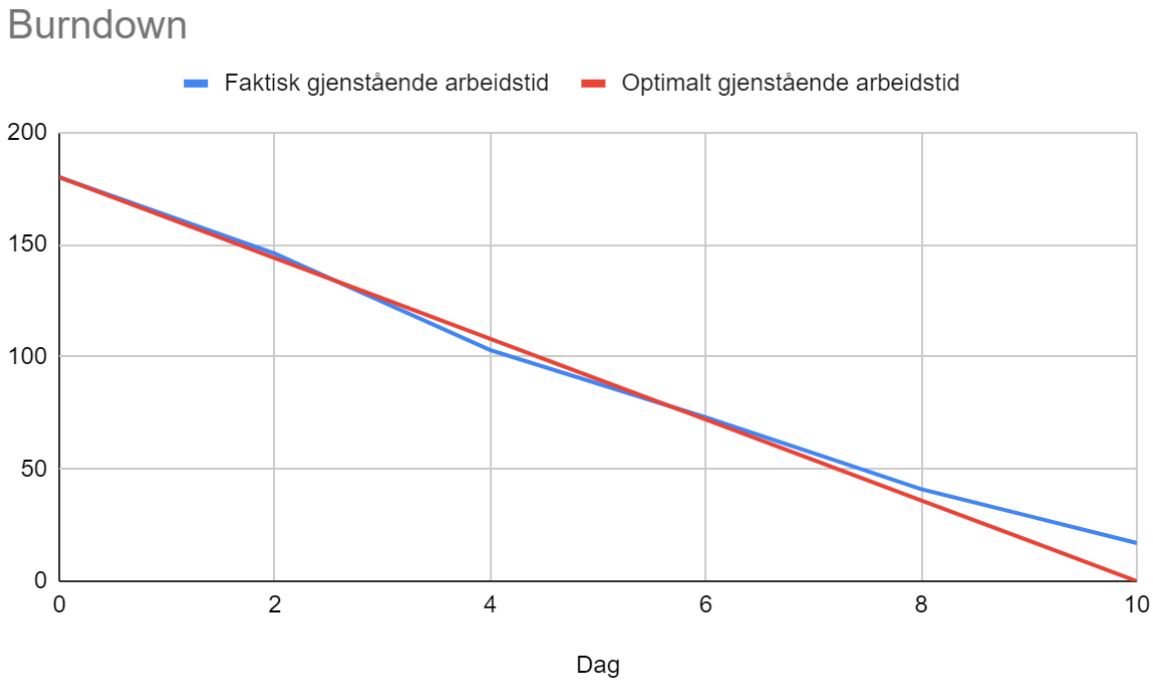
I dette kapittelet presenteres dokumenter utarbeidet i forbindelse med sprint 1. Sprinten foregikk over en tidsperiode på 2 uker fra 27.01 til 07.02.

3.1 Sprint-backlog

Tabell 3.1.1: Sprint-backlog for sprint 1. Oversikt over arbeidsoppgaver og deres tilhørende systemkrav, med estimert innsats og gjenstående arbeid.

Sprintmål: Opprette en side for grafisk fremstilling av data fra ulike gjennomkjøringer, hvor dataene hentes ut fra en database.							
Systemkrav fra produkt backlog	**Sprint 1 arbeidsoppgave**	**Originalt estimerte innsats**	**Gjenstående tid etter dag:**				
			2	**4**	**6**	**8**	**10**
11. Som bruker ønsker jeg å ha flere grafiske fremstillinger av dataen slik at flere aspekter av den mekaniske avluseren kan vurderes	Side som framstiller dataen, utgangspunkt i Wireframes	42	38	30	20	10	0
	Utforme graf for trykk	20	18	14	6	2	0
	Utforme graf/framstilling av støt	32	30	25	25	20	15
	Opprette service-klasser for å hente ut data fra backend	16	13	5	2	1	0
	Skrive tester for service-klassene	10	6	3	2	0	0
6. Som bruker ønsker jeg at all historiske data lagres slik at jeg kan se gamle gjennomkjøringer	Lage databasemodell	20	10	0	0	0	0
	Lage SQL-scripts for å opprette database.	8	6	4	4	0	0
	Opprette relevante endepunkter for å hente ut data fra databasen på backend	14	10	7	4	2	0
	Skrive tester for relevante endepunkter	8	5	5	4	4	2
	Opprette testdata for databasen	10	10	10	6	2	0
	SUM	**180**	**146**	**103**	**73**	**41**	**17**

3.2 Burndown-chart



Figur 3.2.1: Burndown-chart for sprint 1. Viser forskjell mellom faktisk- og optimalt gjenstående arbeid.

3.3 Retrospektiv

Tabell 3.3.1: «What went well» for sprint 1.

What went well	**Votes**
Vi fikk fullført målet vårt om å vise frem en trykkgraf	2
Vi har laget mye av infrastrukturen vi trenger senere i prosjektet	2
Vi arbeidet godt og jevnt med tanke på at vi hadde mye annet som også foregikk	0
Bruk av git har gått bra (branching, merging osv)	0
God kommunikasjon og samarbeid.	0
Bra fordeling av frontend/backend.	0
Oppsett av sprint-backlog og burndown er konkret og enkel å følge.	0

Folk arbeider når de skal og vet hva de skal gjøre, ikke alt for mange avsporinger.	0
Vi tok oss tid til å lage en god plan for sprinten.	0

Tabell 3.3.2: «What can be improved» for sprint 1.

What can be improved	**Votes**
Enda tydeligere lunsjtidspunkt. Også andre pauser? 5-minuttere? Øker effektivitet?	0
Tenke igjennom og skrive ned hvorfor vi bruker diverse teknologier, biblioteker.	0
Jeg synes vi burde fikse litt på databasen på serversiden, og implementere sequelize på en annen måte. Dette kan gjøre det enklere å teste	0
Alle servertestene må forbedres.	0
Tydeligere sprintmål. Konkretisert.	0
Mer tid til å tenke gjennom retrospektiv, og gjennomføre denne rett etter Sprint Review.	0
Sette klare oppmøtetidspunkt. Følge disse? Hehe. Straff? Nei det er kjipt kanskje	0
Oppdatere sprintbacklog og burndownchart oftere (github og docs)	0
Jobbe mer strukturert med dokumentasjonen fremover	0
Mer konkrete sprintoppgaver? Enkelte av oppgavene var kanskje litt for store	0

Tabell 3.3.3: «Action items» for sprint 1.

Action Items	**Votes**
Endre databasesystemet slik at det fungerer bedre med testing.	0
Dersom du tar i bruk ny teknologi eller rammeverk. Gjør research og dokumenter hvorfor du bruker det.	0
Sette av onsdager til dokumentasjon.	0
Lunsj 11.30-12.00. Avtale felles pauser utenom dette også, minimum hver andre time.	0
Sette av siste 15 min av en arbeidsdag til å oppdatere timelister og burndownchart.	0
Prøve å konkretisere sprintmålet og sprintoppgavene.	0
Bli flinkere til å møte til oppsatt tid. Hvis noen ikke kan møte til tida bør dette varsles om senest kvelden før.	0
Gjennomføre retrospektiv rett etter Sprint Review. Alle bør tenke igjennom hva som gikk bra/kan forbedres på forhånd.	0

4 Sprint 2

I dette kapittelet presenteres dokumenter utarbeidet i forbindelse med sprint 2. Sprinten foregikk over en tidsperiode på 3 uker fra 17.02 til 09.03.

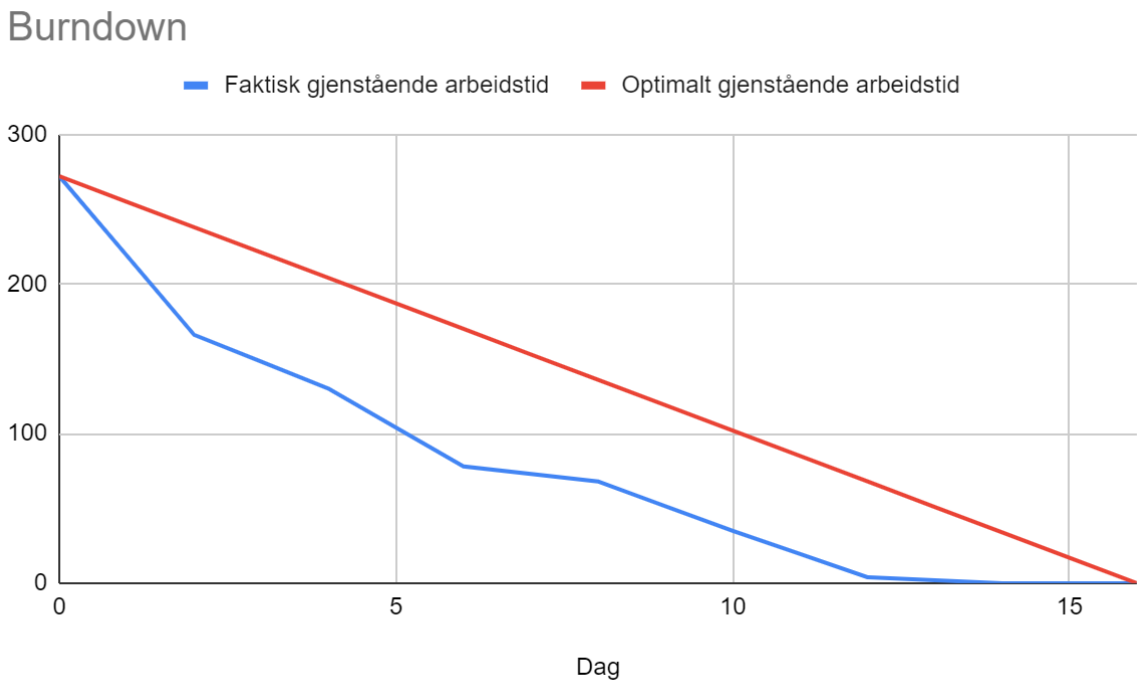
4.1 Sprint-backlog

Tabell 4.1.1: Sprint-backlog for sprint 2. Oversikt over arbeidsoppgaver og deres tilhørende systemkrav, med estimert innsats og gjenstående arbeid.

SPRINTMÅL: Opprette navigerbar nettside med mulighet for å sammenligne gjennomkjøringer, filtrere trykk-grafer og vise histogram for støt.									
Systemkrav fra produkt backlog	**Sprint 2 arbeidsoppgave**	**Originalt estimerte innsats**	**Gjenstående tid etter dag:**						
			2	**4**	**6**	**8**	**10**	**12**	**14**	**16**
11. Som bruker ønsker jeg å ha flere grafiske fremstillinger av dataen slik at flere aspekter av den mekaniske avluseren kan vurderes	Funksjon for utregning av støt	32	32	32	30	26	14	6	2	0
	Utforme graf/histogram for støt/g-krefter	16	16	16	16	16	8	4	1	0
	Funksjon for utregning av G-krefter	32	30	8	8	8	4	2	0	0
	Funksjon for å finne startpunkt på trykkgraf / sette ned til 1 bar	24	8	8	4	4	2	1	0	0
	Enkel filtrering av trykkgraf	32	5	5	5	4	2	1	0	0
	Opprette en funksjon som finner passering av magnetbånd	32	16	10	10	10	5	2	1	0
8. Som bruker ønsker jeg å ha mulighet til å se to forskjellige gjennomkjøringer opp mot hverandre slik at jeg kan sammenligne de forskjellige gjennomkjøringene	Fikse "sammenligne gjennomkjøringer" knappen, slik at to gjennomkjøringer vises	16	15	12	0	0	0	0	0	0
	Legge til valgmulighet for hvilken gjennomkjøring den skal sammenlignes med	24	24	24	1	0	0	0	0	0
	Utforme oversikt over hvilken gjennomkjøring som skal vises (m/sammenligning)	32	20	15	4	0	0	0	0	0

								0		
	Opprette en navigerbar struktur	32	2	2	2	0	0	0	0	0
	SUM	**272**	**166**	**130**	**78**	**68**	**35**	**16**	**4**	**0**

4.2 Burndown-chart



Figur 4.2.1: Burndown-chart for sprint 2. Viser forskjell mellom faktisk- og optimalt gjenstående arbeid.

4.3 Retrospektiv

Tabell 4.3.1: «What went well» for sprint 2.

What went well	**Votes**
Git-merging gikk overraskende smertefritt, lite tidkrevende konflikter	1
Arbeidet effektivt, gjorde ferdig sprintoppgaver tidsnok	0
Mer effektiv jobbing nå som vi har blitt bedre til å ta pauser	0

Vi jobbet så effektivt at vi slapp opp for ting å gjøre mot slutten av sprinten	0
Mye funksjonalitet og navigerbarhet allerede på plass, slik at vi har en del å vise fram	0
Tiltakene fra forrige retrospektiv har stort sett blitt fulgt	0
Har fått gjort mye i løpet av sprinten	0

Tabell 4.3.2: «What can be improved» for sprint 2.

What can be improved	**Votes**
Jobb i egen branch, ikke develop	2
Glemte å føre burndown-chart på slutten av sprinten	1
Fortsatt ikke bruk av CI/CD	1
Har fortsatt ikke begrunnet hvorfor vi bruker de forskjellige bibliotekene/teknologiene vi gjør	0
Jobbe mer med hovedrapporten (sette av tid/ha en som er ansvarlig for fremdrift?)	0
Noen av tingene vi hadde planlagt tok mye mindre tid enn forventet, som gjorde at vi endte opp med litt lite ting å gjøre mot slutten. (Så kanskje bedre planlegging?)	0
Sette opp sprintmål som er konkret og beskrivende	0
Skrive tester underveis, ikke etterpå -> smidig utvikling	0
Glemmer av stand-up flere ganger	0
Ingen(?) skrev notater underveis for å forbedre retrospektiv, hvertfall ikke jeg personlig	0

Tabell 4.3.3: «Action items» for sprint 2.

Action Items	**Votes**
Sette opp CI/CD	0
Sett opp disposisjon på hovedrapport, slik at det blir enklere å jobbe på den. Sett av 2 timer hver dag til å jobbe på hovedrapporten	0
Sander er ansvarlig for fremdrift på hovedrapporten	0
Ingenting å gjøre? => Skriv hovedrapport	0
Hvis du fullfører en ny funksjon/nytt endepunkt, skriv test. (Samme hvis du oppdaterer ett endepunkt, oppdaterer test)	0

5 Sprint 3

I dette kapittelet presenteres dokumenter utarbeidet i forbindelse med sprint 3. Sprinten foregikk over en tidsperiode på 3 uker fra 10.03 til 30.03.

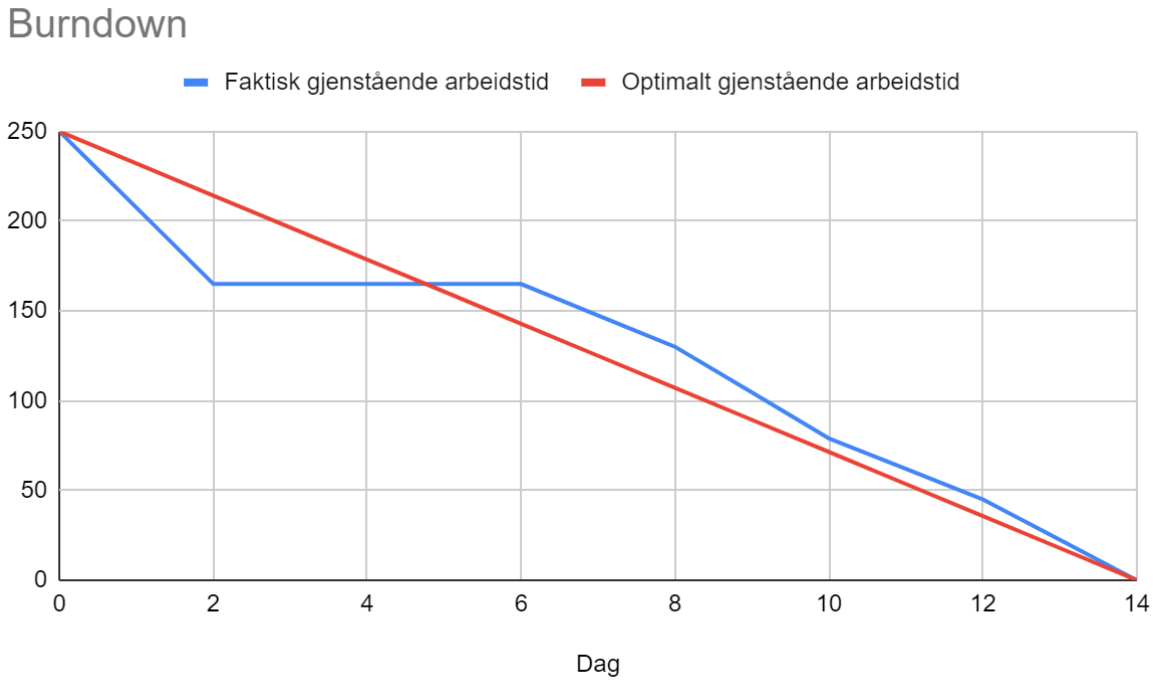
5.1 Sprint-backlog

Tabell 5.1.1: Sprint-backlog for sprint 3. Oversikt over arbeidsoppgaver og deres tilhørende systemkrav, med estimert innsats og gjenstående arbeid.

SPRINTMÅL: Opprette mulighet for å laste opp binærfil, samt eksportere graf og datafiler									
Systemkrav fra produkt backlog	**Sprint 3 arbeidsoppgave**	**Originalt estimerte innsats**	**Gjenstående innsats etter dag:**						
			2	**4**	**6**	**8**	**10**	**12**	**14**
Mulighet for opplastning av binærfil (10)	Service-klasse for å overføre fil til serveren	10	10	10	10	2	1	1	0
	Utforme sider for opplastning av gjennomkjøring	10	10	10	10	5	2	2	0
	Endepunkt for opplastning av binærfil	20	5	5	5	2	1	0	0
	Klasse som leser ut verdier fra binærfilen og oppretter JSON-objekter	35	20	20	20	15	5	5	0
	Metoder for å legge inn data fra json filer inn i databasen	15	10	10	10	6	4	2	0
	Lagre binærfilene lokalt på serveren	15	5	5	5	4	3	2	0
	Legge inn utregning av støt og posisjonsestimering av magnetbånd ved opplastning	15	10	10	10	6	5	2	0
Eksportere filer (7)	Utforme side for eksportering av fil, med ulike valgmuligheter	15	0	0	0	0	0	0	0
	Klasse som oppretter en pdf-rapport med aktuelle data	35	30	30	30	30	30	15	0
	Metode for å eksportere dataene til en bildefil (png)	25	20	20	20	20	20	10	0
	Metode for å eksportere dataene til en csv-fil	15	15	15	15	10	7	5	0
	Endepunkt for å hente ut binærfil	30	30	30	30	30	1	1	0
	Service-klasse for å hente binærfil fra serveren	10	10	10	10	10	1	1	0
	Sammenligne 6 gj.kjøringer	5	1	1	1	0	5	4	0

	Graf for temperatur	5	1	1	1	0	5	2	0
	SUM	**250**	**165**	**165**	**165**	**130**	**79**	**45**	**0**

5.2 Burndown-chart



Figur 5.2.1: Burndown-chart for sprint 3. Viser forskjell mellom faktisk- og optimalt gjenstående arbeid.

5.3 Retrospektiv

Tabell 5.3.1: «What went well» for sprint 3.

What went well	**Votes**
Endelig fått satt opp CI på GitHub, det var på tide!	0
Overgangen til "hjemmekontor" har fungert overraskende bra.	0
Bedre på arbeidsfordeling: Når noen ikke har noe å gjøre jobbes det med hovedrapport.	0
Nydelig git-merging, selv om flere til og med jobbet på samme fil	0
Vi fikk gjort det vi planla å gjøre.	0

Kommet i gang med hovedrapporten og fått skrevet en god del teori.	0

Tabell 5.3.2: «What can be improved» for sprint 3.

What can be improved	**Votes**
Har ikke satt av 2 timer hver dag til å jobbe med hovedrapport som ble bestemt retrospektiv for sprint 2.	8
Noterer ikke til retrospektivet underveis, lett å glemme når man kommer til slutten	0
Glemmer iblant å oppdatere burndown-chart.	0
Mangler en del tester for ny kode/funksjonalitet implementert denne sprinten	0

Tabell 5.3.3: «Action items» for sprint 3.

Action Items	**Votes**
Alle oppretter en tekstfil hvor de kontinerlig skriver ned punkter til neste retrospektiv.	0
Scrum Master (Herman) tar ansvar for at burndown chart oppdateres annenhver dag.	0
Gå gjennom tester som er skrevet etter sprinten er ferdig, og få gjort evt. tester som mangler så fort som mulig.	0
Få på plass et mer konkret forskningsspørsmål så fort som mulig.	0
Sette av mer tid til å jobbe med hovedrapporten fremover.	0

6 Sprint 4

I dette kapittelet presenteres dokumenter utarbeidet i forbindelse med sprint 4. Sprinten foregikk over en tidsperiode på 3 uker fra 31.03 til 20.04.

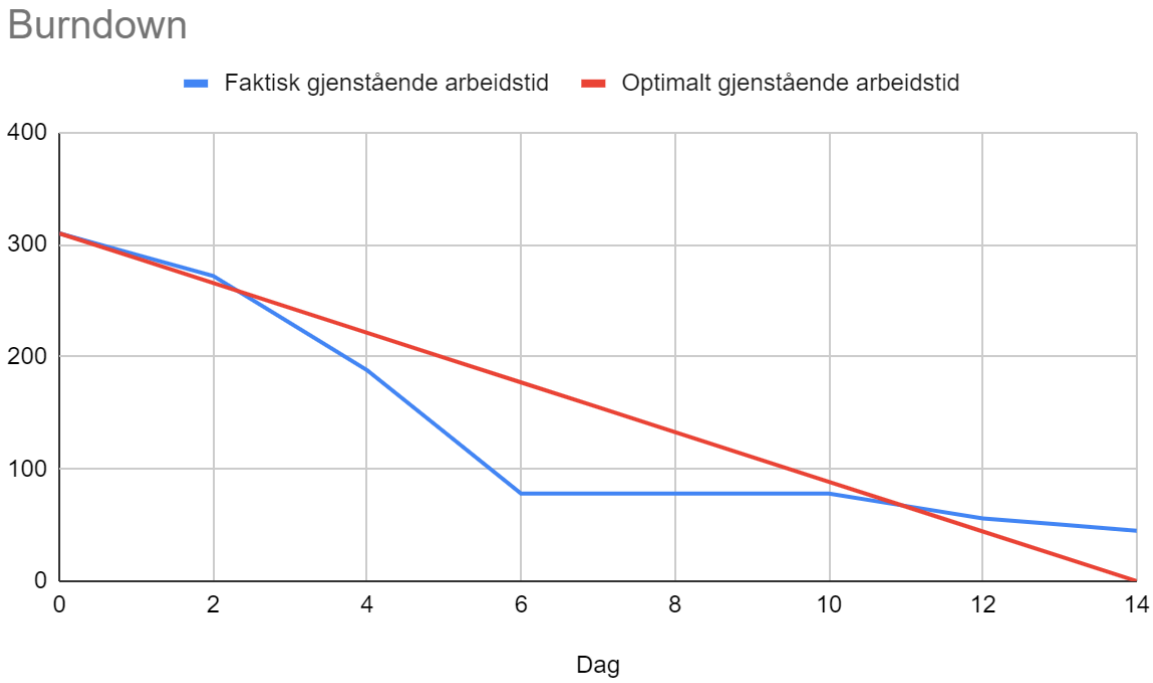
6.1 Sprint-backlog

Tabell 6.1.1: Sprint-backlog for sprint 4. Oversikt over arbeidsoppgaver og deres tilhørende systemkrav, med estimert innsats og gjenstående arbeid.

SPRINTMÅL: Implementere sikkerhet slik at brukere kun har tilgang til informasjon de har rettigheter til å se.									
Systemkrav fra produkt backlog	**Sprint 4 arbeidsoppgave**	**Originalt estimerte innsats**	**Gjenstående innsats etter dag:**						
			2	**4**	**6**	**8**	**10**	**12**	**14**
(1) Som administrator ønsker jeg å opprette brukere for å gi nye brukere tilgang til systemet	Utforme side for oppretting av og oversikt over brukere	20	15	2	2	2	2	1	0
	Opprette en epost-tjeneste for utsending av lenke	15	15	4	0	0	0	0	0
	Side for å opprette passord for ny bruker	5	2	1	1	1	1	0	0
	Service-klasse for å opprette og endre brukere (1 og 3)	5	5	0	0	0	0	0	0
	Endepunkt for å opprette bruker	5	5	1	1	1	1	0	0
	Opprette hashing av passord på klientside (1, 2 og 3)	20	20	20	0	0	0	0	0
	Opprette hashing av passord på serverside (1, 2 og 3)	20	0	0	0	0	0	0	0
(2) Som bruker ønsker jeg å logge inn slik at jeg kan få tilgang til systemets funksjoner	Sikre alle sider på frontend slik at kun innloggede brukere har tilgang	20	20	20	10	10	10	5	0
	Sette opp middleware på serversiden som sjekker om brukeren er logget inn	5	5	5	5	5	5	0	0
	Opprette klasse som oppretter, lagrer og validerer tokens	20	20	10	0	0	0	0	0

	Implementere selve innloggingen på klient	10	10	3	1	1	1	0	0
	Endepunkt på serversiden for innlogging	15	15	5	0	0	0	0	0
(3) Som bruker ønsker jeg å kunne endre passordet mitt slik at jeg kan ta i bruk ønsket passord.	Opprette side for å endre passord	10	5	2	2	2	2	2	0
	Endepunkt for å endre bruker	15	15	15	5	5	5	0	0
(12) Som administrator ønsker jeg å ha mulighet til å legge inn nye bedrifter i systemet slik at data fra en gjennomkjøring kan knyttes til bedriften det tilhører	Endepunkt for å opprette bedrifter	10	10	0	0	0	0	0	0
	Service-klasse for å opprette bedrifter	5	5	0	0	0	0	0	0
	Sette opp kryptering ved overføring av data mellom klient og server	30	25	25	25	25	25	25	25
Disse vurderer vi om vi får tid til:	Oversikt over opplastede filer	15	15	10	2	2	2	2	0
	Fikse desimering av graf	15	15	15	0	0	0	0	0
	Zooming/panorering	20	20	20	2	2	2	1	0
	Notater til gjennomkjøringer	10	10	10	2	2	2	0	0
	Flytte grense for utregning av støt	20	20	20	20	20	20	20	20
	SUM	**310**	**272**	**188**	**78**	**78**	**78**	**56**	**45**

6.2 Burndown-chart



Figur 6.2.1: Burndown-chart for sprint 4. Viser forskjell mellom faktisk- og optimalt gjenstående arbeid.

6.3 Retrospektiv

Tabell 6.3.1: «What went well» for sprint 4.

What went well	**Votes**
Bra at vi har gjennomført brukertester	0
Har vært mer vekt på dokumentasjon.	0
Føring av burndown-chart har gått mye bedre enn tidligere!	0
Folk har vært flinke til å hoppe rett til dokumentasjon dersom de ikke har noe å gjøre	0
Fortsatt relativt smertefri git-merging.	0
Jobbet effektivt og bra med systemdokumentasjon	0
Vi har fått implementert gode sikkerhetsmekanismer på nettsiden	0

Tabell 6.3.2: «What can be improved» for sprint 4.

What can be improved	**Votes**
Det å møte 10 min for sent hver dag utgjør mye i lengden, og hindrer at vi kommer i gang på morgenen...	0
Fokusere enda mer på de viktigste funksjonene i applikasjonen, ikke bruke for mye tid på småting	0
Mange har fortsatt ikke ført retrospektiv-notater underveis.	0
Før push til develop har vi ikke sjekket om testene fungerer.	0
Hovedrapporten blir fortsatt litt "glemt bort"	0

Tabell 6.3.3: «Action items» for sprint 4.

Action Items	**Votes**
Alle møter kl 8 hver morgen	0
Sjekk om testene fungerer før det pushes til develop	0
Hovedrapporten skal være i hovedfokus fremover	0

Utvikling av brukergrensesnitt og brukerfunksjoner for Sensorfisk

Rapport fra brukertester

Vedlegg G

Forfattere:

Herman Ryen Martinsen

Sander Nicolausson

Trond Jacob Rondestvedt

Jørgen Aasvestad

23.04.2020

Innholdsfortegnelse

1 Innledning

I forbindelse med utvikling av web-applikasjonen for SINTEF OCEAN AS, har det blitt gjennomført brukertester, med den hensikt å forbedre sluttproduktet samt avdekke mangler eller problemområder i et tidlig stadium. Utviklingsteamet har i nåværende tidsrom kommet langt nok i prosessen til å gjennomføre de ønskede testene på en god måte. Brukertestene ble gjennomført med følgende mål:

1. Vurdere brukervennligheten til systemet.
2. Forbedre brukskvaliteten til systemet.

Ut fra de ovennevnte målene har det blitt utarbeidet forskningsspørsmål som ønskes besvart ved hjelp av testene. Forskningsspørsmålene har den hensikt å utdype de overordnete målene for testingen, slik at disse blir konkretiserte og målbare. Følgende forskningsspørsmål har blitt brukt:

- Hvor enkelt skjønner brukerne hva som er klikkbart?
- Hvor enkelt synes en bruker det er å laste opp/ned en binærfil?
- Klarer en bruker å eksportere gjennomkjøringer?
- Hvor enkelt klarer brukerne å navigere rundt i applikasjonen?

SINTEF OCEAN AS, her representert ved Sveinung Johan Ohrem, har anskaffet seks deltakere til brukertestingen, derav fire SINTEF-ansatte og to ansatte fra en av SINTEFs kunder. Alle deltakerne har lite kunnskap om systemet fra tidligere og flertallet har aldri sett nettsiden tidligere. I presentasjonen av de innledende spørsmålene er alder oppsatt med 10-års intervaller og yrke fjernet. Dette er gjort med den hensikt å anonymisere data slik testdeltakerne ble informert om på forhånd. Brukertestene ble gjennomført ved hjelp av videosamtale gjennom Microsoft Teams. I utgangspunktet var det tenkt at testlederen skulle dele skjerm og at deltakeren deretter skulle “ta kontroll” over skjermen slik at det kunne kjøres lokalt. På grunn av restriksjoner i funksjonalitet mellom ulike organisasjoner i Teams kunne ikke dette gjennomføres, så reserveløsningen ble iverksatt. Det ble dermed port-forwarding på ruteren slik at deltakerne kunne aksessere systemet i sin egen nettleser og deretter dele skjermen. Testene varte i omtrent 30 minutter og besto av tre hoveddeler: innledende spørsmål, oppgaver og avsluttende spørsmål. For hver av oppgavene ble det definert ønsket standard og suksesskriterium for å bedømme hvorvidt oppgaven er ferdig og ønskelig løst.

2 Resultater

Følgende kapittel presenterer resultater og produsert data ved brukertestene. Dette vil kun være aktuell rådata satt sammen, uten noen form for intervensjon eller drøfting av data. Resultatene som presenteres danner grunnlaget for å fastsette problemområder og eventuelle løsninger for systemet. Brukertestene var delt i tre hoveddeler: innledende spørsmål, oppgaver og avsluttende spørsmål, noe som også er tilfellet i presentasjon av resultatene.

For å forenkle presentasjonene av datamaterialet er det tatt i bruk enkelte forkortelser. På denne måten vil visualiseringen få forbedret oversiktlighet. Disse er som følger:

S1 - Testdeltaker fra SINTEF OCEAN AS, i dette tilfellet nr. 1.

K1 - Testdeltaker fra kunde av SINTEF, i dette tilfellet nr. 1.

2.1 Innledende spørsmål

Tabell 2.1: Viser oversikt over besvarelsene på innledende spørsmål for alle deltakere.

Testdeltaker	Alder	Bosted	Teknisk innsikt (selvrapportert)
S1	20-30	Trondheim	Bruker en del PC på jobb, diverse verktøy og programmer på PC daglig.
S2	20-30	Trondheim	Over gjennomsnittet
S3	50-60	Trondheim	Bruker PC hele tiden. Definitivt over gjennomsnittet.
S4	30-40	Trondheim	Bruker PC veldig mye, og litt robotikk.
K1	30-40	Frøya	Over gjennomsnittet.
K2	50-60	Frøya	Daglig bruk av PC, ellers ikke mye mer teknisk bakgrunn.

2.2 Oppgaver

2.2.1 Oppgave 1 - Innlogging

Ønsket standard: -
Suksesskriterium: Testdeltakeren får logget inn
Resultat:
Tabell 2.2.1: Resultater for innloggingsoppgave.

Testdeltaker	Ønsket standard	Suksess-kriterium	Notater
S1	-	✔	Knirkefritt.
S2	-	✔	Bruker noe lengre tid.
S3	-	✔	Logger inn greit.
S4	-	✔	Det går fint.
K1	-	✔	Logger inn enkelt.
K2	-	✔	Det går fint.

2.2.2 Oppgave 2 - Opplasting av gjennomkjøring

Ønsket standard: Gjennomkjøringen er lastet opp med korrekt informasjon
Suksesskriterium: Binærfilen er lastet opp til systemet
Resultat:
Tabell 2.2.2: Resultater for opplasting av gjennomkjøring.

Testdeltaker	Ønsket standard	Suksess-kriterium	Notater
S1	✔	✔	Fyller inn all informasjon greit. Ser at Stavanger ikke ligger i systemet, og ser muligheten for å legge til relativt raskt.
S2	✔	✔	Tror han har en fil og leter lenge etter denne. Prøver å laste opp 20+ filer, blir stoppet av testleder. Finner ikke Stavanger, og trenger tips fra testleder til å finne ut hvordan man legger til lokasjon.
S3	✔	✔	Trykker først på gjennomkjøringer. Finner deretter last opp data. Finner ikke Stavanger, og heller ikke knappen for å legge til lokasjonen. Pga. skjermstørrelse og oppløsning ble knappen mye mindre enn den vanligvis er. Den finnes med hjelp fra testleder.

S4	✔	✔	Kommer seg til last-opp siden med en gang. Ser at Stavanger ikke ligger inne og på tenke litt, men finner knappen for å legge til lokasjon.
K1	✔	✔	Finner frem til last opp data siden med en gang. Skjønner at man kan legge til egen lokasjon.
K2	✔	✔	Bruker litt tid på å finne ut hvordan man legger til lokasjon, men finner det ut uten hjelp.

2.2.3 Oppgave 3 - Sammenlign gjennomkjøringer

Ønsket standard: Klikker på "sammenlign"-knapp direkte på visningssiden
Suksesskriterium: Testdeltakeren får sammenliknet trykk-grafen
Resultat:
Tabell 2.2.3: Resultater for sammenligning av gjennomkjøringer

Testdeltaker	Ønsket standard	Suksess-kriterium	Notater
S1	✔	✔	Trykker på sammenlign-knappen. Trykker først direkte inn på en annen gjennomkjøring istedenfor å huke av og sammenligne.
S2	✔	✔	Sammenligner korrekt ved å trykke sammenlign-knappen samt huke av ønsket gjennomkjøring.
S3	✔	✔	Huker av riktig og sammenligner grafene.
S4	x	✔	Trykker først på gjennomkjøringer og deretter huker av to gjennomkjøringer derfra.
K1	✔	✔	Trykker sammenlign-knappen og huker av en ekstra gjennomkjøring.
K2	✔	✔	Trykker sammenlign-knappen og velger korrekt gjennomkjøring.

2.2.4 Oppgave 4 - Eksportere csv

Ønsket standard: Ingen feilklikk før man trykker "eksporter"-knappen
Suksesskriterium: Deltakeren får eksportert en csv-fil med ønsket data
Resultat:
Tabell 2.2.4: Resultater for eksportering av fil.

Testdeltaker	Ønsket standard	Suksess-kriterium	Notater
S1	✔	✔	Deltakeren klarer å eksportere filer med korrekt data.
S2	x	✔	Prøver å trykke på last-ned knappen. Video-samtalen ligger over og blokkerer "eksporter"-knappen, testleder tipser om å flytte vinudet. Deltakeren eksporterer så korrekte filer.
S3	x	✔	Tror ikke første det er mulig å eksportere alle grafene, trykker rundt på ulike grafer. Trykker på eksporter-knappen og får valgt korrekt informasjon.
S4	x	✔	Finner eksporter-knappen med en gang og eksporterer korrekt data. Får en feilmelding fra systemet, skyldes feil i systemet og er ikke brukerens feil.
K1	x	✔	Finner ikke hvordan man eksporterer graf, men skjønner det etter tips fra testleder.
K2	x	✔	Finner ikke eksporter-knappen med en gang.

2.2.5 Oppgave 5 - Legge til bedrift

Ønsket standard: Komme til "administrer bedrifter"-siden med maks ett feilklikk
Suksesskriterium: Deltakeren får lagt inn bedriften i systemet
Resultat:
Tabell 2.2.5: Resultater for å legge til bedrift.

Testdeltaker	Ønsket standard	Suksess-kriterium	Notater
S1	✔	✔	Fyller inn all informasjon og klikker på knappen.
S2	✔	✔	Finner frem til bedriftsoversikten. Prøver å endre bedrift istedenfor å legge til ny. Fortsetter å prøve å endre bedrift, testleder gir tips om det skal legges inn ny bedrift.
S3	✔	✔	Oppgave utført uten feil.

S4	x	✔	Bruker litt tid for å finne fram til bedriftsoversikten. Fyller inn informasjon kjapt og enkelt etterpå.
K1	✔	✔	Oppgave utført uten feil
K2	✔	✔	Har problemer med å finne "ny bedrift"-knappen da denne ligger nedenfor skjermen. Det skrolles nedover og knappen oppdages.

2.2.6 Oppgave 6 - Legge til bruker

Ønsket standard: Komme til "administrer brukere"-siden med maks ett feilklikk
Suksesskriterium: Deltakeren får lagt til bruker i systemet.
Resultat:
Tabell 2.2.6: Resultater for å legge til bruker.

Testdeltaker	Ønsket standard	Suksess-kriterium	Notater
S1	✔	✔	Oppgave utført uten feil.
S2	x	✔	Har litt problemer med å finne brukere-knappen.
S3	✔	✔	Oppgave utført uten feil.
S4	✔	✔	Finner brukeroversikten raskt og fyller inn korrekt informasjon.
K1	✔	✔	Oppgave utført uten feil.
K2	✔	✔	Oppgave utført uten feil.

2.2.7 Oppgave 7 - Endre informasjon behandlingsenhet

Ønsket standard: Deltakeren forstår hva som er redigeringsknappen uten hjelp
Suksesskriterium: Deltakeren får endret korrekt behandlingsenhet
Resultat:
Tabell 2.2.7: Resultater for endring av behandlingsenhet.

Testdeltaker	Ønsket standard	Suksess-kriterium	Notater
S1	✔	✔	Oppgave utført raskt og feilfritt.
S2	✔	✔	Oppgave utført uten feil.

S3	✔	✔	Finner behandlingsenheter og korrekt knapp med en gang.
S4	✔	✔	Finner fram korrekt og enkelt.
K1	✔	✔	Oppgave utført uten feil.
K2	✔	✔	Oppgave utført uten feil.

2.2.8 Oppgave 8 - Slette behandlingsenhet

Ønsket standard: -
Suksesskriterium: Behandlingsenheten er slettet fra systemet
Resultat:
Tabell 2.2.8: Resultater for sletting av behandlingsenhet.

Testdeltaker	Ønsket standard	Suksess-kriterium	Notater
S1	-	✔	Oppgave utført uten feil.
S2	-	✔	Oppgave utført uten feil.
S3	-	✔	Oppgave utført uten feil.
S4	-	✔	Oppgave utført uten feil.
K1	-	✔	Oppgave utført uten feil.
K2	-	✔	Oppgave utført uten feil.

2.2.9 Oppgave 9 - Sletting av gjennomkjøring og nedlasting

Ønsket standard: Gjennomkjøring slettes før nedlasting av binærfil
Suksesskriterium: Gjennomkjøringen er slettet og deltakeren har tilhørende binærfil
Resultat:
Tabell 2.2.9: Resultater for sletting av gjennomkjøring og nedlasting av binærfil.

Testdeltaker	Ønsket standard	Suksess-kriterium	Notater
S1	✔	✔	Sletting av gjennomkjøring går fint. Problemer med å finne filtreringsknappen, men klarer det uten hjelp. Klarer derfra å få vist slettede gjennomkjøringer.
S2	✔	✔	Sletting av gjennomkjøring går raskt. Finner ikke hvor man kan se slettede gjennomkjøringer. Etter tips fra

			testleder klarer vedkommende å finne den slettede gjennomkjøringen.
S3	✔	✔	Sletter gjennomkjøring, men finner ikke filtreringsknappen. Skjønner heller ikke at man kan vise slettede gjennomkjøringer derfra, og trenger tips fra testleder.
S4	✔	✔	Får slettet gjennomkjøring og finner filtreringsknappen og den slettede gjennomkjøringen umiddelbart.
K1	✔	✔	Trykker seg først inn på gjennomkjøringen. Skjønner etterpå at han må gå tilbake til oversikten og slette derfra. Klarer resten av oppgaven raskt.
K2	✔	✔	Sletter gjennomkjøringen raskt, og finner ut hvordan man viser slettede gjennomkjøringer etter litt leting.

2.2.10 Oppgave 10 - Logg ut

Ønsket standard: Deltaker bruker under 10 sekunder
Suksesskriterium: Deltaker er logget ut av systemet
Resultat:
Tabell 2.2.10: Resultater for å logge ut.

Testdeltaker	Ønsket standard	Suksess-kriterium	Notater
S1	✔	✔	
S2	✔	✔	
S3	✔	✔	
S4	✔	✔	
K1	✔	✔	
K2	✔	✔	

2.3 Avsluttende spørsmål

Tabell 2.3.1: Besvarelser på avsluttende spørsmål, relatert til helhetsinntrykk og forstyrrende elementer.

Testdeltaker	Helhetsinntrykk	Forstyrrende elementer?
S1	Bra helhetsinntrykk, det ser bra ut. Lett i bruk og forståelig.	Nei.
S2	Bra.	Ikke egentlig. Trykker på slett-knappen for gjennomkjøringer ved et uhell.
S3	Enkelt og greit, men var ikke lett å finne slettede gjennomkjøringer.	Trykker på slett-knappen for gjennomkjøringer ved et uhell.
S4	Lett navigerbart. Gjerne legg til flere ikoner, slik det er gjort på kontrollpanelet. Slettet en gjennomkjøring med et uhell.	Var noe vanskelig å finne slettede gjennomkjøringer, kanskje flytte dette til en annen plass.
K1	Så bra ut.	Kommer ikke på noe.
K2	Virker enkelt og greit. Vil alltid være litt tilvenning til et nytt system, føler at man finner det man ønsker etter å ha trykket seg litt rundt.	Man må scrolle ned hele tiden på mindre skjermer.

Tabell 2.3.2: Besvarelser på avsluttende spørsmål, relatert til vanskelige oppgaver og manglende funksjonalitet.

Testdeltaker	Noe som var vanskelig?	Manglende funksjonalitet?
S1	Det var ikke intuitivt å finne ut at man måtte markere begge gjennomkjøringer, men det gir muligens mer mening å gjøre det slik. Litt vanskelig å trykke på de små check-boksene.	Bedrifter og brukere er ikke nødvendig for kunder av Sintef.
S2	Nei ikke noe spesielt vanskelig.	Synes ikke det mangler noe særlig med nettstedet.
S3	Ingenting tungvint eller vanskelig.	Mulighet for å legge de forskjellige grafene over hverandre, for eksempel trykk og temperatur. Legge til ekstra informasjon om gjennomkjøringer. Legge til lokasjon i oversikten når man er inne i en gjennomkjøring.
S4	Vanskelig å finne slettede gjennomkjøringer.	Støtte for flere språk, gjerne engelsk.
K1	Slettede gjennomkjøringer var vanskelig å finne.	-
K2	Vanskelig å skjønne hvordan man byttet graf, kanskje legge til en beskrivelse på hvordan det gjøres.	Filtreringsknappen burde vært bedre markert. Kanskje legge til en dialog-boks når man holder musepeker over den. Legge til en linje som viser hvor man er i systemet.

3 Diskusjon

Diskusjonsdelen vil se nærmere på resultatene som ble funnet i løpet av brukertestene. Hensikten blir å analysere disse slik at man får tolket dataene og kan konkretisere hva som er blitt funnet ut. Resultatene vil blant annet bli sett på i lys av de fastsatte forskningsspørsmålene, samt brukes til å peke ut ulike problemområder for systemet, og eventuelle løsninger som kan være aktuelle.

3.1 Problemområder

Fra resultatene ser man at alle deltakerne har bestått alle oppgavene, altså nådd det ønskede suksesskriteriet. Dette trenger likevel ikke å bety at systemet er perfekt/feilfritt, da det kan være flere aspekter som spiller inn for denne høye suksessraten. For det første, vil vanskelighetsgraden på oppgavene være en faktor, og det kan være mulig å argumentere for at de skulle vært vanskeligere. I tillegg var det enkelte deltakere som trengte tips fra testleder for å få få oppfylt suksesskriteriet. Til tross for dette er det en tydelig enighet om at systemet er oversiktlig og enkelt i bruk, som kommer fram både underveis i testingen og gjennom de avsluttedende spørsmålene.

Fra oppgaveoversikten ser vi at flere av deltakerne støter på problemer på de samme områdene. Dette kan tyde på svakheter eller problemer i systemet som kan være verdt å se nærmere på. Vi ønsker derfor å kartlegge problemer som er mulig å oppdage ved den innledende analysen. Ut fra resultatene har vi kommet fram til følgende problemområder (nummerering tilsvarer ikke nødvendigvis prioritering):

1. **“Slett gjennomkjøring”-knappen antas å være der man trykker for å se gjennomkjøringen.** Både deltaker S2, S3 og S4 slettet gjennomkjøringer ved et uhell når de ville se grafene. Dette kan tyde på en svakhet i plasseringen av knappen, altså at den ikke er intuitiv for den gjennomsnittlige bruker. I tillegg er det ingen bekreftelses-dialog etter å ha trykket slett-knappen, noe som vil være ekstra sårbart hvis knappen har en noe unaturlig plassering.

2. **Vanskelig å finne slettede gjennomkjøringer.** Alle deltakerne brukte lang tid for å finne disse, samt at tre av dem fortalte at den ikke var intuitivt plassert. Enkelte

deltakere mente at denne ikke burde vært under "filtrering", men heller vært en egen knapp eller sjekkboks på oversiktsiden.

3. **"Last ned"-knappen ble forvekslet som eksportering av graf.** Det oppstår noe forvirring når deltakerne skal eksportere graf. Deltaker S2 trykker last-ned knappen og tror denne vil eksportere grafen, mens andre deltakere også sier at de i utgangspunktet vurderte å trykke denne knappen.

4. **På små skjermer må man stadig bla seg nedover på sidene.** I de fleste tilfellene ble sidene vist på korrekt og naturlig måte, men deltaker K2 hadde en noe mindre skjerm på sin bærbare datamaskin, og måtte derfor bla nedover på sidene for å se all informasjonen. Dette var spesielt problematisk på graf-oversikt siden, da knapper som "eksporter", "sammenlign" og "zoom ut" kom utenfor synsfeltet på skjermen.

3.2 Foreslåtte løsninger

Med utgangspunkt i de avdekte problemområdene er det utarbeidet en rekke forslag til forbedringer. Merk at dette er problemer avdekket i den innledende analysen og det kan dermed komme endringer i et senere tidsperspektiv etter en grundigere analyse og tilhørende rapport. Som nevnt tidligere er ikke nummereringen av problemene representativ for ønsket prioritetsrekkefølge, men de tas i bruk for å knytte løsningen direkte opp mot ønsket problem.

Følgende løsninger/forbedringer foreslås til utviklingsteamet:

1. **Knappene "Se gjennomkjøring" og "Slett gjennomkjøring" foreslås å bytte plass.** De fleste av testdeltakerne ønsket å trykke knappen på høyresiden for å gå videre til gjennomkjøringen, og det vil derfor være naturlig at disse bytter plass slik at slettingen foregår på venstre side. I tillegg foreslås det å legge til en popup-dialog som verifiserer at brukeren faktisk ønsker å slette den valgte gjennomkjøringen. Dette vil redusere sannsynligheten betydelig for å slette en uønsket gjennomkjøring.

2. **Anbefaler å synliggjøre slettede gjennomkjøringer, i form av en knapp/sjekkboks på gjennomkjørings-oversikten.** En mulighet er at man ved hjelp av dette vil få mulighet til å kun se slettede gjennomkjøringer (eventuelt i tillegg til ikke-slettede) og filtrere direkte på disse. Det vil dermed bli tydeligere å finne ut hvordan man ser de slettede gjennomkjøringene direkte fra oversikt-siden.

3. **Klargjøre betydningen av last-ned knappen.** Per nå er den litt diffus da det kun står "Last ned" på knappen. En eventuell løsning kan være å endre navnet til "Last ned binærfil" eller noe lignende, slik at det kommer tydelig fram at dette ikke vil være å eksportere graf eller noe lignende.

4. **Utvidet støtte for diverse skjermstørrelser.** Her vil det være naturlig å tilpasse siden forskjellig til ulike skjermstørrelser. Hovedprinsippet er at det vil være ønskelig å få graf-oversikten med tilhørende knapper i ett skjermbilde, uten at det skal være behov for å bla seg nedover på siden. Dette vil i hovedsak gjelde for datamaskiner, da mobiler/tablets sannsynligvis vil behøve noe scrolling.

3.3 Forskningsspørsmål

For å kunne stadfeste hvilke aspekter av systemet vi ønsket å se på gjennom brukertestene ble det utarbeidet fire forskningsspørsmål som nå kan besvares:

Hvor enkelt skjønner brukerne hva som er klikkbart?

- En gjennomgående trend er at brukerne har lite problemer med å skjønne hva som er klikkbart. Det var ingen registrerte feilklikk på en komponent/bilde som ikke var klikkbart i applikasjonen. Dette kan tyde på at de klikkbare komponentene på siden er utformet på en intuitiv måte.

Hvor enkelt synes en bruker det er å laste opp/ned en binærfil?

- Opplastning og nedlastning av binærfil er noe alle deltakerne klarer å gjennomføre uten hjelp av testleder. Det er rimelig å anta at dette er en intuitiv prosess når de klarer det første gang på egen hånd. Det er enkelte deltakere som trenger litt betenkningstid for å gjennomføre, men dette er gjerne en naturlig fremgangsmåte ved nye systemer.

Klarer en bruker å eksportere gjennomkjøringer?

- Under eksportering av gjennomkjøringer er det noe problemer som oppstår. Alle deltakerne får til slutt eksportert en gjennomkjøring, men enkelte trenger hjelp av testleder. Last ned-knappen forveksles som en eksportering-knapp, noe som virker forvirrende for enkelte av deltakerne. Dette er et oppdaget problemområde som kan løses ved å implementere den foreslåtte løsningen nevnt i foregående kapittel.

Hvor enkelt klarer brukerne å navigere rundt i applikasjonen?

- Deltakerne navigerer seg svært feilfritt rundt i applikasjonen. Med et beskjedent antall feilklikk og hurtige oppgaveløsninger er det mye som tyder på at navigasjonen er relativt intuitiv for førstegangs-brukere også. Dette gjenspeiles også i deltakernes svar på de avsluttende spørsmålene.

Martinsen, H. R., Nicolausson, S.,
Rondestvedt, T. J. & Aasvestad, J.

Utvikling av brukergrensesnitt og brukerfunksjoner for Sensorfisk

NTNU
Kunnskap for en bedre verden