Gift Picker App

About

The Gift Picker App is a tool for collecting gift ideas in a very simple way. It helps to pick out the perfect gift for any occasion. Dad's birthday, grandma's and grandpa's anniversary, or even your cousin's bat mitzva; keep all these events in one central place and organize different wish lists for different people. This offline-first web app allows you to write down your ideas anywhere and anytime - even without an internet connection.

Try Gift Picker App now for free!

Technologies

This project was created using Create React App and Create React App PWA Template.

All pages are loaded using React Components, routing is handled by React Router DOM, and most operations are performed using React Hooks. Offline capabilities are made possible by the ServiceWorker implemented in the CRA PWA Template.

The Materialize CSS framework is used for styling, along with Material Icons for the icons.

Google Firebase is an essential part of this app, as it manages authentication through Firebase Authentication, database through Cloud Firestore, and hosting through Firebase Hosting.

File Structure

The app was created using Create React App and the CRA PWA Template (see Technologies for reference). In this section I want to explain the changes I've brought to this file structure and the content of these files, but also the files I've added to this project.

```
gift-picker-app-react
├── README.md
├── firebase.json
├── package.json
├── package-lock.json
├── build-deploy-script.txt
├── .eslintcache
├── .firebaserc
├── .gitignore
│
├── node_modules
│   └── ...
│
├── public
│   ├── 404.html
│   ├── index.html
│   ├── favicon.ico
│   ├── manifest.json
│   ├── logo192.png
│    ├── logo512.png
│    └── robots.txt
│
└── src
    ├── logo.svg
    ├── App.css
    ├── index.css
    ├── App.js
    ├── App.test.js
    ├── data.js
    ├── functions.js
    ├── index.js
    ├── reportWebVitals.js
    ├── service-worker.js
    ├── serviceWorkerRegistration.js
    ├── setupTests.js
    │
    └── components
        ├── Home.js
        ├── DateInfoCard.js
        ├── CreateEvent.js
        ├── ReadEvent.js
        ├── UpdateEvent.js
        └── Firebase.js
```

`./build-deploy-script.txt`

Contains commands employed when building and (re)deploying a new version of the app.

`./public/`

`public/404.html`

Generated by Firebase CLI when configuring hosting for the app. Completely unnused by that the app.

`public/index.html`

This is the central point of the app; the page that's loaded when someone opens the app. However, none of this file's contents are ever shown, as the actual app content gets loaded through scripts. This mainly contains `<meta/>` and `<link/>` tags that are necessary for the app to work.

`public/favicon.ico`, `public/logo192.png`, and `public/logo512.png`

All of these files contain the logo of the app, to be used in different places (for example: icon on bookmarks, or app icon when app is downloaded onto mobile device). These are used within `public/index.html` and linked to in `public/manifest.json`

`public/manifest.json`

Contains metadata such as name of the app, links to icons and logo images to use.

`./src/`

`src/App.js`

Contains code that is used throughout all pages of the app (navigation & routes). The `<nav/>` contains links to `/` (Home), `/createEvent/` (CreateEvent), and `/readProfile/[id]/` (ReadProfile) where `[id]` is the id of the currently logged in user.

`src/data.js`

Contains all kinds of constant data objects that are used in different components of the App (event types, placeholders, ...).

`src/functions.js`

Contains all kinds of functions that are used in different components of the App (data conversion, form checking, ...).

`src/index.js`

This renders the `<BrowserRouter/>` and `<App/>` components, and registers the `ServiceWorker`.

`src/components/`

`components/Home.js`

Currently does not contain a calendar, but will in the future. On this calendar a date could be selected and events would be displayed in the `<DateInfoCard/>` component. Home would also include an alternative view in which all events would be displayed in a scrollable list of rows.

`components/DateInfoCard.js`

Displays the name of events for the selected date. These names are clickable and link to `/readEvent/[id]/` where `[id]` is the event id.

`components/CreateEvent.js`

Renders a form where data can be entered and submitted in order to add an event to the database. This component is coded as a React class component, and not a function component like the other components used in this app.

`components/ReadEvent.js`

Renders rows of data with name and value taken from an event. The event displayed is the one who's id is found in the current URL (`/readEvent/[idOfTheEventWeWantToRead]/`). It also contains a link to a `/updateEvent/[id]/` where `[id]` is the id of the event we are currently reading, a "delete event" button, and a button that links to the previous page.

`components/UpdateEvent.js`

Renders the same form as the `<CreateEvent/>` component. In this case however, the form already has it's input fields filled with data taken from the event that we want to update, and the submit button will push the new data into the selected event. The data in the form is of the event who's id is found in the current URL (`/updateEvent/[idOfTheEventIWantToUpdate]/`). It also contains a button that links to the previous page.

`components/Firebase.js`

Contains the Firebase configuration and options for accessing the Firestore. The `firebase` object is then exported for use in other app components.