# Beskrivning Meeting Calendar

## App

### Import:

import React, {useState} from "react";

import Navbar from "./components/Navbar";

import Sidebar from "./components/Sidebar";

import "./App.css";

import MeetingForm from "./components/MeetingForm";

import MeetingList from "./components/MeetingList";

import Footer from "./components/Footer";

### Komponentstruktur:

const App = () => {

// useStates- variabler  
 // funktioner

  return (

    // Renderingsdelen

  );

};

export default App;

### State- variabler:

  const [meetings, setMeetings] = useState([]);

  const [editingMeeting, setEditingMeeting] = useState(null);

Förklaring:

**meetings** är en array variabel som till en början är tom och som innehåller alla möten.

**setMeetings** är en funktion som kommer att ta emot en array och uppdatera state- variabeln **meetings**.

**editingMeeting** är en variabel vars värde från början är null och som kommer att hålla koll på pågående uppdateringar.

**setEditingMeeting** är en funktion som uppdaterar editingMeeting när en uppdatering påbörjas eller avslutas.

### Funktioner:

#### deletMeeting

const deleteMeeting = (id) => {

    setMeetings(prevMeetings => prevMeetings.filter(meeting => meeting.id !==id));

##### Syfte

Dess uppgift är att ta emot ett id från **MeetingList** och genom setMeetings returnera en lista där posten med id:s som skickas som parameter inte finns.

###### Tryggas av:

När användaren klickar på knappen med papperskorgen vid varje mötespost.

###### Hur den fungerar:

**setMeetings** anropas med en funktion som parameter. Resultatet av funktionen är en lista som inte kommer att innehålla posten där papperskorgsknappen klickades.

**setMeetings** är en så kallad updaterfunktion, vars syfte är att uppdatera state- variabeln **meetings**.

**prevMeetings** däremot fungerar som en snapshot av **meetings**. Den arraylistan kommer sedan att filtreras och alla poster som inte har det id som skickades kommer att vara med i den uppdaterade listan **meetings**.

#### addMeeting

const addMeeting = (meeting) =>{

    setMeetings(prevMeetings => [

      ...prevMeetings,

      { ...meeting, id: prevMeetings.length + 1 }

    ]);

  };

##### Syfte

Uppdaterar meetings med en ny post som är det som använder fyller i formuläret.

###### Tryggas av:

När användaren klickar på knappen **Create Meeting** i formuläret.

###### Hur den fungerar:

Här arbetar funktionen med två kopior. Man skapar först en tom array och befolkar den med en kopia av alla möten som redan existerar med **…prevMeetings**. Sedan skapar man en kopia av parametern meeting som är data från formuläret med **…meeting** och kompletterar posten med ett unik id som är arrayens längd plus 1.

#### UpdateMeeting

const updateMeeting = (updateMeeting)=>{

    setMeetings(prevMeetings =>

      prevMeetings.map(meetting =>

        meetting.id === updateMeeting.id ? updateMeeting: meetting

      )

    );

  };

##### Syfte

Uppdaterar meetings genom att skicka informationen från den uppdaterade posten som finns i formuläret.

###### Tryggas av:

Användaren klickar först på **Edit-** knappen vid listan bland alla poster vilket gör att postens data flyttas till formuläret. Då ändras knappen längst ner från **Create Meeting** till **Update Meeting** och det är den knappen som triggar funktionen.

###### Hur den fungerar:

Funktionen tar emot hela posten som uppdateras. Sedan går funktionen igenom en kopia av **meetings** med funktionen **map** där logiken säger att om en post med samma id som den som finns i den skickade posten och den aktuella posten som loopas igenom från den aktuella listan hittas, ersätts den posten med den skickade parametern, annars behålls posten som den är.

### Renderings delen

return (

    <>

      <div className="container">

        <Navbar />

        <div className="row mt-3">

          <Sidebar />

          <div className="col-md-9">

            <MeetingForm

              onAddMeeting={addMeeting}

              onUpdateMeeting={updateMeeting}

              editingMeeting={editingMeeting}

              setEditingMeeting={setEditingMeeting}

            />

            <MeetingList

              meetings={meetings}

              onDeleteMeeting={deleteMeeting}

              onEditMeeting={setEditingMeeting}

            />

          </div>

        </div>

      </div>

      <Footer />

    </>

  );

**1. Fragment (<> ... </>)**

* De yttersta taggarna <> ... </> är ett så kallat **React Fragment**. Det används för att gruppera flera element utan att lägga till ett extra DOM-element i HTML-strukturen.

**2. <div className="container">**

* En container-div används ofta för att centrera och ge marginaler till innehållet, ofta med hjälp av CSS-ramverk som Bootstrap.

**3. <Navbar />**

* Komponent som visar en navigationsmeny överst på sidan.

**4. <div className="row mt-3">**

* En rad (row) med Bootstrap-klassen mt-3 (margin-top 3) för att ge lite avstånd uppåt.

**5. <Sidebar />**

* Komponent som visar en sidopanel, ofta för navigation eller extra information.

**6. <div className="col-md-9">**

* En kolumn som tar upp 9 av 12 kolumnplatser på medelstora skärmar och uppåt (Bootstrap). Här placeras huvudinnehållet.

**7. <MeetingForm ... />**

* Ett formulär för att skapa eller redigera möten. Får flera props:
  + onAddMeeting: Funktion för att lägga till möte.
  + onUpdateMeeting: Funktion för att uppdatera möte.
  + editingMeeting: Håller koll på om ett möte redigeras.
  + setEditingMeeting: Funktion för att ändra vilket möte som redigeras.

**8. <MeetingList ... />**

* En lista som visar alla möten. Får props:
  + meetings: Array med alla möten.
  + onDeleteMeeting: Funktion för att ta bort möte.
  + onEditMeeting: Funktion för att börja redigera ett möte.

**9. <Footer />**

* Komponent som visar sidfoten längst ner på sidan.

## Meetingform

### Översikt

MeetingForm är en React-komponent som används för att skapa eller redigera möten. Den hanterar formulärfält, validering och skickar data vidare till föräldrakomponenten via props-funktioner. Komponenten använder flera state-variabler för att hålla reda på formulärdata och eventuella felmeddelanden

### Props

Komponenten tar emot följande props från App:

* **onAddMeeting:** Funktion som anropas när ett nytt möte ska läggas till.
* **onUpdateMeeting:** Funktion som anropas när ett befintligt möte ska uppdateras.
* **editingMeeting:** Om satt, betyder det att ett möte håller på att redigeras (annars skapas ett nytt).
* **setEditingMeeting:** Funktion för att sätta eller nollställa vilket möte som redigeras.

### Import

import React, { useEffect, useState } from "react";

### Komponentstruktur

const MeetingForm = ({onAddMeeting, onUpdateMeeting, editingMeeting, setEditingMeeting}

) => {

// useStates- variabler  
 // funktioner

  return (

    //Renderingsdelen

  );

};

export default App;

### State-variabler

const [title, setTitle] = useState("");

  const [titleError, setTitleError] = useState("");

  const [date, setDate] = useState("");

  const [time, setTime] = useState("");

  const [dateTimeError, setDateTimeError] = useState("");

  const [level, setLevel] = useState("Beginner");

#### Förklaring

* **title, setTitle:** Håller mötets titel.
* **titleError, setTitleError:** Håller felmeddelande för titel.
* **date, setDate:** Håller mötesdatum.
* **time, setTime:** Håller mötestid.
* **dateTimeError, setDateTimeError:** Felmeddelande för datum/tid.
* **level, setLevel:** Håller mötesnivå (t.ex. Beginner, Intermediate).

### Funktioner

#### useEffect

  useEffect(() => {

    if (editingMeeting) {

      setTitle(editingMeeting.title);

      setDate(editingMeeting.date);

      setTime(editingMeeting.time);

      setLevel(editingMeeting.level);

    }

  }, [editingMeeting]);

##### Syfte och trigger

Den här funktionen används exklusivt när användaren klickar på Edit- knappen.

Villkoret editingMeeting uppfylls och informationen flyttas över från posten man klickade från till formuläret för vidare redigering.

##### Hur den fungerar

Värdet skickas som en prop från App till MeetingForm.

Funktionen anropas när användaren klickar på Edit- knappen.

Den kontrollerar först att editingMeeting är ett objekt och om så är fallet uppdateras fälten tack vare kopplingen mellan state och formulärfältet.

### Renderings delen