Design Document

Pawandeep Suryavanshi
IIT Jodhpur

suryavanshi.1@iitj.ac.in
Mobile : +91 9754929765

Web Application Idea

The idea is to create a web application where users can sign in and start evaluation sessions which will evaluate their emotional state based on their facial expressions. Users just need to sign in, start the session and keep the tab open in their browser and then they can continue their other work, and in the background this web app will record their facial expressions. Users can view their emotional states in past sessions in the analytics section of the web app. This recorded data can also be trained to generate mental health scores and similar data which can be used by doctors and psychiatrists to help their patients.

Tech Stack

- Frontend - React Framework is used.
- Backend - Express (Node.js) is used.
- Database - Mongodb
- Classification of image into 7 facial expressions - Deepface Library is used. Github

Application Architecture

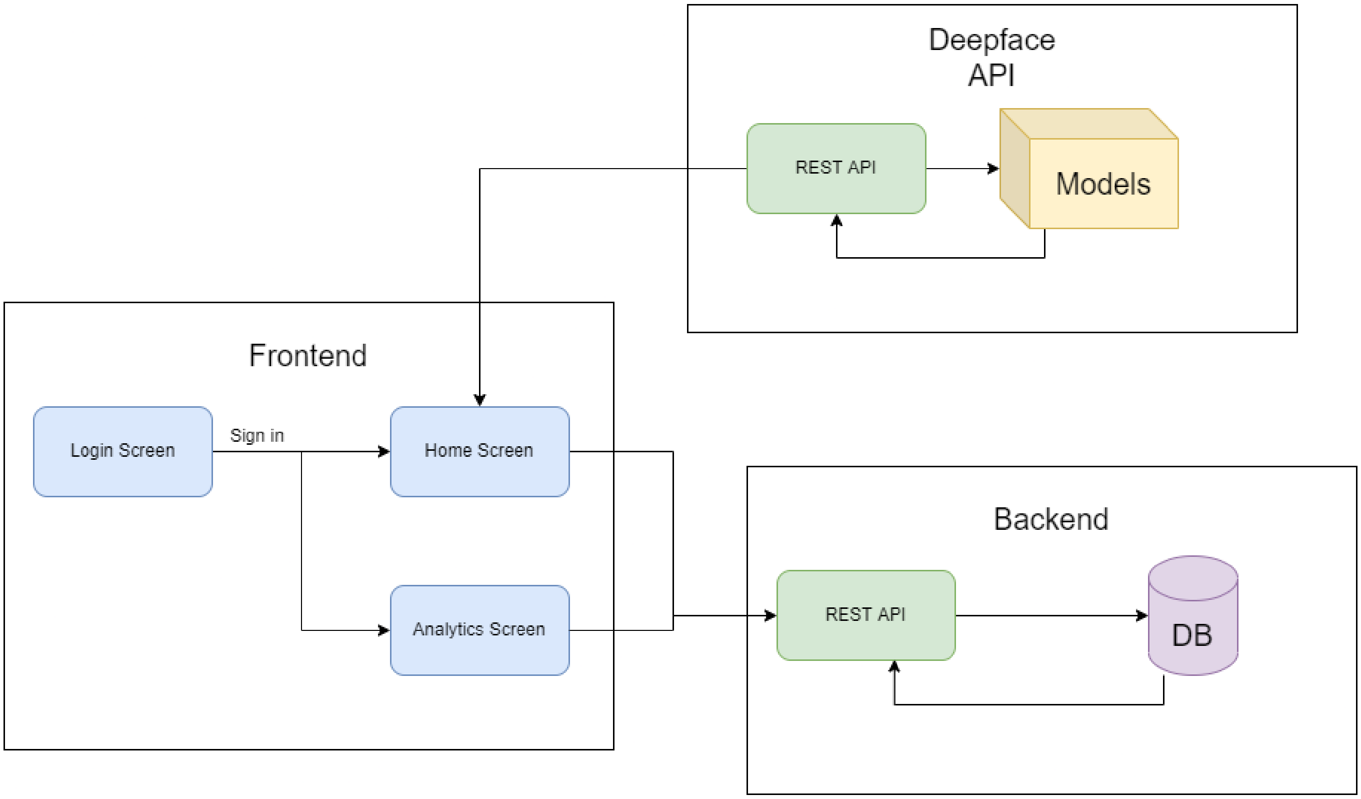
There are three screens in the frontend - Login Screen, Home Screen and Analytics Screen.

User sign is done using firebase's signinwithgoogle api in the login screen.

After that user is redirected to the home screen, where he can start evaluation sessions. At the start of the session, a new session record is created in the database, which will store the current session's recorded data about the user's emotional state. This new session's id is returned to the frontend.

Then, a webcam video stream starts on the home screen. Two frames per second are recorded from the stream and sent to the deepface api which runs its classification models on the frames and then sends the classification data on the frames. Then this classification data is stored in a buffer. Once every 10 seconds this buffer data is sent through the REST API to the backend server which stores it inside the current session document in the database.

In the analytics screen, HTTP requests are sent to the backend server, which preprocesses the data of all the sessions and then sends it to be shown in the form of charts on the frontend. The charts are shown session wise in the frontend with the option to switch between the sessions.



An illustration of the above architecture

Decisions/Choices in the Architecture

- Database
  Nosql database chosen instead of sql database because Nosql databases are more flexible as they have dynamic schema, and we have to store hierarchical data, hence nosql database is best suited for this web application's needs.

- AI Functionality
  Separate REST api service is used for AI functionality as it was difficult to use deepface libraries in frontend or backend. Using this approach keeps the frontend, backend and AI service separate, hence frontend developers can focus only on how data is visualized, backend developers can focus only on how the data is being stored and served to the frontend and the AI engineer can focus only on great classification accuracies. AI service was not called from the backend as then we will have to first send the image data to the server and then to the AI service which is inefficient. The deepface api was provided in the deepface's public repository on github, link.
- Taking 2 frames per second for classification
  Taking 2 frames means we are recording change in expressions every 0.5 second, and it is too little time for the human expression to show something meaningful, hence increasing these frames will not help us gain anything new. Keeping the frames at 2 also keeps the AI service from being overloaded with requests. We can also bring it down to 1 frame per second.
- Upload classification data to the backend every 10 seconds.
  This is done to avoid overloading the backend with requests, it can also be increased to 30 seconds or more.

Future Plans

Future plans include creating a tailored model for our requirements, and converting the data in meaningful medical format to be used by medical professionals. Including medical professionals in the application with access to view their patient's emotional history and reports.