Trafikksmart

En web applikasjon

INF219 Vårsemesteret 2020

Knut Mathias Gaard Storvestre

Institutt for informatikk

UNIVERSITETET I BERGEN

24. juni 2020

Oppsummering

I denne oppgaven har jeg valgt å se på utvikling av en ny applikasjon som har fått navnet Trafikksmart. Målsetningen er å lage en applikasjon med et brukervennlig grensesnitt som er anvendelig for alle, også helt vanlige brukere med begrenset teknologikompetanse.

Front-end skal fremstå enkelt og minimalistisk. Back-end skal derimot være mer komplisert for å kunne løse mange ulike problemstillinger og systematisere informasjon slik at den gir mening for brukerne.

Trafikksmart er utviklet med et klart formål, nemlig å gi brukerne pålitelig informasjon om trafikale forhold samlet i en applikasjon. I dag finnes denne informasjonen, men den er spredt hos ulike leverandører av informasjonstjenester. Brukerne må søke i ulike kilder. Dette tar tid og informasjonen de finner kan til dels være ufullstendig, upresis, utdatert eller feilaktig. Det finnes heller ikke egnede «profesjonelle» bruker-drevne applikasjoner som omhandler trafikkinformasjon i Norge. Det finnes bruker-initierte Facebook-sider for noen fjelloverganger. Informasjon legges ut ad-hoc og siden er bare kjent for et begrenset antall brukere.

Ideen om Trafikksmart er å etablere en veldrevet plattform for et aktivt og engasjert brukersamfunn som vil dele trafikal informasjon. Trafikksmart skal gjøre brukerne i stand til med noen enkle tastetrykk å hente ut beslutningsstøtte for å velge hvilken rute de skal kjøre for å unngå kø, eller kanskje finne et mer egnet tidspunkt for turen de planlegger å kjøre.

I begynnelsen gjorde jeg en studie av ulike andre utviklingsløp for å få innsikt i hva som gjøres når applikasjoner utvikles. Jeg lette etter en problemstilling der det var et udekket behov for en ny løsning i markedet. Jeg beskrev løsningen og satte noen overordnede krav til prosjektet. Jeg beskrev ulike brukerhistorier og designkriterier. Deretter så jeg på hvordan utviklingsprosjektet kunne anvende teknologier som React, Typescript, Node.js, Firebase, Material-UI. Hensikten med å bruke disse teknologiene er å gi brukeren en effektiv og enkel brukeropplevelse. Dette skal gjøre at Trafikksmart blir en attraktiv plattform som samler et tilstrekkelig antall brukere og dermed gir verdi til alle som besøker applikasjonen i hverdagen. Jeg så på implementering av applikasjonen. Helt til sist så jeg på hvilke deler av utviklingsløpet som fungerte bra, og hvor det hadde vært større utfordringer.

Konklusjonen på dette var at det var en lærerik prosess, der ikke alltid ting går som planlagt. Hvis det tas ut læring av de hindrene jeg støtte på, kan det brukes for å drive andre prosjekt på en enda bedre måte senere.

Min veileder, Anya Helene Bagge, har vært til uvurderlig hjelp i prosessen og jeg ønsker å takke for bistand i prosjektet.

Innholdsfortegnelse

1 Introduksjon

Denne applikasjon har til hensikt å etablere en brukervennlig plattform for å skape et aktivt brukerdrevet samfunn slik at trafikanter effektivt kan hente frem et pålitelig oversiktsbilde på ett eneste sted. Dette gjør at de får informasjon om trafikkforhold som de stoler på slik at de effektivt kan planlegge reise og spare tid og ikke minst unngå unødvendig forurensing av miljøet.

1.1 Motivasjon

Utviklingen av en applikasjon kan ta lang tid, være kostnadskrevende og teknologisk utfordrende. Det er derfor viktig at det er et reelt brukerbehov og aller helst en betalingsvilje enten i offentlig sektor eller i det private markedet slik at utviklings-kostnadene vil bli dekket. Det finnes også eksempler på utviklingsløp som er basert på fri programmering og åpen kildekode som utviklingen av Linux. Her er det mange ubetalte timer som har gått med for at et alternativt operativsystem til de store kommersielle aktørene skal fungere til glede for mange brukere.

Bakgrunnen for Trafikksmart-prosjektet er behovet for effektivt å finne pålitelig informasjon om trafikale forhold. Mange kan kjenne seg igjen i situasjoner der de har prøvd å finne slik informasjon. Det er imidlertid i dag mange kilder som må sjekkes, informasjonen kan være motstridende og vi kan bli usikre på om kilden er pålitelig eller oppdatert. Informasjonsgapet kan gjøre at vi risikerer å velge en kjørerute der det er bilkø. Da kan vi komme for sent til viktige avtaler som å hente barn i barne-hagen eller simpelthen helt unødvendig sløse bort verdifull tid i kø. Dette har en samfunnsøkonomisk kostnad og skaper irritasjon for enkeltmennesker i hverdagen.

Det er i dag mange alternative kilder der vi kan innhente trafikkinformasjon som trafikkmeldinger på radio og i nettaviser eller Google Maps sin trafikkapplikasjon.

Vegvesenet har investert store summer i nettsider og de har satt opp store lystavler langs veiene. Lystavlene er ikke alltid i bruk, eller oppdatert. De gjør også at informasjonen kommer sent ut til bilisten som bokstavelig talt er på veien og har et begrenset antall valgmuligheter for å omdirigere kjøreruten sin.

Vegvesenet sin nett-løsning er kartbasert, men den fremstår i dag som en noe utdatert versjon av Google Maps. Informasjonen som formidles her angår i hovedsak planlagt veiarbeid. Den gir også informasjon når det er innført kolonnekjøring eller fjelloverganger er stengt. I praksis har deres løsning flere utfordringer. Først og fremst stemmer ikke alltid informasjonen med virkeligheten fordi det er forsinkelser i

oppdateringene. Bilister har opplevd at det står at veien er stengt, men når de kommer til veibommen på fjellet viser den seg å være åpen. Bilistene kan derfor ikke stole på informasjonen.

Vegvesenets løsning synes å ha et potensial for å gi mer effektiv informasjon til trafikantene. Verdifull tilleggsinformasjon kunne eksempelvis ha vært om de koblet værdata som snøfall og vindstyrke og dermed kunne kalkulere ut intervaller som anga risiko for innføring av kolonnekjøring eller veistengning på fjellovergangene. I dag er det slik at du kan kjøre hjemmefra med informasjon om at veien er åpen, men når du kommer opp på høyfjellet møter du en stengt bom uten at dette er varslet. En effektiv tjeneste kan angi sannsynligheter i form av grønt, gult eller oransje fare-varsel. Ved et oransje farevarsel kan det f.eks være 75% sjanse for at veien vil bli stengt.

Vegvesenet angir ingen informasjon om hvor mange biler de tar med i hver kolonne eller hvor mange biler som venter ved bommen. Slik informasjon kunne hjelpe bilistene til å ta bedre valg. Vegvesenet kunne også fått en mer effektiv løsning om de hadde gitt adgang for bilister til å legge ut informasjon om trafikkuhell eller hendelser som skaper køer. Mange bilister vil gjerne dele informasjon med andre i samme situasjon. De kan fortelle andre hvor mange som venter på neste kolonne slik at andre bilister kan ta bedre beslutninger om reisen sin.

For å kompensere for informasjonsgapet har enkeltpersoner etablert Facebook-grupper som «Kolonnekjøring Hardangervidda» og «Kolonnekjøring Hol-Aurland». Her deler de informasjon om antall biler som venter, værforhold og informasjon de får uformelt fra andre bilister eller fra Vegvesenets mannskap. Ulempen er imidlertid at det ikke alltid er bilister som føler at det er deres ansvar å legge ut informasjon på Facebook-siden. De kan også være uvitende om at en slik informasjonsdelingsarena finnes.

Brukerdrevne løsninger er fremtidsrettede, da kostnadene reduseres ved at brukerne bidrar med gratisarbeid. De er smidigere og gir bedre beslutningsstøtte til bilistene. Vegvesenet har ikke åpnet for brukerinvolvering, noe som kunne gitt en verdifull dynamikk.

Min motivasjon har derfor vært å utvikle en applikasjon som skal være populær og bli brukt av en kritisk masse, slik at den kan gi verdi til trafikanter på norske veier. Den kan også være skalerbar ved å utvikles tilpasset til bilister i andre land.

1.2 Beskrivelse av løsning

Trafikksmart er utviklet mot to ulike brukergrupper. Den første gruppen er aktive medlemmer som registrerer seg. Den andre gruppen er mer tilfeldige brukere som søker etter trafikkinformasjon fra tid til annen fordi de har hørt at andre har gode

erfaringer med å innhente informasjon her. One-stop-shopping effekten: «Det er her vi går når vi skal ha trafikkinformasjon».

1.2.1 Registrerte brukere av applikasjonen

- Kan finne frem til spesifikk trafikkinformasjon for de områdene de skal ferdes i
- Kan registrere trafikkinformasjon som er til nytte for andre brukere
- De kan like eller ikke like andre sine innlegg og kommentere andres innlegg
- De ulike brukerne kan bli evaluert utfra antall likes. De får poeng etter hvor relevant andre brukere mener deres innlegg og kommentarer tidligere har vært. Denne ideen kommer fra Stack Overflow.
- Registrerte brukere legger igjen e-post adresse

1.2.2 Uregistrerte brukere

- Kan lese informasjonen andre har lagt ut, men kan ikke like eller kommentere innlegg
- Vil få anledning til å registrere seg som bruker hvis de ønsker

Målsetningen min er at applikasjonen skal se ryddig og troverdig ut, slik at den oppfattes å være seriøs og dermed et attraktivt sted å hente informasjon eller bidra ovenfor andre med viktig trafikal informasjon. Applikasjonen skal være brukerdrevet og utgjøre et brukersamfunn av entusiaster.

Den skal være brukervennlig og intuitiv, også for mindre erfarne teknologibrukere.

Oppsummeringsvis så er hensikten med prosjektet å utvikle en applikasjon som kan dekke et brukerbehov som ikke er dekket i dag, nemlig å etablere en plattform for å dele trafikal informasjon til en bred brukergruppe på en effektiv og enkel måte.

Systemet skal ha to deler - server og klient. Disse delene skal kommunisere med hverandre via en API.

1.3 Overordnede krav til prosjektet (målsettinger)

Ved oppstarten satte jeg noen minimumskrav og målsetninger for både for web applikasjonen og back-end serveren.

1.3.1 Overordnede krav for server-implementering

- Rask og skalerbar
- Tilgjengelig 24/7
- Lave utviklingskostnader
- Autentiseringsløsninger hvor passord lagres på en sikker måte
- Godt dokumentert struktur slik at løsningen kan vedlikeholdes og utvikles videre

1.3.2 Mål for klient-implementering

- Intuitivt brukergrensesnitt
- Godt dokumentert
- Session cookies
- Passord skal sendes til serveren på en sikker måte

1.3.3 Minimum Viable Product (MVP)

- Førsteutgaven av løsningen skal være velfungerende
- Brukerne skal kunne registrere seg og logge inn
- Det skal være mulig å se andres meldinger
- Siden skal ikke gå ned på grunn av brukerfeil
- Back-end og front-end kommuniser ved hjelp av en API

2 Kravspesifikasjon og analyse

Dette kapitelet beskriver mer detaljerte krav enn de overordnede kravene som ble beskrevet i forrige kapittel. Prosessen for å utarbeide kravene har vært en intern analyse der jeg har testet ut løsninger på ulike potensielle brukere. Interessentene ville i et større prosjekt vært Vegvesenet, ulike nettaviser samt brukere av applikasjonen. Jeg har ikke gjort utfyllende interessentanalyser, men har tenkt rundt interessentenes behov.

Jeg har identifisert de funksjonelle kravene til trafikksmart applikasjonen.

2.1 Funksjonelle krav

- Registrering
    - Systemet skal la nye brukere registrere seg ved hjelp av e-post.
    - Det skal ikke være mulig å kunne registrere seg med et svakt passord
    - Hver bruker må ha et unikt brukernavn og e-post
    - Brukeren vil få en feilmelding om hva som må endres hvis kravene ovenfor ikke tilfredsstilles
    - Cookie skal returneres til brukerens nettleser etter registrering
- Login
    - Systemet skal gi en feilmelding til brukeren hvis passord eller brukernavn er feil
    - Cookie skal returneres til brukerens nettleser etter registrering
- Nedlasting
    - Når nettleseren venter på respons fra server skal dette vises med tekst eller et ladeikon
- Nettsider
    - Login
    - Registrering

    - Hjem
        - Kart som viser hvor ulike trafikkhendelser har inntruffet
        - Meldinger fra brukerne som beskriver de ulike hendelsene

2.2 Ikke-funksjonelle krav

Overordnet så er min målsetning å oppnå en enkel og godt dokumentert kode. For denne applikasjonen legges det vekt på å unngå kodeduplikasjon. Det skal være oversiktlig og dette oppnås ved at koden fordeles på flere filer. Hvis koden begynner å bli for komplisert, kan det være vanskelig å utvikle den videre. Hver fil skal være godt dokumentert og det skal være kommentarer hvor koden starter å bli uoversiktlig.

- Back-end
    - Det skal være mulig å laste opp profilbilde
    - Passord skal ikke kunne leses av administratoren av databasen
    - Godt dokumentert
    - Filene skal lagres sikkert
    - Skal være beskyttet mot hacking
    - Input fra front-end/brukere skal sjekkes for skadelig innhold

- Front-end
    - Det skal være enkelt å navigere mellom sidene
    - Hurtig responstid
    - Input feltene skal valideres
    - Animasjon under nedlasting
    - Komfortabel skrifttype, fargevalg og lett leselig tekst

2.3 Brukerhistorier

Denne delen omhandler de viktigste brukerhistoriene for systemet. Dette er uformelle skriftlige beskrivelser som er skrevet utfra en brukers ståsted. Det er brukt en spesiell setningsstruktur (Agile Alliance, 2020) som inneholder disse tre punktene.

- Som en ... (**Hvem** har lyst til å oppnå noe)
- Har jeg lyst til ... (**Hva** de har lyst til å oppnå)
- Slik at ... (**Hvorfor** de har lyst til å oppnå dette)

2.3.1 For trafikanter

- Som jobbpendler ønsker jeg effektivt å få oversikt over trafikken før jeg kjører på jobb, eller hjem igjen, uten å måtte logge inn og sjekke mange ulike nettsteder før jeg reiser
- Som hytteturist ønsker jeg effektivt å få oversikt over stengte veier eller kolonnekjøring på fjellovergangene, veiarbeid, omkjøringer eller køer som kan gjøre at jeg velger å ta turen på et annet tidspunkt slik at jeg slipper å sitte

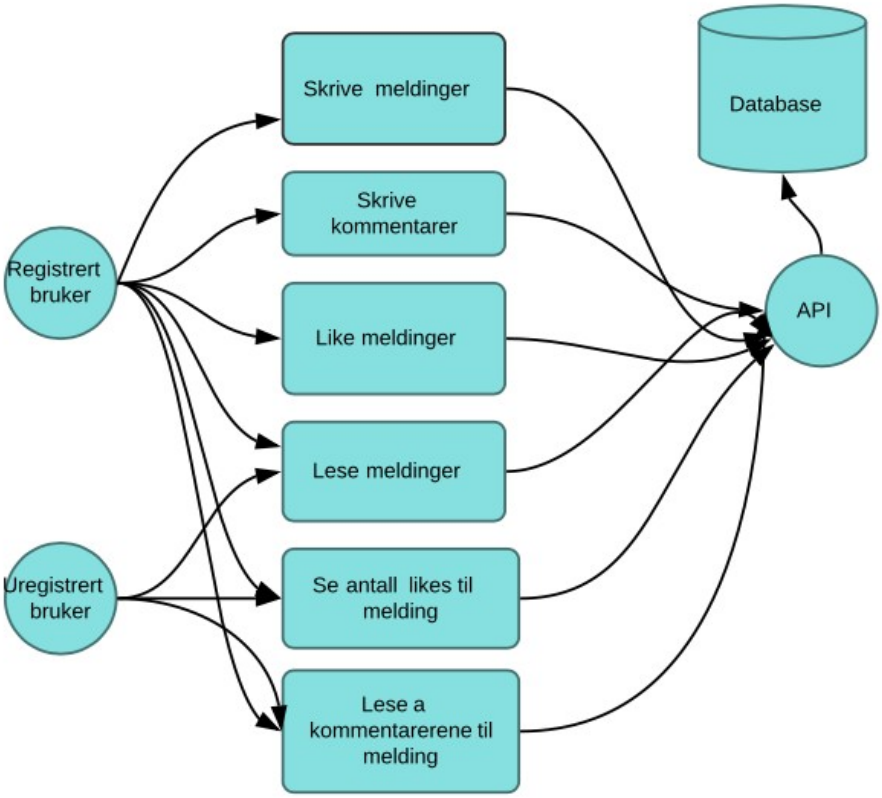
flere timer i kø oppe på høyfjellet. Jeg ønsker å gå til et trygt og oppdatert nettsted der all relevant informasjon finnes i stedet for å sjekke mange ulike informasjonskilder som Vegvesenet, nettaviser, Facebook sider, værtjenester osv.

- Som en vanlig bruker ønsker jeg ikke å måtte ha tung teknologibakgrunn for å kunne bruke denne nettsiden

2.3.2 For interessenter som Vegvesenet, nettaviser, Facebook etc.

- Som en offentlig organisasjon som skal sørge for at veiene i Norge fungerer iht. trafikantenes behov ønsker jeg å kunne legge inn oppdaterte kommentarer for å hjelpe bilistene til å ta gode valg for avreisetidspunkter slik at Norge ikke lider et samfunnsøkonomisk tap ved at bilister sitter unødvendig mye i kø
- Som en offentlig organisasjon som skal sørge for at veiene i Norge fungerer ønsker jeg at bilistene på en tryggere måte kan komme seg over fjellovergangene. Jeg ønsker at de skal ha en nettside der de kan finne informasjon slik at de kjører over på et tidspunkt der det er mindre risiko for å tilbringe mange timer i snøstorm for å vente på neste kolonne eller åpning av
- Som en nettavis ønsker jeg at det er en felles plattform med høy integritet der all troverdig informasjon som er relevant for bilister finnes samlet
- Som en nasjonal kringkaster ønsker jeg at det er et felles nettsted med høy integritet, der all troverdig informasjon som er relevant for bilister, finnes samlet

2.4 Brukereksempler



Figur 1: Brukereksempler

Over vises et brukerdiagram. Det gir et oversiktsbilde over hvordan ulike brukere kommer i kontakt med applikasjonen. Dette viser hva de ulike brukerne er autorisert til å gjøre.

3 Design

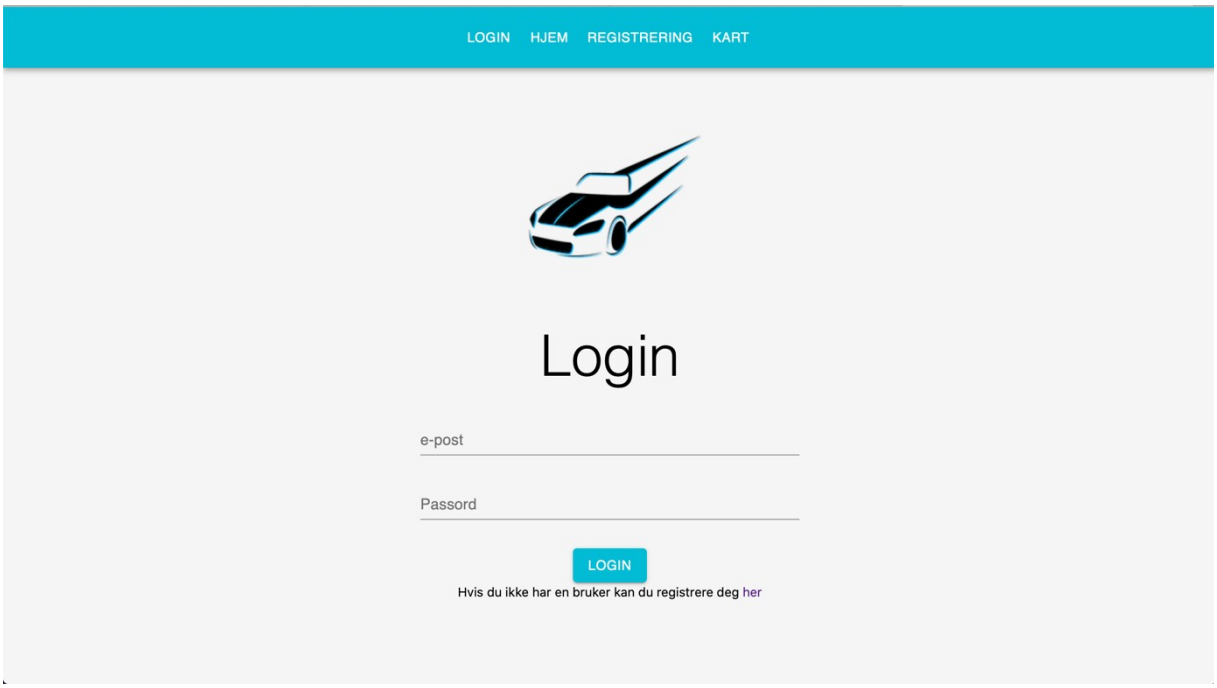
Dette kapitlet beskriver hvordan Trafikksmart er designet. Det er to hovedhensyn som er vektlagt – selvforklarende og intuitivt design, samt en minimalistisk billedflate der ikke brukeren overlesses med informasjon. Ellers er det lagt vekt på de kravene som er beskrevet i kapitlet over.

Den visuelle designen er implementert direkte. I andre prosjekter kan en prototype først prøves ut på et brukerpanel for deretter å settes i produksjon. Deretter bruker vi sprinter til å stadig forbedre applikasjonen.

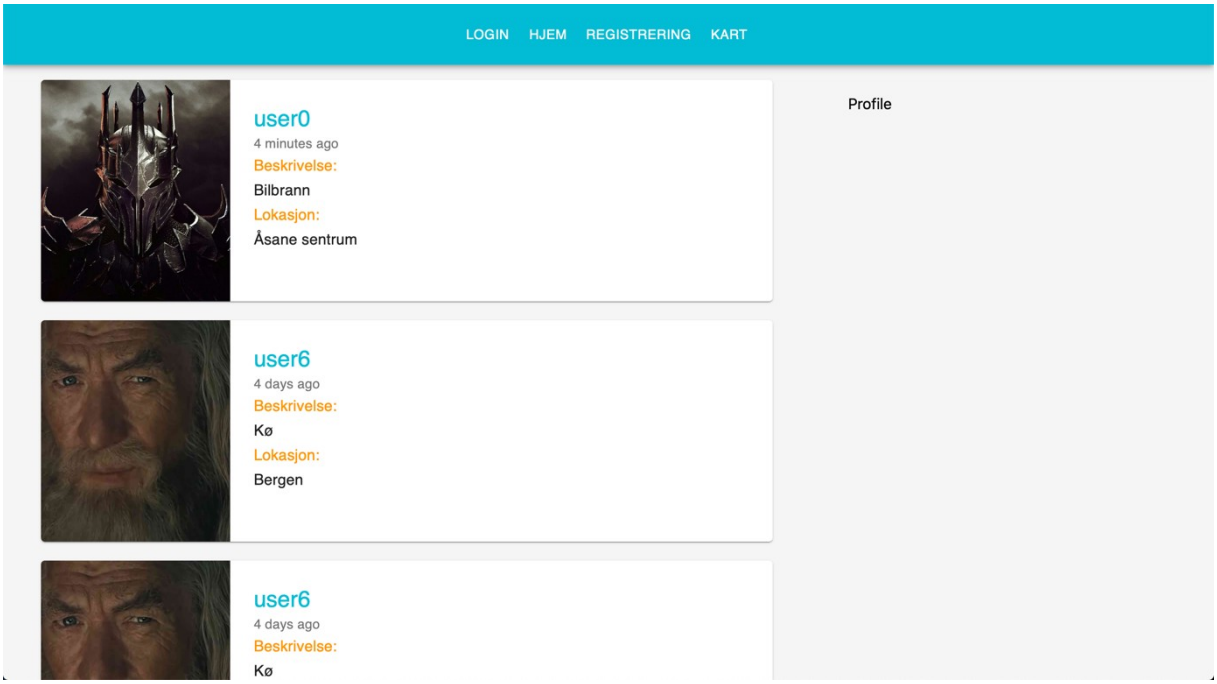
3.1 Front-end

3.1.1 GUI

Fargevalget er lyseblått som hovedfarge og oransje som sekundærfarge. Det er et enkelt og ryddig design. GUI designen vises under ved de ulike skjermbildene.



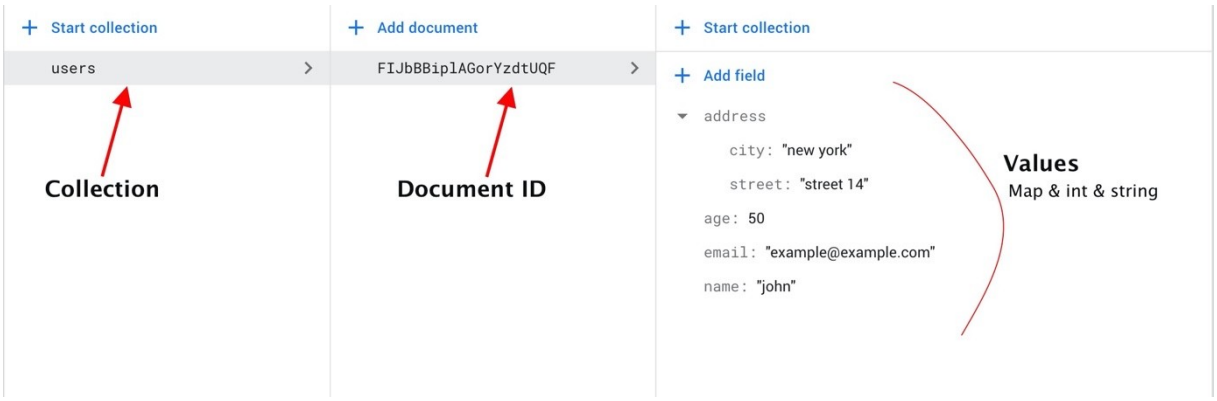
Figur 2: Innloggings-skjermbilde



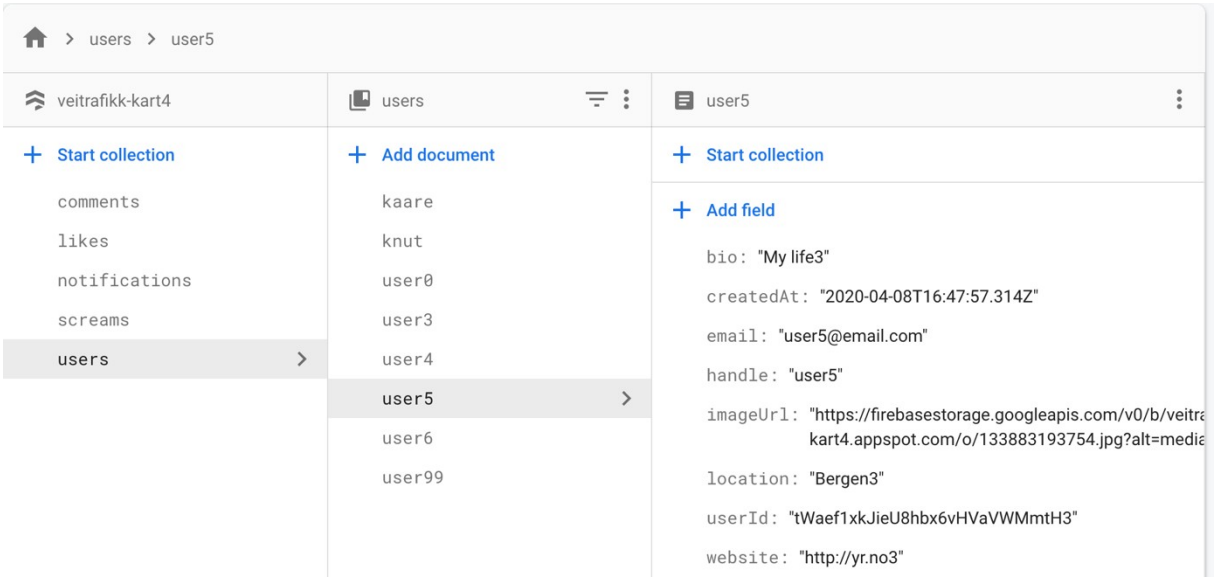
Figur 3: Hjemmesiden

3.1.2 Back-end

Databasen er bygget opp ved hjelp av Firebase Firestore (FirebaseTeam, 2020). Dette er en NoSQL database. Data lagres ved hjelp av Collections og Documents. Problemet med Firestore er at dataene er «immutable». Dette innebærer utfordringer hvis det blir behov for å endre data som er opprettet i databasen. Dette er problematisk av flere årsaker. Det medfører mye ekstraarbeid hvis du har lyst til å endre navn på Collections eller gjøre andre endringer med dataene. Dette er tungvint hvis du har lyst til å endre navn på en Collection. Da må du opprette en ny Collection med nytt navn og flytte all data fra den gamle Collection manuelt over til den nye.



Figur 4: Eksempel på Firestore DB



Figur 5: Min Firestore DB

Oversikt over tabellene i databasen

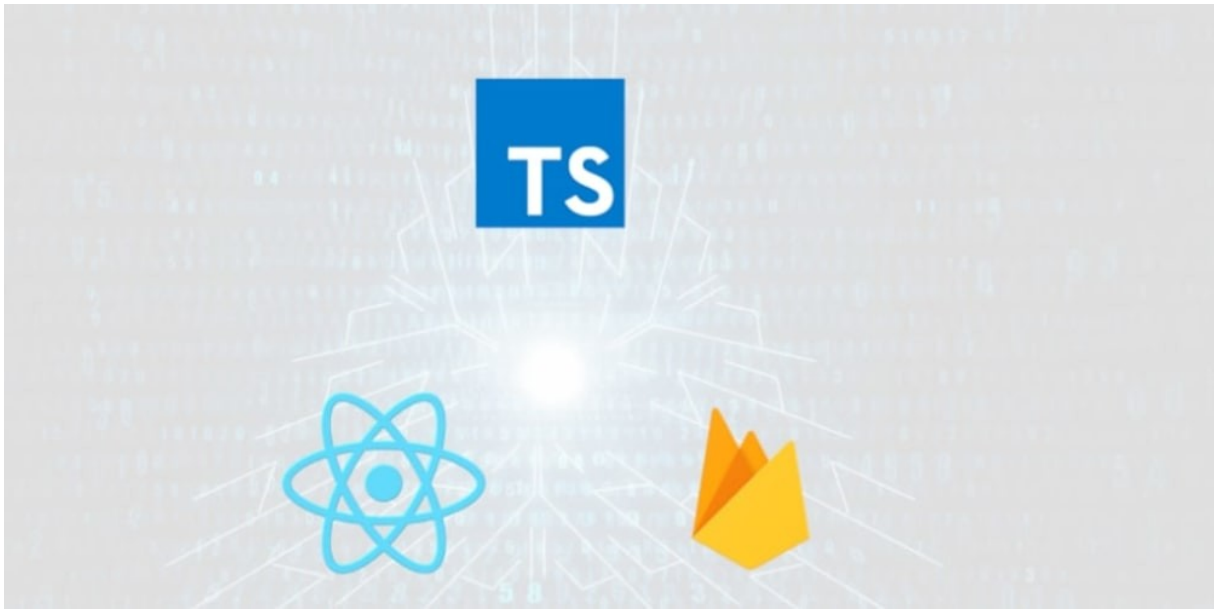
- Comments
  - Body – selve kommentaren
  - createdAt – Når kommentaren ble skrevet
  - screamId – ID til scream/post som kommenteres
  - userHandle - brukernavn
  - userImage - Refereanse tin profilbilde
- Likes
  - ScreamID – ID av meldingen som likes
  - userHandle – navn på den som trykker like
- Notifications
  - createdAt
  - read – boolean variabel som er True hvis brukeren har lest notifikasjonen
  - recipient – navn på bruker som skal få notifikasjonen
  - screamId – referanse til scream som det angår
  - sender – navn på bruker som trigger notifikasjonen
  - type – enten comment eller like
- Screams
  - Body – beskrivelse av hendelsen
  - commentCount – antall kommentarer
  - createdAt
  - location – hvor hendelsen har tatt plasss
  - userHandle – brukernavn til den som har skrevet

- Users
  - Biografi
  - createdAt - Tidspunkt av registrering
  - Email
  - Handle – brukernavn
  - ImageURL - Refereanse tin profilbilde
  - Lokasjon
  - userId
  - Website – hjemmeside

3.2 Anvendte teknologier

I en web applikasjon brukes det ulike teknologier. Trafikksmart applikasjonen bruker Type Script som programmeringsspråk, NodeJS for dependencies, React for utvikling av klient delen av applikasjonen og Firebase som back-end og database. TypeScript

er en typesikker versjon av JavaScript som arbeider sammen med vanlig JavaScript. React er et bibliotek til TypeScript.



Figur 6: Anvendte teknologier

3.2.1 Begrunnelse for anvendte teknologier

- TS (TypeScript)
  - Er som JavaScript bare typesikkert. Typescript kompilerer til ren JavaScript kode.
  - Gjør JavaScript koden enklere å forstå og feilsøke.
  - Nesten all dokumentasjon til Firebase er skrevet i TypesScript eller JavaScript.
- React
  - Godt dokumentert
  - React bruker komponenter. Komponenter er gjenbrukbare biter av kode som jobber i isolasjon fra hverandre og returnerer HTML-kode. Adgangen til gjenbruk forenkler arbeidet.
  - Rask responstid hos brukeren.
- Firebase
  - Enklere å lære seg enn AWS (Amazon Web Services) som er tyngre å sette opp.
  - Enkelt å kommunisere med databasen med enkle http-forespørsler
  - «Callable functions» gjør det enkelt å skrive oversiktlig back-end kode som trigges av HTTP-forespørsler.

3.3 Utrulling

Jeg brukte Firebase som er en pay-as-you-go skytjeneste. Firebase en sky data-behandlingsplattform laget av Google med mye funksjonalitet. I dette prosjektet har vi brukt disse funksjonene;

- Cloud Firestore – trygg lagring av data, NoSQL database
- Cloud Functions – for å betjene back-end API
- Firebase Authentication – trygg lagring av passordene til brukerne
- Firebase Storage- for lagring av profilbilder

4 Implementering

Jeg har brukt *Git* og *Github* som versjonskontroll slik at jeg kan gå tilbake til tidligere versjoner. Dette er også praktisk hvis det gjøres feil. Da kan det enkelt gås tilbake og gjøres nye forsøk hvis dette er nødvendig. Jeg har laget en ny gren fra masterversjonen for hver ny funksjonalitet som blir implementeres i systemet.

4.1 Back-end

Prosjektet har brukt *npm* (node package manager) for å laste ned og styre back-end «dependencies». Dependencies er kode som har blitt utviklet av tredjeparter. Dette gjør utviklingen av produktet langt mer effektivt fordi man unngår å måtte finne opp hjulet på nytt hvor hver nye funksjon.

4.1.1 Oppdatering av back-end

For å oppdatere back-end/koden på serverene bruker jeg Firebase CLI. Firebase CLI er bygget på NodeJS som igjen administreres av npm. For å oppdatere back-end bruker jeg «firebase deploy --only functions».

```
~/INF219/veitrafikk-kart4 refactoringAndRenaming
› firebase deploy --only functions
⚠  functions: package.json indicates an outdated version of firebase-functions.
Please upgrade using npm install --save firebase-functions@latest in your functions directory.

    Deploying to 'veitrafikk-kart4'...

i  deploying functions
Running command: npm --prefix "$RESOURCE_DIR" run lint

> functions@ lint /Users/knut/INF219/veitrafikk-kart4/functions
> tslint --project tsconfig.json


WARNING: /Users/knut/INF219/veitrafikk-kart4/functions/src/handlers/posts.ts:10:17 - Identifier 'screams' is never reassigned; use 'const' instead of 'let'.
WARNING: /Users/knut/INF219/veitrafikk-kart4/functions/src/handlers/users.ts:112:9 - Identifier 'userData' is never reassigned; use 'const' instead of 'let'.
WARNING: /Users/knut/INF219/veitrafikk-kart4/functions/src/handlers/users.ts:157:9 - Identifier 'userData' is never reassigned; use 'const' instead of 'let'.
WARNING: /Users/knut/INF219/veitrafikk-kart4/functions/src/handlers/users.ts:265:9 - Identifier 'batch' is never reassigned; use 'const' instead of 'let'.
WARNING: /Users/knut/INF219/veitrafikk-kart4/functions/src/index.ts:127:17 - Identifier 'batch' is never reassigned; use 'const' instead of 'let'.
WARNING: /Users/knut/INF219/veitrafikk-kart4/functions/src/util/validators.ts:18:9 - Identifier 'errors' is never reassigned; use 'const' instead of 'let'.
WARNING: /Users/knut/INF219/veitrafikk-kart4/functions/src/util/validators.ts:39:9 - Identifier 'errors' is never reassigned; use 'const' instead of 'let'.
WARNING: /Users/knut/INF219/veitrafikk-kart4/functions/src/util/validators.ts:60:9 - Identifier 'userDetails' is never reassigned; use 'const' instead of 'let'.

Running command: npm --prefix "$RESOURCE_DIR" run build

> functions@ build /Users/knut/INF219/veitrafikk-kart4/functions
> tsc

✔  functions: Finished running predeploy script.
i  functions: ensuring required API cloudfunctions.googleapis.com is enabled...
✔  functions: required API cloudfunctions.googleapis.com is enabled
i  functions: preparing functions directory for uploading...
i  functions: packaged functions (71.6 KB) for uploading
✔  functions: functions folder uploaded successfully
i  functions: updating Node.js 8 function api(europe-west1)...
i  functions: updating Node.js 8 function createNotificationOnLike(europe-west1)...
i  functions: updating Node.js 8 function deleteNotificationOnUnLike(europe-west1)...
i  functions: updating Node.js 8 function createNotificationOnComment(europe-west1)...
i  functions: updating Node.js 8 function onUserImageChange(europe-west1)...
i  functions: updating Node.js 8 function onScreamDelete(europe-west1)...
✔  functions[onUserImageChange(europe-west1)]: Successful update operation.
✔  functions[deleteNotificationOnUnLike(europe-west1)]: Successful update operation.
✔  functions[onScreamDelete(europe-west1)]: Successful update operation.
✔  functions[createNotificationOnComment(europe-west1)]: Successful update operation.
✔  functions[createNotificationOnLike(europe-west1)]: Successful update operation.
✔  functions[api(europe-west1)]: Successful update operation.

✔  Deploy complete!

Project Console: https://console.firebase.google.com/project/veitrafikk-kart4/overview

~/INF219/veitrafikk-kart4 refactoringAndRenaming 1m 40s
›
```

Figur 7: Eksempel på «deployment» av Firebase Functions

4.1.2 Dependencies

I back-end bruker vi følgende «dependencies»

Figur 8: Back-end dependencies

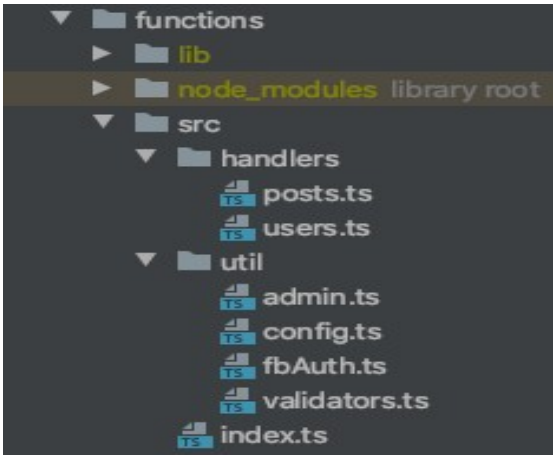
- Busboy (NumerousAuthors, 2019)
  - Brukes for å analysere innkommende HTML-data/Forms

    - Brukes i dette tilfellet til å la brukeren laste opp en bildefil til «Google Cloud Storage».
- Express (NumerousAuthors, 2019) (Node.js Foundation, 2017)
    - Brukes til å sette opp middle ware
    - Definerer «Routing Table» som er brukt for å bestemme hvilken handling som skal utføres utfra hvilken HTTP forespørsel (GET, POST, DELETE, PUT) og URL som har blitt sendt fra klienten/front-end.
- Firebase-admin (FirebaseTeam, 2020)
    - Gir applikasjonen privilegier til å kunne lese og skrive all data som ligger i Firestore DB
    - Brukes blant annet i functions/src/users.ts til å laste opp bilder og functions/src/util/fbAuth.ts til å authentisere cookies.
- Firebase-functions (FirebaseTeam, 2020)
    - Serverless framework
    - Firebase functions er koden i back-end. Denne koden trigges av spesifikke HTTP forespørsler som sendes fra klienten.
- Firebase (FirebaseTeam, 2020)
    - Brukes til å initiere applikasjonen. Gjøre det mulig for Firebase å finne ditt spesifikke prosjekt utav alle prosjektene som ligger på Firebase servere.

4.1.3 Oversikt over senere implementering

Når forespørselen kommer fra klienten vil den først gå til Index.ts som har referanser til alle funksjonene som er i filene til handlers. Disse filene tar seg av alle HTTP forespørslene som serveren kan utføre. Disse filene bruker igjen util/fbAuth for autentisering og util/validators for å validere og klienten kommer med en gyldig input til server.

Util/Admin.ts brukes til å opprette kommunikasjon med skyen. Ved at den sender api-nøkler sammen med masse annen data fra util/config.ts som Firebase bruker til å identifisere hvilket prosjekt du jobber med.



Figur 9: API's directory structure

4.2 Front-end

For å initiere front-end brukte jeg «npx create-react-app --typescript» som fungerer som en boilerplate til å videre utvikle en React applikasjon med typescript. Jeg bruker også npm for å holde styr på dependencies. Front-end er implementert slik at den kommuniserer med server via HTTP forespørsler.

4.2.1 Bruk av applikasjon utenfor Norges grenser

Web applikasjonen er tiltenkt bruk I Norge. Det kan være turister eller fremmedspråklige som har bruk for informasjonen som gjelder norske veier på engelsk eller andre språk. Dette løses ved å ha et ikon som eksempelvis et flagg der du kan veksle mellom norsk og engelsk. Den kan også tenkes å være av interesse for andre land der informasjonsflyten mellom myndigheter og publikum når det gjelder veiene ikke er god nok.

I dette prosjektet ble en flerspråklig applikasjon ikke utviklet, men det kan la seg gjøre ved bruk av ulike tilgjengelige oversetter moduler. Erfaringene med slike oversetter moduler blir stadig bedre etterhvert som de utvikles videre. En senere modell av web applikasjon kan derfor ha som ambisjon å tilby flere språk.

4.2.2 Styling av visuelle komponenter

De fleste visuelle komponentene i front-end er laget ved hjelp av Material-UI. Material-UI er et front-end bibliotek som inneholder mange gode estetiske og funksjonelle komponenter.

4.3 Status

Web applikasjonen er ikke klar for offentlig bruk. Den mangler fortsatt kryptering av passord og meldingene som blir registrert er fortsatt ikke synkronisert med kartet. I den endelige versjonen ønsker jeg å flytte kartet over til hjemmesiden.

På neste side følger en tabell som viser hvilke funksjoner som har blitt implementert, delvis implementert eller ikke implementert.

Funksjonalitet	**Status**
Registrering	Implementert • Registrering av nye brukere • Passord styrke sjekk • Feilmeldinger som forklarer hva brukeren må endre hvis ikke registreringskravene er tilfredsstilt • Brukeren logges inn hvis registreringen er vellykket • Knapp som fører til loginsiden hvis brukeren allerede har en brukerkonto • Brukeren registreres med et generisk profilbilde Ikke implementert • Passordet er kryptert når det sendes over HTTP
Users	Implementert • Seperasjon mellom ulike brukere • Endre profilbilde (Kun implementer i back-end) • Se hvilken Post
Hjemmeside	Implementert • Viser meldinger o Beskrivelse o Lokasjon • Viser profilbilde av brukeren som har skrevet melding Ikke implementert • Kommentarfelt til meldinger med kommentarer • Det er ikke mulig å skrive meldinger (implementer i back-end) • Det er ikke mulig å se antall likes til de ulike meldingene (implementert i back-end) • Det er ikke mulig å like meldinger (implementert i back-end)
Kart	Implementert

	• Kart fra google maps Ikke implementert • Mangler ikoner for hvor de ulike trafikkhendelsene har hendt
Login	Implementert • En cookie vil bli returnert hvis brukeren logger inn med riktig brukernavn og passord • Brukeren vil få en feilmelding hvis brukernavn eller passord er feil • Knapp som fører til registeringssiden hvis brukeren allerede har en brukerkonto. Ikke implementert • Passordet er kryptert når det sendes over HTTP

Tabell 10: Oversikt over implementering

Kilder

Agile Alliance, 2020. *User-story-template.* [Online]
Available at: https://www.agilealliance.org/glossary/user-story-template/
[Accessed 24 June 2020].

NumerousAuthors, 2019. *npmjs.* [Online]
Available at: https://www.npmjs.com/package/busboy
[Accessed 24 juni 2020].

NumerousAuthors, 2019. *npmjs.* [Internett]
Available at: https://www.npmjs.com/package/express
[Funnet 24 june 2020].

FirebaseTeam, 2020. *firebase.google.* [Online]
Available at: https://firebase.google.com/docs/firestore
[Accessed 24 Juni 2020].

FirebaseTeam, 2020. *firebase.google.* [Internett]
Available at: https://firebase.google.com/docs/database/admin/start
[Funnet 24 juni 2020].

FirebaseTeam, 2020. *firebase.google.* [Internett]
Available at: https://firebase.google.com/docs/functions
[Funnet 24 juni 2020].

FirebaseTeam, 2020. *firebase.google.* [Internett]
Available at: https://firebase.google.com/docs/web/setup
[Funnet 24 juni 2020].

Node.js Foundation, 2017. *Express.* [Online]
Available at: https://expressjs.com/
[Accessed 24 juni 2020].