

Introduksjon

Applikasjonen er en enkel stemmegivningsapp utviklet med React som tillater brukeren å legge til kandidater man kan stemme på, samt følge med på stemmehistorikken.

Funksjonaliteter

- **Totale stemmer:** det totale antallet stemmer gitt på tvers av kandidater vises under overskriften.
- **Legge til kandidater:** brukere kan legge til nye kandidater ved å skrive inn kandidatens navn i et tekstfelt med teksten "Enter candidate name.." og trykke på knappen "Add Candidate". Hver kandidat får sitt eget kort.
- **Søke etter kandidater:** brukere kan søke etter kandidater ved å skrive inn kandidatnavnet i tekstfeltet "Search candidates". Søket utføres fortløpende.
- **Stemmegivning:** brukere kan gi stemmer til kandidater ved å trykke på knappen "+1 Vote" eller trekke fra stemmer ved å trykke på knappen "-1 Vote". En kandidat kan ikke ha mindre enn 0 stemmer.
- **Slette kandidater:** brukeren kan slette en kandidat ved å trykke på knappen "Delete".
- **Endre kandidatnavn:** brukeren kan endre på en kandidats navn ved å trykke på knappen "Edit Name", og deretter endre på navnet som dukker opp i tekstfeltet. For å lagre det nye navnet kan brukeren trykke på "Save", og for å kansellere kan brukeren trykke på "Cancel".
- **Stemmehistorikk:** en oversikt over alle stemmer som blir gitt, med kandidatens navn og om stemmen er +1 eller -1. Hvis kandidatens navn blir endret, endres også navnet i stemmehistorikken. Hvis en bruker gir -1 stemme til en kandidat med 0 stemmer, vises ingen ting i stemmehistorikken da kandidater ikke kan ha mindre enn 0 stemmer.

Mappestruktur

```
vote-app/  
├── src/  
│   ├── App.jsx  
│   ├── App.css  
│   ├── components/  
│   │   ├── AnimatedBackground.jsx  
│   │   ├── CandidateCard.jsx  
│   │   ├── CandidateControls.jsx  
│   │   ├── CandidateList.jsx  
│   │   ├── Header.jsx  
│   │   ├── TotalVotes.jsx  
│   │   ├── VoteButtons.jsx  
│   │   └── VoteHistory.jsx
```

- **App.jsx:** hovedkomponenten i applikasjonen som håndterer den overordnede tilstanden og logikken. `useState` sørger for å administrere kandidater, stemmehistorikk og søkefunksjonalitet på en dynamisk måte. Filen renderer komponentene i appen, samt inneholder også funksjoner for å legge til, slette, oppdatere kandidater, samt håndterer stemmegiving og søkefunksjonen.

- **App.css:** inneholder CSS-stilarkene som definerer stilene og layouten til appen.
- **AnimatedBackground:** gir en animert bakgrunn ved å bruke biblioteket `@firecms/neat` ved å render et `<canvas>` element.
- **Header.jsx:** inneholder en med en `<header>` `<H1>` overskrift.
- **CandidateControls.jsx:** Gir brukeren mulighet til å legge til nye kandidater og søke blant eksisterende kandidater. Den inneholder et tekstfelt for å skrive inn kandidatens navn og en knapp for å legge til kandidaten, samt et tekstfelt for å søke.
- **CandidateList.jsx:** renderer en liste over alle kandidater ved å mappe over `candidates` arrayet og opprette et `CandidateCard` for hver kandidat. Sørger for at riktig informasjon og funksjoner blir sendt som props til hvert `CandidateCard`.
- **CandidateCard:** viser detaljene for hver enkelt kandidat. Inneholder kandidatens navn, antall stemmer, og gir funksjonalitet for å stemme, redigere kandidatens navn og slette kandidaten.
- **VoteButtons:** inneholder knappene for å gi positive eller negative stemmer til en kandidat. Når en knapp trykkes på blir `onVote` funksjonen kalt på med enten +1 eller -1 som parameter.
- **TotalVotes:** teller som viser det totale antallet stemmer som er gitt på tvers av kandidatene. Beregner summen av stemmer ved å bruke `reduce` funksjonen i `candidates` arrayet.
- **VoteHistory.jsx:** Viser en liste over alle gyldige stemmegivninger som har blitt utført. For hver stemme registreres kandidatens navn og endringen i antall stemmer. Historikken oppdateres ikke når en kandidat med 0 stemmer får en negativ stemme.

Et eksempel på hvordan `useState` er brukt i applikasjonen

I `CandidateControls.jsx` brukes `useState` for å håndtere inputfeltet for å legge til nye kandidater. `const [newName, setNewName] = useState('')` initialiserer en tom streng som startverdi, `newName` inneholder den nåværende verdien og `setNewName` er funksjonen som brukes for å oppdatere verdien. Inputfeltet hvor navnet på den nye kandidaten skrives inn er kontrollert av `value={newName}`. Hver gang brukeren skriver oppdateres staten via `onChange={(e) => setNewName(e.target.value)}`. Når den nye kandidaten legges til, nullstilles input-feltet med `setNewName('')`.