# Dokumentasjon

## Introduksjon

I denne delen av oppgaven går jeg nærmere inn på applikasjonen min i detalj. Her har jeg dokumentert koden i prosjektet mitt med bilder og forklaringer. Jeg har skrevet litt om hva koden gjør og hvordan den brukes i applikasjonen.

## Teknisk oppsett og konfigurasjon

package.json er en konfigurasjonsfil som inneholder det som trengs for å installere og kjøre prosjektet mitt. I tillegg til det standard react oppsett har jeg lagt til dependencies for auth0, designsystemet og jest dette er alle ulike avhengigheter som mitt prosjekt trenger. Her ser man også hvilken versjon jeg bruker.

Nedenfor er ett utklipp av dependencies fra package.json.

Et bilde som inneholder tekst, skjermbilde, meny

KI-generert innhold kan være feil.

Index

Setter opp react prosjekte og bruker react router for navigasjon mellom siden som i dette tilfelle er oppslagside og detaljside, med bruk av auth0Provider er den pakket rundt hele prosjektet sånn at hele siden har tilgang til autentiseringskonteksten.

Et bilde som inneholder tekst, skjermbilde

KI-generert innhold kan være feil.

## Innlogging

En komponent for Inn logging og en komponent for ut logging, begge knappenen som bruker auth0 for å logge inn eller ut brukere

Et bilde som inneholder tekst, programvare, Multimedieprogramvare, skjermbilde

KI-generert innhold kan være feil.

Et bilde som inneholder tekst, skjermbilde, Font

KI-generert innhold kan være feil.

## 

## Api forespørsler

Virksomhet

Satt opp to typer for de forskjellige responsene jeg får fra enhetsregisteret sitt api, en for en enkel virksomhet og en for flere virksomheter. FetchVirksomhetRespons brukes for å hente flere virksomheter til oppslagsiden, mens fetchSingelVirksomhetsRespons brukes når det trengs kun en virksomhet til detaljsiden.

Et bilde som inneholder tekst, skjermbilde, Font

KI-generert innhold kan være feil.

Fetch kall som fjerner mellomrom fra organisasjonsnummeret og henter data fra enhetsregisteret på et spesifikt organisasjonsnummer. Blir brukt når bruker går inn på en virksomhet for å få mere informasjon.

Et bilde som inneholder skjermbilde, tekst, programvare, Multimedieprogramvare

KI-generert innhold kan være feil.

Fetch kall som henter alle virksomheter som har med parametere for å kunne gjøre et søk etter virksomhets navn, side tall for paginering og val av hvor mange virksomheter man vil se om gange på oppslagsiden.

Et bilde som inneholder tekst, skjermbilde, programvare

KI-generert innhold kan være feil.

Fetch kall som tar valgt kommune nummer å gjør et fetch kall mot alle virksomheter gå gir ut virksomheter med valgt kommune.

Et bilde som inneholder tekst, skjermbilde, programvare, Multimedieprogramvare

KI-generert innhold kan være feil.

Fetch kall som henter ut alle kommuner som blir brukt får å gi en oversikt over kommune nummer og navn for kommune filter. For å sette opp de 2 andre filtrene er det samme prosess for å få tilgang på dataen.

Et bilde som inneholder tekst, programvare, Multimedieprogramvare, skjermbilde

KI-generert innhold kan være feil.

## Felles komponenter

Detail-Row

Detail-row.tsx er en komponent som setter opp en description list med props som defineres når den blir tatt i bruk, så når man tar den i bruk i for eksempel detaljside.tsx så trenger man kun å skrive inn dataen man vil vise fram, altså lable og children. Dette er et eksempel på en felleskomponent, det vil si at den kan gjenbrukes i andre deler av prosjektet med egendefinert innhold. Fordelen med felleskomponenter er at man trenger kun å oppdatere koden en plass og den vil endres alle steder den blir brukt. Dette gir bedre oversikt over koden, enklere vedlikehold og potensielt mindre kodelinjer.

Et bilde som inneholder tekst, skjermbilde, programvare, Multimedieprogramvare

KI-generert innhold kan være feil.

## Oppslagside

Bruker useState for å oppdatere og lagre data i komponenten, gjøre brukergrensesnittet interaktivt og for å kontrollere hva som vises.

Har da en del useStates som dekker litt forskjellige behov, lagre data fra api og database (virksomheter og komunner), lagrer forskjellig bruker input (searchValue og valgtKommune) og lagring av feilmelding (error).

Konstantene fra useAuth0 gir informasjon om bruker og om man er logget inn eller ikke.

Et bilde som inneholder tekst, skjermbilde, Font

KI-generert innhold kan være feil.

Alle funksjonene bruker async/await for å hente data asynkront og håndterer både suksess og feil. FetchVirksomhetData henter en liste med virksomheter, Fetch kommuner henter en liste over kommuner og filter kommune henter virksomheter filtrert på valgt kommune i filtret.

Et bilde som inneholder tekst, skjermbilde, programvare

KI-generert innhold kan være feil.

Hær kjøres de forskjellige funksjonene i en useEffect som har ulike dipendencies etter hvor ofte det skal opptateres og hva som skal endres for å kreve et nytt kall

Et bilde som inneholder tekst, skjermbilde, Font

KI-generert innhold kan være feil.

Håndterer søk, hvis searchValue endres vil det gjøres en ny fetch med bruker inputen og searchResult staten blir oppdatert med ny data.

Et bilde som inneholder tekst, skjermbilde, programvare, Multimedieprogramvare

KI-generert innhold kan være feil.

Hånterer endring på sider i pagineringen når bruker vi navigere vidre i resultatene. HandelClear brukes når bruker nullstiller søket sitt, da vil de fleste useStatene nullstilles og vise helet søket uten noe filtrering på filter eller søk

Et bilde som inneholder tekst, skjermbilde, Font

KI-generert innhold kan være feil.

## Detaljside

Henter organisasjonsnummeret fra URL-en ved hjelp av useParams. Deretter, i en useEffect, kalles funksjonen fetchSingleVirksomhet med det gitte organisasjonsnummeret til valgt virksomhet. Funksjonene er asynkrone (async) slik at data kan hentes uten å blokkere resten av applikasjonen. Bruker await for å vente på at dataene skal lastes inn før de håndteres videre.

Et bilde som inneholder tekst, skjermbilde, programvare

KI-generert innhold kan være feil.

## Database

Database oppsett

Satt opp en enkel database som lagrer user\_id, filter\_id og filter\_name.

Siden det er satt opp at User\_ id skal bli lagret i databasen blir jo personver og sikkerhet dratt inn i dette, da det som lagres er user.sub som er en unik identifikator for hvær bruker, da kan det være greit å informere brukerern at når den velger å logge inn å lagre sine egne filter vil denne dataen lagres i databsen

Et bilde som inneholder tekst, skjermbilde, programvare, Multimedieprogramvare

KI-generert innhold kan være feil.

Server.js

Her har jeg satt opp et endepunkt for lagring av selv lagde filter som lagrer userId, filterId og filterName i databasen vise på bilde ovenfor. Planen videre i server.js var å sette opp mulighet for å slette selv lagde filter. Lagring av egendeffinerte filter fungrer ikke helt et oppstår en feil når man skal lagre noe jeg ikke har hadde tid til å få fikset.

Et bilde som inneholder tekst, skjermbilde, programvare, Multimedieprogramvare

KI-generert innhold kan være feil.

ValidateParams er satt opp sånn at når man setter opp flere endepunkt skal man slippe å ha for mye duplisert kode å forholde seg til

Et bilde som inneholder tekst, skjermbilde, Font

KI-generert innhold kan være feil.

Handel filter  
Lagrer userId, filterId og filterName i databasen på endepunkte filter som er satt opp i server.js

Et bilde som inneholder tekst, skjermbilde, programvare

KI-generert innhold kan være feil.

## Testing

Setuptests.ts

Dette oppsette kreves for å ta i bruk jest som er et testrammeverk.



snapshotResolver

Bestemmer hvor snapshot filene blir plassert i prosjektet

Et bilde som inneholder tekst, Font, skjermbilde

KI-generert innhold kan være feil.

Jest.setup

Denne koden sørger for at disse globale variabler er tilgjengelig i testmiljøet som jest kjøres i.

Et bilde som inneholder tekst, skjermbilde, Font, line

KI-generert innhold kan være feil.

jest.config.js

Dette er en konfigurasjonsfil som gjør det mulig å kjøre tester i et miljø som simulerer en nettleser-DOM (Document Object Model), noe som er nødvendig for å teste React-komponenter. Den spesifiserer også hvordan testdekning skal samles inn, og hvordan mock-regler skal håndtere importerte filer som bilder, fonter og CSS, slik at testene ikke feiler på grunn av slike avhengigheter.

Et bilde som inneholder tekst, skjermbilde, programvare, Multimedieprogramvare

KI-generert innhold kan være feil.

Mock – testdata

Et bilde som inneholder tekst, skjermbilde, programvare

KI-generert innhold kan være feil.

Tester

Fetch-mock brukes i forbindelse med Jest-testing for å simulere et fetch-kall med forhåndsdefinert responsdata. Den aktiverer mocking av fetch og sørger for at mocken tilbakestilles mellom testene

Et bilde som inneholder tekst, Font, skjermbilde, nummer

KI-generert innhold kan være feil.

Her er et eksempel på en enhets test av fetch kall hvor jeg bruker mock data.

Et bilde som inneholder tekst, skjermbilde, Font

KI-generert innhold kan være feil.

SnapShot test som rendrer hele detaljsiden. Snapshot tester er nyttig for å oppdage visuelle eller strukturelle endringer i komponenten.

Et bilde som inneholder tekst, skjermbilde, programvare

KI-generert innhold kan være feil.

## Universell utforming og WCAG

Har brukt ulike automatiserte testverktøy som WAVE og lighthouse for å teste primært WCAG. Disse verktøyene dekker rundt 20-40% av tilfellene, det er derfor viktig å teste manuelt med skjermleser og tastatur i tillegg.

Wave evaluation tool tester hovedsakelig Overskriftsstruktur, ARIA-roller, attributter, semantisk HTML, alternativ-tekst på bilder og feil bruk av elementer eller manglede attributer.

Et bilde som inneholder tekst, skjermbilde, programvare, Dataikon

KI-generert innhold kan være feil.

lighthouse

Lighthouse tester både ytelse, tilgjengelighet, best praksis og SEO. Det gir et helhets overblikk av hele siden og de forskjellige aspektene.

Som man ser på bildene under er det tydelig at mobilvisning får en lavere score på performance enn desktop. Dette skyldes blant annet at flere ressurser må lastes på små skjermer og når lighthouse simulerer på mobil bruker den begrenset cpu og tregere nettverk.

Desktopvisning

Et bilde som inneholder tekst, skjermbilde, programvare, Font

KI-generert innhold kan være feil.

Mobilvisning

Et bilde som inneholder tekst, skjermbilde, programvare, Nettside

KI-generert innhold kan være feil.

# Evaluering

Oppgaven gikk ut på å sette opp en moderne og brukervennlig applikasjon for den lokale avisa med hjelp av data fra brønnøysundregistrene og åpne data med innlogging, søk og filtrering, For å gi rask tilgang til relevant informasjon uten å måtte sette opp filter hver gang.

Planlegging

Planleggings delen gikk veldig bra fikk satt opp en plan for hvilken teknologi jeg skulle ta i bruk samt litt om andre muligheter, ble satt opp en skisse og en plan for framdrift.

Gjennomføring

Planen for gjennomføringen ble fulgt ganske så bra det ble litt knot i oppsett av fetch kall, det ble brukt lengre tid på grunn av feil i koden og små problemer vart støtt på i oppsete av resultCard. Det var blant annet noen små feil i oppsettet av interface for å ta i bruk forretningsaddresse det ble til slutt løst når jeg fikk litt hjelp til å finne ut hvordan jeg skulle løse feilen med typene i interfacene og informasjonen som skulle vises. Det gjorde at noen oppgaver ble litt forsyd men det ble fort fikest opp i da dert var beregnet litt merer tid enn nødvendig på oppsettet av infromasjon i detaljsiden og routing mellom oppslag og detaljsiden de 2 oppagavene ble byttet rekkefølge på da det virket mere logiskt å få opp routingen mellom de 2 sidene før det skulle informasjon og en fungerende tilbake knapp inn. Vider ble planen fulgt til etter oppgave 14 med å sette opp databasen var rundt den tiden jeg innså at det ikke var lagt opp noe oppgave for fungerende filtrering så da måtte det prioriteres etter databasen var færdig satt opp da det ble vanskeling å legge opp for egendefinerte filter hvis det ikke var noen filter å bruke.

Avvik

Etter hvert i gjennomføringe la jeg merke til at det var noen avvik/mangler, jeg hadde gjort en liten feil tolkning av hvordan filtrene skulle se ut og settes opp, når jeg begynte å sette opp utsene for filtreringen innså jeg at det ble helt feil at det kun skulle ligg 1 filtrerings meny når det var 3 forskjellige ting som brukeren skulle kunne filtrere på så da ble det lagt inn 2 nedtrekks menyer, en for kommuner og en for organisasjonsformer mens filteret for konkurs ble lagt inn som radioknapper. Det andre avvike var at jeg så det hadde blit glemt å sette opp en oppgave for å få filtrene til å skulle fungerer, dette ble da oppdaget litt senere, så på grunn av liten tid ble det kun lagt inn funksjonalitet på filtrering av kommuner.

Jeg ble ganske fornøyd med sluttproduktet. Applikasjonen fungerte nesten helt etter alt som skulle har vært med det som ikke ble helt ferdig med det totalt oppsette av lagring å sletting av egendeffinerte filter i databasen

Dokumentasjon

Dokumentasjonen ble ganske lett og oversiktlig Noe forbedringspotensial finnes i å utdype enkelte variabler og forklare eventuelle avhengigheter eller forutsetninger.

Egenvurdering

Kunnen nokk vært skrevet litt mere på noen punkter i evalueringen men tiden for fort fra meg så en tanke til en annen gang blir å sette av litt mere tid på både evalueringen og litt mere tid til dokumentasjonen også

# Kilder

<https://www.uutilsynet.no/uubloggen/verktoy-testing-av-universell-utforming/2567>

<https://www.uutilsynet.no/veiledning/veiledning/704>

<https://data.brreg.no/enhetsregisteret/api/dokumentasjon/no/index.html#tag/Enheter/operation/hentEnheter>

<https://data.brreg.no/enhetsregisteret/api/dokumentasjon/no/index.html#tag/Kommuner/operation/hentKommuner>

<https://expressjs.com/>

<https://node-postgres.com/>

Noen deler av teksten har jeg fått hjelp av chat-gpt til å formulere, og enkle forklaringer på kode og begreper.