Architectural Design Document

Globetrotter

1 Introduction

This document presents the architecture for the software for the Globetrotter project. This project performs functions such as allowing the user to generate custom trip itineraries based on certain form select options/constraints (trip length, temperature preference, number of destinations, etc). The system generates detailed itineraries accompanied by an interactive map, powered by the Google Maps API, displaying all planned destinations. Users can then save these itineraries to their profile for later reference.

1.1 System Objectives

The goals of this project are to provide an easy and efficient way for travel lovers to find new destinations and places to visit. The OpenAI API integration allows for users to be exposed to new cultures, countries, environments, etc. that they wouldn't typically search for. The ability to save these itineraries to their profiles allows users to return to these itineraries in the future and make these trips a reality. Additionally, the Google Maps API integration enhances the experience by providing a visual representation of each trip.

1.2 Hardware, Software, and Human Interfaces

The hardware interfaces consist of user devices such as smartphones (iOS) where users will be able to interact with my app. For networking, the app will primarily rely on wireless connections (Wi-Fi) to facilitate interactions. It will connect to Firebase for data storage and retrieval, as well as the Google Maps API and OpenAI API for various functionalities. On the software side, Firebase will store user data, including saved itineraries, travel destination wishlists, and profile customization options. The OpenAI API will generate trip itineraries based on user-inputted form preferences. The Google Maps API will visualize these itineraries by plotting destination pins on an interactive map. The backend will use a threaded server to efficiently handle multiple user requests and interactions. For human interfaces, the Globetrotter UI will provide a user-friendly experience, featuring a travel preferences form for generating itineraries, an interface for customizing and editing user profiles, and a settings page for managing app preferences.

2 CSCI Descriptions

This application will be made in three components: the **client side CSC**, the **server side CSC**, and **database CSC**.

- **Client CSC**

  The client will consist of multiple pages, each invoked through functions and connected via a tab menu bar. These pages operate at the same level, with their features implemented as component functions within them. Since data is shared through the server and database, each page can be considered an independent CSU. However, the settings page will be accessed via a sidebar that appears when the user presses the settings icon in the top-right corner of the profile page, making it part of the profile CSU rather than a separate one.

  The Client CSU will contain:
  - Tab Menu Bar CSU: The tab bar will allow the user to navigate to different pages in the app.
  - Home Page CSU: The main page of the app, which will contain:
    - Travel Preferences Form CSU: A modal where users specify travel preferences (environment type, trip duration, etc).
    - Results CSU: Once user presses submit, the modal closes and the resulting itineraries are displayed on the home page, including the maps generated with Google Maps API.
  - Profile Page CSU: The page presenting profile information and settings page. There will be a separate smaller tab menu bar in the profile page for saved itineraries and travel wishlist.
    - Saved Itineraries CSU: Users can view all of their saved itineraries and can save up to 5. They will be buttons stacked vertically and will open up all of the details in a modal when clicked. You can edit the information in the itinerary or delete it if you don't need it anymore.
    - Travel Wishlist CSU: This will be a space for users to remember places that they want to travel to. Users can input cities, countries, specific attractions, etc. There will be an option to communicate with OpenAI API and create an itinerary directly from each option listed.
    - Settings Page CSU: The settings page will be triggered from the settings icon in the top right of the profile page and will open a sidebar with options to sign out, edit profile, change password, etc.

- Login Page CSU: Used for users to log into their account.
- Sign Up Page CSU: Used for users to create an account.

- **Server CSC**

  The server will be run using Expo, a framework for building cross-platform applications. The server will not have a graphical user interface; instead, it will consist of function calls that fetch data from the database and return results to the client.

  The Database CSC will contain:
  - Database API CSU: Handles selecting the appropriate database and retrieving specific data as needed.

2.1 Concept of Execution

To run Globetrotter, the server must be started first. Once the server is running, it will establish a connection to Firebase, enabling data storage and retrieval. After the server is active, clients can launch the application and connect to the server to access necessary information.

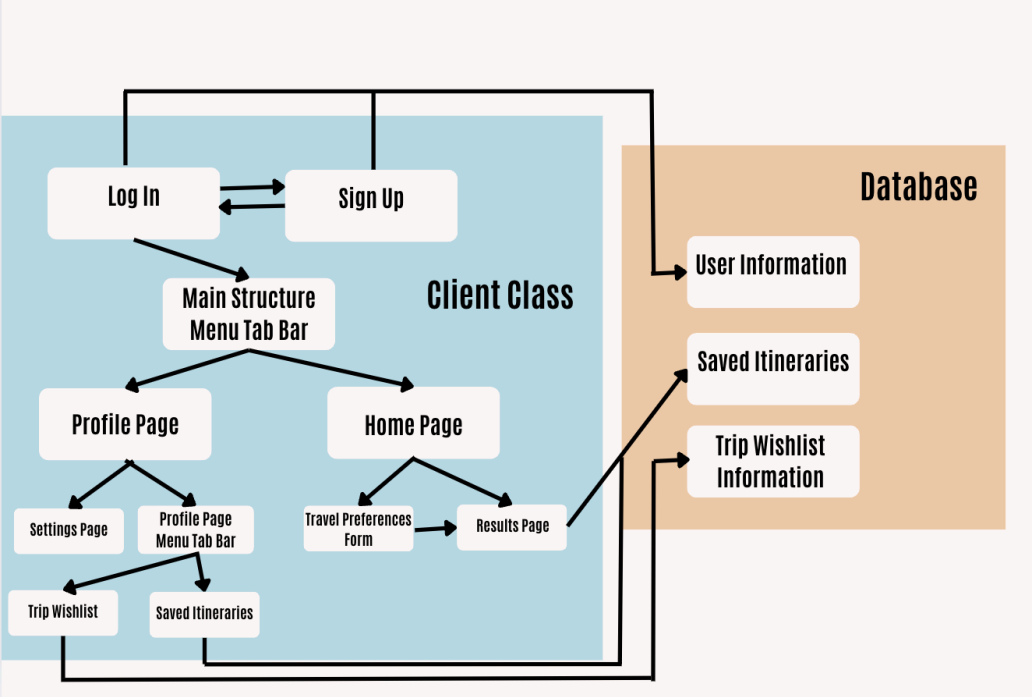
Upon opening the application, users will be prompted to log into their accounts. After authentication via the database, they will gain access to the pages and information available to them. The application will start on the Home Page, where users can navigate to other sections using the menu bar. Any actions taken—such as creating or saving itineraries or adding destinations to the travel wishlist—will be reflected in the database in real time.

2.2 Interface Design

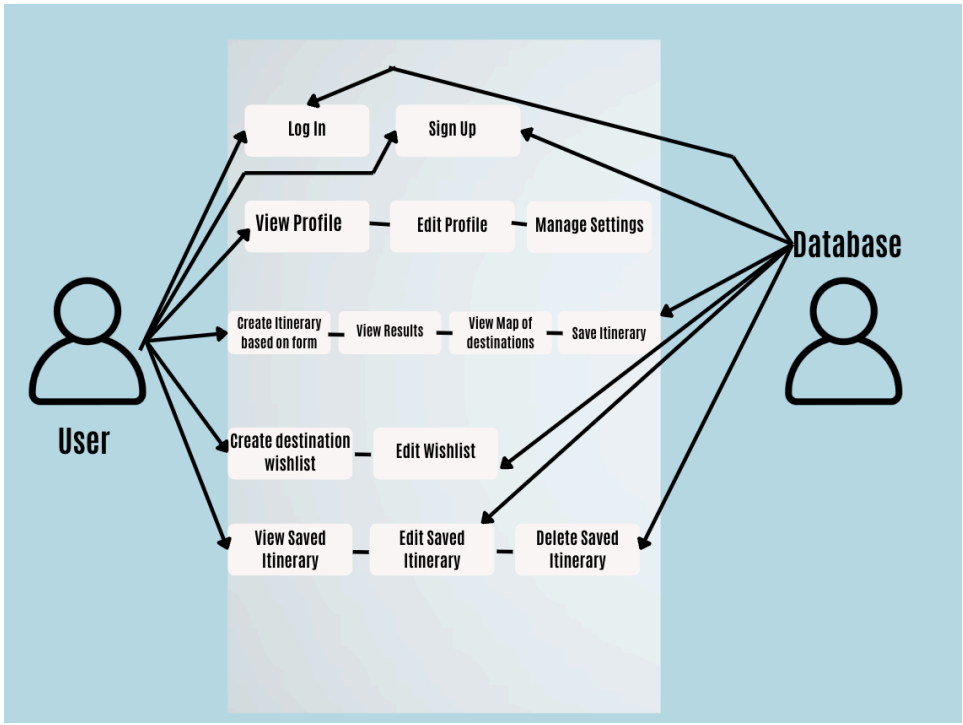
The following provides diagrams representing the interface designs for this project.

2.2.1 Interface Identification and Diagrams

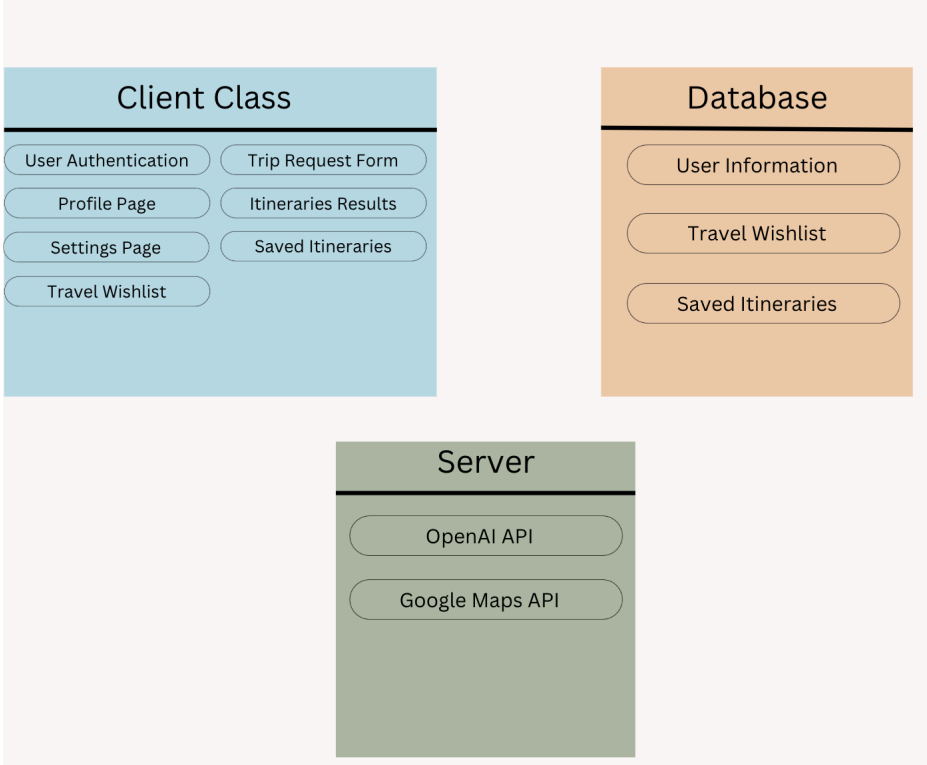
2.2.1.1 Client Class and Function Modules



2.2.1.2 Use Case Diagram



2.2.1.3 Package Diagram



2.2.2 Project Interactions

Client-Server Interactions:

The interaction between the client and server will occur through RESTful API calls using HTTP protocols. When a user performs an action on the client side, such as saving an itinerary or submitting travel preferences, the client will formulate a request and send it to the server. The server will then process this request, perform specific operations and return a response. The client will receive this response and update the UI accordingly.

Server-Database Interactions:

The server will interact with Firebase to store/retrieve data. Some of these interactions will include user authentication, storing/retrieving saved itineraries, and managing travel destination wishlists.

OpenAI API Integration:

The server will communicate with the OpenAI API to generate custom trip itineraries based on user preferences. When a user submits travel preferences through the form, the client will send the form data to the server. The server will then format this data into a prompt for the OpenAI API. The server will then send this request to the

OpenAI API via HTTPS. The OpenAI API will process this request and return an itinerary. Finally, the server will parse the response and send the formatted itinerary back to the client.

Google Maps API Integration:

The application will interact with Google Maps to visualize the generated itineraries. When an itinerary is generated or retrieved, the application will extract destination information from the itinerary, send requests to the API, receive map data, and display the map with plotted destinations.

All of the other components in the application will communicate through a state management system. This will allow for modular design where components can be developed and tested independently.

3 Preliminary User Manual

To get started, download the Globetrotter app on your mobile device. Securely log in to the app with a username/email, or register your username/email to get started.

Once logged in, you'll see the home page which consists of a button that contains the travel preferences form modal when clicked. To generate an itinerary, click that button and fill out the form. Once you are done, click submit, and you will now see your itinerary results with a map generated with pinned destinations. You can filter these results by price, distance, etc. and you can save any itinerary that you want to your profile. The menu bar at the bottom of the page includes tabs for **profile** and **home** (current page).

Tapping the **profile** icon takes you to your profile page, where your username will be displayed along with a profile picture that you can add from your camera roll. There will also be a settings icon on the top right of the menu, that will open the settings menu when clicked. Here, you can edit your profile information, change your password, or log out. On the profile page, there is another menu bar. One of the tabs contains your travel wishlist, where you add future destinations that you want to travel to. The other tab features itineraries that you generated that you saved to your profile. You can edit or delete these itineraries.

To get back to the main page, tap the **home** icon.