Høgskulen på Vestlandet

Fakultet for ingeniør- og naturvitskap

Institutt for datateknologi, elektroteknologi og realfag

Trygg Mestringsarena
Systemdokumentasjon

Versjon <3.0>

REVISJONSHISTORIE

Dato	Versjon	Beskrivelse	Forfatter
24.04.2022	<1.0>	Første iterasjon av systemdokumentasjon	Magnus, Gard og Jaran
06.05.2022	<2.0>	Oppdatert kapittel 2, 3 og 4	Gard og Jaran
22.05.2022	<3.0>	Oppdatert arkitekturskisse, samt korrektur før innlevering	Magnus, Gard og Jaran

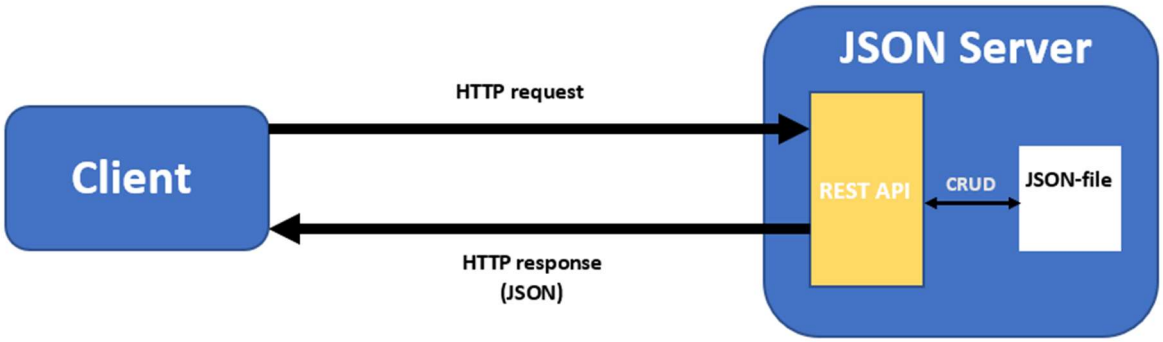
INNHOLDSFORTEGNELSE

1 INNLEDNING

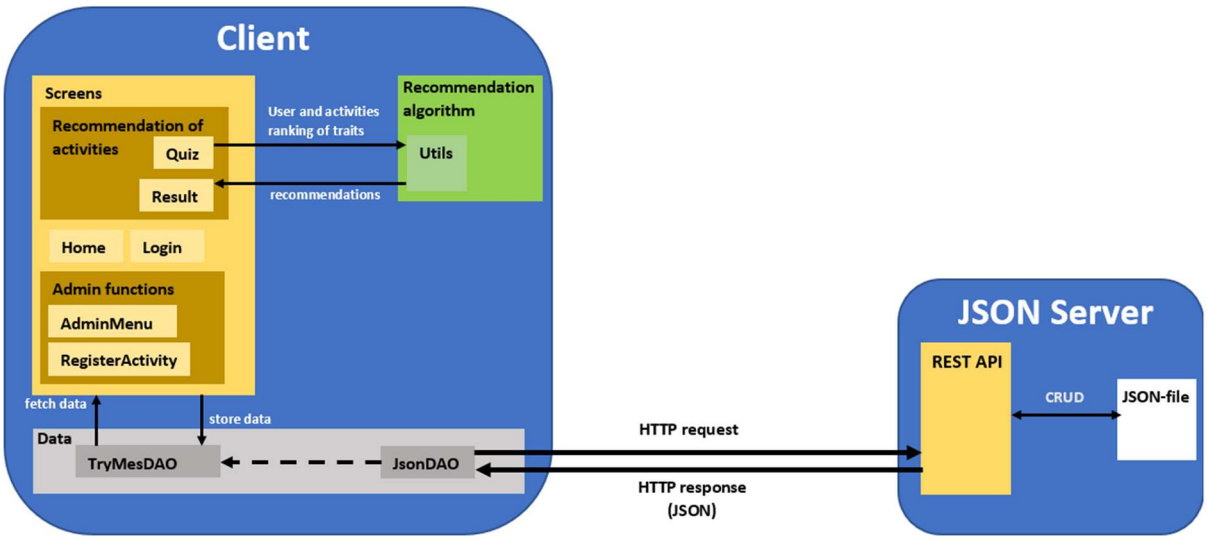
Følgende dokument inneholder systemdokumentasjon for webapplikasjonen *Trygg Mestringsarena*. Hensikten med dokumentet er å beskrive hvordan systemet er designet og laget, og det skal gi tilstrekkelig informasjon for å kunne vedlikeholde og videreutvikle systemet. Dokumentet inneholder en arkitekturskisse for systemet, en fil- og katalogstruktur for prosjektet, en databasemodell, sikkerhetstiltak i applikasjonen, og en forklaring av server-tjenester som brukes i løsningen. I tillegg inneholder dokumentet instrukser for installasjon og kjøring av applikasjonen, dokumentasjon av kildekode og en beskrivelse av hvordan systemet blir testet.

2 ARKITEKTUR

Systemet er bygget opp med en REST(REstful State Transfer)-basert klient-tjener arkitektur der klienten håndterer brukergrensesnitt og forretningslogikk, mens tjeneren håndterer datakilder som brukes i systemet. Tjeneren er implementert ved bruk av et JavaScript bibliotek som heter JSON Server (typicode, 2022) og er kun tiltenkt for prototyping av klientapplikasjonen. En beskrivelse av REST-ressursene er gitt i kapittel 6. Klienten kommuniserer med JSON-serveren gjennom et DAO grensesnitt, slik at det lettere skal være mulig å erstatte backend-løsningen for datalagring i senere tid. Figur 1 viser en arkitekturskisse for REST arkitektur ved bruk av JSON-server, mens figur 2 viser arkitekturen for *Trygg Mestringsarena* applikasjonen.



Figur 1 - Arkitekturskisse for REST arkitektur



Figur 2 - Arkitekturskisse for Trygg Mestringsarena

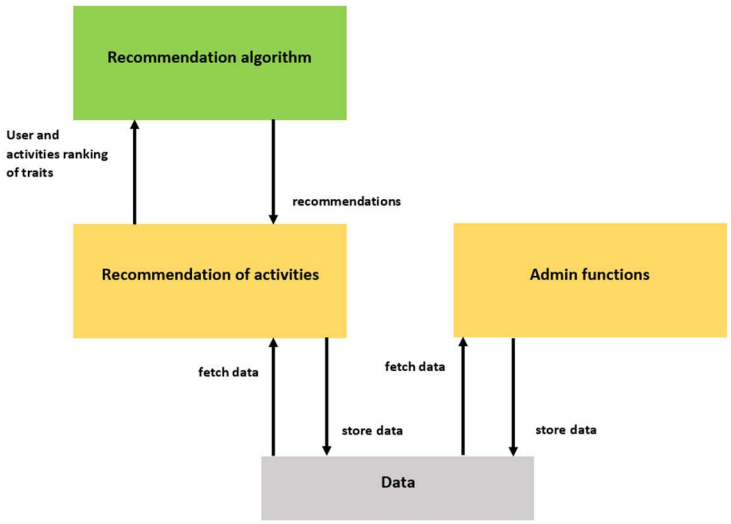
Klientapplikasjonen er bygget opp av følgende lag:

- **Recommendation algorithm:** Anbefalingsalgoritmen er implementert som en rekke funksjoner i *Utils.tsx*. Funksjonene utgjør forretningslogikken for å hente ut spørsmål til kartlegging av preferanser og registrering av aktivitet, lage anbefalinger til bruker, og bestemme hvilke aktiviteter som skal vises på resultatsiden.

- **Screens:** består av en rekke funksjonelle React komponenter som bestemmer hva som skal vises på skjermen. React komponentene inneholder diverse tilstander som brukes for å oppdatere siden basert på data som endrer seg over tid og/eller etter brukerinteraksjoner. Tilstandene blir oppdatert med verdier fra data som er lagret i backend, fra verdier som returneres av hjelpefunksjonene i *Utils.tsx,* og/eller verdier fra brukerinteraksjoner.

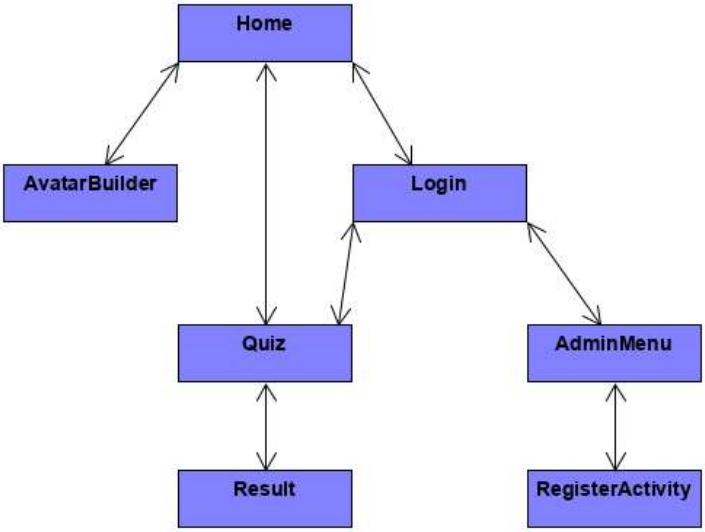
- **Data:** håndterer kommunikasjonen med den bakomliggende datalagringen brukt i systemet. Komponentene i screens-laget kommuniserer med data-laget gjennom et grensesnitt kalt *TryMesDAO*. Dette grensesnittet definerer de funksjonene som implementasjonen av DAO for datalagringen må tilby. Grensesnittet er implementert av en klasse kalt *JsonDAO* som sender HTTP forespørsler for å hente og lagre data på JSON-serveren.

En overordnet skisse av arkitekturen med fokus på de ulike funksjonene i programmet, er vist i figur 3.



Figur 3 - Oversikt over funksjoner i programmet

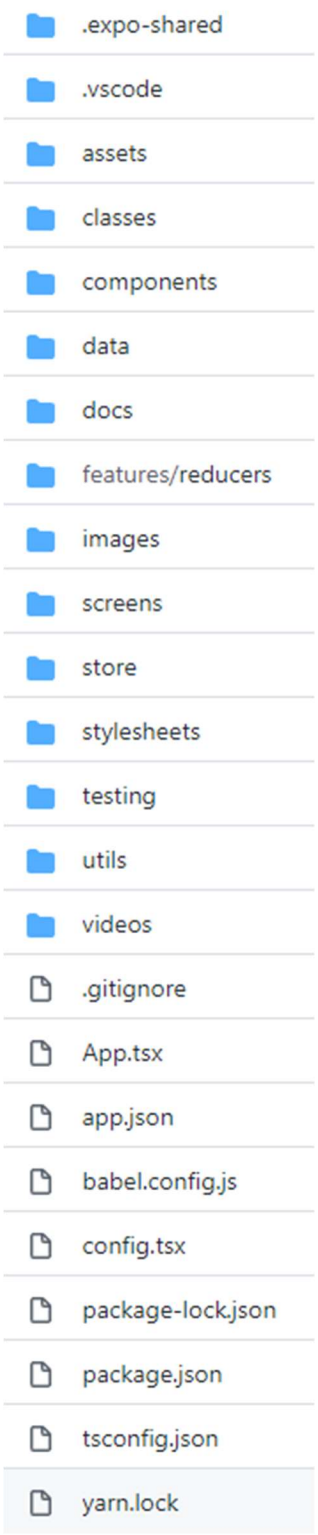
For å kontrollere navigeringen i applikasjonen er det brukt React Navigation. React Navigation er et React bibliotek som tilbyr funksjonalitet for å definere en navigasjonsstruktur (React Navigation, u.å). Navigasjonsstrukturen består av en navigator med en rekke Screen komponenter som er tilknyttet de ulike React komponentene i Screens-laget. Figur 4 viser navigasjonsstrukturen mellom de ulike sidene i *Trygg Mestringsarena* applikasjonen.



Figur 4 - Navigasjonsstruktur for Trygg Mestringsarena

3 PROSJEKTSTRUKTUR

Figur 5 viser overordnet fil- og katalogstruktur for prosjektet, med forklaringer av de viktigste elementene for prosjektet.

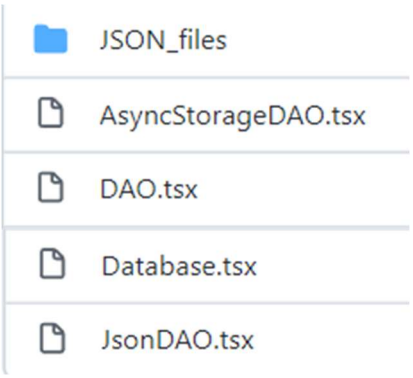


- JavaScript klasser knyttet til JSON-objektene oppgitt i kapittel 5
- React komponenter som gjenbrukes i applikasjonen
- DAO grensesnitt og DAO implementasjon, samt lokal JSON fil
- Generert dokumentasjon av kildekoden
- Reducer funksjoner for avatarbygger
- Bilder brukt i applikasjonen
- React komponenter for sidene i applikasjonen
- Tilstandstre for avatarbygger
- Globale stiler for applikasjonen
- Testeksempler og enhetstester for applikasjonen
- Konstanter og hjelpefunksjoner som brukes i applikasjonen
- Videofiler brukt i applikasjonen
- Rot-komponenten for applikasjonen
- Konfigurasjonsinnstillinger for applikasjonen
- Egendefinerte konfigurasjoner
- Metadata for prosjektet
- Konfigurasjonsinnstillinger for TypeScript

Figur 5 - Fil- og katalogstruktur

Figur 6 og 7 inneholder detaljer for mappene *data* og *screens*.

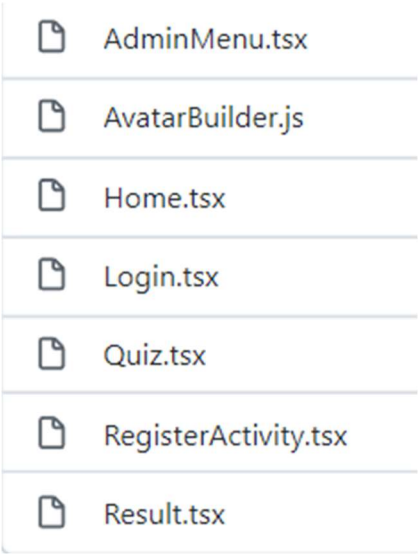
data



- Inneholder en lokal JSON fil for prosjektet
- Implementasjon av DAO ved bruk av AsyncStorage
- Inneholder en definisjon av grensesnitt for DAO
- Angir implementasjonen av DAO som skal benyttes
- Implementasjon av DAO ved bruk av JSON-server

Figur 6 - data-mappe

screens



- Administratormeny
- Avatarbygger
- Startside for applikasjonen
- Innloggingsside
- Kartleggingsside
- Side for registrering og redigering av aktivitet
- Resultatside

Figur 7 - screens-mappe

4 KLASSEDIAGRAM

Siden *Trygg Mestringsarena* applikasjonen er bygget opp som et funksjonelt program, vil ikke et klassediagram være relevant for prosjektet. En beskrivelse av entitetene fra problemdomene er imidlertid gitt i kapittel 5. Oppdeling av funksjonalitet er beskrevet i kapittel 2 *Arkitektur*.

5 DATABASEMODELL

Strukturen til dataene som brukes i applikasjonen er definert i JSON-strukturen nedenfor. Strukturen består av JSON-objektene aktiviteter ("activities"), egenskaper ("traits"), brukere ("users") og regioner ("regions").

```
{
    "activities":[
        {
            "id": number,
            "name": string,
            "picture": string,
            "website": string,
            "comment": string,
            "traitRanks":[
                {
                    "traitId": number,
                    "rank": number
                }
            ],
            "organisations":[
                {
                    "orgNr": number,
                    "name": string,
                    "email": string,
                    "phoneNumber": string,
                    "regions": [number],
                }
            ]
        }
    ],
    "traits":[
        {
            "id": number,
            "name": string,
            "category": string,
            "description": string,
            "scoreParam": boolean,
            "relevanceParam": boolean,
            "questions":[
                {
                  "text": string,
                  "picture": string,
                  "video": string,
                  "info": string,
                  "ageGroups": [string]
                }
            ]
        }
    ],
    "users":[
        {
            "id": number,
            "name": string,
            "traitRanks":[
                {
                    "traitId": number,
                    "rank": number
                }
            ],
            "recommendations":[
                {
                    "activityId": number,
                    "score": number,
                    "accuracy": number,
                    "predictionStrength": string,
                    "plausible": string
                }
            ]
        }
    ],
    "regions":[
        {
            "id": number,
            "country": string,
            "county": string,
            "municipality": string
        }
    ]
}
```

Figur 8 - JSON-struktur for Trygg Mestringsarena

6 SERVER-TJENESTER

JSON-serveren inneholder ressurser bestemt av strukturen i en JSON-fil. I *Trygg Mestringsarena* prosjektet består serveren av ressursene "/activities", "/traits", "/users" og "/regions". For å lese, skrive til eller redigere disse ressursene, er det mulig å sende HTTP forespørsler som GET, POST og PUT til serveren.

REST-ressurser:

- "/activities" - alle aktiviteter i systemet
- "/activities/{id}" - aktivitet med angitt id
- "/traits" - alle egenskaper i systemet
- "/traits/{id}" - egenskap med angitt id
- "/users" - alle brukere i systemet
- "/users/{id}" - bruker med angitt id
- "/regions" - alle regioner i systemet
- "/regions/{id}" - region med angitt id

7 SIKKERHET

De administrative funksjonene i applikasjonen krever at man er innlogget som administrator. På innloggingssiden må bruker oppgi gyldig brukernavn og passord for å komme til administratorsiden. Siden programmet ikke er koblet opp mot en SQL-database eller tar i bruk tokens, er det ikke implementert beskyttelse for nevnte eksempler.

For å gi en viss sikkerhet knyttet til administratorpassordet, er det hardkodet inn en ferdig generert SHA-3 (Secure Hash Algorithm 3) streng som representerer det korrekte passordet. Når bruker prøver å logge inn vil inntastet passord hashes ved bruk av SHA-3 og den resulterende hashverdien vil sammenlignes med den hardkodete hashverdien. Hvis brukernavn og passord er korrekt, blir brukeren videresendt til administratorsiden.

8 INSTALLASJON OG KJØRING

Installasjonsveiledning for å laste ned og kjøre prosjektet på egen maskin. For å kjøre applikasjonen er det flere programmer og pakker som må installeres på datamaskinen. Stegene blir vist i rekkefølge nedenfor.

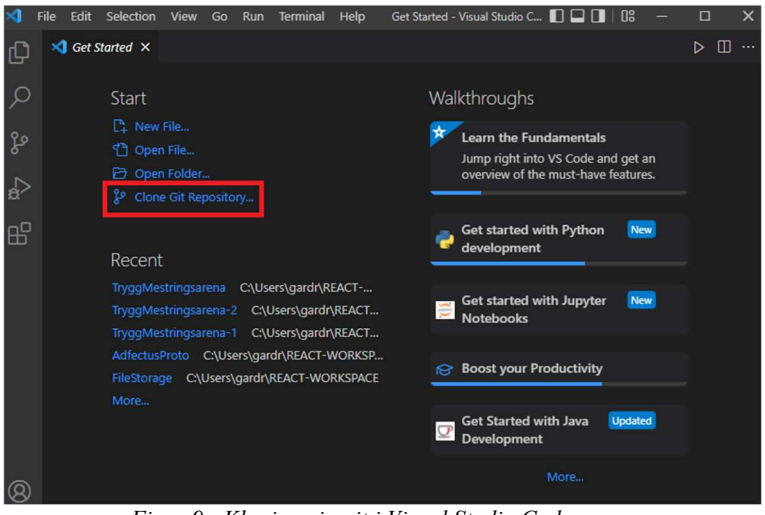
1) **Installere *Visual Studio Code***
   - Visual Studio Code er et utviklingsprogram som hjelper med å redigere, feilsøke og korrigere kode (*Hva er Visual Studio Code?*, 2022).
   - Før nedlasting, huk av «Add 'Open with Code' action…» på begge alternativene
   - **Installeringslink:** https://code.visualstudio.com/Download

2) **Installere *Node.js***
   - Node.js blir brukt for å raskt formidle informasjon til og fra en nettleser og installere ulike pakker i programmet (*Hvordan fungerer Node.JS?,* 2022).
   - **Installeringslink:** https://nodejs.org/en/

3) **Installere *Expo CLI***
   - Expo CLI blir brukt for å kjøre programmet (EXPO, u.å).
   - Installasjon skjer fra terminalen. For å få tilgang til terminalen søker man enten "cmd" eller "ledetekst" blant programmer på datamaskinen.
   - I terminalen utføres kommandoen: **npm install -g expo-cli**

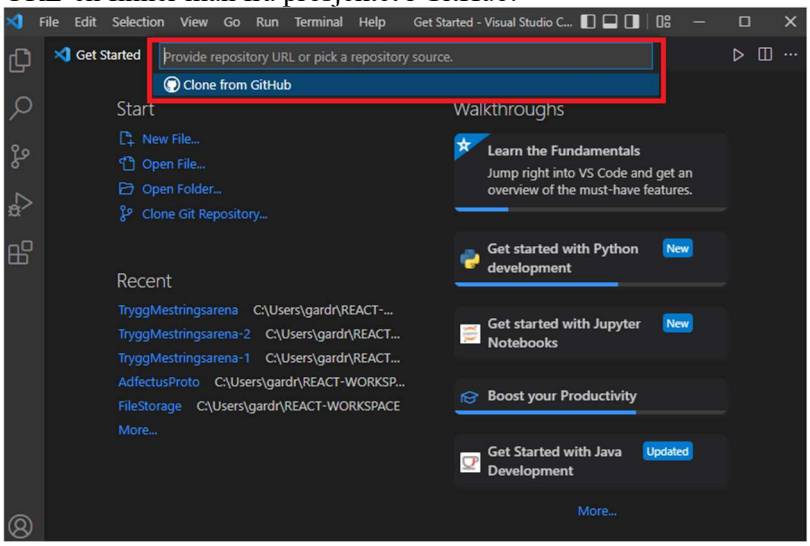
4) **Installere *Git***
   - Git blir brukt for å hente prosjektet fra GitHub
   - **Installeringslink:** https://git-scm.com/downloads

5) **Hente programmet fra GitHub**
   a) I et tomt Visual Studio Code prosjekt vil det komme opp en startskjerm. Trykk på *"Clone Git Repository"*



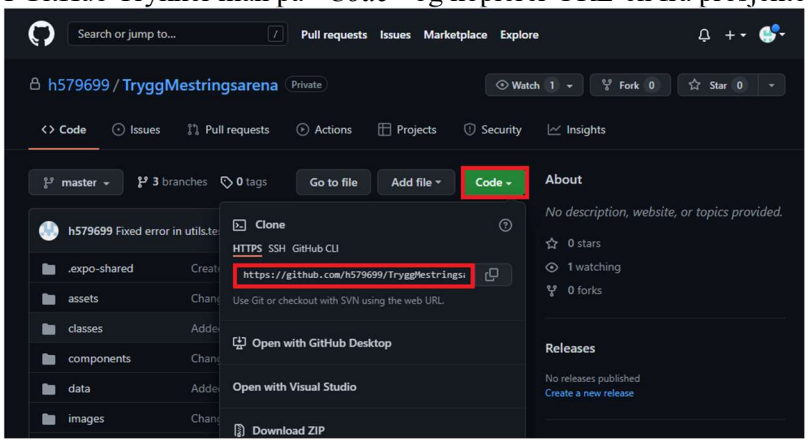
Figur 9 - Kloning via git i Visual Studio Code

b) Etter steg 5.a vil det komme opp et vindu som ber om en URL.
URL'en finner man fra prosjektet i GitHub.



Figur 10 - Inntasting av GitHub URL

c) I GitHub Trykker man på *"Code"* og kopierer URL'en fra prosjektet



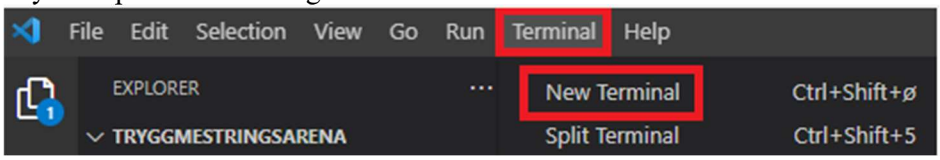
Figur 11 – URL fra GitHub

d) Lim så URL'en inn i Visual Studio Code og velg en installasjonsmappe.
Etter dette skal programmet være lastet ned på maskinen.

6) Kjøring av programmet

a) Hvis prosjektet er lukket, lokaliser installasjonsmappen fra 5.d og høyreklikk.
Finn "Open with Code" og da skal Visual Studio Code åpnes automatisk med prosjektet.

b) Trykk så på "Terminal" og deretter "New Terminal"



Figur 12 – Åpning av terminal i Visual Studio Code

c) I terminalen på Visual Studio Code skriver man så inn kommandoen: **npm install**.
- Dette installerer de nødvendige pakkene som er angitt i package.json

d) Når punkt 6.c er ferdig, utfør følgende kommando i terminalen: **npm start**
- Dette vil starte serveren og programmet

e) For å åpne programmet i nettleseren, trykk på *"w"* i terminalen

7) Kjøre applikasjonen på tablet/mobil

a) For å kunne kjøre programmet på en ekstern enhet, må applikasjonen Expo Go lastes ned på enheten. Installering er tilgjengelig fra Apple Store og Google Play.

b) Etter punkt 6.d vises en QR-kode i terminalen. Skann koden med Expo Go (Android) eller kamera applikasjonen (iOS). Fra kamera applikasjonen vil en notifikasjon om å åpne Expo Go komme opp. Trykk på denne.

c) Prosjektet bygges på enheten og vil etter vellykket bygging kunne brukes.

9 DOKUMENTASJON AV KILDEKODE

Dokumentasjon av kildekode er et viktig verktøy for å skape en felles forståelse for koden som blir skrevet. Det er viktig at teksten er enkel og forklarende for andre, noe som er spesielt viktig når man har flere utviklere i et felles prosjekt (Chalimov, 2017).

I prosjektet blir TSDoc benyttet i kildekoden. TSDoc inneholder flere tagger som blant annet kan brukes for å dokumentere parametere og returverdier til funksjoner. Figur 13 viser et eksempel på dette. All dokumentasjon er skrevet på engelsk, noe som er standard for programutvikling.

```
/**
 * Removes all non-numeric characters from a string
 * @param {string} text
 * @returns {string} text only containing numbers
 */
const removeChars = (text:string): string =>{
    return text.replace(/[^0-9]/g, '');
};
```

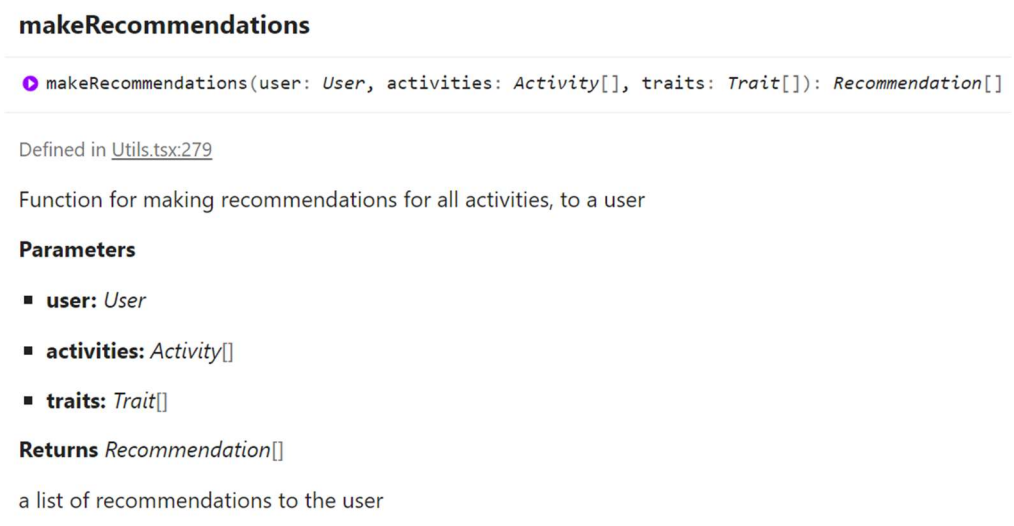
Figur 13 - Eksempel som viser TSDoc i prosjektet

En annen form for dokumentasjon i koden, er små kommentarsetninger som består av "//" etterfulgt av tekst. Denne formen for dokumentasjon blir ofte brukt for å forklare hensikten til ulike variabler, konstanter og metodekall.

```
//Receives the width and height of userscreen
const screenWidth = Dimensions.get('window').width;
const screenHeight = Dimensions.get('window').height;
```

Figur 14 - Dokumentasjon av konstanter i prosjektet

Mappen *docs* i prosjektet inneholder generert dokumentasjon av enkelte av filene i applikasjonen. For å generere dokumentasjon har gruppen benyttet seg av TypeDoc som gjør det mulig å generere dokumentasjon fra kommentarer i koden i TypeScript prosjekter (TypeDoc, u.å).
Kommandoen **npx typedoc --out <mappe for lagring av dokumenter> <startpunkt>** kan brukes for å generere dokumentasjon fra et gitt startpunkt i prosjektet. Resultatet er HTML sider som inneholder dokumentasjonen. Figur 15 viser et eksempel på generert dokumentasjon for funksjonen *makeRecommendations*.



makeRecommendations

```
makeRecommendations(user: User, activities: Activity[], traits: Trait[]): Recommendation[]
```

Defined in Utils.tsx:279

Function for making recommendations for all activities, to a user

Parameters

- **user:** *User*
- **activities:** *Activity*[]
- **traits:** *Trait*[]

Returns *Recommendation*[]

a list of recommendations to the user

Figur 15 - Generert dokumentasjon for funksjonen makeRecommendations

10 TESTING

Mappen *testing* inneholder enhetstester for hjelpefunksjoner knyttet til gjennomføring av kartleggingen, anbefaling av aktiviteter og visning av resultat. Det er ikke blitt opprettet tester for innlogging, registrering/justering av aktiviteter og datalagringsfunksjoner.

Enhetstestene dekker:

- Skåren er 0 hvis aktiviteten og bruker har lik rangering av alle egenskaper
- Maksimal skåre er lik antall skåre-egenskaper ganget med 2
- Treffprosenten er 100% dersom aktivitet og bruker har samme rangering av egenskaper.
- Anbefalingsstyrken er høy/middels/lav i forhold til antall ubesvarte rangeringer.
- En aktivitet ikke er relevant dersom aktivitetens rangering er 3 og brukerens rangering er 1 på en av relevanse-egenskapene
- En aktivitet er relevant dersom skåren av relevanse-egenskapene er mindre enn 2.
- Aktuell måltallet får korrekt verdi i forhold til brukers rangeringer.
- En perfekt sammenligning med en aktivitet gir 100% på anbefalingen av aktiviteten.
- Alle egenskaper har -1 som verdi før gjennomført kartlegging.
- Registreringen av svar fra kartleggingen gir riktig verdi til riktig egenskap.
- Om "getFilledQuestions" setter rangeringene av egenskaper lik rangeringen til en gitt aktivitet.
- Brukerens svar på ja/nei spørsmål er korrekt skrudd av eller på i resultatet.
- Aktiviteter som har ja på en egenskap som en bruker har svart nei på, er skrudd av.

For å lage enhetstester har gruppen benyttet seg av testrammeverket Jest (Jest, u.å). For å kjøre testene utfører man kommandoen *npm run test* eller *npm test* i terminalen, fra roten av prosjektet.

```
  √ Max score if user rankings and activity rankings are completely different (1 ms)
  √ Accuracy is 100.00 if activity and user rankings are the same (score is 0) (1 ms)
  √ Prediction strength is "Høy"/high if the user has less than 3 unanswered score parameters, "Middels"/medium if
less than 6, "Lav"/low if less than 9, "Ugyldig"/invalid otherwise (7 ms)
  √ An activity is not relevant if one of the relevance parameters is 3 and the user ranking is 1 (2 ms)
  √ An activity is relevant if the score of the relevance parameters is less than 2 (1 ms)
  √ Relevance is "Ja"/yes if isRelevant, "Nei"/no if notRelevant, "Kanskje"/maybe otherwise (1 ms)
  √ Perfect match is perfect recommendation
  √ By default all rankings in questionAnswers are -1
  √ registerRankings registers the question answers to the corresponding trait ranks of user or activity
  √ getFilledQuestionAnswers sets the answers by default to be equal to the activity ranking of the traits (2 ms)
  √ The yes/no traits that the user has answered no/0 to, is toggled off
  √ Activities answering yes to a toggled off trait, is also toggled off (1 ms)

Test Suites: 1 passed, 1 total
Tests:       14 passed, 14 total
Snapshots:   0 total
Time:        1.446 s
Ran all test suites.
```

Figur 16 - Resultat av kjøring av enhetstester

11 REFERANSER

- Chalimov, A. (2017) *Source code documentation best practices*. Tilgjengelig fra: https://easternpeak.com/blog/source-code-documentation-best-practices/ (Hentet: 20.04.2022).
- EXPO (u.å) *EXPO CLI.* Tilgjengelig fra: https://docs.expo.dev/workflow/expo-cli/ (Hentet: 21.04.2022).
- *Hva er Visual Studio Code?* (2022) Tilgjengelig fra: https://no.education-wiki.com/3958588-what-is-visual-studio-code (Hentet: 21.04.2022).
- *Hvordan fungerer Node.JS?* (2022) Tilgjengelig fra: https://no.education-wiki.com/8915130-how-node.js-works (Hentet: 21.04.2022).
- Jest (u.å) *Jest - Delightful JavaScript Testing.* Tilgjengelig fra: https://jestjs.io/ (Hentet: 18.04.2022).
- React Navigation (u.å) *Getting Started.* Tilgjengelig fra: https://reactnavigation.org/docs/getting-started/ (Hentet: 20.04.2022).
- TSDoc (u.å) *What is TSDoc?* Tilgjengelig fra: https://tsdoc.org/ (Hentet: 20.04.2022).
- TypeDoc (u.å) *TYPE DOC*. Tilgjengelig fra: https://typedoc.org/ (Hentet: 24.04.2022).
- typicode (2022) *JSON Server.* Tilgjengelig fra: https://github.com/typicode/json-server (Hentet: 13.04.2022).