Universitetet i Stavanger

# DET TEKNISK-NATURVITENSKAPELIGE FAKULTET

# BACHELOROPPGAVE

| | |
|---|---|
| Studieprogram/spesialisering:<br>Bachelor I data, ingeniørfag | Vårsemesteret, 2021<br>Åpen / ~~Konfidensiell~~ |
| Forfatter:<br>Sveinung Fjermestad, Tor Håkon Carlsson, Oddvar Nordbø Øksendal | |
| Fagansvarlig: Erlend Tøssebro<br>Veileder(e): Erlend Tøssebro | |
| Tittel på bachelorpppgaven: Behaviour Mapper v2<br>Engelsk tittel: Behaviour Mapper v2 | |
| Studiepoeng: 20 Poeng | |
| Emneord:<br>Behaviour mapping<br>Webapplikasjon<br>Geographic Survey | Sidetall: 61 + https://github.com/Oddvar-N-O/BehaviourMapperV2Levering<br>Stavanger, 15.05.21 |


Behaviour Mapper v2
Tor Håkon Carlsson
Sveinung Fjermestad
Oddvar Nordbø Øksendal

# Sammendrag

Denne bacheloroppgaven hadde som mål å lage en webapplikasjon som skal bli brukt til atferdskartlegging og geografiske spørreundersøkelser. Oppgaven ble gitt av Universitetet i Stavanger. Applikasjonen skal kunne erstatte kartlegging på papir, og det var ønskelig at den var rask å bruke og at man skulle ha mulighet for å hente ut data digitalt etter endt kartlegging.

Gruppen stod fri til å velge hvilke teknologier som skulle bli brukt, men det var viktig at applikasjonen brukte teknologier slik at den enkelt kunne administreres og vedlikeholdes av andre i ettertid. Applikasjonen er lagd for å oppfylle disse ønskene.

Rapporten starter med å gå gjennom begrunnelse for valg av løsninger og utviklingsverktøy. I begrunnelsen vurderer vi de ulike alternativene etter hvor godt de oppfyller kravene nevnt over. Noen valg av utviklingsverktøy har større påvirkning på applikasjonen enn andre, fordi det vil være en stor jobb å bytte de ut.

Etter valg av utviklingsverktøy kommer en del av rapporten som beskriver hvordan applikasjonen er konstruert. Der kommer det frem hvilke løsninger som er tatt i bruk og noe om hvordan disse fungerer når applikasjonen er i bruk. Konstruksjonsdelen blir forklart for at leseren skal forstå hva som skjer på de forskjellige sidene, slik at det skal gå raskt å kunne jobbe videre med applikasjonen i ettertid.

Etter konstruksjonsdelen følger et diskusjonskapittel der det blant annet blir gått gjennom brukertester og tilbakemeldinger. Det diskuteres også løsninger som er valgt, og hvilke forbedringer og mangler det er i applikasjonen. Konklusjonen forteller til slutt at den ferdige applikasjonen har møtt de aller fleste av målene som var ønsket. Den ferdige applikasjonen erstatter atferdskartlegging og geografiske spørreundersøkelser på papir fullt ut. Den gjør det enklere for brukeren både under kartlegging, og i etterbehandling av data.

Applikasjonen er publisert og finnes på https://www.ux.uis.no/behaviourmapper/.

**Innholdsfortegnelse**

# Terminologiliste

- *Mapping* – kartleggingsiden i applikasjonen
- *Atferdskartlegging* – Å kartlegge hva et område brukes til. Et annet ord for Behaviour mapping. Forklares i kapittel 2.1.
- *Geografiske spørreundersøkelser* -Intervjue brukere om et utvalgt geografisk område. Forklares i kapittel 2.2.
- *Tilstand* – State i React
- *Hook* – En type React funksjon forklart i kapittel 7.3.4.
- *Shape-fil* – Fil som inneholder geografisk informasjon.
- *Lag (iht. Shape-fil)* – Informasjonen i shape-filer legges "oppå" andre filer i programmer som kan lese shape-filer. Det legges som et bilde, og blir lagt oppå et bakgrunnsbilde.
- *Hendelse* – Refererer til atferd som kartlegges
- *Ikon* – Refererer til bildene til en hendelse
- *API* – Programmeringsgrensesnitt mot backend i applikasjonen.
- *OIDC* – OpenID Connect: Protokoll i bruk for å håndtere data ved innlogging.
- *Applikasjonen* – Slik det er brukt i rapporten henviser til hele den ferdige webapplikasjonen tilgjengelig på: https://www.ux.uis.no/behaviourmapper/.

# 1. Introduksjon

## 1.1. Beskrivelse av oppgaven

Oppgaven var å lage en applikasjon som kunne brukes til atferdskartlegging og geografiske spørreundersøkelser.

Dette er den viktigste funksjonaliteten som var ønsket:

1. Applikasjonen skal kunne bli brukt til atferdskartlegging og geografiske spørreundersøkelser.
2. Brukerne skal kunne velge et kartutsnitt som enten lastes opp eller hentes fra en nettbasert løsning.
3. For atferdselkartlegging skal applikasjonen ha flere ikoner som brukes i kartleggingen. Brukeren skal kunne velge ikon og plassere dem på kartet. Det skal være forskjellige typer ikoner som bestemmer hvem personen er og hvilken handling personen utfører.
4. Brukerne skal kunne eksportere sine prosjekter til en shapefil. En shapefil er en fil som inneholder geografisk data i form av områder, linjer eller punkter.
5. Applikasjonen skal ha en sikker innloggingsfunksjon.
6. Applikasjonen skal være laget slik at kartleggingen og spørreundersøkelsene går så raskt som mulig.
7. Applikasjonen skal bruke teknologier som gjør at den er lett å vedlikeholde og videreutvikle for andre.

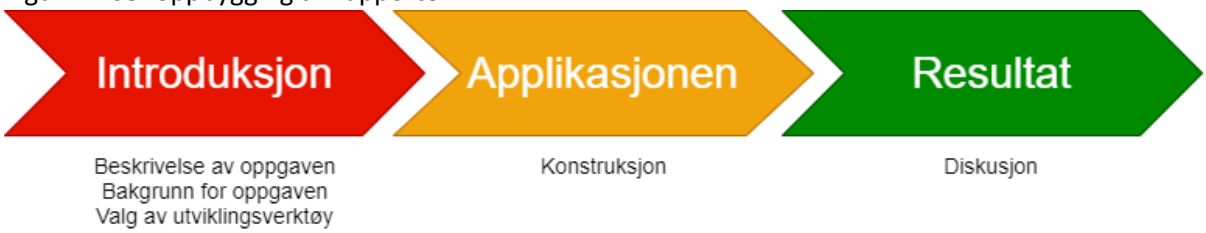
## 1.2. Grunnlag for oppgaven

Denne oppgaven ble foreslått av Erlend Tøssebro, Førsteamanuensis ved Institutt for data- og elektroteknologi på Universitetet i Stavanger (heretter UiS), på oppfordring av Daniela Müller-Eie, Førsteamanuensis ved Instituttet for industriell økonomi på UiS. Bakgrunnen for oppgaven er at det tidligere har eksistert to applikasjoner som av ulike grunner måtte erstattes. Den ene digitaliserte atferdskartlegging, og den andre digitaliserte geografiske spørreundersøkelser. Atferdskartleggingsprogrammet "*Behaviour Mapper – a tablet Application for behaviour mapping*" av Alexander Wiig Sørensen og Espen Stuvik Vier, var skrevet for en eldre versjon av iOS og sluttet å fungere på nyere versjoner av operativsystemet. For at applikasjonen skulle fungere måtte den oppdateres jevnlig av en med kunnskap til programmeringsspråket Swift og iOS-programmering. Det andre programmet, "*CityShare - webapplikasjon for kartlegging, forskning og analyse av geografiske områder*" av Mohammed Guniem, sluttet å virke da lisensen til Google Maps endret seg.

Vi hadde derfor som mål å lage en webapplikasjon som kunne brukes til atferdskartlegging og geografiske spørreundersøkelser, som er lett å bruke og vedlikeholde, og som ikke vil få problemer med at lisenser endres.

## 1.3. Oppbygging av rapporten

Figur 1 viser oppbygging av rapporten.



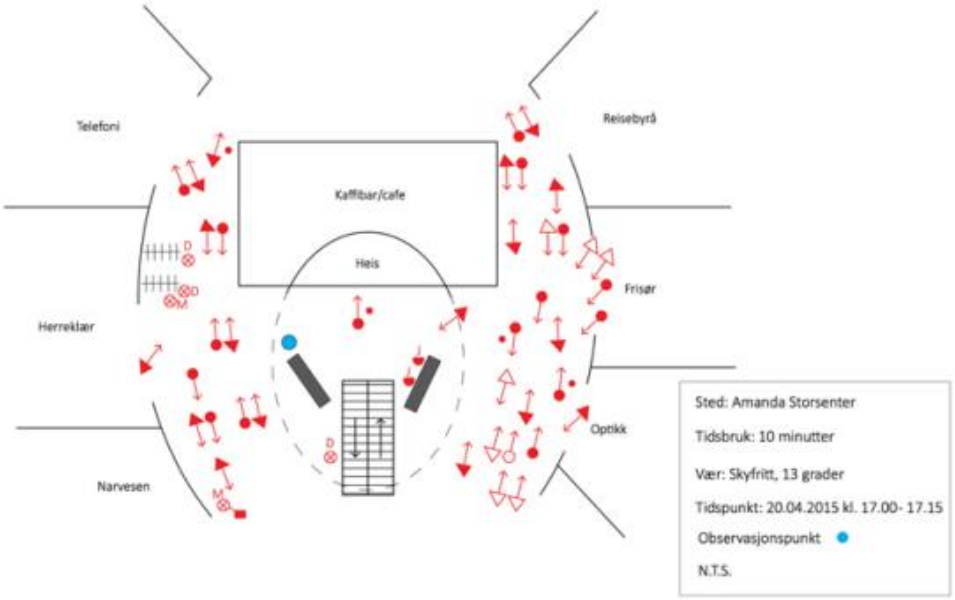

*Figur 1 Oppbyggingen av rapporten*

# 2. Bakgrunn

Dette kapitelet forklarer begreper og konsepter som vil brukes videre i oppgaven, slik at det kommer frem hva vi har forsøkt å gjøre.

## 2.1. Atferdskartlegging

Analytikere og byplanleggere har nytte av å vite hva ulike områder brukes til, og atferdskartlegging er et viktig redskap for å få kunnskaper om dette. Atferdskartlegging går ut på å velge et avgrenset område for en gitt periode. Den som kartlegger området registrerer hvilke typer mennesker som befinner seg der, og hva de gjør. Er det menn eller damer? Snakker de med noen, går de tur med hunden eller sykler de? Kartleggingen kan foregå både inne og ute, i mange forskjellige typer områder. Atferdskartlegging kan brukes til å finne ut hvordan et område blir brukt, og om det blir brukt slik det var tiltenkt.




*Figur 2: Et eksempel på atferdskartlegging [1]*

Over i Figur 2 ser vi et eksempel på atferdskartlegging. Ikonene vist i bildet over er mennesker, og pilene er hvilken vei de beveger seg. I atferdskartlegging kan man ha ulike ikoner. En kan for eksempel ha et ikon for forskjellige handlinger (sykle, gå, kjøre bil, sitte) og separere menn, kvinner og barn. I Figur 2 over kan for eksempel de helt fargefylte objekter være menn, mens de som bare har farget omriss er damer. Trekant kan bety gående, mens runding kan være syklende.

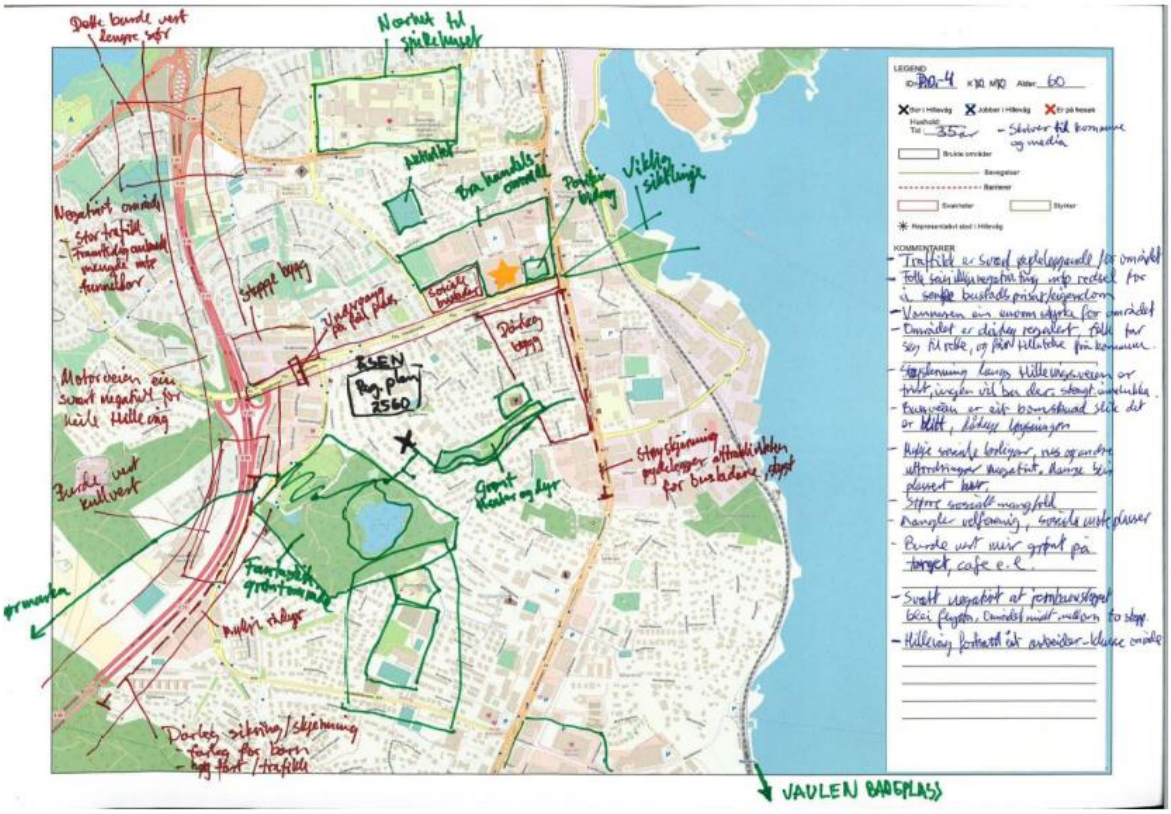
1 Madsen, Å., 2015. Kjøpesentres påvirkning på sentrum - En casestudie av Svortland sentrum og Hellvik Senteret (s 129) , Stavanger: Universitet i Stavanger.

De fleste ikonene som er brukt i webapplikasjonen vår er hentet fra "*Madsen, Å., 2015. Kjøpesentres påvirkning på sentrum - En casestudie av Svortland sentrum og Hellvik Senteret*" [2]. Dette er en masteroppgave i Byutvikling og Urban Design skrevet på UiS.

## 2.2. Geografiske Spørreundersøkelser

Et annet viktig redskap for analytikere og byplanleggere er geografiske spørreundersøkelser, der analytikere intervjuer mennesker innenfor gitte geografiske områder.

De stiller gjerne intervjuobjektet noen ferdig definerte spørsmål, i tillegg er det ofte et åpent spørsmål til slutt. Analytikerne tegner på kart under spørreundersøkelsen, for å knytte svarene opp mot områder, punkt, og bevegelsesmønstre (tilsvarende linjer på kartet). Man benytter seg ofte av fargekoder for å vise til intervjuobjektets følelser til det han eller hun svarer på. Fargekodene er da grønt for noe positivt og rødt for noe negativt.




*Figur 3 Et eksempel på en geografisk spørreundersøkelse på papir [3]*

## 2.3. Målet med Oppgaven

I dag blir størstedelen av atferdskartlegging og geografiske spørreundersøkelser gjort på papir. Målet med oppgaven er å lage en webapplikasjon som kan digitalisere dette arbeidet. Det er ønskelig at applikasjonen skal effektivisere kartleggingen og undersøkelsene, da det er tidkrevende å gjøre disse på papir. I tillegg blir kartlegginger som er gjort på papir fort uoversiktelige etter hvert som mer og

[2] Madsen, Å., 2015. Kjøpesentres påvirkning på sentrum - En casestudie av Svortland sentrum og Hellvik Senteret (s 18) , Stavanger: Universitet i Stavanger.

[3] Figur 3: Jonvik, Merete & Lindland, Kristiane & Tvedt, Helge L. & Müller-Eie, Daniela & Melberg, Kjersti (2018) Hillevåg – En sosiokulturell stedsanalyse. IRIS. Rapport under produksjon.

mer data blir registrert, og alle registrerte data må overføres manuelt om en ønsker å lagre data digitalt. Applikasjonen skal derfor ha mulighet for å overføre data til forskjellige filformat. Applikasjonen skal være intuitiv, rask og enkel å bruke. Den skal også være lett å vedlikeholde, og den skal bruke åpne lisenser så langt det lar seg gjøre. Applikasjonen skal hovedsakelig bli lagd for nettbrett, men vil også være tilgjengelig for andre enheter.

# 3. Valg av utviklingsverktøy

I dette kapittelet gjøres det rede for hvilke utviklingsverktøy, programmeringsspråk og teknologier som er brukt for å løse oppgaven, og hvorfor de har blitt valgt. Det forklares hvordan disse hjelper å oppnå målene til oppgaven henvist til i kapittel 2.3 Målet med Oppgaven.

Kort oppsummert brukes JavaScript-rammeverket React i frontend, og CSS for styling. En kort innføring i React er gitt i appendikset i kapittel 7.3. Backend er bygget i Python, og benytter seg av rammeverket Flask. Databasen er bygget med SQLite. Begrunnelsene for disse valgene er gitt i avsnittene under.

## 3.1. Arbeidsplattform: GitHub og Travis

GitHub ble brukt som arbeidsplattform for versjonskontroll av applikasjonen. Ettersom GitHub er kjent fra undervisningen ved UiS ble dette et naturlig valg. GitHub er enkelt i bruk og det har en stor brukerbase; noe som gjør det enkelt å finne løsninger til eventuelle problemer.

Det sterkest vurderte alternativet til GitHub var GitLab, som er veldig likt GitHub. En fordel med GitLab er at de har egne innebygde testrammeverk. GitHub har også fått dette i nyere tid, men de er enn så lenge mindre utviklet. At GitHub undervises i på UiS, og at de som vil bygge videre på applikasjonen da kan antas å ha erfaring med det, gir GitHub en klar fordel ovenfor GitLab. For å få til automatisk testing av applikasjonen har testrammeverket Travis blitt integrert i applikasjonen. Det ble sammenlignet med GitHubs innebygde testrammeverk, men Travis ble valgt fordi det er mer utviklet og har bedre dokumentasjon.

Det er enkelt å bytte arbeidsplattform, og dermed er ikke dette et fastsatt valg for de som skal videreutvikle og vedlikeholde applikasjonen. Dette står i kontrast med valget av kodespråk, som det ville blitt mye arbeid å forandre på.

## 3.2. Frontend: JavaScript og React

Det ble tidlig bestemt å skrive frontend i JavaScript, og det var to hovedgrunner til dette. Den første er at JavaScript er det mest brukte frontend-språket, som kjøres i hele 95% [4] av alle websider. Den andre er at Javascript kan kjøre direkte i en nettleser, og er kompatibelt med alle moderne nettlesere [5]. Javascript har også mange rammeverk som gjør det lettere og raskere å lage applikasjoner.

For valg av rammeverk stod valget mellom React og Vue. Både Vue [6] og React er under åpen lisens, selv om React [7] har en noe unik lisens.

### 3.2.1. Valg av rammeverk: React versus Vue

React har følgende fordeler:

- React er et av de raskeste rammeverkene som eksisterer, noe som vil være nyttig dersom det er mye informasjon som skal lastes ned.

---

[4] Vinugayathri. 2020, 5 JavaScript Alternatives for Front End Development, https://www.clariontech.com/blog/5-javascript-alternatives-for-front-end-development, Lastet ned 16.04.2021

[5] @Fyrd and @Lensco, 2021 , CANIUSE, https://caniuse.com/?search=JavaScript , lastet ned 26.04.21

[6] Evan You. "*The Progressive JavaScript Framework*", 2014. https://vuejs.org/ Lastet ned den 28.04.2021.

[7] Petersen, John V., "*Legal Notes: What's the Deal with ReactJS's Licensing Scheme?*". Publisert i *CODE Magazine* Januar/Februar 2017, Sist oppdatert: February 19, 2019 https://www.codemag.com/article/1701041/ Lastet ned den 28.04.2021

- React er utviklet for å lage responsive applikasjoner, det vil si at innhold vises dynamisk basert på data. Dette gjøres i React ved bruk av tilstandsobjekter. Hvis et tilstandsobjekt forandrer seg vil dette reflekteres på skjermen uten at hele siden må lastes opp på nytt.
- React har gjenbrukbare komponenter, noe som er nyttig for applikasjonen. Vi har brukt dette blant annet for å lage ikonbiblioteket, der hvert ikon er en egen komponent. Hver gang en ikonkomponent lages sendes det inn nye data via props, slik at alle ikonene er unike.
- Komponenter samler både JSX og JavaScript, dette gjør koden mer oversiktlig da kode som hører sammen ligger i en og samme fil.

React har følgende svakheter [8]:

- Det var nytt for to av oss, og tok derfor tid å lære.
- Man må lære seg å bruke JSX i tillegg til å forstå React.
- React har to ulike måter å skrive komponenter på, noe som gjør at en har to forskjellige måter å gjøre de samme tingene på, som kan være forvirrende.
- På grunn av den kjappe utviklingen av React har det ikke like god dokumentasjon [9] som Vue [10].

Vue har følgende fordeler:

- Gruppen har hatt undervisning i Vue ved UiS.
- Det er antatt at de som skal videreutvikkle applikasjonen også vil ha det.
- Feilsøking i Vue er enkelt, da feilsøkingsprosessen kjører parallelt med koden mens koden blir skrevet.
- Vue er enklere å lære enn React.
- Vue har også gjenbrukbare komponenter.
- Vue er enkelt å integrere med andre funksjonsbiblioteker.

Vue har også en del svakheter:

- Det har problemer med å kjøre i Safari.
- Det er færre utvidelser til Vue.
- Vue har en mindre brukerbase fordi rammeverket er nyere, og fordi det ikke ble laget av en teknologigigant som kan spre det fort.

React ble valgt som rammeverket for applikasjonen. Hovedgrunnen til at React ble valgt var at den har en større brukerbase som gjør det lettere å finne utvidelser og svar på problemstillinger. Vues problemer med Safari var også med å påvirke avgjørelsen da et av målene var at applikasjonen skulle kunne brukes i alle nettlesere.

[8] JavaTpoint, 2011-2018, Pros and Cons of ReactJS, https://www.javatpoint.com/pros-and-cons-of-react, 26.04.21
[9] Pawar, Gauri, 2020, The Good, the Bad, & the Not So Bad About React.js Development https://eluminoustechnologies.com/blog/pros-and-cons-of-react-js-development/, 23.04.21
[10] Nowak, Maja, 2020, Vue vs React in 2021: Which Framework to Choose and When, https://www.monterail.com/blog/vue-vs-react-2021, 23.04.21

## 3.3. Backend: Python og Flask

I backend sto valget mellom å bruke Python, Java og Go. Alle språkene undervises i ved UiS. Hvert av språkene kan kjøre på alle tilgjengelige plattformer (Mac, Linux, Windows) og har åpen kildekode. Følgende egenskaper var grunnlaget for vurdering:

**Python:** Python er et objektorientert tolket språk. Python er både et av de raskeste programmeringsspråkene å programmere i og en av de enkleste språkene å forstå, da språket er skapt for å være enkelt. Det er derfor et bra valg for mindre erfarne utviklere.

**Java:** Java er et objektorientert kompilert språk, og har en stor brukerbase med mange rammeverk. Java bruker JVM for å tolke kompilert kode og var tidligere det mest brukte programmeringsspråket i verden. I de siste årene har Python tatt over tronen, og blir nå brukt mer enn Java [11].

**Go:** Go er et funksjonsbasert kompilert språk. Det har et sterkt fokus på parallell-programmering, og er også enkelt å lese da det for eksempel har veldig få innebygde nøkkelord. Nøkkelord er reserverte ord som ikke kan brukes fritt, som *true*, *int* og *function*. I motsetning til Java kompilerer Go koden uten å tolke det, ved å skrive all koden om til binærform og så kjøre den.

### 3.3.1. Valg av programmeringsspråk i backend

Kort oppsummert skal applikasjonen være lett å bruke, lett å vedlikeholde, bruke åpne lisenser så langt det lar seg gjøre, og være rask i registreringen av nye personer. Brukervennlighet har mest med frontend å gjøre, derfor ble det sett bort fra i denne valgprosessen. Både Python, Java og Go har åpne lisenser.

Det ble sterkt vurdert å bruke Go, ettersom Go er kjappere enn Java [12] og Python [13]. I tillegg er syntaksen til Go enklere enn Java, og det er god støtte for samtidig kjøring av kode. Go har imidlertid ingen feilhåndtering, noe som gjør det vanskeligere å jobbe med, og det er heller ikke så mange utvidelsespakker eller rammeverk i Go. På bakgrunn av dette ble ikke Go valgt.

Både Python og Java har mange utvidelsespakker. De har også store brukerbaser, noe som vil gjøre det enklere å vedlikeholde koden for dem som skal se på prosjektet i ettertid. Java har mer funksjonalitet, mens Python er enklere å forstå. Python kjører saktere enn Java, og bruker mer minne. Til tross for mindre funksjonalitet har Python all den funksjonaliteten som er nødvendig for applikasjonen vår, da applikasjonens backend i hovedsak skal skrive til databasen, og vil fungere som et mellomledd mellom databasen og frontend. I og med at det er en mindre applikasjon antas det at kjøretiden ikke vil bli et problem. Et av gruppemedlemmene som skrev oppgaven hadde ikke brukt Java tidligere, mens Python hadde alle god kjennskap til. Da det var viktigere å jobbe med noe som var enkelt å bruke og forstå, enn å ha utvidet funksjonalitet, ble Python valgt.

Som nevnt over måtte språket som valgtes være "*lett å vedlikeholde og raskt å sette seg inn i for nye personer som skal vedlikeholde det*". Den store brukerbasen til Python og den enkle syntaksen gjør

---

11 Johnson, Jonathan, 2020, Java vs Go: What's The Difference?, https://www.bmc.com/blogs/go-vs-java/, 21.04.21

12 crowd sourced, 2021, The Computer Language Benchmarks Game, https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/go.html, 21.04.21

13 crowd sourced, 2021, The Computer Language Benchmarks Game, https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/go-python3.html, 21.04.21

det enkelt å vedlikeholde. Python har også mange pakker og utvidelser som blant annet gjør det enkelt å eksportere informasjonen fra databasen til shape-filer. I tillegg er det pakker som lager kryptotilfeldige tall for å øke sikkerheten i applikasjonen, noe som det ikke er anbefalt å lage selv. [14]

### 3.3.2. Valg av rammeverk

Python tilbyr flere typer rammeverk for å tjene web API-er. Et API, også kalt et programmeringsgrensesnitt, er en kobling som gjør at spesifikke deler av en kode i en programvare kan aktiveres fra en annen programvare [15].

Det ble sett nærmere på Flask, FastAPI og Hug. Flask har tre store fordeler for de som skal videreutvikle applikasjonen: det blir brukt i to programmeringsfag på UiS, det har en stor brukerbase, og det er enkelt å forstå. Fordi dette er store styrker, ble de andre to rammeverkene satt opp mot Flask.

**Flask versus FastAPI:** FastAPI er kjappere enn Flask, og det er enklere å teste i. FastAPI støtter asynkron kode, noe som er en klar fordel dersom tjenerdelen vil få mange forespørsler. Det ble imidlertid antatt att denne applikasjonen ikke ville bli brukt av mange nok til at dette ville være en stor fordel. FastAPI har en veldig rik dokumentasjon, men dette er også tilfellet med Flask. FastAPI kan fort få en veldig stor "main.py" fil fordi alt er knyttet opp mot appen, men dette er også en svakhet i Flask.

Alt i alt ble det bestemt at styrkene til FastAPI ikke gjør opp for Flasks fordeler, og FastAPI ble derfor ikke valgt over Flask.

**Flask versus Hug:** Hug er blant de raskere rammeverkene til Python, og det raskeste for Python 3 [16]. API-en og logikken til applikasjonen er klart adskilt, slik at Hug er et sikkert rammeverk. Hug har ingen innebygd databasebehandlingen, slik at man må kombinere det med et funksjonsbibliotek som SQLAlchemy. Hug kan også ha problemer med å håndtere forespørsler fra klientdelen. Det virket usannsynlig at dette prosjektet kom til å bruke pre-prosessering av forespørsler, men det virket ikke sikkert når valget av rammeverk foregikk.

Hugs imponerende fart var irrelevant av samme grunn som FastAPIs hurtighet. Sikkerheten gjorde et positivt inntrykk, men manglende innebygd databasebehandling var et minus. Uten andre reelle fortrinn enn farten ble Hug også lagt til siden.

Oppsummert ble Python og Flask valgt av følgende grunner:

- Begge er enkle å forstå, noe som gjør dem lettere å jobbe med og vedlikeholde.
- Både Python og Flask undervises ved UiS.
- Python har mange pakker og utvidelser.
- Begge har en stor og aktiv brukerbase, noe som gjør det enklere å vedlikeholde.

[14] Cox, Joseph, 2015, Why You Don't Roll Your Own Crypto, https://www.vice.com/en/article/wnx8nq/why-you-dont-roll-your-own-crypto, 07.05.21
[15] Rossen, Eirik, 2019, API i Store norske leksikon på snl.no, https://snl.no/API, 26.04.21
[16] Crosley, Timothy Edmund, 2021, Embrace the APIs of the future, https://www.hug.rest/, 14.04.21

## 3.4. Databasen

### 3.4.1. MySQL sammenlignet med SQLite

I valget av Databasesystem sto det mellom MySQL og SQLite. Begge har åpen kildekode. I forhold til MySQL er SQLite enklere på alle vis; det er enklere å sette opp og enklere å konfigurere, men er også enklere i negativ forstand da det har færre muligheter. SQLite er i det Offentlige Domene, mens MySQL er eid av Oracle. MySQL støtter trettifire datatyper [17], mens SQLite bare støtter fem [18]. SQLite har ingen datatype satt av for dato, men har "Date And Time" funksjoner som kan lagre data som TEXT, REAL og INTEGER verdier, slik at begge databasesystemene vil være tilstrekkelige for våre behov.

Fordelene til MySQL er: "*sikker pengeoverføring, skalerbarhet, høy tilgjengelighet, pålitelighet, enkelt å laste ned*". Dens svakheter er "*lang utviklingstid, replikasjonskonflikter, loggkostnader, spørringsproblemer, konneksjonstap under høyt brukerpress*" [19]. MySQL fungerer godt med datamengder i alle størrelser mens SQLite ikke fungerer så bra med store datamengder.

"*Store datamengder*" er imidlertid et relativt begrep. SQLite kan håndtere "*mindre enn 100 000 treff per dag*" [20] og vedlikeholde en database på 281 terabytes [21]. MySQL kan håndtere "*mellom 2 000 og 50 000 spørringer per sekund, beroende på serveren*" og har et maksimum databasestørrelse på 65 536 terabytes [22]. SQLite gir brukerne ett nivå av tilgang til databasen, mens MySQL har flere tilgangsnivå. Brukerne av databasen til applikasjonen trenger bare ett nivå.

SQLite skiller seg ut fra andre databasesystem i at databasen er en del av applikasjonen, og dermed ikke trenger en egen server for å kjøre. Det gjør databasen atskillig lettere å sette opp og konfigurere. Det gjør også at applikasjonen er mer portabel.

MySQL blir ofte brukt i større applikasjoner som har høyere krav til konfigurering og sikkerhet, mens SQLite, som er den mest distribuerte databasetypen i verden [23], ofte blir brukt til mindre applikasjoner som ikke har behov for å dele databasetilgang inn i ulike nivå.

### 3.4.2. Valg av Databasesystem

MySQL ble brukt i starten av prosjektet, men etter rådføring med veileder og etter å ha lest oss mer opp byttet vi til SQLite. SQLite er ikke like bra som MySQL til å håndtere store datamengder, men applikasjonen vår skal sannsynligvis ikke håndtere store mengder av verken data eller brukere. SQLite trenger ikke en egen databasetjener, noe som gjør den enklere å sette opp. Det at SQLite bare gir ett tilgangsnivå til databasen ble ikke en begrensing, fordi applikasjonen bare trenger ett tilgangsnivå. Det at SQLite er i det Offentlige Domene er også en fordel, fordi det ikke eies av en bedrift som plutselig kan forandre lisensen slik som skjedde med Google Maps høsten 2018. SQLite ble derfor det beste valget for oss.

---

[17] Refsnes Data, 1999-2021, SQL Data Types for MySQL, SQL Server, and MS Access , https://www.w3schools.com/sql/sql_datatypes.asp, 20.04.21

[18] SQLite.org, 2021, Datatypes In SQLite Version 3, https://www.sqlite.org/datatype3.html, 20.04.21

[19] Branson, Tony, 2017, THE 5 BEST REASONS TO CHOOSE MYSQL – AND ITS 5 BIGGEST CHALLENGES, https://dataconomy.com/2017/04/5-reasons-challenges-mysql/, 20.04.21

[20] SQLite.org, 2021, Appropriate Uses For SQLite, sqlite.org/whentouse.html, 20.04.21

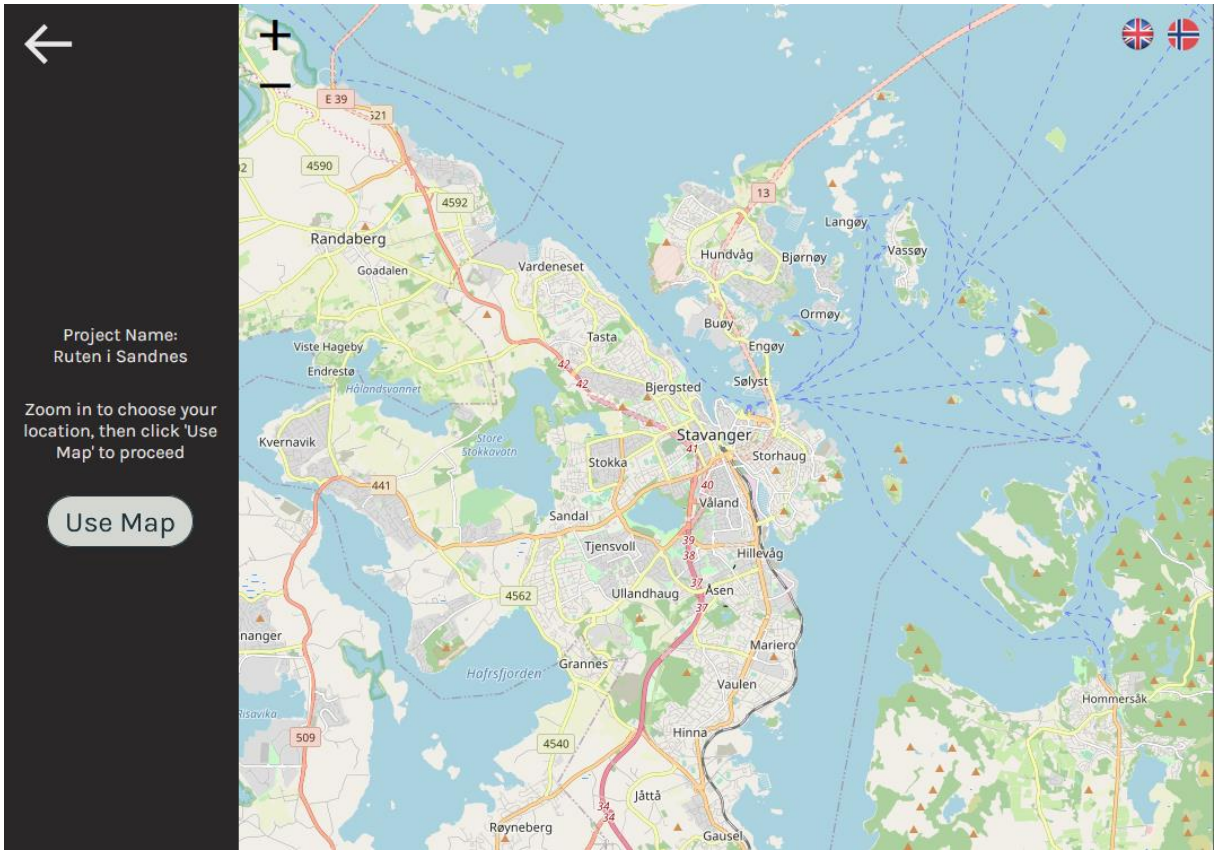
[21] SQLite.org, 2021, Limits In SQLite, https://www.sqlite.org/limits.html, 20.04.21

[22] Oracle Corporation, 2021, 12.4 Limits on Table Size, https://dev.mysql.com/doc/mysql-reslimits-excerpt/8.0/en/table-size-limit.html, 20.04.21

[23] SQLite.org, 2021, What Is SQLite?, https://www.sqlite.org/index.html, 20.04.21

## 3.5. Verdenskartet: OpenStreetMap og OpenLayers

Brukerne skal kunne benytte seg av et dynamisk verdenskart for å velge ut det kartutsnittet de vil ha. Kartsystemet som brukes skal kunne brukes over lengre tid og være lett å vedlikeholde.




*Figur 4 Verdenskartet og bildet der en kan velge kartutsnitt i applikasjonen.*

Et naturlig valg av kartsystem kunne vært Google Maps, men Google endret for noen år tilbake lisensen sin slik at det ikke kan brukes til dette formålet uten at de som eier applikasjonen påtar seg kostnader. Derfor var det et ønske om å bruke et system som ikke krevde lisens, og valget falt da på OpenStreetMap. OpenStreetMap er et "*gratis, redigerbar kart over hele verden som blir bygd av frivillige, i stor grad fra bunnen av, og publisert med en fri-bruk lisens.*" [24]. Det regnes med at OpenStreetMap vil fortsette å være et fri-bruk lisens program i det offentlige domene så lenge den blir vedlikeholdt av frivillige, likt Wikipedia, og at det derfor er en trygg kilde til kartbilder også i fremtiden.

Det skal også være mulig å la brukerne laste opp egne kart, slik at applikasjonen vil være brukbar selv om OpenStreetMap skulle endre lisens, eller dersom brukeren vil kartlegge et område som ikke finnes i et verdenskart, som innsiden av et kjøpesenter.

### 3.5.1. Valg av kartbehandler: OpenLayers vs Leaflet

Det ble sett på to funksjonsbiblioteker for å integrere OpenStreetMap i JavaScript. Disse var OpenLayers og Leaflet.

---

24 Creative Commons Attribution-ShareAlike 2.0 license, 2021, About OpenStreetMap, https://wiki.openstreetmap.org/wiki/About_OpenStreetMap, 21.04.21

- Leaflet er et enkelt funksjonsbibliotek som er brukervennlig og greit å forstå. Det er en av de ledende bibliotekene for kartbehandling.
- OpenLayers er et større bibliotek, men er også mer komplisert å forstå. OpenLayers blir i hovedsak brukt i forbindelse med "Geografisk Informasjons Systemer" (GIS).
- Begge disse programvarene er åpen kildekode.

Det ble gjort forsøk på å få et fungerende kart inn i React-koden med begge disse funksjonsbibliotekene, men det ble bestemt å bruke OpenLayers etter en grundigere vurdering. Dette skyldtes ikke bare at OpenLayers er enklere å integrere med GIS programmer, men også at den økte funksjonaliteten gjør OpenLayers-kart kompatible med GeoJSON og Canvas2D. Canvas2D gjør det enkelt å lage ett binærspråk-objekt, og deretter et ".png" bilde, av det relevante kartutsnittet. GeoJSON gir oss en alternativ måte å overføre geografisk informasjon, men ble valgt bort til fordel for CSV, da CSV-filer var enklere å lage.

```
this.map = new Map({
  target: null,
  layers: [new TileLayer({source: new OSM()})],
  view: new View({center: transform([5.733107, 58.969975],
    'EPSG:4326', 'EPSG:3857'), zoom: 12}),
  controls: [new Zoom()]
});
```

*Figur 5: Kartobjektet i Javascript.*

## 3.6. Innlogging

I innloggingsdelen sto valget mellom å bruke Feide eller å lage et egenlaget innloggingssystem. Feide er "*den nasjonale fellesløsningen for trygg innlogging og datadeling i utdanningssektoren.*" [25]. UiS er allerede tjenestetilbyder for Feide, noe som gjør det enklere å implementere fordi programutviklere knyttet opp mot UiS vil kunne få tilgang til tjenesten uten større anstrengelser.

Fordelene med et egenlaget innloggingssystem er at man kan definere hvilken informasjon brukerne skal sende inn, mens man i et eksisterende system må begrense seg til informasjonen som systemet lagrer. Fordelen med å bruke et eksisterende innloggingssystem er at innloggingsfunksjonaliteten allerede er innebygd og inneholder sikkerhetstiltak for å beskytte brukerne. Brukernes passord og privatinformasjon er sikret av tjenestetilbyderen, og utviklerne får da mer tid til å utvikle applikasjonens funksjonalitet.

I datasikkerhetsfaget på UiS læres det at det er anbefalt å bruke en tredjepart når det kommer til sikkerhetsfunksjonalitet ettersom det vil bli jevnlig oppdatert og at det generelt sett er tryggere enn en å lage sikkerheten selv. Dette vil også gjøre applikasjonene enklere å vedlikeholde. Ettersom ingen i gruppen har noen ekspertise innen det å lage innloggingssystemer, og et av kravene var å ha en sikker innloggingsfunksjon, ble Feide valgt til innlogging. Den brukerinformasjonen som trengs til applikasjonen, blir allerede lagret i Feide. De har også mer informasjon om brukeren som kan brukes dersom dette blir ønskelig under fremtidig utvikling. Informasjonen Feide har tilgjengelig og som kan brukes i videreutvikling av applikasjonen er det mulig å få tilgang til via dataporten [49] Valg av innloggingsprotokoller for Feide: OpenID og SAML 2.0

Når så Feide ble valgt å bruke til innlogging måtte det foretas et nytt valg, fordi Feide støtter to protokoller for å håndtere innloggingen. De to protokollene er SAML 2.0 og OpenID Connect. Begge

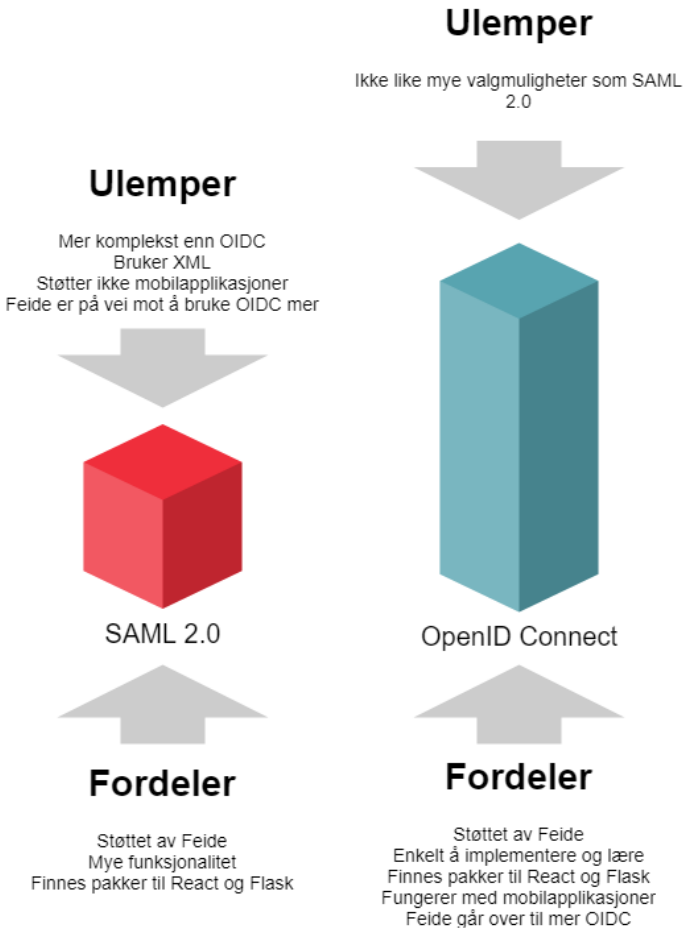
25 Uninett, 2021, Fordeler med Feide – trygg og enkel innlogging og datadeling , https://www.feide.no/fordeler-med-feide, 17.04.21

er godt etablert og i bruk i mange steder. SAML 2.0 er den eldste av disse og bruker XML for å håndtere forespørsler. OpenID Connect heretter referert til som OIDC er nyere og bruker JSON tokens istedenfor XML. Feide lister opp tre faktorer for å bestemme hvilken protokoll som skal brukes, disse er som følger:

- "Eksisterende støtte i applikasjoner/rammeverk: Noen applikasjoner og webrammeverk har innebygget støtte for en av protokollene. I det tilfellet er det som regel enklest å bruke den protokollen som allerede er støttet.
- Støtte for innlogging til mobilapplikasjoner: OIDC protokollen er enklere å bruke når en logger inn på mobilapplikasjoner. Å bruke denne protokollen lar mobilapplikasjonen direkte få tak i en OAuth 2.0 access token, som brukes for å få tilgang til APIer.
- Kompleksiteten til protokollen: OIDC protokollen er generelt sett enklere enn SAML 2.0 protokollen. " [26].

Spesielt de to siste punktene er relevante for oss. I en mer dyptgående artikkel av Petteri Stenius hos Ubisecure fra 2020 [27] konkluderer han med at valget mellom disse protokollene i hovedsak vil være en subjektiv beslutning, men med noen markante unntak der man bør velgde OIDC. Disse er: appen er en mobilapplikasjon, den skal brukes på lærestadiet, apputviklerne kan ikke XML og vil ikke bruke tid på å lære det.

[26] Uninett, 2021, Adding Feide login to a service, https://docs.feide.no/service_providers/getting_started/add_feide_login.html, 27.04.21
[27] Stenius, Petteri, 2020, The differences between OpenID Connect (OIDC), OAuth 2.0 & SAML, https://www.ubisecure.com/education/differences-between-saml-oauth-openid-connect/, 27.04.21




*Figur 6 Fordeler og ulemper SAML 2.0 vs OIDC*

Som illustrert i Figur 6 er det flere fordeler og færre ulemper ved bruk av OIDC enn ved å bruke SAML 2.0. OIDC har også noen fordeler som er svært passende for applikasjonen, nemlig det at den ikke krever XML kunnskaper, fungerer på mobilapplikasjoner og er mindre kompleks en SAML 2.0. Fordi XML ikke undervises på UiS ble OIDC derfor det åpenbare valget. I tillegg går Feide også gradvis over til OIDC [26].

## 3.7. Distribuering

Når det kom til å gjøre applikasjonen vår tilgjengelig for brukere fantes det flere alternativer; Google Cloud web hosting, Amazon Web Services, Bluehost og Dreamhost er alle store navn innen distribuering av websider. I tillegg hadde vi tilgang til Unixmiljøet på UiS sine servere.

Google[28] og Amazon [29] er gratis en liten periode, men de egner seg bedre for utvikling enn produksjon, fordi de etter hvert vil kreve betaling for å hoste siden. Bluehost og Dreamhost koster begge ca. 25 kroner i måneden. Unixmiljøet på UiS er gratis å bruke, og ved å hoste det her vil

[28] Google, 2021, Google Cloud pricing, https://cloud.google.com/pricing, 18.04.21

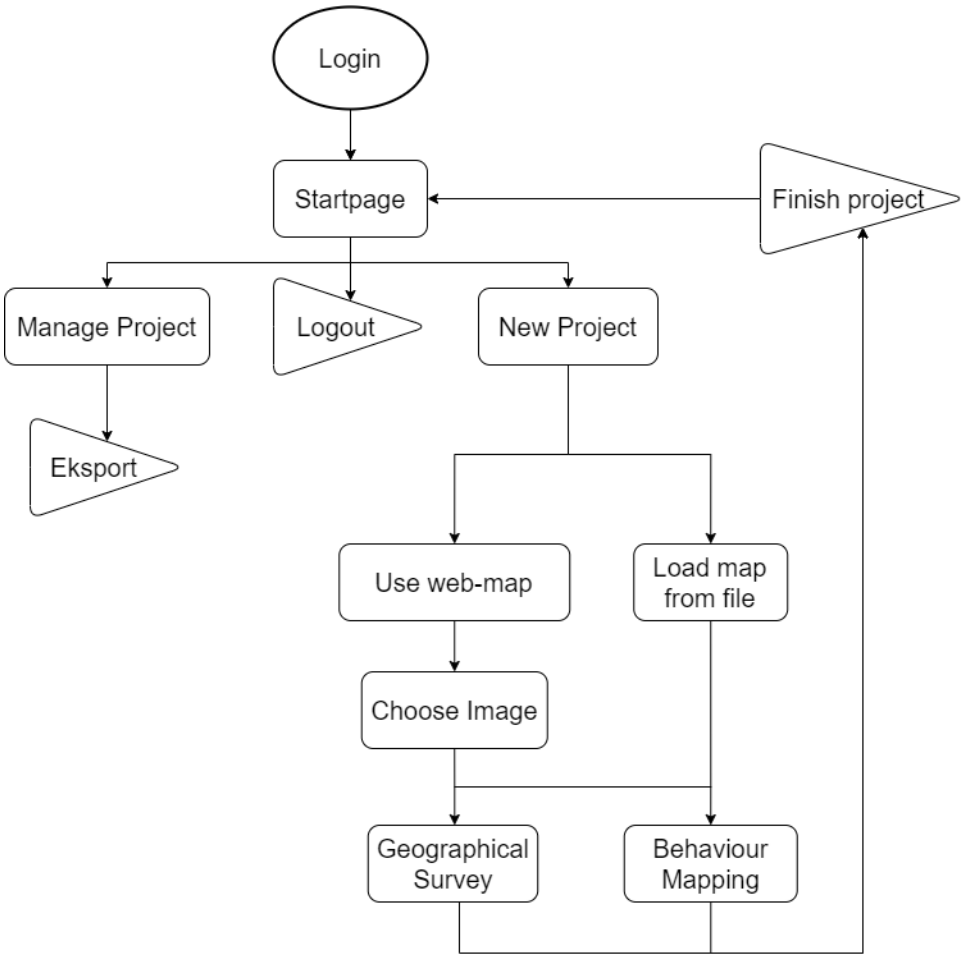
[29] Amazon Web Services, 2021, AWS Free Tier, https://aws.amazon.com/free/?all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free%20Tier%20Types=*all&awsf.Free%20Tier%20Categories=*all, 18.04.21

administratorer knyttet til UiS kunne bli gitt full tilgang til distribusjonen av applikasjonen. I tillegg vil applikasjonen komme inn under domenet til UiS, slik at det ikke vil bli behov for å registrere et nytt domene. Sikkerhetsmessig er det også en fordel med å bruke UiS sitt miljø, da serveren deres allerede har sertifikat for HTTPS. Dersom en av de andre tilbyderne ble valgt kunne det blitt mer tidkrevende å ordne.

Å få applikasjonen under domenet til UiS hadde ikke vært mulig dersom en av de andre valgmulighetene ble benyttet. Valget om å ha det på UiS vil også være en fordel for fremtidig vedlikehold og videreutvikling av applikasjonen, fordi hele utviklingsprosessen videre ligger lokalt på UiS. Kompetansen på å drifte Apache-tjeneren for applikasjonen ligger også lokalt hos UiS. Det ble derfor bestemt at applikasjonen skulle bli distribuert fra Unixmiljøet på UiS.

# 4. Konstruksjon

Her beskrives den endelige konstruksjonen av applikasjonen. I Figur 7 vises alle veier en bruker kan ta, og hvordan applikasjonen ser ut fra brukersiden.




*Figur 7 Oppbygging av applikasjonen*

## 4.1. Oppbygging av kodebasen

Koden er delt inn i én mappe for frontend, og én mappe for backend. I backend ligger databasen og all logikk tilhørende backend. Det er én mappe for selve koden og én for statiske filer. Mappen med de statiske filene ligger sammen med selve koden. frontend ligger alle JavaScript- og CSS-filer, samt eksterne moduler applikasjonen er avhengig av. Det er en mappe for alle komponenter tilhørende en side, og en mappe for alle websider til applikasjonen.

## 4.2. Frontend

### 4.2.1. App.js - applikasjonens hjerte

App.js er applikasjonens hjerte. Den importerer alle brukte komponenter, og samler dem til én enkelt App-komponent. Denne filen inneholder en router som gjør det mulig å vise forskjellige sider når en

går til en ny URL. Applikasjonen har en funksjon som gjør det mulig å bytte språk, og denne funksjonaliteten er tilgjengelig på alle sidene i applikasjonen.

```
function App() {
  return (
    <AuthContext>
      <Suspense fallback="loading">
        <div className="App">
          <Router basename="/behaviourmapper">
            <LanguageSelector></LanguageSelector>
            <Switch>
              <Route path='/' exact component={Login} />
              <Route path='/startpage' component={Startpage} />
              <Route path='/newproject' component={NewProject} />
              <Route path='/chooseimage' component={ChooseImage} />
              <Route path='/mapping' component={BehaviourMapping} />
              <Route path='/manageProject' component={ManageProject} />
            </Switch>
          </Router>
        </div>
      </Suspense>
    </AuthContext>
  );
}
```

*Figur 8 App.js - applikasjonens hjerte*

I kodeeksempelet i Figur 8 er alt som er plassert inni <Switch> komponenter som vises om en går til komponentens URL. Komponenten LanguageSelector, som er plassert utforbi <Switch>, vil alltid bli returnert, og vises derfor på alle sidene i applikasjonen.

### 4.2.2. Visuell utforming

Alle sider og komponenter har en tilhørende CSS-fil som bestemmer hvordan sidene ser ut, og det er brukt noen metoder som går igjen i hele applikasjonen. For å få en god sidestruktur brukes flexbox, som enkelt lar en plassere og ikke minst sentrere innhold. Alt innhold på en side er lagt inn i en eller flere flex-konteinere, og da kan man enkelt bestemme om innhold for eksempel skal vises radvis eller kolonnevis, eller om innhold skal være sentrert. For å få applikasjonen til å fylle hele skjermen og at det ikke skulle være nødvendig å skrolle opp eller ned for å se innhold uansett hvilken type enhet man var på, er “Root”-kontaineren til hver side er satt til 100vh (view height). Det gjør at applikasjonen fyller 100% av høyden til vinduet i nettleseren, og at høyden på kontaineren tilpasser seg skjermen den vises på. Et eksempel på dette er vist i Figur 9.

```
.startpage {
    height: 100vh;
    width: 100%;
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    }
```

*Figur 9 CSS til startsiden.*

Det ble lagd en fargepalett for applikasjonen som definerte hvilke farger som skulle brukes. Denne vises i Figur 10. Fargepaletten ble malen for nesten alle fargevalg i applikasjonen. Applikasjonen har et bakgrunnsbilde tatt av Dava Darshan og er hentet fra en nettside [30] som tillater kommersielt bruk av sine biler. Fonten som er brukt i applikasjonen heter Karla, og er tilgjengelig hos Google Fonts [31]. Det eneste unntaket til denne fargepaletten er knappen "finish project" i "mapping".


*Figur 10 Fargepaletten til applikasjonen*

### 4.2.3. Innlogging

Innloggingssiden i frontend er enkel i den forstand at det er lite logikk som kjøres på siden. Her sendes brukeren direkte til innloggingsmetoden i backend. Backend-delen av innlogging er beskrevet mer i detalj i kapittel 4.4 Innlogging med Feide. I dette delkapittelet vil det beskrives kort hva i frontend som gjør at innloggingen fungerer.

#### *Egendefinert hook*

For å få til innlogging med Feide, og for å få backend til å snakke med frontend, måtte det lages en egendefinert hook. [32]

```
function useSetUserSession() {
  let accessToken = useToken()
  useEffect(() => {
    if (accessToken!==null) {
      var fetchstring = `https://auth.dataporten.no/openid/userinfo`;
      fetch(fetchstring, {
        method: 'GET',
        headers: {
          Authorization: `Bearer ${accessToken.access_token}`,
        }
      }).then(res => res.json()).then(data => {
        window.sessionStorage.setItem('uID',data.sub);
      });
    }
  }, [accessToken]);

}
```

*Figur 11 Koden for hvordan vi henter og lagrer bruker id som en sesjonsvariabel.*

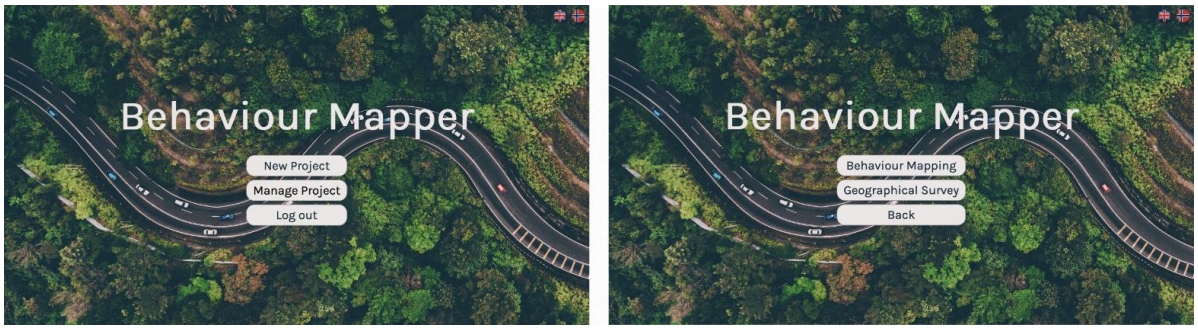
Kodesnutten i Figur 11 viser den egendefinerte hooken "useSetUserSession". I kodesnutten er "useToken" til hooken som sendes fra react-pkce pakken. React-pkce pakken er den som håndterer kommunikasjonen mellom applikasjonen og Feide. Resultatet fra "useToken()" brukes i en ny hook i

[30] Darshan, Deva, 2018, Aerial View of Road in the Middle of Trees, https://www.pexels.com/photo/aerial-view-of-road-in-the-middle-of-trees-1173777/, 19.01.21
[31] Google fonts, Karla, https://fonts.google.com/specimen/Karla 12.05.2021
[32] Facebook Inc, 2021, Building Your Own Hooks, https://reactjs.org/docs/hooks-custom.html, 18.04.21

funksjonen useSetUserSession(). Hele denne funksjonen brukes for å sette en øktvariabel. Øktvariabelen verifiserer at det er riktig bruker som spør etter informasjon i databasen. Det ble nødvendig å ha hooken i "/startpage" fordi reackt-pkce-pakken som brukes til å håndtere koblingen mot Feide allerede bruker en React hook, og fordi React ikke tillater hooker inne i hverandre.

### 4.2.4. Startside

Etter innlogging blir brukeren videresendt til applikasjonens startside. Siden har et enkelt dynamisk menysystem som lar brukeren velge om en skal lage et nytt prosjekt, bestemme prosjekttype, behandle eksisterende prosjekt eller logge ut. Om brukeren velger å lage et nytt prosjekt forandrer valgene i menyen seg slik at brukeren bestemme prosjekttype og kilde for kartet. Dette er vist i Figur 12.

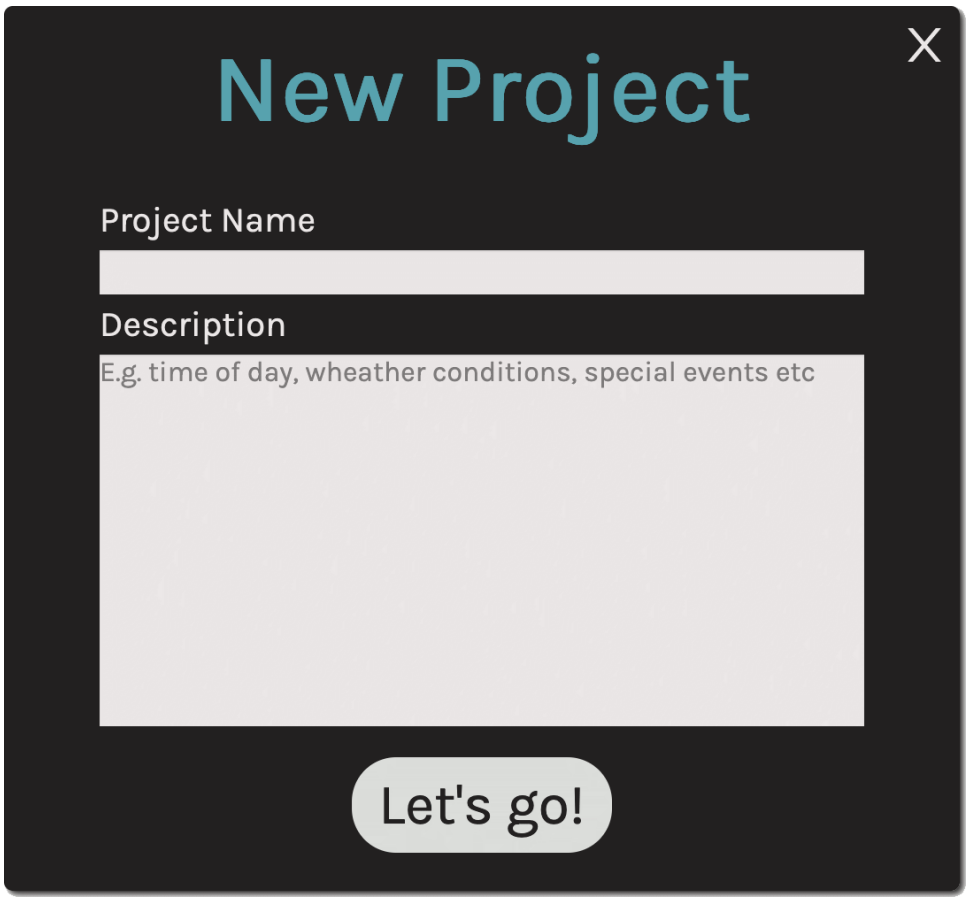



*Figur 12 Startsiden med dynamiske menyer*

Det ble brukt et tilstandsobjekt for å gjøre innholdet reaktivt. Hver meny har sitt eget tilstandsobjekt, og når dette blir satt til true, vises den aktuelle menyen. Koden er laget slik at kun en meny vises på et gitt tidspunkt.

### 4.2.5. Å starte et nytt prosjekt

Etter å ha valgt prosjekttype og kartløsning, blir brukeren sendt videre til newproject. Valgmulighetene man får på denne siden varierer ut ifra valgene man har tatt tidligere. Skal brukeren laste opp kart selv vil det være mulighet for dette. Da kan brukeren også legge til koordinatr, som vil være nødvendig for å få lastet ned shapefiler. Hvis prosjekttypen er geografisk spørreundersøkelse, vil man ha mulighet til å legge inn spørsmål til spørreundersøkelsen her. Brukeren får kun opp de valgene som er relevante for prosjektet. Valgene som blir tatt i startpage vil også bestemme hvilken side man blir sendt til videre. Dette blir gjort ved å sende tilstandsobjekter mellom de forskjellige sidene.

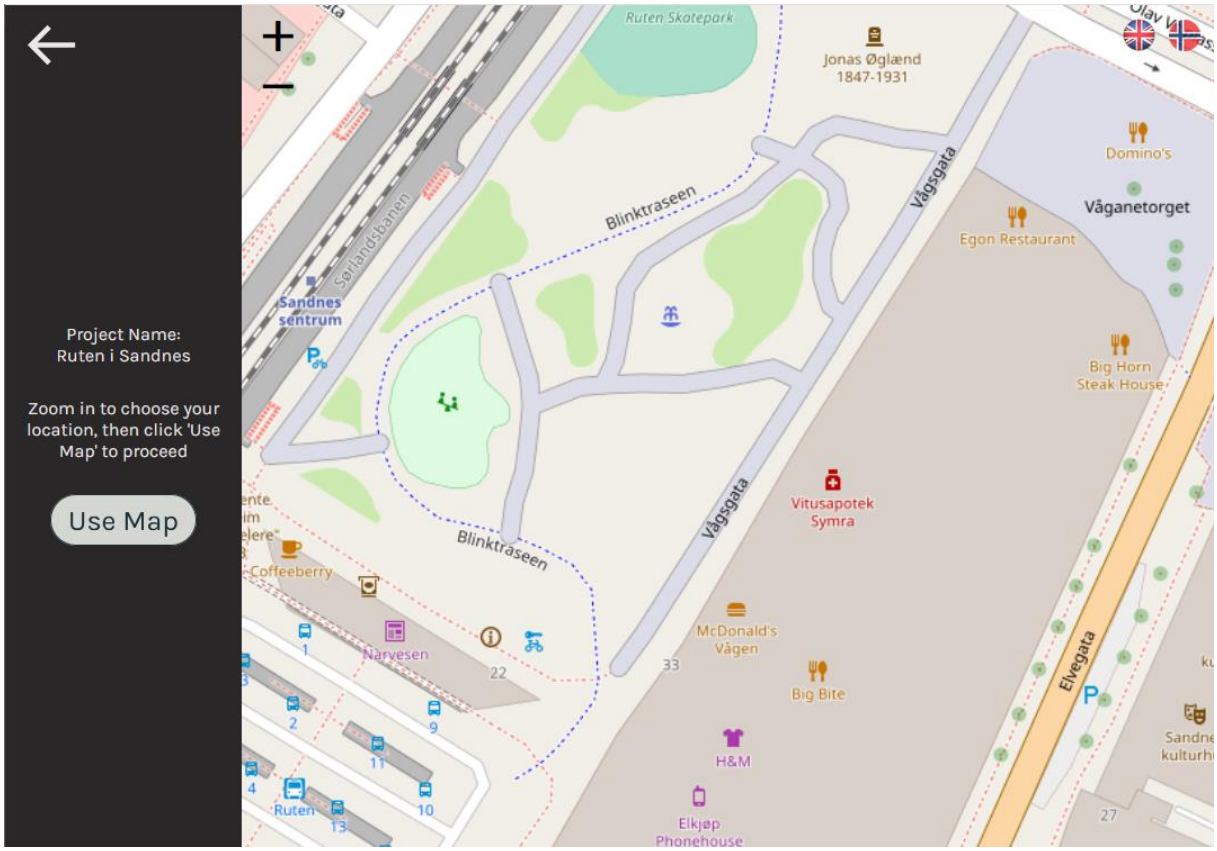



*Figur 13: Nytt prosjekt siden i mapping*

Noen av feltene i skjemaet er påkrevd for å komme videre, som for eksempel prosjektnavn. Etter å ha fylt inn nødvendige felt og trykt på "Let's go!" blir brukeren enten sendt videre til mapping (om det har blitt lastet opp et bilde) eller til siden for å velge kartutsnitt (hvis web-basert kart er valgt). Hvis brukeren har lastet opp kart selv blir et nytt prosjekt opprettet for brukeren hvor prosjektnavn, brukerID, beskrivelse, kart og startdato sendes og lagres i databasen. Prosjektet får i tillegg en egen prosjektID for å kunne identifisere prosjektet. Hvis brukeren valgte å bruke et web-basert kart sendes ingenting til databasen på dette tidspunktet. Dette gjøres da når man velger kartutsnitt, slik at man kan gå tilbake til der man oppretter prosjektet og endre prosjektnavn og beskrivelse uten å måtte gjøre ekstra spørringer til databasen.

### 4.2.6. Valg av kartutsnitt

Hvis brukeren benytter seg av den web-baserte kartløsningen blir brukeren sendt videre til en side for å velge kartutsnitt. Her kan brukeren velge hvilket kartutsnitt som skal brukes i kartleggingsprosessen.




*Figur 14: Siden der man velger kartutsnitt*

Brukeren kan zoome inn og ut på kartet, samt forflytte seg hvor en vil. Siden inneholder et sidefelt som viser navnet til prosjektet, samt ber brukeren om å zoome inn til det ønskede kartutsnittet og så velge det. Om brukeren vil endre navn eller beskrivelse, har sidefeltet en pil som leder tilbake til "/newproject" siden. Når brukeren har funnet det kartutsnittet som er ønskelig går en videre til mapping ved å trykke på "Use Map"-knappen. Siden har også to knapper en kan bruke til å zoome inn eller ut. Etter å ha trykket på "Use Map" opprettes prosjektet som beskrevet i avsnitt 4.2.5.

### Funksjonalitet

Når brukeren trykker på "Use Map" skal applikasjonen:

- Opprette et prosjektobjekt i databasen.
- Lage en bildefil av det valgte kartutsnittet og laste opp bildet.
- Gå videre til mapping, og ta med bildet fra steg 2 dit.

For å oppnå punkt 2, å laste opp bildet til databasen, gjør en funksjon følgende:

1. Lager et canvasobjekt med den samme størrelsen som kartet.
2. Oppretter en "drawing-context", heretter referert til som et tegningsobjekt.
3. Den tegner OpenLayers-kartutsnittet på tegningsobjektet og gjør tegningsobjektet om til en bildefil.

```
fetch(mapCanvas.toDataURL())
    .then(res => res.blob())
    .then(blob => {
      blob.lastModifiedDate = new Date();
      blob.name = this.state.projectName + ".png";
      var file = new File([blob], blob.name,
        { lastModified: new Date().getTime(), type: blob.type })
      const data = new FormData();
      data.append('file', file);
      data.append('p_id', this.state.p_id);
      data.append('u_id', this.state.u_id);
      data.append('map', true);
      fetch(window.backend_url + 'upload', {
        method: 'POST',
        body: data,
      }).then(setTimeout(
        () => this.redirectToMapping(), 2000));
  });
}
```

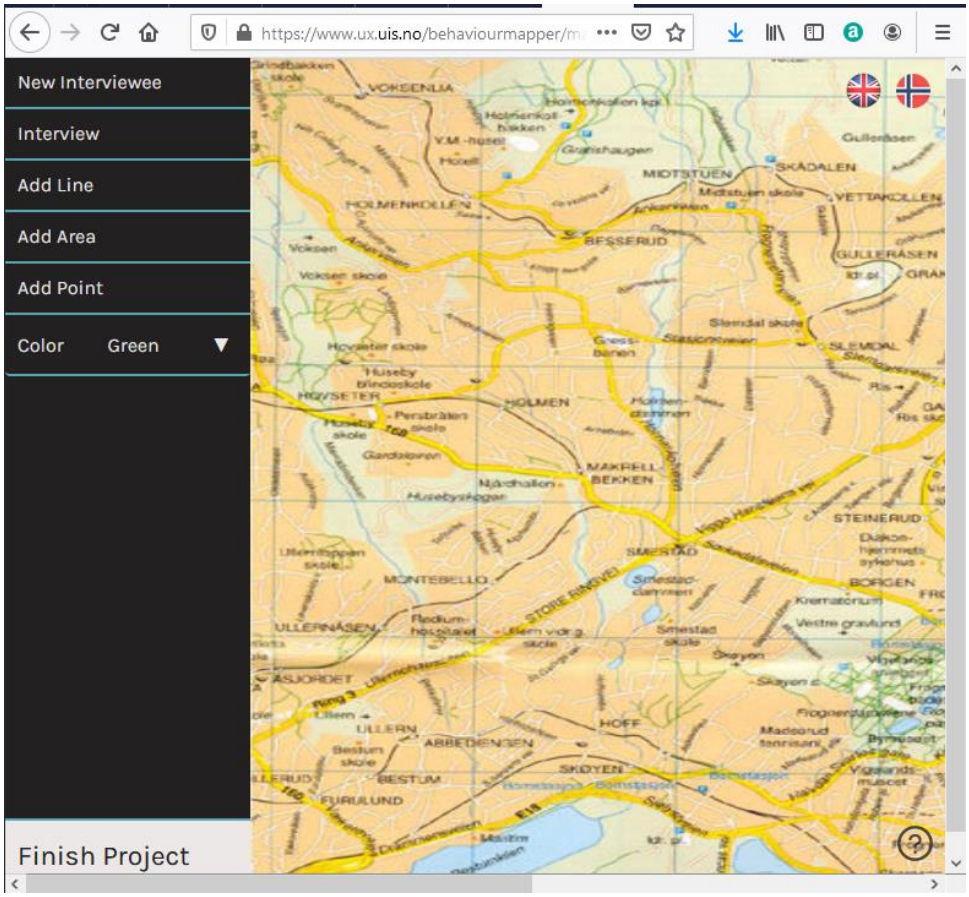
*Figur 15 Hvordan tegningsobjektet gjøres til en blob.*

Figur 15 viser hvordan tegningsobjektet gjøres om til en blob, hvordan blobben gjøres om til en png-bildefil, samt hvordan nødvendige data legges til før bildefilen sendes til backend.

### 4.2.7. Mapping

Det er på denne siden hoveddelen av funksjonaliteten til prosjektet befinner seg. Her kan man gjøre både atferdskartlegging og geografiske spørreundersøkelser. Hvilken modus brukeren er i bestemmes som allerede nevnt ut ifra valg gjort tidligere.

Dersom man laster opp bilder vil bildet beholde det samme størrelsesforholdet konstant. Dersom bildet velges fra verdenskartet vil det strekkes og komprimeres slik at det fyller hele skjermen. En css forandring som vil gjøre at både opplastede og webbaserte bilder beholder samme høyde-bredde-forhold er beskrevet i kapittel 5.5.

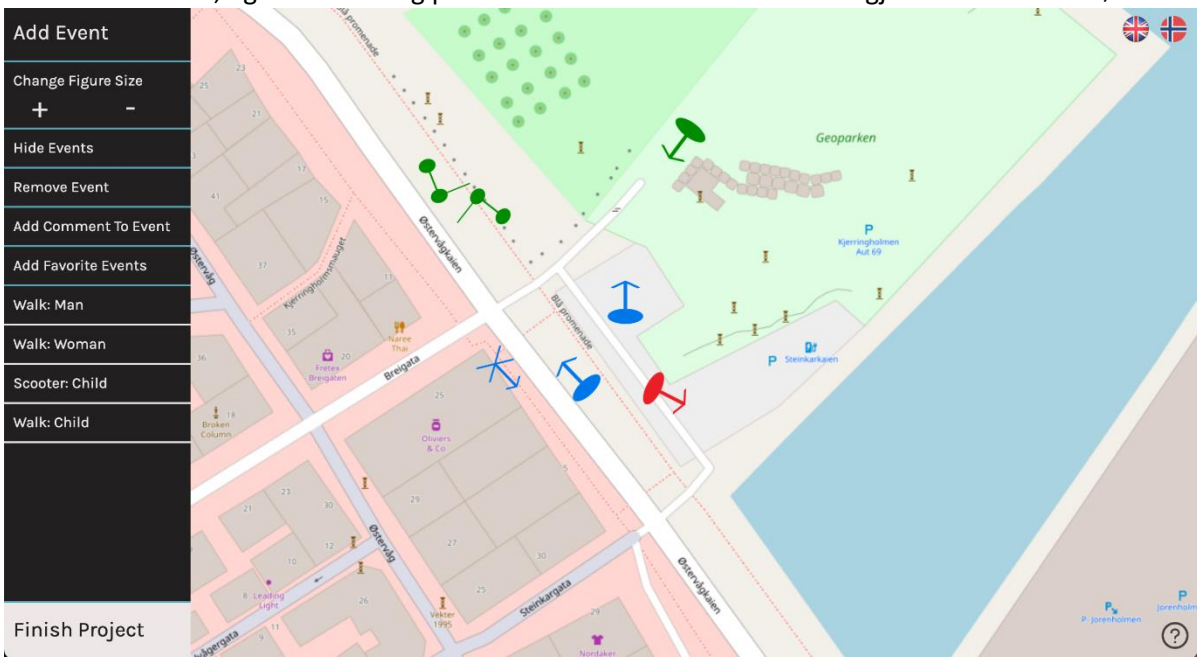



*Figur 16 Dette bildet ble lastet opp manuelt, og strekkes/komprimeres derfor ikke med skjermen*

Dersom kartet forandrer seg i henhold til skjermen (kartet forstørres/forminskes) vil ikonene flyttes til rett plass ved hjelp av funksjonen placeEventsAfterChange.

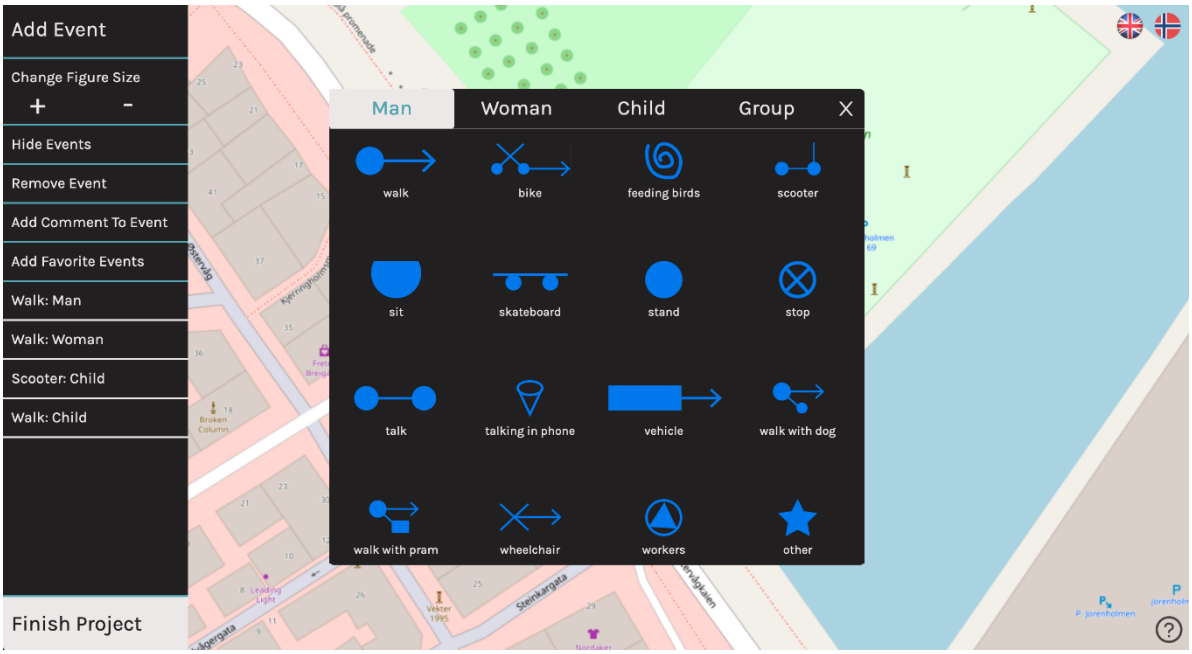
### 4.2.8. Atferdskartlegging

I denne delen skal man kunne plassere hendelsene på kartutsnittet. Hendelsene vises som ikoner på kartet, som vist under i Figur 17 Atferdskartlegging slik brukeren ser det.. Man kan snu hendelsene den veien man vil, og de holder seg på samme sted selv om ikonene blir gjort mindre eller større.




*Figur 17 Atferdskartlegging slik brukeren ser det.*

Når brukeren klikker på kartutsnittet kalles en funksjon som enten plasserer en hendelse eller snur en allerede eksisterende hendelse. Dette gjelder kun hvis man klikker på selve kartet, klikker man direkte på en hendelse vil man velge hendelsen slik at man kan legge til en kommentar, forandre en allerede eksisterende kommentar, eller slette hendelsen. Knappene "Add Comment" og "Remove Icon" utfører de respektive hendelsene. Man kan endre størrelsen på hendelsene ved å trykke på "Change size of events", og alle hendelsene i prosjektet vil ha samme størrelse.




*Figur 18 Det brukeren vil se etter å ha trykket på "Add Event"*

Brukerne velger ikoner ved å trykke på "Add Event" eller ved å velge et av brukerens favorittikoner (Vist i Figur 18). Dette er beskrevet mer i delkapittel på side . Knappen som her viser "Hide Icons" vil veksle mellom dette og "Show Icon", og viser eller skjuler alle hendelsene som har blitt registrert. Knappen "Finish Project", som er markert med grønn tekst og som er nederst i sidefeltet vil ta et bilde av prosjektet med alle ikonene til stede og sende brukeren tilbake til startsiden. Dette skjermbildet vil så kunne vises på siden for å behandle prosjekt.

### Å plassere og peke et ikon

Som nevnt kan man ved å trykke på kartet enten plassere et ikon, eller snu et ikon. Hvilken handling man tar bestemmes av en Boolsk verdi.

Ikonene er lagret som png-filer. For å plassere bildet gjør vi følgende:

- Koordinatene til musepekeren fra hendelsesobjektet sendes inn til en funksjon.
- Disse verdiene forandres slik at midten av ikonet havner der musepekeren er.
- X-verdien økes med 200 piksler er for å ta hensyn til sidefeltet.
- Dersom brukeren har brukt scrollbaren finner applikasjonen avstanden scrollbaren har flyttet seg, og dette legger denne til hendelsens koordinater.

Deretter kalles det på createIconObjekt for å lage et objekt som inneholder informasjon om bildet. Dette objektet vil brukes senere for å flytte ikonene når bildet forminskes eller forstørres.

Funksjonen pointIcon vises i Figur 19:

```
pointIcon() {
   var degreerot = Math.atan2(
      this.state.ourMouseCoord.x - this.state.ourIconCoord.x,
      -(this.state.ourMouseCoord.y - this.state.ourIconCoord.y),
   );
   var degrees = degreerot*180/Math.PI - 90;
   var round_degree = Math.round(degrees);
   var string_degree = 'rotate(' + round_degree.toString() +'deg)';

   this.setState({
     ourIconCoord: {
       x: this.state.ourIconCoord.x,
       y: this.state.ourIconCoord.y,
       degree: string_degree
     }
   });
   var img = document.getElementById(this.state.ourIconID.toString());
   if (string_degree != null) {
     img.style.transform = string_degree;
   }
 }
```

*Figur 19 Koden for å peke ikoner.*

I pointIcon brukes en funksjon fra "Math"-importen for å finne tangenten mellom senteret til det valgte bildet og musepekeren. Dette gjøres om til grader ved annen funksjonalitet i Math rammeverket, og så lages en streng som brukes for å forandre på ikonets rotasjon. Etterpå legges rotasjonen til som en av ikonets verdier, slik at også rotasjonen kan sendes til databasen.

### Å legge til hendelser

Når brukeren trykker på "add event"-knappen i sidefeltet kommer det opp et vindu hvor en kan velge mellom forskjellige symboler som representerer kjønn og handling. Se Figur 18. Øverst i dette

vinduet er det fire knapper hvor brukeren kan velge å registrere en mann, kvinne, barn eller en gruppe. Når en har funnet det rette ikonet og trykker på det, forsvinner vinduet, og en kan plassere ikonet på kartet.

Hvert ikon blir initialisiert som en komponent kalt "Icon", og hvert ikon er en del av en større komponent kalt "AllIcons". I AllIcons gjøres en spørring til databasen for å få tak i data (farge og beskrivelse) til hvert enkelt ikon, og disse dataene legges så inn i fire forskjellige lister basert på hvilken farge ikonene er. Videre brukes en innebygd JavaScript-funksjon kalt "map", som itererer gjennom listene. For hver iterasjon lages en Icon-komponent hvor farge og beskrivelse sendes inn som props til komponenten "Icon". Hvert enkelt ikon blir herfra hentet fra databasen. For hver spørring blir farge og beskrivelse sendt inn som argumenter, dette avgjør hvilket ikon som blir hentet.

### Å velge ikon

Man velger ikon med å trykke på det. Som nevnt over kan ikonet da slettes eller gis en kommentar.

### Ta skjermbilde

Når brukeren går inn i "manageProject", vil han/hun få opp en liste med alle prosjekt brukeren tidligere har opprettet. For å gjøre det lettere å indentifisere det ønskede prosjektet for brukeren, har hvert prosjekt et tilhørende skjermbilde av prosjektet. For å ta dette skjermbilde brukes et bibliotek kalt dom-to-image [33]. Dom-to-image lar en spesifisere hvilken del av koden en vil ta skjermbilde av, og er konfigurert til å ta bilde av brukerens kart samt alle ikonene som er utplassert. Selve bildet blir tatt når brukeren trykker på finish project-knappen i sidefeltet. Ved å implementere denne funksjonaliteten her, sikres det at bildet blir tatt når alle ønskede ikoner er lagt til. Bildet sendes så til databasen, denne prosessen er beskrevet nærmere i delkapittelet om Opplasting og hente bilde API35.

### Slette hendelser

Dersom brukeren har valgt et ikon ved å trykke på det vil dette ikonet fjernes når brukeren trykker på "Remove Event". "Remove event" vil også fjerne hendelsen i databasen dersom det eksisterer der. Objektet "createIconObject" brukes av funksjonen "removeIcon" for å knytte ikonet i frontend mot sin representasjon i databasen.

Dersom hendelsen slettes fra frontend før den sendes inn til databasen vil den ikke slettes fra databasen.  Dette er beskrevet på slutten av kapittel 5.5

### Legge til kommentarer til hendelser

For å kunne favne bredere ved de ferdigdefinerte hendelsene ble det lagt til funksjonalitet for å feste en kommentar til en valgt hendelse. For å få til dette ble det brukt mye av funksjonaliteten som blir brukt når en sletter hendelser. En får en den korrekte id-en til hendelsen og sender en oppdatering til databasen, samt legger til kommentarer i en tilstand for å holde styr på hvilken kommentar som hører til hvilken hendelse.

### Endre størrelse på ikoner

Ettersom zoom-nivået på kartet som brukes i kartleggingen vil variere, var det også ønskelig å kunne variere størrelsen på ikonene for å tilpasse de til kartet. For å endre størrelsen på hendelser ble det

[33] Saienko, Anatolii and Bakaus, Paul, 2017, DOM to Image, https://github.com/tsayen/dom-to-image, 17.04.21

lagt til to ekstra knapper i kartleggingsiden. Utordringen med å endre på størrelsen var å få hendelsene til å ikke flytte på seg når figuren ble større eller mindre. Alt ble håndtert ved å bruk av tilstands-objekter, og html-dom.

#### Hurtigmeny med hendelser

For å gjøre det raskere å legge til nye hendelser, var det et ønske om å lage en hurtigmeny hvor brukeren av applikasjonen selv kunne legge til de mest brukte hendelsene. På denne måten ble det ikke nødvendig å gå inn på menyen hver gang en skulle registrere en ny hendelse. Hurtigmenyen er laget som en ny komponent som bygger på komponenten AllIcons.

### 4.2.9. Geografiske spørreundersøkelser

For geografiske spørreundersøkelser var det behov for å tegne linjer og punkt, ramme inn områder og stille spørsmål til intervjuobjekter. For å få mulighet til å tegne på kartet valgte vi å bruke et HTML-lerret [34], som er et element som blir brukt til å tegne på. Et HTML-lerret har flere innebygde funksjoner som gjør det mulig å tegne og tilfredsstilte dermed kravene til denne delen av applikasjonen.

#### Tegne linjer, områder og punkter

For å tegne registrerer applikasjonen først alle koordinatene der en har markert, og når brukeren er ferdig med tegningen, kaller applikasjonen på en funksjon som tegner alle punktene til den respektive figuren på kartet. Det er også laget en funksjon for å lage en pilspiss på linje for å definere retning. Funksjonen for å lage pilspissen ble hentet fra nettet. [35]

#### Velge farge

For å kunne velge farge på figurene som tegnes på HTML-lerretet er det en meny der valgene lagres i et tilstandsobjekt. Denne brukes igjen til å bestemme farge når figurene tegnes. For å få menyen til å se bra ut bruktes CSS-kode av Raúl Barrera [36].

### 4.2.10. Manage project

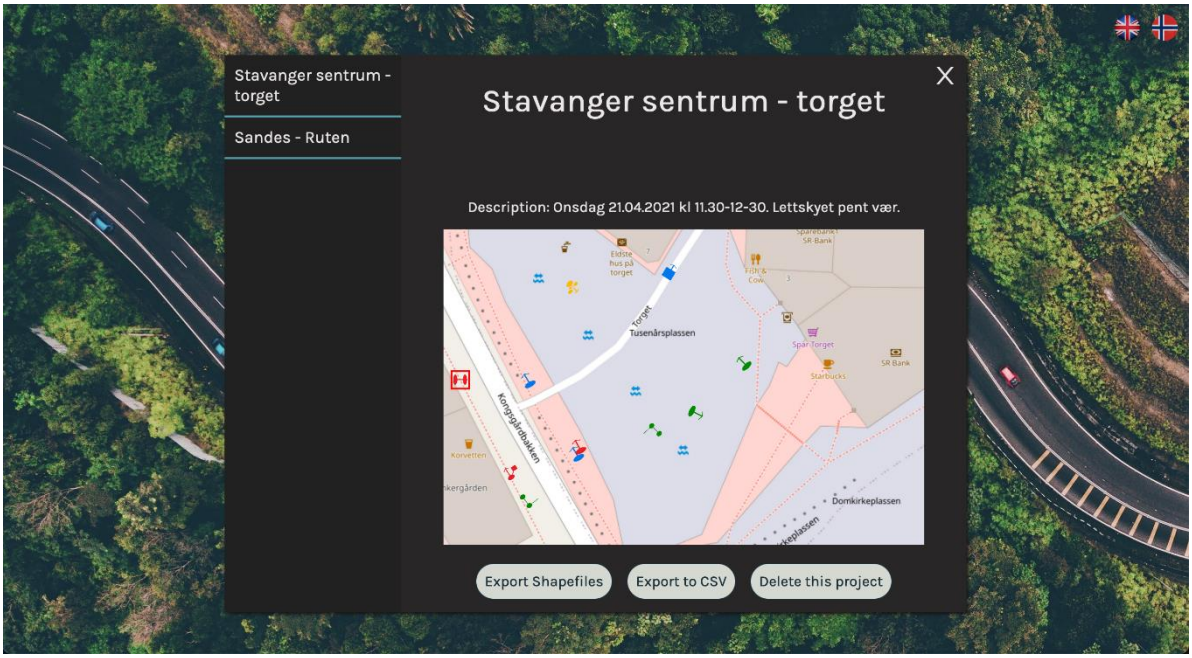
Når man går inn i manage project hentes alle prosjektene til den innloggede brukeren fra databasen. Trykker man på et av prosjektene i menyen til venstre, vil en React hook hente all relevant prosjektinformasjon før det vises. Disse dataene sendes til en komponent som er barnet til sidens

[34] Refsnes Data, 1999-2021, SQL HTML Canvas Graphics, https://www.w3schools.com/html/html5_canvas.asp, 20.04.21

[35] www.programmersought.com, 2018-2021, How to draw a line with an arrow on the canvas?, https://www.programmersought.com/article/16051099203/, 20.03.21

[36] Barrera, Raúl, 2016, Pure CSS Select, https://codepen.io/raubaca/pen/VejpQP, 13.05.21

root-komponent. De relevante dataene, samt en funksjon, sendes til barnekomponenten ved hjelp av props.




*Figur 20 Behandle prosjekt siden, med et valgt prosjekt.*

Når brukeren trykker på "Export Shapefiles" vil programmet eksportere shapefilene til det valgte prosjektet i en zip-fil. Dette gjøres i backend, og diskuteres mer i delkapittelet Shape-fil Eksportering API på side 36. Brukeren exporterer en "csv" fil ved å trykke på knappen "Export to csv". Trykker brukeren på "Delete this project" vil brukeren få et varsel der en må bekrefte om en vil slette prosjektet. Hvis valget bekreftes, vil prosjektet og all tilhørig informasjon i databasen slettes.

### 4.2.11. Flere språk

I utgangspunktet var applikasjonen kun tilgjengelig på engelsk da den ville bli brukt av flere internasjonale studenter. Etter tilbakemelding fra en av testerne av applikasjonen ble det også implementert en norsk versjon av siden. Dette ble gjort ved å bruke rammeverket i18Next. i18Next støtter React ved bruk av pakken React i18Next. I18Next gjør det mulig å legge til flere språk og skifte mellom dem ved å opprette en JSON-fil for hvert ønsket språk. En utfordring var at alle ikonnavn kom direkte fra databasen, og måtte dermed oversettes på en annen måte. Dette ble løst ved å hente ut nåværende brukte språk fra i18Next, som vist i kodesegmentet i Figur 21. Deretter ble det lagd to seksjoner, en på engelsk og en på norsk. Seksjonene vises hvis språket den er skrevet på stemmer overens med språket som hentes fra i18Next.

```
<div className={i18next.languages[0] === "en" ? "icon-description" : "invisible"}>
<div className={i18next.languages[0] === "no" ? "icon-description" : "invisible"}>
```

*Figur 21 Eksempel på oversetting.*

### 4.2.12. Testing

Testene i frontend er laget for å sjekke at ikke noe blir ødelagt etter hvert som GitHub-grener sys sammen med hovedversjonen. Når en bruker create-react-app [37] følger det automatisk med Jest for å kjøre tester. Testene i applikasjonen sjekker at en komponent eller side blir vist uten feil. Figur 22 viser et eksempel på hvordan en slik test ser ut.

```
it('displays the Manageprojectpage', () => {
    const testlist = shallow(
        <ManageProject />
  );
  const sidebarComponents = testlist.find('div')
  expect(sidebarComponents.length).toEqual(3);
});
```

*Figur 22: Eksempel av en test i frontend.*

[37] Facebook Inc, 2021, Create React App, https://create-react-app.dev/, 19.01.21

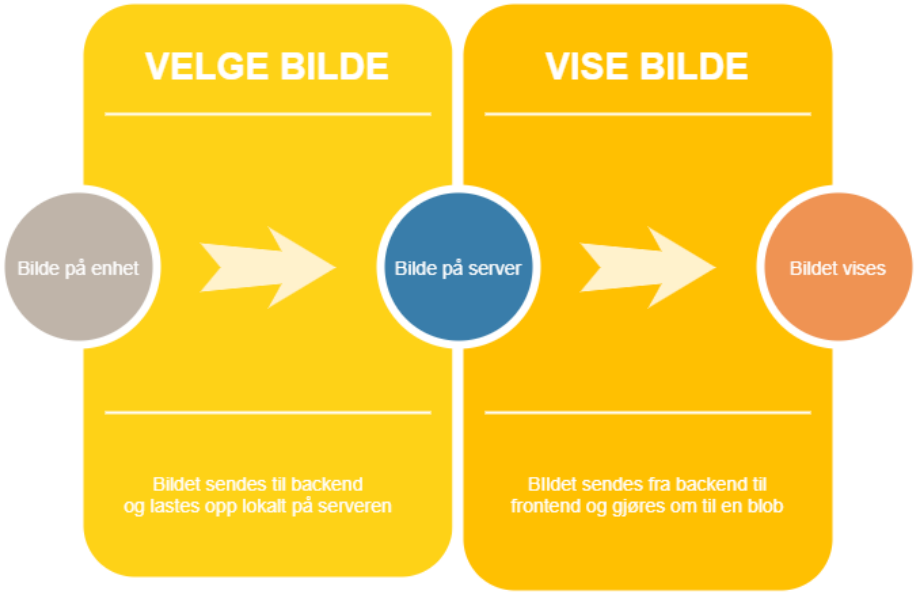
## 4.3. Backend

### 4.3.1. Python og Flask

I all hovedsak er det to guider som ble fulgt for å lage backend til applikasjonen vår. [38][50] Den første guiden beskriver kort hvordan man får frontend til å snakke med backend, mens den andre beskriver hvordan et fullt flask-prosjekt lages. Det viktigste i den første guiden var hvordan man tillater CORS-forespørsler, samt hvordan man legger til en proxytjener. En proxytjener i denne sammenheng er en referanse i pakken til frontend som beskriver hvor fetch-forespørsler sendes under utvikling.

#### Opplasting og hente bilde API

Alle bildene som brukeren laster opp, samt kartutsnittene de velger og skjermbildene som tas når brukeren avslutter et prosjekt, lastes opp til en mappe på serveren. Referansen til hvor de ligger lagres i databasen og brukes når applikasjonen skal hente bildene frem igjen. Prosessen for hvordan bildet blir behandlet blir vist i grove trekk under i Figur 23.




*Figur 23: Flyten for opplasting og visning av bilder.*

Bildet blir opprettet i eller lastes opp i applikasjonen, og sendes til backend. Deretter lagres det lokalt i en mappe på tjeneren. Når brukeren har bruk for dette bildet sendes en forespørsel til tjeneren, som finner bildet i bildemappen og sender det tilbake til applikasjonen som en blob. En blob eller et binary-large-objekt er et fillignende objekt som kan representere data som ikke er i et JavaScript-format. Binærobjektet gjøres så om til en bildefil i applikasjonen, og er da klart til å vises. Figur 24 viser koden for å gjøre et hentet bilde til en blob.

```
fetch(window.backend_url + `getmap?p_id=${this.state.p_id}&u_id=${this.state.u_id}`).then(res => res.blob())
  .then(images => {
    let image = URL.createObjectURL(images);
    this.setState({mapblob: image});
```

*Figur 24: Koden for å gjøre et hentet bilde til en blob.*

[38] Ukirde, Divyajyoti, 2020, How to create React App with Flask backend?, https://dev.to/divyajyotiuk/how-to-create-react-app-with-flask-backend-2nf7, 23.01.21

For å gjøre opplastingen sikker gjøres følgende: Filnavnet saneres, det sørges for at filnavnet er unikt og filendingene sjekkes. Når dette er klart lagres bildefilen på tjeneren.

Når applikasjonen henter et bilde eller et kart sendes en forespørsel til opplastings API-et i backend. Forespørselen må ha informasjon om hvilket prosjekt den skal hente bildet fra og hvilken bruker det er som sender forespørselen. Om både brukeren og prosjektet er riktig får en tilbake et svar med det ønskede bildet.

### Shape-fil Eksportering API

Brukerne kan eksportere det de har gjort til en ArcGIS fil. Dette API'et gjør følgende: Først hentes relevant informasjon, som er hjørnekoordinatene til området eller bildet. Dersom prosjektet ikke har hjørnekoordinater, vil shape-fil folderen være tom. Så sjekker applikasjonen om prosjektet det dreier seg om er en geografisk spørreundersøkelse eller et atferdskartleggingsposjekt. Dersom prosjektet er et atferdskartleggingsprosjekt vil applikasjonen hente de relevante hendelsene, sortere dem, og skrive dem inn i de relevante shape-filene. Dette gjøres ved å iterere gjennom et Python-dictionary. Etter å ha forandret de relevante shape-filene sendes folderne med forandret innhold til brukeren som en zip-fil.

Dersom prosjektet er en geografisk spørreundersøkelse, vil applikasjonen hente alle intervjuer og alle tilhørende intervjufigurer. Applikasjonen vil så iterere igjennom intervjuobjektene og deres figurer, og forandre på shape-filene i de relevante mappene. Dette gjøres også ved hjelp av Python-dictionary. Mappene vil så bli gitt nye navn og de legges til en zip-fil før de sendes til brukerne. Etter dette tilbakestilles mappenavnene. Slik kan zip-filen inneholde mange flere mapper enn det som finnes opprinnelig på applikasjonen.

### Eksportere til CSV API

Brukerne av applikasjonen skal også kunne eksportere data fra et prosjekt til CSV-filer. Dette blir gjort ved å hente all informasjon som er knyttet til det valgte prosjektet for så å skrive det inn i fire CSV-filer. Deretter pakkes de om til en ZIP-fil som kan lastes ned. Basert på om en har gjennomført atferdskartlegging eller en geografisk spørreundersøkelse vil noen av CSV-filene være tomme, slik at en ikke får med seg unødvendig informasjon.

#### *Legge til og hente informasjon API*

De aller fleste av API-ene til applikasjonen er laget for å hente og legge til informasjon i databasen. Endringen fra det ene API-et til det andre er hovedsakelig SQL-spørringen, med noen unntak. Det kan for eksempel se ut som vist i Figur 25:

```
add_interview = ("INSERT INTO InterviewEvents (interview, p_id) VALUES (?,?)")
args = (request.form.get('interview'), request.form.get('p_id'))
i_id = query_db(add_interview, args)
return {"i_id": i_id}
```

*Figur 25 Eksempel på hvordan en SQL-spørring gjøres imot databasen.*

Her hentes informasjonen fra en HTTP POST-spørring for så å sendes til funksjonen som skriver til databasen. Av alt som returneres fra databasen er det i dette tilfellet bare IDen til intervjuet som skal sendes tilbake. For en GET-spørring vil informasjonen vi ønsker som oftest være mer enn bare en verdi. Da brukes som oftest en for-løkke for å hente ut de ønskede verdiene, og de blir så returnert i JSON-format.

*Testing*

Det har blitt laget noen automatiske tester som tester applikasjonens API-er for å sikre at de fungerer som de skal. Testene er ikke fullverdige tester da de verken tester grensetilfeller eller feilsituasjoner, for å være fullverdige burde de testet flere situasjoner enn det de gjør nå.

For testing brukes rammeverket Pytest [39] som er anbefalt av Flask. Pytest er et mye brukt testrammeverk til Python. I oppsettet til applikasjonen settes det opp mulighet for å bruke flere typer testkjørere. Testkjørerne brukes for å kjøre forskjellige tester på ulikt nivå. Testene tilhørende applikasjonens API-er sjekker om de får tilbake forventet HTTP-responskode eller om de får tilbake de riktige dataene når de skal hente data ut fra databasen. For opplasting lastes det opp et testbilde som sjekkes mot det samme bildet fra databasen for å se om de er identiske, i tillegg sjekkes HTTP-responskoden. Koden for å fikse opplastingstesten fikk vi inspirasjon fra her. [40]

```
def test_upload_image(client):
    image = "bike.png"
    data = { "file": (open("behaviourmapper/static/icons/man/bike.png", 'rb'), image), "p_id": 1, "u_id": "openid"}
    rv = client.post(('/behaviourmapper/upload'), data=data)
    assert rv.status_code == 201
    assert rv.json['file'] == image
```

*Figur 26: Kode for å teste opplasting av bilder.*

### 4.3.2. Database

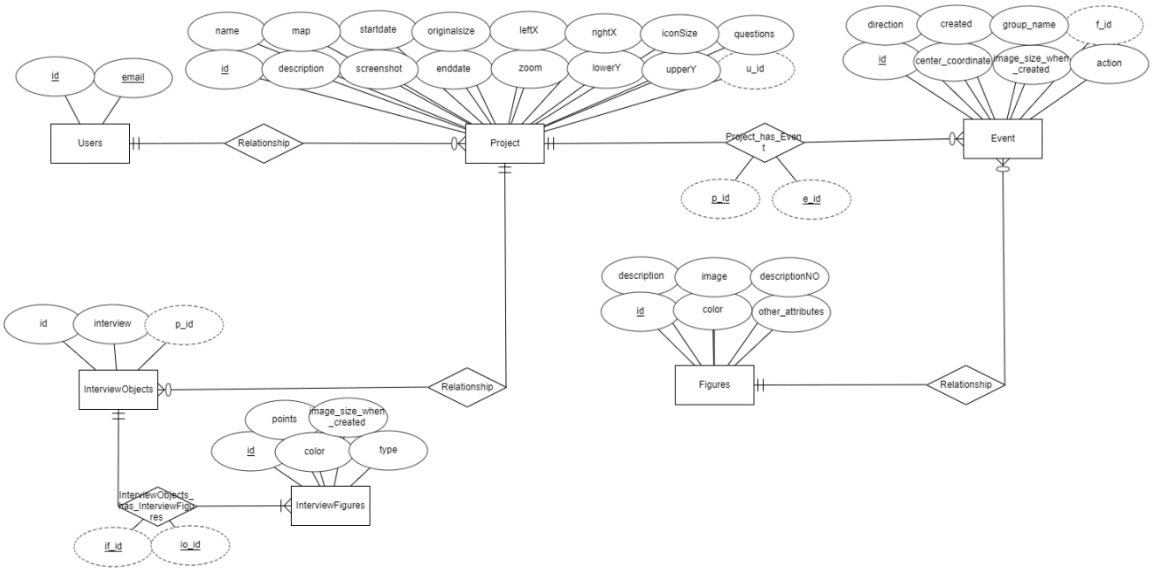
Oppsettet av databasen med SQLite er ganske enkelt. Det fungerer slik at Flask kobler seg opp mot en databasefil på en dynamisk filsti.

Den endelige versjonen av databasen er vist i Figur 27 under. Databasen er delt opp i åtte tabeller. To av tabellene er relasjonstabeller som knytter to andre tabeller sammen. Ellers er det seks tabeller med ulike attributter som bevarer all informasjon om prosjektene, og mye av dette brukes for å lage

[39] Pallets, 2020, Testing Flask Applications, https://flask.palletsprojects.com/en/1.1.x/testing/, 30.01.21

[40] www.programmersought.com, 2018-2021, [Flask] pytest file upload unit test, https://www.programmersought.com/article/57254767787/, 27.04.21

shapefiler.




*Figur 27 ER-diagram av databasen.*

Koden som håndterer opprettelsen av tabeller og legger inn data for figurene som alltid skal være i databasen er håndtert i en SQL-fil. SQL-filen kjøres når det aktiveres en tilpasset kommando i Flask. En tilpasset kommando er en kommando som kjøres fra en terminal som ikke finnes før den har blitt definert i koden. Kommandoen brukt i applikasjonen er hentet ut fra guiden til Flask [34], og vil initialisere databasen dersom den ikke fins. Tabellene som er opprettet har alle en primærnøkkel og noen har fremmednøkler som kobler dem opp mot andre tabeller.

```
CREATE TABLE IF NOT EXISTS [Event] (
    "id" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    "direction" DECIMAL NULL,
    "center_coordinate" VARCHAR NULL,
    "image_size_when_created" VARCHAR NULL,
    "created" TIME NULL,
    "comment" VARCHAR NULL,
    "f_id" INTEGER NOT NULL,
    FOREIGN KEY (f_id) REFERENCES [Figures](id)
);
```

*Figur 28: Utdrag fra koden for å lage tabeller med gitte attributter.*

I Figur 28 vises et eksempel av hvordan koden i SQL-filen for å opprette databasetabeller ser ut.

```
def query_db(query, values, one=False):
    db = get_db()
    cursor = db.execute(query, values)
    rows = cursor.fetchall()
    rows.append(cursor.lastrowid)
    cursor.close()
    db.commit()
    return (rows[0] if rows else None) if one else rows
```

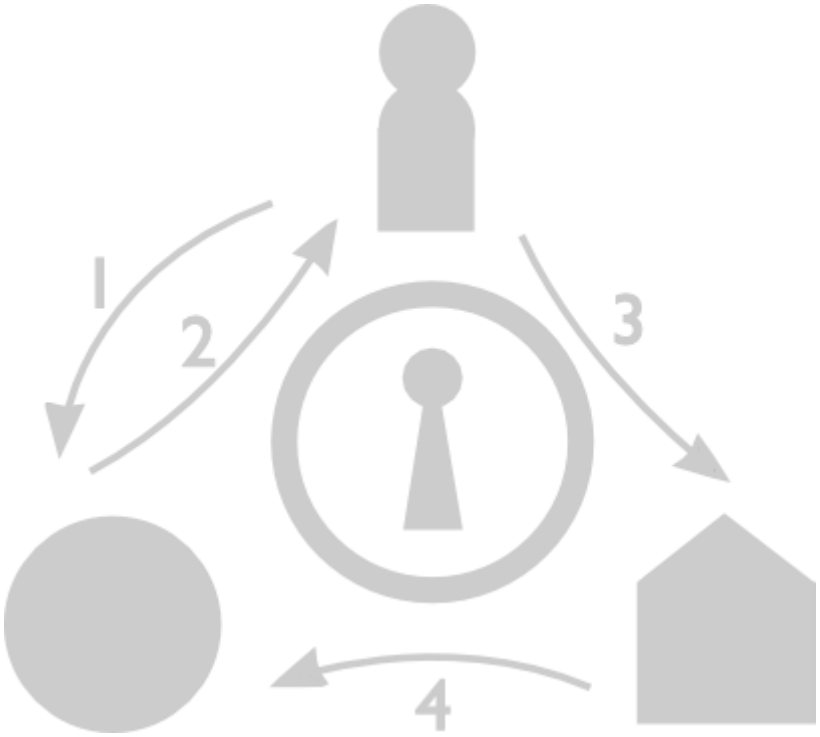
*Figur 29: Kode for spørring mot databasen.*

I Figur 29 vises koden som brukes i de aller fleste spørringene til databasen. Denne koden ligger i en egen fil i backend sammen med flere innstillinger for koblingen av databasen mot Flask. Den henter spørringene og sender de til databasen med gitte innsendte verdier. Det er også sikret ved å bruke forberedte spørringer, som er gjort for å forhindre SQL-injeksjoner. Koden for å kjøre spørringene har vi hentet fra kode vi skrev i Datasikkerhet.

### 4.3.3. Distribusjon til Unix-miljøet/Apachetjener

For å distribuere applikasjonen brukes npm for frontend og wheel i backend for å lage pakker. Når pakkene er klare, kan de legges ut på Apachetjeneren. Å få distribuert var en lengre prosess og større utfordring enn ventet. Hvorfor dette var tilfellet og hvordan vi løste det er beskrevet i kapittel 5.6.3 og deler i kapittel 5.6.1 som handler om shapefiler.

## 4.4. Innlogging med Feide




*Figur 30 Bildet og tekst er hentet fra Feide og er beskrevet i listen under. [ 41]*

Når en bruker logger inn i applikasjonen med sin Feide identitet skjer følgende:

1. *"Brukeren får tilgang til påloggingssiden til applikasjonen i en nettleser.*
2. *Applikasjonen sender brukeren til Feides påloggingstjeneste.*
3. *Brukeren legger inn Feide brukerens brukernavn og passord, som Feide sender til brukerens hjemmeorganisasjon.*
4. *Brukernavn og passord verifiseres av hjemmeorganisasjonen, og hvis de blir verifisert, kan brukerens personlige data sendes til applikasjonen via Feide. Dataene som mottas er forhåndsdefinert."* [41]

### Frontend og backend

For å koble applikasjonen sammen med Feide og OIDC ble det brukt to pakker, én for frontend og én for backend. Valget av flask-oidc [42] som pakke til backend ble gjort fordi den hadde alle nødvendige funksjoner samtidig som den ikke var for komplisert. Alternativet som ble sett på var Authlib [43]. Authlib er i likhet med flask-oidc en pakke som støtter OIDC. Den har mer funksjonalitet og er mer kompleks. Grunnen til at valget endte på flask-oidc var fordi det var lettere å ta i bruk, samt at det hadde støtte for dekoratører som legges til i starten av en funksjon for å fortelle at den for eksempel krever innlogging.

[41] Uninett, 2021, Feide, https://docs.feide.no/general/feide_overview.html, 17.04.21
[42] Uiterwijk, Patrick, 2021, Flask-OIDC, https://flask-oidc.readthedocs.io/en/latest/, 24.03.21
[43] Yang, Hsiaoming, 2021, Authlib: Python Authentication, https://docs.authlib.org/en/latest/index.html, 24.04.21

I frontend ble det forsøkt å følge en guide fra ID-porten som beskrev hvordan man bruker React med OIDC, men den krevde en stor omskriving av applikasjonen for å støtte dekoratører. Løsningen var å benytte oss av react-pkce pakken [44], noe som også er nåværende beste praksis [45]. For at applikasjonen skulle fungere måtte en av kodefilene til pakken endres, da den spurte etter feil url. Endringen er beskrevet i Frontend på side 52 i Appendikset.

```
const clientId = process.env.REACT_APP_CLIENT_ID
const clientSecret = process.env.REACT_APP_CLIENT_SECRET
const provider = process.env.REACT_APP_PROVIDER
```

*Figur 31 Hvordan noen av hemmelighetene i applikasjonen hentes.*

Figur 31viser de verdiene som gjør det mulig å koble seg mot feide i frontend. Disse verdiene hentes fra en .env fil i frontend for å gjøre de vanskeligere å få tak i.

Når brukere logger inn i applikasjonen fra påloggingssiden sendes de til et API i backend som håndterer pålogging. Denne logger brukeren på i backend. I tillegg legges brukerinformasjonen til i databasen om det er en ny bruker, og brukeridentifikasjonen legges i et sesjonsobjekt i backend. Det som blir lagret i databasen om brukere er brukeridentifikasjon og e-post. Disse blir sendt fra Feide etter en suksessfull pålogging.

```
@oidc.require_login
def authenticateUser(u_id):
    if 'username' in session:
        if session['username'] == u_id:
            return True
        else:
            return False
    return False
```

*Figur 32 Koden for å autentisere brukeren.*

For å sikre at det kun er den påloggede brukeren som får tilgang til sine egne data via API-ene i backend, er det en funksjon vist i Figur 32 som sjekker sesjonsvariabelen mot brukeridentifikasjonen som kommer fra frontend.

For å nå funksjonen authenticateUser brukes dekoratøren @oidc.require login, som sjekker om brukeren er pålogget. Hvis dekoratøren ikke brukes, vil en ikke få tilgang til det korrekte sesjonsobjektet. Deretter sjekker authenticateUser brukeridentifikasjonen fra frontend mot brukeridentifikasjonen som er lagret i sessionsobjektet i backend.

For at brukeren skal bli korrekt logget ut må de klikke på logout-knappen i applikasjonen. Pakken vi bruker for OIDC i backend har ikke en tidsstyrt avloggingsmetode, derfor må den kjøres manuelt. Det siste som gjøres i logout-funksjonen er at brukeren sendes til Feide for å logges ut der.

[44] Chris927, 2020, react-pkce, https://www.npmjs.com/package/react-pkce, 26.04.21
[45] Uninett, 2021, Obtaining tokens with Feide, https://docs.feide.no/service_providers/openid_connect/feide_obtaining_tokens.html, 17.04.21

```python
@bp.route('/logout')
@oidc.require_login
def logout():
    session.pop('username', None)
    oidc.logout()
    return redirect("https://auth.dataporten.no/openid/endsession", )
```

*Figur 33 Viser hva som skjer i backend når vi kaller på logout API-et.*

Figur 33 viser avloggings-APIet i backend. Funksjonen som er i frontend fjerner først sesjonsvariabelen og sender så brukeren videre hit.

## 4.5. GitHub og Travis

GitHub gjør det mulig å skrive i den samme kodebasen selv om en er på forskjellige enheter. Det som ble benyttet mest med GitHub er muligheten for å legge til nye problemer slik at en enkelt kan finne nye arbeidsoppgaver når en trenger det. Det ga også mulighet til at flere jobbet med det samme problemet til tider, via "LiveShare" i utviklingsverktøyet Visual Studio Code. GitHub gjør det også mulig å bruke grener der en jobber kun med å en konkret oppgave uten å forandre på andres kode.

For å sikre oss at det ikke ble laget nye problemer når vi smeltet sammen grenene våre til hovedgrenen brukte vi Travis som integrasjonsverktøy. For å få satt opp Travis måtte en .yml-fil lages. Denne beskriver hvordan testene skal kjøres. Den største utfordringen med Travis var å finne en løsning for å kjøre tester for to ulike språk og to ulike steder. Løsningen fantes i dokumentasjonen til Travis og ble å kjøre flere jobber i parallell [46].

[46] TRAVIS CI, GMBH, 2021, Build Matrix, https://docs.travis-ci.com/user/build-matrix/, 20.02.21

# 5. Diskusjon

Dette kapitelet vil ta for seg de valgene vi gjorde underveis i utviklingen, og hvordan vi mener disse valgene oppfyller oppgavemålene. Vi vil ta for oss tilbakemeldingene vi fikk fra testpersoner og lærere, og forklare hvordan de påvirket vår valgprosess. I tilbakemeldingene var det nye ønsker om funksjonalitet til applikasjonen, og vi ble gjort oppmerksomme på svakheter vi selv ikke hadde tenkt på. Vi vil også sammenligne applikasjonen vår med den tidligere applikasjonen for atferdkartlegging skrevet til iOS.

## 5.1. Oppgavens mål

Målene til oppgaven var at applikasjonen skulle være rask i bruk og enkel å vedlikeholde. Vi vil i de kommende avsnittene redegjøre for hvordan valgene vi har tatt har oppnådd dette.

### 5.1.1. Bruk av teknologier

Teknologiene vi valgte skulle gjøre applikasjonen enkel å vedlikeholde. De valgte teknologiene er så langt som mulig godt kjent og mye brukt, og flere av dem undervises på UiS. Om applikasjonen faktisk er enkel å vedlikeholde vil vise seg med tiden, da teknologier stadig utvikler seg og forandrer seg, men alle valg av teknologier er gjort for å oppfylle dette målet. Applikasjonen tar i bruk moderne teknologier som er åpen kildekode og som vil være tilgjengelige i lang tid fremover. De enkleste fungerende løsningene ble valgt. Fordi vi ikke forventer at applikasjonen vil måtte kunne utrette mer i fremtiden enn det den gjør i dag mener vi at dette ikke vil bli et problem senere.

### 5.1.2. Visuell utforming

Den visuelle utformingen av applikasjonen ble gjort med tanke på at det skal være lett for brukeren å forstå hva som skal gjøres til enhver tid. Den er designet for å være effektiv, det vil si at det går minst mulig tid på å sette opp og gjennomføre et prosjekt. Kartleggingen og spørsmålsundersøkelsene skal gå raskt, slik at man får gjort mest mulig på kortest mulig tid. Følgende tiltak har blitt gjort for å få til dette:

- Designet er optimalisert for nettbrett med knapper som er store nok til at det er liten sjanse for feiltrykk.
- Alle menyer er lagt opp med så få valgmuligheter som mulig, for en mest mulig effektiv bruk av applikasjonen.
- Mapping inneholder en “hjelp”-funksjon hvor brukeren kan få opp et bilde som forklarer hvordan denne delen av applikasjonen fungerer.
- Applikasjonen er responsiv og rask.
- I mapping er det mulig å lage en egendefinert liste med favorittikoner, slik at det går ekstra fort å registrere de mest brukte hendelsene.

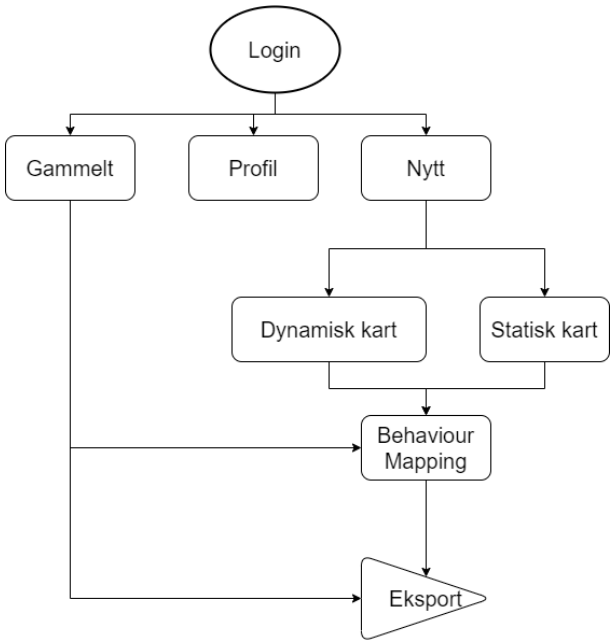
Applikasjonen oppfyller målet om å være raskt i bruk. Man kan si at et minimumskrav er at den skal være minst like rask som tilsvarende registrering på papir. Dette mener vi er nådd, da applikasjonstesterne oppgav at registrering av nye personer/hendelser går fort for seg. Det som kanskje er mest tidkrevende i en manuell kartleggingsprosess er å digitalisere alle data fra kartleggingen. Man må da legge inn én og én hendelse i et program. Applikasjonen vår digitaliserer all informasjonen fra et prosjekt med ett knappetrykk og kan eksporteres videre til et valgt filformat.

## 5.2. Endringer i applikasjonen

Etter hvert som vi kom lenger ut i utviklingsprosessen ble vi gjort klar over at vi måtte revurdere noen av valgene vi hadde tatt tidligere i prosessen. Vi vil i de kommende avsnittene gjøre rede for noen av forandringene som ble gjort som følge av dette.

### 5.2.1. Endring i applikasjonsopsettet

Tidlig i utviklingen ble det laget en tegning som viste hvordan applikasjonen skulle bli seende ut. Denne tegningen tar bare for seg atferdskartleggingen, men flere av sidene har blitt bevart med endret navn. Andre sider, som profil, har derimot blitt forkastet. Utkastet på vårt opprinnelige oppsett er vist under i Figur 34.




*Figur 34: Første utkast av hvilke sider applikasjonen skulle ha.*

Etter å ha arbeidet med prosjektet ble konstruksjonen endret, og den endelige versjonen er vist i Figur 7 i konstruksjonskapittelet.

Endringene i oppsettet, samt grunnen for dem, er som følger:

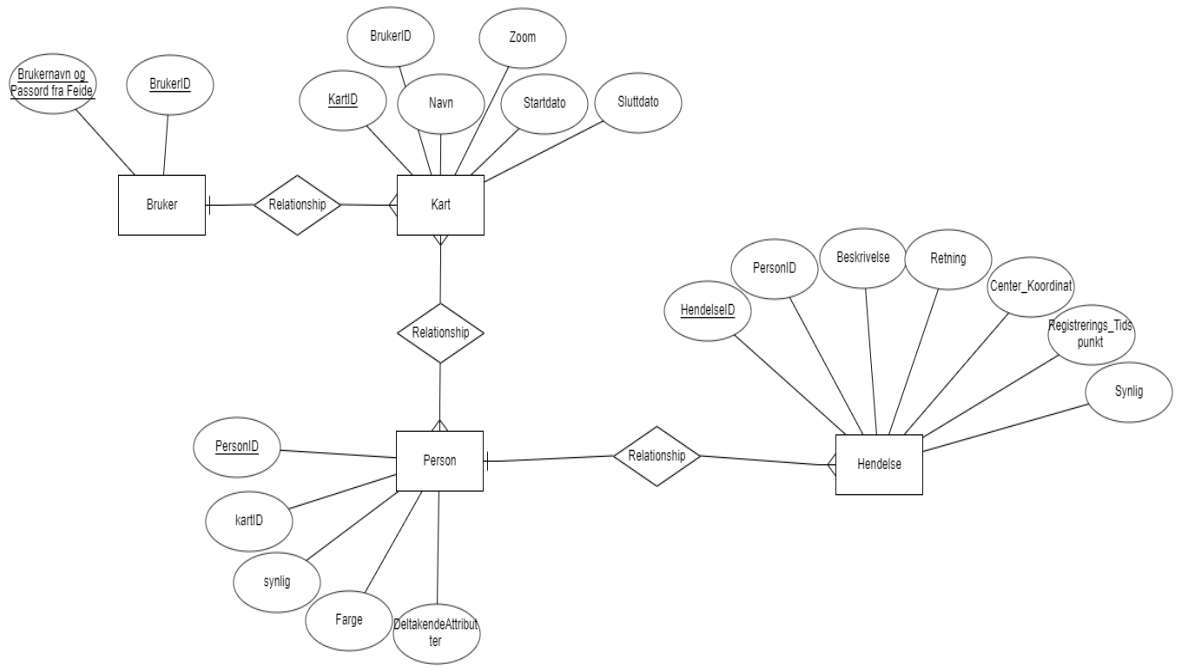
- Vi valgte bort profilsiden, fordi det er Feide som håndterer profilene til brukerne.
- Siden "gammelt", som nå heter "manageproject", har ikke noen kobling til mapping. Vi gjorde om på dette etter en tilbakemelding der det ble forklart at det ikke ville være behov for å endre et prosjekt i etterkant. På et senere tidspunkt fikk vi tilbakemelding om at dette allikevel var ønskelig om mulig. Da vi hadde fjernet denne funksjonen og koden var videreutviklet slik at dette ville blitt en mer komplisert prosess, ble ikke dette prioritert.
- Vi laget en startside der brukeren velger å lage et nytt prosjekt eller se på et som allerede eksisterer. Hvis brukeren skal lage et nytt prosjekt skal han eller hun også velge mellom geografisk spørreundersøkelse og atferdskartlegging. Vi mente det var bra ha å disse valgene på samme sted fordi det da ville bli mer brukervennlig.
- Siden "newproject" (tilsvarende siden "nytt" i Figur 34) sender brukere til siden chooseimage dersom de vil bruke det webbaserte kartet for å finne et kartutsnitt.
- Newproject viser filopplasting hvis tilstanden "fromLoadMap" har en bestemt verdi. Med dette unngikk vi å lage to startsider. Siden vil også be om koordinater for å kunne laste ned til

shape-filer senere, dersom brukeren ikke har valgt et dynamisk-kartutsnitt. Den vil også vise en tekstboks der brukeren kan fylle inn spørsmål, dersom geografisk spørreundersøkelse er valgt.

- Eksportmulighet fra mapping ble fjernet. Dette er bare mulig fra "manageProject". Dette valget ble tatt for å skape tydelighet for brukerne.
- Mapping tar nå brukeren tilbake til "startpage" etter at prosjektet er ferdig. Det ble nødvendig å gjøre det på denne måten fordi mapping ikke lengre har eksportfunksjoner og det ikke ville vært brukervennlig om brukeren måtte skrive inn sin ønskede destinasjon i url'en hele tiden.

### 5.2.2. Endring i databaseoppsettet

I designdelen av utviklingen ble det laget et utkast til hvordan databasen skulle se ut. Utkastet er vist i Figur 35under. Utkastet ble endret på og databasen fikk til slutt åtte tabeller (Figur 27) i stedet for fire. Grunnen til at vi la til flere tabeller var for å få databasen i høyest mulig normalform, samt for å ta høyde for at de geografiske spørreundersøkelsene ble implementert. Grunnen til at vi ville lage en ER-modell av databasen var i utgangspunktet for å gjøre den ryddigere, og for at alle som jobber videre med prosjektet skulle vært kjent med hvordan den er bygd opp.




*Figur 35: Første utkast av ER-modell på hvordan vi så for oss at databasen skulle være.*

## Bugfikser

Ettersom vi som utviklere fortsatt har mye å lære, er ikke koden vi skriver ufeilbarlig med en gang. Det har vist seg gjennom hele utviklingen av applikasjonen at vi har måttet fikse feil som har oppstått underveis. Som et eksempel på dette har vi en liten liste over gjennomgangen gjort under ferdigstilling av applikasjonen. Under denne gjennomgangen ble det funnet flere feil som hadde enkle og raske løsninger. Blant disse var følgende feil:

- Feil med endring av størrelse på hendelser i atferdskartlegging, når en har markert en hendelse.
- Kommentarer kommer ikke til databasen
- Feil med fjerning av hendelser som gjorde at nye hendelser lagt til etterpå ikke kom i listen som sendes til databasen.
- Kan endre størrelsen på hendelser til 0 slik at de forsvinner.
- Etter å ha kommentert kunne en legge til nytt ikon uten å trykke på nytt event.
- CSV fikk feil data i geografiske spørreundersøkelser.
- Eventlistener var ikke ryddet opp når en gitt ut fra mapping og lagde feil
- Skjermbilde av fjernet hendelse prøvde å legge til de hendelser som ikke fantes og feilet.
- Prosjekt får ikke alle data som trengs for shapefil ved opplasting av bilde.
- Feil med CSS og feil med språk som ikke blir oversatt.
- Om en forandret størrelse på vinduet til applikasjonen ble hendelser i atferdskartlegging flyttet på til feil sted.

## 5.3. En sammenligning med "*Behaviour Mapper – a tablet application for behaviour mapping*" fra 2017.

Applikasjonen fra 2017 tok som nevnt i kapittel 1.2 i bruk Swift som programmeringsspråk, og krevde kunnskap om Swift og iOS for å bli vedlikeholdt ved oppdateringer i operativsystemet. Dette er grunnen til at den ikke lenger er i bruk. Dette problemet mener vi er håndtert i våre valg av teknologier, da applikasjonen kjører i en nettleser uten å være avhengig av et spesifikt operativsystem for å fungere. Om vi sammenligner de to applikasjonene er det noe funksjonalitet vi har fått implementert som ikke var en del av versjonen fra 2017.

Den største forskjellen på applikasjonene er at vår applikasjon kan bli brukt både til atferdskartlegging og geografiske spørreundersøkelser, mens versjonen fra 2017 kun fungerte for atferdskartlegging. Applikasjonen vår dekker dermed et ekstra bruksområde, og vil være relevant for flere. Innlogging med Feide er en annen viktig funksjon som ikke var implementert i versjonen fra 2017. Her var det nok ikke behov for innlogging på samme måte ettersom den ikke var en webapplikasjon, og data ble lagret lokalt. Men i all hovedsak mener vi at det er en forbedring som gir ekstra sikkerhet for brukerne. I tillegg håndterer Feide og UiS krav i henhold til personvern. Valg av språk er også noe som ikke ble implementert i den andre versjonen. Vi fikk tilbakemelding på at det å kunne velge språk mellom norsk og engelsk ville være en stor fordel for enkelte som kan komme til å bruke applikasjonen.

Den tidligere applikasjonen tar heller ikke i bruk nøyaktige GPS-data i filene som kan håndteres med shape-filer. Dette var et av de første ønskene fra brukerne at vi skulle implementer. Vi håndterer dette med å overføre ikonkoordinatene sammen med skjermens størrelse da ikonene ble laget, for å finne rett plassering i henhold til det gitte området. Applikasjonen vår har en informasjonsknapp som beskriver hva en skal gjøre i kartleggingsfasen, noe som ikke ble implementert i den tidligere applikasjonen.

De to applikasjonene håndterer opprettelse av ikoner på to forskjellige måter. I applikasjonen fra 2017 oppretter brukeren disse selv ved at man får en rekke forskjellige ikoner å velge mellom, og så kan man legge til den beskrivelsen en selv ønsker. I vår applikasjon er ikonene lagd på forhånd og kan dermed ikke tilpasses om det skulle være behov for det. Om en skulle trenge å registrere noe som ikke er definert i ikonlisten har vi lagt til et ikon for 'annet'. Det er fordeler og ulemper med de to

måtene å gjøre dette på. I den eldre versjonen vil man bruke lenger tid når man oppretter et prosjekt, men man kan selv definere hva som skal kunne kartlegges. I vår versjon blir du bundet til å bruke de ikonene som følger med, skal en kartlegge noe annet må en bruke ikonet for annet, og legge til kommentarer for å beskrive hva som kartlegges. Vi tror allikevel at ikonene som følger med applikasjonen vår vil være egnet til å kartlegge det aller meste en skulle ha behov for, og mener derfor at vår løsning fungerer godt.

Applikasjonen fra 2017 har en funksjonalitet som gjør det mulig å opprette et nytt prosjekt med utgangspunkt i en mal. Dette gjør det mulig å gjenbruke kart og ikoner som er brukt i tidligere prosjekt. Dette er funksjonalitet som gjør det raskere å opprette et nytt prosjekt, og er spesielt nyttig når en oppretter ikonene selv. Ettersom vi gjør dette på en annen måte, er behovet for denne funksjonaliteten mindre for oss.

Vi mener at de to applikasjonene har mange likheter, men totalt at applikasjonen vi har laget har mer funksjonalitet og har brukt teknologier som gjør den til en bedre løsning på lang sikt.

## 5.4. Brukertester og Tilbakemelding

Applikasjonen ble testet av studenter i byplanlegging og to Førsteamanuensiser fra Byplanlegging ved UiS i april 2021. Applikasjonen var ikke ferdig på testtidspunktet og fungerte kun til atferdskartlegging, da geografiske spørreundersøkelser ble implementert på et senere tidspunkt. Flere av tilbakemeldingene som ble gitt var likevel relevante for denne delen av applikasjonen og ble implementert i begge prosjekttypene. Den generelle tilbakemeldingen fra de som testet applikasjonen var at den fungerte bra til Atferdskartlegging. Ifølge Førsteamanuensis Ana Llopis Alvarez var applikasjonen kjapp i bruk, og brukergrensesnittet fungerte bra.

### 5.4.1. Ønskede forbedringer

I tilbakemeldingene etter testing kom det frem at disse forbedringene var ønskelige.

- Mulighet til å endre størrelsen på ikonene til hendelser, for å håndtere ulike zoomnivå av kartet.
- Bilder som lastes opp må beholde originalt størrelsesforhold for å være relevant til bruk senere.
- Det skulle være mulig til å eksportere prosjektdata til CSV.
- Kunne legge til hendelser i en hurtigmeny i sidebaren.
- Lage en advarsel når man avslutter kartleggingen, om at prosjektet ikke vil kunne endres i ettertid.
- Mulighet for å legge til kommentarer til hendelser.
- Mulighet til å slette hendelser underveis i kartlegging.
- Skille shapefilene som inneholdt informasjon om prosjektet i flere lag for å forbedre bruksområdet.
- Legge til en funksjon for å endre språk mellom norsk og engelsk.
- Hjelpeknapp i mapping som beskriver hva de forskjellige valgene gjør i atferdskartleggingen og geografiske spørreundersøkelser.
- Lage mulighet for å zoome inne i mapping.
- Legge til en mulighet for å endre på prosjekt etter at de er ferdige.
- Lage ikoner for grupper av mennesker.
- Å ha et ikon som representerer “annet”, slik at brukeren kan registrere det som ikke passer inn i de andre eksisterende ikonene.

De fleste av disse ønskene ble implementert i applikasjonen. Noe ble ikke prioritert, og er beskrevet i kapitel 5.5.

## 5.5. Potensielle forbedringer og kjente svakheter

Applikasjonen fungerer til det den er ment å gjøre og inneholder mye funksjonalitet. Allikevel er det funksjonalitet vi ikke fikk laget innen tidsfristen, og forbedringer som ikke er implementert. Disse er som følger:

### Konstant forhold mellom høyde og breddeforholdet

Vi ble forklart at forholdet mellom høyden og bredden på opplastede bilder skulle være konstant. Vi var usikre på om dette også gjaldt dynamiske bilder.

Etter diskusjon og epostkorrespondanse kom utviklerne frem til at et dynamisk kart burde fylle hele skjermen, da vi mente dette så bedre ut og hadde skrevet funksjoner som sikret at ikonene kom på rett plass i shapefilene tross forvrengning av kartet.

Dersom vi feilbedømte, og det skulle være et statisk forhold her også, ligger css-en som vil fikse det inne i koden allerede. Da er det bare å forandre bredden til auto i ".map-image" og "#map-image" til auto slik Figur 36 viser.

```
.map-image {
    /* height: 100vh;
    width: auto;*/
    height: 100vh;
    width: calc(100vw - 200px);
    min-height: 300px;
    order: 3;
}

#map-image {
    /* height: 100vh;
    width: auto;*/
    height: 100vh;
    width: calc(100vw - 200px);
    min-height: 300px;
    order: 3;
}
```

*Figur 36 Koden som er kommentert ut er det som sørger for å beholde høyde og breddeforholdet.*

### To get-apier som skulle vært POST

Det er to APIer i backend, "updateproject" og "updateicon", som skulle blitt gjort om fra GET til POST, fordi de oppdaterer databasen.

### Zoom i mapping

Det var tidlig et ønske om å ha mulighet til å zoome inn eller ut på kartet når enn drev med kartlegging. Dette ville gjort applikasjonen mer fleksibel, og det ville gjort det mulig å kartlegge større områder ved behov. Etter en vurdering ble det planlagt to måter å oppnå dette på. Begge ville vært mulig å implementere for webbaserte kart, men ingen ville fungert for kart som brukerne lastet opp selv.

Den første var at vi laget ikoner inne i det dynamiske kartet. For å implementere dette måtte imidlertid applikasjonen ha håndtert prosjekter hvor en laster opp bilder selv og prosjekter der en bruker det webbaserte kartet på to helt forskjellige måter. Dette skyldes at funksjonene som skulle

interagere med det webbaserte kartet måtte vært forskjellige fra funksjonene som skulle brukes til det statiske bildet. Det ble derfor prioritert å bruke en løsning som fungerer for begge prosjekttypene over en løsning som tillot zooming. Løsningen var å gjøre det dynamiske kartet om til et statisk bilde som behandles i mapping på samme måte som opplastede bilder.

En alternativ løsning for å få til en begrenset zoomfunksjonalitet kunne vært å hente et større område enn det brukeren valgte. Applikasjonen kunne da begynt med å vise utsnittet brukeren valgte, og så kunne brukeren ha zoomet ut for vise hele utsnittet som ble hentet fra kartet. Dette ville tillatt noe zooming uten å kreve omskrivning av eksisterende funksjoner, men ville heller ikke fungert for selvopplastede kart.

### Beholde tilstand ved sideoppdatering

Applikasjonen beholder ikke tilstandsobjekter som sendes mellom komponenter ved en sideoppdatering. Det gjør at det vil oppstå en feil dersom siden oppdateres, og at brukerne må gå tilbake til startsiden eller login, da disse sidene ikke får inn tilstandsobjekter fra andre komponenter. For å få ordnet opp i dette måtte applikasjonen lagret informasjonen om tilstand-objektene lokalt, og sjekke om det finnes noe midlertidig lagret informasjon ved en sideoppdatering. Dette er en potensiell ganske stor mangel, men på grunn av at feilen først kom mot slutten av prosjektet fikk vi ikke rettet opp i dette.

Vi hadde tidligere en rekke funksjoner som ville laste opp prosjektets tidligere lagde hendelser fra databasen ved sideoppdatering, slik at dette arbeidet ikke ville forsvinne fra fremsiden. Disse funksjonene er nå ubrukelige, og ble ikke oppdatert med resten av prosjektet etter at vi fikset feideinnloggingen. Denne funksjonsrekken kan også brukes til å feste hendelser på kartet i mapping dersom brukerne vil gå direkte dit fra manageproject. I componentDidMount i mapping hentet vi prosjektets hendelser fra databasen og sendte dem inn i funksjonen loadFormerEvents. Her ble det iterert igjennom hendelsene, plassert på kartet, rotert og lagt inn i tilstandslisten iconObjects slik at vi kan knytte dem opp mot deres tilsvarende databaseobjekter.

### Hashe brukerID i databasen

For å sikre brukeridentifikasjonen bedre, kunne det blitt lagd en hash-funksjon for brukerens ID. Ettersom applikasjonen ikke henter passordet til brukeren utgjør ikke dette en stor sikkerhetsrisiko, men det er allikevel en ønsket funksjonalitet.

### Sende shape- og CSV-filer på epost

I tillegg til å lagre shape- og CSV-filene lokalt på enheten kunne det bli lagd en funksjon som sendte filer direkte til eposten til brukeren som er pålogget. Filene behandles gjerne på flere andre enheter i ettertid, dette hadde derfor vært en funksjon som hadde økt brukervennligheten til applikasjonen.

### La brukerne laste opp egne ikoner

For å gjøre applikasjonen mer fleksibel kunne det vært lagt til en måte for brukere å laste opp og opprette egendefinerte hendelser med både bilde og beskrivelse. Etter tilbakemelding fra testbrukerne har det blitt lagt til et “annet”-ikon som øker fleksibiliteten, men å kunne laste opp egne ikoner ville gjort det enda mer fleksibelt.

### Refakturering

Mapping er applikasjonens hovedside, og inneholder mer kode enn de andre komponentene i applikasjonen. For å gjøre koden mer oversiktlig kunne noen kodesegmenter vært flyttet ut til egne komponenter, slik at en ville fått flere og mindre filer. All kode for sidebaren i mapping kunne for eksempel vært flyttet til en egen fil. Dette ville gjort koden lettere å forstå og jobbe med for andre.

Dette ble ikke prioritert. Et forsøk på å gjøre hendelsene plassert på kartet om til komponenter ble gjort, men ikke fullført.

### Slette tegnede elementer i geografiske spørreundersøkelser

Etter å ha tegnet opp en linje eller et område er det vanskelig å gjøre noe for å fjerne det igjen. Det kunne likevel ha blitt løst ved å lagre alle koordinater og typer figurer det var i et tilstandsobjekt, som så ble brukt for å tegne alle elementene på ny igjen når et ble slettet. Denne løsningen ville vært tidkrevende å få implementert, og ble derfor ikke prioritert.

### Inoformasjonskapsler slettes ikke ved oppdatering

Dersom en bruker har vært pålogget applikasjonen før det kommer en oppdatering og ikke har fått trykket på logout før oppdateringen er lagt ut, vil ikke applikasjonen tillate de noen bruk. Dette er på grunn av informasjonskapsler som ikke slettes, og secret_key i Flask som er et tilfeldig generert tall som genereres hver gang den starter. Dersom kapslene har et annet tall enn det nåværende så vil applikasjonen feile. Den midlertidige løsningen på dette i dag er å manuelt fjerne alle informasjonskapsler knyttet til applikasjonen, eller å bruke privat modus i nettleseren. Under utvikling og testing har dette vært et problem, men når applikasjonen er i produksjon vil det ikke være et like stort problem ettersom den da kjører stabilt på samme secret_key.

### Punkter i geografiske spørreundersøkelser

Når en tegner punkter med applikasjonen slik den er i dag, må en dobbeltklikke for at punktet skal dukke opp. Dersom en kartlegger på en datamaskin uten touch må en også flytte noe på musepekeren for at punktet skal dukke opp. Dette gjør det vanskeligere og mindre brukervennlig å legge ut punkter. Siden applikasjonen i hovedsak er tenkt til bruk på touch-enheter ble det ikke brukt tid på å forbedre dette. Dersom skjermstørrelsen forandres, kommer heller ikke punktene på rett sted, noe som skyldtes en endring gjort på slutten for å ikke endre størrelsesforholdet på kartet.

## Testing

Med unntak av Travis brukte vi de testrammeverkene som var inkludert med create-react-app, og de som ble anbefalt av Flask. Det ble vurdert å erstatte disse testrammeverkene med noe annet, eller å videreutvikle de medfølgende testene, men dette ble valgt bort grunnet manglende tid og erfaring. Hvor velutviklede tester man har vil ha mye å si for hvor trygg man kan være på at applikasjonen fungerer som den skal når den blir oppdatert.
Testene laget til applikasjonen var ikke gode nok, og derfor kom det flere feil som burde vært fanget opp av tester. Blant annet feilen nevnt i avsnittet over, om punkter i geografiske spørreundersøkelser.

### Tilpasse applikasjonen til mobil

Applikasjonen fungerer dårlig på mobil. Det er fordi det ikke ble prioritert å lage et eget design for smarttelefoner. Dette valget ble tatt sammen med veileder ettersom applikasjonen i hovedsak skal brukes på nettbrett. Vi mener allikevel at dette er en svakhet ettersom applikasjonen ville blitt mer fleksibel om den også var mobiltilpasset. De aller fleste har i dag smarttelefoner, og selv om registreringen ville blitt mer knotete på en mindre skjerm ville det økt applikasjonens tilgjengelighet.

### Endre hjørnekoordinater i manageproject

Per dags dato blir brukere bedt om å gi koordinater på opplastede bilder i newproject, men har ikke mulighet å forandre på disse senere. Koordinater fylles inn automatisk i dynamiske kart. Dersom brukeren skriver inn feil koordinater i newproject vill punktene i shape-filen plasseres feil. Det skulle derfor vært en funksjon som lar brukeren forandre hjørnekoordinatene i manageproject. For å unngå

problemer anbefaler vi at brukerne starter et prosjekt i arcGIS eller QGIS og laster opp bildet der, slik at de kan finne hjørnekoordinatene mens/før de er i newproject.

#### Sletting før innsending til databasen

Dersom et ikon slettes fra frontend før den sendes inn til databasen vil den ikke slettes fra databasen. Denne feilen skulle blitt prioritert, men ble oppdaget for sent til å løse. Dette vill føre til at feil ikoner blir slettet dersom brukeren prøver å slette ikoner som legges til ett at denne feilen blir begått. For å unngå denne feilen anbefaler vi at brukere ikke prøver å slette det siste ikonet de la inn, eller at de trykker på "Add Event" og velger et ikon før de sletter det siste ikonet de la tilpå kartet. For å gjøre det enkelt vill vi anbefale brukere å velge et nytt ikon i "Add Event" før de sletter et gammelt ikon.

## 5.6. Utfordringer

Vi vil her forklare litt om de mest krevende prosessene under utviklingen av applikasjonen, og utfordringer som oppstod underveis.

### 5.6.1. Shape-filer

Det har vært utfordrende å få overført den geografiske informasjonen til shape-filformat. Dette skyldes delvis at det har vært rent tekniske utfordringer med å lage kode som samler og sorter informasjon på en passende måte, og delvis en antagelse om at applikasjonen kunne slette mapper på UiS sin tjener.

Det var først utfordrende å få til kode som fungerte greit grunnet manglene erfaring med shape-filer. Det første utkastet av koden hentet alle punktene og plasserte dem på et kart. Etter tilbakemelding om at vi skulle ha flere lag, ett for hvert relevante objekt, måtte vi sortere informasjonen slik at den var lett håndterbar. Dette gjorde vi ved hjelp av dictionaries. Mappene og shape-filene ble i første omgang laget manuelt, noe som var svært tidkrevende. Det første utkastet av koden vi hadde gikk inn i disse mappene og modifiserte filene der før de ble zippet og sendt videre til brukeren. Filene ble 'rensket' ved å lage et nytt lag på de shape-filene som ikke var i bruk. Denne metoden fungerte ikke da koden ble distribuert på serveren, fordi tomme lag av en ukjent grunn ikke erstattet de utfylte lagene i distribusjonen. Det ble derfor bestemt at zip_files funksjonen skulle modifiseres slik at den bare sendte de relevante mappene.

Zip_file funksjonen ble ikke forandret med en gang, noe som skyldtes at vi fikk sendt de relevante mappene på en annen måte. Vi opprettet filer og mapper når det var behov for dem og slettet dem senere. Grunnet at vi ikke hadde slettetillatelse på serveren fungerte ikke dette etter distribusjon. Det hadde oppstått en misforståelse angående de relevante tillatelsene, og det ble antatt at programmet hadde eller ville få rettigheter til å slette mapper. Koden som laget shape-filer til geografiske spørreundersøkelser ble skrevet fra bunn av med tanke på oppretting og sletting av foldere, mens koden som laget shape-filer til atferdskartlegging ble omskrevet slik at også den gjorde det. Fordi vi ikke hadde slettetillatelse på apacheserveren laget koden flere mapper med samme navn som den ikke kunne skille imellom, og som i tillegg tok opp mye plass.

Vi fikk skrevet om koden slik at den kun modifiserte eksisterende filer og mapper, og skrev om funksjonene slik at bare de mappene med endret innhold ble tatt med i ZIP-filen. I tilfelle prosjektet var en geografisk spørreundersøkelse ville koden iterere igjennom Intervjuobjektene, for så å iterere igjennom de relevante shape-fil mappene, modifisere innholdet og navn og så å legge dem til i ZIP-filen. ZIP-filen kan derfor inneholde flere mapper enn det som egentlig eksisterer på serveren, uten at det oppstår et problem verken hos brukeren eller hos serveren.

### 5.6.2. Implementasjon av Feide

Implementasjonen av Feide består av en administrativ del og en teknisk del. Siden UiS allerede var tjenesteleverandør til Feide, ble den administrative delen gjort gjennom dem. Det ble korrespondert med it-avdelingen på UiS på for å få registrert nødvendig informasjon. For å få til den tekniske delen fulgte vi guiden til Feide [47]. Det første steget i guiden var å få tilgang til kundeportalen. Siden UiS ble benyttet som tjenesteleverandør fikk vi tilgang til kundeportalen gjennom dem. Administrasjonstilgang til applikasjonen ordnet it-avdelingen ved UiS [48]. Denne tilgangen gir mulighet til å se informasjonen som trengs for å sette opp OIDC i Dataporten Dashbord [49]. Fordi applikasjonen skal videreutvikles har også veileder fått administratortilgang til denne siden.

Å implementere Feide var krevende ettersom det ble brukt teknologier som gruppen ikke hadde kunnskap om fra før. Feide hadde heller ikke kodeeksempler for den tekniske implementeringen av innloggingen, men gav heller en generell instruks.

### 5.6.3. Distribusjon til Unix-miljøet/Apacheserver

For å få til distribusjonen til Apacheserveren på UiS måtte vi gå via Universitetets kontaktpersonen på UNIX-miljøet. Samtidig måtte backend og frontend klargjøres for å lage pakker som enkelt kunne distribueres på Apacheserveren. Prosessen til en ferdig distribuert applikasjon beskrives her.

#### Klargjøring til distribusjon

Før applikasjonen kunne distribueres måtte det lages pakker for både frontend og backend. I frontend var det klargjort fra starten av, og det var bare en enkel kommando som måtte kjøres. For backend derimot måtte store deler av applikasjonen skrives om for å enklere kunne lage en pakke som kunne distribueres på tjeneren. Før vi kunne begynne distribusjonen måtte vi få en bruker på Unix-miljøet ved UiS, og derfra fikk vi en egen hjemmeside som skulle lenkes opp imot en mappe på hjemmeområdet til brukeren.

#### Frontend

For å distribuere frontend som er skrevet i React og laget med pakkebehandleren NPM, er alt en må gjøre å kjøre kommandoen “npm run build”. Etter denne kommandoen er kjørt bygges det en komprimert versjon av applikasjonen som kan legges ut på webtjeneren en har valgt.
Endringen for å få frontend til å sende forespørsler til backend til den korrekte URL-en var å legge til en global variabel. I utvikling er det en egen innstilling en kan gjøre i pakkebehandleren som bestemmer hvor forespørsler til backend skal sendes. For å få det til å fungere etter distribusjon la vi til en global variabel på starten av alle forespørsler som går til backend. (Figur 37)

```
// window.backend_url = "http://localhost:5000/behaviourmapper/"
window.backend_url = "https://behaviourmapper.ux.uis.no/"
```

*Figur 37 Den globale variabelen som er lagt til på starten av alle forespørlser til backend.*

[47] Uninett, 2021, Getting started , https://docs.feide.no/service_providers/getting_started/index.html, 17.04.21
[48] Uninett, 2021, Arbeidsfordeling administrator og utviklere, https://docs.feide.no/service_providers/manage/openid_connect/registration.html, 17.04.21
[49] Uninett, 2021, Dataporten Dashboard, https://dashboard.dataporten.no/, 17.04.21

Totalt sett var det gode ferdige løsninger som fulgte med den pakken vi hadde valgt i starten av utviklingen. Det gjorde distribusjonen av frontend enkel. Build-pakken som ble laget med npm run build måtte legges inn i den korrekte mappen på serveren som ble lest og er lenket opp imot vår url.

### Backend

Da vi skulle distribuere backend til applikasjonen viste det seg at en del forandringer måtte gjøres for å få det til å fungere. Endringen var å lage en build-pakke i en enkelt fil som så installeres i et virtuelt Python-miljø på serveren istedenfor å kopiere alle filene over manuelt. Pakkefilen som lages bruker Wheel-formatet som er nåværende standard for Python-distribusjon. [40] Prosessen for å komme frem til den nåværende løsningen er beskrevet i delkapittelet Prosessen til distribusjonsløsningen53.

Applikasjonen benytter seg av en Apachetjener og mod_wsgi for å distribueres. Mod_wsgi er en Apache HTTP-tjenermodul som gir et grensesnitt for å tjene Python-baserte webapplikasjoner under Apache. Eieren av Apacheserveren hjalp også til med å få laget loggfiler for feil og tilganger på apachetjeneren. Mod_wsgi modulen er satt opp på apachetjeneren. Det som må gjøres i koden er å lage en wsgi-fil som forteller apachetjener hvordan applikasjonen startes.

```
from behaviourmapper import create_app
application = create_app()
```

*Figur 38 Innholdet i filen som styrer hvordan apachetjeneren starter backend.*

Figur 38 viser hva wsgi-filen inneholder. Denne filen knyttes opp til Apachetjeneren via en konfigurasjonsfil, konfigurasjonsfilen i sin helhet kan ses i kapittel 7.4.

### Prosessen til distribusjonsløsningen

Når backend av applikasjonen ble distribuert måtte det gjøres en del endringer for å finne en fungerende løsning. Løsningen som er beskrevet i delkapittelet Backend tok53 mye tid å komme frem til, og ble løst etter en lang epostkorrespondanse med kontaktpersonen ved UiS.

Dette var utfordrende da vi ikke driftet tjeneren selv og ikke kunne gjøre endringer fritt. Vi hadde heller ikke tidligere erfaringer med distribusjon. Ettersom vi ikke driftet tjeneren selv ble det mye prøving og feiling, etter hvert som vi fikk kontakt med kontaktpersonen ved UiS kom vi inn på rett spor. Det første som ble prøvd ut var løsningen fra guiden til Flask-dokumentasjonen [50]. For å kunne bruke denne måtte vi skrive om flere deler av backend for å kunne bruke de verktøyene og løsningene vist der, blant annet wheel.  Selv om den endelige løsningen vist i guiden ikke fungerte var det å skrive om applikasjonen veldig nyttig, ettersom applikasjonen ble enklere å distribuere. Istedenfor å måtte kopiere alle filer manuelt, kunne vi nå lage en pakke som ble installert med pip.

Kontaktpersonen ved UiS ba oss om å bruke mod_wsgi for å distribuere backend av applikasjonen, men for å få det til å fungere må en konfigurasjonsfil opprettes og konfigureres korrekt på tjeneren. Den lange epostkorrespondansen ble hovedsakelig til her fordi det ble mange tester frem og tilbake for å finne en løsning som fungerte. Løsningen ble funnet sammen basert på tidligere applikasjoner distribuert ved UiS og kilder på internett som beskrev hvordan konfigurasjonen skulle være.

### Pakkedistribusjon

Når en skal gjøre klar pakkene fra utvikling til distribusjon måtte en endre på flere variabler som

[50] Pallets, 2020, Tutorial, https://flask.palletsprojects.com/en/2.0.x/tutorial/, 14.05.21

definerer forskjellige URLer. En ønsket forbedring fra gruppens side var å få forenkle det slik at distribusjonen var enklere og raskere.

Løsningen ble at en i backend sjekker om det finnes en miljøvariabel som sier at applikasjonen kjører i utviklingsmodus, og basert på det sette de gitte variablene slik de skal være. Med denne løsningen er det bare versjonsnummer som må endres når applikasjonen skal distribueres. I frontend må en endre en variabel for å sende til korrekt url i backend i tillegg til å endre versjonsnummer. Løsningen gjorde at nødvendige endringer for å distribuere ble redusert fra ni til tre. Noe som kan være et problem med endringen er at de som utvikler må vite at de har den riktige miljøvariabelen satt i backend. Om dette ikke blir gjort riktig vil applikasjonen sette feil variabler for utvikling. Men løsningen antas likevel å være en forbedring.

### 5.6.4. Tilgang via universitetets VPN

Når applikasjonen skulle testes etter distribusjon, oppdaget vi et uforutsett problem. Applikasjonen fungerte ikke for de som satt på hjemmekontor med maskiner fra UiS da de ikke fikk tilgang til applikasjonen. Etter å ha tatt kontakt med både eieren av serveren og IT-avdelingen ved UiS ble det åpnet for at unix-miljøet til UiS også ble tilgjengelig via VPN-løsningen UiS benytter seg av. Ettersom dette ikke var noe vi hadde kontroll på var dette en helt uforutsett utfordring som gjorde at testingen av applikasjonen gikk tregere.

# 6. Refleksjon og konklusjon

Applikasjonen kan utføre atferdskartlegging og geografiske spørreundersøkelser, og vi mener den tilfredsstiller kravene som var gitt. Den er enklere i bruk enn kartlegging på papir, og har tilnærmet like mye funksjonalitet. Målene for applikasjonen, at den skal være enkel i bruk og vedlikehold og rask i registrering av nye brukere, er møtt.

Vi var heldige som fikk flere testbrukere til applikasjonen. Dette gjorde at vi fikk gjort forbedringer vi ikke hadde tenkt på selv. Helhetsinntrykket hos testerne var ellers positivt, noe som tyder på at applikasjonen fungerte bra allerede på testtidspunktet.

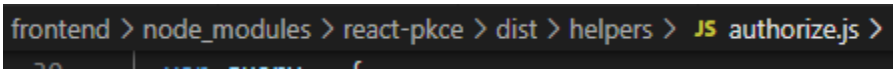
Selv om applikasjonen fortsatt har mangler, synes vi at funksjonaliteten som er implementert er tilfredsstillende. Vi er også fornøyd med mengden funksjonalitet applikasjonen har, ettersom det er første gang vi har skrevet en fullverdig webapplikasjon og det var begrenset med tid. Vi har benyttet oss av rammeverk og sikkerhetsløsninger som er mye brukt og som derfor er trygge løsninger. Dersom vi kunne jobbet videre ville både kodebasen til applikasjonen blitt bedre og de manglene som er ville blitt fjernet.

Gruppen har lært av å jobbe med hverandre, og fått mer erfaring med å jobbe på et prosjekt sammen med andre. Dette er veldig nyttig erfaring å ha med seg videre. Gruppen har prøvd å utnytte gruppemedlemmenes styrker og svakheter, slik at vi jobber best mulig sammen. Samtidig kan det være mer jobb for noen som skal videreutvikle applikasjonen. Ettersom vi var tre ulike personer som jobbet med å bygge kodebasen, vil det være ulik praksis på hvordan kode er skrevet.

Vi merket alle at flyten og effektiviteten når vi jobbet med oppgaven ble bedre på slutten av oppgaveperioden, ettersom alle da hadde satt seg inn i hvordan applikasjonen var skrevet og hadde fått bedre kjennskap til teknologiene applikasjonen benytter seg av. Det gjorde at det ble enklere å løse problemer og legge til ønskede endringer. Applikasjonen er sammensatt av både enkle og komplekse løsninger, og å utvikle disse er noe som har vært veldig lærerikt. Vi har alle utviklet oss og fått erfaring med å jobbe med et større prosjekt, noe som er en nyttig erfaring å ta med videre.

# 7. Appendiks

## 7.1. Endre på authorize.js i react-pkce




*Figur 39 Filen som måtte endres*

Det som måtte endres lå i filen vist i Figur 39 på linje 42. Endringen som måtte gjøres var å få pakken til å gå til /authorization istedenfor /authorize, fordi "/authorization" er endepunktet til feide. Når det er gjort og det lages en byggpakke av frontend følger disse endringene med.

## 7.2. Hvordan distribueringen gjennomføres

Det er noen endringer som må gjøres i koden før distribuering. I index.js må en bytte mellom hvilken som er kommentert ut av linje 13 og 14. I tillegg må versjonsnummer oppdateres i package.json og setup.py Før en sender filer til distribusjon må begge byggpakker lages for både frontend og backend. Disse lages ved bruk av følgende kommandoer inne i mappen for frontend og backend.

Backend: python setup.py bdist_wheel

Frontend: npm run build

Gjennomføringen av distribuering er en liste av kommandoer som kjøres og brukes i samsvar med GitHub. Både Wheel-filen for backend og byggmappen til frontend lastes opp i et GitHubrepository som er laget kun for distribusjon. Frontend kopieres inn i den korrekte mappen og så fungerer den. Backend måtte installeres med pip inne i det virtuelle miljøet for å bli korrekt distribuert.

Kommandoene som brukes for å kjøre en distribusjon til frontend er som følger:

```
cd public_html/
rm -r asset-manifest.json index.html manifest.json robots.txt static/
cp -r ../githubrepository/build/* .
```

For backend er dette kommandoene som kjøres:

```
source public_html/venv/bin/activate
pip install githubrepository\dist\navn_på_nåværende_versjon.whl
```

For at endringer i backend skal skje umiddelbart må en redigere på .wsgi filen til applikasjonen. WSGI-filen er filen som starter flask på Apache-tjeneren. Det er fordi tjeneren ikke sjekker etter endringer i de andre filene like raskt som når det er endringer i den filen.

## 7.3. En kort innføring i React

I frontend har vi brukt Javascript-biblioteket React. Siden React på nåværende tidspunkt ikke undervises ved UiS, kommer det her en kort innføring i biblioteket og hvordan et React-prosjekt er bygget opp.

React er et Javascript-bibliotek utviklet og vedlikeholdt av Facebook. Et prosjekt skrevet i React skiller seg ut fra et vanlig Javascript-prosjekt på flere området, selv om programmeringsspråket er det samme. Et React-prosjekt er bygget opp av komponenter. En komponent er kode som er ment å bli brukt om igjen flere ganger. For at ikke alle komponentene skal se like ut kan komponenter ta inn "props". Dette kan for eksempel være tekst eller et bilde. Dette fungerer på samme måte som hvis

en for eksempel spesifiserer URL-en til en link i en HTML-fil, eller kilden til et bilde i en <img>-tag. På samme måte kan alle komponenter ta inn egne props som gjør den forskjellig fra andre komponenter av samme type.

### 7.3.1. To typer komponenter

I React kan en skrive to forskjellige typer komponenter; funksjonsbaserte komponenter og klassebaserte komponenter. De to ulike typene komponenter hadde tidligere ulik funksjonalitet, noe som gjorde de brukende i forskjellige sammenhenger. Klassebaserte komponenter hadde tidligere mer funksjonalitet enn funksjonsbaserte komponenter, da de var alene om å ha tilgang til state-objekter, heretter referert til som tilstandsobjekter. Et tilstandsobjekt er et objekt som kan forandres, og hver gang det skjer en endring i et tilstandsobjekt vil React oppdatere siden for å reflektere endringen. I senere tid har også funksjonbaserte komponenter fått tilgang til tilstandsobjekt, og mange foretrekker å bruke funksjonbaserte komponenter foran klassebaserte komponenter da en funksjonbasert komponent trenger mindre kode og dermed er lettere å lese [51].

### 7.3.2. Create React App og App.js

For å sette opp applikasjonen vår brukte vi "Create React App" som lar deg sette opp en fungerende applikasjon med én enkel kommando. Kommandoen lager flere filer som fungerer sammen. Applikasjonen inneholder én HTML-fil hvor all kode fra React sendes. Applikasjonens startpunkt er filen App.js. Etter hvert som applikasjonen blir større vil alle komponenter samles her for så å bli til én komponent, som igjen sendes videre. App.js vil derfor importere alle de andre sidene i applikasjonen, og React-ruteren bestemmer hva som vises og når. Flere komponenter kan vises samtidig.

### 7.3.3. JSX

JSX står for JavaScript XML og er en utvidelse til JavaScript. JSX er på overflaten veldig likt HTML, og vil se kjent ut for en som har jobbet med HTML tidligere. JSX gjør det mulig å samle HTML-lignende syntaks og JavaScript i en og samme fil, noe som gjør det mulig å samle kode som hører sammen på ett sted. En bruker gjerne JSX for å definere brukergrensesnittet. JSX kan ikke leses av en nettleser slik som vanlig HTML kan, men produserer React-elementer. [52]

### 7.3.4. Hooks og lifecycle-metoder

Lifecycle-metoder er tilgjengelige i klassekomponenter i React. Dette er metoder som kjører av seg selv på visse tidspunkt i en komponents livssyklus. I livssyklusen til en komponent finnes det tre faser, som er initialisering, oppdatering og demontering. Oppdateringer kan referere til forandringer i komponentens tilstand-objekt. Det enkleste eksemplet for bruk av en lifecycle-metode er metoden componentDidMount(), som kjøres med en gang komponenten blir tatt i bruk. Dette kan for eksempel være et naturlig sted å hente data fra databasen. Hooks er en nyere type funksjonalitet som er tilgengelig i funksjonskomponenter. Hooks gir funksjonskomponenter tilgang til de samme funksjonene som Lifecycle-metodene gir. Det er også hooks som gjør det mulig å bruke tilstandsobjekter i funksjonskomponenter.

---

[51] Milan Parmar. 2021, React: Class VS Functional Components https://medium.com/star-gazers/react-class-vs-functional-components-a49383f65f0e Lastet ned 26.04.2021

[52] Introducing JSX, https://reactjs.org/docs/introducing-jsx.html, lastet ned 14.05.2021

## 7.4. Eksterne filer til appendiks.

Koden til hele applikasjonen ligger her:

https://github.com/Oddvar-N-O/BehaviourMapperV2Levering

**Nødvendige filer for å få kobling mot Feide til å fungere.**

Client_secrets.json (Veileder har kopi)

.env (Veileder har kopi)

**Den ferdige konfigurasjonen til Apache-tjeneren**

Apacheconf.htaccess

```
<IfModule mod_ssl.c>
<VirtualHost *:443>
    ServerName behaviourmapper.ux.uis.no
    ErrorLog     /home/map/w3/behaviourmapper.ux.uis.no/error.log
    TransferLog  /home/map/w3/behaviourmapper.ux.uis.no/access.log
     <Directory /home/stud/oddvarno/public_html/venv/lib/python3.6/site-packages/behaviourmapper/static/>
        Require all granted
    </Directory>
    <Directory  /home/stud/oddvarno/public_html/venv/lib/python3.6/site-packages/behaviourmapper>
        WSGIApplicationGroup %{GLOBAL}
        <Files behaviourmapper.wsgi>
            Require all granted
        </Files>
    </Directory>
    WSGIDaemonProcess behaviourmapper.ux.uis.no python-home=/home/stud/oddvarno/public_html/venv
    WSGIProcessGroup behaviourmapper.ux.uis.no
    WSGIScriptAlias / /home/stud/oddvarno/public_html/venv/lib/python3.6/site-packages/behaviourmapper/behaviourmapper.wsgi

    SSLCertificateFile /etc/letsencrypt/live/behaviourmapper.ux.uis.no/fullchain.pem
    SSLCertificateKeyFile /etc/letsencrypt/live/behaviourmapper.ux.uis.no/privkey.pem
    Include /etc/letsencrypt/options-ssl-apache.conf
</VirtualHost>
</IfModule>
```

*Figur 40 Konfigurasjonen gjort på apachetjeneren.*

# 8. Referanseliste


[1] Madsen, Å., 2015. Kjøpesentres påvirkning på sentrum - En casestudie av Svortland sentrum og Hellvik Senteret (s 129), Stavanger: Universitet i Stavanger.

[2] Madsen, Å., 2015. Kjøpesentres påvirkning på sentrum - En casestudie av Svortland sentrum og Hellvik Senteret (s 18), Stavanger: Universitet i Stavanger.

[3] Jonvik, Merete & Lindland, Kristiane & Tvedt, Helge L. & Müller-Eie, Daniela & Melberg, Kjersti (2018) Hillevåg – En sosiokulturell stedsanalyse. IRIS. Rapport under produksjon.

[4] Vinugayathri. 2020, 5 JavaScript Alternatives for Front End Development, https://www.clariontech.com/blog/5-javascript-alternatives-for-front-end-development, Lastet ned 16.04.2021

[5] @Fyrd and @Lensco, 2021 , CANIUSE, https://caniuse.com/?search=JavaScript , 6.03.21

[6] Evan You. "*The Progressive JavaScript Framework*", 2014. https://vuejs.org/ Lastet ned den 28.04.2021.

[7] Petersen, John V., "*Legal Notes: What's the Deal with ReactJS's Licensing Scheme?*". Publisert i *CODE Magazine* Januar/Februar 2017, Sist oppdatert: February 19, 2019 https://www.codemag.com/article/1701041/ Lastet ned den 28.04.2021

[8] JavaTpoint, 2011-2018, Pros and Cons of ReactJS, https://www.javatpoint.com/pros-and-cons-of-react, 16.04.21

[9] Pawar, Gauri, 2020, The Good, the Bad, & the Not So Bad About React.js Development https://eluminoustechnologies.com/blog/pros-and-cons-of-react-js-development/, 23.04.21

[10] Nowak, Maja, 2020, Vue vs React in 2021: Which Framework to Choose and When, https://www.monterail.com/blog/vue-vs-react-2021, 23.04.21

[11] Johnson, Jonathan, 2020, Java vs Go: What's The Difference?, https://www.bmc.com/blogs/go-vs-java/, 21.04.21

[12] crowd sourced, 2021, The Computer Language Benchmarks Game, https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/go.html, 21.04.21

[13] crowd sourced, 2021, The Computer Language Benchmarks Game, https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/go-python3.html, 21.04.21

[14] Cox, Joseph, 2015, Why You Don't Roll Your Own Crypto, https://www.vice.com/en/article/wnx8nq/why-you-dont-roll-your-own-crypto, 07.05.21

[15] Rossen, Eirik, 2019, API i Store norske leksikon på snl.no, https://snl.no/API, 26.04.21

[16] Crosley, Timothy Edmund, 2021, Embrace the APIs of the future, https://www.hug.rest/, 14.04.21

[17] Refsnes Data, 1999-2021, SQL Data Types for MySQL, SQL Server, and MS Access , https://www.w3schools.com/sql/sql_datatypes.asp, 20.04.21

[18] SQLite.org, 2021, Datatypes In SQLite Version 3, https://www.sqlite.org/datatype3.html, 20.04.21

[19] Branson, Tony, 2017, THE 5 BEST REASONS TO CHOOSE MYSQL – AND ITS 5 BIGGEST CHALLENGES, https://dataconomy.com/2017/04/5-reasons-challenges-mysql/, 20.04.21

[20] SQLite.org, 2021, Appropriate Uses For SQLite, sqlite.org/whentouse.html, 20.04.21

[21] SQLite.org, 2021, Limits In SQLite, https://www.sqlite.org/limits.html, 20.04.21

[22] Oracle Corporation, 2021, 12.4 Limits on Table Size, https://dev.mysql.com/doc/mysql-reslimits-excerpt/8.0/en/table-size-limit.html, 20.04.21

[23] SQLite.org, 2021, What Is SQLite?, https://www.sqlite.org/index.html, 20.04.21

[24] Creative Commons Attribution-ShareAlike 2.0 license, 2021, About OpenStreetMap, https://wiki.openstreetmap.org/wiki/About_OpenStreetMap, 21.04.21

[25] Uninett, 2021, Fordeler med Feide – trygg og enkel innlogging og datadeling , https://www.feide.no/fordeler-med-feide, 17.04.21

[26] Uninett, 2021, Adding Feide login to a service, https://docs.feide.no/service_providers/getting_started/add_feide_login.html, 27.04.21

[27] Stenius, Petteri, 2020, The differences between OpenID Connect (OIDC), OAuth 2.0 & SAML, https://www.ubisecure.com/education/differences-between-saml-oauth-openid-connect/, 27.04.21

[28] Google, 2021, Google Cloud pricing, https://cloud.google.com/pricing, 18.04.21

[29] Amazon Web Services, 2021, AWS Free Tier, https://aws.amazon.com/free/?all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free%20Tier%20Types=*all&awsf.Free%20Tier%20Categories=*all, 18.04.21

[30] Darshan, Deva, 2018, Aerial View of Road in the Middle of Trees, https://www.pexels.com/photo/aerial-view-of-road-in-the-middle-of-trees-1173777/, 19.01.21

[31] Google fonts, Karla, https://fonts.google.com/specimen/Karla 12.05.2021, 11.05.21

[32] Facebook Inc, 2021, Building Your Own Hooks, https://reactjs.org/docs/hooks-custom.html, 18.04.21

[33] Saienko, Anatolii and Bakaus, Paul, 2017, DOM to Image, https://github.com/tsayen/dom-to-image, 17.04.21

[34] Refsnes Data, 1999-2021, SQL HTML Canvas Graphics, https://www.w3schools.com/html/html5_canvas.asp, 20.04.21

[35] www.programmersought.com, 2018-2021, How to draw a line with an arrow on the canvas?, https://www.programmersought.com/article/16051099203/, 20.03.21

[36] Barrera, Raúl, 2016, Pure CSS Select, https://codepen.io/raubaca/pen/VejpQP, 13.05.21

[37] Facebook Inc, 2021, Create React App, https://create-react-app.dev/, 19.01.21

[38] Ukirde, Divyajyoti, 2020, How to create React App with Flask backend?, https://dev.to/divyajyotiuk/how-to-create-react-app-with-flask-backend-2nf7, 23.01.21

[39] Pallets, 2020, Testing Flask Applications, https://flask.palletsprojects.com/en/1.1.x/testing/, 30.01.21

[40] www.programmersought.com, 2018-2021, [Flask] pytest file upload unit test, https://www.programmersought.com/article/57254767787/, 27.04.21

[41] Uninett, 2021, Feide, https://docs.feide.no/general/feide_overview.html, 17.04.21

[42] Uiterwijk, Patrick, 2021, Flask-OIDC, https://flask-oidc.readthedocs.io/en/latest/, 24.03.21

[43] Yang, Hsiaoming, 2021, Authlib: Python Authentication, https://docs.authlib.org/en/latest/index.html, 24.04.21

[44] Chris927, 2020, react-pkce, https://www.npmjs.com/package/react-pkce, 26.04.21

[45] Uninett, 2021, Obtaining tokens with Feide, https://docs.feide.no/service_providers/openid_connect/feide_obtaining_tokens.html, 17.04.21

[46] TRAVIS CI, GMBH, 2021, Build Matrix, https://docs.travis-ci.com/user/build-matrix/, 20.02.217

[47] Uninett, 2021, Getting started , https://docs.feide.no/service_providers/getting_started/index.html, 17.04.21

[48] Uninett, 2021, Arbeidsfordeling administrator og utviklere, https://docs.feide.no/service_providers/manage/openid_connect/registration.html, 17.04.21

[49] Uninett, 2021, Dataporten Dashboard, https://dashboard.dataporten.no/, 17.04.21

[50] Pallets, 2020, Tutorial, https://flask.palletsprojects.com/en/2.0.x/tutorial/, 14.05.21

[51] Parmar, Milan. 2021, React: Class VS Functional Components https://medium.com/star-gazers/react-class-vs-functional-components-a49383f65f0e Lastet ned 26.04.2021

[52] Introducing JSX, https://reactjs.org/docs/introducing-jsx.html, lastet ned 14.05.2021