Calculator Design Document

Author:

Razi Syed

Overview:

This design document details the the design of a basic calculator web application. The proposed solution consists of a frontend UI portion in React as well a third-party authentication API, firebase.

Frontend:

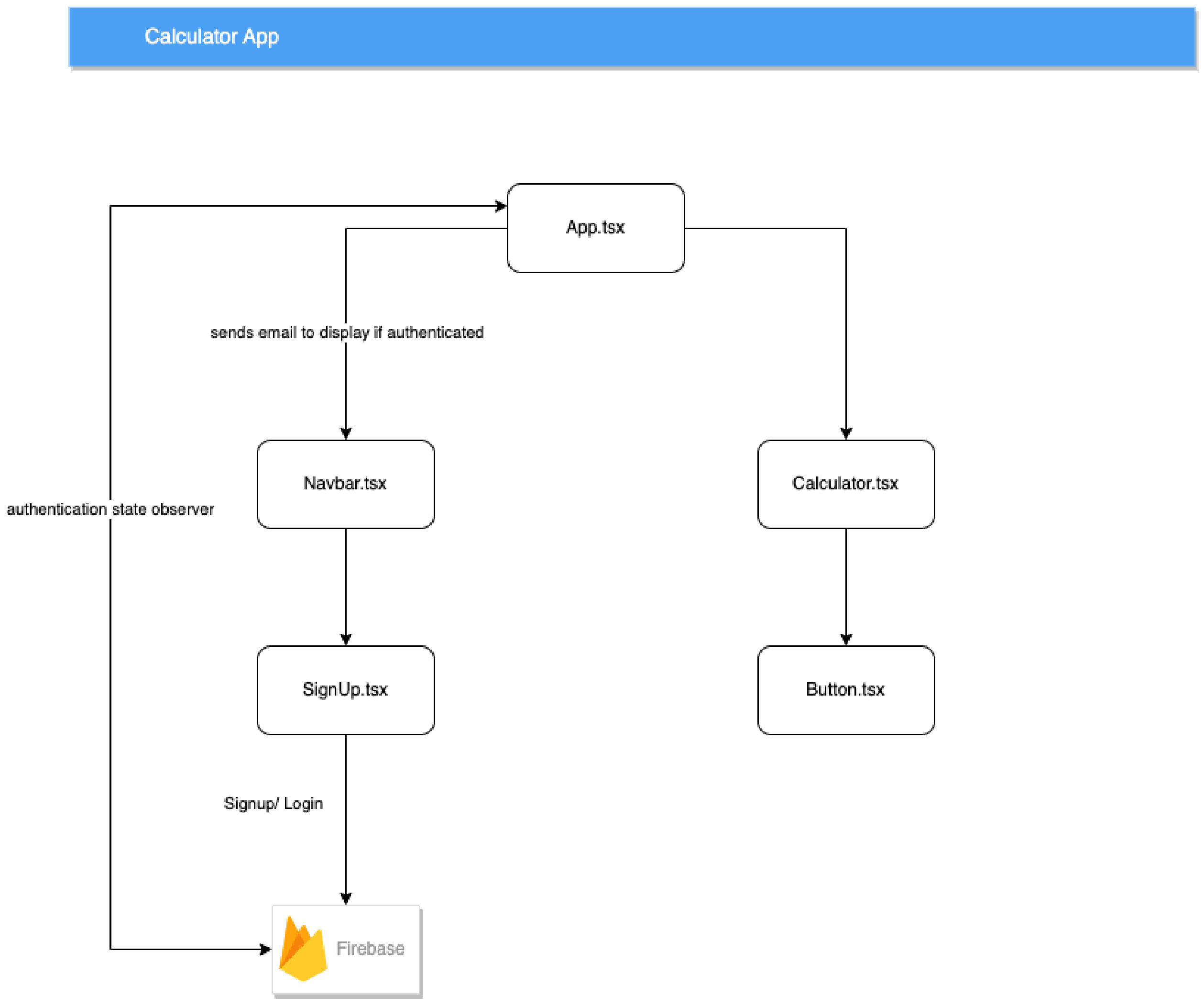
- The application components are written in TypeScript as it has type checking which proves quite useful for a calculator
- The framework used is React for ease of development
- Build tool is Vite for same reason as above
- Styling is done using TailwindCSS as a personal preference and for flexibility

Backend

- The backend service used to handle navigation is firebase
- Used it because it’s very easy to set up and this is quite a small project where I’m not storing additional user details

Design

- I kept the design pretty simple and only created new components when it made sense to do so
- I did not use Context or a Reducer as it was not necessary for the scope of the application; using state and passing props down was sufficient



Components

App.tsx

- This component arranges the core components of the app and has an observer from the firebase library to get the authentication status of the user whenever they sign in/out or sign up
- The components contained are
    - Navbar
    - Calculator

Navbar.tsx

- This is where the user authentication status is displayed
- Has a Signup/ Logout button which changes based on the user authentication status
- The components contained are:
    - SignUp

SignUp.tsx

- This component opens the modal where a user can either sign up or login to their account
- It uses state to manage what type of button to show and what kind of modal the user sees
- It also uses `createUserWithEmailAndPassword` and `signInWithEmailAndPassword` from firebase to handle signup and login

Calculator.jsx

- This is the core component of the application which manages the functionalities of all of the different buttons on the calculator as well as the math calculations
- The main function for calculation here is the `evaluateExpression` function which goes through a series of “replacements” using regex to produce a valid string which can then be passed to the JavaScript `eval` function
- Other functions are for handling different types of calculations, like memory functions or the history function
- It contains buttonData which is an array of objects representing all of the button data needed to create functional data; this helps keep the return statement clean
- It contains the Button component

Button.tsx

- This component is simply a wrapper for the button built-in jsx button tag but uses default props and passed in props to be flexible and handle multiple types of buttons and their functions

- This improves readability