Software Design Detailed Document

Globetrotter

1 Introduction

This document presents the detailed design 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 Detailed Design Descriptions

The following sections contain the descriptions of the details of the design of Globetrotter CSCI. The design is systematically described in terms of the CSCs for the project, and the CSUs for each CSC.

2 Detailed Design Descriptions

2.1 CSC - <u>Tab Menu Bar</u>

This bar allows the user to navigate between the Home and Profile pages.

2.2 CSC - <u>Home Page</u>

The main page of Globetrotter that contains the following:

2.2.1 Travel Preferences Form CSU

A form/modal used for users to plan a trip to a new destination. Preferences will include temperature, vacation style, number of destinations, etc.

2.2.2 Results CSU

Used to show the results from the submitted travel preferences form. Will have a filter option to filter the itineraries by budget, a map indicating the destinations for the trip, and an option to save the itinerary to the user's profile.

2.3 CSC - <u>Profile Page</u>

This page will have all of the user information, including the profile picture, username, settings page, etc.

2.3.1 Saved Itineraries CSU

Users can view all of their saved itineraries and save up to 5. They can edit the itineraries or delete them.

2.3.2 Travel Wishlist CSU

Users can document different cities, countries, attractions, etc that they want to travel to. There will be an option to communicate with OpenAI API and create an itinerary directly from each place.

2.3.3 Settings Page CSU

Users can click the settings icon in the top right of the profile page. There, they can edit their username/password, change their profile picture, sign out, etc.

2.4 CSC - <u>Welcome Page</u>

This is the first page that the user will see when they open the app. From here, they can navigate to the following:

2.4.1 Login Page CSU

Used for users to log in to Globetrotter.

2.4.2 Sign Up Page CSU

Used for users to create an account.

3 Class Descriptions (for each CSU)

3.1 Travel Preferences Form CSU Class

Purpose: Manages the creation and submission of user travel preferences to generate trip itineraries.
Fields:

- Temperature preference: User's desired temperature range for trip.
- Vacation Style Options: Available vacation style choices (business, bachelorette, honeymoon, etc)
- Destination Count: # of desired destinations for trip
- Environment Type: Desired environment for trip (beach, lake, mountains, etc)
- Travel Constraints: Additional user preferences

Methods:

- collectUserPreferences(): Gathers user input from the preferences form and validates it.
- submitUserPreferences(): Sends input to the backend to be processed by openAI API.
- resetForm(): Resets the form.

3.2 Results CSU Class

Purpose: Handles display and management of generated travel itinerary results.
Fields:

- Generated Itineraries: A list of generated/created itineraries.
- Filter Criteria: Current filtering parameters.
- Map View: Interactive map representation of trip.

Methods:

- applyBudgetFilter(): Filters itineraries based on user's budget constraints (slider).
- showDestinationMap(): Generates map with pins in destinations.
- saveItineraryToProfile(): Saves selected itinerary to profile.

3.3 Saved Itineraries CSU Class

Purpose: Manages user's saved itineraries in their profile.
Fields:

- Saved Itineraries: List of trips that the user has saved (max of 5).

Methods:

- listSavedItineraries(): Retrieves and displays saved itineraries in user profile.
- editItinerary(): Allows modification of a saved itinerary.
- deleteItinerary(): Removes a specific itinerary from saved list.

3.4 Travel Wishlist CSU Class

Purpose: A place for user's to remember/list places they want to travel to.
Fields:

- Wishlist Items: Destinations users want to visit

Methods:

- addWishlistItem(): Adds a new destination to the wishlist
- removeWishlistItem(): Deletes a destination from the wishlist
- generateItineraryFromWishlist(): Creates an itinerary using OpenAI for a wishlist item.

3.5 User Settings CSU Class

Purpose: Manages user account settings.
Fields:

- Profile Image Manager: Current profile image.
- Current User: Reference to the logged-in user and their information.

Methods:

- updateUsername(): changes the user's username, reflecting in the database.
- updatePassword(): changes user's password (with security checks), reflecting in the database.
- uploadProfilePicture(): Manages profile picture upload.
- signOut(): Signs current user out of Globetrotter.

3.6 Login Page CSU Class

Purpose: Manages user authentication and log in.
Fields:

- Authentication with MySQL: handles login and validates user info.
- Lockout Duration: Time for account lockout after failed login attempts.

Methods:

- authenticateUser(): Verifies user credentials.
- resetPassword() Initiates password recovery.

3.7 Sign Up CSU Class

Purpose: Creates an account for new users.
Fields:

- Registration Form: Collects user account information.

Methods:

- validataRegistrationDetails(): Checks if provided details meet requirements (not existing user, password length, etc).
- createUserAccount(): Registers new user to database.

4 Detailed Design Descriptions (MySQL)

4.1 User Database CSC

New users will be registered into the user database once they log in. With each user, there will be an id, a fullname, username, email, password (hashed), and the timestamp that the user was created at.

4.2 Saved Itineraries CSC

When users save an itinerary, it will be added to a saved itineraries database. It will grab the user's username from the user database, and save up to 5 itineraries. If a user edits or deletes an itinerary from their profile, the database will be adjusted accordingly.

4.2 Trip Wishlist CSC

When users add a new wishlist destination to their profile, it will be added to a wishlist database. This database will also grab the user's username from the user database.

5 Detailed Interface Designs

The interactions inside client side will be done with React Native Expo's useRouter hook to switch between pages. I will also import { Tabs } from expo-router to create my custom tab navigation between the home and profile pages, and between the wishlist and saved itineraries CSUs in the profile page.

The application will connect to MySQL using Python in the backend (Flask). Within MySQL, there will be a user table with the rows representing new users, and the columns representing information such as username, fullname, password, id, etc. To keep the password secure, I am using bcrypt password hashing. When a new user signs up, a new row will be added to the user database table.

I will have two more tables within my database: Travel Wishlist and Saved Itineraries. Both of these tables will grab the id of the user that is currently signed in from the users table. The Travel Wishlist table will contain information about the destinations that each user inputs into their wishlist on their profile. The saved itineraries table will contain information about the itineraries that user's decided to save after completing the form.

The OpenAI API will be used when users submit their travel preferences form. In the backend, their answers to the form will be crafted into a sentence/query that will be an input for OpenAI. OpenAI will then create an itinerary that will be organized into an itinerary also in the backend. The front end will then grab the itineraries through HTTP methods and display them on the user interface.

6 Detailed Data Structure Descriptions

6.1 Users Database Data Structure

Structure Design: MySQL table with following attributes:

- Id: unique id for each user
- Fullname: full name of the user
- Username: unique username of the user
- Email: user's email address
- Password: hashed password (using bcrypt)
- Created_at: timestamp of user account creation
- Profile_picture: optional reference to user's profile picture

6.2 Saved Itineraries Data Structure

Structure Design: MySQL table with following attributes:

- user_Id: id referencing Users Database
- Itinerary_id: unique id for each itinerary
- Itinerary_name: custom name for each itinerary
- Destinations: List of trip destinations
- Start_date: planned trip start date
- End_date: planned trip end date
- Created_at: timestamp of itinerary creation
- Travel_preferences: list of preferences that were submitted in form
- Location_data: store url to map generated by google maps for itinerary

6.3 Travel Wishlist Data Structure

Structure Design: MySQL table with attributes:

- Wishlist_id: unique id for each wishlist entry
- user_Id: id referencing Users Database
- Destination_name: name of desired destination
- Added_date: timestamp of wishlist entry creation

6.4 Travel Preferences Data Structure

Structure Design: A form/object with the following attributes:

- Temperature_preference: desired temperature range
- Vacation_style: type of vacation
- Destination_count: number of desired destinations
- Environment_type: preferred environment
- Other_constraints: additional user preferences inputted to OpenAI

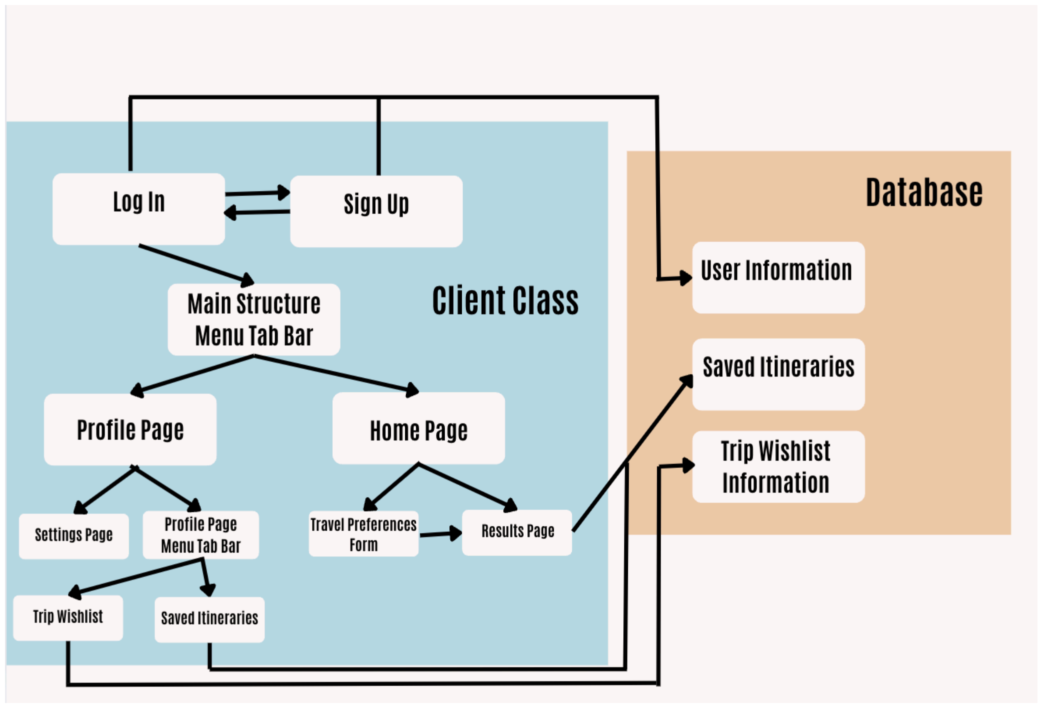
6.5 Results Data Structure

Structure Design: Generated by OpenAI API, with following attributes:

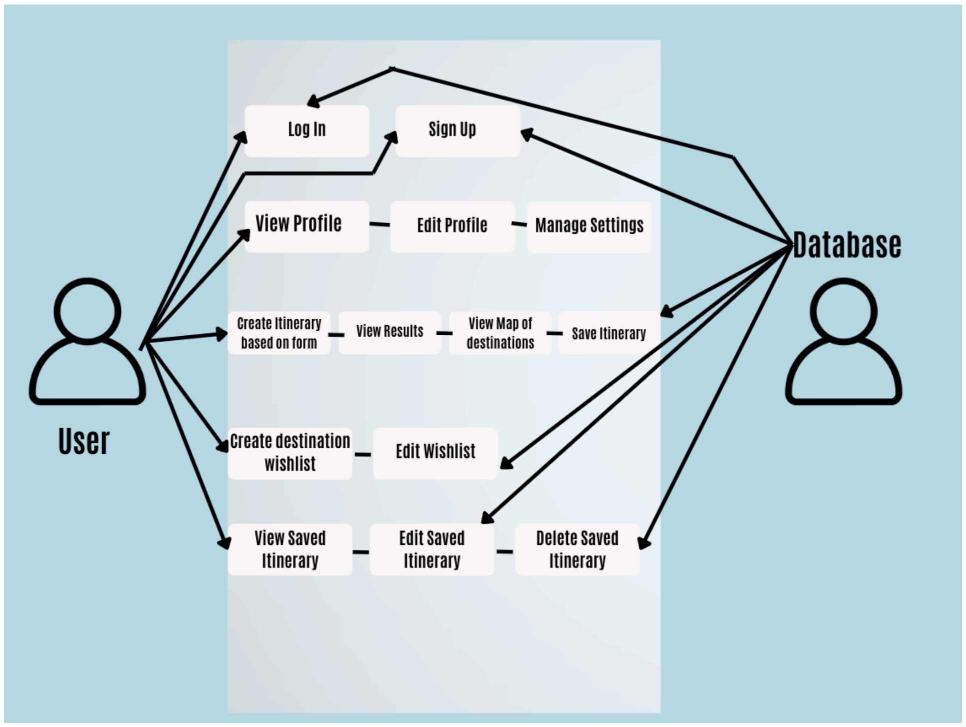
- Itinerary_id: Unique id for id
- Estimated_budget: calculated trip cost
- Destinations: list of destinations
- Travel_preferences: list of preferences that were submitted in form
- Location_data: store url to map generated by google maps for itinerary
- Start_date: planned trip start date
- End_date: planned trip end date
- Created_at: timestamp of itinerary creation

7 Database Design and Descriptions/Diagrams

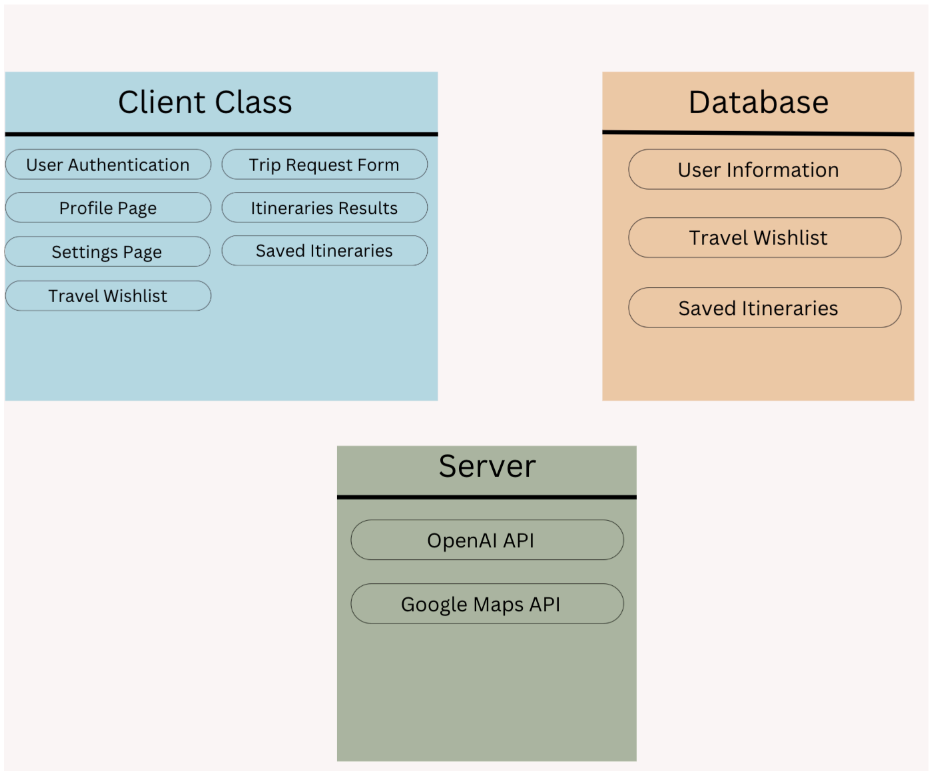
7.1 Client Class and Function Modules



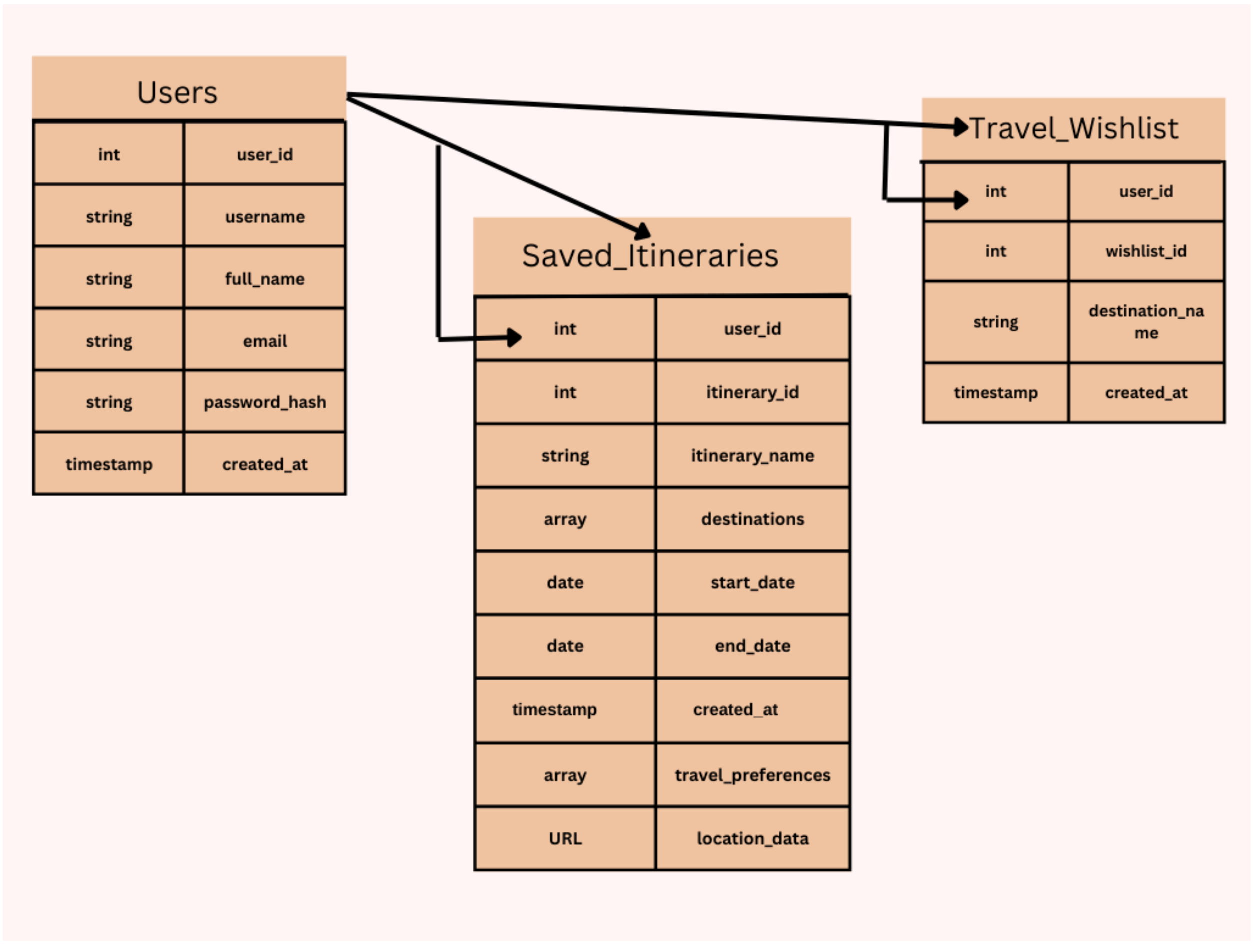
7.2 Use Case Diagram



7.3 Package Diagram



7.4 Entity-Relationship Diagram



Users
int
user_id
string
username
string
full_name
string
email
string
password_hash
timestamp
created_at
Saved_Itineraries
int
user_id
int
itinerary_id
string
itinerary_name
array
destinations
date
start_date
date
end_date
timestamp
created_at
array
travel_preferences
URL
location_data
Travel_Wishlist
int
user_id
int
wishlist_id
string
destination_name
timestamp
created_at