Bacheloroppgave

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

Andreas Henriksen Dahle, Hadar Hayat,
Mathangi Pushparajah

Utvikling av en webapplikasjon for registrering og rapportering av brannsikkerhet

Bacheloroppgave i Dataingeniør
Veileder: Muhammad Ali Norozi
Mai 2023

NTNU
Kunnskap for en bedre verden

Andreas Henriksen Dahle, Hadar Hayat, Mathangi Pushparajah

Utvikling av en webapplikasjon for registrering og rapportering av brannsikkerhet

Bacheloroppgave i Dataingeniør
Veileder: Muhammad Ali Norozi
Mai 2023

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

NTNU
Kunnskap for en bedre verden

Sammendrag

Bacheloroppgaven handler om utviklingen av en webapplikasjon for Rambøll AS. Prosjektet gikk ut på å utvikle en applikasjon for registering og benchmarking av branntekniske løsninger, som skal effektivisere deres prosesser. For å effektivisere systemene skal applikasjonen tilby ulik funksjonalitet for prosjekthåndtering og eksportering.

For å lage ett brukervennlig og tilgjengelig design ble wireframing og brukertester tatt i bruk. Teamet hadde to iterasjoner av brukertester en på wireframe, og en på MVP. For at brukerne skulle være i målgruppen til applikasjonen ble branningeniører fra Rambøll testet på. Resultatetene fra brukertestene ble brukt for forbedring og videreutvikling av applikasjonen.

Teamet brukte Kanban for å effektivisere utviklingsprossessen. Måten dette var utført på var å bruke GitHub's issue boards som Kanbantavler for å holde orden på alle oppgaver som må utføres. Den også hjalp teamet til å være klar over hvem som jobber med de ulike oppgavene.

Resultatet er en webapplikasjon produksjonssatt(eng: deployed) på Rambølls Azure sky, den benytter seg av Microsofts innloggingsystem slik at ansatte kan benytte samme konto overalt. Ved å produksjonssette webapplikasjonen på Rambøll's eksisterende system, kan teamet fokusere på applikasjonens oppsett.

Den resulternede applikasjonen har funksjonalitet for registrering av prosjekter via ett skjema, samt redigering. Brukere kan også eksportere prosjektene til en utfylt word-mal for effektiv rapportskriving, og se statistikk over liknende prosjekter. Applikasjonen har også søkning, sortering og filtrering over alle prosjekter.

Abstract

The bachelor thesis is about the development of a web application for Rambøll AS. The project consisted of developing an application for the registration and benchmarking of analytical fire solutions, that are going to streamline their processes. To streamline systems the application is going to offer a variety of functionalities regarding project handling and exporting.

Wireframing and usability testing were key in making a user friendly and accessible design. The team held two iterations of usability tests, one for the wireframe and one for the MVP. To ensure that the usability tests were accurate the tests were conducted on Rambøll's fire safety engineers, as they are a part of the target audience. The results of the usability tests were used for improving and developing the application.

The team used Kanban to streamline the development process. This was done by using GitHub issue boards as Kanban boards, to keep track of all the tasks that need to be completed. It also assisted the team in finding out who was working on what.

The resulting web application was deployed in Rambøll's Azure Cloud, the use of Microsoft's login system assures that employees can use the same account everywhere. By deploying the web application to the existing systems, the team could focus on the configuration of the application.

The application has functionalities for registration of projects through a form, as well as editing projects. The users can also export a completed Word template to simplify report writing, and view statistics of similar projects. The application also has filtering, searching, and sorting on all available projects.

Forord

Bacheloroppgaven er skrevet i sammenheng med faget IDATT2900 ved Instituttet for datateknologi og informatikk på dataingeniør studiet på NTNU. Produktet er laget for Rambøll AS.

Oppgaven ble valgt da teamet hadde interresse for systemutvikling og syntes en oppgave som var direkte knyttet til arbeidslivet. Jevnlig møter med oppdragsgiver har vært en essensiell del av utviklingsprossesen og teamet derfor vil takke oppdragsgivere Alina Bondarenko og Marcus Lagerkvist fra Rambøll AS for et godt samarbeid og oppfølging gjennom prosjektet. Teamet vil også takke veileder Muhammad Ali Norozi for tips og råd underveis, og Brandon Segermeister fra IT-avdelingen til Rambøll for tekniske hjelp angående produkssjonssetting i Azure.

Trondheim 22. mai 2023

Andreas Henriksen Dahle Hadar Hayat Mathangi Pushparajah

Oppgavetekst

Norges brannrådgivere jobber med et regelverk som er utydelig, vanskelig å forstå og skaper usikkerheter. Dagens bygningsprosess er iterativ og lite digital, hvilket skaper usikker fremdrift og er fordyrende. Alt dette bidrar til at byggverk blir dyre og lite bærekraftige. Rambøll ønsker å se på muligheten for å løse noen av disse utfordringene ved å bygge en form for "benchmarking" verktøy for branntekniske løsninger.

Produktet vil være en webbasert applikasjon hvor ingeniører kan angi løsninger for forskjellige byggverk. Et for form for skjema. Alle løsningene skal lagres i en database og eksporteres i et format som egner seg for en rapport (pdf, word, notion, coda, osv.).

Dette vil kunne hjelpe bygge-, anleggs- og eiendomsnæringen med å nå sine klimaambisjoner. Løsningen er validert med kunder og vil gi muligheten til å jobbe i et svært innovativt selskap med konkrete og spennende problemstillinger.

Innhold

Figurer

Tabeller

Definisjoner, akronymer og symboler

Brannklasse – Prosjektering og gjennomføring legger brannklasser som grunnlag for å sikre byggverkets bæreevne, osv. ved brann. Det er fire brannklasser, der 1 = Liten konsekvens, 2 = middels konsekvens, 3 = stor konsekvens og 4 = særlig stor konsekvens. Konsekvensen bestemmes avhengig av byggverkets bruk, antall personer, størrelse, brannenergi osv. [1].

Bruker - En person som samhandler med applikasjonen.

Brukergrensesnitt - Grensesnittet som lar brukeren samhandle og kommunisere med programmet.

CI - Forkortelse for Continuous Integration. Dette er en programvarepraksis der endringene til koden blir overført (engelsk: committed) i korte intervaller til en felles

repository. Ved å overføre koden oftere, kan en lett finne kilden til feilmeldinger, og det blir også lettere å slå sammen endringene i koden som er gjort av andre i teamet [2].

CLI – Forkortelse for Command-line Interface. Et kommandolinjeprogram som godtar tekstinput for å utføre operativsystemfunksjoner og brukes fortsatt av programvareutviklere og systemadministratorer til tross for utbredelsen av grafiske brukergrensesnitt [3].

Fravik – I brannteknisk sammenheng betyr fravik: "Fravik betyr at det er valgt en alternativ ytelse som krever verifikasjon for oppfyllelse av forskriftskrav." [4]. Altså er det unntakelser fra forskriften for tekniske krav til byggverk, som må dokumenteres med alternative løsninger.

Gantt diagram - Et diagram som illustrerer prosjektplan og tidsplan.

Git - Programvare for versjonskontroll.

GitHub - En nettbasert plattform/verktøy, for å hjelpe programvareutviklere med å holde styr på Git-endringer og generelt bedre organisere et prosjekt.

ISO - Den internasjonale standardiseringsorganisasjonen, er organisasjonen ansvarlige for en rekke internasjonale standarder som inkluderer nettstandarder [5].

MVC - Står for Model-View-Controller, dvs. Modell-Visning-Kontroller. Dette er en programvardesign som følges for å strukturere kildekoden.

MVP - Står for Minimum Viable Product, dvs. Enklest brukbart produkt. Dette er den første ferdige iterasjonen av produktet, med kun de mest elementære funksjonalitetene.

ORM - Object-Relational Mapping, er et verktøy som lar utviklere samhandle med en database ved hjelp av objektorientert programmering. I stedet for å skrive SQL-spørringer for å samhandle med databasen, kan utviklere bruke ORM til å opprette, lese, oppdatere og slette data ved å bruke objekter og metoder i deres programmeringsspråk. Dette kan gjøre det lettere å jobbe med databaser og kan bidra til å redusere mengden standardkode som utviklere trenger å skrive.[6]

RDBMS – Forkortelse for Relational Database Management System. Det er en type database system som lagrer data i form av relaterte tabeller. Relasjonsmodellen ble dannet som en måte å organisere data ved å bruke et sett med tabeller med kolonner og rader, der hver rad representerer en oppføring og hver kolonne representerer et attributt. Tabellene kan relateres til hverandre ved hjelp av vanlige attributter, noe som muliggjør effektiv datainnhenting og administrasjon.[7]

Risikoklasse – For å sikre rømning og redning ved brann, skal risikoklassene legges som grunnlag for prosjektering og gjennomføringen. Det finnes 6 risikoklasser der valget av risikoklassen bestemmes ut fra byggverkets bruksområde og forutsetninger at menneskene i byggverket vet hvordan å bringe seg selv til sikkerhet ved brann [8].

Virksomhet – Beskriver type virksomhet, f.eks. Bolig, Industri, Skole, Arbeidsbrakke osv.

VTEK17 – Dette er en veiledning om tekniske krav til byggverk, og beskriver minimums egenskapene et byggverk må ha for at den kan omfattes som lovlig i Norge. Veiledningen forklarer kravene som forventes, og tilbyr preakseptert ytelser som kan dekke kravene [9].

WCAG 2.1 - Retningslinjer for å gjøre programmer og nettsteder mer tilgjengelige.

1 Introduksjon og relevans

Når en stor bedrift skal effektivisere og øke bærekraften av deres prosesser, vil digitale løsninger være til hjelp. Oppdragsgiver Rambøll AS, et arkitektur-, ingeniør- og konsulentfirma har en brannsikkerhetsavdeling som ønsket en effektivisering av brannsikkerhets rapportskrivingen deres. Dette er i form av en webapplikasjon som lar de fylle ut informasjon om nye prosjekter, eksportere og sammenlikne de med liknende prosjekter.

1.1 Problemstilling

Da teamet leste og diskuterte oppgaveteksten, var spørsmålet hvordan man best skal utvikle en slikk applikasjon. Det teamet kom fram til var at brukervennlighet var sentralt, hvorav brukertester er viktig for å forsikre at den er brukervennlig. Teamet samlet også all overordnet funksjonalitet som skulle implementeres, og kom dermed fram til denne problemstillingen.

«Hvordan lage en brukervennlig webapplikasjon for en mer effektiv og dermed bærekraftig måte for brannrådgivere og ingeniører å registrere, lagre, eksportere og sammenlikne prosjekter på?».

1.2 Rapportens struktur

Rapportens struktur er som følger, den starter med Introduksjon og relevans som tar for seg oppdragsgivers behov, teamets mål og problemstilling. Videre tar rapporten for seg teori og relevant litteratur, dette kapitlet går gjennom alle teoritemaer som er relevante for leseren. Deretter kommer metodekapitlet som går gjennom hvordan produktet ble utviklet, samt valg av teknologi. Resultater er det neste kapitlet som tar for seg alle resultater fra både produkt og metode. Så kommer Diskusjonskapitlet som drøfter resultatene, nest sist er konklusjonen som oppsummerer resultater og svarer på problemstillingen. Til slutt er et kapittel om Samfunnspåvirkning, som handler om miljø og bærekrafts påvirkningene til produktet.

2 Teori og relevant litteratur

Bacheloroppgaven som teamet jobbet med, faller under temaet utvikling, da oppgaven er å lage en webapplikasjon for en bedrift. Utviklingen av ulike applikasjoner er noe som skjer i stor skala, og dekke ulike behov etter ønske fra klienten.

2.1.1 Universell utforming

Norge har laget et regelverk for tilgjengelighet kalt universell utforming som er utviklet for å skape best mulig tilgjengelighet til applikasjonen uavhengig av forutsetninger [10]. Regelverket gjelder webapplikasjoner, apper og automater. Disse reglene må følges dersom man skal lage en offentlig tilgjengelig webapplikasjon i Norge, reglene for utvikling er basert på WCAG 2.0 og setter krav til tilgjengeligheten avhengig om nettsiden er laget for offentlig- eller privatsektor [11].

2.1.2 Menneske maskin interaksjon

Et felt dedikert til å studere og forbedre samspillet mellom en bruker og datateknologi, angående både maskinvare og programvare. Feltet dekker emnene brukergrensesnitt (UI), brukeropplevelse (UX) og Kognisjonsvitenskap. Menneske maskin interaksjon minner svært om UX design, men i er gjerne mer akademisk i sin natur. Dette betyr gjerne en mer vitenskapelig vinkling og metode i arbeidet [12]. Dette kan en for eksempel se gjennom brukertester.

UX design skiller seg fra UI design i det at den tar hensyn til hele brukeropplevelsen i motsetning til bare utseende. Hensikten med UX design er å gjøre produktet så lett, raskt og behagelig som mulig å bruke og ikke bare se bra ut. I prosessen for å komme fram til et godt design er brukertester viktig, da det gir innsikt i brukeropplevelsen [13].

2.1.3 Wireframe

Wireframe er en illustrasjon av applikasjonen, den er gjerne uten farger og enkelt satt opp. Hensikten med en wireframe er å fastsette navigasjon og posisjon av elementene på en gitt side. Disse er et godt utgangspunkt for videre prototyping og design, da de gir et overblikk over sideoppsettet. Siden endringer på wireframe tar mye mindre tid enn en prototype eller applikasjon er det vanlig å kjøre brukertester på wireframe. Hensikten med det er å sjekke om navigasjonen og layout av applikasjonen er brukervennlig, og dermed kunne rette opp feil og mangler før utviklingen starter [14].

2.1.4 Prototype

En prototype i sammenheng av design og wireframes betyr en mockup av den faktiske applikasjonen med farger og alle elementer på plass. Her fastsettes spesifiseres gjerne elementer gjennom farger, font og skygger [15]. Det kan også være en versjon av applikasjonen uten all funksjonalitet, men som har alt av utseende og navigasjon på plass slik at den føles som det ferdige produktet.

2.1.5 Brukertester

Testing av brukergrensesnittet lar utviklere få nye perspektiv på menneske-maskininteraksjonen og brukervennligheten til applikasjonen. Den internasjonale standardiseringsorganisasjonen (ISO) definerer brukertesters hensikt som evnen et produkt har til å oppnå spesifikke mål med effektivitet og tilfredshet [16]. Dette betyr at oppgavene som blir gitt må være presise og dekke all kjernefunksjonalitet i applikasjonen.

En annen viktig faktor er antallet personer det blir testet på, her er det stor uenighet om hvor mange er nødvendige, hvor det ideelle antallet faller et sted mellom fem på det minste og femten på det meste [17]–[19]. Det viktigste med brukertestene er å finne feil og mangler ved applikasjonen, samt få tilbakemeldinger på brukergrensesnittet. Brukertester gjøres vanligvis på wireframe og MVP, da dette ikke er ferdige versjoner av applikasjonen, hvor brukertester kan avdekke mangler som kan rettes opp før lansering [20].

Det er ikke bare antallet personer som må tas hensyn til, det er også hvilke personer man tester. At brukerne som testes representerer målgruppen til produktet er viktig, da det er lite hensiktsmessig å teste personer som ikke skal eller kan bruke produktet. En annen faktor man må vurdere er om man skal teste samme personene flere ganger, da de allerede har kjennskap til applikasjonen. Det er generelt ikke anbefalt å teste på samme personer mer enn en gang, da brukertester skal måle hvor lett det er å bruke programmet uten tidligere erfaring. Selv om testene gjøres på samme personer, kan de bidra med god innsikt, men det vil ikke være representativt for en førstegangsbruker [21].

Det er flere måter å gjennomføre brukertester, da en kan velge om en skal stille spørsmål underveis, få brukeren til å tenke høyt eller stille spørsmål helt til slutt. En brukertest bør helst gjennomføres fysisk da det gir mulighet til å observere om en bruker sliter med å gjøre en oppgave eller ikke. Hvis en får brukeren til å tenke høyt underveis så kan en ulempe være at en ikke vet presisjonen, eller tiden de brukte på å gjennomføre oppgaven. Dette er ikke et problem ved testing av IT-applikasjoner, da en ikke trenger å vite eksakt hvor lang tid en testperson brukte på en gitt oppgave. Hvis en kun stiller spørsmål på slutten vil en ha ulempen av at brukeren ikke husker alt de synes om applikasjonen, og dermed miste viktig data. Det er derfor en vanligvis bruker en kombinasjon av de ulike metodene tilpasset brukertesten [22].

2.1.6 Rammeverk

Rammeverk er kodebaser bygget opp slik at en kan gjenbruke komponenter og ikke må bygge alt fra bunnen av. De lager gjerne struktur for oppbygning av prosjekter og er gjerne brukt i sammenheng med klientprogrammer [23].

2.1.7 Enhetstester

Testing av kode er en viktig del av utviklingsprosessen, da de gjør det lettere å finne feil og svakheter i kodebasen. Det finnes flere måter å teste koden på, som ende til ende testing og enhetstesting. Den mest aktuelle formen for testing i dette prosjektet er enhetstesting, da den tester hver enkelt metode. Noe som gjør applikasjonen mer skalerbar da feilsøking blir lettere, siden en stor andel feil blir fanget av testene før lansering. Det er flere fordeler med enhetstesting som kommer av dypere forståelse av koden. Et eksempel er om en finner metoder vanskelige å teste kommer dette gjerne av metoden har dårlig kohesjon. Generelt

betaler tidsbruken på enhetstesting seg tilbake gjennom at man bruker mindre tid på feilsøking [24].

2.2 Programvarearkitektur

2.2.1 Klient-tjener

Klient tjener er en programvarearkitektur hvor mange klienter sender og mottar forespørsler til og fra en felles tjener. Denne arkitekturen ligger til grunn for nesten alle nettsider, da det er en effektiv måte for mange enheter å ha tilgang til samme data. Modellen er mest effektiv når klientene og tjeneren har forskjellige oppgaver de må gjøre f.eks. at klienten kjører en applikasjon for å fylle inn et skjema, mens tjener behandler data innsendte skjemaer [25].

2.2.2 Rest API

Rest API er et av de mest brukte design prinsippene for kobling mellom klient og tjener. Rest følger en rekke prinsipper, som at alle forespørsler for samme ressurs er like uavhengig av hvor den kommer fra, at klient og tjener er uavhengige og mer. Rest fungerer ved å sende HTTP forespørsler mellom klient og tjener, i form av standard database-forespørsler. Dette inkludere GET, POST, PUT og DELETE, disse forespørslene kalles CRUD [26].

2.2.3 Relasjonsdatabase

Relasjonsdatabase er et system basert på tabeller (relasjoner), hvor rader og kolonner må være unike, for rader gjøres dette gjerne gjennom en primærnøkkel. En primærnøkkel er en unik variabel som brukes for å indentifisere raden. For å koble sammen tabeller brukes fremmednøkler, dette er primærnøkler fra en annen tabell. For å aksessere data i en relasjonsdatabase må man kjøre spørringer i SQL, som er basert på relasjonsalgebra [27].

2.2.4 MVC

Model-View-Controller er et mønster innenfor programvaredesign som brukes for å sette opp brukergrensesnitt, data og logikk. Den skiller mellom applikasjonens forretningslogikk og visning. "Model" skal vise datastrukturen til applikasjonen. "View" skal sette opp applikasjonens design, og viser all relevant informasjon på siden for en god brukeropplevelse. "Controller" har ansvar for interaksjon mellom "Model" og "View", f.eks. manipulere data og oppdatere det i visningen [28].

2.2.5 Kobling

Kobling viser forholdet mellom to eller flere komponenter. Høy kobling er når komponenter er sterkt avhengig av hverandre og en slik kobling er vanskelig å endre og vedlikeholde siden det er lett for feil å oppstå. En typisk feil som kan oppstå er når en endrer på interne variabler av en komponent gjennom flere komponenter, noe som kan føre til at en mister kontroll over hvilke komponenter som har gjort endringen og gjør det vanskelig å feilsøke hvis variablene har feil verdier. Det er ønskelig med lav kobling mellom komponenter som er lite avhengig av hverandre slik at det er lettere å teste dem isolert og er lett å endre og vedlikeholde [29]. Dette gjøre det også lettere for eventuelle utvidelser på en applikasjon som bygger på komponenter som allerede eksisterer. Et eksempel på dette er et "Konto" objekt som inneholder en liste med "Prosjekt" objekter, men et "Prosjekt" objekt trenger ikke å lagre

hvilket "Konto" objekt den tilhører til. Den kan i stedet lagre hvilken konto id den tilhører til, og dette fører til lavere kobling mellom komponenter.

2.2.6 Kohesjon

Kohesjon forteller hvilke oppgaver en komponent gjennomfører internt. Høy kohesjon viser at komponenter gjennomfører oppgaver den var ment å gjøre, mens lav kohesjon gjennomfører oppgaver som er utenfor sitt område. Et program må ha høy kohesjon slik at den er oversiktlig og det er lett å forstå hva ulike komponenter har ansvar over [29]. Det vil si, for eksempel, en "Person" komponent inneholder detaljer til en person som navn og e-post. Funksjonen for å sende en e-post tilhører ikke til komponenten "Person" siden den skal bare inneholder detaljene til selve personen. Funksjonen for å sende e-post må derfor være i en annen komponent som "E-post" for å sikre at komponentene har høy kohesjon.

2.2.7 Aggregering

Et enveis forhold mellom komponenter. For eksempel, en skole kan ha mange studenter, men en student kan ikke tilhøre til mange skoler. Et slikt forhold gjøre det mulig å fjerne en komponent uten å påvirke den andre den har et forhold til. Et "student" objekt kan fjernes uten at den påvirker et "skole" objekt [30].

2.2.8 Komposisjon

Det er en strengere form av aggregering. I komposisjon, er komponenter avhengig av hverandre, slik at den ene komponenten kan ikke eksistere uten den andre. Et eksempel på dette er at et "Bok" objekt kan ikke eksistere uten et "Forfatter" objekt, så hvis et "Forfatter" objekt slettes, så blir alle "Bok" objekter knyttet til det "Forfatter" objektet slettet [30].

2.2.9 Virtuelle nettverk

Kommunikasjonen i en virtuelle nettverk kan skje mellom vanlige PC-er, virtuelle maskiner, virtuelle servere og andre enheter. I motsetning til et fysisk nettverk som bruker kabler og maskinvare for å kommunisere mellom enheter, gjør virtuelle nettverket dette gjennom programvare ved å bruke virtuelle versjoner av tradisjonelle nettverksverktøy som svitsjer og nettverkskort. Noen fordeler med virtuelle nettverk er at det reduserer kostnader og er lettere å håndtere. Det er fleksibelt ved at det tilbyr ulike alternativer for nettverksstruktur og konfigurasjon, og det har bedre kontroll over nettverkstrafikken [31]. Virtuelle nettverk sikrer kommunikasjon mellom enheter slik at bare de som har tilgang til nettverket får aksess til nettverket og det blir mulig å kommunisere trygt uten bekymring for angrep.

2.3 Smidige Utviklingsmetoder

Smidig utvikling er en metode som gir klienten mer innsikt i utviklingen av produktet, samt lar utviklere gjøre endringer etter kundes ønsker. Kjernen i smidig utvikling er en prosess hvor det gjøres endringer i korte sykluser, med fokus på samarbeid og kommunikasjon. Det finnes mange måter en kan adaptere grunnlaget smidig utvikling setter, som SCRUM eller Kanban [32].

2.3.1 Scrum

Scrum er en ofte brukt utviklingsmetode som fokuserer på å gjøre utviklingene i korte intensive perioder, kalt sprinter. Før sprintene, settes alle mål opp i et produkt-backlog, hvorav noen av disse blir valgt ut til sprint backloggen. I løpet av sprinten holdes det korte daglige møter, standups, hvor alle forteller hva de har gjort og hva målet for dagen er, målet med disse er å holde møtene til det minimale og fremdeles få nok informasjon til å programmere. Etter hver sprint holdes det en review, hvor teamet vurderer hva som gikk bra og hva som kan forbedres. Hensikten med dette er å la teamet justere målsetting og arbeidsmetode til neste sprint [33].

2.3.2 Kanban

En måte å drive smidig utvikling på er ved bruk av en Kanbantavle, hvor man har som mål å visualisere arbeidsmengden, flyten av oppgaver, samt minimalisere oppgaver som jobbes på. Kanban tavler er ikke eksklusive til utvikling, men brukes i flere former for produksjon. Det kommer opprinnelig fra Toyota fabrikkene hvor det ble brukt i produksjonen av biler. [34]

Som utviklingsmetode minner Kanban tavler litt om Scrum sine sprinttavler, den største forskjellen mellom metodene er bruken av sprinter og roller. Scrum har et rigid og fast system for sprinter hvor utviklingsfasene deles opp, og har faste roller for hver sprint. Kanban har derimot en mer fleksibel løsning hvor målet er å bli ferdig med enkelt oppgaver så fort som mulig før man begynner på en ny oppgave, dette forhindrer forvirring og rot i utviklingsprosessen [35].

3 Metode

3.1 Forskningsmetode

Å hente inn data på en systematisk og nøytral måte, og videreføre dette til kunnskap og teorier kalles forskningsmetode. Begrepet betegner både empiriske undersøkelser og drøfting av teorier [36]. I lys av en systemutviklingsoppgave er de empiriske undersøkelsene relevante for brukertester og drøfting av teoretisk materiale relevant for kunnskapen oppgaven baseres på.

Det er flere måter å gjennomføre empiriske undersøkelser, hvorav hoved metodene er kvalitative og kvantitative undersøkelser. For brukertestene gjennomført i denne oppgaven er det mest relevant å ta for seg den kvalitative metoden, da brukertester faller innenfor denne kategorien [37].

3.2 Brukertester

Brukertester er en essensiell del av prosessen og en viktig del av forskningsmetoden, da de gir innsikt i feil, og mangler i design og funksjonalitet. Siden applikasjonen kun skal brukes av branningeniører og -rådgivere var det naturlig å gjennomføre brukertester på dem, over å gjennomføre testene på tilfeldige personer. Det var seks branningeniører og -rådgivere tilgjengelig for brukertesten, dette ifølge flere rapporter bør være nok for å avdekke feil og mangler ved programmet.

Brukertestingen på wireframe og MVP ble utført på en liknende måte, da oppgavene som brukerne skal gjennomføre er like. Og brukerne ble bedt om å «tenke høyt» når de gikk gjennom applikasjonen, dette lar teamet notere ned meninger fortløpende. Denne metoden ble foretrukket over å kun stille spørsmål til slutt, da mange meninger om applikasjonen blir glemt underveis. I begge brukertestene ble det også gjennomført en anonym meningsmåling, for å få den ærlige meningen til brukerne om applikasjonen. Den største forskjellen er at feil oppdaget i første brukertesten er rettet opp i brukertesten på MVP. I tillegg er MVP en mer realistisk brukeropplevelse, da den har tilnærmet all funksjonalitet som den ferdige applikasjonen har. En annen viktig endring er at fokuset går fra å få meninger rundt design og navigasjon, til en mer helhetlig brukeropplevelse av applikasjonen.

3.3 Valg av teknologi

Da valget av teknologi skulle gjøres for denne oppgaven var det flere faktorer som måtte tas hensyn til, da Rambøll AS ikke er et IT-firma er det relevant å tilpasse valg av teknologi deretter. Det er også en viktig faktor at teknologien skal ha god støtte og være rask, samt at applikasjonen skal behøve minimalt med vedlikehold. Et element som ikke var like essensielt var kapasitet til å håndtere mange brukere samtidig, da antallet brukere er lavere enn trafikken på en tradisjonell nettside. Grunnen til dette er at applikasjonen kun skal brukes av Rambølls branningeniører i Norge.

3.4 Valg av klientteknologi

Det var ikke satt noen faste krav for klientteknologi, men oppdragsgiver har kjennskap til React fra før. Dette er ikke alene nok grunnlag til å velge teknologi, men gjør at teamet vekter React over andre alternativer dersom det ikke er noen store fordeler med en annen teknologi. Et godt alternativ var Vue som teamet allerede har erfaring med. Andre alternativer var også vurdert, men ble valgt bort grunnet mangel på erfaring samt ikke store nok fordeler over Vue eller React.

Den største forskjellen mellom Vue og React er at Vue er et rammeverk og React er et bibliotek. Dette betyr i praksis at for å ha samme funksjonalitet som Vue må man installere flere pakker med React. Ellers er en annen forskjell at mange pakkene i økosystemet til Vue er utviklet av Vue i motsetning til React hvor de er utviklet av felleskapet. Begge teknologiene bruker en virtuell Dom og komponent basert utvikling, og de er begge raske [38]. Basert på at det ikke er noen store fordeler med å velge annen teknologi enn React, valgte teamet å bruke React da dette er en teknologi oppdragsgiver har kjennskap med.

3.4.1 NPM

NPM er et register som inneholder nesten alle JavaScript-pakker og ble brukt for installasjon av disse. Det var også brukt som CLI i prosjektet da den er enkel å bruke og har ikke mange negative sider over Yarn annet en hastighet på installasjoner som ikke har mye å si for et prosjekt av denne størrelsen.

3.5 Valg av tjenerteknologi

For valget av tjenerspråk var de mest relevante alternativene Java med Springboot, og NodeJS. Da begge er gode alternativer er det flere elementer som må tas hensyn til. Som nevnt tidligere er ikke Rambøll et IT-firma derfor vektes teknologi de har kjennskap til høyere. Dette er ikke alene nok for å fastsette et valg. Andre viktige faktorer å ta hensyn til er læringskurve, stabilitet og støtte.

Når det kommer til valget mellom Springboot eller Nodejs må en se på fordelene og ulempene mer hver teknologi. Hvor NodeJs sine største styrker inkluderer at den er lettvektig, har god I/O håndtering og at den er asynkron som betyr at den vil «sove» når den ikke gjør noe. Den har også støtte fra et rikt bibliotek av pakker med NPM [39]. Fordelene med Springboot er at den håndterer flertrådet programmering bedre enn Nodejs og har «type safety», den har i likhet med Nodejs tilgang til et rikt bibliotek med avhengigheter[40]. En applikasjon av denne typen har ikke behov for en mer rigid «type» struktur og flertrådet programmering, men hadde heller behov for en lettvektig løsning. Derfor valgte teamet Nodejs og Springboot.

3.6 Valg av databaseteknologi

Siden databasen skulle være i Rambøll sine tjenersystemer må database språket være kompatibelt med Microsoft Azure, da dette er skysystemet Rambøll bruker. Dette betyr fortsatt at det er mange mulige valg, hvorav MySQL, SQL og PostgreSQL var alternativene som ble vurdert.

3.6.1 MySQL

MySQL er en open-source RDBMS basert på Structured Query Language (SQL). Den er kjent for sin pålitelighet, ytelse og brukervennlighet, noe som gjør den til et populært valg for webapplikasjoner og for bruk med Express-rammeverket og Sequelize ORM [41].

En av de viktigste fordelene med MySQL er brukervennlighet. Den har en enkel og intuitiv syntaks og god dokumentasjon, noe som gjør den enkel å lære og bruke. I tillegg har MySQL et stort og aktivt fellesskap som gir ressurser og støtte til utviklere. MySQL er også kjent for ytelsen. Den er designet for å være rask og effektiv, noe som gjør den godt egnet for bruk i nettapplikasjoner med høy trafikk.

Når det gjelder sikkerhet, tilbyr MySQL flere funksjoner for å sikre sikkerheten til dataene lagret i databasen. Det inkluderer støtte for SSL/TLS-kryptering, "access control mechanisms" og datakryptering. Det er selvfølgelig fortsatt viktig å følge beste praksis når en konfigurerer MySQL for å sikre at dataene dine er sikre [41].

Å bruke MySQL med Sequelize ORM og Express-rammeverket kan ofte være en god idé fordi det kan bidra til en jevn programvare funksjonalitet. Sequelize er en ORM som fungerer utmerket med MySQL. Det gjør også MySQL med Node.js levedyktig. MySQL er en pålitelig og brukervennlig RDBMS som passer godt for bruk med både Express-rammeverket og Sequelize ORM. Alt dette har hatt påvirkning på valget om hvilke RDBMS teamet brukte. I tillegg var det viktig for å bruke teknologier som er godt dokumentert, ikke bare hver for seg, men også sammenhengende. Teamet fant ut at det er enkelt å finne dokumentasjon og eksempler på metoder og funksjoner som bruker MySQL, Sequelize og Express sammenhengende. Dette var veldig viktig ikke bare for teamet, men også med tanke på videre utvikling, da det vil gjøre det enklere for de som eventuelt tar over prosjektet.

3.7 Tredjeparts bibliotek

3.7.1 DOCX Js

DOCX Js er en pakke brukt for å skrive til Word-filer, da det var behov for å lage en mal for rapporter som lages av programmet. Den er bygget opp på word.xml og gjør programmering av Word-filer lettere da den har nesten all funksjonalitet tilgjengelig. Pakken mangler funksjonalitet for automatisk oppdatering av kildelister. Referanser må derfor gjøres manuelt gjennom lenker [42].

3.7.2 Chart Js

Chart Js gir muligheten for å visualisere data i form av grafer, den tar in lister i form av et JSON objekt med tall, og navn på kolonner, og returner ett gitt diagram [43]. I applikasjonen genererer den stolpe diagrammer basert på risikoklasse og virksomhet i forhold til fravik.

3.7.3 Sequelize

Sequelize er en promise-basert ORM for moderne TypeScript og JavaScript. Den støtter ulike databaser, inkludert PostgreSQL, MySQL, MariaDB og flere. Den gir støtte for «unmanaged transactions» og «managed transactions», relasjoner og mer.[44]

Sequelize lar brukeren definere modellene veldig enkelt og gjøre valgfri bruk av automatisk databasesynkronisering. Brukeren kan også definere assosiasjoner mellom modellene og derfor la Sequelize håndtere relasjoner mellom databasetabeller. En annen funksjon i Sequelize er «soft deleting», denne lar en bruker merke data som slettet i stedet for å fjerne dem en gang for alle fra databasen [45].

Å bruke en ORM som Sequelize kan ha flere fordeler fremfor å lage tabellene i databasen separat. Noen av disse fordelene inkluderer det enkle oppsettet og støtten for flere databaseteknologier som MySQL, MariaDB og PostgreSQL. Sequelize støtter også migreringer og har et CLI-verktøy for å danne og «seede» data. Det gir også en solid transaksjonsstøtte med både «unmanaged transactions» og «managed transactions» [46].

Bruk av Sequelize kan imidlertid ha sine ulemper. For eksempel kan det være tregere enn å bruke innebygd SQL fordi det må oversette de objektorienterte spørringene til SQL og sende dem til databasen. I tillegg kan Sequelize i noen tilfeller generere kompliserte spørringer, og den mangler NoSQL-støtte.

Når det gjelder sikkerhet, kan ORM som Sequelize bidra til å forhindre SQL-Injection ved å fjerne behovet for å skrive "raw" SQL-spørringer og gi et høyere abstraksjonsnivå når en arbeider med databasen. Det er likevel viktig å følge beste praksis for sikker koding og validere brukerinput på riktig måte [47].

3.7.4 Express

Express er et populært webapplikasjons-rammeverk for Node.js. Den er designet for å være minimal og fleksibel, og gir et robust sett med funksjoner for å bygge både web- og mobil-applikasjoner.

Grunnlaget til Express er evnen til å forenkle prosessen med å bygge tjener-baserte applikasjoner. Det gir et enkelt og brukervennlig API for å «route» forespørsler, håndtere «middelvare» og gjengi dynamisk innhold [48]. Dette gjør det enkelt å organisere applikasjonens funksjonalitet og legge til nyttige nye funksjoner.

Express gir også en rekke avanserte funksjoner som robust «routing», støtte for flere ferdiglagde mal-verktøy og muligheten til å forhandle om innholdstype [49]. Disse funksjonene gjør det enkelt å bygge komplekse webapplikasjoner med Express.

Når det gjelder sikkerhet, er det flere fremgangsmåter en kan følge for å sikre at Express-applikasjonens sikkerhet. Disse inkluderer bruk av de mest oppdaterte og ikke-sårbare versjoner av Express, bruk av TLS for å sikre tilkoblingen og dataene, bruk av «Helmet» for å sette sikkerhetsrelaterte HTTP-headers, sikring av «cookies» og forhindrer «brute-force»-angrep mot autorisasjon [50].

3.7.5 Axios

Axios er en populær «promise»-basert HTTP-klient for node.js. Den er designet for å være isomorf, noe som betyr at den samme kodebasen kan brukes både på tjenersiden og klientsiden. Dette gjør det enkelt å bruke Axios i en rekke miljøer og plattformer [51].

En av nøkkelfunksjonene til Axios er evnen til å lage HTTP-forespørsler som GET, POST, PUT, DELETE og mer. Den gir også en rekke avanserte funksjoner som å transformere forespørsler og responsdata og kansellere forespørsler. Disse funksjonene gjør det enkelt å tilpasse og kontrollere oppførselen til HTTP-forespørsler [52].

Axios gir brukere også muligheten til å legge inn brukerautentisering i en webapplikasjon ved å legge til autorisasjonstoken i forespørselens header. Det er ulike måter å gjøre dette på, en måte kan for eksempel være ved å bruke Axios Interceptors for å fange opp alle forespørsler og legge til autorisasjon i headeren automatisk [53]. Axios håndterer også den underliggende sikkerheten til tilkoblingen ved å bruke den opprinnelige node.js http-modulen på tjenersiden og XMLHttpRequests på klientsiden. Disse modulene håndterer SSL/TLS-kryptering for å sikre at data overføres sikkert [51].

3.8 Valg av skyteknologi

Da Rambøll er integrert i Microsoft sine systemer og bruker Azure for tjenere, databaser og innlogging systemer, dette betyr at applikasjonene er lettere å integrere inn i deres systemer enn å opprette et separat system for programmet.

3.8.1 App Service

App Service er en http-basert tjeneste som brukes for å bygge og produksjonssette applikasjoner og APIs. Den støtter ulike programmeringsspråk som Java, Ruby, Python og mer, inkludert Node.js som ble brukt for applikasjonen. Den også har funksjonalitet for å ta i bruk tjenester f.eks. CI (Continuous Integration) fra Azure Devops, GitHub og Docker Hub [54]. App Service var brukt for å produksjonssette klient og tjener, der begge var rullet ut sammen på en App Service.

Det ble lagt inn autentisering på App Servicen der klient og tjener var produkssjonssatt. Tilgangen til en «Identity Provider» var gitt fra Rambøll som kunne kobles til App Servicen for å integrere applikasjonen med Rambølls innloggingssystem.

3.8.2 Azure Database for MySQL

Azure Database for MySQL er en relasjonsdatabasetjeneste. Det finnes to produksjonssettings modeller for databasen, der den ene er kalt for «Flexible Server», og den andre er «Single server». «Flexible Server» er brukt for enkel produksjonssetting av databaser, og er lett å skalere og den bruker mindre ressurser for ulike oppgaver som sikkerhetskopiering, høy tilgjengelighet, sikkerhet og overvåkning. «Single Tjener» er strukturert på en måte slik at den ikke støtter store tilpasninger på databasen over tid i motsetning til «Flexible Server», og i tillegg til dette er «Single Server» ikke lenger mye brukt siden ressursen stoppes over tid, og det er anbefalt å bruke og flytte over til «Flexible Server» [55]. Applikasjonen bruker «Flexible Server» for databasen.

3.8.3 Azure AD

Azure AD står for Azure Active Directory. Dette er en sky-basert administrasjonstjeneste som håndterer identiteter og tilgang til ulike ressurs. En har tilgang til eksterne ressurser som Microsoft 365 og Azure Portal ved å gå gjennom Azure AD. Den også fungerer for interne ressurser f.eks. håndtere tilgang til ulike applikasjoner som en bedrifts bruker [56].

3.8.4 Azure Managed Identity

Azure Managed Identities gjør det mulig for utviklere å håndtere hemmeligheter, legitimasjon, sertifikater, og nøkler på en lettere måte for å sikre og autentisere kommunikasjonen mellom applikasjoner og tjenester. Applikasjonen er produksjonssatt i en Azure App Service og måten kommunikasjonen mellom applikasjonen og database er sikret på er at App Service bruker Azure Managed Identity til å hente Azure AD tokens som brukes for å aksessere databasen. På den måten slipper teamet å håndtere sikkerheten manuelt da alt blir håndtert gjennom Azure, og utviklere trenger bare å teamet token som hentes via Azure Managed Identity [57].

3.8.5 GitHub

For versjonskontroll ble GitHub brukt, da det har rik funksjonalitet. Det ble laget to repositories en for klient og en for tjener. I tillegg ble GitHub prosjekter brukt som issue board. Dette gav en oversiktlig måte å vite hvem som jobbet med hva. Github ble også brukt for CI/CD da Github «Actions» håndterte bygging og kjøring før det ble videre produksjonssatt i Azure.

3.9 Valg av utviklingsmetode

Når det kom til valg av utviklingsmetode så er det flere mulige metoder som var aktuelle, dette inkluderer Kanban og SCRUM. En ikke-smidig utviklingsmetode virket uaktuelt da oppdragsgiver bør kunne gi tilbakemeldinger i utviklingsprosessen. Smidig utvikling lar teamet gjøre endringer etter oppdragsgivers ønsker underveis og gjør retting av feil lettere. Teamet valgte en adaptasjon av Kanban som utviklingsmetode over Scrum, da den rigide strukturen til Scrum sprinter ikke passet med teamets timeplan. Det ble satt opp to Kanbantavler i Github, en for klient og en for tjener. Her ble alle overordnede oppgaver lagt inn, disse ble delt inn i mindre underoppgaver ved behov. De overordne oppgavene lot teamet jobbe med oppgaver uten store «merge» konflikter.

For å holde oversikt og koble deler godt sammen ble det hold møter innad i teamet flere ganger i uken, hvor progresjon ble delt og sammenkobling gjort. Det var også sentralt å opprettholde kommunikasjon mellom møter ved spørsmål ol. For kommunikasjon ble Discord brukt da det gir mulighet å opprette flere kanaler for å holde diskusjon oversiktlig. For alle filer som ikke var programmeringsrelaterte ble Google Drive brukt, dette gir mulighet til samarbeid på filer i sanntid og gir lett oversikt over alle felles filer. Det lagres også tidligere versjoner av filer som er nyttig dersom noe slettes eller endres ved uhell.

For å holde utviklingen smidig var hyppige møter med oppdragsgiver essensielt. Dette gjør at de har oversikt over progresjon og mulighet til å gi tilbakemeldinger underveis. Ved design av wireframe var deres tilbakemelding essensiell da de tekstlige beskrivelsene av behovene til applikasjonen kan tolkes forskjellig. Dette gjør det lettere å endre design før utviklingen startet.

3.10 Arbeids og rollefordeling

Rollefordeling ble gjort tidlig i prosjektet under skriving av arbeidskontrakt. Arbeidsfordelingen ble gjort smidig underveis i prosjektet, ved hjelp av hyppige møter og Kanbantavler. De fastsatte rollene kan sees i arbeidskontrakten i vedlegg 2.

4 Resultater

4.1 Vitenskapelige og ingeniørfaglige resultater

I Systemutviklingsoppgave er ikke de vitenskapelige resultatene i fokus, men fremdeles et viktig element å ta hensyn til. Brukertester er et godt eksempel på ett vitenskapelig resultat som har hatt stor påvirkning på det ferdige produktet.

4.1.1 Brukertester

Resultatene fra brukertestene ble rapportert i vedlegg 6 og 7, det ble gjennomført to brukertester. En på wireframe og en på MVP, disse to rundene hadde seks branningeniører som brukere. Dette var alle tilgjengelige brukere i målgruppen, derfor ble testene gjennomført på de samme personene. Bruketestene hadde som overordnede mål å teste brukervennlighet av navigasjon og design.

Brukertesten av wireframe avdekket flere feil og mangler i design, samt hjalp teamet ta noen valg angående navigasjon og layout. Et element som ble diskutert var hvorvidt applikasjonen skulle ha en vanlig navigasjonsbar på toppen av siden eller en festet på venstre side. Resultatet av brukertesten var at det var likt i meningsmålingen. Dette viser at det ikke var en klar preferanse for hvor menyelementene skal ligge.

Flere knapper i applikasjonen var lite intuitive, dette inkluderer databasefeltet i navigasjonsbaren, lagre prosjekt knappen i nytt prosjekt og bytte språk knappen. Det å kunne bytte språk ble forkastet som funksjonalitet underveis, mens databasefeltet ble endret til alle prosjekter. Knappen for å lagre prosjekt samt andre knapper i programmet fikk samme formgivning, slik at de ble lett gjenkjennelige.

Brukertesten av MVP gav verdifull innsikt i mangler og feil programmet hadde som har blitt rapportert i. Dette inkluderte detaljer i felter som skulle fylles inn, og ønsker om å kunne laste opp ferdige rapporter i systemet. Tilbakemeldingene og innsikten gitt var essensiell, da teamets kunnskap i brannsikkerhets fagfeltet er begrenset. Det gav også et perspektiv på hvordan programmet kom til å bli brukt i praksis, da en MVP er minner svært om det ferdige produktet.

Ett element som viste seg å være lite intuitivt og har vært en diskusjon innad i teamet er legg til fravik knappen kontra, nesteknappen på fraviksdelen av nytt prosjekt. Disse ble forvirret under brukertesten, da legg til fravik knappen var plassert langt unna neste knappen. For å gjøre brukeren mer oppmerksom på knappen ble den lagt over nesteknappen. Det ble også gitt tilbakemeldinger på Word-fila dette var ikke en del av wireframe brukertesten, da denne følger en mal allerede i bruk av branningeniører. Generelt var tilbakemeldingen at fila så riktig ut utenom noen små korreksjoner.

Brukerne synes applikasjonen var intuitiv og designet var bra, dette kan også observeres gjennom at det var ingen som hadde store problemer med å navigere applikasjonen.

4.1.2 Wireframes og MVP

Wireframes ble brukt for første runde med brukertesting, mens andre runde med brukertester ble utført på MVP. Flere versjoner av wireframes ble laget, hvor første versjonen var en klikkbar PDF-fil som kan sees i vedlegg 10. Denne versjonen var brukt for den første brukertesten, hvor tilbakemeldinger og forslag fra brukertesten og oppdragsgiver var utgangspunktet for de neste versjonene av wireframen.

Andre versjonen av wireframes var lagd i Adobe XD som også er en klikkbar prototype. Den ble laget med flere detaljer enn første versjonen og følger Rambøll's fargepalett som brukes i sluttproduktet. Dette kan sees i vedlegg 12. Den tredje versjonen av wireframes er en oppdatert versjon av første versjon etter tilbakemelding og forslag fra første brukertesting og fra oppdragsgiveren med unntak av fargepalett til Rambøll. Sammenlignet med første versjonen av wireframes ble tredje versjonen utformet med grundigere detaljer som vedlegg 11 viser.

Den første og tredje versjonen av wireframes følger ikke fargepaletten til Rambøll siden disse var brukt mest som veiledning til å sette opp hvor ulike elementene i applikasjonen skal plasseres. Den andre versjonen var laget med farger, med flere detaljer enn den første versjonen for å visualisere applikasjonen med fargene. Å ha tre versjoner av wireframes kan virke som mye for en enkelt applikasjon, men det hindret store endringer på applikasjonen under utviklingsfasen siden wireframes ble testet og revidert etter tilbakemeldinger og forslag fra brukertesting, og i flere runder med oppdragsgiver.

MVP er applikasjonen som ble utviklet med all sentral funksjonalitet med fargepaletten til Rambøll. I denne fasen, ble MVP testet med brukerne slik at teamet kan oppdage eventuelle feil, eller viktige endringer som må gjøres før applikasjonen klargjøres for lansering. Oppdragsgiveren også hadde mulighet til å teste applikasjonen grundig. Testing på MVP også avslørt småfeil, og ting som manglet. Dette hjalp teamet til å forbedre applikasjonen slik at den blir mer brukervennlig og forståelig til sluttbrukere.

4.2 Funksjonelle egenskaper

Funksjonelle egenskaper	Fullført	Ikke fullført	Kommentar
Eksportere skjema til .docx format	X		
Brukere skal kunne redigere prosjekter de har opprettet	X		
Brukere skal kunne logge inn	X		

Brukere skal kunne se en enkel oversikt over tidligere rapporter skrevet av dem, både gjennomførte og aktive oppdrag	X*		*Delvis fullført, brukere kan se oversikt over tidligere prosjekter skrevet av dem, men dette er ikke delt opp i gjennomførte og aktive oppdrag.
Brukere skal kunne se en enkel oversikt over alle prosjekter som ligger i databasen.	X		
Brukere kan dobbeltklikke et prosjekt, eller trykke på “Mer info” knappen for å se et sammendrag over valgte prosjekt.	X		
En skal kunne opprette et nytt prosjekt	X		
Bruker skal få et sammendrag av prosjektet de har laget rett etter det er opprettet		X	I stedet for å vise sammendrag av prosjektet etter det har blitt opprettet, blir brukeren tatt til «Mine prosjekter» siden, der de kan se at prosjektet har blitt lagt inn i tabellen.
Bruker skal kunne fylle inn felter med generell informasjon om prosjektet f.eks. prosjektnavn og nummer	X		
Bruker skal kunne fylle inn informasjon om selve bygget, hvorav noen felter er nedtrekks menyer	X		
Det skal også være mulig for bruker å fylle inn hvilke fravik et bygg har, med mulighet til å legge inn flere fravik om det er nødvendig	X		
Når brukere skal velge preakseptert ytelse ved hjelp av bokstaver og punkter, så skal bare teksten for ytelseskrav vises.	X		
Bruker skal kunne sammenlikne fravik i et bygg med fravik i liknende bygg ved hjelp av visuelle oversikter med grafer ved hjelp av benchmark knappen.	X		Ordet Benchmark er gjort om til Statistikk
Bruker skal kunne bruke en benchmark funksjonalitet for å se statistikk for bygg med tilsvarende virksomhet, og en knapp for å trykke og bruke samme fravikende løsning i sitt eget prosjekt.	X*		*Delvis fullført, en kan se på statistikk med tilsvarende virksomhet og risikoklasse, men kan ikke trykke på for å bruke samme fravikende løsning i eget prosjekt

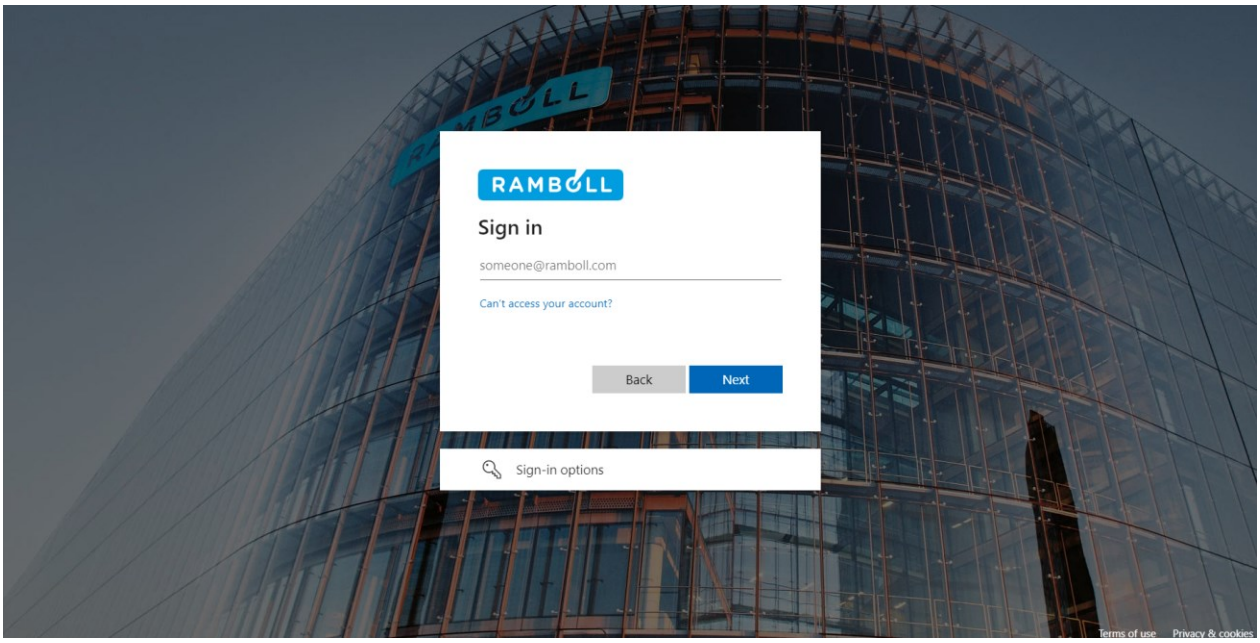
Det skal være mulig å sortere listen over alle prosjekt på ulike faktorer	X		
Det skal være mulig å filtrere listen over alle prosjekt på ulike faktorer både forsiden som viser alle prosjekter og i Benchmark siden	X		Ordet Benchmark er gjort om til Statistikk. Se vedlegg 3 for liste med faktorer som kan filtreres.
Bruker skal kunne se andres prosjekt	X		
Brukeren skal kunne se når og hvem som har redigert på et prosjekt på siden som viser mer info om et prosjekt.		X	Valgfri funksjonalitet som ikke kunne implementeres grunnet lite tid.
Det skal være en side som beskriver programmet	X		
Det skal være en “Ofte stilte spørsmål” side		X	Forkastet av oppdragsgiver
Det skal være et “hjelp” avsnitt med kontaktinfo til it-support hos bedriften		X	Forkastet av oppdragsgiver
Brukere kunne generere statistikk grafer med all data som ligger i databasen.	X		
Brukere skal få tilbakemelding om forespørsler til databasen var suksessfullt eller ikke.	X		
Brukere skal få varsel om det blir prøvd å lukke et skjema uten å lagre det eller slette et oppdrag fra databasen, slik at brukeren kan bekrefte handlingen sin.	X		
Brukere får tilbakemelding om de prøve å fortsette/lagre skjema uten å fylle ut/velge alle feltene.	X		
Det skal være lett for brukere å navigere gjennom webapplikasjonen ved å bruke en navigasjonsbar.	X		
Når brukeren oppretter et nytt prosjekt, så skal de kunne de se sin egen progresjon når de fyller ut skjema.	X		

Sluttdato på et oppdrag skal settes til dagens dato som standard, med mulighet til å endre datoen.	X		
Applikasjonen skal være på norsk som standard språk, med mulighet for brukeren å endre det til engelsk ved å klikke på en språk knapp/ikon.		X	Forkastet av oppdragsgiver
Brukere skal kunne laste opp bilder til prosjektene de har opprettet.		X	Forkastet av oppdragsgiver
Brukere skal kunne være logget inn for en lengre periode (noen timer), med en timeout funksjon. Hvis tiden går ut og brukeren har ikke lagret prosjektet, så får brukeren et varsel for å lagre prosjektet.		X	En høy prioritet funksjonalitet som måtte droppes begrunnet lite tid
Bruker skal få et valg om å fortsette å jobbe videre på et tidligere prosjekt som ble forlatt uten å lagres ved å lukke nettsiden.		X	En lav prioritet funksjonalitet som ble droppet begrunnet lite tid. Brukeren får varsel at data blir mistet om de prøve å lukke nettsiden uten å lagre prosjekt.

Tabell 4.1 Tabell over funksjonelle egenskaper

Nær alle funksjonelle krav oppdragsgiver ønsket har blitt fullført, det var flere krav som ble forkastet underveis da oppdragsgiver ikke ville ha de med. Dette inkluderer funksjonaliteten for å endre språk, FAQ siden, kontaktinfo til IT-avdeling og kunne laste opp bilde av prosjekt.

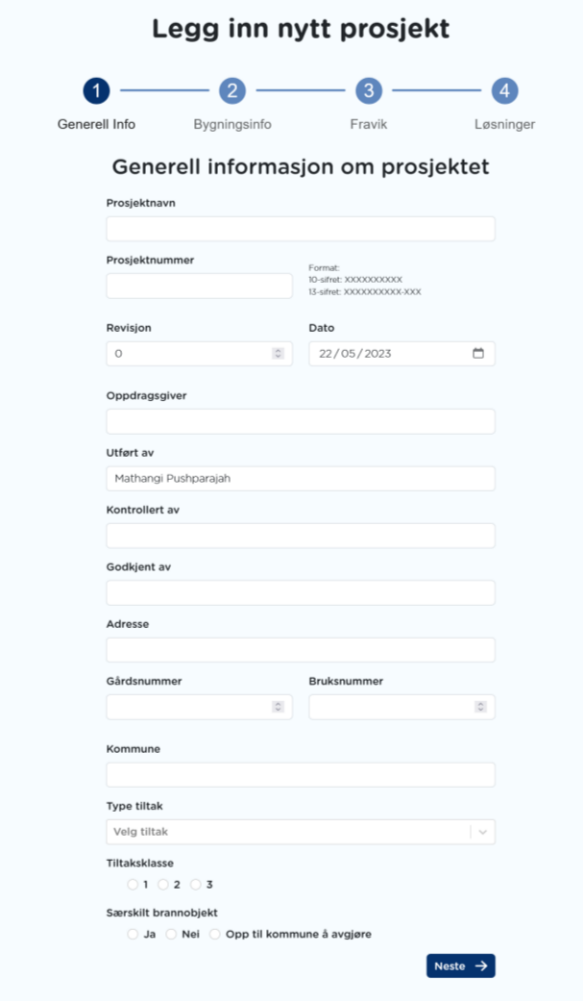
Den fullførte funksjonaliteten kan grupperes inn i 7 kategorier: Nytt prosjekt, Mine prosjekt, Statistikk, Om prosjekt, Alle Prosjekt, prosjektinfo og navigasjon.



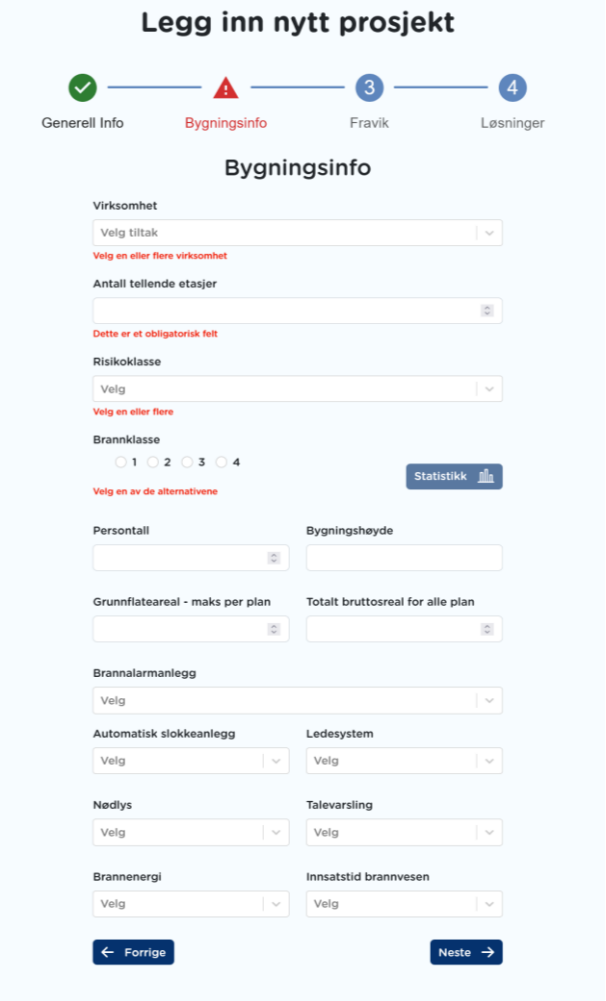
Figur 4.1 Innlogging til Rambølls konto, dette kreves for å ha tilgang til nettsiden

4.2.1 Nytt prosjekt

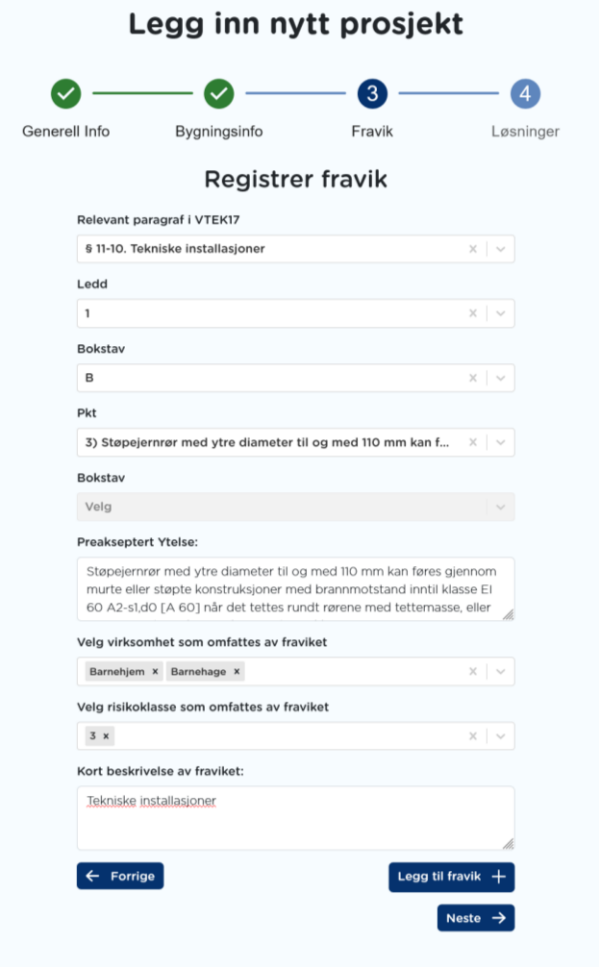
For å lage ett nytt prosjekt må brukeren fylle ut ett skjema, dette inneholder alle relevante felter som er inkludert i malen som genereres når det er ferdig fylt ut. Her er det validering av mange felter for å forsikre at brukeren ikke kan fylle inn feil type data, samt ikke gå videre uten å fylle ut et minimum antall felter. Siden har også en progresjonsmeter på toppen av skjemaet som gir oversikt over hvor langt de har kommet, og lar bruker navigere frem og tilbake mellom sidene.



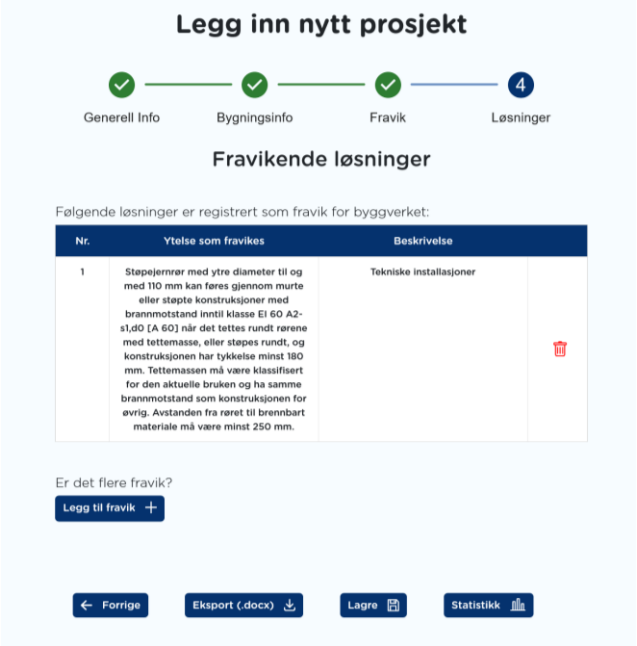
Figur 4.3 Første side av nytt prosjektskjema, bilde er beskjært for å få plass til resten av skjemaet.



Figur 4.2 Andre side av nytt prosjekt skjema, inneholder felter med informasjon om bygningen



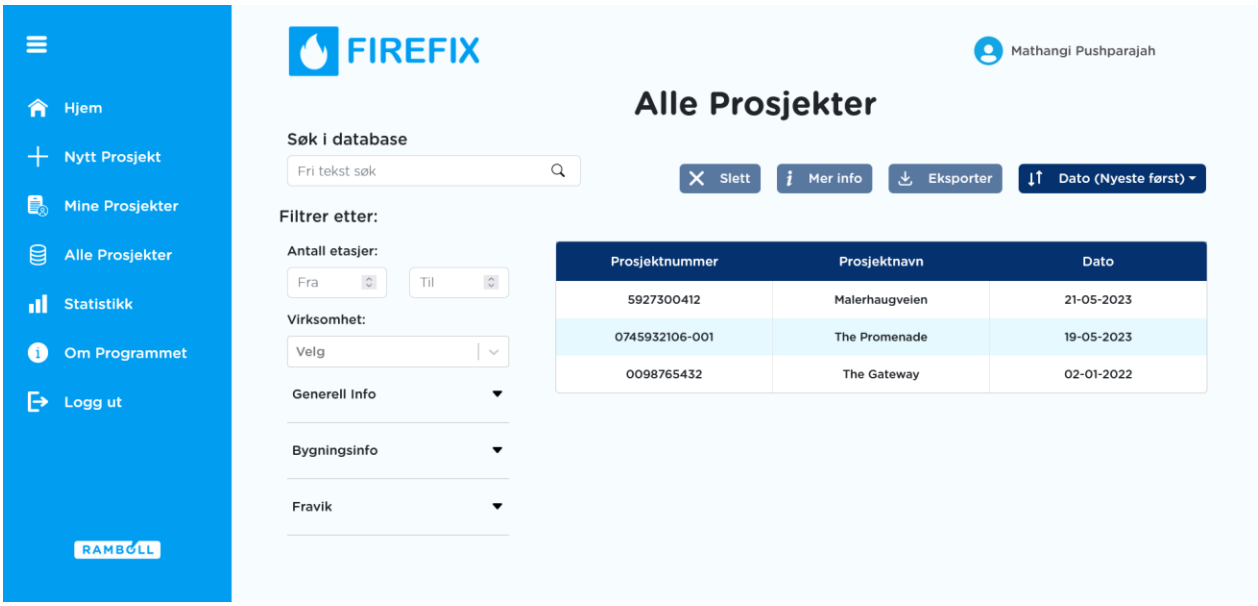
Figur 4.5 Tredje side av nytt prosjekt skjema, bruker kan fylle ut fravik til prosjektet



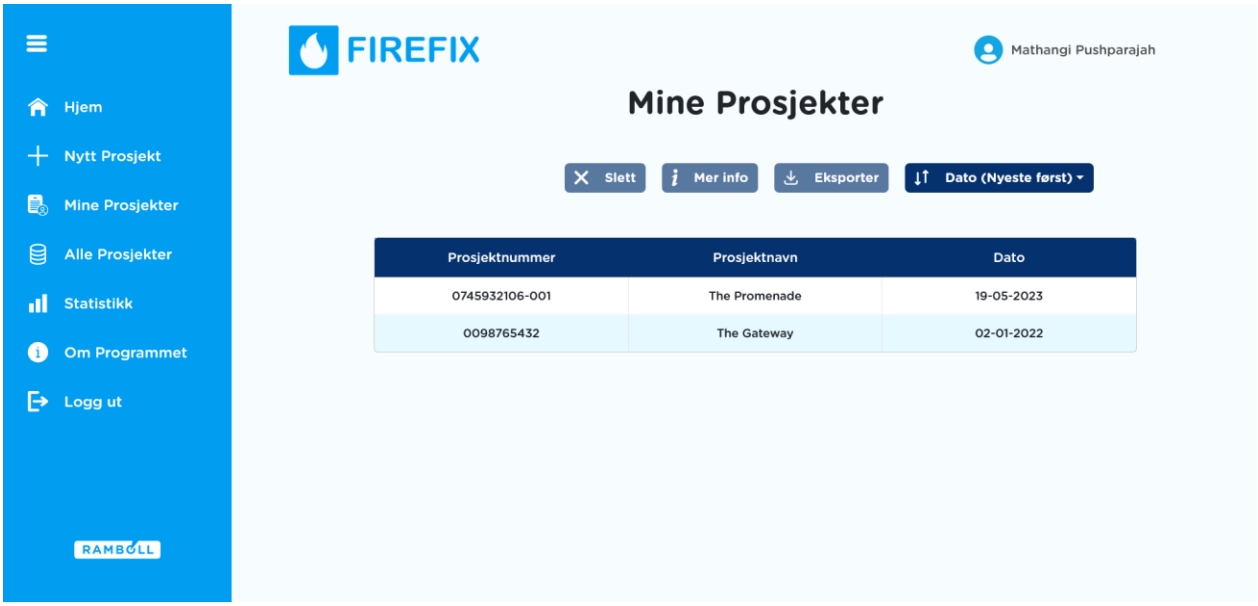
Figur 4.4 Fjerde side av nytt prosjekt skjema, inneholder oversikt over alle valgte fravik. Har også knapper for eksportering til Word og se statistikk over likende prosjekter.

4.2.2 Alle- og mine prosjekter

Alle prosjekter siden inneholder mye funksjonalitet for filtrering og sortering av prosjektene slik at det er lett for brukeren å finne prosjektet de leter etter. Filtreringsalternativene er sortert etter kategori, for en oversiktlig fordeling. All filtrering, søking og sortering er håndtert av tjener.



Figur 4.6 Siden for alle prosjekter, har mulighet for å søke og filtrere en rekke parametere



Figur 4.7 Side for Mine prosjekter, viser alle prosjekter brukeren har laget

4.2.3 Prosjekt informasjon

For å se detaljert informasjon om et gitt prosjekt dobbelttrykker man på ett tabellelement eller mer infoknappen. Prosjektinformasjons siden inneholder all viktig informasjon fra prosjektet, samt muligheten til å eksportere informasjonen til en Word-mal. Brukeren kan også laste opp eller ned en ferdig rapport for det gjeldene prosjektet.

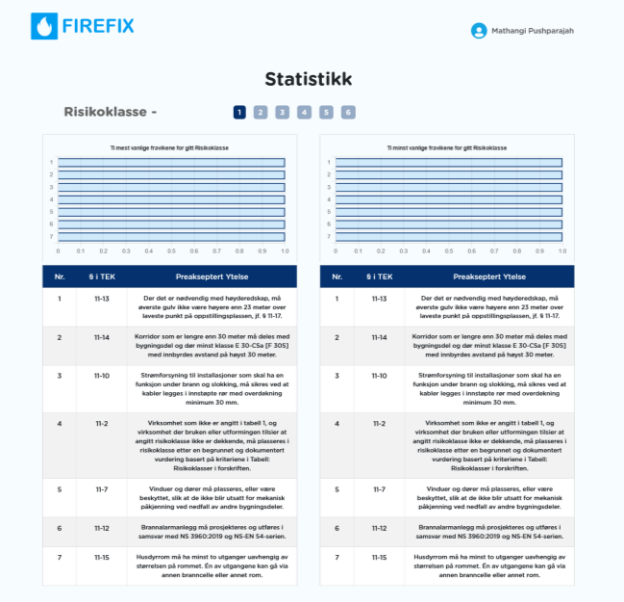
Figur 4.8 Side for prosjektinformasjon, inneholder mer detaljert informasjon for gitt prosjekt. Har også mulighet til å redigere prosjektet. Det er også funksjonalitet for å kunne laste opp ferdige rapporter, samt laste ned en ferdig rapport.

4.2.4 Word

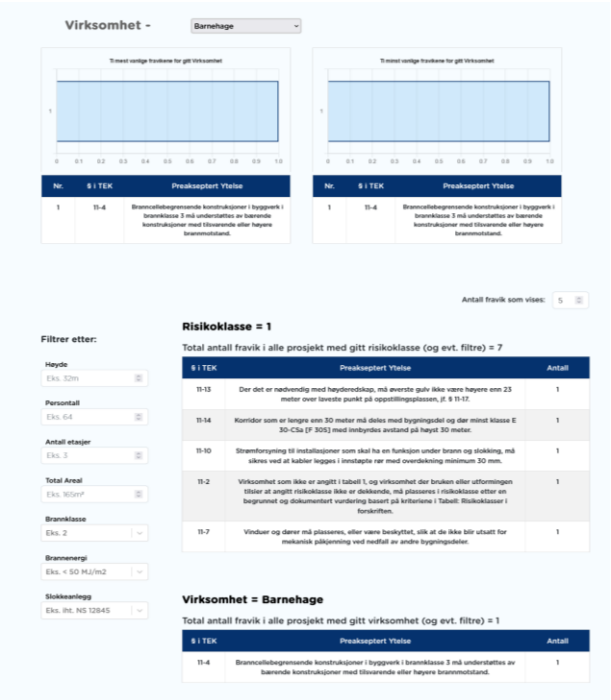
En sentral funksjonalitet av applikasjonen er muligheten til å kunne eksportere prosjekter til en ferdig utfylt Word-mal. Dette lar brukeren lett skrive ferdig rapporten for det gitte prosjektet. Denne Word-malen er en ferdig utfylt versjon av malen Rambølls branningeniører bruker ved rapportering av prosjekter. Når rapporten er ferdig skrevet kan den lastes opp av brukeren, slik at andre branningeniører kan finne den i alle prosjekter. For å se et eksempel av malen se vedlegg 13.

4.2.5 Statistikk

Å kunne se statistikk over de vanligste fravikene i samme risikoklasse og samme virksomhet var funksjonalitet oppdragsgiver hadde høy prioritet på. Funksjonaliteten kalt Benchmark ble innlemmet i statistikk da oppdragsgiver og teamet ble enige om å ha en felles side. Denne siden er tilgjengelig fra menyen eller nytt prosjekt. Den gir oversikt over de ti mest valgte fravikene for gitt risikoklasse og virksomhet. Den inkluderer også en tabell som gir oversikt over alle fravik valgt for den gitte klassen.



Figur 4.10 Første halvdel av statistikkside, inneholder statstikk over fravik for valgt risiko klasse.



Figur 4.9 Andre halvdel av statistikkside, inneholder statstikk over virksomhet og tabeller med fravik som kan filtreres.

4.2.6 Responsiveness

Web-applikasjonen er designet med tanke på et responsiv design for å sikre at den ser ut og fungerer bra på forskjellige skjermstørrelser. Det er imidlertid verdt å merke seg at appen ikke er beregnet for bruk på smarttelefoner, da dette var spesifisert av oppdragsgiver som unødvendig. I stedet har det vært fokus på å sikre at appen fungerer godt på datamaskiner og nettbrett.
For å oppnå dette var flere teknikker brukt i CSS-koden. En av disse var "@media" egenskapen, som tillater å lage forskjellige versjoner av deler av CSS koden for forskjellige skjermtyper. Dette betyr at oppsettet og stilen til appen kan tilpasses størrelsen på skjermen den vises på.

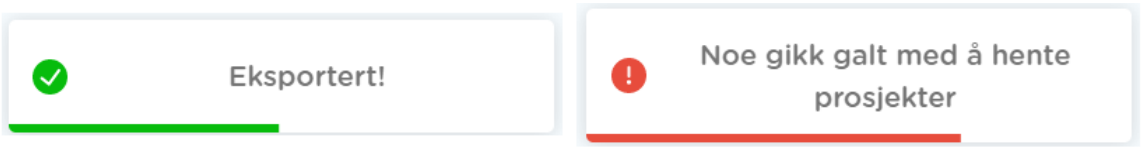
En annen teknikk som ble brukt var å spesifisere dimensjoner i prosenter i stedet for piksler. Dette tillater teamet å sikre at elementer som bredde, høyde og marginer alltid var i forhold, og uavhengig av skjermstørrelsen. Det ble også brukt piksel-verdier for egenskaper som "max-width" og "min-width" for å forhindre at elementer blir forvrengt på veldig store eller veldig små skjermer.

I tillegg til disse teknikkene brukte teamet også CSS-animasjoner og overganger for å forbedre brukeropplevelsen. Disse bidrar til å gjøre appen mer sømløs og komplett. For eksempel ble knapper, nedtrekks-menyer og tabellelementer gjort responsive ved å endre opasiteten når brukeren holder markøren over eller et element er deaktivert. Et annet eksempel på bruk av CSS-animasjoner er i sidebar i appen. Sidebar-en er laget slik at den kan trekkes inn og ut, noe som gir brukeren mer plass å jobbe med. Dette bidrar til å

forbedre den generelle brukervennligheten til appen og gjør det lettere for brukere å fokusere på arbeidet sitt.

Til slutt fulgte teamet nøye med på hvordan feil, varslinger og meldinger ble vist til brukeren. Valideringsfeil ble vist under hvert input-felt i rødt slik at brukerne enkelt kunne se hvor eventuelle problemer var. For generelle meldinger, som suksess-meldinger eller andre feilmeldinger, brukte vi Toast-varsler. Disse ble timet og animert, med en grønn hake for vellykkede operasjoner og et rødt utropstegn for feil (Se figur 4.11).

Totalt sett, har fokuset på å ha ett responsivt design bidratt til å sikre at web-applikasjonen ser ut og fungerer bra på en rekke enheter. Ved å bruke teknikker som "@media" egenskapen og prosentbaserte dimensjoner har det blitt laget et fleksibelt layout som tilpasser seg ulike skjermstørrelser. I tillegg har bruk av CSS-animasjoner og overganger bidratt til å forbedre brukeropplevelsen og få appen til å føles mer polert. Med tanke på videre utvikling, applikasjonen kan veldig enkelt videre utvikles til å kunne tilpasses mobiltelefoner, ved å legge til en ny "@media" egenskap med riktig skjermstørrelse og gjøre endringer på CSS-koden bare hvor nødvendig.



Figur 4.11 Toast-varslinger, suksess-melding og feilmelding

4.3 Ikke funksjonelle egenskaper

De ikke funksjonelle egenskapene som ble definert i visjonsdokumentet (vedlegg 3) inkluderer enkel utvidelse, raskt og tilgjengelig, sikkerhet, brukergrensesnitt. Det er også verdt å nevne personvern da GDPR er viktig å følge.

4.3.1 Enhetstester

Testing var viktig i utviklingen av tjenersiden programmet, da det er god test dekning av «service» metodene, og testing av Word-fil generering. Testingen av Word-fil delen av programmet er ikke veldig dyp, men vil finne ut av om tabeller lages riktig og om selve Word-fila lages. Testene av service bruker et eller flere testprosjekt for å gi data inn og ut, disse vil fange om noen av metodene stopper å fungere. Det er ingen tester av kontroller metodene da dette ikke er naturlig å gjøre med enhetstesting siden kontroller metodene kun tar imot og sender api-kall. Derfor er det naturlig å teste kontroller metodene med ende til ende testing. Det er totalt 68% på dekningsgraden til testene. Mer om dette kan leses i vedlegg 5. Teamet vurderte det som ineffektivt å teste klient, da tidsbegrensinger gjorde at tjener tok prioritet. Klienten ble testet jevnlig gjennom å kjøre applikasjonen, samt flere forsøk på å ødelegge den.

4.3.2 Personvern
Applikasjonene har fulgt GDPR, da det ikke lagres noen informasjon om brukeren annet enn epost. Det er heller ikke brukt noen cookies for å spore brukeren, men det lages en cookie ved innlogging da dette er en del av Microsoft sitt innloggingssystem.

4.3.3 Sikkerhet
Sikkerheten til systemet er et element som for det meste ikke løses av applikasjonen selv med av Azure. For å ha tilgang til nettsiden må man logge inn med en Rambøll konto, dette gjelder også alle api-kall da både klient og tjener kjører på samme App service.

Ett annet område hvor sikkerhet er viktig er databasen, her løses sikkerheten av Azure AD med Azure Managed Identity, hvor passordet til databasen aldri lagres i ren tekst hos tjener, men heller bruker en token for å få tilgang til databasen. App Service sender forespørsel til Azure Managed Identity om å få tilgang til databasen, og hvis App Servicen har lov til å aksessere databasen, så sender Azure Managed Identity en forespørsel til Azure AD for å hente token. Da token er sendt til App Service, kan App Service bruke token til å få tilgang til databasen. Databasen sjekker om token stemmer med Azure AD for å autentisere forespørselen.

4.3.4 Brukergrensesnitt
Det er flere ikke funksjonelle egenskaper som omhandler det generelle brukergrensesnittet til applikasjonen. Dette inkluderer universell utforming, intuitiv navigasjon og at den er lettlært.

Oppdragsgiver hadde som ønske å følge fargepaletten til Rambøll som kan sees i figur 5.1, de to hovedfargene i denne paletten følger ikke universell utforming. Den sier også at det er nødvendig å bruke nok cyan og hvit for at programmet skal være gjenkjennelig som et Rambøll-program.

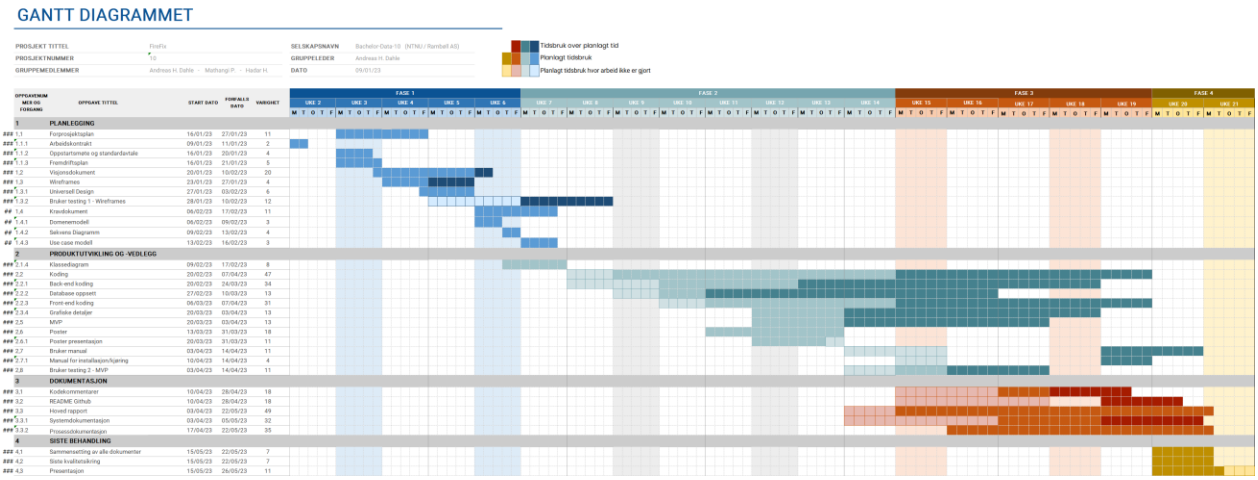
For å sikre at navigasjonen er intuitiv ble det holdt to runder med brukertester, hvor navigasjon var det viktigste punktet. Det var et fokus på holde navigasjonen til færrest mulig klikk, men samtidig holde alt ryddig. Når det kommer til å gjøre applikasjon lett å lære så var intuitivitet viktig og om programmet siden som gir brukeren informasjon om hvordan programmet brukes.

4.3.5 Produksjonssetting
Applikasjonen er hostet på en App Service som er en del av Azure sine skytjenester, hvor databasen er hostet i en MySQL-database. Måten applikasjonen har blitt produksjonssatt er ved å bruke GitHub Actions til å kjøre «Firefix-Backend» repository med en «build» mappe som inneholder static assets til klient. Static assets er CSS, Javascript og bildefiler som beskriver utseende til de ulike sidene av web-applikasjon og hvordan funksjonalitetene skal oppføre seg.

4.4 Administrative resultater

4.4.1 Måloppfyllelse i forhold til fremdriftsplan
Fremdriftsplanen satt opp i Gantt diagrammet ble fulgt etter beste evne, det ble en stor forsinkelse på utviklingen forårsaket av venting på svar fra it-avdelingen til Rambøll. Det ble

sendt e-poster som endte opp i søppelposten til IT-ansvarlig, som hindret produksjonssetting til Azure. For å minske forsinkelser i progresjon begynte teamet med rapportskriving før applikasjonen var rullet ut til Azure.



Tabell 4.2 Gantt diagram hvor lyse farger er planlagttidsbruk som ikke ble gjennomført, vanlig farger er planlagt tidsbruk og mørke farger er tidsbruk over planlagt.

4.4.2 Timeregnskap

Et krav i bacheloroppgaven var å føre timelister sortert på aktivitet, teamet oppdaterte timelistene kontinuerlig. Hver uke ble statusrapporter laget som oppsummerte antallet timer brukt og hva hvert teammedlem hadde gjort. I tabell 4.3 kan en se teamets timebruk per kategori, og det fulle timeregnskapet ligger i vedlegg 9.

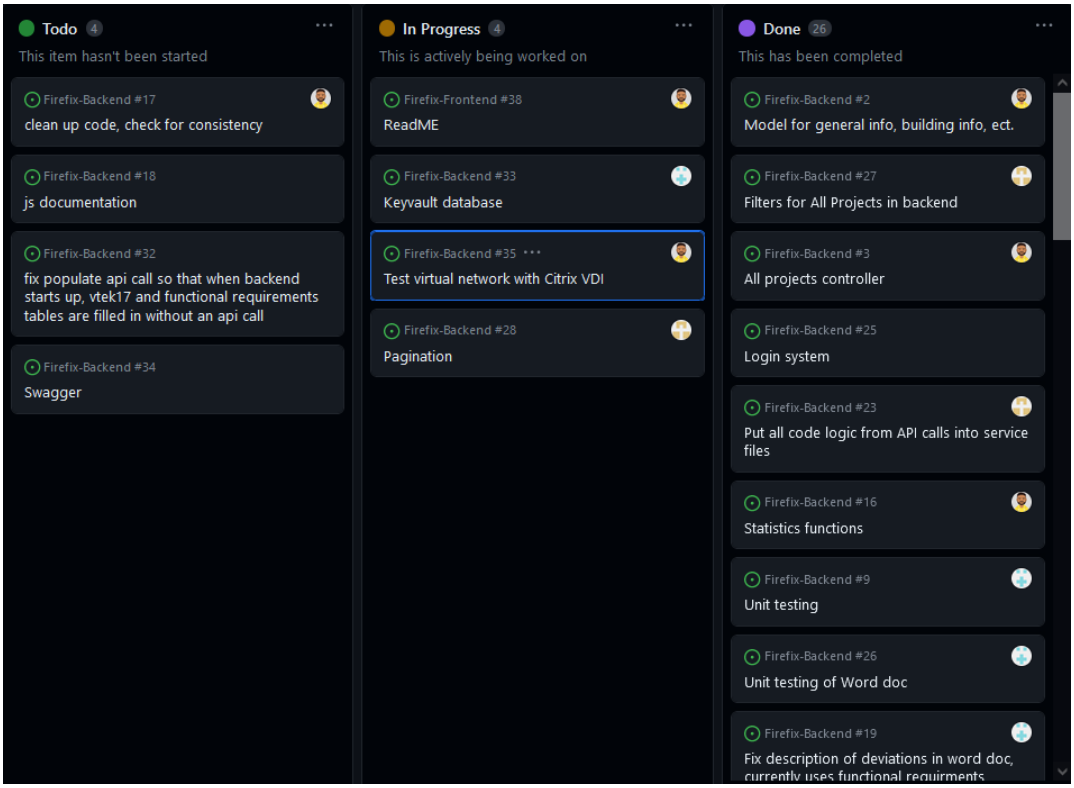
Aktivitet	Sum timer/aktivitet
Selvopplæring	**31.5**
Informasjonssøk	**117.0**
Brukertesting	**38.5**
Prototyping	**45.5**
Koding	**571.0**
Testing av kode	**46.0**
Feilretting	**29.5**
Prosjektrapportering	**175.0**
Presentasjon inkludert forberedelse	**0.5**
Team møter	**164.4**
Team møter med veileder	**7.5**

Formell dokumentasjon	**23.5**
Arbeid med forprosjektplan	**20.0**
Arbeid med visjonsdokument	**19.5**
Arbeid med kravdokumentasjon	**9.5**
Arbeid med fremdriftsplan	**5**
Vitenskapsteori og -metode	**22.5**
Tegning	**17.5**
Variert	**54.5**
Use case	**1.2**
Sekvensdiagram	**2**
Sum timer	**1401.6**

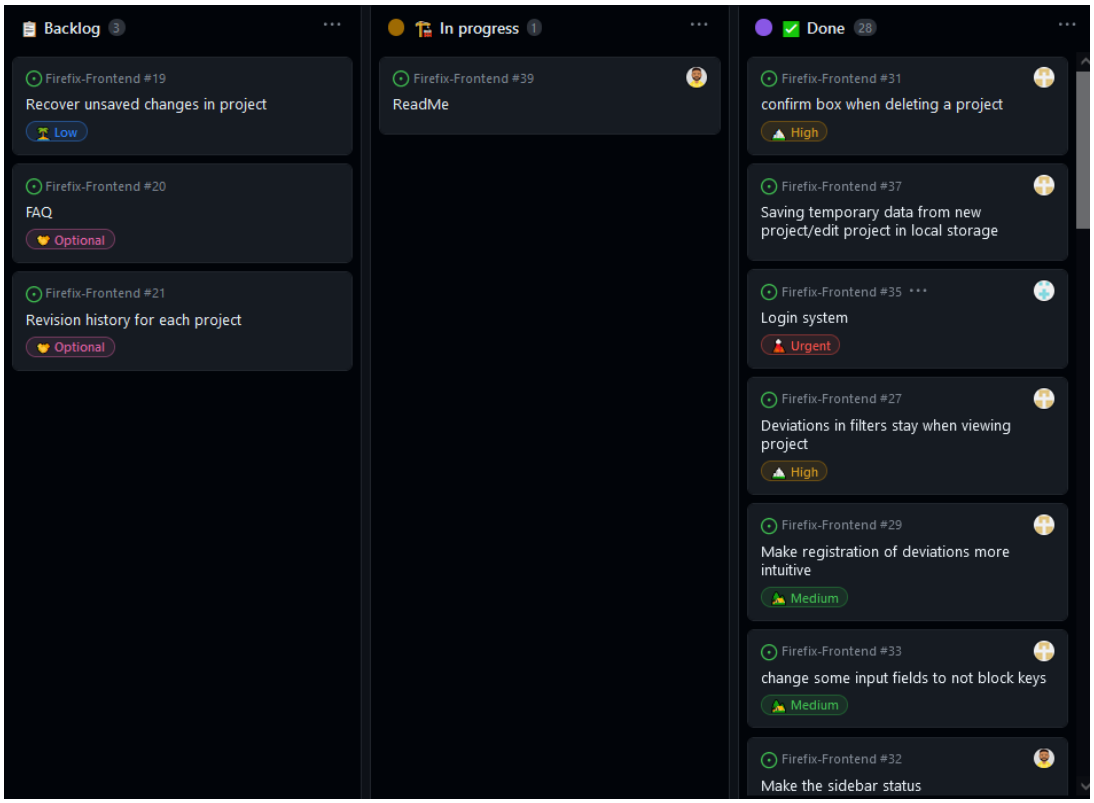
Tabell 4.3 Oversikt over timebruk fordelt på oppgave

4.4.3 Utviklingsprosess

Teamet brukte Kanbantavler under oppgaven og delte oppgavene i prioritet bekreftet av oppdragsgiver. Github prosjekter ble brukt for å lage Kanban tavlene, en for klient og en for tjener. Dette lot teamet holde oversikt over hvilke oppgaver som måtte gjøres, og hvem som gjennomførte de. I Figur 4.12, og 4.13 kan en se tavlene.



Figur 4.12 Github issue board for back-end i uke 18



Figur 4.13 Github issue board for front-end i uke 18

5 Diskusjon

5.1.1 Brukertester

Brukertesten av Wireframe gav nyttige resultater for design av en mer ferdig prototype. Det var flere brukere som synes «database» var et lite intuitivt navn for «alle prosjekter», det var også mange som slet med lagring av nytt prosjekt. At flere knapper hadde forskjellig design skapte unødig forvirring blant brukerne, derfor valgte teamet å lage et felles design for knapper i applikasjonen. Videre var det noen elementer som ble forkastet etter testing av wireframe som knappen for å endre språk, da applikasjonen kun skal brukes av ansatte i Norge.

Resultatet av meningsmålingen om menyen skulle være øverst eller festet på siden av applikasjonen var likt, derfor ble ett valg tatt av teamet i samarbeid med oppdragsgiver. Det ble bestemt at det var best å ha en meny festet til siden da dette gav best mulighet for å legge til flere elementer underveis i utviklingen, og til videre utvikling. Dette viste seg å være et godt valg da oppdragsgiver ønsket statistikk-siden tilgjengelig fra menyen og ikke bare fra nytt prosjekt.

Resultatene fra brukertesten av MVP ble brukt for å forbedre applikasjonen, da alle småfeil funnet av brukerne ble rettet opp. Dette inkluderer tilbakemeldinger på felter i nytt prosjekt, og i Word-fila. Disse tilbakemeldingene var verdifulle, da teamet ikke har noe kunnskap om hvordan branningeniører jobber i praksis. De gav også ønsker om funksjonalitet, hvorav et ønske om å kunne laste opp ferdige rapporter ble gitt. Dette forslaget ble tatt opp med oppdragsgiver, som var enige om at det var nyttig funksjonalitet. Derfor prioriterte teamet å gjennomføre denne funksjonaliteten over andre som hadde lavt eller valgfritt prioritetsnivå. Annet innhold som brukerne foreslo ble ikke implementert som kontaktinformasjon til IT-avdelingen, da oppdragsgiver ikke ønsket det i applikasjonen.

Brukerne i begge testene var de samme, dette er vanligvis ikke ønskelig da kunnskapen de har fra første brukertesten kan hindre gode resultater. Dette gjør at man ikke får en gitt brukers intuitive reaksjon og bruk av applikasjonen. Det at ingen slet med å navigere programmet i MVP brukertesten er et tydelig eksempel på at brukerne har kjennskap til applikasjonen fra før. I dette tilfellet var antallet personer i målgruppen som var tilgjengelige for brukertester svært lavt, derfor mente teamet at det var mer verdifullt å teste på alle brukerne i begge runder, enn å teste på tre brukere om gangen. En annen grunn til at det ikke hadde en stor innvirkning på resultatene er at brukertesteten for wireframe og MVP hadde forskjellig fokus. De gikk gjennom liknende oppgaver, men fokus fra wireframe var navigasjon og layout, mens for MVP var detaljer ved input og Word fila i fokus.

5.1.2 Navigasjon

Under brukertestene var posisjonen av navigasjonsmenyen ett spørsmål gitt til brukeren, hvor teamet ikke fikk ett entydig svar dette resulterte i at det ble tatt ett valg om å plassere den på venstre side. Dette gir best skalerbarhet for videre utvikling da det er lett å legge til flere elementer i menyen uten mye arbeid.

5.1.3 Visning av prosjekter

Applikasjonen har to sider for å vise prosjekter, hvor den viktigste er Alle prosjekter. Dette er en side som over tid vil inneholde mange prosjekter derfor var skalering et element som måtte tas hensyn til. Derfor er sortering, og filtrering håndtert av tjener, dette gir mulighet til en mer effektiv behandling av prosjekter. For å kunne oversiktlig vise et stort antall prosjekter ble paginering lagt til, da dette kun gir brukeren ett visst antall prosjekter om gangen og gjør applikasjonen mer oversiktlig. Dette er enda en grunn til at alt håndteres av tjener da API-kallene kun trenger å motta et gitt antall prosjekter og ikke alle prosjekter i databasen.

Styrken med å ha filtrering, paginering, sortering og søk som funksjonalitet i alle prosjekter er at det er lettere å finne fram til prosjektet man leter etter dersom det er veldig mange prosjekter. Svakheten med det er at siden kan være overveldende i at det er mange valg og mye informasjon å ta inn. Dette er håndtert ved å gruppere filtrene, slik at all ikke vises samtidig.

5.1.4 Statistikk

Statistikk siden gir bruker oversikt over de mest og minst valgte fravikene for en gitt risikoklasse og virksomhet. Denne siden har gjennomgått flere iterasjoner da kommunikasjon med oppdragsgiver har spisset inn hva som skulle være på siden. I utgangspunktet var det et ønske om en separat «benchmark» side som man kunne trykke på ett fravik for å legge til i prosjektet man jobbet med. Dette ble forkastet til fordel for en felles statistikk side hvor en kan se fravikene i forhold til sitt eget prosjekt.

En svakhet ved statistikksiden er at den ikke viser noen trender over endring i fravik over tid. Dette er noe som ikke var prioritert høyt av oppdragsgiver, men er av nytte og aktuelt for dersom applikasjonen skal videreutvikles. Et annet element som kunne vært implementert er muligheten for å trykke på ett fravik å legge det rett inn i prosjektet, dette ble nedprioritert i utviklingsfasen da statistikksiden ble konkretisert.

En styrke ved siden er at alle fravikene i stolpediagrammene ligger også i tabellene sortert i samme rekkefølge. I tabellen ligger også den preaksepterte ytelsen som kan gi brukeren en bedre ide om hvilket fravik det er i tabellen.

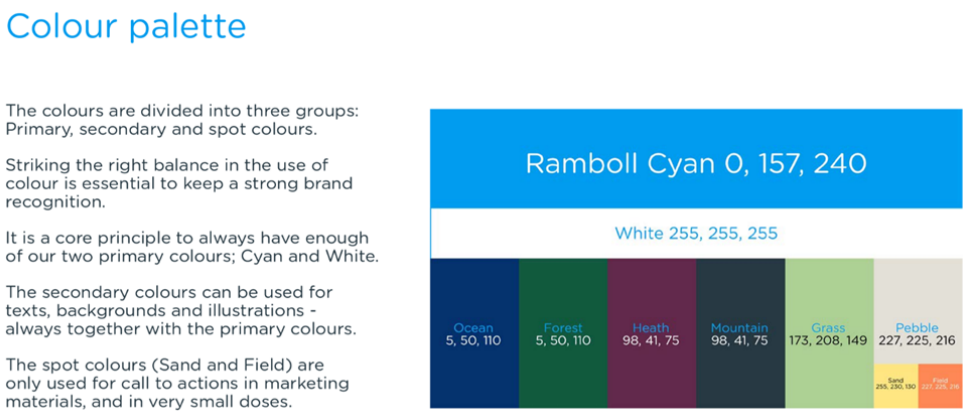
5.1.5 Eksport av Wordfil

Word-filen programmet generer er tilnærmet lik malen oppdragsgiver gav teamet, med noen begrensninger. Som at pakken som ble brukt for generering ikke støtter en kildeliste, derfor er denne laget manuelt, den største forskjellen med dette er at den ikke oppdaterer seg automatisk. En annen begrensning er at det dukker opp et varsel om at Word filen referer til andre filer når man åpner fila første gang. Dette kommer av måten Word lager innholdsfortegnelser på og blir borte dersom man redigerer noe i fila.

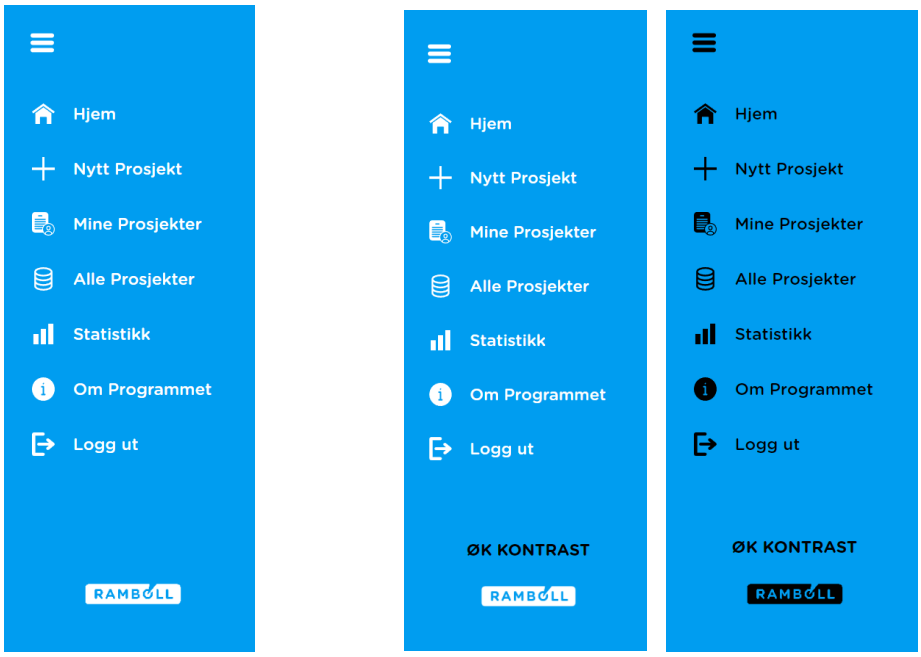
Styrkene til genereringen er at alt fungerer som det ville gjort i en vanlig Word-fil som lister, tabeller osv. I tillegg reduserer det arbeidsmengden til ansatte hos Rambøll da malen er ferdig utfylt. En svakhet i måten eksporten ble programmert på er at det er svært mange metodekall, dette kan reduseres ved videreutvikling ved å kun ha en metode for paragraf og lagre teksten i variabler. Årsaken til metodekallene er en kombinasjon av lite erfaring med pakken og måten dokumentasjonen er laget på.

5.1.6 Fargepalett

Fargene i navigasjonsmenyen og logo oppfyller ikke WCAG2.1, da kontrasten mellom cyan og hvit ikke er høy nok. Applikasjonen ble laget etter designprofilen og fargepaletten til Rambøll etter krav fra oppdragsgiver. Den offisielle fargepalleten som en kan se i Figur 5.1 sier at det skal være nok bruk av cyan og hvit for at applikasjonen skal være gjenkjennelig som et Rambøll-produkt. Siden applikasjonen skal brukes internt hos Rambøll hvor alle applikasjoner har denne fargepaletten, og de har ikke hatt noen problemer med det, derfor har oppdragsgiver vurdert det som ok å bruke denne fargekombinasjonen.



Figur 5.1 Rambølls fargepalett



Figur 5.2 Navigasjonsbar uten og med kontrast

Et kompromiss som kunne ha blitt brukt for å oppfylle WCAG 2.1, var å legge en knapp i navigasjonsbaren som lar brukeren til å endre fargen på teksten til en farge som følger

WCAG 2.1. Som bildene til høyre i Figur 5.2 viser, ved å trykke på «ØK KONTRAST» ville tekst, ikoner, og logoen på navigasjonsbar endret til svart som oppfyller krav i WCAG 2.1 om at lys farger skal ha mørk tekst for god kontrast. Ved å ha knappen til å endre kontrasten, ville det ha fulgt fargepaletten til Rambøll, men samtidig være tilgjengelig for alle uavhengig av forutsetninger. Etter tilbakemelding fra oppdragsgiver, var det bestemt å holde til det første designet som bilde til venstre i Figur 5.2 viser.

5.1.7 Produksjonssetting

Når det kom til produksjonssetting var ikke Azure et krav, men fordelen med det var enkel integrasjon med de eksisterende tjenersystemene og kontoene. Det var tidskrevende å forstå og teste om produksjonssettingsprosessen fungerer, og teamet opplevde noen vanskeligheter underveis. Det at teamet valgte å rulle ut applikasjonen etter å ha lagd en fungerende versjon hjalp heller ikke siden det ble oppdaget flere problemer der det måtte kontaktes ulike personer for å finne løsning på det. Disse var blant annet, ingen eller begrenset tilgang til Rambøll ressurser. Noen av forespørslene som ble sendt til Rambøll's tekniske hjelp tok lengre tid enn forventet, særlig når kontaktpersonene bor i en forskjellig tidssone enn teamet som var i Norge.

Hva som kunne ha blitt gjort bedre på denne delen var at teamet skulle ha begynt med produksjonssetting tidligere, i hvert fall det ville ha vært bedre å få utrullet og testet på en testversjon av applikasjonen tidlig i utviklingsprosessen i stedet for å vente til en fungerende versjon av applikasjonen. Hvis en testapplikasjon var utrullet tidlig i utviklingsprosessen, ville teamet ha oppdaget hvor tidskrevende det blir å rulle ut applikasjonen med lite kunnskap om Azure, og om hvilke Rambøll og Azure ressurser teamet trenger tilgang til. Dette ville ha ført til at teamet tok kontakt med ulike personer i god tid for å løse problemene som var opplevd under produksjonssetting.

Noen endringer var gjort på måten applikasjonen ble rullet ut for å oppfylle grunnleggende autentisering på databasen som forklart i neste delkapittel, Sikkerhet. Første produksjonssetting var utført slik at klient og tjener kjørte på en App Service hver. Her, hadde klient autentisering på og dette var ikke særlig vanskelig siden innloggingen samt sikkerhet rundt det ble håndtert av Rambøll's eksisterende innloggingsside. Det oppsto problem da det ble oppdaget at klienten kan ikke sende forespørsler til tjeneren hvis den har autentisering. Etter å ha prøvd å løse dette til ingen nytte, måtte teamet endre oppsettet slik at klient og tjener er utrullet sammen i en App Service. Dette førte til at tjener er beskyttet mot aksess utenfor klient/web-applikasjonen. Ved å bruke GitHub Actions, er koden satt opp slik at når «Firefix-Backend» «repository» blir rullet ut til App Service blir det lagt en «build» mappe med static assets som tilhører klient. På den måten blir klient og tjener utrullet sammen i en App Service.

5.1.8 Sikkerhet

Applikasjonen er produksjonssatt i Azure som er en tjeneste som Rambøll allerede bruker, og tilgangen til ressurser som Rambøll bruker gjør det lettere å integrere applikasjonen med sikkerhet til Rambølls innloggingssystem og tjenersystem. Derfor trenger ikke applikasjonen å håndtere innlogging og lagring av passord. Dette gjør at sikkerheten rundt brukere er høyere enn et innloggingssystem laget av teamet. Da hele applikasjonen er hostet på en App service vil ikke API kall være mulig med mindre man er logget inn på en Rambøll-konto. Dette er fordi

alle API-kall gjøres internt i appservicen. Det lagres også ikke noe annen informasjon om brukeren enn epost, derfor er det ikke noe risiko for lekking av personinformasjon. Databasen er rullet ut til en MySQL tjener i Azure, hvor sikkerheten i kobling mellom tjener og database håndteres av Azure Active Directory og autentisering, fordelen med dette er at det ikke lagres noe passord til databasen i tjenerkoden.

En ulempe med å rulle ut applikasjonen til Azure er at teamet ikke er kjent med teknologien, særlig rundt sikkerhetspraksisen som Rambøll tar i bruk. Det å rulle ut applikasjonen til Azure gikk stort sett greit med informasjonen som var tilgjengelig, men det samme var det ikke for sikkerhet. Det var vanskelig å finne ut hvordan sikkerhetsløsningene skulle implementeres i databasen, og hvordan klient og tjener kan kommunisere på en trygg måte. Dette gjør at det er lett å gjøre feil eller bruke lenger tid enn forventet. Det er også vanskelig å få det testet godt med begrenset kunnskap rundt Azure ressurser.

5.2 Diskusjon av administrative resultater

5.2.1 Fremdriftsplan

Da teamet ikke har erfaring med å planlegge arbeidsprosesser over en lengre tidsperiode, ble noen av tidsmålene overskredet. Dette gjelder spesielt utviklingsfasen da svar fra IT ansvarlig hos oppdragsgiver tok lenger tid enn forventet. Dette gjorde også at prosjektrapporteringen startet en uke senere enn planlagt, siden noen elementer tar tid å få svar på har rapportskriving og utvikling gått i parallell i mye større grad enn planlagt.

Et annet element som tok lenger tid enn estimert var utrulling av applikasjonen til Azure. Det ble prøvd mange forskjellige løsninger på hvordan applikasjonen hostes i Azure. Prøving og feiling ledet til at den bør hostes på en app service og ikke to. Det samme gjaldt autentisering og sikkerhet i kobling mellom database og tjener, samt klient og tjener. Disse faktorene førte til at det tok lenger tid enn forventet. En erfaring som kan være til nytte for andre som skal jobbe med teknologi man ikke er kjent med er å ta hensyn til at det kan ta lenger tid enn forventet.

5.2.2 Utviklingsprosess

Når det kommer til utviklingsprosessen, ble Kanban brukt. Teamet benyttet Github prosjekter sine issue boards som Kanban tavler, for å gi oversikt over alle oppgavene som måtte gjøres. I begynnelsen var alle overordnede oppgaver listet og fordelt mellom teammedlemmene, dette gav liten mulighet for merge-konflikter og hindret at en jobbet med samme oppgave. Svakheten med denne måten å jobbe på er at det å koble sammen deler kan være vanskelig, men god kommunikasjon og samarbeid på oppgaver som overlappet hindret store problemer. For å opprettholde god kommunikasjon ble møter holdt jevnlige innad i teamet, disse var både digitale og fysiske avhengig av tilgjengelighet.

Fordelen med den mer fleksible utviklingen med Kanban tavler er muligheten å tilpasse utviklingen til teamets timeplaner, men den har ulempen at progresjonen ikke er like strukturert som Scrum. Det ble allikevel satt noen overordnede mål som skulle nås, som MVP måtte være ferdig til andre fase av brukertestene. Det ble ikke satt noen mindre delmål på papiret, men innad i teamet ble alle medlemmer oppdatert på når ett gitt medlem tenkte å bli

ferdig med funksjonalitet. Dette fungerte greit, men et mer organisert system hadde fungerte noe bedre får å få bedre oversikt over når oppgaver skal være ferdig.

Gjennom hele prosessen har det vært essensielt å inkludere oppdragsgiver, da tilbakemeldingene har tiltatt endringer underveis. Et problem som teamet støtte på underveis var at den ene av oppdragsgiverne gikk ut med pappapermisjon, dette gjorde at svar tok noe lenger tid, men den resterende oppdragsgiveren var tilgjengelig ved behov.

5.2.3 Gruppesamarbeid

Samarbeidet innad i gruppa har fungert godt, da alle medlemmer har lagt inn omtrent lik arbeidsinnsats. Det har også vært god kommunikasjon, hvor alle stiller spørsmål ved behov og holder hverandre oppdatert med meldinger og møter. Innad i gruppen har det ikke vært noen alvorlige konflikter, som betyr at det ikke har vært nødvendig å ta i bruk arbeidskontrakten. Gruppemedlemmene har sagt sine meningene sin under møter, og diskusjoner om hvilken løsning som skal anvendes har blitt holdt.

6 Konklusjon og videre arbeid

6.1 Konklusjon

Oppgaven skulle håndtere problemstillingen: «Hvordan lage en brukervennlig webapplikasjon for en mer effektiv og dermed bærekraftig måte for brannrådgivere og ingeniører å registrere, lagre, eksportere og sammenlikne prosjekter på?»

Brukertester og meningsmålinger etter disse har hjulpet teamet med å måle brukervennligheten til applikasjonen, brukertestene ble gjennomført i to iterasjoner, en på wireframe og en på MVP. For at de skulle gi relevante resultater ble brukertestene gjort av branningeniører hos Rambøll. Brukertestene gav gode tilbakemeldinger som ble brukt til å forbedre navigasjon og utformingen til programmet. Da brukerne fant det lettere å bruke applikasjonen samt var fornøyd med utseende kan en konkludere med at applikasjonen var brukervennlig.

Gjennom utviklingen ble prinsipper for brukervennlig design og menneske maskin interaksjon brukt. Dette kombinert med kravet om å bruke Rambølls fargepalett gav utgangspunktet til det endelige designet.

Applikasjonen har implementert all kjernefunksjonalitet i problemstillingen og, rullet den ut i Azure skyen til oppdragsgiver. Dette lar brannrådgivere, og ingeniører bruke sin vanlige konto til å logge på nettsiden å få tilgang til all funksjonalitet.

6.2 Videre arbeid

Applikasjonen har noe funksjonalitet som ikke er implementert, dette består av en kombinasjon av nedprioritert funksjonalitet og forslag fra brukertester som ikke ble prioritert av oppdragsgiver.

6.2.1 Gjenoppretting av tapte prosjekter

En funksjonalitet som ble nedprioritert er å gjenopprette et prosjekt dersom en forlater siden uten å lagre. Det er allerede implementert lagring dersom man forlater til en annen side innad i applikasjonen, samt et varsel dersom man prøver å forlate applikasjonen med ett ulagret prosjekt. Det scenarioet som ikke dekkes er dersom pc-en skrur seg av eller at en klarer å lukke hele nettleseren.

6.2.2 Roller

Det å ha redigerings rettigheter til alle prosjekter er noe som ble bestemt tidlig i utviklingen av oppdragsgiver. Dette fungerer i mindre skala, da alle brukerne er ansatte av Rambøll Norge, dersom applikasjonen skal brukes av Rambøll kontorer på utsiden av Norge ville en rollefordeling være naturlig. Slik at man bare kan se og redigere prosjekter innenfor sitt kontor eller land.

6.2.3 Revisjonshistorie og gjenoppretting av tidligere versjoner

Ved videreutvikling er det naturlig å legge til en måte å se revisjonshistorikken til ett prosjekt, da dette gir mulighet til å se hvem og når noen redigerte ett gitt prosjekt. Det gir også mulighet til å resette prosjektet til en eldre versjon, dersom endringer har blitt gjort ved et uhell.

6.2.4 Utdyping av statistikksiden

Grunnet tidsbegrensinger ble avansert statistikk nedprioritert, dette er noe som er naturlig å jobbe med dersom applikasjonen skal videreutvikles. Den kan f.eks. vise trender over valgte fravik, eller la brukeren trykke på ett fravik å legge det til i prosjektet de jobber med.

6.2.5 Sikkerhet

Selv om applikasjonen ikke skal være tilgjengelig for andre enn Rambøll ansatte er sikkerhet fremdeles viktig, til videre arbeid er å integrere applikasjonen i det virtuelle nettet til Rambøll viktig. Dette ble ikke gjennomført da det var store forsinkelser i kommunikasjonen med it-avdelingen til Rambøll.

7 Samfunnspåvirkning

7.1 Bærekraft

Den overordnede samfunns påvirkningen er lav da dette er en intern løsning hos en enkelt bedrift, men bedriftens drift vil effektiviseres og et av deres mål er bærekraftig sikring av bygg. Dette vil indirekte påvirke samfunnet positivt, men ikke i noen stor grad.

Som nåværende løsning bruker bedriften en manuell utfylling av Word-maler og lagrer rapportene i deres filsystemer. Dette ble vurdert som ineffektivt og uorganisert. Ved å fylle i formen i løsningen laget av teamet vil prosessen effektiviseres, programmet lar en også laste opp ferdige rapporter som gjør det lettere å finne andres rapporter. Denne effektiviseringen og sentraliseringen av lagring vil gjøre prosessen raskere. Da Rambøll er en bedrift som fokuserer på bærekraft i deres arbeid vil en effektivisering av deres prosesser øke bærekraften deres.

Livsløpsanalyser er en måte å analysere utslippene og bærekraften til et produkt helt fra produksjon til kasting. Dette inkluderer alt fra produksjonskostnader til transportutslipp til muligheten for resirkulering. Siden man tar hensyn til hele livsløpet, kan en bruke analysen til å ta miljøbevisste valg [58].

Det er vanskelig å gjøre konkrete livsløpsanalyser på datasystemer da de ikke korresponderer direkte med noe fysisk. En kan måle bærekraften av tjenerne som hoster nettsidene som blir laget eller kjøretiden og strømforbruket til datamaskinene programmeringen gjøres på, men i begge tilfeller er det vanskelig å stadfeste konkrete tall på forbruket.

8 Referanser

[1] Direktoratet for byggkvalitet, "§ 11-3. Brannklasser," *Direktoratet for byggkvalitet*. https://dibk.no/regelverk/byggteknisk-forskrift-tek17/11/i/11-3 (accessed May 14, 2023).

[2] GitHub Docs, "About continuous integration," *GitHub Docs*. https://docs.github.com/en/actions/automating-builds-and-tests/about-continuous-integration (accessed May 21, 2023).

[3] "What is CLI." https://www.w3schools.com/whatis/whatis_cli.asp (accessed May 21, 2023).

[4] "Faguttrykk." http://www.kbt.no/faguttrykk.asp?Uttrykk=fravik (accessed May 18, 2023).

[5] "ISO - Developing standards," *ISO*. https://www.iso.org/developing-standards.html (accessed Apr. 18, 2023).

[6] "What is an ORM (Object Relational Mapper)?," *Prisma's Data Guide*. https://www.prisma.io/dataguide/types/relational/what-is-an-orm (accessed May 21, 2023).

[7] "What Is A Relational Database (RDBMS)? | Google Cloud." https://cloud.google.com/learn/what-is-a-relational-database (accessed May 21, 2023).

[8] Direktoratet for byggkvalitet, "§ 11-2. Risikoklasser," *Direktoratet for byggkvalitet*. https://dibk.no/regelverk/byggteknisk-forskrift-tek17/11/i/11-2 (accessed May 14, 2023).

[9] Direktoratet for byggkvalitet, "Byggteknisk forskrift (TEK17) med veiledning," *Direktoratet for byggkvalitet*. https://dibk.no/regelverk/byggteknisk-forskrift-tek17 (accessed May 14, 2023).

[10] "Kvifor universell utforming av ikt? | Tilsynet for universell utforming av ikt." https://www.uutilsynet.no/veiledning/kvifor-universell-utforming-av-ikt/240 (accessed Apr. 19, 2023).

[11] "Kva seier forskrifta? | Tilsynet for universell utforming av ikt." https://www.uutilsynet.no/regelverk/kva-seier-forskrifta/153 (accessed Apr. 19, 2023).

[12] "What is Human-Computer Interaction (HCI)?," *The Interaction Design Foundation*. https://www.interaction-design.org/literature/topics/human-computer-interaction (accessed May 19, 2023).

[13] "What is User Experience (UX) Design?," *The Interaction Design Foundation*. https://www.interaction-design.org/literature/topics/ux-design (accessed May 20, 2023).

[14] A. S. for P. Affairs, "Wireframing," Sep. 06, 2013. https://www.usability.gov/how-to-and-tools/methods/wireframing.html (accessed Apr. 18, 2023).

[15] A. S. for P. Affairs, "Prototyping," Feb. 19, 2014. https://www.usability.gov/how-to-and-tools/methods/prototyping.html (accessed Apr. 18, 2023).

[16] "ISO standards - Usability Partners." https://www.usabilitypartners.se/about-usability/iso-standards.php (accessed Apr. 18, 2023).

[17] A. S. for P. Affairs, "Determining the Correct Number of Usability Test Participants," Oct. 08, 2013. https://www.usability.gov/get-involved/blog/2006/09/correct-number-of-test-participants.html (accessed Apr. 18, 2023).

[18] W. L. in R.-B. U. Experience, "Why You Only Need to Test with 5 Users," *Nielsen Norman Group*. https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/ (accessed Apr. 18, 2023).

[19] J. Spool, "Testing web sites: five users is nowhere near enough.," Mar. 2001, doi: 10.1145/634067.634236.

[20] O. L. Aiyegbusi, "Key methodological considerations for usability testing of electronic patient-reported outcome (ePRO) systems," *Qual Life Res*, vol. 29, no. 2, pp. 325–333, 2020, doi: 10.1007/s11136-019-02329-z.

[21] *Using usability-test participants multiple times (Kara Pernice)*, (Oct. 23, 2016). Accessed: May 18, 2023. [Online Video]. Available: https://www.youtube.com/watch?v=y5dArcUcw40

[22] A. S. for P. Affairs, "Running a Usability Test," May 15, 2014. https://www.usability.gov/how-to-and-tools/methods/running-usability-tests.html (accessed Apr. 19, 2023).

[23] "What is a Framework? Software Frameworks Definition," *freeCodeCamp.org*, Sep. 06, 2022. https://www.freecodecamp.org/news/what-is-a-framework-software-frameworks-definition/ (accessed May 20, 2023).

[24] A. Hunt and D. Thomas, *Pragmatic unit testing in Java with JUnit*. in Pragmatic starter kit series, no. 2. Raleigh, N.C: Pragmatic Bookshelf, 2003.

[25] "Client-server architecture | Definition, Characteristics, & Advantages | Britannica." https://www.britannica.com/technology/client-server-architecture (accessed Apr. 20, 2023).

[26] "What is a REST API? | IBM." https://www.ibm.com/topics/rest-apis (accessed Apr. 18, 2023).

[27] J. L. Harrington, *Relational Database Design and Implementation*. Morgan Kaufmann, 2016.

[28] "MVC - MDN Web Docs Glossary: Definitions of Web-related terms | MDN," Feb. 21, 2023. https://developer.mozilla.org/en-US/docs/Glossary/MVC (accessed May 04, 2023).

[29] "Software Engineering Principle — Coupling and Cohesion," *Educative: Interactive Courses for Software Developers*. https://www.educative.io/answers/software-engineering-principle-coupling-and-cohesion (accessed May 04, 2023).

[30] GeeksforGeeks, "Association, Composition and Aggregation in Java," *GeeksforGeeks*, Mar. 24, 2017. https://www.geeksforgeeks.org/association-composition-aggregation-java/ (accessed Feb. 28, 2023).

[31] "What is Virtual Networking?," *VMware*. https://www.vmware.com/topics/glossary/content/virtual-networking.html (accessed May 21, 2023).

[32] P. Abrahamsson, O. Salo, and J. Ronkainen, "Agile Software Development Methods: Review and Analysis," 2002.

[33] "What is Scrum?," *Scrum.org*. https://www.scrum.org/learning-series/what-is-scrum (accessed Apr. 19, 2023).

[34] M. O. Ahmad, J. Markkula, and M. Oivo, "Kanban in software development: A systematic literature review," in *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*, Sep. 2013, pp. 9–16. doi: 10.1109/SEAA.2013.28.

[35] D. I. K. Sjøberg, A. Johnsen, and J. Solberg, "Quantifying the Effect of Using Kanban versus Scrum: A Case Study," *IEEE Software*, vol. 29, no. 5, pp. 47–53, Sep. 2012, doi: 10.1109/MS.2012.110.

[36] S. Grønmo, "forskningsmetode - samfunnsvitenskap," *Store norske leksikon*. May 10, 2021. Accessed: May 14, 2023. [Online]. Available: https://snl.no/forskningsmetode_-_samfunnsvitenskap

[37] S. Grønmo, "kvalitativ metode," *Store norske leksikon*. Jan. 16, 2023. Accessed: May 14, 2023. [Online]. Available: https://snl.no/kvalitativ_metode

[38] "Vue vs React: What is The Best JS Technologies in 2023?" https://www.codica.com/blog/react-vs-vue/ (accessed May 22, 2023).

[39] "About," *Node.js*. https://nodejs.org/en/about (accessed Apr. 20, 2023).

[40] R. James, "Node.js vs. Spring Boot — Which Should You Choose?," *Medium*, Jan. 22, 2020. https://betterprogramming.pub/node-js-vs-spring-boot-which-should-you-choose-2366c2f76587 (accessed May 22, 2023).

[41] "What is MySQL?" https://www.oracle.com/mysql/what-is-mysql/ (accessed May 21, 2023).

[42] D. Miu, "Docx js." https://docx.js.org/#/ (accessed May 21, 2023).

[43] "Chart.js | Chart.js." https://www.chartjs.org/docs/latest/ (accessed May 21, 2023).

[44] "Sequelize v6 | Sequelize," May 11, 2023. https://sequelize.org/docs/v6/ (accessed May 21, 2023).

[45] "Paranoid | Sequelize," May 11, 2023. https://sequelize.org/docs/v6/core-concepts/paranoid/ (accessed May 21, 2023).

[46] "Transactions | Sequelize," May 11, 2023. https://sequelize.org/docs/v6/other-topics/transactions/ (accessed May 21, 2023).

[47] "Raw Queries | Sequelize," May 11, 2023. https://sequelize.org/docs/v6/core-concepts/raw-queries/ (accessed May 21, 2023).

[48] "Using Express middleware." https://expressjs.com/en/guide/using-middleware.html (accessed May 21, 2023).

[49] "Template Engines." https://expressjs.com/en/resources/template-engines.html (accessed May 21, 2023).

[50] "Security Best Practices for Express in Production." https://expressjs.com/en/advanced/best-practice-security.html (accessed May 21, 2023).

[51] "Getting Started | Axios Docs." https://axios-http.com/docs/intro (accessed May 21, 2023).

[52] "Minimal Example | Axios Docs." https://axios-http.com/docs/example (accessed May 21, 2023).

[53] "Interceptors | Axios Docs." https://axios-http.com/docs/interceptors (accessed May 21, 2023).

[54] Microsoft, "Overview - Azure App Service," Mar. 14, 2023. https://learn.microsoft.com/en-us/azure/app-service/overview (accessed May 14, 2023).

[55] Microsoft, "Overview - Azure Database for MySQL," Mar. 28, 2023. https://learn.microsoft.com/en-us/azure/mysql/single-server/overview (accessed May 14, 2023).

[56] Microsoft, "What is Azure Active Directory? - Microsoft Entra," Feb. 21, 2023. https://learn.microsoft.com/en-us/azure/active-directory/fundamentals/active-directory-whatis (accessed May 21, 2023).

[57] Microsoft, "Managed identities for Azure resources - Microsoft Entra," Jan. 27, 2023. https://learn.microsoft.com/en-us/azure/active-directory/managed-identities-azure-resources/overview (accessed May 21, 2023).

9 Vedlegg

Vedlegg 1 – Forprosjektplan

Vedlegg 2 – Prosessdokumentasjon

Vedlegg 3 – Visjonsdokument

Vedlegg 4 – Kravdokumentasjon

Vedlegg 5 – Systemdokumentasjon

Vedlegg 6 – Brukertestrapport wireframe

Vedlegg 7 – Brukertestrapport MVP

Vedlegg 8 – Fremdriftsplan

Vedlegg 9 – Fremdriftsplan faktisk varighet

Vedlegg 10 – Wireframe V1

Vedlegg 11 – Wireframe V3

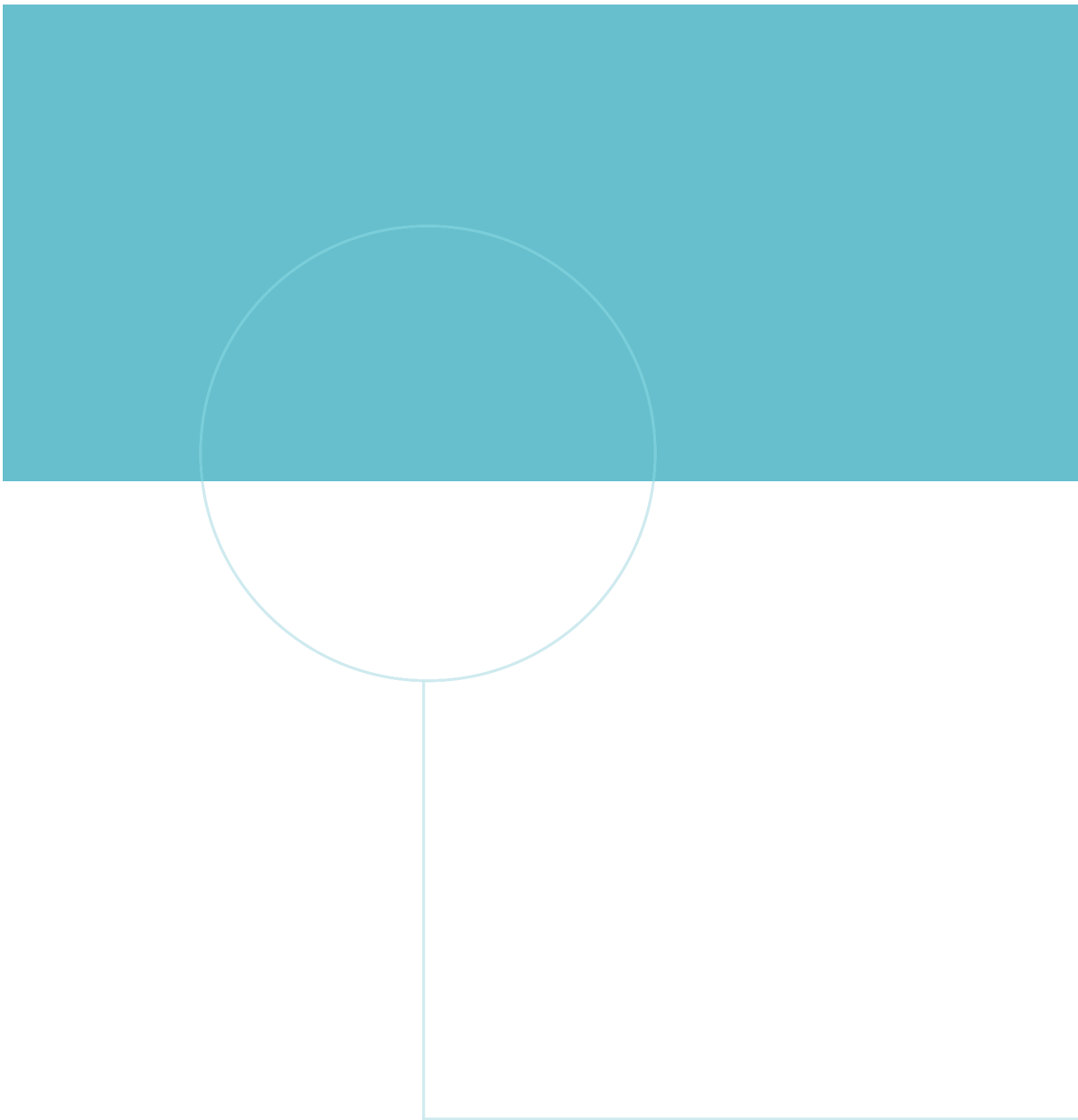
Vedlegg 12 – Adobe XD Prototype: https://xd.adobe.com/view/bacf3d02-4f94-4137-8fd5-5f48fb70c3b2-ef9c/?fullscreen

Vedlegg 13 – F-RAP-002 Fraviksdokumentasjon

Vedlegg 14 – Adresseliste

Vedlegg 15 – Sekvensdiagrammer

Vedlegg 16 – Domenemodell



NTNU

Kunnskap for en bedre verden