CS30700
Design Document

Team 9

- Aastha Sinha
- Dhairya Amin
- Kevin Gao
- Siddhesh Patil
- Stephen Burns
- Upmanyu Shrivastava

Index

Purpose

In these nascent, stressful years of college life, you should find time to relax. In a university like Purdue, finding friends to travel with can be a challenging task, as they may not share the same interests as you. While platforms to plan your trips do exist, namely Penroads and Reddit, these are hard to curate, making it difficult for users to trust the people they find on there. We want to make an app for Purdue students which not only fosters trust but also helps them find travel buddies and plan their next vacation.

Functional Requirements

1. User account

   As a user
   - I would like to sign-up for a Travelin account with my Purdue email.
   - I would like to log-in to my Travelin account using my username and password.
   - I would like to reset my password if I forget it.
   - I would like to logout from my Travelin account.
   - I would like to add a bio to my profile.
   - I would like to see my ratings.
   - I would like to see reviews about me.
   - I would like to edit my profile.
   - I would like to add photos of places I traveled to on my profile.
   - I would like to delete my account
   - (If time allows) I would like to add a profile picture.

2. Home page

   As a user
   - I would like to see my recommended destinations on my home page.
   - I would like to easily navigate to different pages from the home page.

3. Find friends

   As a user
   - I would like to see people recommended to me based on my interests.
   - I would like to filter the people recommended to me.
   - I would like to see other people's profile.
   - I would like to rate the people.
   - I would like to leave a review on their profile.
   - I would like to edit the rating and review I posted on other users profile.
   - I would like to send people messages.

4. Forum

   As a user
   - I would like to create a post.
   - I would like to reply to a post.
   - I would like to post pictures.

- I would like to see who created or replied to a post.
- I would like to upvote a post.
- I would like to downvote a post.

5. Hotels and Flights

   As a user
   - I would like to find hotels at selected destinations.
   - (If time allows) I would like to make bookings at hotels.
   - I would like to find flights to my destination.
   - (If time allows) I would book a flight to my destination.

6. Messaging

   As a user
   - I would like to send messages to other users.
   - I would like to receive messages from other users.
   - I would like to receive notifications.
   - (If time allows) I would like to make groups.
   - (If time allows) I would like to send pictures.

Non-Functional Requirements

1. Performance

   As a Developer
   - I would like the application to be able to be used on Android phones.
   - I would like the application to run smoothly without crashing.
   - I would like most non-API calls in the app to be in less than 500ms while API calls should generally be 1 second or less.
   - I would like UI transitions to go smoothly.

2. Server

   As a Developer
   - I would like the server to support real-time server-client communication.
   - I would like the server to be able to store users data to a database

3. Appearance

   As a Developer
   - I would want the application to be aesthetically pleasing and intuitive to use.

4. Security

   As a Developer
   - I would like the user login to be secure.
   - I would like the user's password to be hashed and salted before storing in the database.
   - I would like no user information to be stored locally.

5. Usability

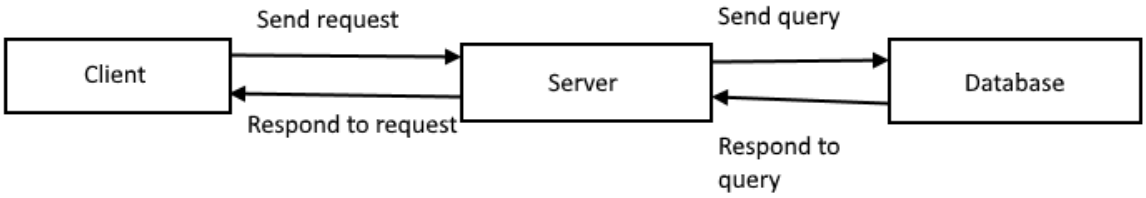
   As a Developer

   - I would like the application to be accessible on a wide array of Android devices.
   - I would want the project to be kept as maintainable as possible for future work.

Design Outline

High-Level Overview

This project will be a mobile Android application that allows users to plan trips and coordinate with other users who have similar plans based on user's preferences. This application will use the client-server model, where a Realm cloud server handles access from a large number of clients. The server will accept client requests, validate requests, access or store data in the database, and return responses to the client(s) accordingly.



1. Client
    a. Client provides the user an interface with the system.
    b. Client sends queries to the server.
    c. Client receives and interprets responses from the server and renders changes in the user interface.

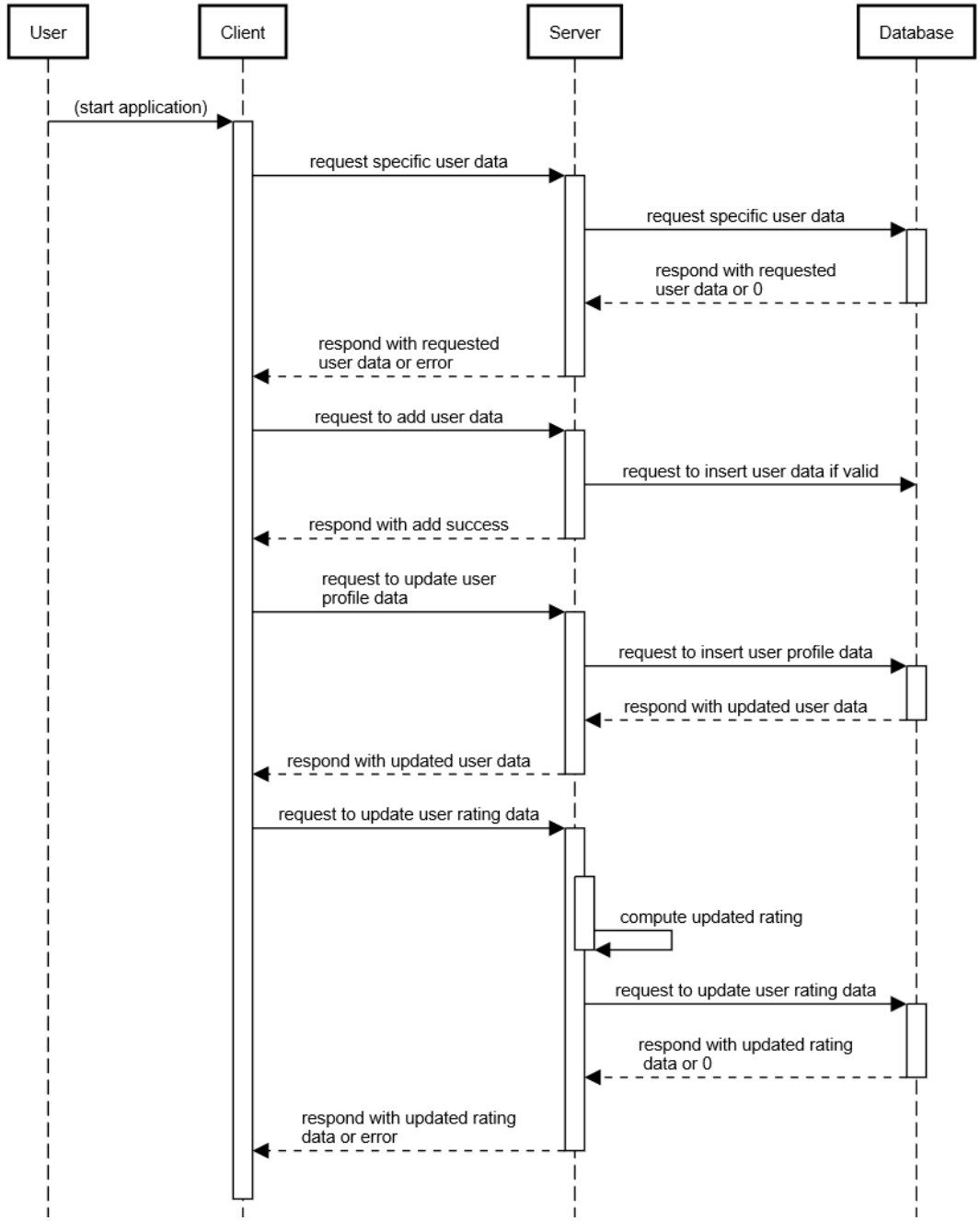
2. Server
    a. Server receives and handles requests from the clients.
    b. Server validates requests and sends queries to the database to insert, modify, or obtain data as needed.
    c. Server creates appropriate responses and sends them to targeted clients.

3. Database
    a. A relational database stores all the data used in the application, such as user information, forums, and posts, etc.
    b. Database responds to queries from the server and sends the corresponding data back to the server.

Sequence of Events Overview

The below sequence diagram shows the typical interaction between clients and the database. The sequence is started by a user logging into or starting up the app. As the user logs in, the client sends a query to the server. The server handles the request and then sends a query to the database, which returns the corresponding information. After the client has logged in, it can send further requests to the server for other actions such as managing profile data, planning trips, communicating with users, and rating other users. In order to respond to these requests, the server queries the database to acquire data. The database also responds with updated data whenever its data is updated. Once the server has the requested data from the

database, it returns the parsed data to the client.



Design Issues

Functional Issues

1. How do we allow users to communicate with each other?
    - Option 1: Allow a forum for posts
    - Option 2: User to user messages
    - Option 3: Both

Choice: Option 3

Justification: The purpose of allowing users to communicate with each other in a peer-to-peer chat is for them to see if they want to travel together, while also hashing out the details. The forum, on the other hand, can be used for miscellaneous purposes like searching for multiple people to travel with, discussing an event or even looking for suggestions from people.

2. How do we allow users to set interest and location tags?
    - Option 1: Users select pre-existing tags
    - Option 2: Allow the user to create tags
    - Option 3: Users select pre-existing tags and are allowed to create tags for themselves that can be approved by an admin before they are public.

Choice: Option 3

Justification: Creators of the app may not be able to come up with a vast enough suite of tags to encompass everyone's interests. However, allowing uncensored user-created tags could result in spelling errors or confusing results. Users can select from pre-existing tags and if they find their interests not adequately represented they can create a tag that can be made public after admin approval. This adds some maintenance concerns in the short term but, with building up the number of tags, we can reduce maintenance and improve user experience in the long run.

3. Do we make user reviews anonymous?
    - Option 1: Yes
    - Option 2: No
    - Option 3: Allow the users to decide

Choice: Option 2

Justification: Anonymous reviews would allow people to leave disingenuous or fake reviews for a user, resulting in incorrect ratings for said user. Allowing the user to see who is leaving the reviews prevents the possibility of a skewed rating.

4. What information do we require from the user at account creation?
    - Option 1: Username, email, and password
    - Option 2: Username and password
    - Option 3: Email and password

Choice: Option 1

Justification: While setting up an account, we plan on asking the user for a unique username that can be stored in the database. This will allow other users to easily find them and leave

ratings/start conversations. Getting an email and password is also important as the email not only acts as an added layer of verification but also can be used to reset the password if the user forgets it.

5. What information about the user do we display on their profile page?
    - Option 1: Name, profile picture, bio, rating, interests, and message user
    - Option 2: Photos, places visited, and user's posts
    - Option 3: Both options

Choice: Option 3

Justification: When looking for a travel companion, ratings and interests are helpful to look for a user who likes similar places, but checking out the images of locations explored by the user can give a better understanding of specific places the user might be interested in. Other than chatting with them, looking at their posts might help the user get to know their personality better and help find a compatible travel companion.

Non-functional Issues

1. What query database do we use?
    - Option 1: Realm
    - Option 2: SQLite

Choice: Option 1

Justification: Realm is easy to use as it uses objects to store data and the data models are defined using normal Java classes. Query results in Realm are automatically converted to objects whereas SQLite query results are returned in a cursor object which requires longer operation time.

2. Memory usage versus the number of queries (response time)
    - Option 1: Store more data structures in classes to keep track of relationships but use more memory
    - Option 2: Store fewer data structures in classes and use a search function to find relationships, increasing query/response times

Choice: Option 2

Justification: We determined that the possibility of higher response times is acceptable within the constraints of our app. One reason we decided this is because it makes the code less cluttered and easier to read when working in a team. We also decided that it was more important to preserve storage space in the database/server than it is to ensure minimally shorter response times.

3. What Android programming language do we use?
    - Option 1: Java
    - Option 2: Kotlin

Choice: Option 1

Justification: None of our team members have experience developing with Kotlin. Every member has significant experience with programming with Java, but not much experience with developing an android app. Developing android with java will be easier for every

member to pick up because of previous experience and Java having various tutorials to help learners.

4. How do we keep the recommended companions data for a user updated?
    - Option 1: Update automatically after a time interval (for example, 1 minute)
    - Option 2: Update when user starts the app
    - Option 3: Update when a user adds/deletes a new tag

Choice: Option 1

Justification: Updating the recommendations in a time interval covers all possible cases for updates. It shouldn't be updated when a user adds/deletes a tag as other users might be adding tags too and this might result in very quick changes in the recommended list. Updating over time will allow users to check out profiles and go back and forth to find travel companions.

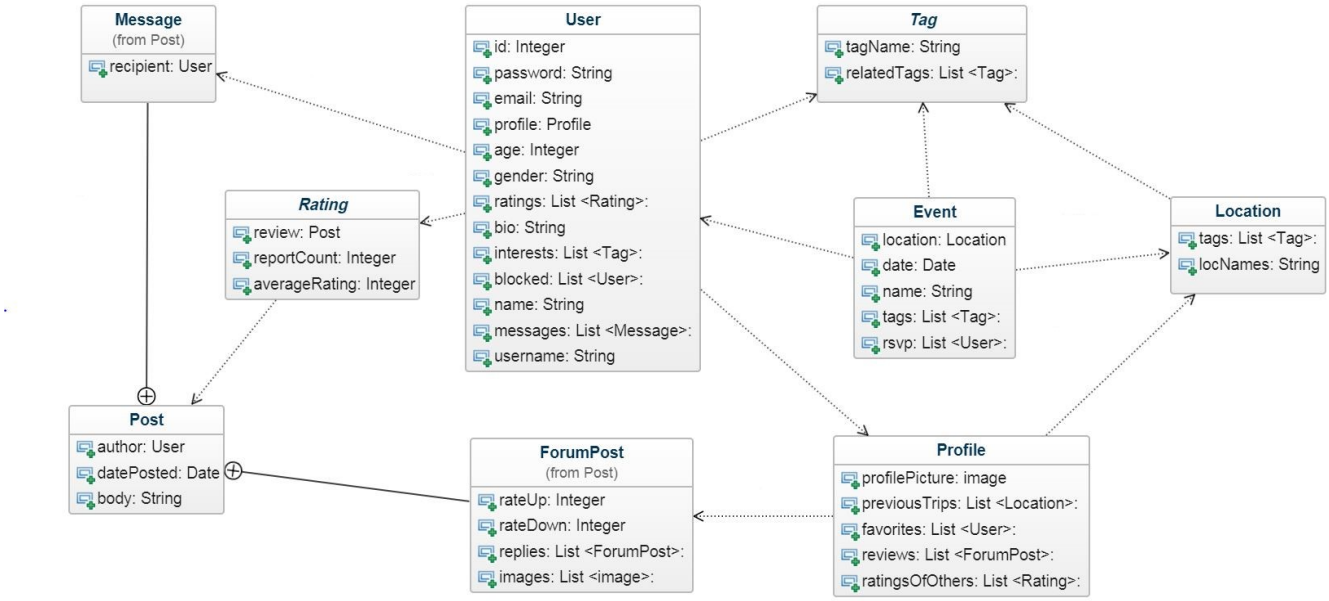
5. How compatible do we want our app to be with older versions of Android (what is the minimum API we should use?)

Choice: The minimum API we will use is API 15 (Android 4.0.3)

Justification: Using this API allows compatibility with the majority of Android devices that are being used today. As of now, we do not foresee any circumstances in which a newer API must be used. If it happens that using a feature only available in a newer API is needed to make the app function correctly, this decision will be changed.

Design Details

Class Design



Description of Classes and Interaction between Classes

- **User**
  - User object is created when someone signs up in our application
  - Each user will be assigned a unique id and username
  - Each user will have a name, email, and password for login
  - Each user chooses their interest tags.
  - Each user has a rating which is the average rating of all ratings received
  - Each user has age, gender, and a bio
  - Each user has a profile
  - Each user has a list of blocked users
  - Each user has a chat box with other users. It is a hashmap of user and list of messages
  - Each user has a count of the number of times reported by other users (hidden to the user)

- **Profile**
  - Profile object is created when a user is created
  - Each profile has a profile picture
  - Each profile has a list of places visited by the user
  - Each profile has a list of posts made by the user
  - Each profile has a list of ratings given to other users
  - Each profile has a list of other users (friends)

- **Rating**
  - Rating object is created when a user is created
  - Each user's rating has a list of posted reviews given by other users
  - Each user's rating has a review score

- **Post**
  - Post object is created when a user posts something
  - Each post has the author user and the date it was created on.
  - Each post has the content of the post

- **ForumPost**
  - Forum post object is created when the user posts in the user
  - Each forum post has upvotes and downvotes awarded by other users
  - Each forum post has a list of replies which are forum posts
  - Each forum post has a list of images

- **Tags**
  - Tag objects are preset with name and ID
  - These are interest tags that a user can choose and display on their profile
  - Each tag has a list of related tags that a user might be interested in

- **Flight**
  - Each flight object has origin and destination names that a user queries
  - Each flight object has a departure date that a user queries

- **Hotel**
  - Each hotel object has a city name and number of guests
  - Each hotel has a radius from the city's center
  - Option to choose the best rate only

- **Location**
  - Location object is preset with name and ID
  - Each location has a list of interest tags

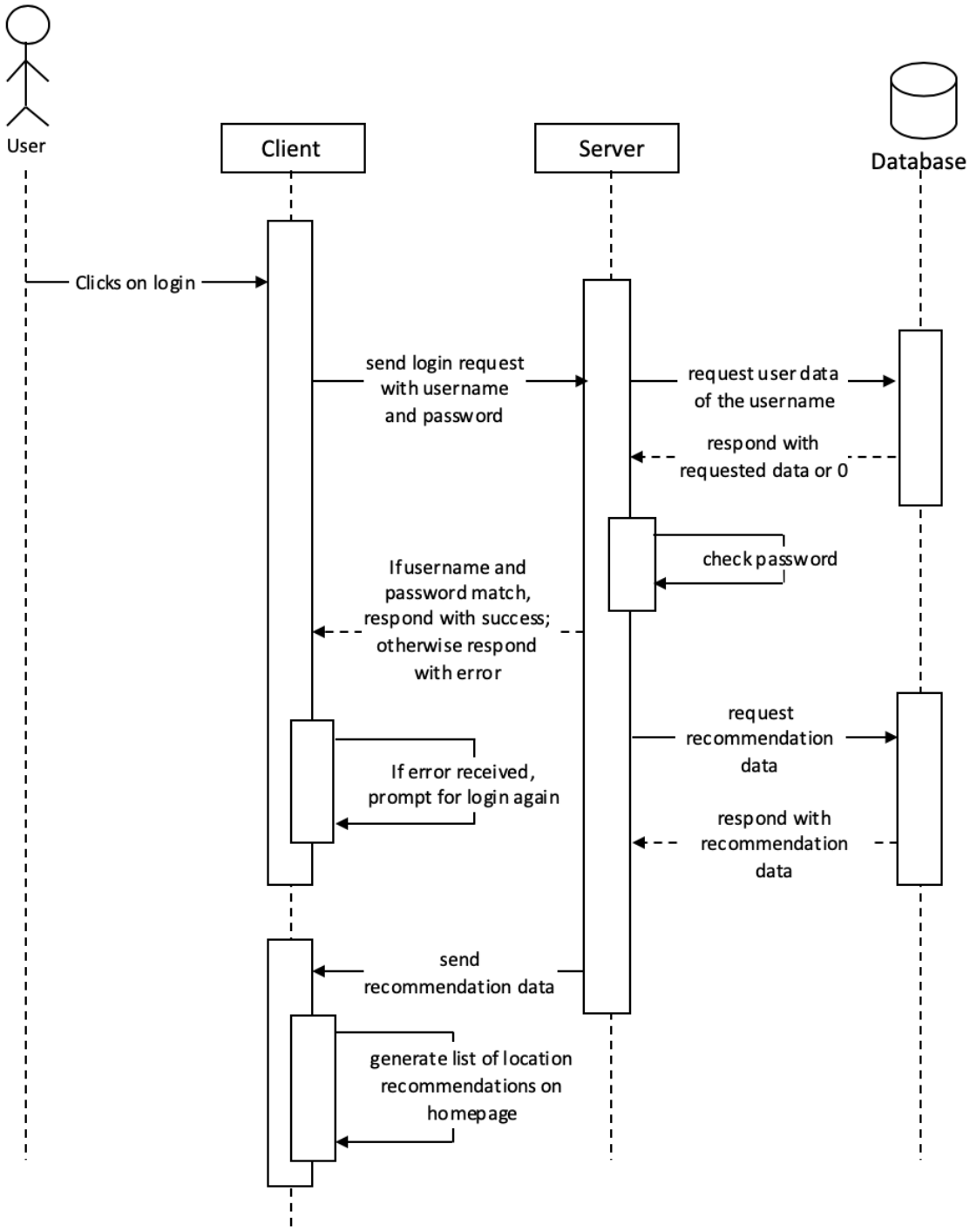
- **Message**
  - Message object is created when a user sends a message
  - Each message is a type of post that a user sends to another user

- **Event**
  - Event object is created when a user creates an event
  - Each event has a location, start date, and name.
  - Each event has a list of related tags, user RSVP'd and forum posts.

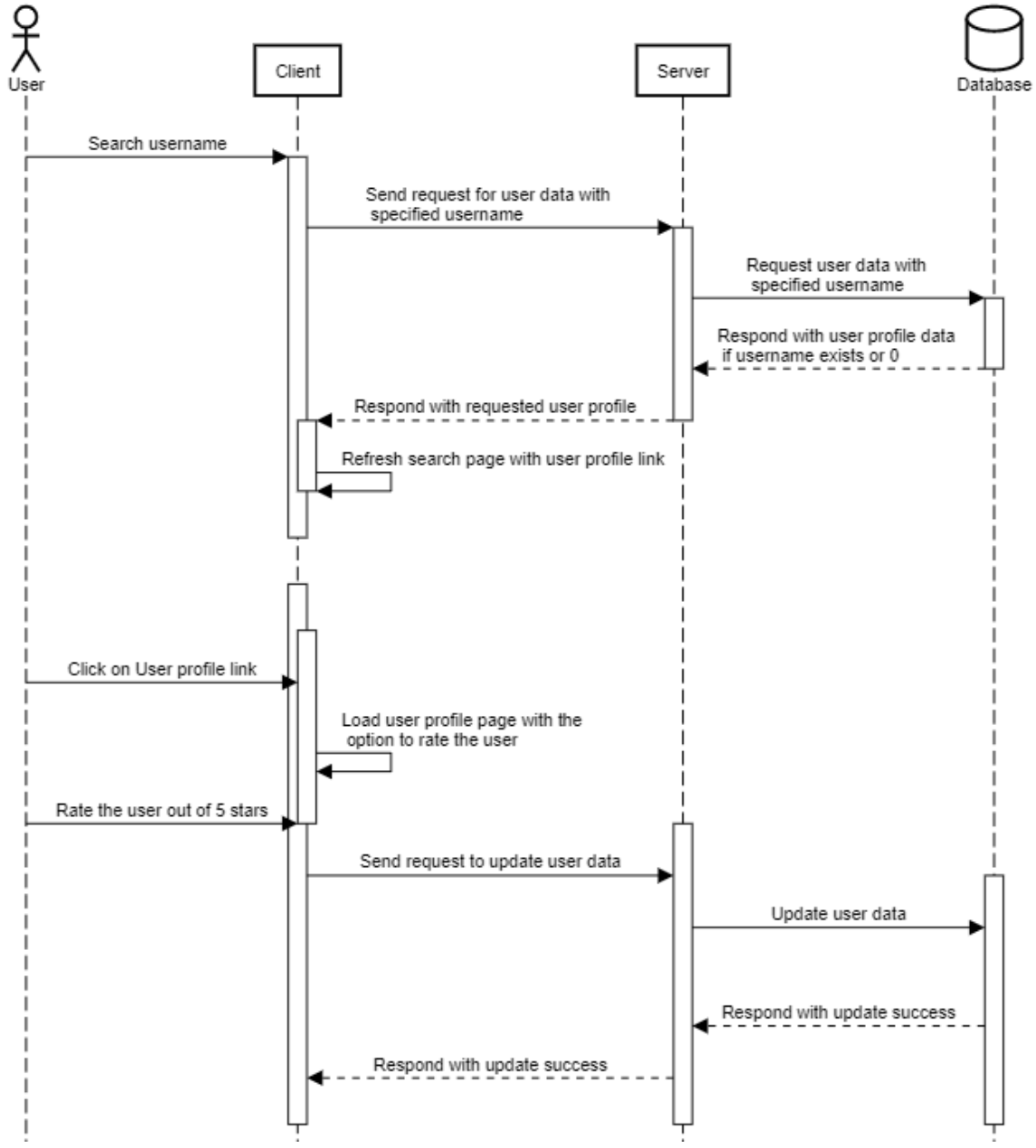
Sequence Diagram

The diagrams on the following five pages portray sequences of major events in our android application including user login, rating a user, searching for users, sending messages, and finding flights. When user performs an action on the android device, the client will acquire the data needed from the database. The client will receive data and update the android device's display consistently.

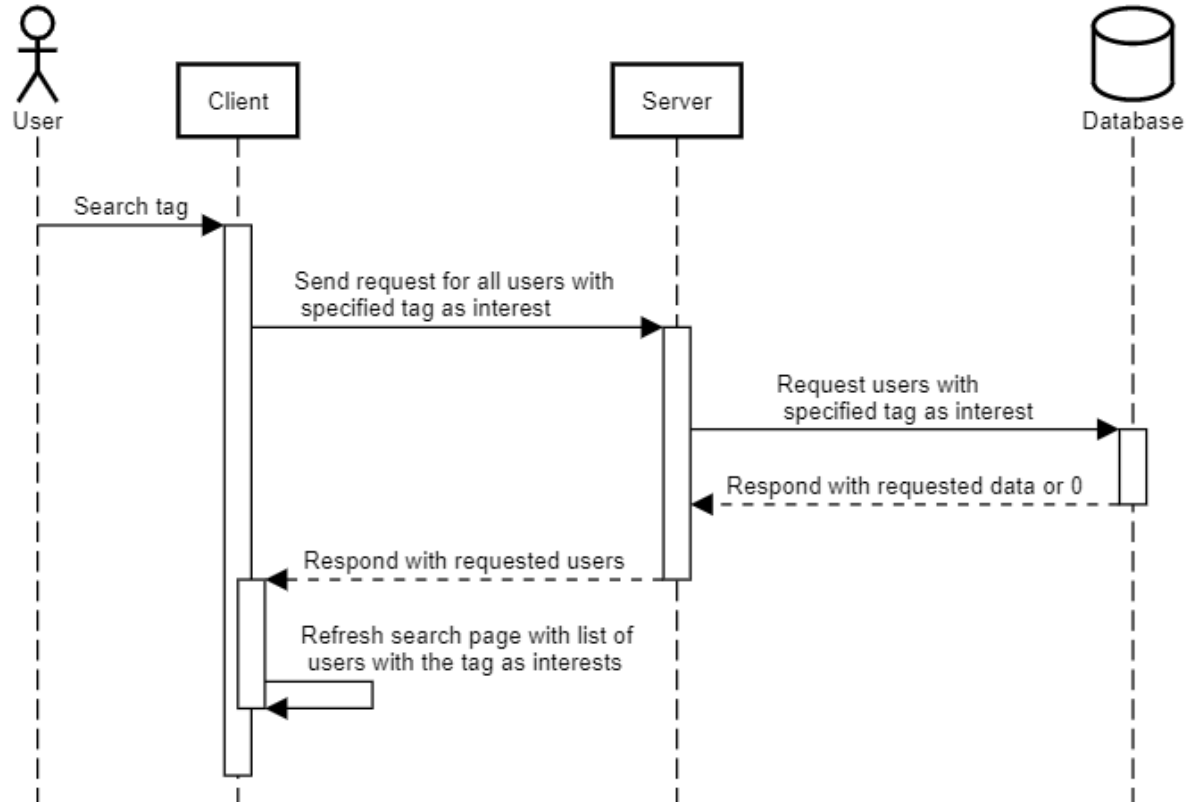
1. Sequence of events when users login



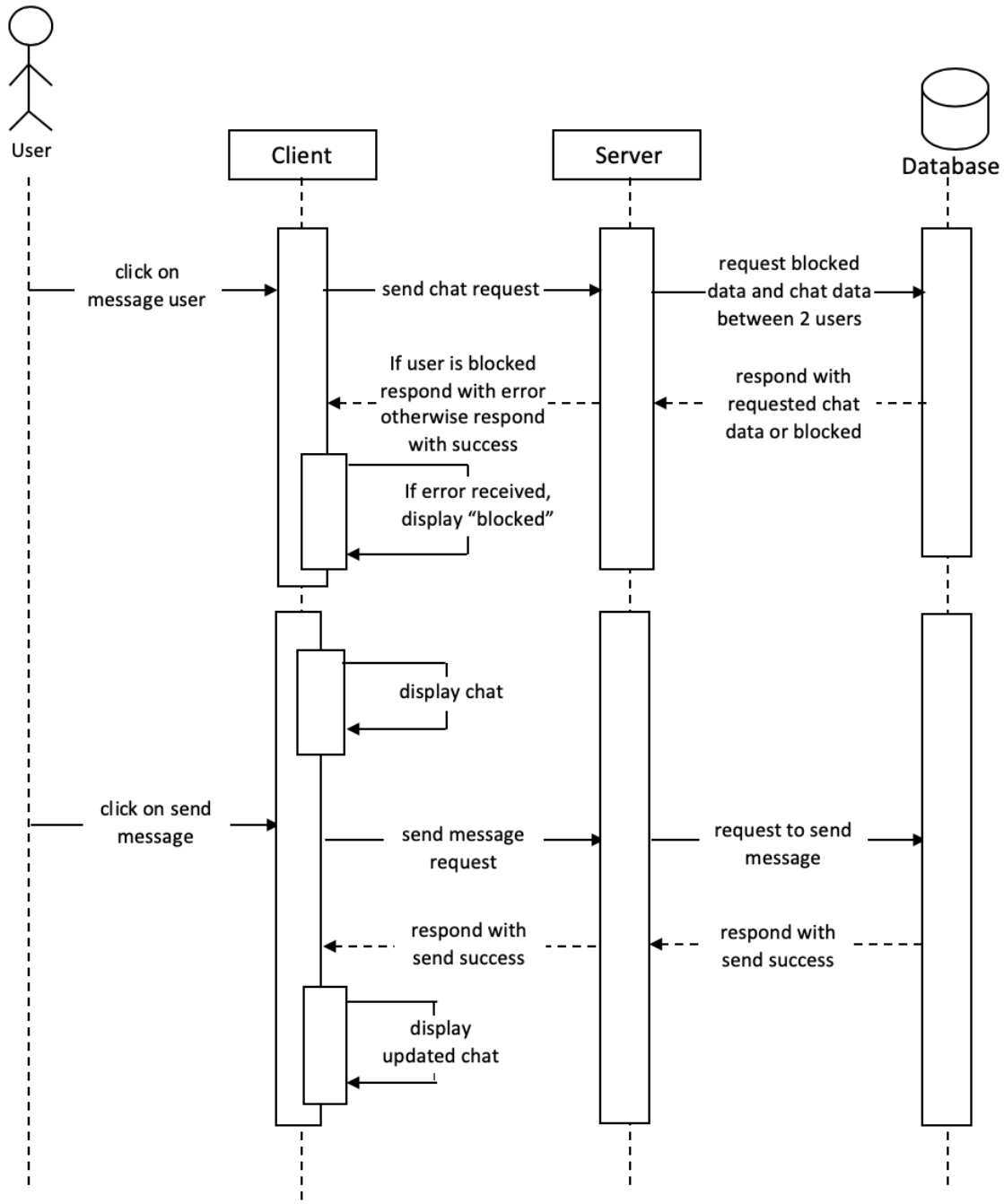
2. Sequence of events when users rate a certain user.



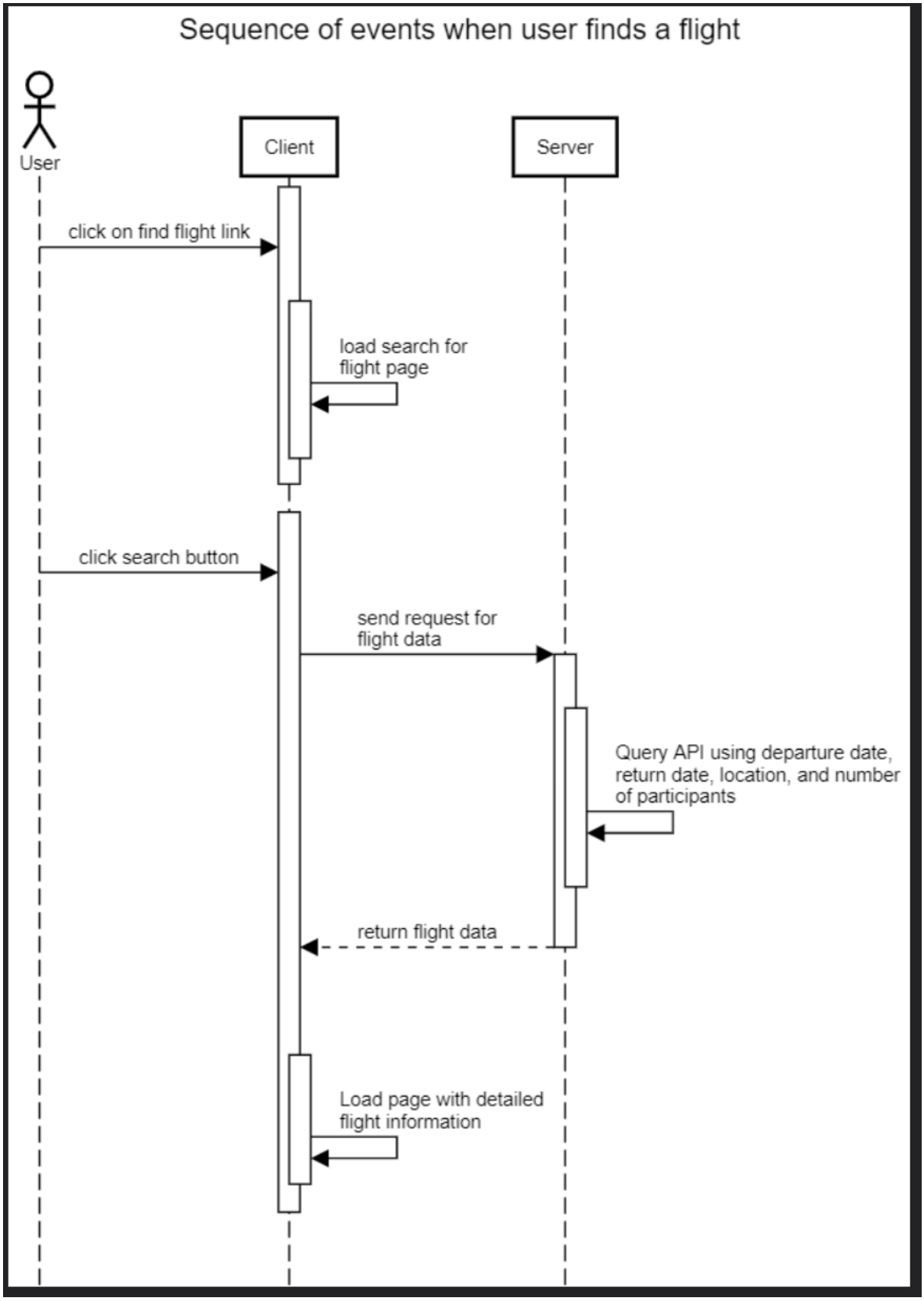
3. Sequence of events when users search other users based on similar interests(tags)



4. Sequence of events when users send messages to each other.

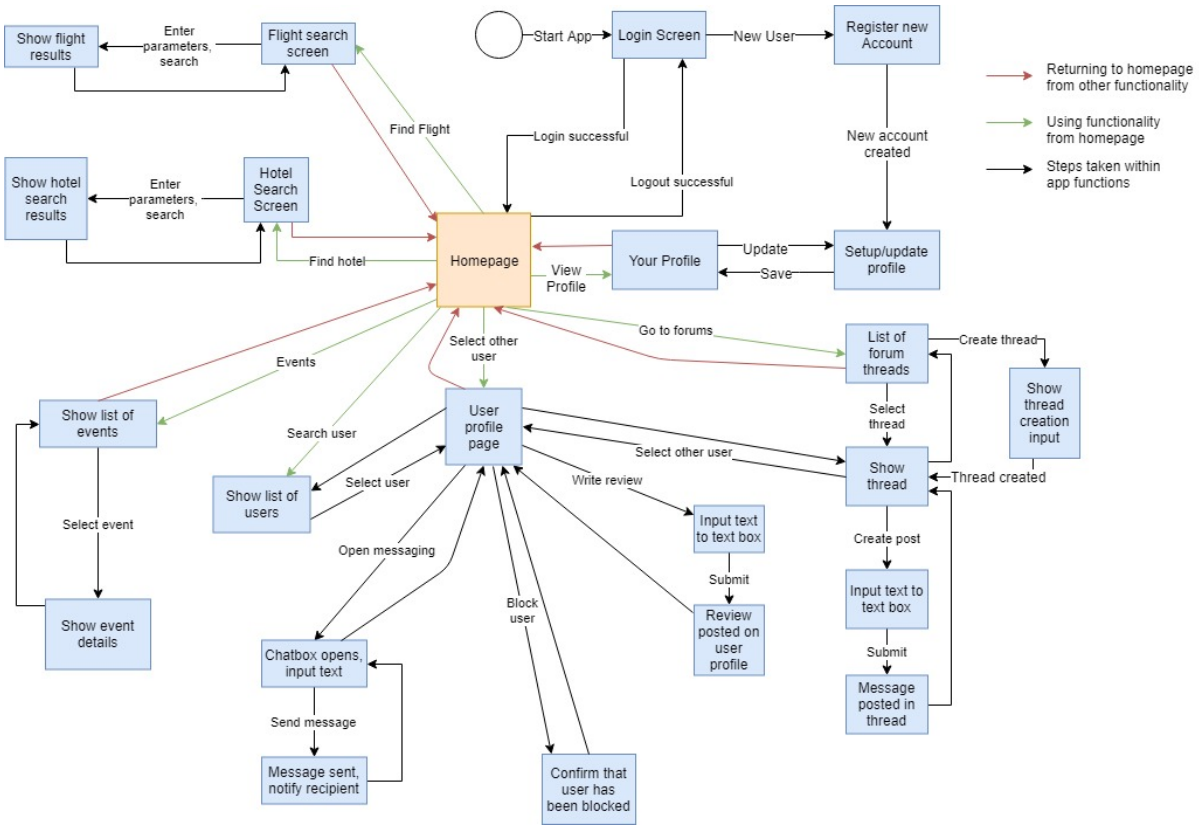


5. Sequence of events when users find flights for their travels



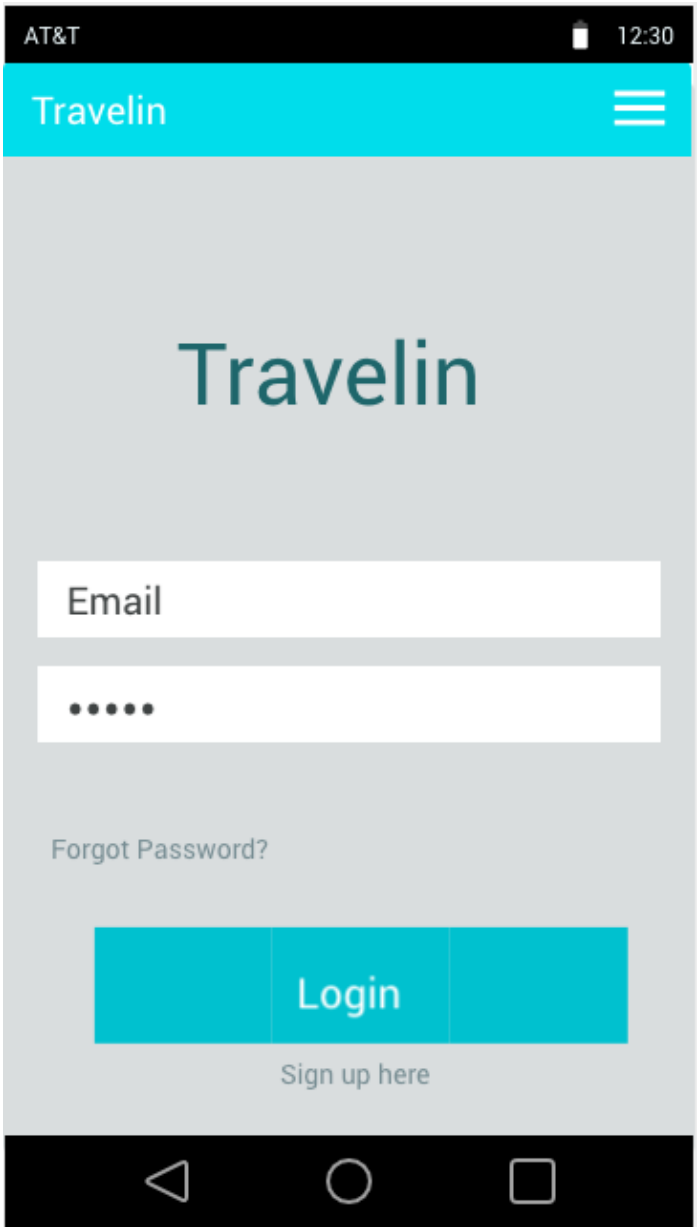
Navigation Flow Map

The design of our navigation emphasizes simplicity and the ability to quickly access any feature of the app. After login, if an account exists, the user will be sent straight to the homepage. If not, they will create an account and then be sent to the homepage. The homepage contains links to navigate to each individual feature of the app.

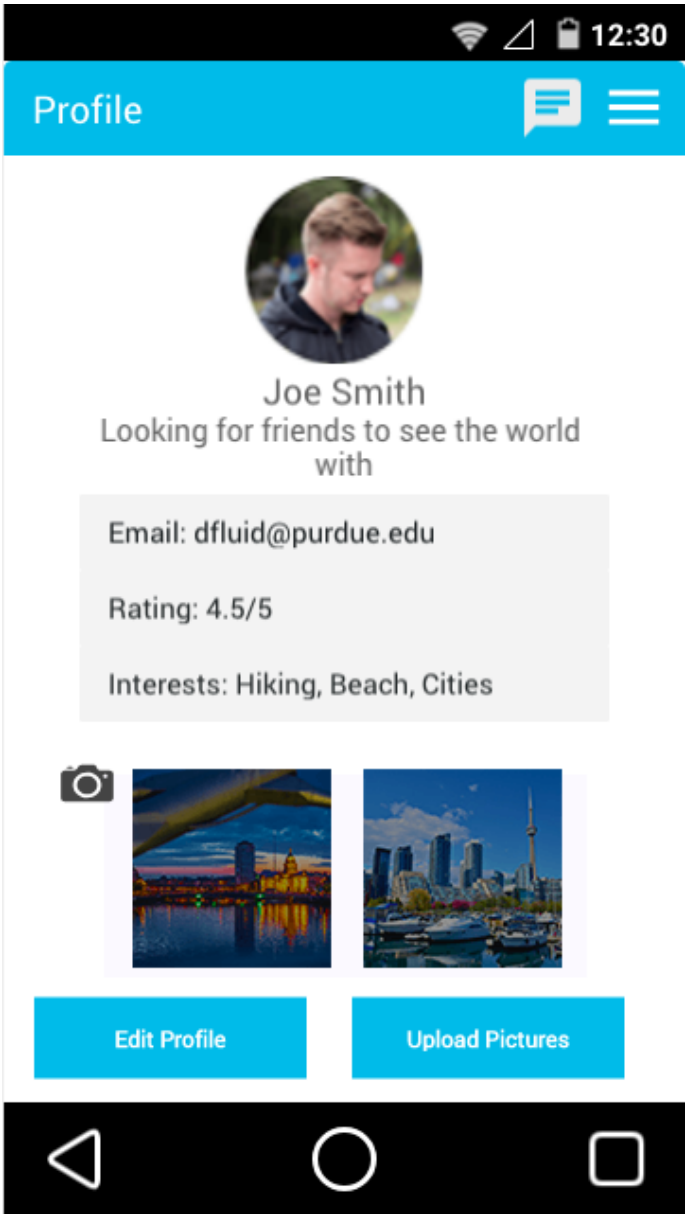


This diagram displays the way in which the user will be able to navigate between the different modules of the app. If time allows, a function bar will be implemented to allow users to move from any module to any other module.

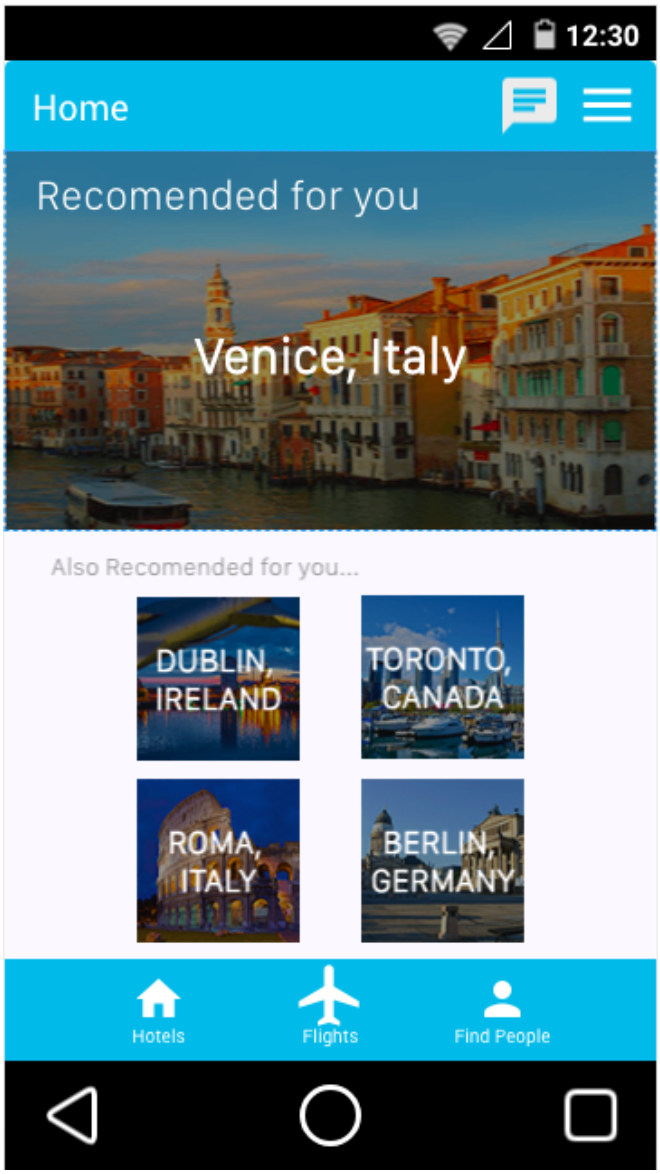
UI Mockup



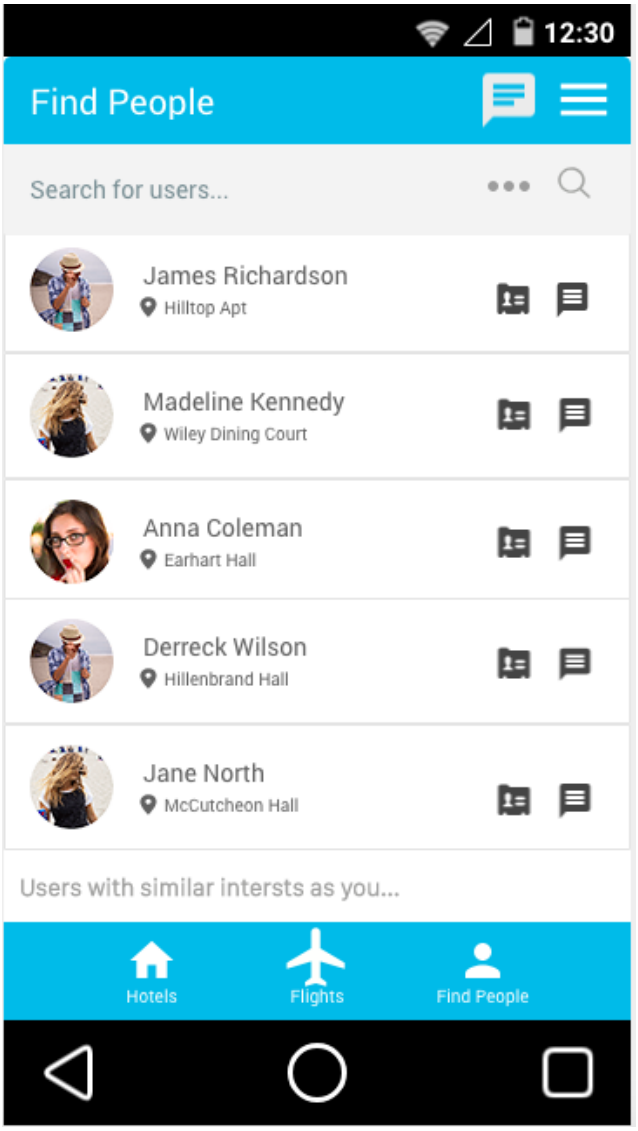
This is the Login page. We have a Forgot Password button for the user if they forget their password. There is also a button for a user to signup if they don't have an account.



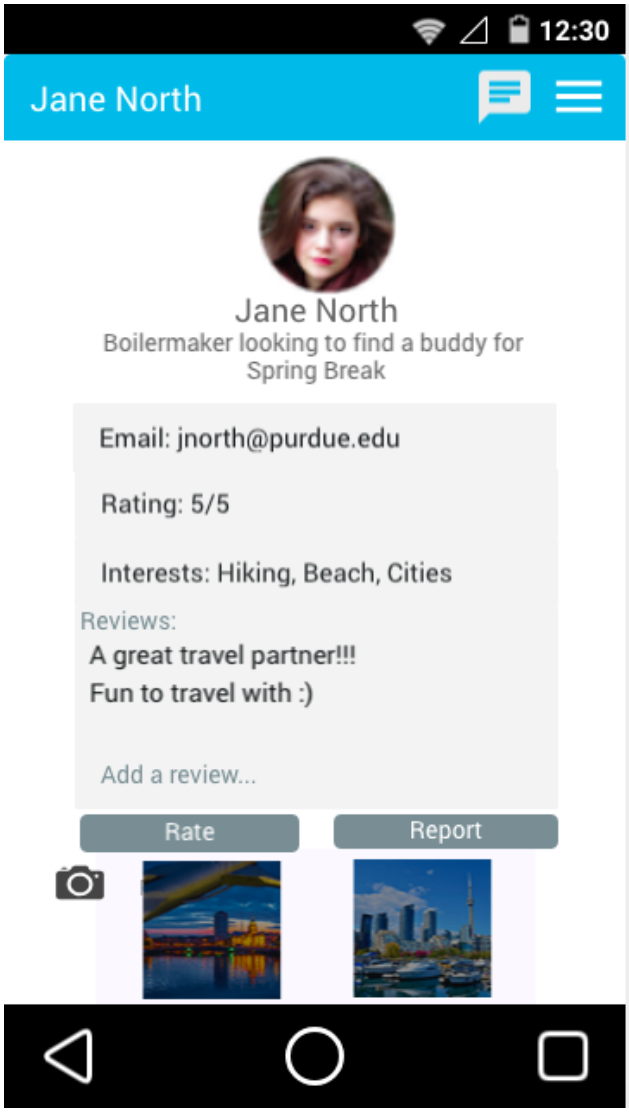
This is the profile page of the user where they will see the information about themselves. We also have a edit profile button where a user can edit their profile and also delete their account. The upload pictures button lets you upload pictures on your profile from your previous trips for other users to see. The navigation bar on the top displays the name of the page you are on and also has a direct shortcut to chats and a dropdown menu to navigate to other pages.



This is the home page where you see destination recommendations based on your interests. It also has a navigation bar in the bottom to easily switch between different pages like hotels, flights and find people.



This is the find people page where you will see other users with similar interests as yours. Next to their names, there will be two icons, one to go to their profile and the other to chat with them. Using the search bar on the top you will be able to search for a particular user and also filter the search results based on filters like gender and rating.



This is the profile of a different user. It will show reviews posted about that person and there will be an option to either rate (and review) and report that user.