Eksamensdokument for gruppe 38

Gruppemedlemmer:

- Tobias Berger Christoffersen
- Ehsam Josefzai
- Madelen Kristiansen
- Sebastian Magnor Nordstrand

Github brukernavn:

- **SebastianMN12** - Sebastian Magnor Nordstrand
- **Madelekr** - Madelen Kristiansen
- **vanDerBC** - Tobias Berger Christoffersen
- **ehsamjo12** - Ehsam Josefzai

Lenke til Github Repository:
https://github.com/Madelenkr/Eksamen_UIN25
Lenke til Sanity:
https://www.sanity.io/organizations/otWnZFbcg/project/eqbspp1a?orgId=otWnZFbcg

Vansklighetskrav:

Vi valgte å gå for vanskelighetsgrad B.

Gruppe arbeidet:

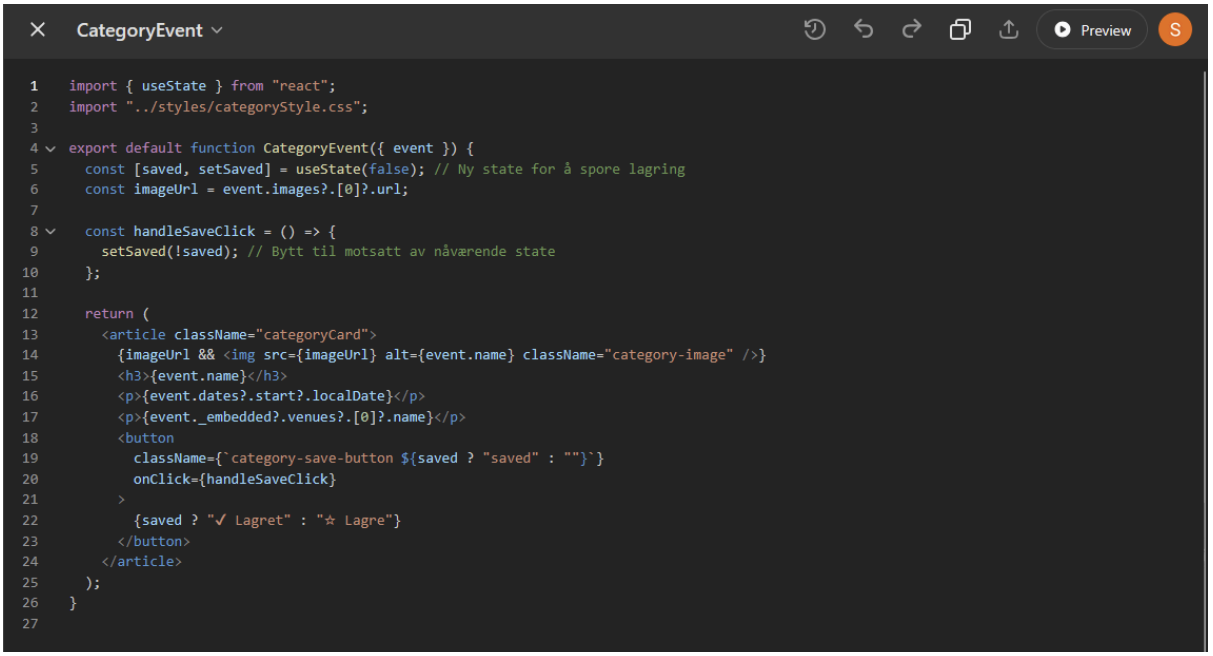
Gruppen har valgt å møtes fysisk for å oppnå bedre oversikt over kodearbeidet, samarbeidet, fordele oppgaver og forbedre den generelle kommunikasjonen. Ehsam har i liten grad deltatt på disse møtene, noe som har gjort det utfordrende å opprettholde god kommunikasjon med ham, dette har vært litt utfordrende. Resten av

gruppen har hatt et godt samarbeid.

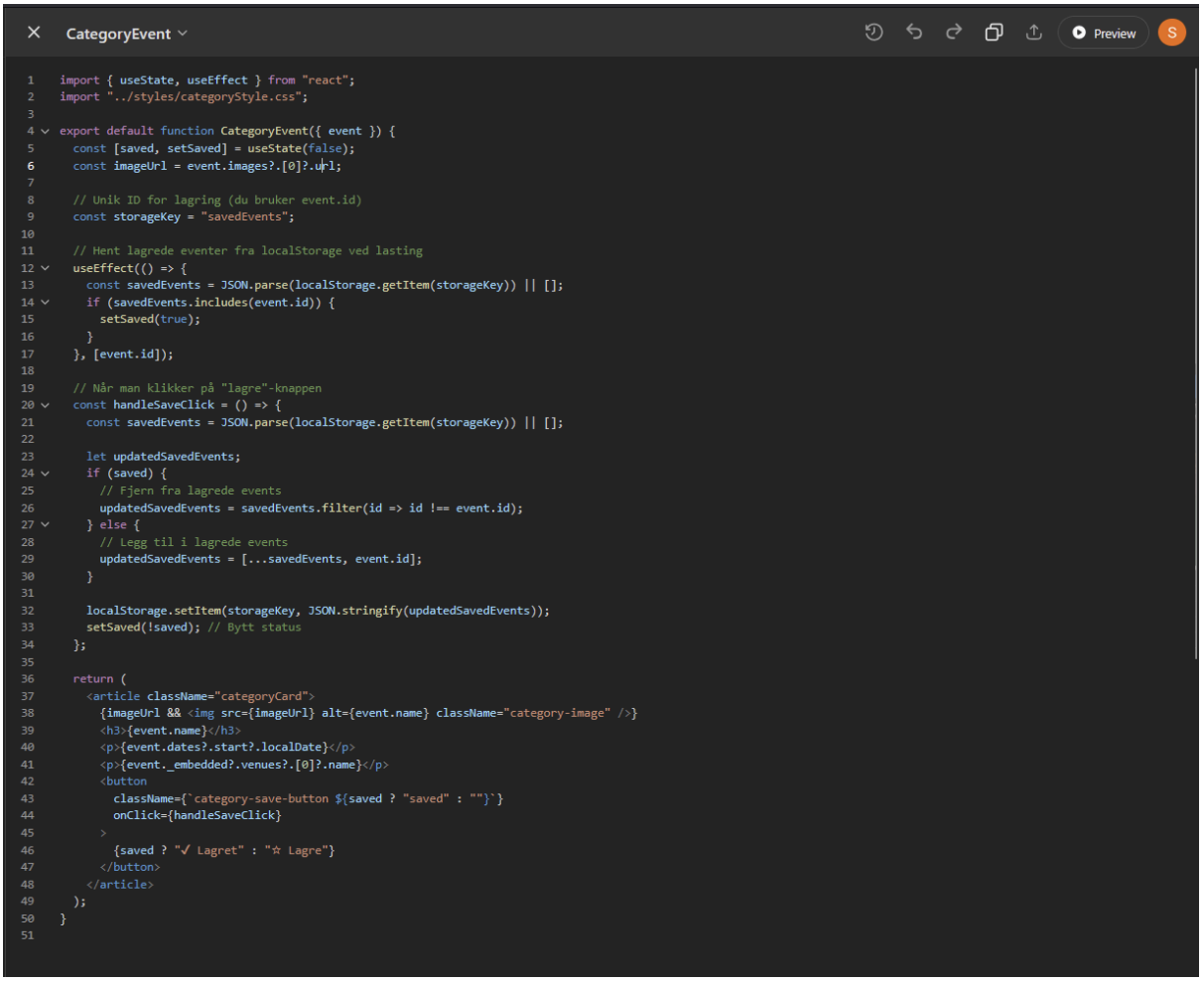
Kilder:

Kilde for CategoryEvents.js:
Vi brukte ChatGPT akkurat til den funksjonen å legge til en knapp som ble lagret i localstorage, da for å emulere at den har blitt lagret i en "ønskeliste".

Spørringen min var følgende:
"Jeg har disse sidene og skal ha en knapp på categoryevent som endrer seg når du trykker på den. tenk på det som en "lagre" knapp som visuelt viser at du har trykket på den, her er koden for sidene:"
Deretter limte jeg inn det vi hadde av kode på CategoryEvent.js fra før av.



Etter denne spørringen, spurte jeg om GPT kunne endre det så den ble lagret i localstorage med spørringen: "Hvordan får jeg det til i localstorage?"
og jeg fikk da følgende svar:



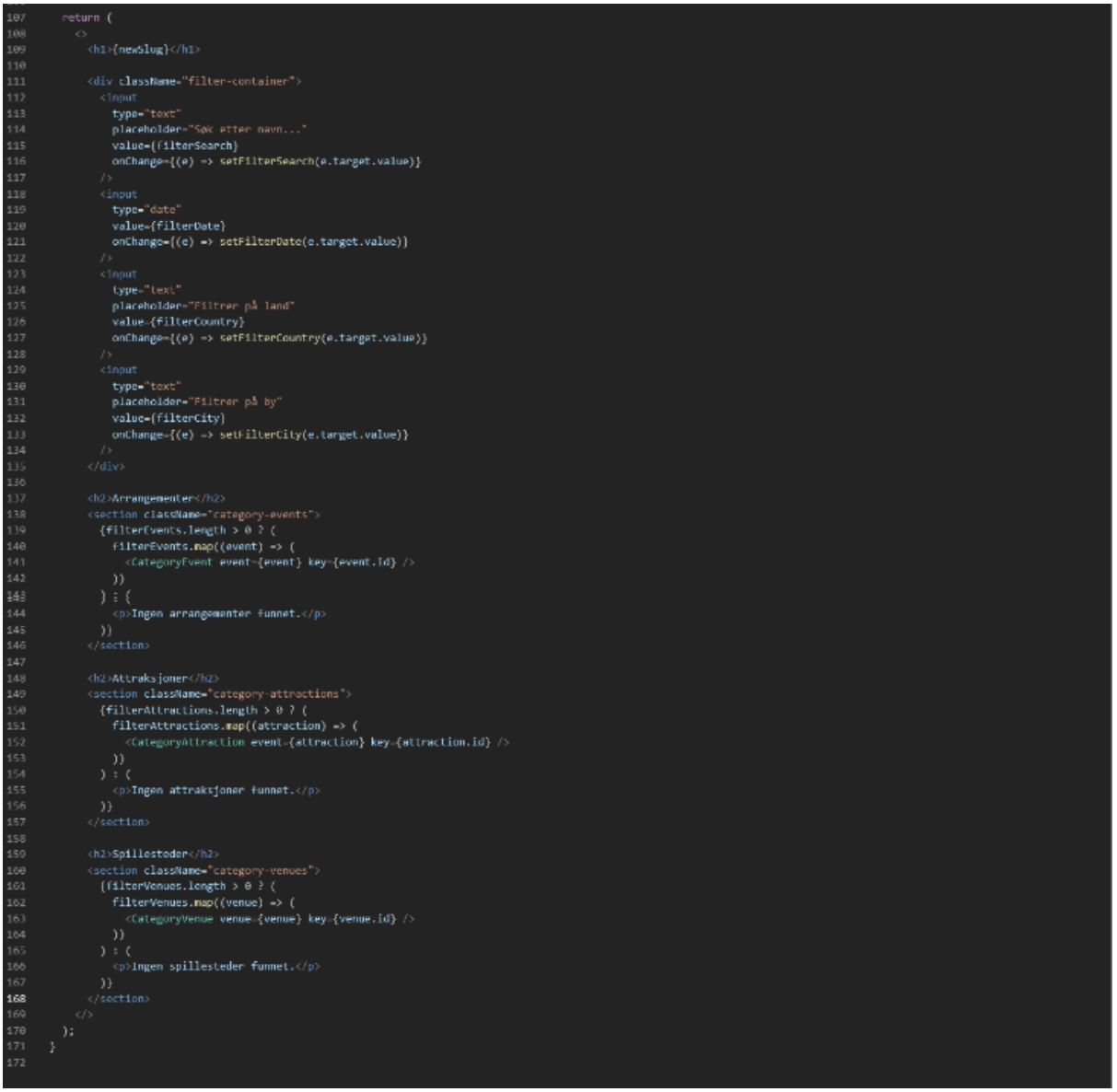
Denne biten med kode er for å ha en knapp som “lagrer” et valgt arrengement i en “ønskeliste”. koden jeg fikk, sørger for at denne funksjonen blir lagret i localstorage, sånn at den knappen forblir trykket selv med en refresh av siden. Jeg har også i etterkant modifisert koden etter behov.

Kilde for filtrering på CategoryPage:
Inne på CategoryPage.js så skal det være en funksjon for å filtrere innholdet basert på dato, land og by.
Jeg ga GPT denne spørringen: “På denne siden skal jeg ha en filtrering hvor jeg kan dato, land og by, hva må gjøres?” deretter ga jeg koden vi hadde for CategoryPage opp til det punktet. GPT ga meg da følgende kode:

CategoryPage

Preview

```jsx
import { useEffect, useState } from "react";
import { useParams } from "react-router-dom";
import CategoryEvent from "./CategoryEvent";
import CategoryAttraction from "./CategoryAttraction";
import CategoryVenue from "./CategoryVenue";
import "../styles/categoryPage.css";

export default function CategoryPage() {
  const { slug } = useParams();
  const [events, setEvents] = useState([]);
  const [attractions, setAttractions] = useState([]);
  const [venues, setVenues] = useState([]);
  const [filterSearch, setFilterSearch] = useState("");
  const [filterDate, setFilterDate] = useState("");
  const [filterCountry, setFilterCountry] = useState("");
  const [filterCity, setFilterCity] = useState("");

  const newSlug = slug.toUpperCase().replace("_", "/");

  const slugTranslate = {
    musikk: "music",
    sport: "sports",
    teater_show: "shows"
  };
  const translateSlug = slugTranslate[slug.toLowerCase()] || slug;

  // Hent events
  const getEvents = async () => {
    try {
      const response = await fetch(
        `https://app.ticketmaster.com/discovery/v2/events.json?apikey=QqvpEAdIbQPJB9GGqnSKAZvmpXwz79Y2&classificationName=${translateSlug}&locale=*&size=20`
      );
      const data = await response.json();
      setEvents(data._embedded?.events || []);
    } catch (error) {
      console.error("Feil under henting av events fra API", error);
    }
  };

  // Hent attractions
  const getAttractions = async () => {
    try {
      const response = await fetch(
        `https://app.ticketmaster.com/discovery/v2/attractions.json?apikey=QqvpEAdIbQPJB9GGqnSKAZvmpXwz79Y2&classificationName=${translateSlug}&locale=*&size=20`
      );
      const data = await response.json();
      setAttractions(data._embedded?.attractions || []);
    } catch (error) {
      console.error("Feil under henting av attractions fra API", error);
    }
  };

  useEffect(() => {
    getEvents();
    getAttractions();
  }, [slug]);

  // Filtrering av events
  const actualEvents = events.filter(
    (event) =>
      event.dates?.start &&
      event.classifications?.some((c) =>
        ["music", "sports", "arts & theatre", "miscellaneous"].includes(
          c.segment?.name?.toLowerCase()
        )
      )
  );

  // Filtrering av venues
  const venuesData = actualEvents
    .map((event) => event._embedded?.venues || [])
    .flat()
    .filter(
      (venue, index, self) =>
        venue && self.findIndex((v) => v.id === venue.id) === index
    );

  useEffect(() => {
    setVenues(venuesData);
  }, [events]);

  const filterEvents = events.filter((event) => {
    const nameMatch = event.name?.toLowerCase().includes(filterSearch.toLowerCase());
    const dateMatch = filterDate === "" || event.dates?.start?.localDate === filterDate;
    const countryMatch =
      filterCountry === "" ||
      event._embedded?.venues?.some((venue) =>
        venue.country?.name?.toLowerCase() === filterCountry.toLowerCase()
      );
    const cityMatch =
      filterCity === "" ||
      event._embedded?.venues?.some((venue) =>
        venue.city?.name?.toLowerCase() === filterCity.toLowerCase()
      );

    return nameMatch && dateMatch && countryMatch && cityMatch;
  });

  const filterAttractions = attractions.filter((attraction) =>
    attraction.name?.toLowerCase().includes(filterSearch.toLowerCase())
  );

  const filterVenues = venues.filter((venue) =>
    venue.name?.toLowerCase().includes(filterSearch.toLowerCase())
  );

```



```
  return (
    <>
      <h1>{newSlug}</h1>

      <div className="filter-container">
        <input
          type="text"
          placeholder="Søk etter navn..."
          value={filterSearch}
          onChange={(e) => setFilterSearch(e.target.value)}
        />
        <input
          type="date"
          value={filterDate}
          onChange={(e) => setFilterDate(e.target.value)}
        />
        <input
          type="text"
          placeholder="Filtrer på land"
          value={filterCountry}
          onChange={(e) => setFilterCountry(e.target.value)}
        />
        <input
          type="text"
          placeholder="Filtrer på by"
          value={filterCity}
          onChange={(e) => setFilterCity(e.target.value)}
        />
      </div>

      <h2>Arrangementer</h2>
      <section className="category-events">
        {filterEvents.length > 0 ? (
          filterEvents.map((event) => (
            <CategoryEvent event={event} key={event.id} />
          ))
        ) : (
          <p>Ingen arrangementer funnet.</p>
        )}
      </section>

      <h2>Attraksjoner</h2>
      <section className="category-attractions">
        {filterAttractions.length > 0 ? (
          filterAttractions.map((attraction) => (
            <CategoryAttraction event={attraction} key={attraction.id} />
          ))
        ) : (
          <p>Ingen attraksjoner funnet.</p>
        )}
      </section>

      <h2>Spillesteder</h2>
      <section className="category-venues">
        {filterVenues.length > 0 ? (
          filterVenues.map((venue) => (
            <CategoryVenue venue={venue} key={venue.id} />
          ))
        ) : (
          <p>Ingen spillesteder funnet.</p>
        )}
      </section>
    </>
  );
}
```

Denne koden førte oss på riktig vei, men vi var nødt til å endre deler da GPT hadde feil i koden som førte til at den ikke fungerte.
Så ved å bruke deler av GPT sin kode, kombinert med vår egen og våre rettinger av koden, så fikk vi da et funksjonelt filtreringssystem på siden.

Kilde for SanityEventPage.jsx:
Vi hadde større problemer med den siste biten av B-kravet, hvor vi skulle få opp hvilke andre brukere som har det arrangementet på enten ønskeliste eller tidligere kjøp på SanityEventPage.jsx.
Vi gikk da for å spør ChatGPT med hjelp, jeg brukte den følgende spørringen:
"Jeg trenger å liste opp alle brukere som har dette arrengementet på enten ønskeliste eller tidligere kjøp fra en sanity database"
Deretter ga jeg koden vi hadde så langt. Chatten ble relativt lang, så har lagt til en lenke med hele chatten.
Her er en lenke til samtalen med GPT
https://chatgpt.com/share/682a0a3c-1f84-800a-89ca-bc886cf7b078

Etter noen prompts med ChatGPT, så kom vi fram til en løsning med å kombinere GPT sin kode sammen med vår egen kode for å få en funksjonell funksjon.

Login-funksjon forklart - Eksamensprosjekt

1. Login-funksjon uten autentisering

Hva var poenget?

Jeg trengte ikke en ekte innlogging med backend eller autentisering. Jeg ville bare at brukeren skulle skrive inn et navn, trykke "Logg inn" , og da skulle appen vise en ny side (dashboardet).

Hvordan jeg gjorde det

Jeg brukte useState for å lagre både brukernavn og innloggingsstatus.
Når noen logger inn, settes isLoggedIn til true, og da vises DashboardView i stedet for LoginForm.

```
const [username, setUsername] = useState('');
const [isLoggedIn, setLocalIsLoggedIn] = useState(false);
```

Alt skjer bare i frontend. Det er ingen ekte validering, og passordet lagres ikke, det er kun for å simulere hvordan det ville fungert.

Komponentene

Jeg har delt opp funksjonen i tre komponenter:

- **LoginForm.jsx** skjemaet for brukernavn/passord
- **Dashboard.jsx** bestemmer hva som skal vises, og holder på login-status
- **DashboardView.jsx** det brukeren ser når de er "logget inn"

Hvordan det henger sammen:

1. Når brukeren logger inn:

```
// Kalles når bruker logger inn, lagrer brukernavn og oppdaterer login-state
const handleLogin = (name) => { //
  setUsername(name);
  setIsLoggedIn(true);
  setLocalIsLoggedIn(true);
};
```

2. Når brukeren logger ut:

```
// Funksjonen som logger ut brukeren og tilbakestiller tilstand
const handleLogout = () => {
  setIsLoggedIn(false);
  setLocalIsLoggedIn(false);
  setUsername('');
};
```

Alt vises via en ternary-operator i return:

```
// Funksjonen som logger ut brukeren og tilbakestiller tilstand
const handleLogout = () => {
  setIsLoggedIn(false);
  setLocalIsLoggedIn(false);

  setUsername('');
};
```

Forklaring av hele useEffect i DashboardView

Når DashboardView åpnes, så bruker jeg useEffect for å hente inn data. Det skjer bare en gang, fordi jeg har tom array [] i slutten. Først så henter jeg alle brukere fra Sanity, og etter det henter jeg detaljer for events som er i ønskelisten og tidligere kjøp hos hver bruker.

1. GROQ-spørring til Sanity

Jeg skriver en GROQ-spørring som henter navn, alder, kjønn, bilde, ønskeliste og tidligere kjøp. Inne i ønskelisten og kjøpene følger jeg referansen -> for å hente event og apiId, for det trenger jeg når jeg skal hente mer data fra Ticketmaster.
Etter det bruker jeg client.fetch(query) for å hente dataen. Når jeg får den, så bruker jeg .map() og async for å jobbe med hver bruker.

2. async funksjon og fetchEvents

Jeg lager en funksjon som heter fetchEvents(eventList), og den bruker jeg på både ønskeliste og tidligere kjøp. Jeg bruker async/await fordi fetch() tar litt tid og jeg må vente på svar. Derfor må det være async funksjon for at det skal funke.

3. Jeg bruker Promise.all

Inni fetchEvents() så bruker jeg Promise.all() fordi jeg vil hente mange events samtidig. Hvis det er f.eks. 5 eventer, så venter jeg ikke en og en, men tar alle samtidig. Det er mer effektivt og raskere.

4. Hvorfor jeg ikke bruker fetch fra Ticketmaster

I denne versjonen har jeg ikke lagt inn fetch()-kall mot Ticketmaster fordi det ga meg mye CORS-feilmeldinger da jeg testet det. Ticketmaster tillater ikke at man henter data direkte fra frontend uten spesielle tillatelser. Derfor valgte jeg å bare bruke dataen jeg allerede har fra Sanity, altså eventnavn og beskrivelse, og så sette images til en tom liste. Dette gjør at appen ikke krasjer, og det kommer heller ingen feilmeldinger i nettleseren.

5. Jeg setter brukeren inn i state

Når ønskeliste og tidligere kjøp er ferdig hentet for en bruker, returnerer jeg et nytt objekt med hele brukeren og de ferdige eventlistene. Alt dette sendes til setUsers, som setter det i state slik at jeg kan bruke det videre i komponenten.

```
useEffect(() => {
  // GROQ-spørring, som henter data fra sanity
  const query = `*[_type == "user"]{name, age, gender, profileImage {asset -> }, wishlist[]->{event, apiId, description},previousPurchases[]->{event ,apiId, description}}`;

  client.fetch(query).then(async (userData) => {
        // Mapper hver bruker og henter eventdata fra Ticketmaster for ønskeliste og kjøp
    setUsers( await Promise.all (userData.map(async (user) => {
        const fetchEvents = async (eventList) =>
        Promise.all((eventList || []).map(async (event) => {
            // Hvis hentingen feiler, så brukes data fra Sanity
            return {id: event.apiId, name: event.event, description: event.description , images: []};
        }));
        return {
            ...user,
            wishlist: await fetchEvents(user.wishlist),
            previousPurchases: await fetchEvents(user.previousPurchases),
        };
    })
  )
  );
});
}, []);
```

Problem med gamle koden:

Jeg hadde en funksjon i Dashboard som brukte fetch() til å hente events fra Ticketmaster API basert på apiId. Dette ble gjort inne i en try/catch-blokk for å håndtere feil.

Feil:

Når jeg testet det i nettleseren, fikk jeg CORS-feil. Ticketmaster tillater ikke at man henter data direkte fra frontend. Nettleseren blokkerer forespørselen fordi den mangler riktige CORS-headere. Resultatet var at jeg fikk mange feilmeldinger og eventene ikke ble lastet inn.

Løsning:

Jeg valgte å fjerne denne delen av koden midlertidig for å unngå feilmeldingene. I stedet bruker jeg nå dataen som allerede finnes i Sanity (f.eks. event-tittel og beskrivelse). Det gjør at Dashboard fortsatt fungerer uten at koden krasjer.

Hva som var vanskelig

- Å strukturere funksjonen riktig inne i useEffect
- Å hente mange API-kall samtidig og vente på at alt skulle bli ferdig
- Å unngå at appen krasjet hvis noe feilet
- Å rydde opp i variabler og navn så det ble forståelig

Hva jeg lærte av det

Jeg fikk god hjelp av ChatGPT til å rydde opp i funksjonen og gjøre det mer oversiktlig. Det viktigste jeg fikk på plass var:

- Å bruke Promise.all() riktig
- Å sørge for at bilder vises bare hvis de finnes (data.images || [])
- Å kombinere data fra to kilder og vise det på en ryddig måte

Tidligere erfaring som hjalp meg

Jeg har jobbet med Java og REST API-er før i backend, programmering 2 og i innføring i programmering . Derfor var det lett å forstå hvordan fetch() fungerer, og hvordan JSON-data hentes og vises i frontend.

Verktøy og funksjoner jeg brukte

- useState og useEffect
- GROQ-spørring (fra Sanity)
- Promise.all() for flere fetch-kall samtidig
- JSX og manuell CSS for visning

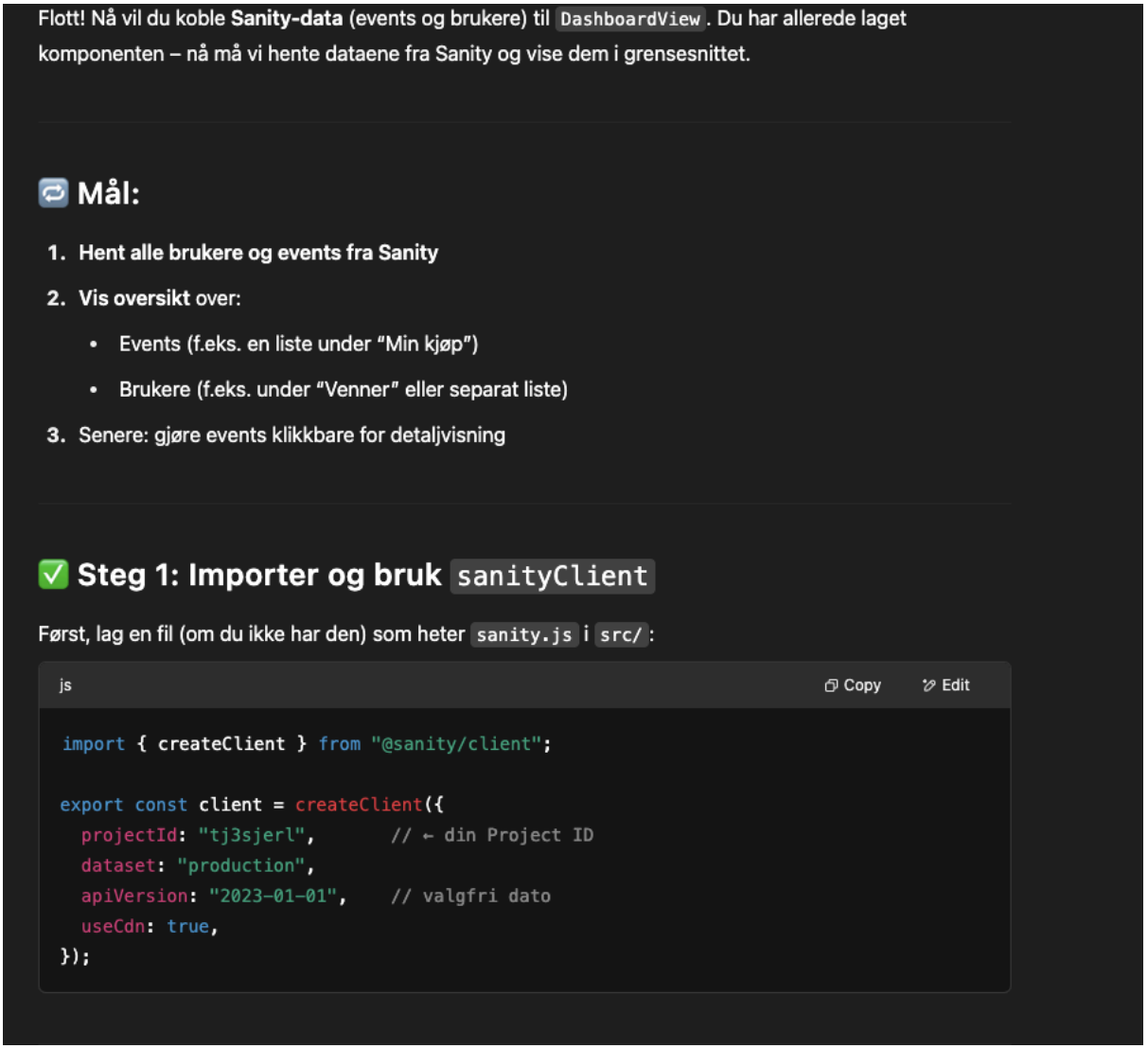
Kilder:

- https://www.sanity.io/docs/content-lake/query-cheat-sheet
- https://www.sanity.io/docs/content-lake/how-queries-work
- https://carl-topham.com/articles/guide-to-groq
- https://stackoverflow.com/questions/74561313/how-to-add-isloggedin-fuction-in-react-and-change-navbars-everytime-a-user-logs
- https://react.dev/reference/react/useState
- https://paulodeb.hashnode.dev/how-to-create-a-fake-react-authentication-with-usecontext-react-hook
- https://www.w3schools.com/react/react_forms.asp

Dashboard view brukt Ai for å kunne koble det til dashboard. jsx og rette feil og veiledning:

hvordan kobler jeg dette til dashboard import "../styles/dashboard.css";

```
export default function DashboardView({ username, onlogout }) {
  return (
    <section className="dashboard">
      <article>
        <h3 className="dashboard-title">Min side</h3>
      </article>


      <nav className="dashboard-nav" aria-label="Min navigasjon">
        <ul className="dashboard-nav-liste">
          <li className="dashboard-nav-item"><a href="#">Venner</a></li>
          <li className="dashboard-nav-item" ><a href="#">Min kjøp</a></li>
          <li className="dashboard-nav-item"><a href="#">Min ønskeliste</a></li>
        </ul>
        </nav>
      <section className="logout-section">
        <button onClick={onlogout} className= "logout-button">Logg ut</button>
      </section>
      </section>

  );
}
```



Sendte en kode for å sjekke om logikken og syntaks stemte:

min kode :

```
import "../styles/dashboard.css";
import { useState, useEffect } from "react";
import { client } from "../client";
import user from "../sanityDashboard/schemas/user";

export default function DashboardView({ username, onlogout }) {
  const [users, setUsers] = useState([]);
  const [events, setEvents] = useState([]);

  useEffect(() => { // Ikke slett dette dette er spørring språk for
sanity til å hente data!!!!!!
    client.fetch('*[_type == "user"]{..., wishlist[]->,
previousPurchases[]->}')
      .then(data => setUsers(data))
      .catch(err => console.error(err))

    client.fetch('*[_type == "event"]')
      .then(data => setEvents(data))
      .catch(err => console.error(err))
  }, [])


  return (
    <section className="dashboard">
      <article>
        <h3 className="dashboard-title">Min side</h3>
      </article>


      <nav className="dashboard-nav" aria-label="Min
navigasjon">
        <ul className="dashboard-nav-liste">
          <li className="dashboard-nav-item"><a
```

Chat sin respons:

Bra at du deler koden – her er det noen **feil som må fikses**, slik at React forstår og viser innholdet riktig.
Jeg hjelper deg fikse det strukturert og semantisk uten `main`, `header` eller `div`.

❗ Feil i koden:

1. `users.map(...)` **er ikke inne i** `{ }` → du må bruke `{users.map(...)}` i JSX
2. Du skriver `user_id`, men det skal være `user._id`
3. `user.birthate` har skrivefeil → skal være `user.birthDate`
4. Feil i `wishlist.map` – mangler `{}` og har `</li>` uten innhold

🔍 1. Feil: `users.map(...)` var ikke inne i `{}`

Du skrev:

jsx Copy Edit

```jsx
<section className="dashboard-innhold">
  users.map((user) => (
    ...
  ))
</section>
```

Feil:

- React forstår ikke `users.map(...)` som tekst – du må bruke **krøllparanteser** for all JavaScript i JSX.

Riktig:

jsx Copy Edit

```jsx
<section className="dashboard-innhold">
  {users.map((user) => (
    ...
  ))}
</section>
```

🔍 2. Feil: `key={user_id}`

Du skrev:

jsx Copy Edit

```jsx
<section key={user_id}>
```

Feil:

- `user_id` er ikke definert. Det skal være `user._id` – dette får du fra Sanity automatisk.

Riktig:

jsx Copy Edit

- `user_id` er ikke definert. Det skal være `user._id` – dette får du fra Sanity automatisk.

Riktig:

jsx Copy Edit

```jsx
<section key={user._id}>
```

🔍 3. Feil: `birthate` har skrivefeil

Du skrev:

jsx Copy Edit

```jsx
<p>{user.birthate}</p>
```

Feil:

- Du har en typo: `birthate` → burde være `birthDate` (som i schemaet).

Riktig:

jsx Copy Edit

```jsx
<p>{user.birthDate}</p>
```

🔍 4. Feil: `wishlist.map(...)` var feil formattert

Du skrev:

jsx Copy Edit

```jsx
<ul>
  user.wishlist?.map((event) => (
    <li key={event._id}></li>
    event.title}<br />
    <a href={`/sanity-event/${event.apiId}`}>Se mer om dette kjøpet</a>
</ul>
```

Feil:

- `map(...)` var ikke inne i `{ }`
- `<li>` var tom og lukket for tidlig

Feil:

- `map(...)` var ikke inne i `{ }`
- `<li>` var tom og lukket for tidlig
- `event.title` og `<a>` var **utenfor** `li`

Riktig:

jsx Copy Edit

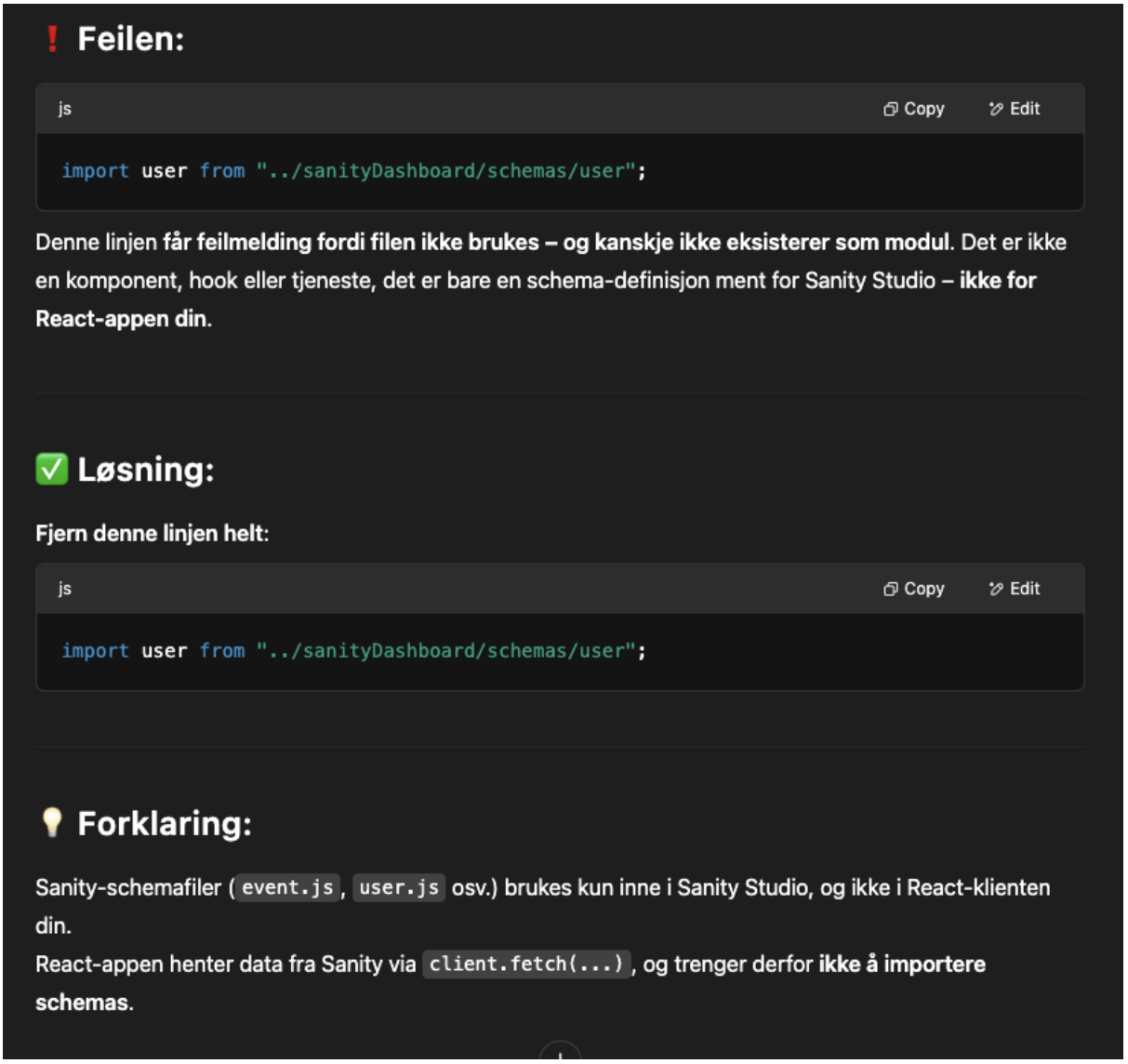
```jsx
<ul>
  {user.wishlist?.map((event) => (
    <li key={event._id}>
      {event.title}<br />
      <a href={`/sanity-event/${event.apiId}`}>Se mer om dette kjøpet</a>
    </li>
  ))}
</ul>
```

Fikk feil melding i console :

Sendte dette til Chat:

```
[plugin:vite:import-analysis] Failed to resolve import "../sanityDashboard/schemas/user" from
"src/components/DashboardView.jsx". Does the file exist?

/Users/ehsamjosefzai/Documents/GitHub/Eksamen_UIN25/src/components/DashboardView.jsx:4:17
19 |  import { useState, useEffect } from "react";
20 |  import { client } from "../client";
21 |  import user from "../sanityDashboard/schemas/user";
   |                   ^
22 |  export default function DashboardView({ username, onlogout }) {
23 |    _s();

    at TransformPluginContext._formatLog (file:///Users/ehsamjosefzai/Documents/GitHub/Eksamen_UIN25/node_modules/vite/dist/node/chun
    at TransformPluginContext.error (file:///Users/ehsamjosefzai/Documents/GitHub/Eksamen_UIN25/node_modules/vite/dist/node/chunks/de
    at normalizeUrl (file:///Users/ehsamjosefzai/Documents/GitHub/Eksamen_UIN25/node_modules/vite/dist/node/chunks/dep-Bn81Esdm.js:4(
    at process.processTicksAndRejections (node:internal/process/task_queues:105:5)
    at async file:///Users/ehsamjosefzai/Documents/GitHub/Eksamen_UIN25/node_modules/vite/dist/node/chunks/dep-Bn81Esdm.js:40594:37
    at async Promise.all (index 6)
    at async TransformPluginContext.transform (file:///Users/ehsamjosefzai/Documents/GitHub/Eksamen_UIN25/node_modules/vite/dist/node
    at async EnvironmentPluginContainer.transform (file:///Users/ehsamjosefzai/Documents/GitHub/Eksamen_UIN25/node_modules/vite/dist/
    at async loadAndTransform (file:///Users/ehsamjosefzai/Documents/GitHub/Eksamen_UIN25/node_modules/vite/dist/node/chunks/dep-Bn8
    at async viteTransformMiddleware (file:///Users/ehsamjosefzai/Documents/GitHub/Eksamen_UIN25/node_modules/vite/dist/node/chunks/d

Click outside, press Esc key, or fix the code to dismiss.
You can also disable this overlay by setting server.hmr.overlay to false in vite.config.js.
```

Chat sin respons:



❗ Feilen:

```
js                                                    Copy    Edit
import user from "../sanityDashboard/schemas/user";
```

Denne linjen **får feilmelding fordi filen ikke brukes – og kanskje ikke eksisterer som modul**. Det er ikke en komponent, hook eller tjeneste, det er bare en schema-definisjon ment for Sanity Studio – **ikke for React-appen din**.

✅ Løsning:

Fjern denne linjen helt:

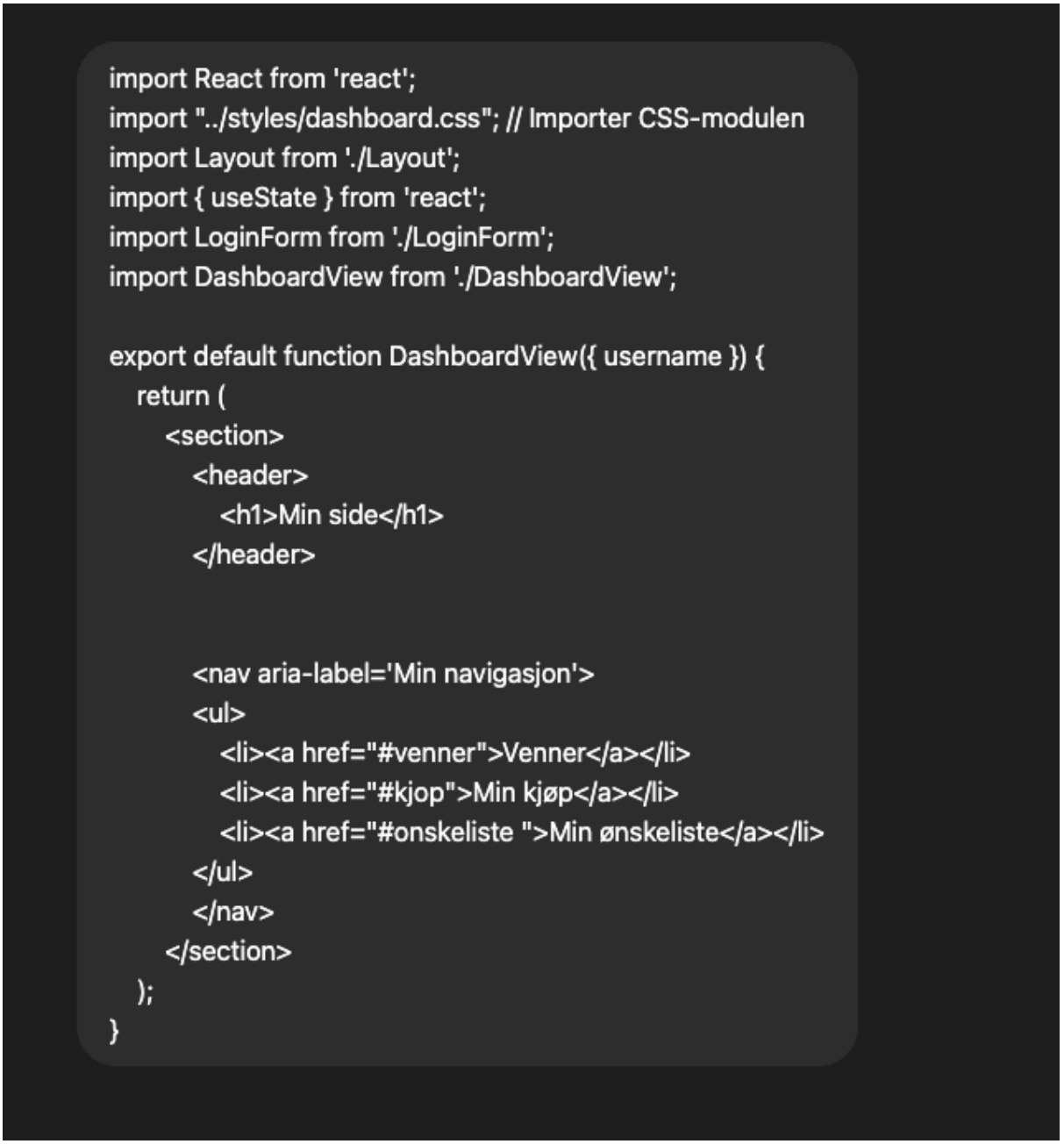
```
js                                                    Copy    Edit
import user from "../sanityDashboard/schemas/user";
```

💡 Forklaring:

Sanity-schemafiler (`event.js`, `user.js` osv.) brukes kun inne i Sanity Studio, og ikke i React-klienten din.
React-appen henter data fra Sanity via `client.fetch(...)`, og trenger derfor **ikke å importere schemas**.

Dashboard view:

Min spørring:



```
import React from 'react';
import "../styles/dashboard.css"; // Importer CSS-modulen
import Layout from './Layout';
import { useState } from 'react';
import LoginForm from './LoginForm';
import DashboardView from './DashboardView';

export default function DashboardView({ username }) {
  return (
    <section>
      <header>
        <h1>Min side</h1>
      </header>


      <nav aria-label='Min navigasjon'>
      <ul>
        <li><a href="#venner">Venner</a></li>
        <li><a href="#kjop">Min kjøp</a></li>
        <li><a href="#onskeliste ">Min ønskeliste</a></li>
      </ul>
      </nav>
    </section>
  );
}
```

Chat respons:

❌ Problemet:

Du har definert `DashboardView` to ganger:

- Først importerer du den:

```
js                                                    Copy    Edit
import DashboardView from './DashboardView';
```

- Og så definerer du **en ny komponent med samme navn** rett under:

```
js                                                    Copy    Edit
export default function DashboardView({ username }) { ... }
```

Dette fører til at komponenten prøver å **importere seg selv**, og kan føre til bugs, feil oppførsel eller infinite loops.

✅ Løsning:

Du må **skille** mellom komponentfilene.

Enten:

- Dette er din **egentlige** `DashboardView.jsx`
  → Da skal du **fjerne importen** fra seg selv.

```
js                                                    Copy    Edit
// ❌ FJERN DENNE:
import DashboardView from './DashboardView';
```

Eller:

- Dette er en ny komponent (f.eks. `DashboardMenu` eller `DashboardContent`)
  → Da bør du **endre navnet på funksjonen** og **endre eksportnavnet**

Logg ut knappen funksjon:

når man trykker på logg ut knapper skal man bli logget ut

Ja – og det kan vi enkelt få til!

Når du trykker på **"Logg ut"-knappen**, skal disse to tingene skje:

1. Brukeren skal **logges ut** (state: `isLoggedIn` → `false`)
2. Dashboard-siden skal **vise innloggingsskjema igjen**

✅ Løsning: Oppdater `Dashboard.jsx`

Du har sikkert noe slikt nå:

jsx Copy Edit

```jsx
export default function Dashboard({ setIsLoggedIn }) {
  const [username, setUsername] = useState('');
  const [isLoggedInLocal, setIsLoggedInLocal] = useState(false);
```

🔧 Legg til en `handleLogout`-funksjon:

jsx Copy Edit

```jsx
const handleLogout = () => {
  setIsLoggedIn(false);       // globalt (Layout)
  setIsLoggedInLocal(false);  // lokalt (Dashboard)
  setUsername('');            // tømmer navnet
};
```

🔄 Bruk den i `DashboardView`:

jsx Copy Edit

```jsx
<DashboardView username={username} onLogout={handleLogout} />
```

✅ Du må:

1. Lage en `handleLogout`-funksjon
2. Sende den som prop til `DashboardView`
3. Bruke `onLogout` i knappen i `DashboardView`

Avsluttende refleksjon over Dashboard sidene

Oppsummering for dashboard og loginForm siden.:

Jeg brukte ChatGPT litt underveis, mest for å finne småfeil i syntaks og få ideer til hvordan jeg kunne strukturere FetchEvents bedre og koble sammen dataene. Jeg skrev det meste selv, men det hjalp å få bekrefta at jeg var på rett vei.

Autocomplete i VS Code var også supernyttig – spesielt for å få oversikt over props og felter mens jeg bygde komponentene.