EECS 582: Final Project

Initial Architecture Document

Version 1.2

Table of Contents

Team and Project Overview

Team Number: 18

Team Members: Sophia Jacob, Anna Lin, Kusuma Murthy, Nimra Syed, Nikka Vuong

Project Name: Elysian

Project Synopsis:

With the mobile app, users are given machine learning curated, swipe-based recommendations to create personalized itineraries and share travel experiences with their social feed.

Architecture Description

Overview:

The mobile app comprises five components: frontend, backend, Firebase, AWS, and the AI recommendation engine. The frontend will act as the interactive Graphical User Interface (GUI) for the users to create their profile, log in, swipe through their generated recommendations, share and explore travel experiences, and more. The backend will handle all the business logic to Firebase and AWS, as well as communicate with the AI engine, allowing for the seamless integration between the two components. Firebase will be used for authentication and storing text-based user data, which includes post details, disliked and favorited cities, user preferences, and user data. AWS will also be used as another database to store pictures for user posts. Both cloud components will query relevant information needed for the different endpoints in the app. Finally, the AI recommendation engine brings the system's core intelligence that creates real-time, dynamic traveling recommendations that are personalized to the user's profile and swipe actions.

Frontend:

The frontend is built using React Native for an iOS mobile app. With the integration of React Navigation, users can seamlessly move from various pages, such as the Home, Login, Recommendation, Favorites, and User Profile pages. It interacts with the Firebase authentication to allow users to create and log in or out of their accounts securely. When creating an account, users will be required to answer questions to build their personalized travel profile. This data will be stored in the Firebase database and later queried by the Backend and AI Recommendation Engine component to generate tailored travel destinations, which are displayed on the curated Recommendations Page. This Recommendations Page is a swipe-based interface showcasing the personalized travel recommendations, each featuring an image and a brief description along with the destination that can be accessed with a tap, to help users explore potential trips. Users can swipe right to save destinations they like, which are then added to the Favorites Page, or swipe left to dismiss destinations. The Home Page features a social media-styled feed that displays travel content from different users, creating an engaging experience.

Backend:

The backend for the mobile app is built using Flask and hosted on Render. It is responsible for integrating the Database and AI Recommendation Engine component. It will use the AI Recommendation Engine component to generate recommendations based on the user's input received from the frontend. Based on the information retrieved from the frontend, the backend will use a Python script to query the right user account logging into the session and return the respective account/profile information back to the frontend. The backend is also responsible for passing the information received from the frontend to the Database and AI Recommendation Engine components so they can perform their respective tasks, such as adding a new entry in the database or generating new recommendations. The backend manages the maintenance of the system's activity, which is not visible to the user on the frontend. Therefore, the backend will include RESTful API endpoints, built with Flask, to

help with this smooth communication between the frontend and the other components of the app.

Firebase + AWS:

The database is built using Firebase Realtime Storage. The first database table stores all user-related data, including user profiles, travel preferences, liked places, and dislikes. This table will be integrated with the authentication, using a unique user ID (UID) for secure storing and retrieving of user information. These tables will be accessed by both the Backend and the AI Recommendation Engine to generate accurate and personalized travel locations. The authentication system leverages Firebase Authentication, which provides a secure and user-friendly way to manage sign-up and login features. The flow of the authentication is as follows: when a user opens the mobile app for the first time, they can either create an account or log in using their existing credentials and/or email account. Once the user signs in, Firebase handles the OAuth process in the background and generates a UID (stored in Firebase as mentioned) for that user. This UID acts as the main identifier for the user throughout the system, linking their preferences, questionnaire responses, and AI-generated recommendations. After a successful login, Firebase returns a secure ID token, which the mobile app stores locally and sends with each request to the Flask backend. The backend uses Firebase's Admin SDK to verify the token before performing any actions, which ensures that all data access is authenticated.

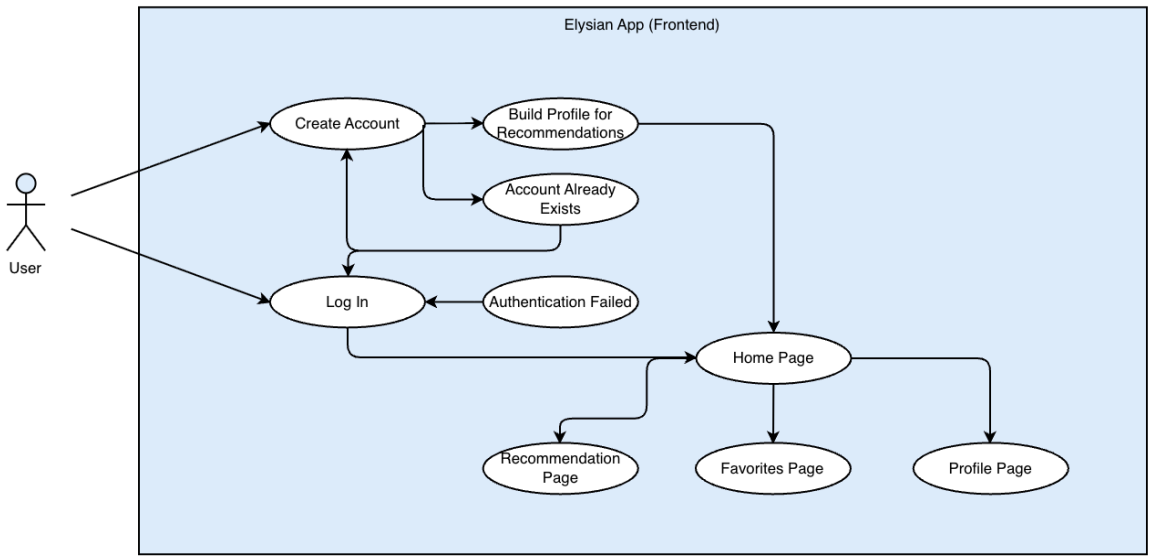
AWS S3, IAM, Lambda functions, and API Gateways are used to allow the app to store the user's uploaded images. The users have the choice to either upload an image from their image library or manually take a picture. In order for users to be able to see a feed-like style of all the user's posts, the front end pulls the uploaded pictures from AWS's S3 bucket. IAM roles are used to securely grant Lambda functions the necessary permissions to access the S3 bucket. This helps ensure that only the authorized backend processes can read and/or write to/from the storage.

AI Recommendation Engine:

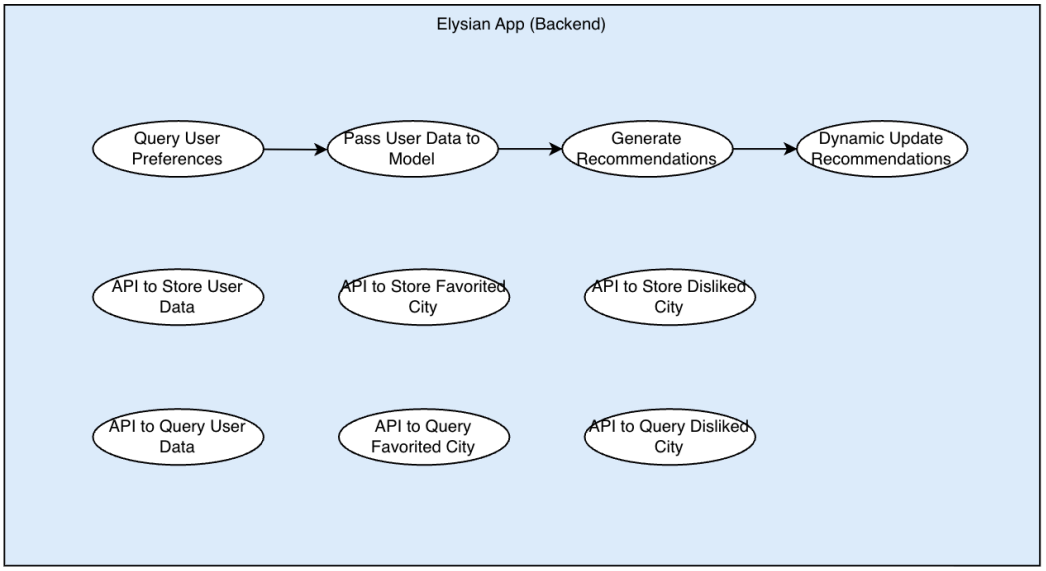
With this mobile app, the team aims to create a tailored AI recommendation engine that will utilize user preferences as inputs to the model and convert them to feature vectors to optimize the different user data and their weights in the model. When the user sets up their profile, they are asked to answer a series of questions, which serve as the initial user vector for the model. The model will start to recommend certain cities based on these preferences, which the user can then swipe left to dislike the recommendation or swipe right to like the recommendation. With every swipe, the user's embedding vector will get adjusted to match this new data of favorited and disliked cities, which helps give more precise recommendations. All the initial inputs for this model will be stored in the Database component of the system and will be queried when needed in the AI Recommendation Engine. This will allow the model to learn from these inputs to output city locations that fit the most requirements. The favorited and disliked cities are also stored in their own tables, respectively. The engine will be trained with previous datasets of user data and will use best practices when creating the model. The ML model is a two-tower architecture trained with binary cross-entropy loss and using the sigmoid activation function. Trained on top of this is the dynamic layer, which generates more recommendations based on the user’s favorites. To facilitate the deployment of the engine, the Backend component is used to host the model and communicate with the endpoints. Finally, the favorite cities will be stored in the Database component in a separate table and will be displayed by the Frontend component once the user fills their profile information.

UML Diagrams

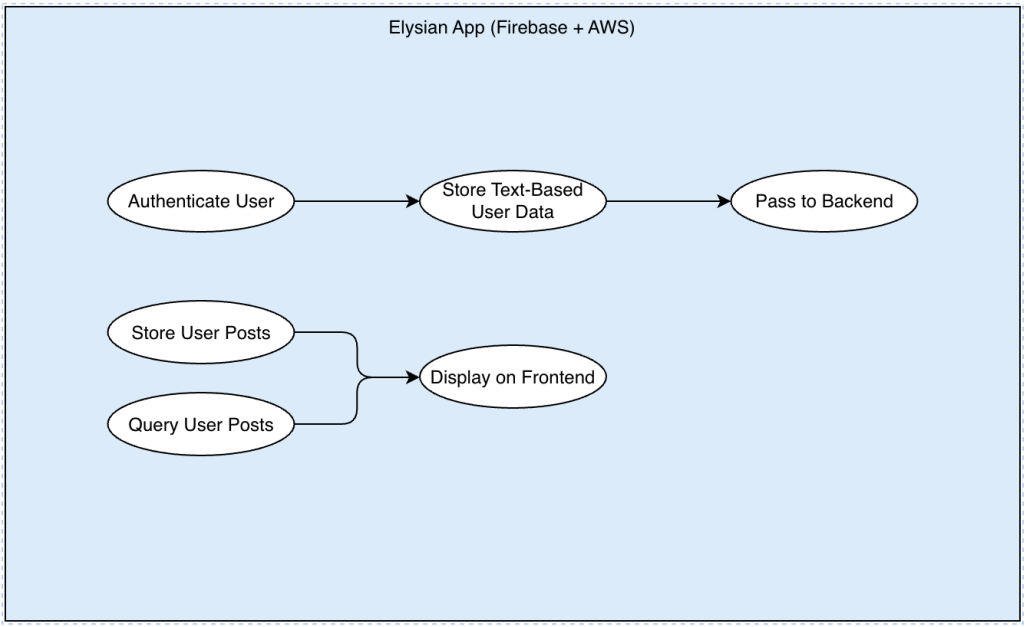
Frontend System:



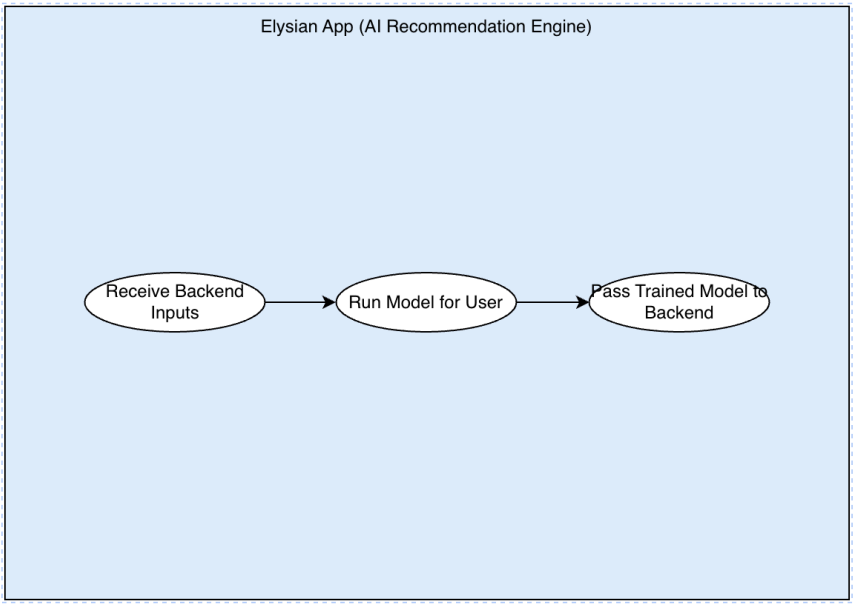
Backend System:



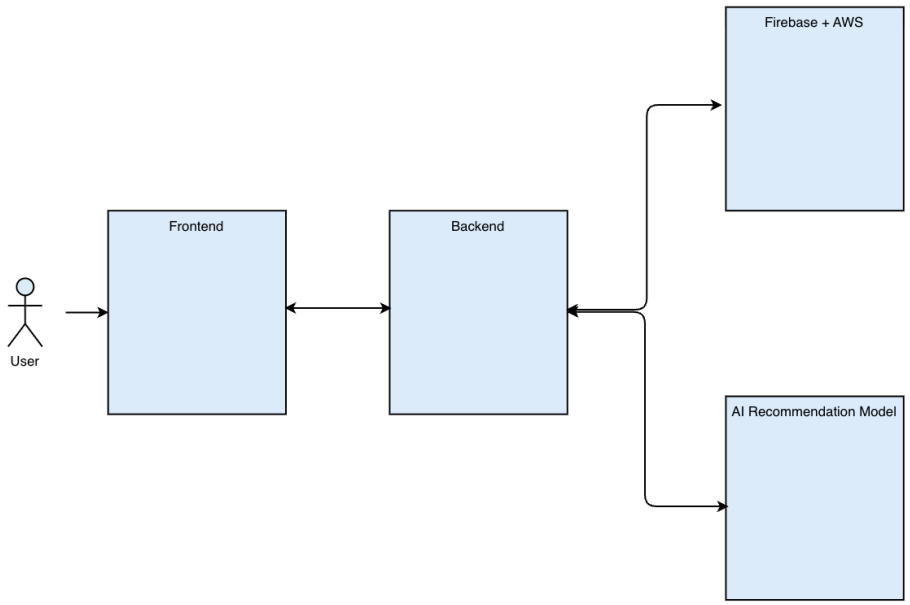
Firebase + AWS:



AI Recommendation Engine:



Holistic System:



**** Note: All our Project Meeting Logs will be housed in the GitHub Repository on the Wiki Page. Please reference it as needed. We also have our Sprint Board and tickets created on the GitHub page.**