RandezVous

Design Document

Team 6
Vincent Zhao
Nathanael Aou
Elaine Huang
Olivia Liu
Bryan Yoo

Table of Contents

Purpose

Many social media apps today want you to engage with others online and struggle to promote in-person, meaningful interactions. Furthermore, it is often hard for young people to find friends in new environments like college/post-college life. Existing social networking services such as Instagram and TikTok want you to stay engaged on their apps and websites in order to earn money through ads while we hope to encourage people to get offline and meet up in person. Our team wants to solve this issue by providing users with an app that encourages and incentivizes friend groups to meet up spontaneously and consistently by notifying users at random times to group up at random locations. Essentially, a random rendezvous, or RandezVous.

Functional Requirements:

1. As a user, I would like to register for a RandezVous account
2. As a user, I would like to log in to my account
3. As a user I would like to log out of my account
4. As a user, I would like to change my username to any unclaimed username
5. As a user, I would like to be able to link my account to an email (sign in with Google)
6. As a user, I would like to upload my own profile picture
7. As a user, I would like to reset my password for my account
8. As a user, I would like to be able to delete my account
9. As a user, I would like to create a new group
10. As a user, I would like to name and rename my group
11. As a user, I would like to upload an icon for my group
12. As a user, I would like to search for other users
13. As a user, I would like to invite other users to my group
14. As a user, I would like to receive a notification when I am invited to a group
15. As a user, I would like to be able to report users
16. As a user, I would like to be able to make my group private or public
17. As a user, I would like to be able to be able to remove people from my group if I own the group
18. As a user, I would like to view a list of nearby public groups
19. As a user, I would like to select a public group to see its members
20. As a user, I would like to select a member to see their profile
21. As a user, I would like to join existing public groups
22. As a user, I would like to leave the group I am currently in
23. As a user, I would like to view a local map based on my current location
24. As a user, I would like to view my own live location on the map
25. As a user, I would like to view the live locations of other groupmates on the map
26. As a user, I would like to receive a notification at a random time daily telling me to go to the location of the “beacon”
27. As a user, I would like the location of the beacon to be somewhat nearby all users in my group

28. As a user, I would like to view the location of the beacon on the map (replacing its previous location)
29. As a user, I would like to view a confirmation when I have reached the beacon
30. As a user, I would like to be notified when other groupmates have reached the beacon
31. As a user, I would like to get reminded to reach the beacon if I have not yet
32. As a user, I would like to be notified when all groupmates reach the beacon
33. As a user, I would like to gain points when I arrive at the beacon depending on how quick I am and if other groupmates have reached the beacon yet
34. As a user, I would like the option to turn off/pause the spawning of beacons if I own the group
35. As a user, I would like to be able to set the frequency of the spawning of beacons (eg. once every 3 days)
36. As a user, I would like to report beacon locations that are not safe or dangerous
37. As a user, I would like to view my own points as well as my group mates' points on a leaderboard
38. As a user, I would like to gain group points for every full group meet up RandezVous we achieve
39. As a user, I would like to see how my group compares to other groups of similar size
40. As a user, I would like to see available achievements I have yet to unlock
41. As a user, I would like to be able to unlock new achievements
42. As a user, I would like to view all my unlocked achievements in my profile
43. As a user, I would like to turn off notifications.
44. As a user, I would like to send friend requests to other users
45. As a user, I would like to see all pending friend requests from other users
46. As a user, I would like to accept or decline friend requests from other users
47. As a user, I would like to be able to block other users (from friend requests and group invites)
48. As a user, I would like a faq/instructions page that explain how the app works
49. As a user, I would like a contact us page to report any bugs
50. As a user, I would like a privacy policy page to learn how the app tracks my location

If time allows:

51. As a user, I would like to view the total points of other groups around me on a leaderboard
52. As a user, I would like to view a “shop” where I can spend my points on cosmetic features for the beacon such as replacing it with a variety of icons or renaming the beacon
53. As a user, I would like to be able to chat within a group
54. As a user, I would like to be able to collect streaks of RandezVous where I earn more points for consecutive days.
55. As a user, I would like to be able to be in multiple groups at the same time
56. As a user, I would like the option to set beacon spawn locations to only “interesting locations” like parks and coffee shops instead of completely random locations

Non-Functional Requirements:

Architecture and Performance
We plan to build a separate frontend and backend for this application. The frontend will be a Flutter application which can be used on both iOS and Android mobile devices. The frontend will allow us to interact with the user as well as pull location information from the user's device. The frontend will interact with the backend to do more computationally heavy operations as well as interact with the database. Interaction between backend and frontend will be done through HTTP requests.

The backend will be an Express.js application that will interact with and will be a RESTful API that the frontend will be able to make requests to. Express.js is a framework for creating RESTful APIs in JavaScript which allows us to take advantage of our existing knowledge of JavaScript combined with the performant nature of Express.js. API requests should have a maximum response time of 500 milliseconds under normal usage conditions. We will use Firebase/Supabase for our database, which has web sockets specifically designed for real-time synching, allowing us to push updates automatically without polling or manual refreshes. This can help us lower the time it takes for data to update in the app down to under 5 seconds. We also plan to use mobile devices' built in location tracking features and look up users' last updated location instead of constantly tracking their location. This way, the app should not consume more than 10-15% of battery power per hour under normal usage.

Security
Security is very important for RandezVous as we are collecting user location information which is sensitive and should be kept secure. This is why we will implement security rules to prevent users from accessing the location data of anyone else unless they are part of the same group. Furthermore, since we are using services like the Google Maps API, it is important that our application is not spammed with repeated requests that could greatly increase our billing, so we will add protections against unauthenticated requests on the backend. We also want to use Firebase or Supabase for authentication to make sure that signing into accounts is secure.

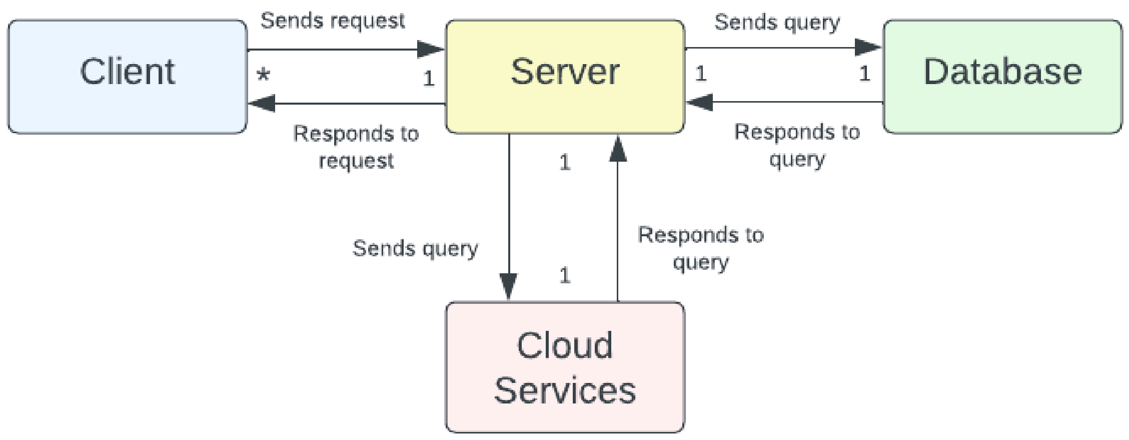
Usability
RandezVous is a social networking platform, so the UI should be intuitive for the average user to navigate through. Furthermore, since the concept of the app isn't the most straightforward, we will create an instruction UI to explain certain screens when opened by a user for the first time. Furthermore, we want the game to be accessible to as many users as possible, so we aim to make it accessible for various different phone screen sizes.

Hosting/Deployment
The frontend and backend will be deployed separately. As it is a mobile application, we want to first be able to install the application through development tools (TestFlight and Android SDK) and then, if time permits, major app stores like the Google Play Store and Apple App Store. We plan to initially test the backend on local machines and then later on deploy the backend to Google Cloud Provider (GCP) since Firebase is built on GCP.

Design Outline

High Level Overview

Usage of this app requires multiple clients connected to the server at the same time. The clients will send requests to the server when they need user data, primarily information about their own profile as well as other users. The server will request the appropriate data from the database and return it to the clients. We also will use cloud services such as user authentication which will be requested from the server.



High Level UML Diagram

System Components

1. Client
    a. Provides the user interface and runs on mobile devices
    b. Accommodates both iOS and Android operating systems using Flutter
    c. Communicates with the **server** using a combination of RESTful API requests to the Express server and real-time updates via Firebase Firestore
    d. Displays information such as user profiles, group details, live locations, and beacon status.
    e. Provides an interactive and dynamic user experience, enabling users to track locations, join groups, and receive notifications.
2. Server
    a. Acts as the core infrastructure, handling user authentication, data processing, and API requests
    b. Built with Express.js, exposing RESTful API endpoints for user management, group interactions, and beacon events

   c. Validates **client** requests and sends queries to the **database** to add, update or get data
   d. Sends queries to **cloud services** for features such as user authentication
   e. Manages complex logic, such as notifications, leaderboard calculations, and beacon spawn logic.
   f. Implements security measures for authentication and data access control, leveraging Firebase Authentication for user sign-ins

3. Database
   a. Firebase Firestore acts as the primary database, storing user profiles, group data, and achievements in JSON format
   b. Responds to **server** queries and sends the requested data
   c. Retains the location data of users and the status of beacons; maintains a snapshot of the most recent location and beacon status updates for future synchronization
4. Cloud Services
   a. Responds to queries from **server**
   b. Uses Firebase Authentication for secure storage of user credentials
   c. Serverless architecture with Firebase Cloud Functions to handle event-driven tasks
   d. Integrates Firebase Cloud Messaging (FCM) for push notifications

Design Issues

Functional Issues

1. What of the following information will we request from each user when creating a new account?
    a. First and last name
    b. Username
    c. Password
    d. Email address
    e. Phone number
    f. Age

    Choice: A, B, C, D
    Justification: The first and last name is required so we can display it on their profile. This should not be a privacy issue because we allow users to set their account to private so strangers cannot view their profile. The username is required so we can identify users by their unique username (no two users should share a username, allowing users to easily search for other users). A password is required to ensure secure login, especially since this is a location sharing app. An email address is required so that users cannot create an excessive amount of alternate accounts. It can also be used for 2FA and password resetting features if we choose to implement them in the future. We can also email information regarding app updates if needed. We decided that collecting users' phone numbers is unnecessary since we can already contact them through email. Furthermore, using email for 2FA instead of SMS is a cheaper alternative. Finally, we chose to not require users' age because it is not necessary for our app's purposes.

2. How often should we send out the notification that the beacon has been placed?
    a. Once a day at a user set time
    b. Once a day at a random time
    c. Once randomly every time interval where the user decides the time interval

    Choice: C
    Justification: Since the point of this app is to encourage spontaneous meetups, we decided that the beacon should be placed at both random places and random times, eliminating option A. We originally thought of having the beacon appear once a day, but we realized that some friend groups may be busy on certain days and thought it would be better to give users the option to choose the frequency of beacon placement. However, we still want users to frequently meet up, so we plan to reward users for having a meetup streak. In addition, we will set a minimum value for the time interval so users cannot set the interval to a very small time period to "farm" points.

3. How should we calculate where the beacon is placed?
    a. Random location in proximity to any user in the group
    b. Random location in proximity to the center/average location of all users in the group
    c. Random location in proximity to rotating users in the group

    Choice: A
    Justification: We chose to place the beacon randomly near any of the users in the group to make the range of possible locations more broad. In addition, it would be difficult to calculate the center location of all users every time the app sent out a notification. We also chose to not have the location be near a designated user, where the designated user rotates between all group members. This is because it could be difficult to keep track of the designated user if beacons are ignored across multiple days and it would provide them an unfair advantage to reaching the beacon first.

4. How should points be rewarded when users reach the beacon?
    a. Only the first user gains points
    b. All users who reach the beacon gains equal amount of points
    c. All users who reach the beacon gain points but according to the order they arrive

    Choice: C
    Justification: If only the first user gains points, there is no incentive for other users to join the first user which defeats the purpose of our app. However, if all users gain the same amount of points no matter what order they arrive, there is no incentive to get to the beacon quickly. Thus, we decided to reward the first player with the most points and reward later players with less and less points.

5. How often do we obtain users' location?
    a. Continuously, even when user is offline
    b. Periodically, when user is online

    Choice: B
    Justification: Constantly tracking all users' location even when they are offline is computationally intensive and costly. Instead, we will only obtain and update a user's location in the app's map when they go online. To check when the user arrives at the beacon, they can press a "I'm here" button which will then confirm if their location matches the beacon's location. Also, it is more interesting if users can only see other users' last updated location so they don't know how close others actually are to the beacon.

Non-Functional Issues

1. What frontend framework should we use?
    a. React Native (JavaScript)
    b. Flutter (Dart)
    c. SwiftUI (Swift)
    d. Jetpack Compose (Kotlin)

    Choice: B
    Justification: We chose Flutter (Dart) as our frontend framework because it enables cross-platform development, allowing us to build a single codebase that runs efficiently on both iOS and Android. Its use of the Skia graphics engine ensures smooth animations and high performance, making the app feel responsive and fast. In addition, since Flutter and Firebase were both developed by Google, it may be easier to integrate together.

2. What backend framework should we use?
    a. Express/Node (JavaScript)
    b. Spring Boot (Java)
    c. Flask (Python)
    d. Django (Python)

    Choice: A
    Justification: We chose Express.js because it is a lightweight, flexible backend framework for Node.js, ideal for building fast RESTful APIs. It integrates well with databases like Firebase and supports real-time interactions, making it a strong choice for our app. Express.js and Node.js are also very popular frameworks that many companies use, so it would be beneficial to be more familiar with them.

3. What database should we use to store app data?
    a. MongoDB
    b. Firebase Firestore
    c. DynamoDB

    Choice: B
    Justification: We chose to use Firebase Firestore because it provides real-time synchronization, making it ideal for location tracking and leaderboard updates. Its NoSQL document-based structure allows flexible data storage, adapting well to user interactions. Additionally, seamless integration with Firebase services simplifies authentication and notifications.

4. What should we use to track and display user locations?
    a. Flutter Geolocator Plugin
    b. Flutter Location Plugin

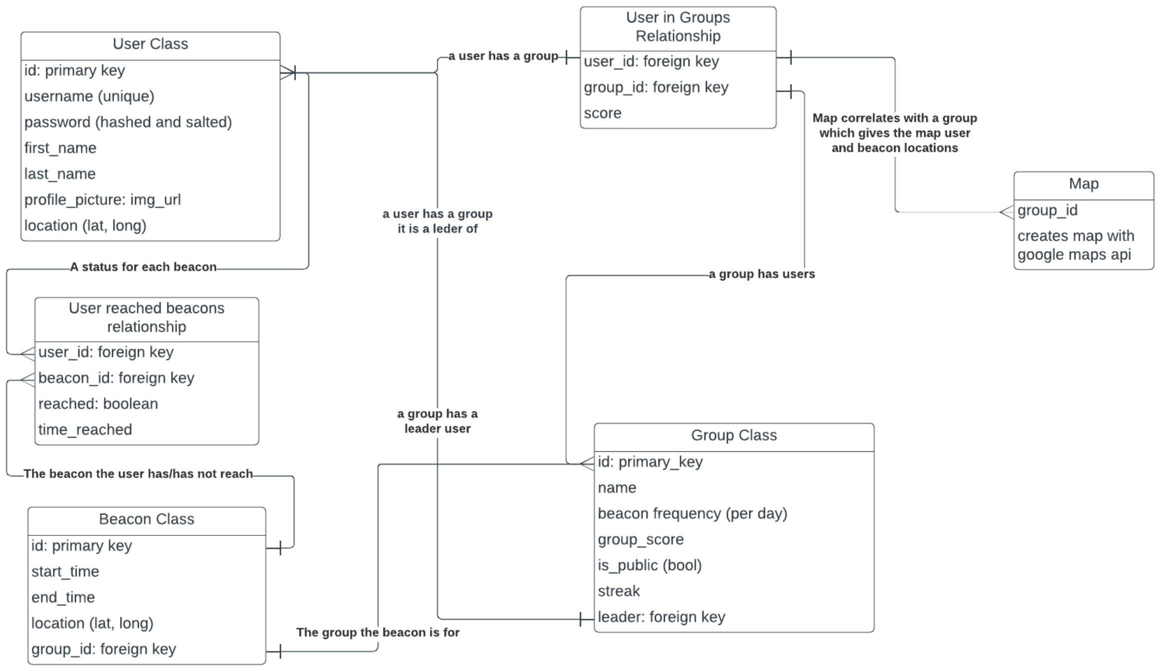
c. Google Maps SDK
d. Native Platform APIs (Apple Core Location for iOS, Android Location Services for Android)

Choice: A and C

Justification: We plan to use a combination of the Flutter Geolocator Plugin and Google Maps SDK. We chose to use a Flutter plugin because it is already built into Flutter, and we chose the Geolocator Plugin because it has more downloads and documentation compared to the Location Plugin. Google Maps SDK will also be useful to display user and beacon locations on a map with an intuitive, premade design.

Design Details

Class Level Design



Class Descriptions

- User Class
  - A User represents an individual who creates an account in the system.
  - Each user has a unique ID (primary key).
  - Users have a username, password (hashed and salted), first name, last name, and profile picture (stored as an image URL).
  - Users have a location represented by latitude and longitude.
  - Users can join multiple groups and may also lead a group.
  - A User has groups that they have joined and their scores within those groups
  - A User has beacons that they need to reach/have reached

- Group
  - Group object created when a user creates a new group
  - Each group has a unique ID (primary key).
  - The group name is set by the creator.
  - Groups have multiple users who have individual scores for the group
  - A leader (foreign key referencing a User) manages the group.
  - Groups have a beacon frequency (how often beacon events occur per day).
  - Each group maintains a group score and a streak for continuous participation.

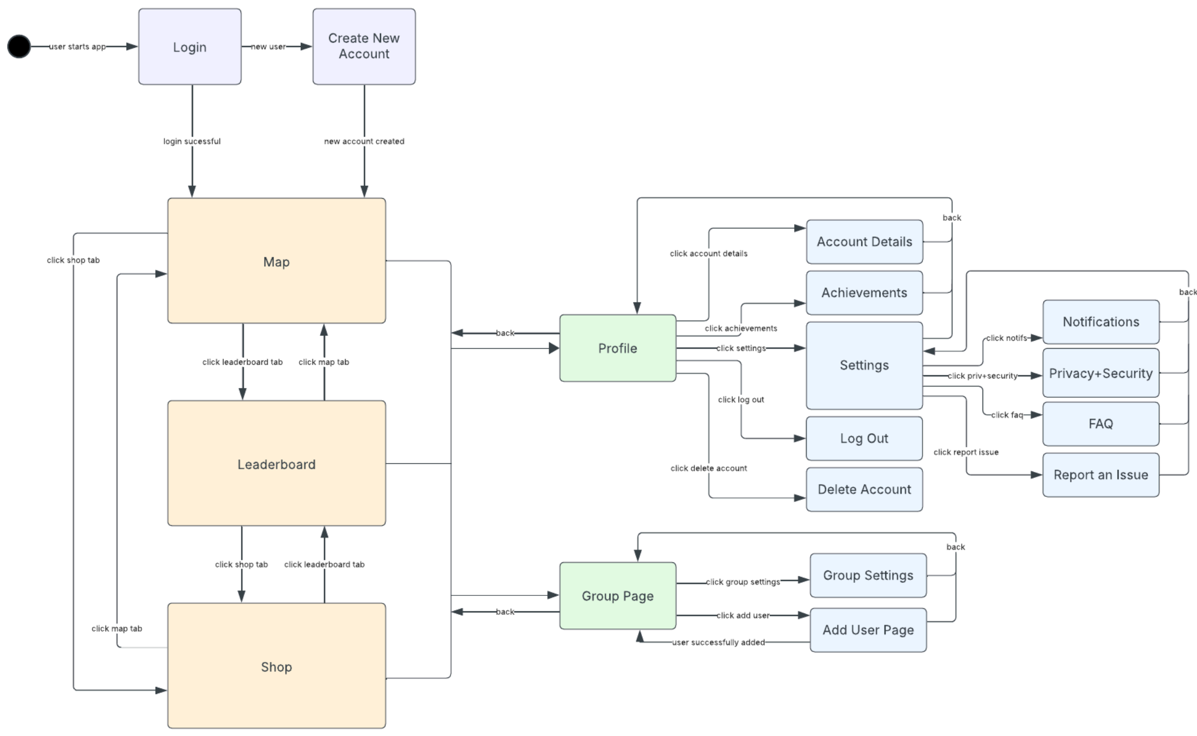
- Groups can be public or private (controlled by the is_public boolean field).

- Beacon Event
    - Each beacon has a unique ID (primary key).
    - Beacons have a start time and end time indicating their availability.
    - Each beacon is assigned a specific location (latitude and longitude).
    - Beacons are associated with a group (foreign key), meaning only members of that group can interact with them.
    - Each beacon has a list of users that have and are yet to reach the beacon

- Map
    - It is associated with a group_id to display relevant locations (users in the group and beacon location).
    - Uses Google Maps API to display user and beacon locations.

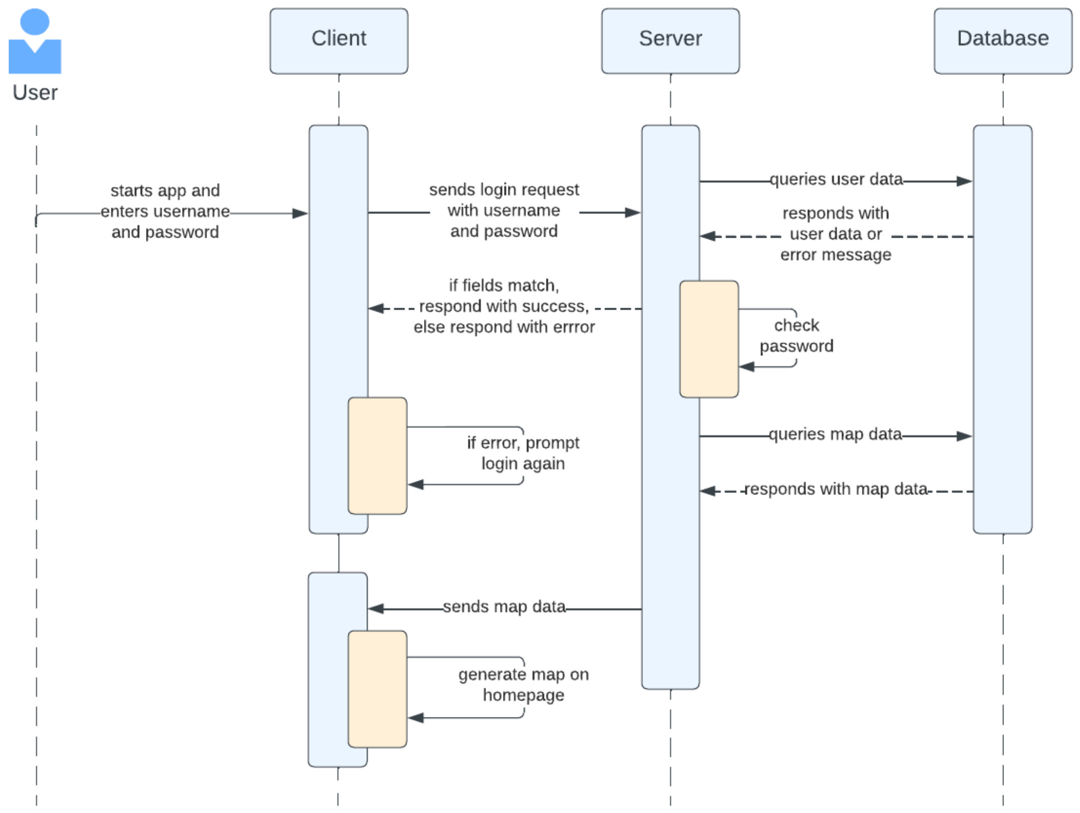
Navigational Flowchart

Upon opening the app, users are restricted to the login page until they successfully login or make a new account. Afterwards, they are sent to the map page, which is one of the three main pages along with leaderboard and shop. Users navigate between these three pages via a navigation bar at the bottom of the UI. On any of these three pages, users can also access their profile page and their group page via icons at the top right and left, respectively, of the UI. These two pages display a list of more specific pages that the user can select to view.

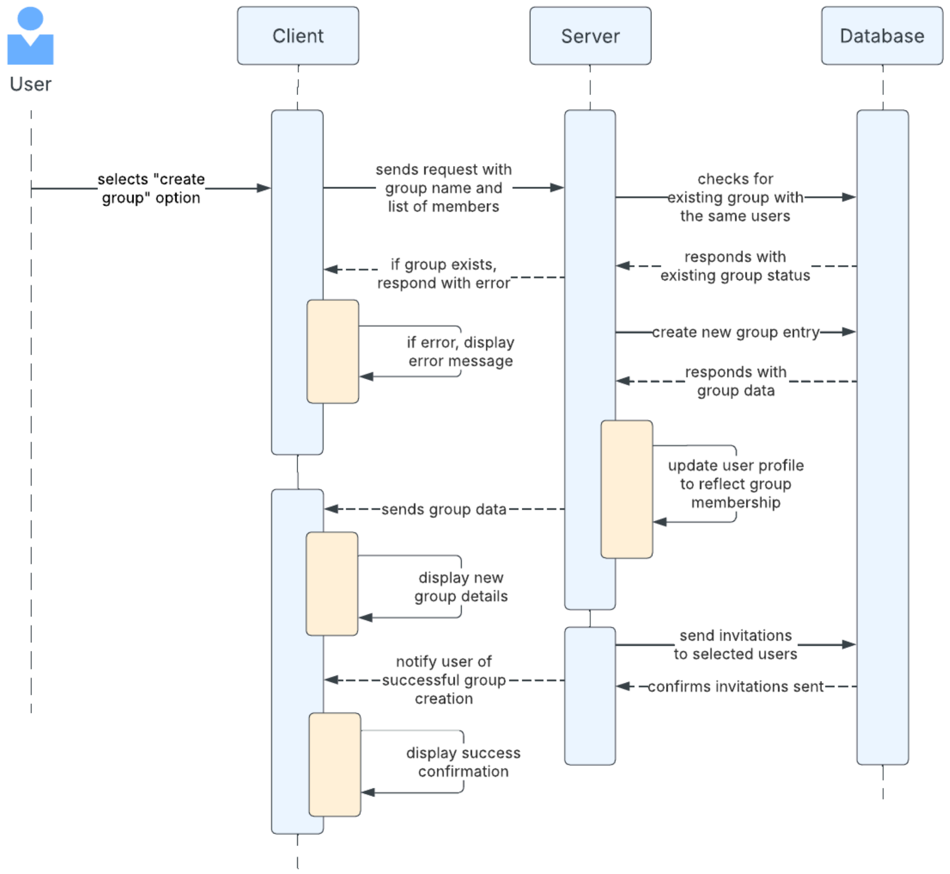


Sequence Diagrams

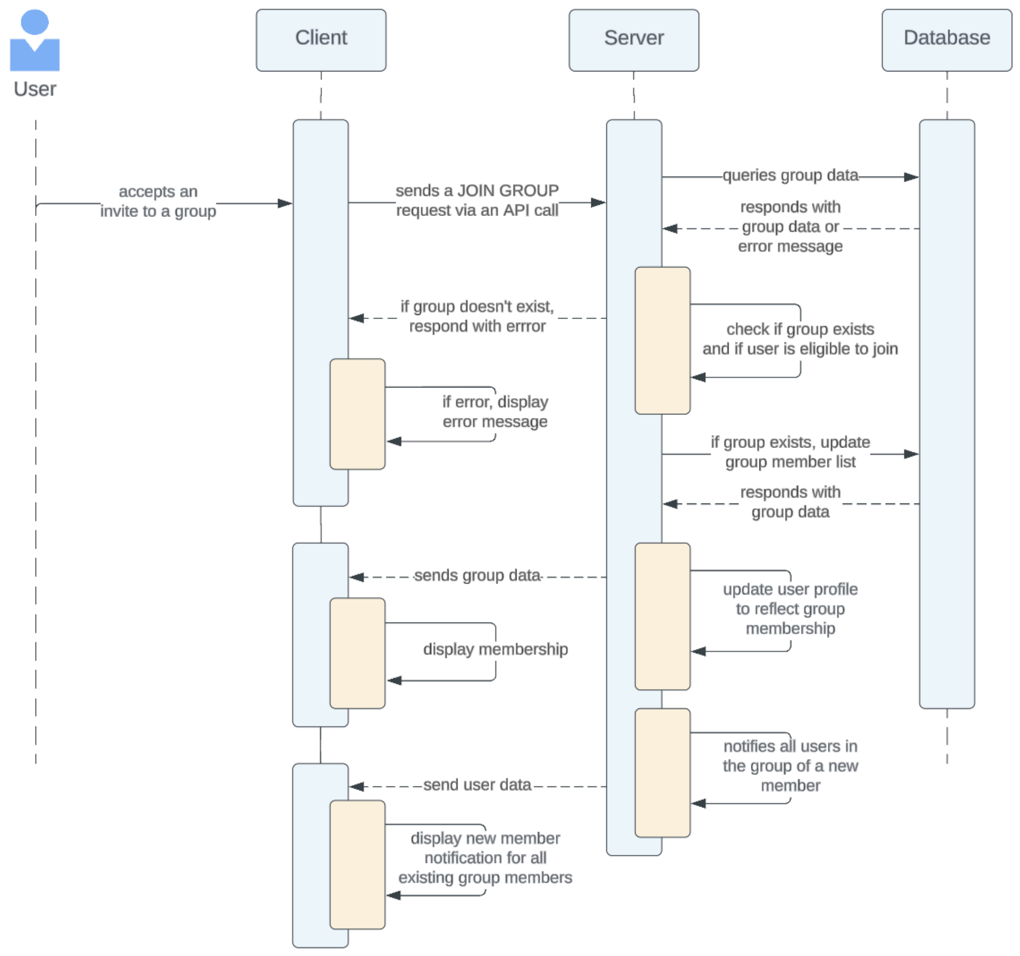
1. Sequence of events when users log in


User
Client
Server
Database
starts app and enters username and password
sends login request with username and password
queries user data
responds with user data or error message
check password
if fields match, respond with success, else respond with errror
if error, prompt login again
queries map data
responds with map data
sends map data
generate map on homepage

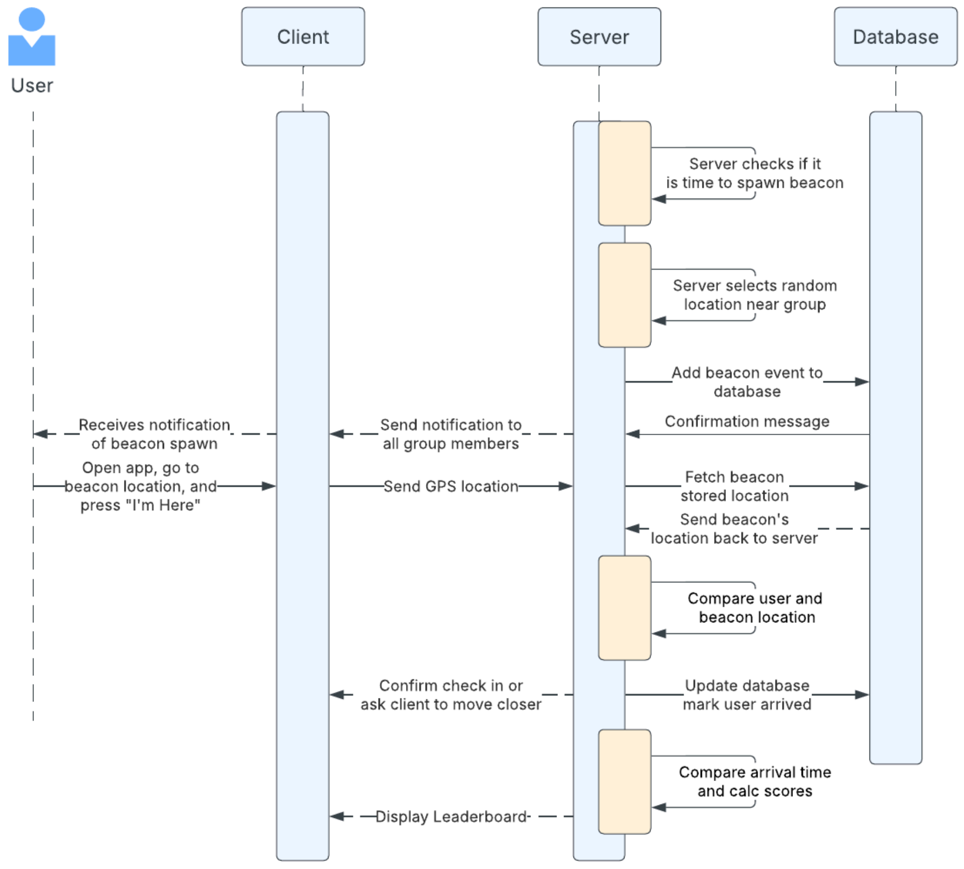
2. Sequence of events when a user creates a group


User
Client
Server
Database
selects "create group" option
sends request with group name and list of members
checks for existing group with the same users
responds with existing group status
if group exists, respond with error
if error, display error message
create new group entry
responds with group data
update user profile to reflect group membership
sends group data
display new group details
send invitations to selected users
confirms invitations sent
notify user of successful group creation
display success confirmation

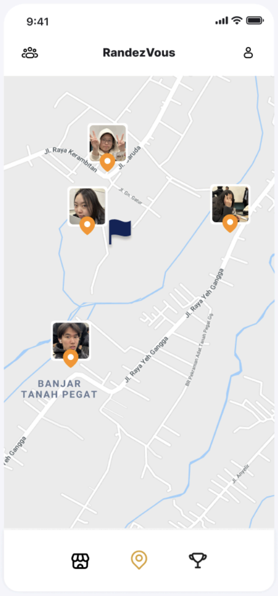
3. Sequence of events when users join a group

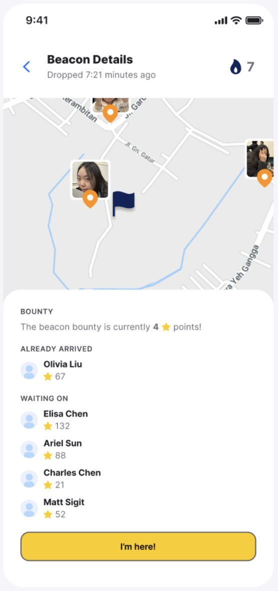

User
Client
Server
Database
accepts an invite to a group
sends a JOIN GROUP request via an API call
queries group data
responds with group data or error message
if group doesn't exist, respond with errror
check if group exists and if user is eligible to join
if error, display error message
if group exists, update group member list
responds with group data
sends group data
update user profile to reflect group membership
display membership
notifies all users in the group of a new member
send user data
display new member notification for all existing group members

4. Sequence of events when Beacon Event occurs


User
Client
Server
Database
Server checks if it is time to spawn beacon
Server selects random location near group
Add beacon event to database
Confirmation message
Receives notification of beacon spawn
Send notification to all group members
Open app, go to beacon location, and press "I'm Here"
Send GPS location
Fetch beacon stored location
Send beacon's location back to server
Compare user and beacon location
Confirm check in or ask client to move closer
Update database mark user arrived
Compare arrival time and calc scores
Display Leaderboard

UI Mockups




9:41

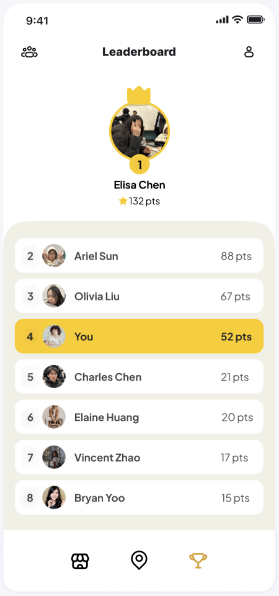
Beacon Details
Dropped 7:21 minutes ago
7

BOUNTY
The beacon bounty is currently 4 ⭐ points!

ALREADY ARRIVED
Olivia Liu ⭐ 67

WAITING ON
Elisa Chen ⭐ 132
Ariel Sun ⭐ 88
Charles Chen ⭐ 21
Matt Sigit ⭐ 52

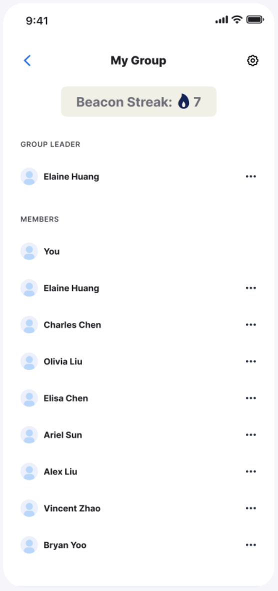
I'm here!


9:41

Leaderboard

1 Elisa Chen ⭐132 pts

2	Ariel Sun	88 pts
3	Olivia Liu	67 pts
4	You	52 pts
5	Charles Chen	21 pts
6	Elaine Huang	20 pts
7	Vincent Zhao	17 pts
8	Bryan Yoo	15 pts

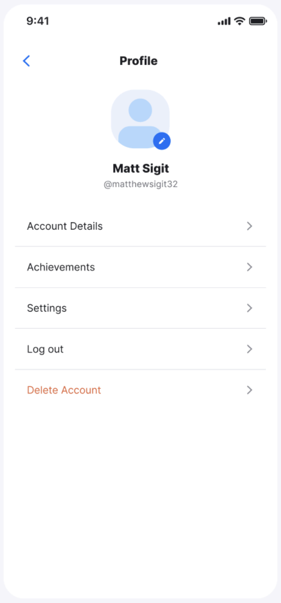

9:41

My Group

Beacon Streak: 7

GROUP LEADER
Elaine Huang

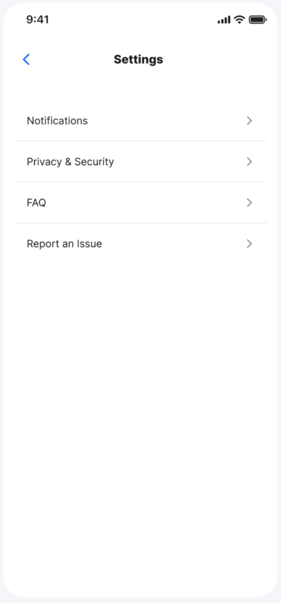
MEMBERS
You
Elaine Huang
Charles Chen
Olivia Liu
Elisa Chen
Ariel Sun
Alex Liu
Vincent Zhao
Bryan Yoo


9:41

Profile

Matt Sigit
@matthewsigit32

Account Details
Achievements
Settings
Log out
Delete Account


9:41

Settings

Notifications
Privacy & Security
FAQ
Report an Issue