Høgskulen på Vestlandet

Fakultet for ingeniør- og naturvitskap

Institutt for datateknologi, elektroteknologi og realfag

PreGO!
Systemdokumentasjon

Versjon <2.0>

REVISJONSHISTORIE

Dato	Versjon	Beskrivelse	Forfatter
22.04.2023	1	Begynte å fylle opp punkter	William,Tommy
24.04.2023	1	installasjon og kjøring	William,Tommy
02.05.2023	1.5	Skrev mer på Servertjeneste, kontinuerlig integrasjon og testing.	Tommy
20.5.2023	2.1	Oppdaterte testing	Tommy
21.5.2023	2.2	ryddet opp	William,Tommy

INNHOLDSFORTEGNELSE

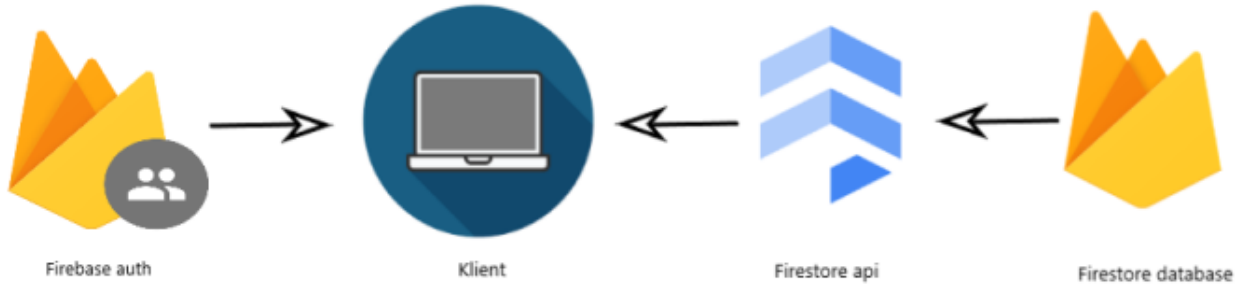
1 INNLEDNING

Dette dokumentet omhandler systemdokumentasjonen for prototypen PreGO!. Formålet med dokumentet er å gi en oversikt over hvordan systemet er sammensatt. Dokumentet beskriver systemets arkitektur, prosjektstruktur, klassediagram og serverens funksjonalitet i forhold til databasemodell og servertjenester. Videre dekkes tekniske aspekter som sikkerhet, installasjon og kjøring av prototypen, samt dokumentasjon av kildekoden og kontinuerlig integrasjon og testing.

2 Arkitektur

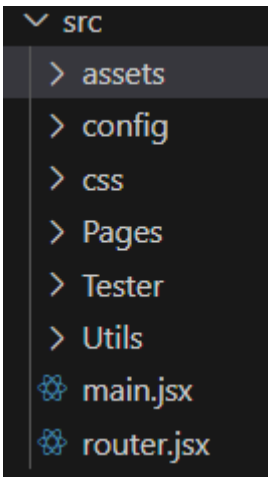
- minimum viable product (MVP)
    - Klient server arkitektur
    - Lager en prototype for oppdragsgiver for å teste om funksjoner de ønsker å tilby er nyttig for klinikere.
    - Bare nødvendige funksjoner
        - Få tilbakemeldinger for funksjonene til videreutvikling

Systemet vi har laget er en simpel klient-server arkitektur der hvor funksjoner som benytter av seg Firestore api ligger direkte på den relevante sidene eller en mappe med funksjoner istedenfor for å lage en DAO(Data access object) der hvor du har alt database logikken som henting, endring og sletting av data fra databasen separert fra sidene. Dette tilbyr fordeler som gjenbrukbar kode og enklere testing av logikk('Data access object', 2023). Hovedgrunnen til at prototypen ikke benytter seg av et DAO objekt er fordi prototypen er kodet i react som benytter seg av komponentbasert arkitektur og at firebase kan hente data direkte gjennom API til Firestore database, men en del av funksjonene som er knyttet til spørringer mot databasen ligger i en separat mappe som gjør det lettere å teste funksjonene



figur 1: viser en arkitekturskisse som illustrerer en innlogging på klienten ved hjelp av Firebase autentisering. Etter vellykket autentisering bruker klienten Firestore API til å få tilgang til Firestore-databasen, hvor det er mulig å utføre operasjoner for å hente, endre og legge til data.

3 PROSJEKTSTRUKTUR



Figur 2 - Viser mappe strukturen vår. Forklaring under.

Config: Informasjon som blir brukt av Firebase SDK for å kommunisere med prosjektet.
https://firebase.google.com/docs/web/setup#initialize-firebase-in-your-app

CSS: Kode for hvordan sidene skal presentere innholdet i webprosjektet. Hver side har sin egen liggende her.

Pages: Her ligger kode for hva som skal bli presentert på hver enkelt side og logikken til hver eneste funksjon på webapplikasjonen

Tester: Her ligger enhetstester for prosjektet med mock data.

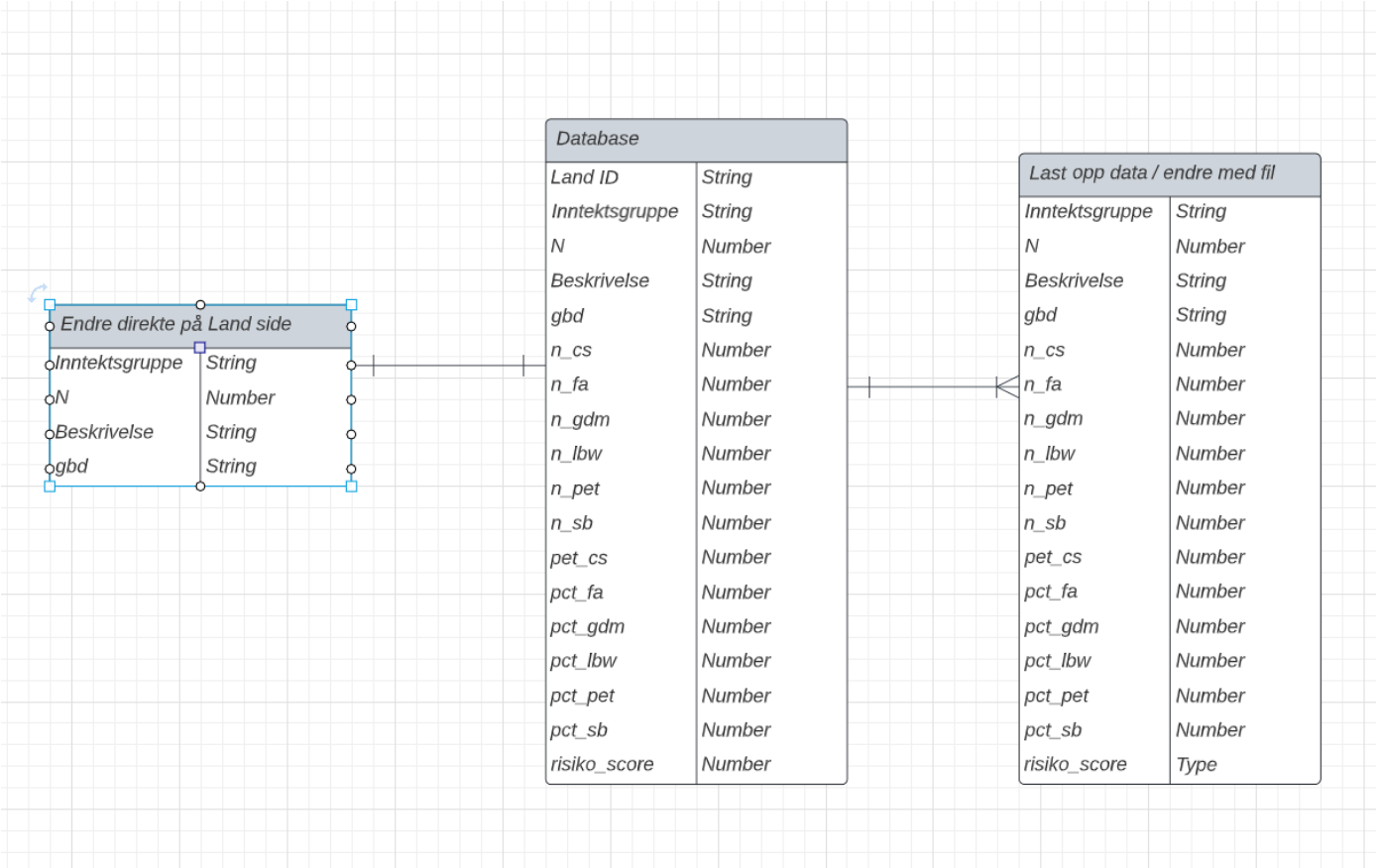
Utils: inneholder noen funksjoner for henting data fra database, risikoscore algorithm og route funksjoner

Main.jsx: Denne filen fungerer som en startpunkt for webapplikasjonen og hjelper med å presentere webapplikasjonen for brukeren.

Router.jsx: Hjelper brukerne med å navigere til de forskjellige sidene ved å si hvilken side som skal bli presentert etter URL-adressen.

4 KLASSEDIAGRAM

Siden webapplikasjonen PreGo! risikovurderingsverktøy prototype er mer komponent drevet vil ikke et klassediagram være veldig relevant for sidene våre. Men et klassediagram i forhold til henting og endring av data fra databasen vil være mulig å illustrere.

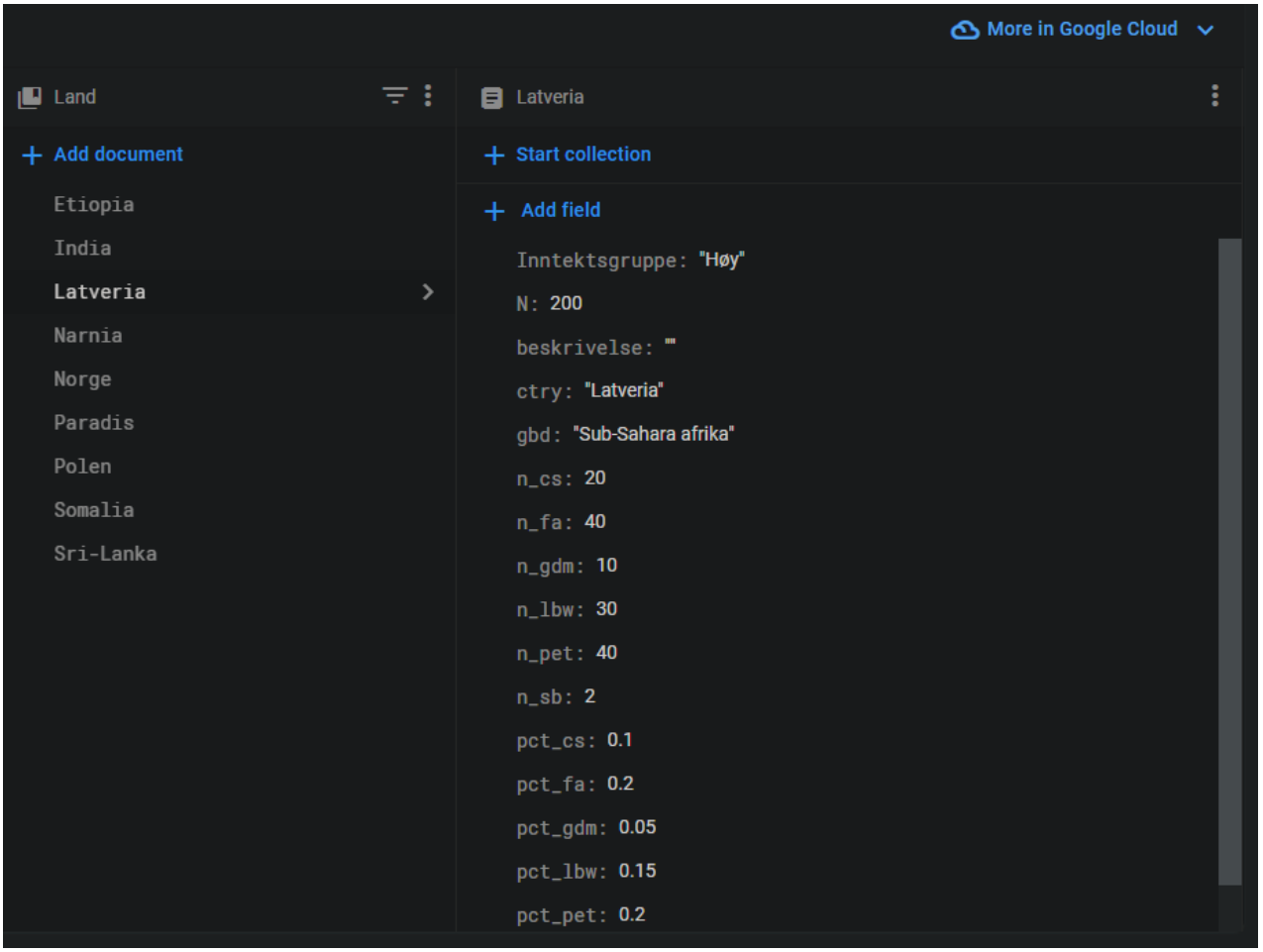


Figur 3 - Viser hvordan webapplikasjonen og databasen samhandler

Forklaring på de forskjellige variablene vil være på kapittel 5

5 DATABASEMODELL

Siden vi bruker Firestore database vil vi kunne hente dataene vi trenger fra databasen og lagre dem i konstanter eller objekter uten å måtte manipulere dataene. I Firebase database blir collection brukt som sammenlign av relaterte dokumenter. Et "document" er en samling av nøkkelverdipar som representerer en post i databasen. I databasen til PreGo! er navnet på kolleksjonen "Land" og hvert dokument i samlingen representerer et land. Et "field" er en spesifikk datadel i et dokument.



Figur 4: Viser database strukturen vår i Firestore

. I databasen til PreGO! prototypen består vært land av disse variablene/fields:

- **Inntektsgruppe:** Sier noe om inntekten til et land. I prototypen blir lav, middels og høy brukt.
- **N:** Antall personer gravide innvandrerkvinner som er registrert i databasen.
- **Beskrivelse:** En kort beskrivelse med relevant info om landet for klinikere i møte med gravide kvinner fra landet. Her kan det stå om det er krig i landet, medisinske praksiser i landet og mer. I prototypen ble det hovedsakelig brukt en generell beskrivelse av landet, ettersom at infoen
- **cntry:** Navnet på landet
- **n_:** Alle fields i databasen som starter med "n_" beskriver antall gravide kvinner som har blitt påvirket av en spesifikk risiko, for eksempel n_cs står for antall keisersnitt.
- **pct_:** Alle fields i databasen som starter med "pct_" beskriver prosent av kvinner som har blitt påvirket av en spesifikk risiko for eksempel pct_cs står for prosent keisersnitt.

NB! Dataene i databasen er fiktive.

6 Server-tjenester

Firestore database er en NoSQL dokumentbasert tjeneste som tilbyr en rekke med servertjenester for utviklere. her er noen eksempler(https://firebase.google.com/docs/firestore):

Sanntidsdata Synkronisering av data mellom klienter gjør det mulig å lage applikasjoner som gir klientene oppdatert informasjon mellom hverandre med en gang. Dette kan bli brukt i webapplikasjonen PreGo! for å gi brukere tilgang til oppdatert informasjon med en gang uten at de trenger å tenke på det. Ettersom gruppen lager en prototype har dette ikke blitt implementert i webapplikasjonen, men kan være nyttig for senere utvikling.

Skalerbarheten til Firestore tilbyr sømløst og blir tilpasset etter behovet til applikasjonen uten å gjøre noen konfigurasjoner. Ettersom webapplikasjonen PreGo! bare er en prototype er ikke dette veldig relevant utenom at gratis tjeneste tilbudet til Firestore er mer enn nok for å utvikle webapplikasjonen, men vil være relevant hvis applikasjonen skal bli videreutviklet med Firestore som database.

Sikkerheten som Firestore tilbyr er blant annet kryptering, administrasjonstilgang og revisjonslogging etter bransjestandarden.

Enkel å ta i bruk Firestore API med tilgang til mye dokumentasjon og et stort felleskap.

7 SIKKERHET

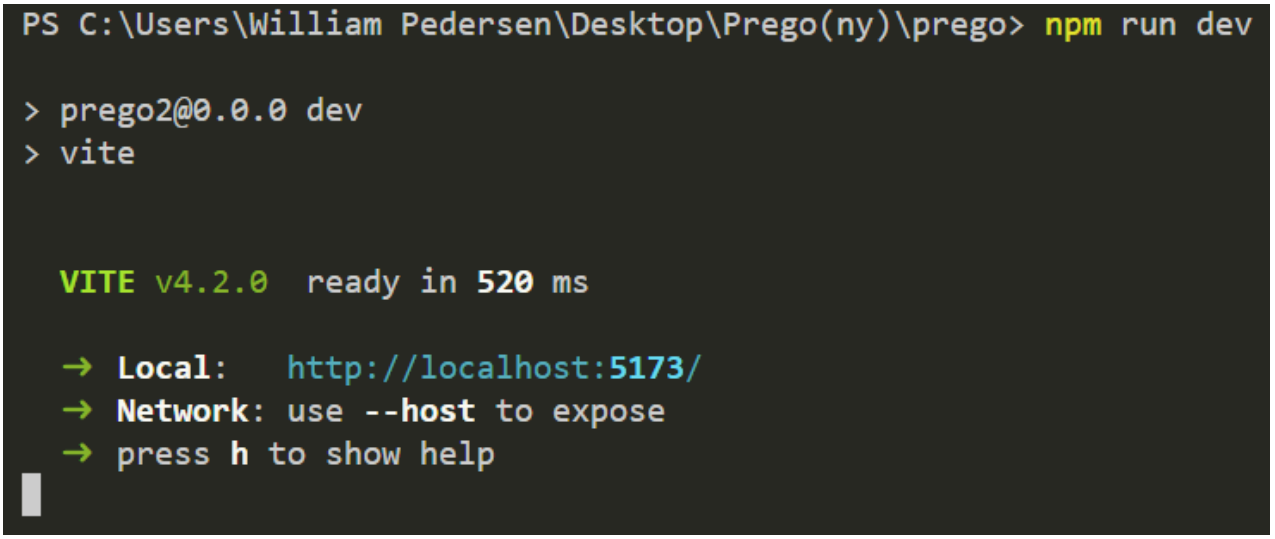
Denne webapplikasjonen gir brukerne muligheten til å logge seg inn som enten bruker eller administrator. Begge disse kategoriene har tilgang til hardkodet passord, som er lagret innenfor Firebase Authenticator-systemet. For å sikre passordene, har vi benyttet oss av krypteringsmetoden Scrypt. Videre har vi implementert Firebase som en del av vår sikkerhetsstrategi. Dette betyr at vi unngår sårbarheter knyttet til SQL-injeksjon, som kan være et stort problem for applikasjoner som benytter seg av SQL-databaser.

8 INSTALLASJON OG KJØRING

For kjøring av kode anbefaler gruppen Visual studio code som gruppen har brukt for kjøring av prosjektet.

Kjøring av koden:

1. For å kjøre koden last ned node.js:https://nodejs.org/en/download
2. deretter åpne koden i en editor
3. Åpen terminal vindu og sørg for at du er i riktig directory
4. skriv "npm install vite" i terminalen for å laste ned nødvendige pakker til prosjektet,
5. til slutt skriv "npm run dev" i terminalen for å kjøre koden og vent til du får en link for lokal server sett under:



Figur 5: Oppstart av webapplikasjonen

Når du er inne på prosjektet kan man se i "readme"-filen for å sjekke brukernavn og passord for å logge deg inn.

Feilsøking:

Hvis du får melding om at du mangler noen pakker eller linken ikke dukker opp, prøv å skriv disse inn i terminalen:

npm install chart.js

npm install react-chartjs-2

npm install vite

npm install graph.js

npm install router

npm install firebase

9 DOKUMENTASJON AV KILDEKODE

Dokumentasjonen av koden i denne applikasjonen er gjort med et særlig fokus på å dokumentere programmeringsgrensesnittet, som ofte kan være mindre leselig og forståelig enn kode som har ansvar for brukergrensesnittet.

Imidlertid har det vært mindre fokus på dokumentasjon av koden som har ansvar for brukergrensesnittet. Dette skyldes i hovedsak at denne koden ofte er mye mer lesbar og intuitiv, og dermed krever mindre dokumentasjon for å være forståelig. Generelt sett vil imidlertid all kode som utgjør applikasjonen bli dokumentert på en grundig og omfattende måte, for å sikre at den er lett å vedlikeholde og utvide i fremtiden.

Dette inkluderer både kode for programmeringsgrensesnittet og brukergrensesnittet, slik at hele applikasjonen er fullt dokumentert og tilgjengelig for utviklere som trenger å jobbe med den.

```
// Updates the data for a specific country in Firebase and fetches the updated data
const handleSaveClick = async () => {
  const docRef = doc(db, "Land", localStorage.getItem("userInput"));
  await updateDoc(docRef, editedData);
  getLand();
  setIsEditing(false);
};

// Cancels the editing mode and resets the editedData state
const handleCancelClick = () => {
  setIsEditing(false);
  setEditedData({
    Inntektsgruppe: items.Inntektsgruppe,
    gbd: items.gbd,
    beskrivelse: items.beskrivelse,
  });
};
```

Figur 6: eksempel på dokumentasjon

10 KONTINUERLIG INTEGRASJON OG TESTING

I mappen som heter “Tester” ligger det enhetstester for hjelpefunksjoner for spørringer mot databasen og logikken til noen hjelpe funksjoner. Det kan være mulig å lage en “end to end” test senere som simulerer bruker opplevelsen, men dette er bare noe ekstra vi vurderer å sette for testing av visuelle endringer i applikasjonen som riktig feilmelding ved feil brukernavn eller passord.

Funksjoner som blir testet i enhetstester:

- **getNorgeData():** Tester om dataen som blir hentet fra Norge i databasen stemmer
- **getLandData(userInput):** Tester om riktig data blir hentet etter det brukeren har valgt, tester også om GBD data blir brukt dersom det er lite data fra brukerens valgte land.
- **riskScoreCalc(land, landliste):** Tester om kalkulering av risikoscore av brukerens valgte land stemmer
- **riskScoreRang(land, landliste):** Tester om rangeringen av alle landene i databasen blir rangert riktig etter risikoscore

For å lage enhetstester har gruppen benyttet vitest som er et testrammeverk. For å kjøre testene skriver man kommandoen “npm run test” i terminalen fra rotet til prosjektet.

```
 DEV  v0.30.1 D:/prego
      Coverage enabled with c8

 ✓ src/Tester/PreGo.test.jsx (5) 939ms

 Test Files  1 passed (1)
      Tests  5 passed (5)
   Start at  13:33:34
   Duration  6.91s (transform 188ms, setup 0ms, collect 4.25s, tests 939ms, environment 0ms, prepare 1.23s)

 % Coverage report from c8
----------------------|---------|----------|---------|---------|-------------------
File                  | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
----------------------|---------|----------|---------|---------|-------------------
All files             |     100 |      100 |     100 |     100 |
 Tester               |     100 |      100 |     100 |     100 |
  testData.jsx        |     100 |      100 |     100 |     100 |
 Utils                |     100 |      100 |     100 |     100 |
  function.jsx        |     100 |      100 |     100 |     100 |
 config               |     100 |      100 |     100 |     100 |
  firebase-config.js  |     100 |      100 |     100 |     100 |
----------------------|---------|----------|---------|---------|-------------------

 PASS  Waiting for file changes...
       press h to show help, press q to quit
```

Etter at du har kjørt kommandoen vil du få en “coverage report” som viser hvilke linjer av koden som blir dekket og hvilken som ikke blir dekket og under der vil du se om testene ble godkjent eller ikke.

11 REFERANSER

‘Data access object’ (2023) *Wikipedia*. Available at: https://en.wikipedia.org/w/index.php?title=Data_access_object&oldid=1144632078 (Accessed: 24 April 2023).

Firestore (no date) *Firebase*. Available at: https://firebase.google.com/docs/firestore (Accessed: 6 May 2023).

‘Minimum viable product’ (2023) *Wikipedia*. Available at: https://en.wikipedia.org/w/index.php?title=Minimum_viable_product&oldid=1149013154 (Accessed: 6 May 2023).