Project Design Document
Project Name: BluMap
Team 23
Members: Daniyal Bekinalkar, Mert Karabulut, Stanley Kim, Meet Patel

Purpose

In the world of travel planning, many travelers face the combined challenge of creating detailed, practical itineraries and the wish to share and work together on these experiences with others, a requirement that existing travel planning solutions, such as TripAdvisor and Expedia, fail to fully meet. These current platforms often concentrate on wider aspects of trip organization, like reservations and destination recommendations, but miss thorough, hour-by-hour itinerary customization and a strong, community-led system for trip sharing, communication, and joint planning. Our website application, BluMap, is ready to transform this area by launching an all-encompassing travel planning platform that not only enables travelers to carefully plan every hour of their day with individualized activities and streamlined routes, but also promotes a lively community setting where these plans can be shared, talked about, and improved with other users. This method distinguishes our project by combining detailed trip planning with social interaction and community involvement, elements that are noticeably missing in present travel planning tools, thereby providing a more complete and captivating travel planning experience.

Functional Requirements:

Core

1. As a user, I would like to view events, attractions, restaurants, hotels, and transportation in a location so that I can add them to my itinerary.
2. As a user, I would like to receive timely notifications for upcoming activities.
3. As a user, I would like to receive notifications about changes in the itinerary and important updates related to my trip.
4. As a user, I would like real-time travel planning capabilities that lets me adjust my event itinerary on the go so that I can ensure flexibility during the trip.
5. As a user, I would like personalized recommendations based on current location and preference when adjusting events on an active itinerary.
6. As a user, I would like to save my travel history, including past itineraries.
7. As a user, I would like to be able to view my saved travel histories.
8. As a user, I would like to access my frequently planned activities.
9. As a user, I would like journey optimization strategies, including route optimization, suggesting attractions, and ordering for visiting attractions, to maximize efficiency and enjoyment so I can reach the destination in the quickest time.
10. As a user, I would like to be given travel and itinerary recommendations based on my preferences.

11. As a user, I would like to be given travel and itinerary recommendations based on the trip or my location.

Utils

1. As a user, I would like to be able to use the site without an account to view others' itineraries.
2. As a user, I would like to be able to create and register an account.
3. As a user, I would like to be able to log in for a personalized experience.
4. As a user, I would like the ability to personalize the look and feel of the application by choosing from different themes or color schemes.
5. As a user, I would like to create detailed, hour-by-hour trip itineraries for different types of trips so that I can plan my travel effectively.
6. As a user, I would like the option to use the application in multiple languages to cater to a diverse user base.
7. As a user, I would like a tutorial so that I can use the app correctly.

Social

1. As a user, I would like a post-reputation system so that I can review other user content and their created itineraries.
2. As a user, I would like to be able to get badges or rewards, for achieving certain milestones or actively participating in the community.
3. As a user, I would like to be able to upload photos to add to my itinerary.
4. As a user, I would like to be able to see my photos from my past travels.
5. As a user, I would like the ability to share my trip itinerary, experiences, and photos with others on the social tab of the application.
6. As a user, I would like the ability to share my trip itinerary, experiences, and photos on other social media platforms directly from the application.
7. As a user, I would like to view emergency contact information, local emergency services, and travel advisories for each destination.
8. As a user, I would like to manage my emergency contact information, local emergency services, and travel advisories for each destination.
9. As a user, I would like to leave reviews and ratings for specific activities or locations within my itinerary, contributing to the overall community-driven system.
10. As a user, I would like to comment on itineraries designed by myself and other users.
11. As a user, I would like to be able to activate an itinerary with another user as we go through the active itinerary together.
12. As a user, I would like to be able to follow other users.
13. As a user, I would like to be able to get trip or itinerary recommendations based on my preferences or location.
14. As a user, I would like to be able to block other users.
15. As a user, I would like to be able to make my account public or private.
16. As a user, I would like to be able to add friends on the platform.

17. As a user, I would like to be able to view my user profile as well as other user profiles.
18. As a user, I would like to be able to chat with other users through the application.
19. As a user, I would like to be able to anonymize itinerary event information.

Admin

1. As an admin, I would like to manage user accounts, which includes viewing, suspending, or deleting them if necessary.
2. As an admin, I would like to access analytics and insights on user engagement.
3. As an admin, I would like to access information about popular destinations.
4. As an admin, I would like to access frequently planned activities.
5. As an admin, I would like to moderate user-generated content, including reviews, comments, and uploaded media, to ensure compliance with community guidelines and prevent inappropriate or offensive content.
6. As an admin, I would like a reporting system where users can flag content or report issues.
7. As an admin, I would like to verify user identities, especially for users who contribute significantly to the community, to enhance the credibility of reviews and recommendations.
8. As an admin, I would like the ability to send important announcements or notifications to users, either individually or as a group, to communicate updates or address issues.

Non-Functional Requirements

Architecture and Performance

In the development of our application, we are adopting a modular architecture that separates the frontend and backend components to streamline the development process and avoid any potential integration challenges. The frontend will utilize React to provide a dynamic user interface, whereas the backend, powered by Node.js through Express.js, will focus on efficiency and scalability. We aim to optimize backend performance to target a response time of under 500 ms to ensure a seamless user experience. Achieving this enhanced responsiveness is vital for maintaining user satisfaction and engagement.

System Requirements

Our system requirements will intentionally be designed to be accessible and user-friendly, ensuring that users can interact with our platform using any device with internet connectivity. Whether accessed through a laptop, desktop, tablet, or smartphone, our web-based application will offer flexible use across all modern devices. However, using up-to-date web browsers will be necessary to ensure complete functionality and the optimal user experience on our platform. We are also committed to maintaining a high level of system uptime for reliability. Our objective is to provide consistent, uninterrupted access to our web-based project, emphasizing our dedication to system dependability and user convenience, making the platform dependable and accessible at all times.

Scalability and Reliability

For scalability, we will initially support 100 simultaneous requests with a clear plan to optimize the program to handle 1,000 simultaneous requests as the demand for more users increases over time. Additionally, we will conduct rigorous testing and quality checks to identify any issues within our program to ensure a reliable and trustworthy application. We will use a rigorous testing process that includes using Python Selenium. This will allow us to automate user interactions for testing, which can simulate user registration, explore different features, and verify common tasks. As part of our testing process, we will closely monitor and measure the error rate to ensure the application's repairability and our goal is to keep the error rate below 1% to provide an enhanced user experience and quickly address any issues that may arise.

Security

Security will be a critical priority for BluMap, as we will be housing various pieces of sensitive information. To prevent any security incidents and address potential concerns, we will implement Auth0 for robust user authentication and use PostgreSQL for secure data storage to effectively protect users' data. Furthermore, we will provide the option for users to keep their itineraries private, ensuring these remain confidential until after their trip concludes. This feature will guarantee that users' privacy is maintained throughout their travels. Moving forward, we will continuously prioritize and strengthen our security measures to safeguard user information and uphold their trust.

Usability

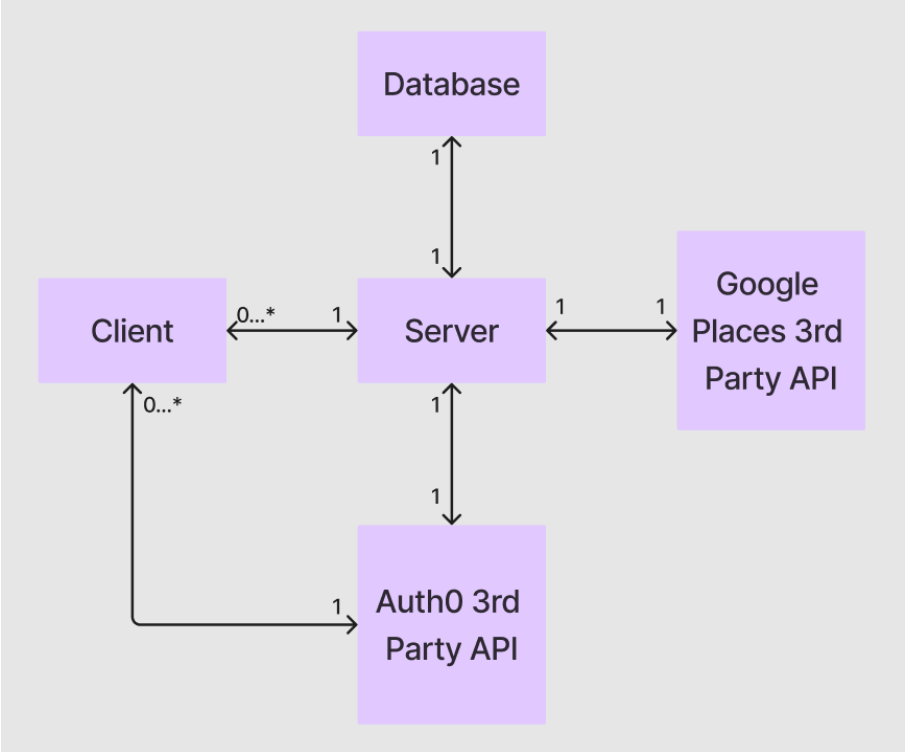
To enhance usability, we will design a user-friendly interface that ensures ease of navigation. This will include the integration of a tutorial, advanced search and recommendations, streamlined profile management, and timely notifications and alerts. By concentrating on these aspects we will strive to make the app user-friendly, providing a complete travel planning experience.

Deployment

Regarding deployment, we plan to strategically launch the web application on AWS, utilizing Kubernetes for optimal accessibility and resource management. This approach is in line with our objective to make the application universally accessible, catering to any target audience we aim to reach. The adoption of this deployment strategy will not only facilitate the community's growth at any desired rate but will also ensure enhanced scalability and constant uptime, reinforcing the reliability and reach of our application.

Design Outline

High-Level Overview



Our project is a web application designed to share itineraries that users have created. Each user can create itineraries and share them on social media, allowing others to view and utilize them. Operating on the client-server model. We chose the client-server model for its scalability and efficiency in handling multiple simultaneous user requests. This model divides the workload between service providers (servers) and service requesters (clients), facilitating a distributed architecture that can serve a large number of users without degradation in performance. Our platform also utilizes the NodeJS framework in JavaScript to efficiently manage concurrent access from multiple clients. The server seamlessly processes client requests, interacts with the database to store or retrieve data, and provides timely feedback to users as necessary.

System Components:

1. Client:
    a. Purpose: Serves as the user interface for interaction with the system. It provides a responsive and intuitive experience tailored to the needs of travelers.
    b. Usage:
        i. Client provides users with an interface to interact with the system, which facilitates the creation, viewing, and sharing of itineraries.
        ii. Client sends requests to the server, including user actions such as creating, updating, or requesting itineraries.

iii. Client receives and interprets responses from the server and renders changes in the user interface, such as displaying a newly created itinerary or showing updates to an existing one.

2. Server:
    a. Purpose: Acts as the central processor for the application, handling requests from clients. It performs data operations by communicating with the database and integrates external APIs to enrich the application's functionality.
    b. Usage:
        i. Server receives and handles requests from multiple clients and handles them efficiently.
        ii. Server interacts with the database to perform operations such as storing new itineraries, updating existing ones, or fetching itineraries for user requests.
        iii. Server utilizes the asynchronous capabilities of NodeJS to manage concurrent access from various clients.

3. Database:
    a. Purpose: A secure repository for storing all application data, including itineraries, user accounts, and shared content. It ensures data integrity and quick access to stored information.
    b. Usage:
        i. Database stores all data related to itineraries, user accounts, and shared content, ensuring that data is available and secured.
        ii. Database efficiently processes queries from the server to retrieve or modify data based on user actions and sends the extracted data back to the server.

4. Auth0:
    a. Purpose: Manages user authentication and account information, simplifying the login process across various methods and enhancing security.
    b. Usage:
        i. Auth0 will be used for authentication and user account information. It allows a much easier development flow than designing individual login flows for all login methods.

5. Google Places API:
    a. Purpose: Assists in locating tourist attractions for trip planning, offering users detailed information about destinations. Also calculates travel times between destinations, facilitating efficient itinerary planning.
    b. Usage:
        i. Google Maps Places API will be used to help search for tourist attractions in locations that a user is designing a trip for.
        ii. Google Maps Places API will be used for giving location-based activity recommendations.
        iii. Google Maps Routes API will be used for calculating travel time.

Interactions Between System Components:

1. Client-Server Interaction: Users interact with the client interface to perform actions (create, update, view itineraries), which are sent as requests to the server. The server processes these requests, interacting with the database or external APIs as needed, and returns the relevant data or confirmation back to the client.
2. Server-Database Interaction: The server requests data operations from the database (e.g., storing new itineraries or fetching user accounts), which processes these queries and returns the results to the server for further action or response to the client.
3. Server-Auth0 Interaction: For authentication processes, the server communicates with Auth0 to verify user credentials and manage session information, ensuring secure access to the platform.
4. Server-Google APIs Interaction: When users plan trips, the server uses Google Places API for destination information and Google Maps Routes API for travel time calculations, integrating this data into the itineraries.
5. Client-Auth0 Interaction: The Client also is redirected to Auth0 for authentication purposes to allow the user to securely login through several different options, and then sends the received auth code to the server.

Design Issues

Functional Issues:

1. What information do we need to sign up for an account?
   - Option 1: Username and Password only
   - Option 2: Username, password, email address
   - Option 3: Username, password, email address, phone number

Choice: Option 2

Justification: We chose option 2, which requires the users to provide a username, password, and email address. This is the most suitable choice for our social media app designed for itineraries. The inclusion of an email address is crucial for user verification, account recovery, and communication within the platforms. It allows us to maintain the security of user accounts and keep users informed about important updates. We don't require a phone number for sign-up because we aim to provide our users with some privacy since we believe that an email address is sufficient for creating and accessing their accounts while preserving their personal information.

2. What kind of details do we allow for the itineraries?
   - Option 1: Events, trips, visit schedules
   - Option 2: Events, trips, dates, and visit schedules
   - Option 3: Events, trips, dates, visit schedules, descriptions

Choice: Option 3

Justification: We chose option 3, which allows users to include events, trips, dates, visit schedules, and descriptions in their itineraries. By providing these details, we encourage the users to create detailed travel plans within our app. Events and trips offer clear organization, while dates and visit schedules add precise timing for each activity. The inclusion of descriptions allows the users to provide context as well. This approach ensures that our app supports a wide range of travel planning needs, from simple day trips to detailed and informative vacation plans.

3. How do we build the recommendation system?
   - Option 1: Matrix diagonalization
   - Option 2: Weighting system
   - Option 3: Content-based filtering
   - Option 4: Graph-Based Model

Choice: Option 2

Justification: We have chosen option 2, the weighting system, as the foundation of our recommendation system because it offers a more practical and efficient approach. Matrix diagonalization, while powerful, typically involves complex AI techniques that can be very challenging and resource-intensive to implement. Content-based filtering tailors recommendations by aligning them closely with users' established preferences, however, this

limits diversity in recommendations. The graph-based model uses a web of connection between users and items, however, demands significant computational resources with high levels of maintenance. On the other hand, a weighting system allows us to assign importance values to various user preferences and factors. This method is faster and easier to build, making it more accessible and allowing the team to create a robust recommendation system.

4. How do we allow the users to communicate with each other?

- Option 1: Commenting System
- Option 2: Real-time DM chat, and Commenting System
- Option 3: Realtime DM

Choice: Option 2

Justification: We have decided on option 2, which includes both a real-time chatroom and a commenting system, to enable users to communicate with each other effectively within our app. This approach ensures a versatile and comprehensive communication experience. Real-time Direct Messaging allows users for one-on-one or group conversations, which enhances the direct interaction between users. Also, the commenting system allows the users to engage in discussions, share insights, and provide feedback on various content. Having both the Realtime DM chat and commenting system ensures that users have full access to the communication capabilities, encouraging the community to thrive.

5. What details should be included on the user profile page?

- Option 1: Username and profile picture
- Option 2: Username, profile picture, bio, itineraries
- Option 3: Username, profile picture, bio, travel interest, links to social media, itineraries

Choice: Option 3

Justification: Implementing all usernames, profile pictures, bios, travel interests, links to social media, and itineraries is the most effective approach to having good community engagement and enhancing the user experience. This option allows users to express their personality and travel interests through their profiles. This makes it easier to connect with other individuals with similar interests. Also, allowing users to display their created content further encourages community participation as well. Therefore, it is most beneficial to have all of these features implemented in the user profile page.

Non-Functional Issues:

1. What is the framework, language, or library for Frontend?
   - React
   - PHP
   - NextJS
   - JQuery

Choice: NextJS

Justification: Next.js is our choice for the frontend framework as it builds upon React's strengths while also enhancing both developer and user experiences. With Next.js, we benefit from server-side rendering which improves the page load times. It also contains a built-in routing system for navigation, which also enhances the overall usability of our travel platform. React does not offer out-of-box features that NextJS contains, PHP does not directly align with modern JavaScript-based applications, and JQuery lacks the comprehensive framework features compared to NextJS

2. What is the framework for Backend?
   - Java SpringBoot
   - Node Express
   - Python Flask

Choice: Node Express

Justification: Node.js with Express.js is the ideal choice for our backend framework, since it handles the JSON file seamlessly, and has functionalities that facilitate smooth communication between the frontend and backend components of our travel planning platform. Also, it enables the use of the same JavaScript classes for handling data on both the frontend and backend, promoting code reusability and reducing development time. Java SpringBoot and Python Flask do not offer the same level of seamless integration with JavaScript frontend technologies and may be more difficult to implement compared to the Node Express in a full JavaScript stack environment.

3. What is the database for BluMap?
   - Firebase Firestore
   - MongoDB
   - PostgreSQL

Choice: PostgreSQL

Justification: PostgreSQL is a secure database that allows queries to be made through SQL. Unlike NoSQL databases, PostgreSQL ensures data consistency and makes it easier to simplify data modeling and management tasks. Also, the wide range of extensions possible provides scalability and flexibility in our project. Firebase Firestone and MongoDB, despite being popular

amongst developers, may not provide the same level of query complexity and relational data handling as PostgreSQL.

4. What API is the best to use for tourist location and travel time data?
- Google Places API
- TripAdvisor API
- Yelp Fusion API

Choice: Google Places API

Justification: Google Places API stands out because of the variety of data, and integration capabilities. We can easily access detailed information about tourist attractions, accommodations, and points of interest worldwide, which helps us enhance our travel planning platform to offer extensive and accurate locations, and also enhance their travel experiences. Additionally, it provides support and documentation, which ensures smooth implementation and reliable performance for our application. On the other hand, TripAdvisor API and Yelp Fusion API may not offer the same breadth of geological data or integration flexibility for the project overall as Google Places API.

5. What Authentication system to use in BluMap?
- Firebase Auth
- Auth0
- Custom Auth

Choice: Auth0

Justification: Auth0 is chosen for its robust, flexible authentication solutions that support a wide range of login methods, including social login and multi-factor authentication, without the complexity of developing and maintaining a custom authentication system. Its extensive documentation, SDKs, and quick setup processes allow our team to focus on core application features rather than security concerns. Auth0's security standards, compliance with the latest regulations, and customizable user management capabilities offer a secure, user-friendly authentication experience, superior to the limitations of Firebase Auth and the resource-intensive nature of building custom authentication.

6. How can we host the application?
- AWS
- Vercel
- Firebase
- Supabase

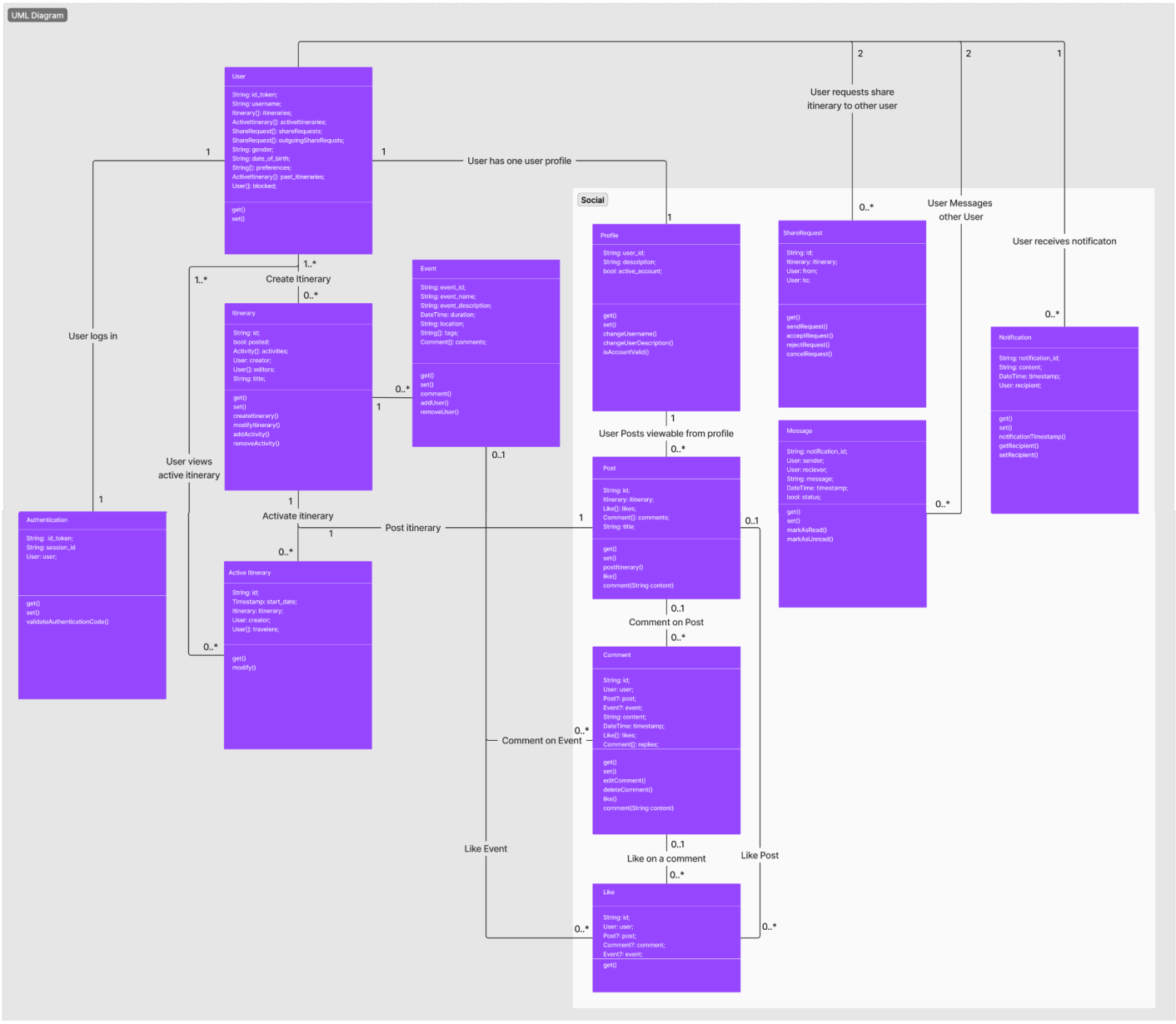
Choice: AWS

Justification: AWS provides a comprehensive and scalable cloud hosting solution that aligns with the diverse requirements of our travel planning application. Its vast array of services, including compute, storage, database, and networking options, allows for customized infrastructure setups

tailored to specific application needs, such as high availability, data redundancy, and global reach. AWS's scalability ensures that our application can grow seamlessly with user demand, offering tools for automated scaling and resource management. The flexibility to optimize costs, robust security measures, and widespread data center locations make AWS the most versatile and reliable choice for hosting our application, offering unmatched performance and reliability compared to other options.

Design Details

Class Design



Description of Classes and Interaction between Classes

The classes are designed based on the objects in our application. Each class has a list of attributes which are the characteristics owned by each object

- **User**
    - Stores unique credentials including ID token, username, and personal information such as gender and date of birth.
    - Manages a list of itineraries, active itineraries, share requests, and blocked users.
    - Offers methods to retrieve (get) and update (set) user information.
    - Serves as the central entity that interacts with other classes for itinerary creation, social activities, and authentication.
    - Connects to the Profile, Itinerary, Active Itinerary, ShareRequest, and Message classes, representing the user's presence and activity on the platform.
- **Itinerary**
    - Contains details of a travel plan, including ID, title, and a list of associated events and activities.
    - Owned by a User, who can create, modify, add activities to, or remove activities from the itinerary.
    - Provides methods to retrieve (get) and update (set) itinerary details.
    - Is shareable with other users, thereby linking to the ShareRequest class.
    - Serves as the backbone for the user's travel planning and connects to the Event class to detail each activity.
- **Event**
    - Details a specific activity within an Itinerary, with attributes like name, description, location, and duration.
    - Provides methods for users to retrieve (get), update (set), comment on, and manage (add/remove) participants in the event.
    - Stores a list of comments to facilitate user interaction and discussion on the event.
    - Linked to the Itinerary class as a component of a travel plan.
    - Connects to the Comment class, allowing users to engage with each event.
- **Message**
    - Manages user-to-user communication, with attributes such as notification ID, sender, receiver, message content, and status.
    - Includes methods to retrieve (get) and update (set) message details, as well as to mark messages as read or unread.
    - Part of the messaging system within the platform facilitates private conversations.
    - Directly related to the User class, allowing for direct communication.
    - Linked to the Notification class, indicating the receipt of a new message.

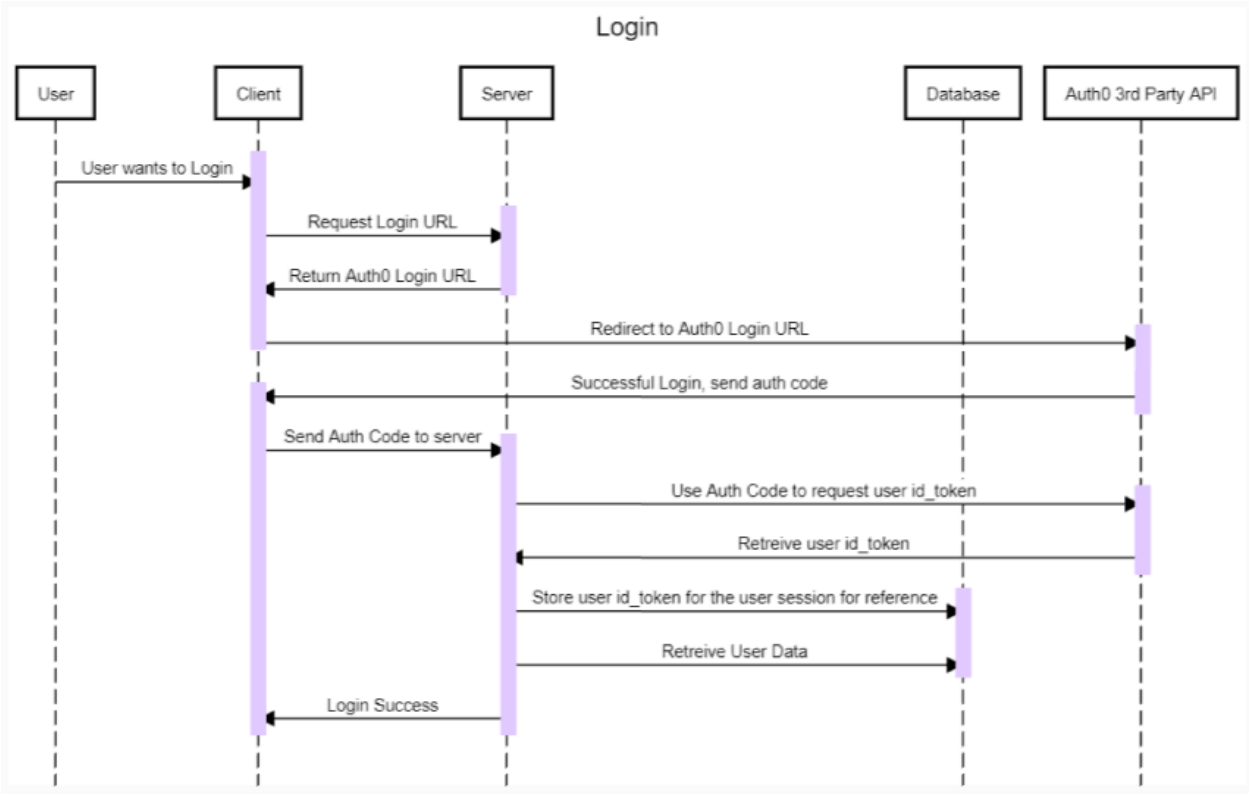
- **Authentication**
    - Handles the process of user authentication, storing ID tokens and session IDs.
    - Offers methods to retrieve (get) and update (set) authentication information, and to validate authentication codes.
    - Directly associated with the User class, ensuring that user access is secure.
- **Notification**
    - Sends alerts to users with details like notification ID, content, timestamp, and recipient.
    - Allows users to retrieve (get) and update (set) notifications, and manage recipients.
    - Connected to the User class, informing them of new activities or interactions.
    - Essential for keeping users engaged and informed within the platform.
    - Can be triggered by various user activities, such as receiving a message or a share request.
- **Profile**
    - Contains user's profile information, such as user ID, description, and account status.
    - Allows users to retrieve (get) profile details and change username or description, as well as validate account status.
    - Directly associated with the User class, representing the user's identity on the platform.
    - Essential for displaying user information to others within the platform.
    - Active status indicates whether the user's account is currently in use.
- **Post**
    - Linked to an Itinerary, a Post contains discussions, comments, and likes related to travel experiences.
    - Features methods for users to retrieve (get) and update (set) post details, and to interact with the post through likes and comments.
    - Acts as a medium for user expression and sharing of travel-related content.
    - Tied to the Comment and Like classes, enabling dynamic social interactions.
    - Central to the platform's social engagement, allowing users to contribute and collaborate.
- **Active Itinerary**
    - A specialized Itinerary in use, with additional attributes like start date and references to the traveling users.
    - Provides a method to modify the details of the active travel experience.
    - Links to the User class to represent the current and live travel activities of the user.
    - Allows for dynamic and real-time updating of travel plans.
    - Represents the active travel schedule of a user on the platform.

- **ShareRequest**
    - Symbolizes a collaborative request to join or contribute to an Itinerary, with identifying attributes and user references.
    - Contains methods to send, accept, reject, or cancel collaboration requests.
    - Facilitates sharing and cooperative planning of travel experiences.
    - Links to the User and Itinerary classes, highlighting the collaborative aspect of the platform.
    - Essential for expanding the social network and planning capabilities within the application.
- **Comment**
    - Maintains user discussions with attributes such as ID, user, post content, timestamp, likes, and replies.
    - Offers methods to retrieve (get), update (set), edit, delete, like, and reply to comments, facilitating active engagement.
    - Serves as the interactive layer of the social aspect, enabling users to engage with posts and events.
    - Linked to the Post and Event classes, encouraging community-driven discussions.
    - Influences the user experience by providing a space for feedback and interaction on shared content.
- **Like**
    - Records a user's approval or interest in a post or comment with attributes like ID, user, post, and comment.
    - Only has a method to retrieve (get) the like details, representing a simple interaction.
    - Contributes to the social validation and popularity of content within the platform.
    - Connected to the Post and Comment classes, indicating user engagement.
    - Impacts the visibility of posts and comments by serving as a metric for user interest.

Sequence Diagrams

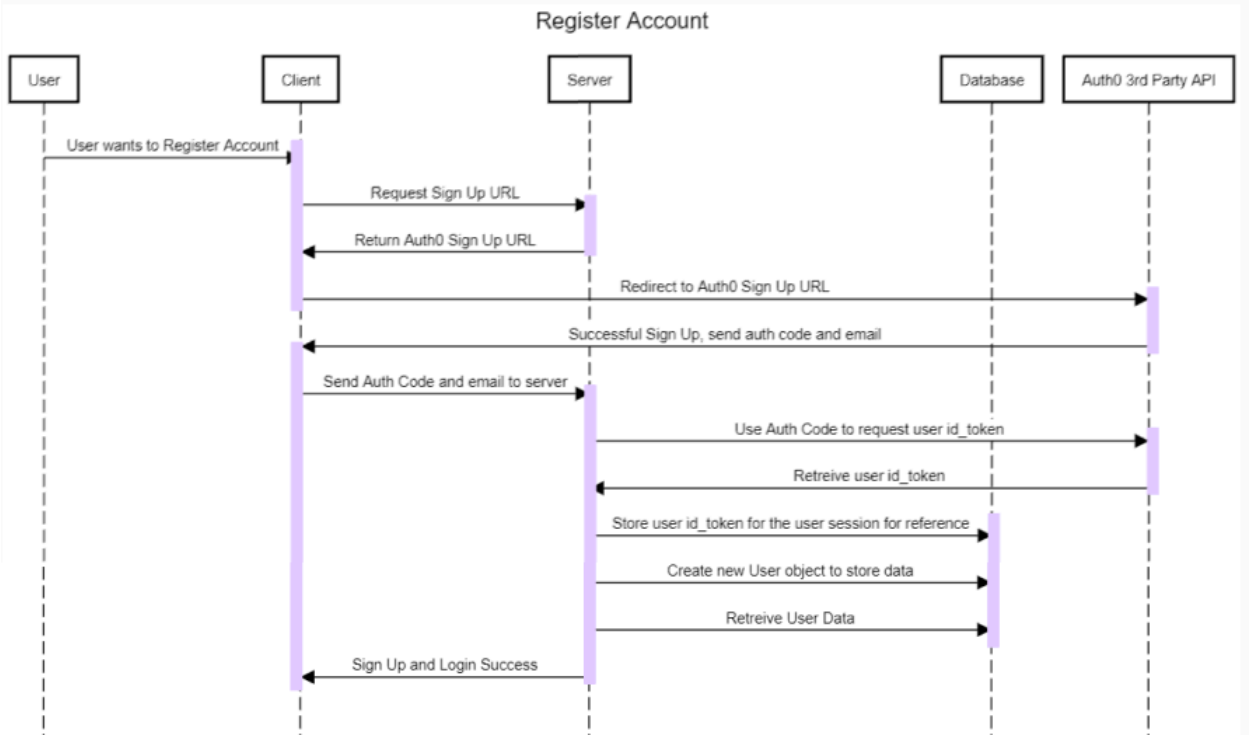
The diagrams illustrate key concepts of the user's interaction within BluMap. Actions initiated by users on the fronted interface lead to API requests, as well as triggering client requests to the server, which the server fetches or updates the data from the database. The client uses these responses to update the user interface dynamically.

1. The sequence of events when users log in.



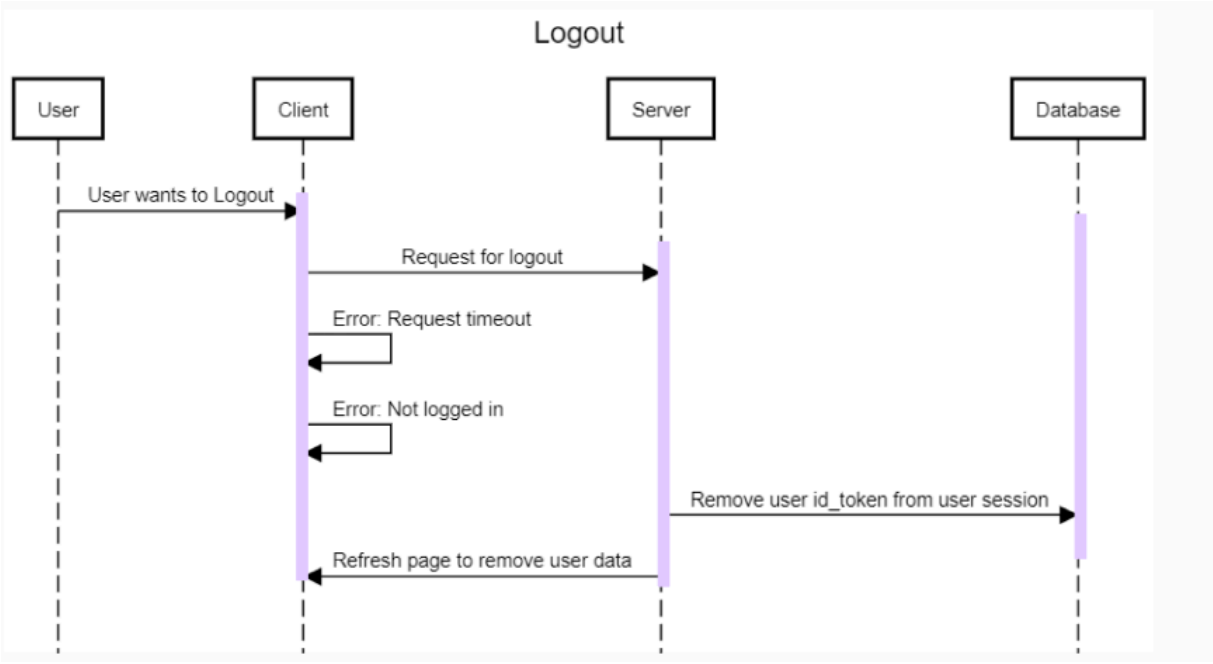
This is a diagram of the login authentication flow through the 3rd party Auth0 service. This uses OAuth2.0 to successfully authorize users to BluMap.

2. The sequence of events when the user wants to create an account.



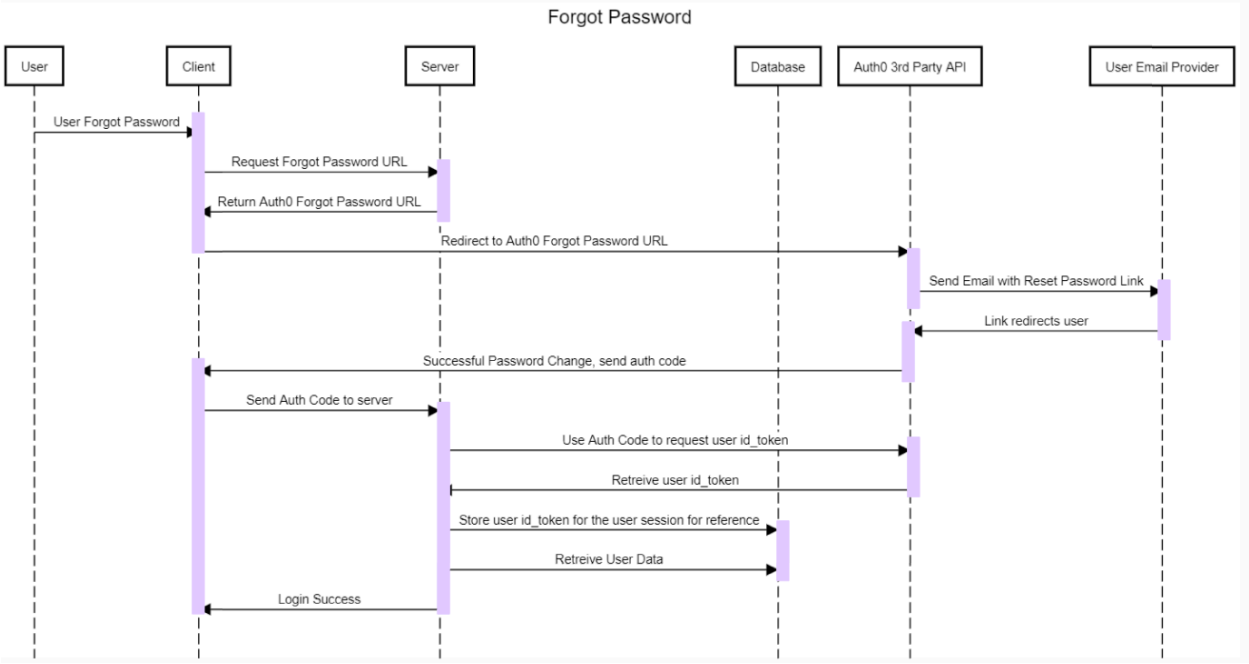
This is a diagram of the login authentication flow through the 3rd party Auth0 service. This uses OAuth2.0 to successfully help the users make an account on BluMap.

3. The sequence of events when the user wants to log out.



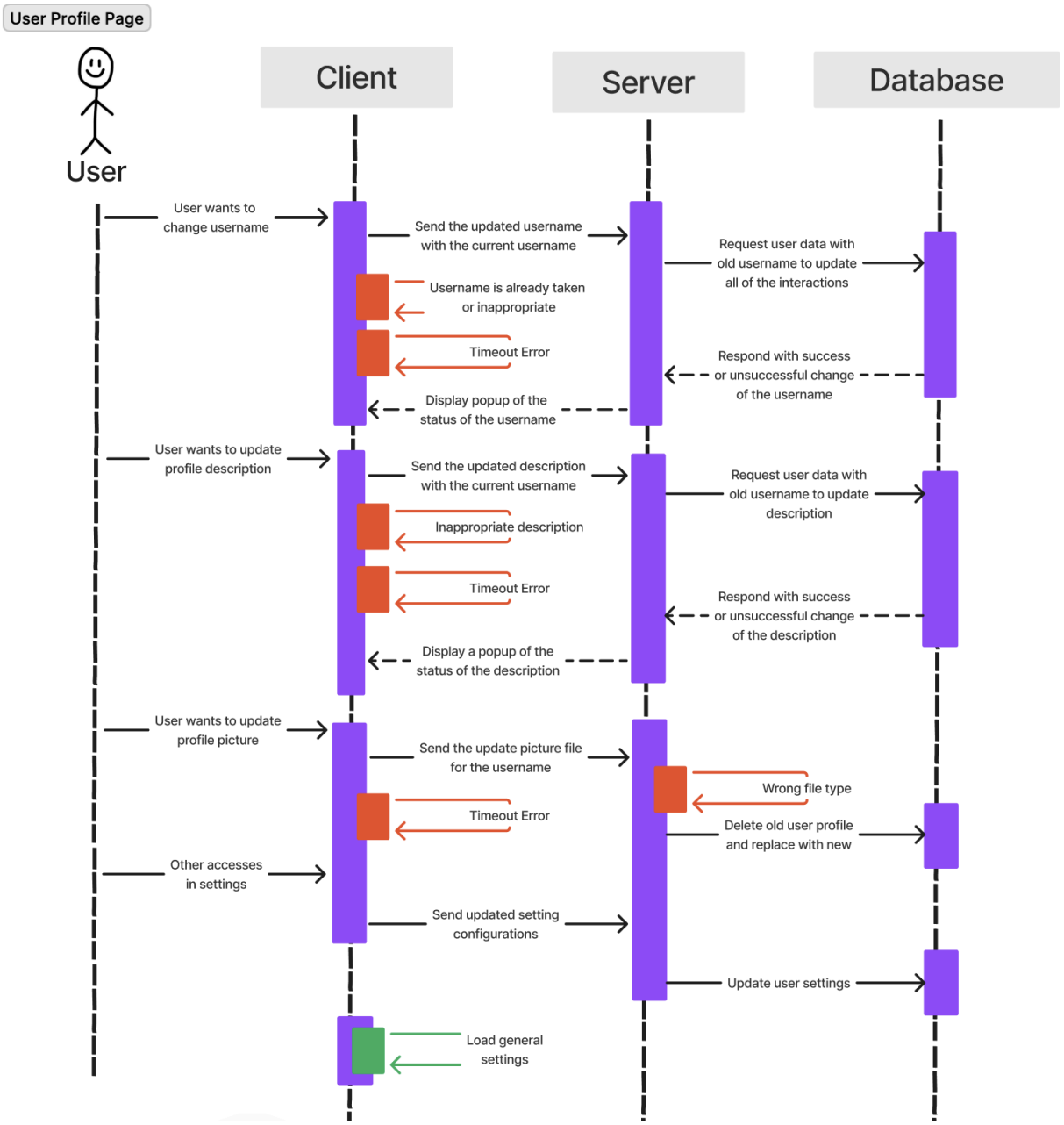
This is a diagram of the logout authentication flow. The user ID token is removed, and the webpage is reloaded to ensure that cached user information is forgotten.

4. The sequence of events when the user forgets the password.



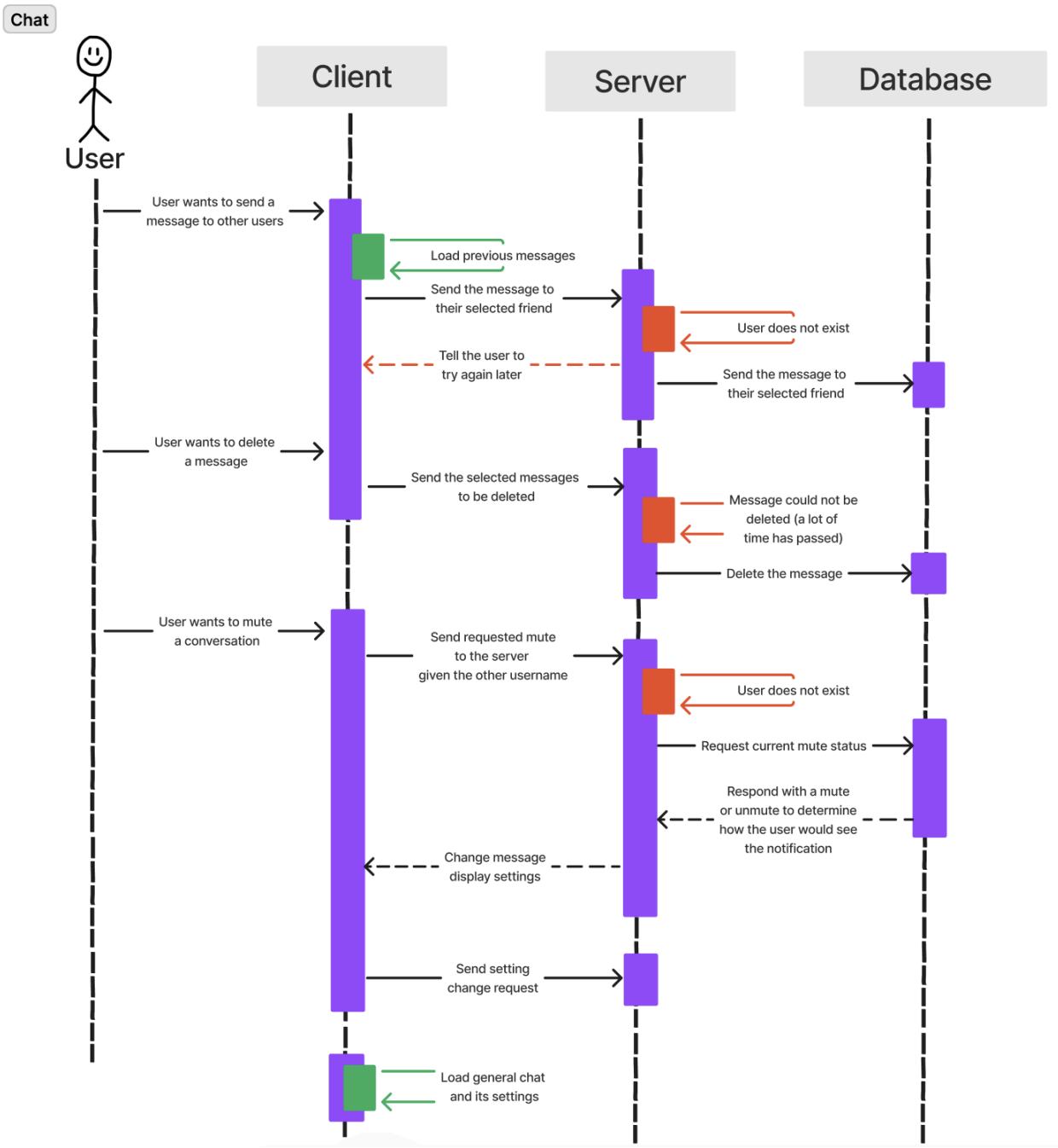
This is a diagram of the forgotten password authentication flow through the 3rd party Auth0 service. The third-party Auth0 provider handles the authentication and we handle the login after the new password is established through the OAuth2.0 authentication flow.

5. The sequence of events for the user profile page.



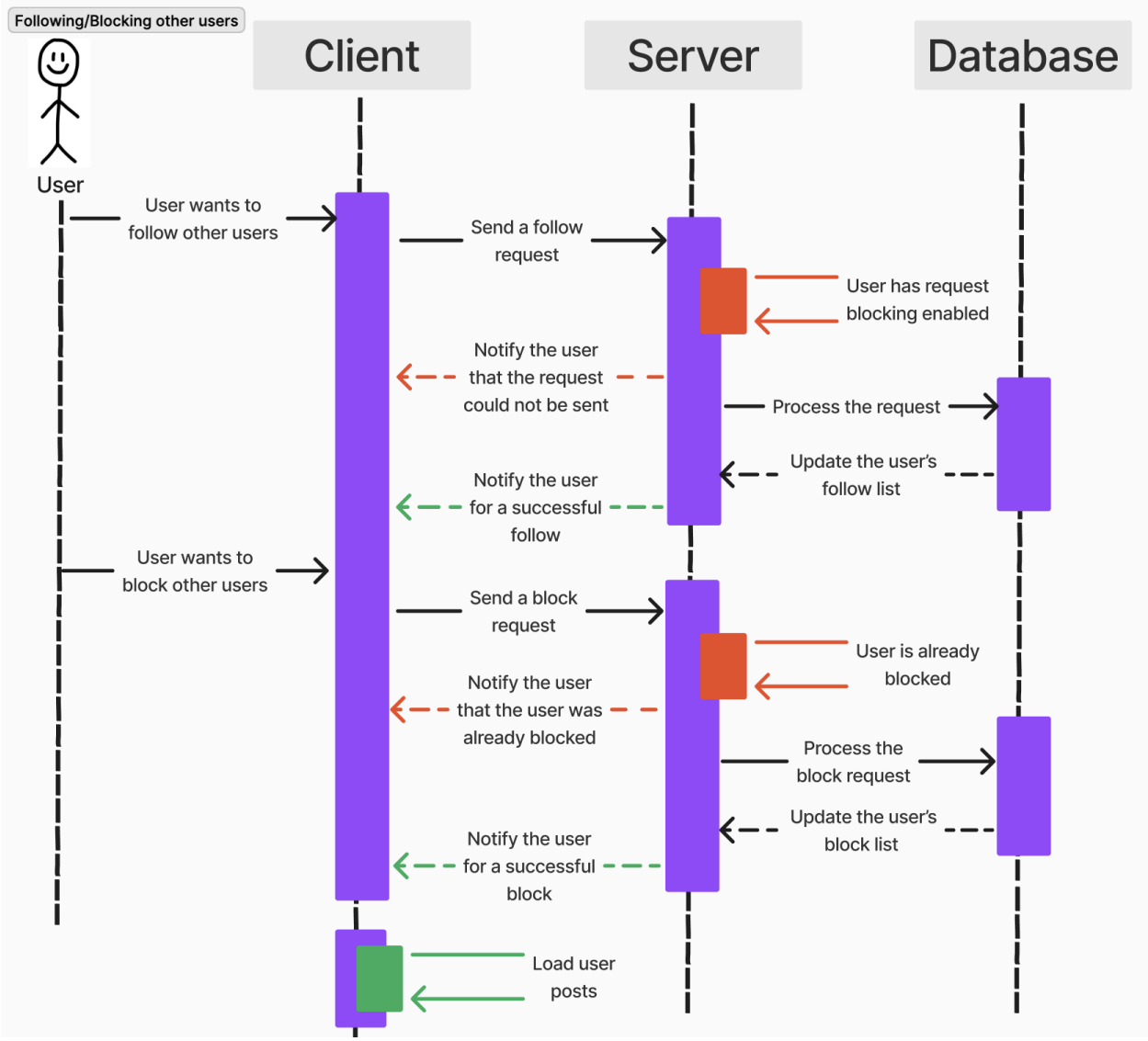
This design diagram outlines processes for updating a user's username, profile description, profile picture, and settings. It involves checking for availability, appropriate language, and correct file types, with the system responding with success or failure statuses and displaying popups to inform the user accordingly.

6. The sequence of events when the user wants to chat with others.



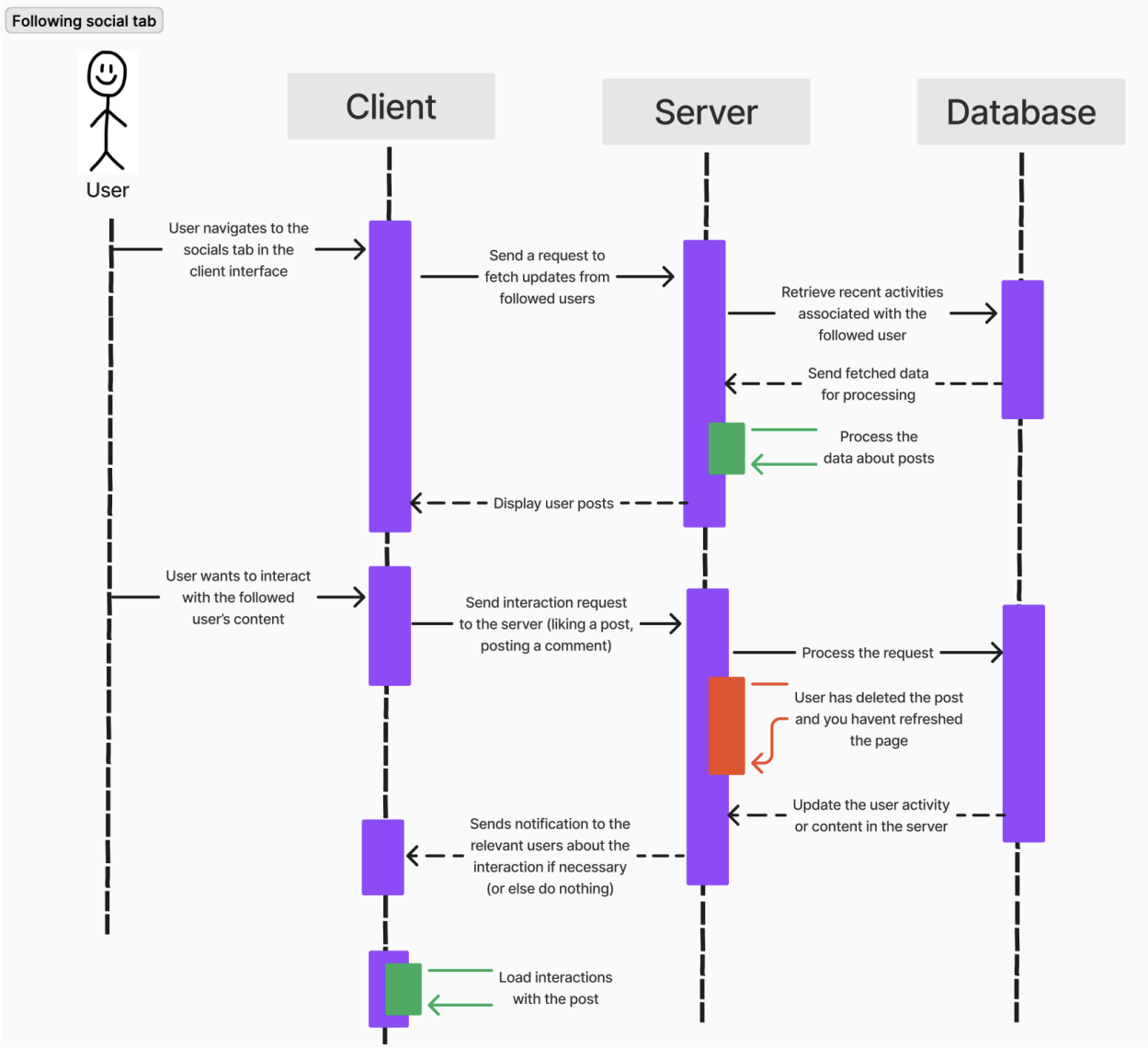
This sequence describes messaging functionalities including sending messages, deleting messages, and muting conversations. It involves loading previous messages, handling errors like non-existing users, and managing message deletion and muting requests. Users can also adjust message display settings with the option to mute or unmute conversations, providing flexibility in their messaging experience.

7. The sequence of events when the user wants to follow or block other users.



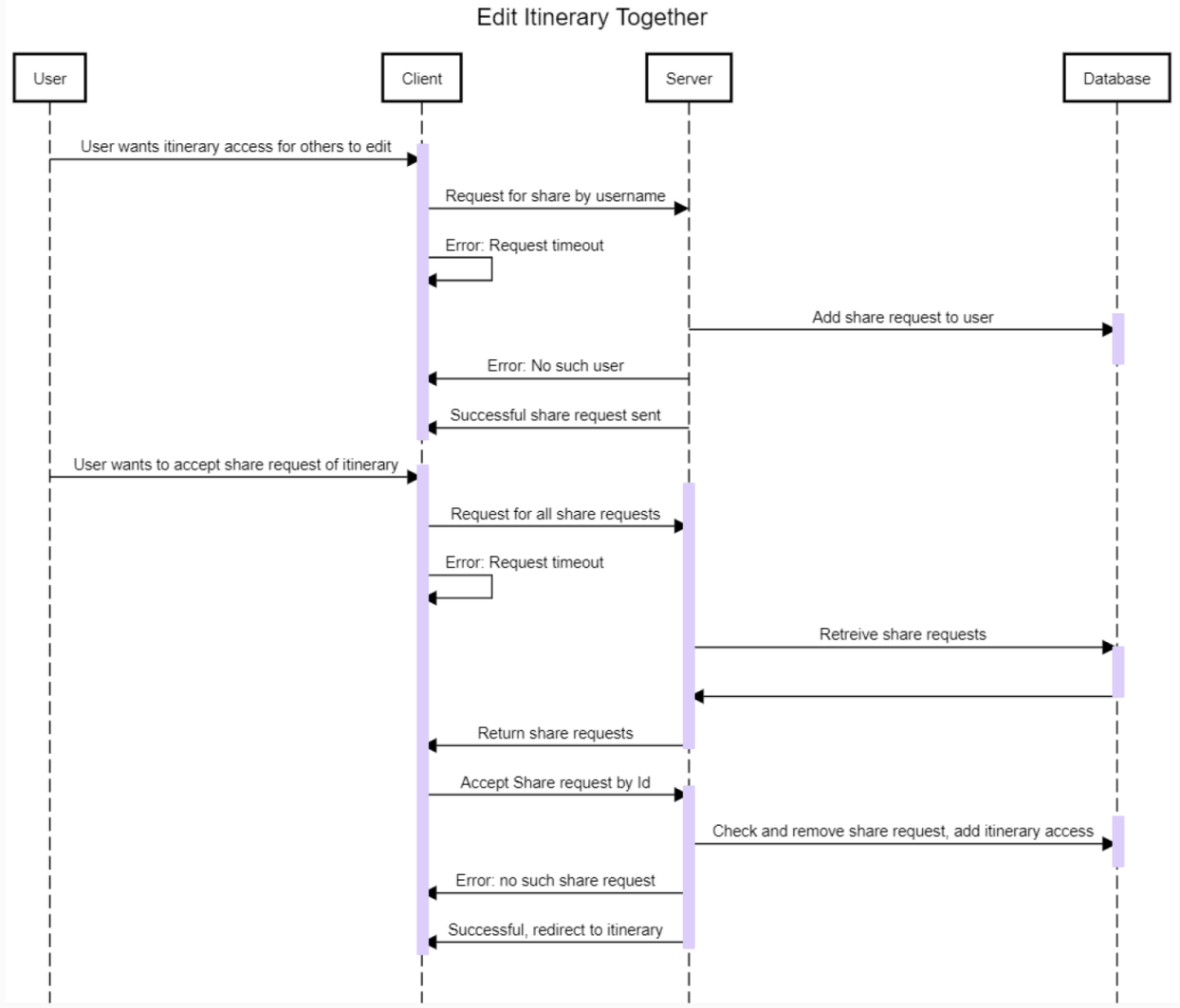
This sequence outlines the actions for following and blocking other users within the platform. Users can send follow requests, with notifications indicating successful follows and updates to their follow lists. Similarly, users can send block requests, receiving notifications upon successful blocks and updates to their block lists. The sequence ensures users are informed of the status of their requests and that their lists are updated accordingly.

8. The sequence of events of the following social tab.



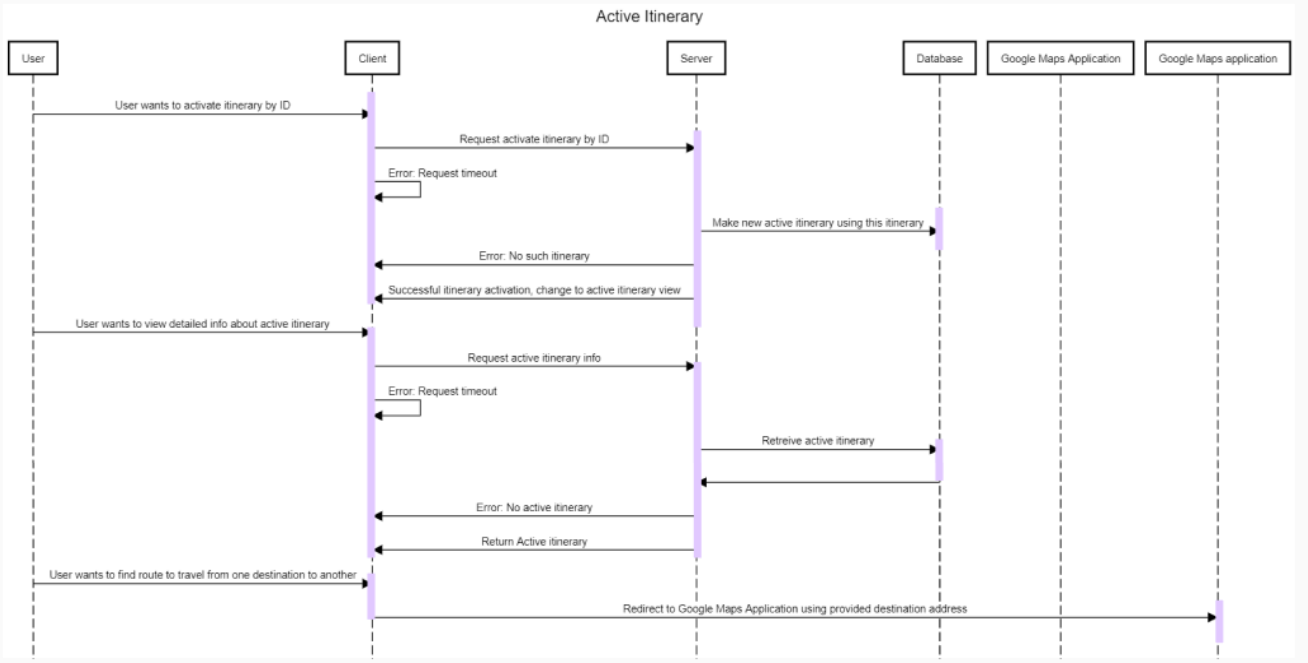
In this sequence, the user initiates interactions with content from followed users within the social tab of the client interface. The client sends requests to fetch updates and recent activities from the server, which are then processed and displayed to the user. Interactions such as liking posts or commenting trigger requests to the server, which processes them accordingly, updating user activities and content as needed. Notifications are sent to relevant users if interactions occur on deleted posts, ensuring seamless engagement with the platform's content.

9. The sequence of events when the user wants to edit itinerary access to others.



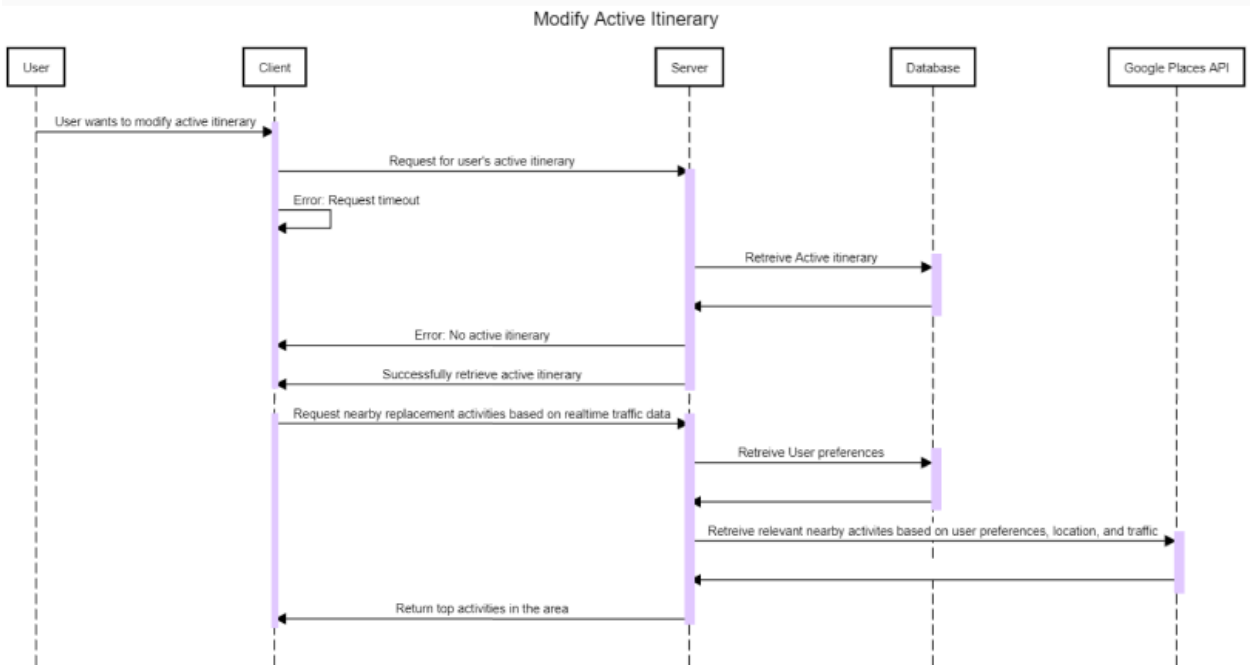
In this sequence, the user requests to share itinerary editing access with the client, which interacts with the server and updates the database if the recipient exists. Also, when accepting a share request, the client retrieves all requests from the server and the server confirms and grants access to the itinerary. Errors such as timeouts or non-existing users or requests are handled and communicated to the user through the client interface.

10. The sequence of events when the user wants to activate an itinerary for a trip.



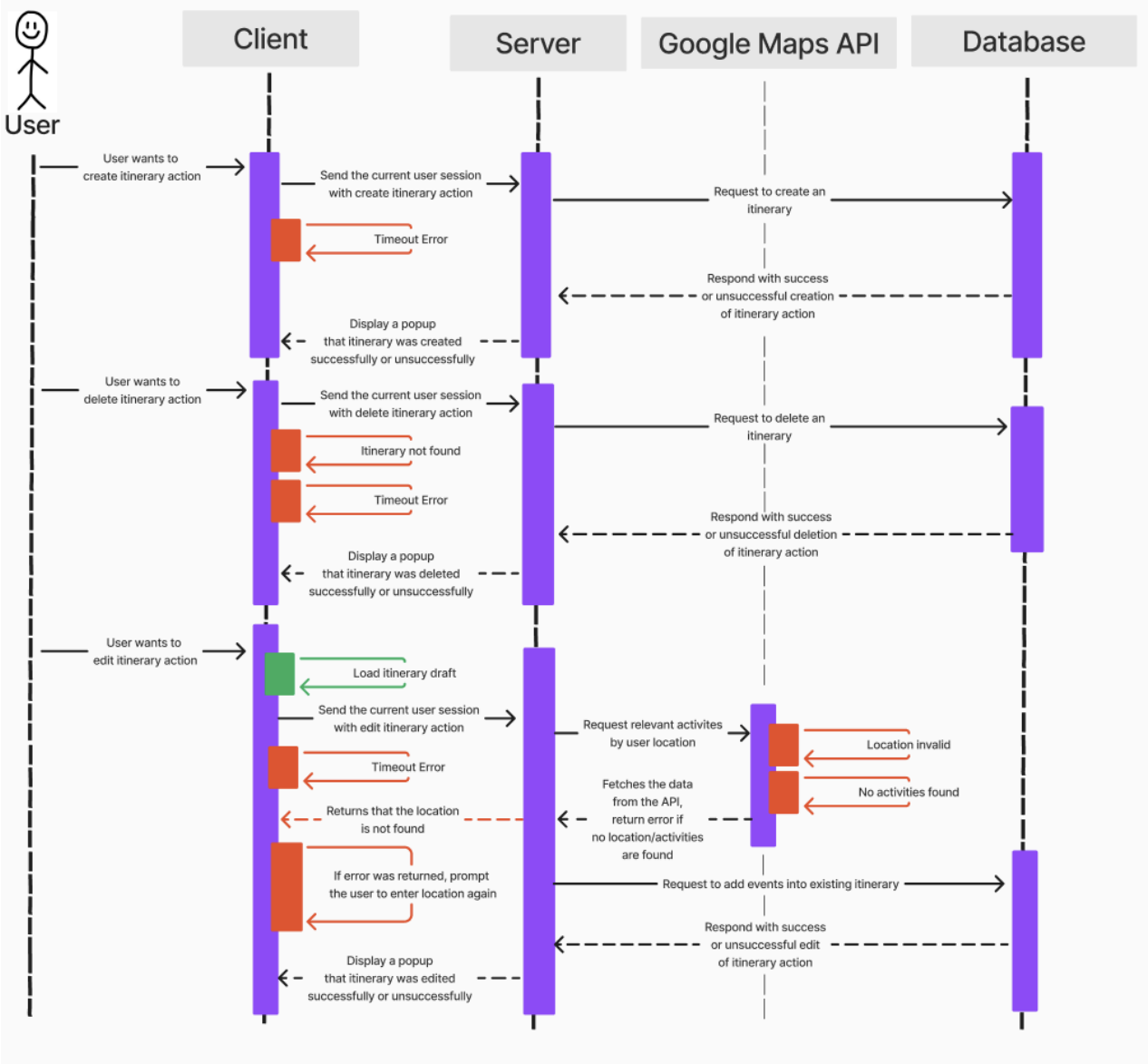
This sequence diagram illustrates the process for activating an itinerary, where the user's request to the client leads to server interactions with the database, which culminates in the activation of travel plans aided by Google Maps. In The event of an error or change, the server handles exceptions and communicates back to the user.

11. The sequence of events when the user wants to update an active itinerary.



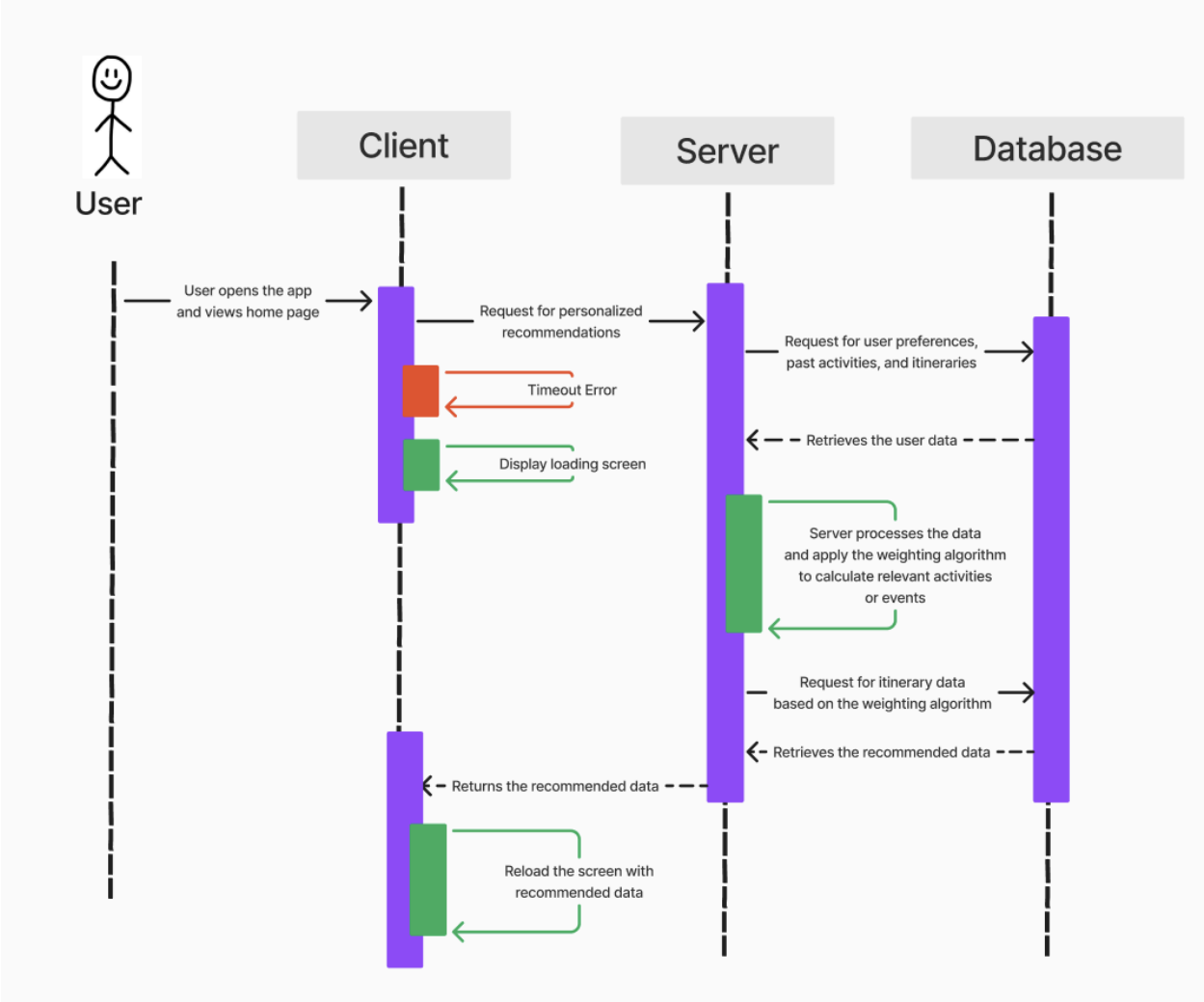
This sequence diagram explains the steps for updating an active itinerary: the user's modification request triggers the client to solicit current itinerary data from the server, which then uses Google Places API and the database to provide updated activity suggestions based on live data and user preferences.

12. The sequence of events when creating, deleting, or editing itinerary action.



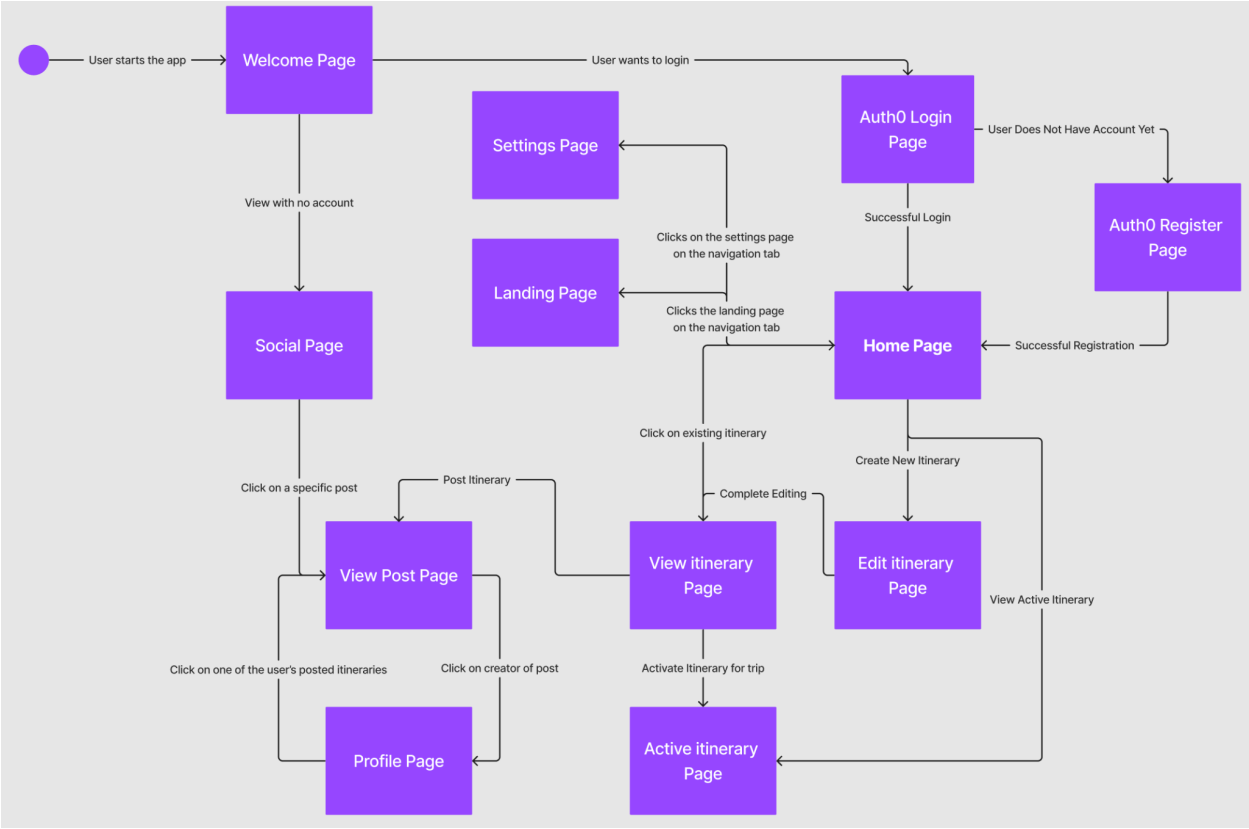
This is the sequence of events when the users want to create, delete, or edit their travel plans through the app. When the user opts to create an itinerary, the app communicates with the server, which interacts with the database to finalize the creation of the process. If the user wishes to delete an itinerary, the app sends a deletion request to the server. Editing an itinerary involves loading the current draft, making changes, and submitting those changes to the server, which may include fetching activities based on the user's location from the Google Maps API.

13. The sequence of events for the recommendation system.



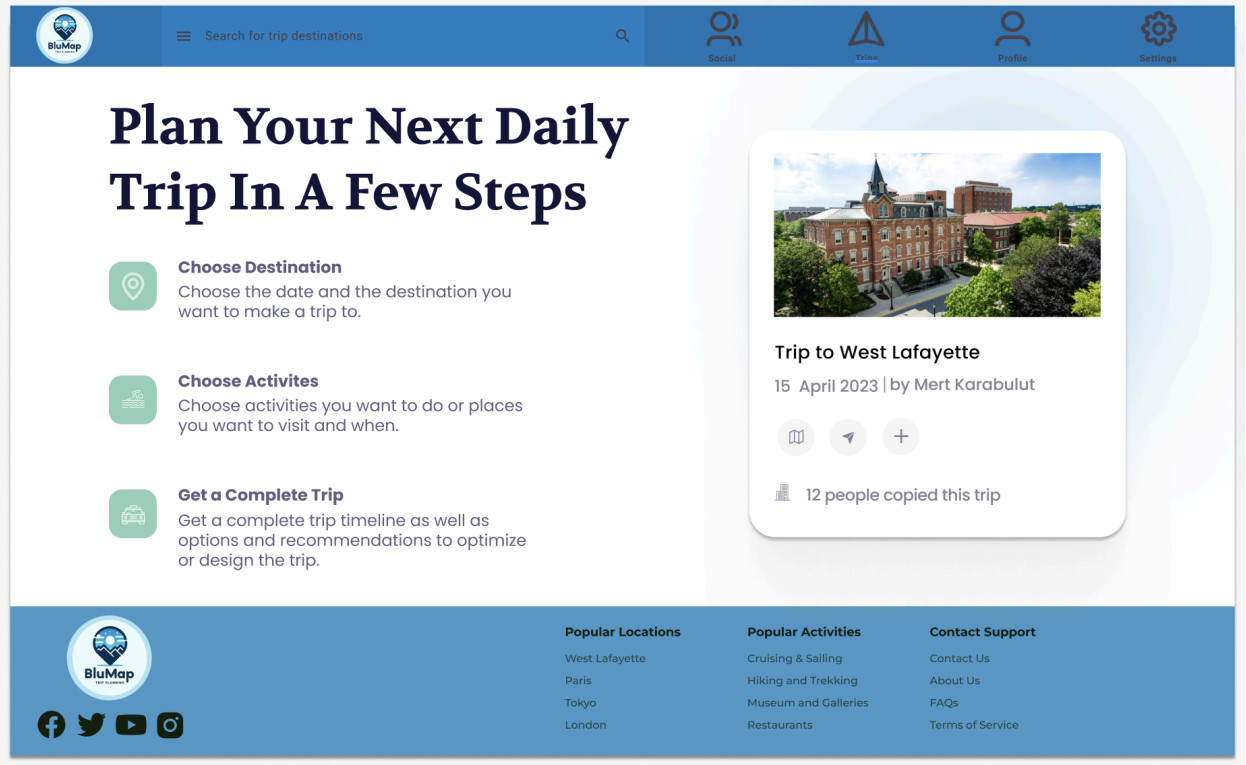
This sequence of events depicts the process for the recommendation system. When the user opens the home page, a request is made for personalized recommendations. The server fetches the user's preferences, past activities, and itineraries from the database. Using a weighting algorithm, the server processes this data to calculate relevant activities or events and sends back the recommended data. Once the recommendations are received, the app reloads the screen with this personalized information.

Navigation Flow Map

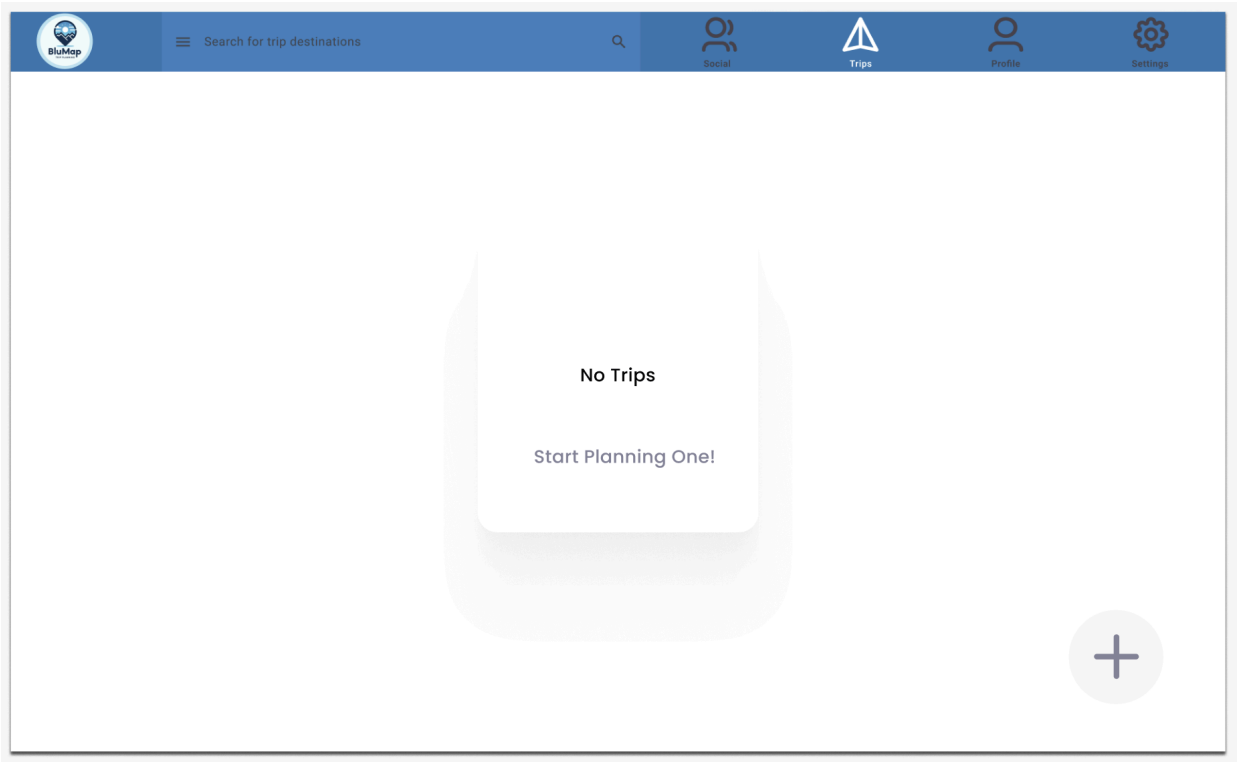


This is the application navigation flow. All pages will have the navigation bar present for access, which can be used to access other relevant pages. The application is usable as a user and a non-user, with only limited functionality available for non-users. The arrows describe how a user can move from one-page state to another.

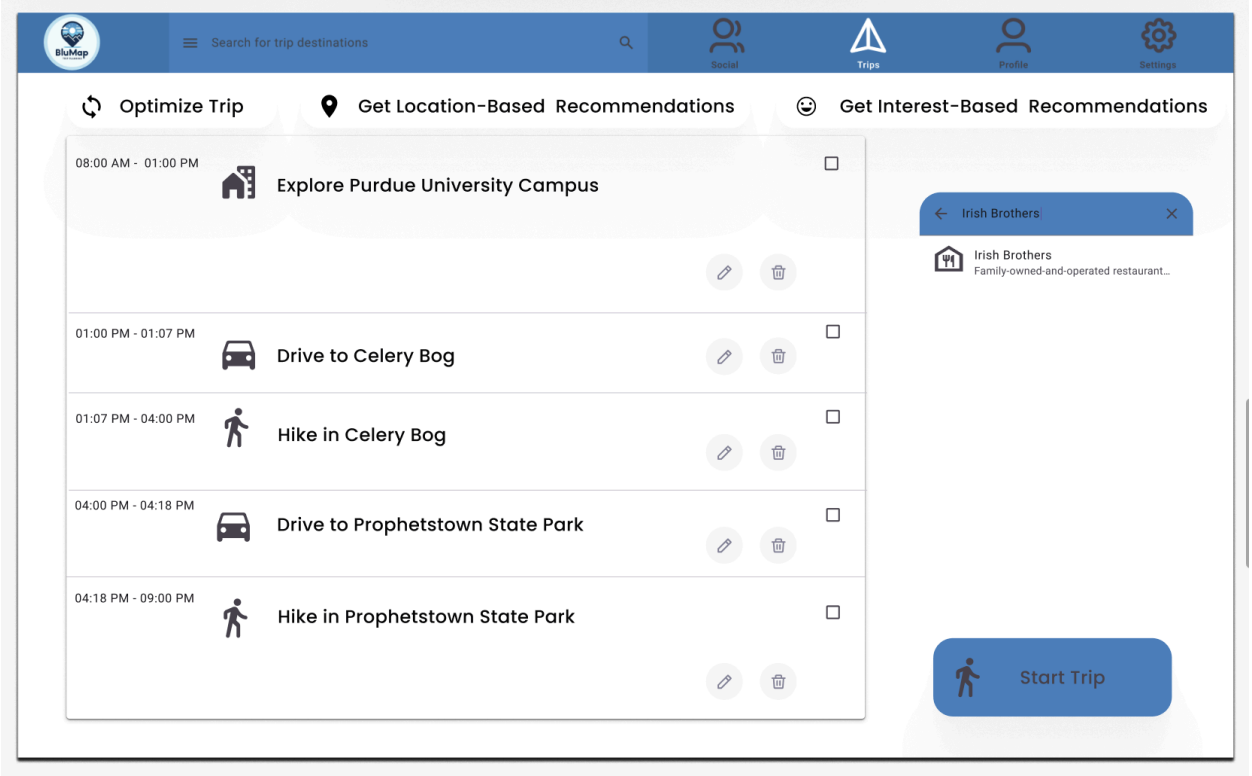
UI Mockups



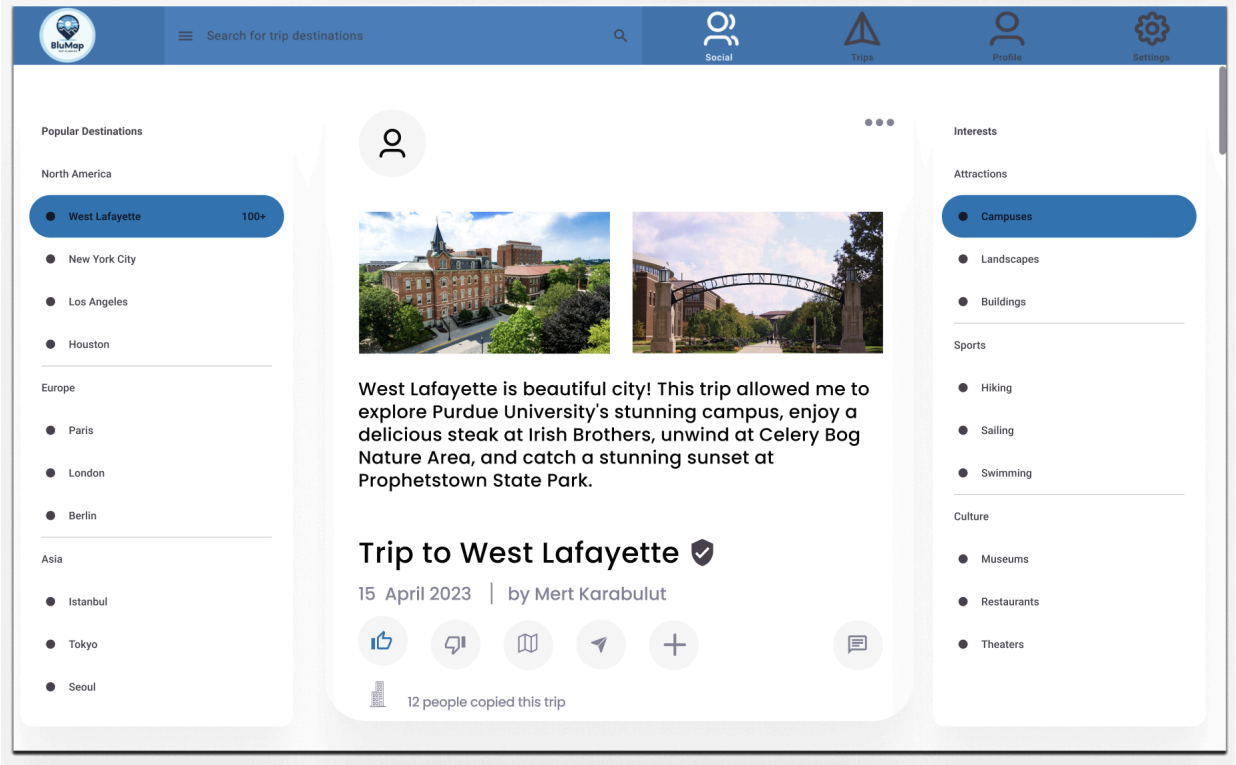
This is the landing page. This page features a navigation bar for the streamlined navigation of the website, which will be featured on every page, as well as a footer unique to this page that contains BluMap's official social media links, as well as other helpful links for trip browsing and customer support purposes. Furthermore, the search bar on the navigation bar allows users to quickly search destinations they would like to make a trip to. The body of the page features a quick summary of our application by displaying trip planning steps and a simplified shared trip. The color of the website generally utilizes shades of blue to adhere to our application name BluMap.



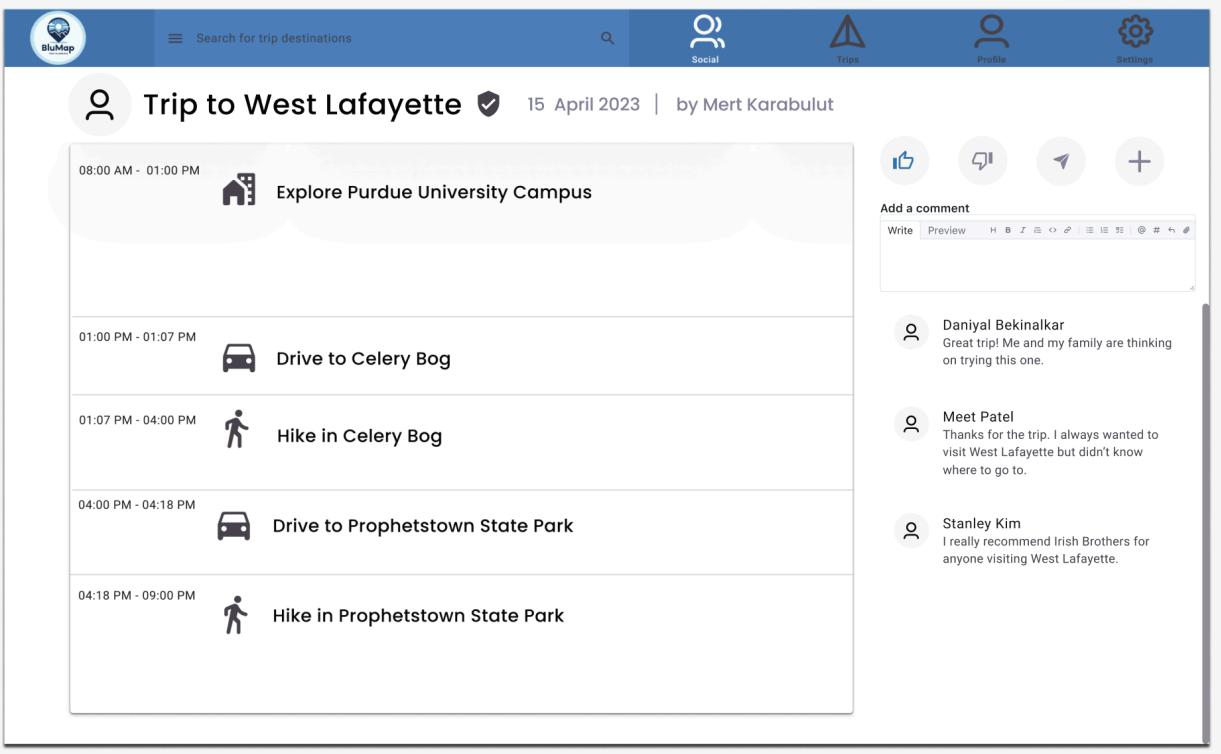
This is a user's trips page that currently has no active, past, or planned trips. If the user wants to start planning a new trip they would have to press the plus button at the right bottom of the screen.



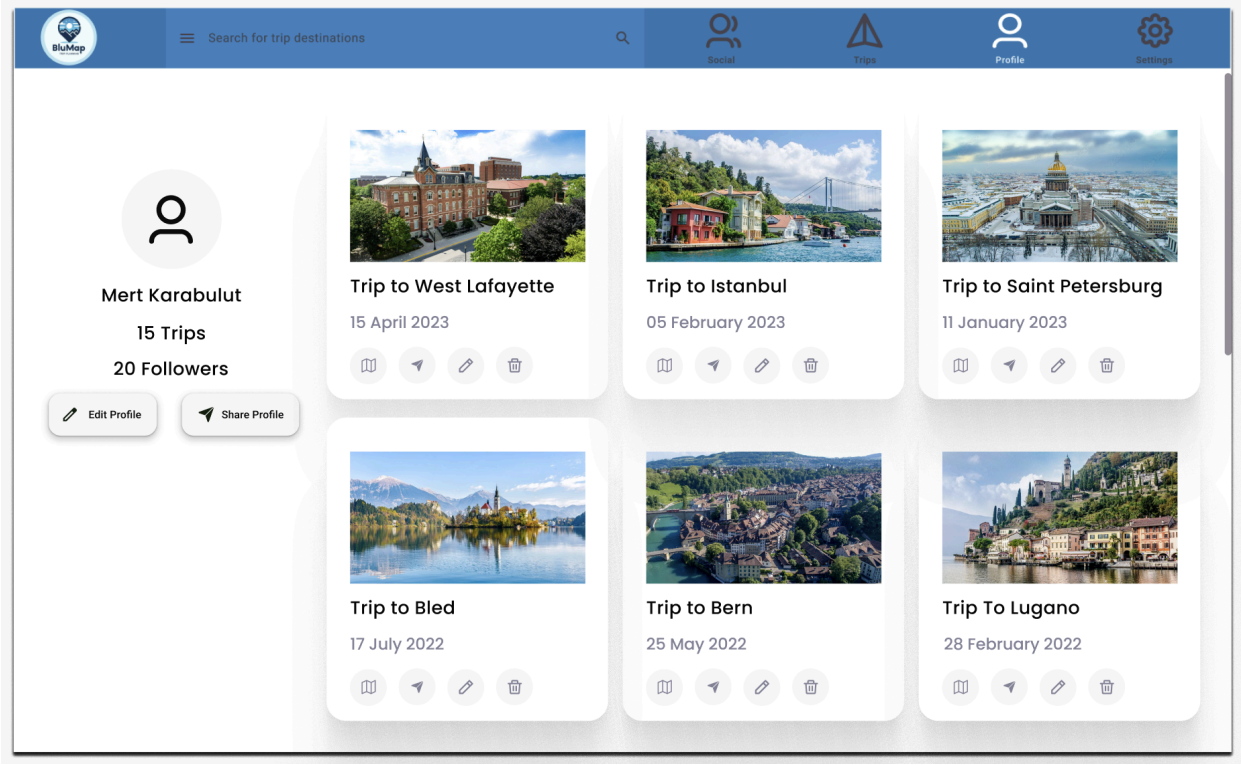
This page shows a planned itinerary on a user's trip page. The body of the page features the daily itinerary that lists the activities the user will do and between what time intervals they will do it. The unchecked box next to these activities indicates that the user hasn't yet completed these activities, and will automatically or manually be checked once the activity is complete. The pencil icon next to these activities allows the user to edit this activity whereas the trash icon allows the user to delete this activity. The driving activities are automatically added between activities after getting travel time information from Google Maps API. The rightmost search bar on the page allows users to search attractions, places, or locations to add to their trip. Also, the optimize and get recommendation buttons on top of the schedule offer the users helpful trip optimization or preference-based activity recommendation utilities. If a user wants to start the trip, all they have to do is click the Start Trip button. Clicking on each activity will also reveal more information such as descriptions.



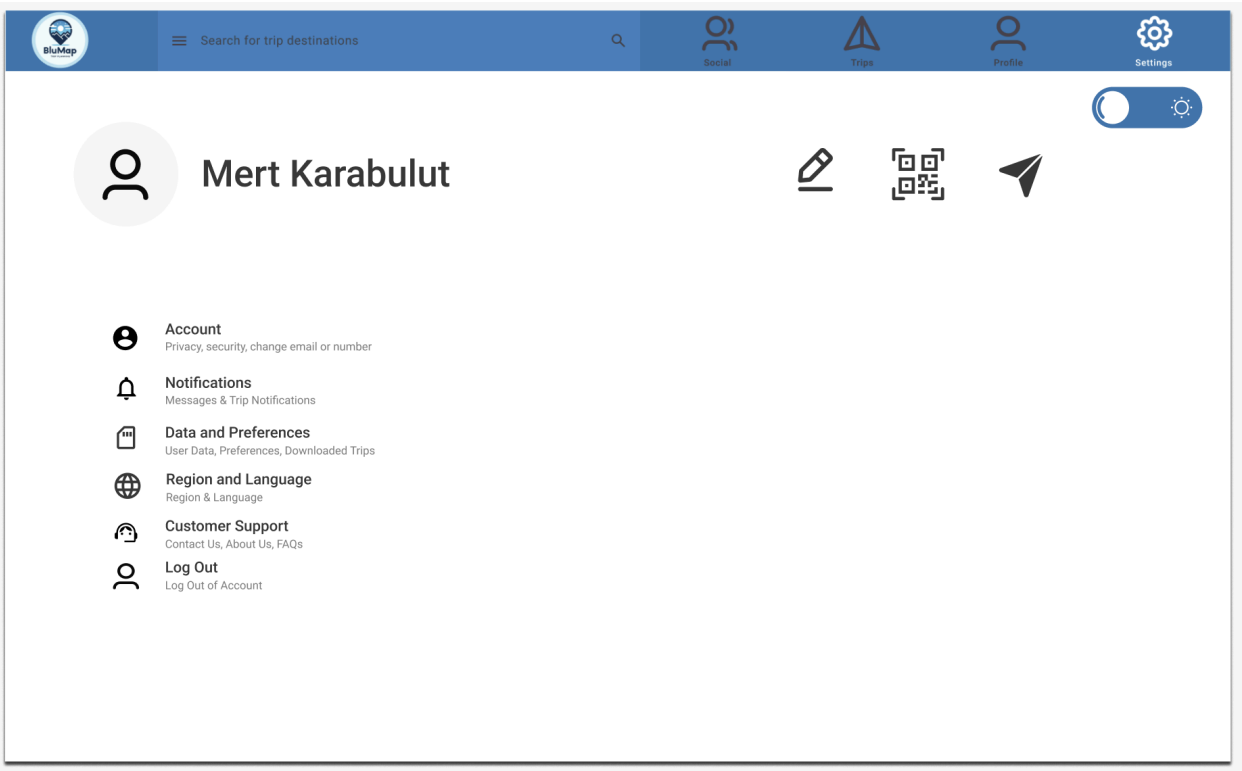
This is the application's social page. It features a shared trip by another user as well as side navigation bars that offer location or preference-based filtering for easier navigation of the social page. The shared trip features like and dislike buttons, for the community-based reputation system, as well as a map icon for viewing the trips, a share icon for sharing the trip, and a plus icon for adding the trip to your trips. The comment icon also allows users to comment on one another's trips. Furthermore, the verified icon next to the trip location indicates that the completion of this trip has been verified by our application and is a genuine trip. Lastly, this page features a scroll bar on the rightmost border to allow users to scroll through the shared trips.



This screenshot features the trip viewing function on the social page. The body of the page features the shared itinerary as well as the destination, date, and shared information. The icons on the right top of the body section allow the user to like, dislike, share, or save a trip respectively. The page also features a comment section on the right that allows users to see the comments left by other users and to write a comment while utilizing many writing tools.



This is the user's profile page featuring trips they have completed. This page displays information about the user such as trip and follower count. It also allows users to edit or share this profile with the buttons provided below the profile picture. The buttons on the completed trips allow users to view, share, edit, or delete their trips found on the profile. Viewing another user's profile page would display the same view except for the edit buttons that allows users to do adjustments to their public profile. This page will also include bio, travel interest, links to social media which was excluded now for simplicity.



This is a user's settings page. This page offers the user's profile photo and their username on the left top of the body. On the right of the username and the profile photo, in order, the edit button allows the user to quickly change their username or profile photo, the QR button allows the user to generate a QR code that is directed to their profile page, and the share button again allows users to share their profile. This page also allows multiple settings categories that allow users to have great control over their accounts and applications. The page also features a dark mode switch on the top right that allows the users to change the UI color scheme of the page between day mode and night mode. The current mode is day mode.