



Design Document

Authors: Kunwar Sahni, Yash Bansal, William Kao, Ibrahim Saeed, Grayson
Harralson

Index

- **Purpose**
 - Functional Requirements
 - Non Functional Requirements

- **Design Outline <Page Number>**
 - Components
 - High Level Overview
 - Sequence of Events Overview

- **Design Issues <Page Number>**
 - Functional Issues
 - Non Functional Issues

- **Design Details <Page Number>**
 - Class Design
 - Sequence Diagram
 - Navigation Flow Map
 - UI Mockup
 - API Routes

Purpose

It can be difficult to plan a large trip using various services to book and keep track of flights, hotels, rental cars. That is why our project will make travel easier through aggregating all travel information into one master itinerary. There are existing services that attempt to create master itineraries but none automatically crawl through emails to create the said itinerary. Using this will allow our app to create an organized final product to simplify travel for all the tourists, explorers and frequent travelers that use MergeTrip.

There are two main applications that have some overlap with MergeTrip. The two applications have similar functionality, but their implementations differ from each. Triplt is an app that creates master itineraries just like ours, but the user has to manually forward the emails to a Triplt email. Another app is Wanderlog, which is an app that is mainly used for planning trips with friends that allows for Gmail syncing. Wanderlog, however, requires \$50 per year just to get access to the email syncing function, which is something we plan to implement in our basic functionality.

Functional Requirements

1. Creating an account/ Sign in
 - a. New users will register with their email, a username, and a password to create an account
 - b. This account will have full access to the application

- c. Existing accounts will be able to sign in to access their own itineraries that are linked with their own accounts
- d. Users can sign out of an account and sign into other accounts
- e. Users can change their username and password if necessary

2. Creating and updating itineraries

- a. Any user that has a registered account can connect/disconnect their Gmail account to their MergeTrip account to automatically enter information
- b. Any user can manually input their travel events and itineraries if they did not connect their Gmail to MergeTrip or
- c. Users can manually update their information so that the events added by MergeTrip are more accurate
- d. Users can manually delete travel events so that events not being attended will not show up on itinerary
- e. Users can see all trip details on a timeline/calendar

3. Creating and Joining Groups

- a. Users can search for other users on MergeTrip
- b. Users can create groups and invite other users to their groups to view and create travel itineraries
- c. Users can leave their groups and remove other users from their groups
- d. Users can invite other members of the group to an event, and can accept other group members' invitations

- e. Users can create roles in their groups

Non Functional Requirements

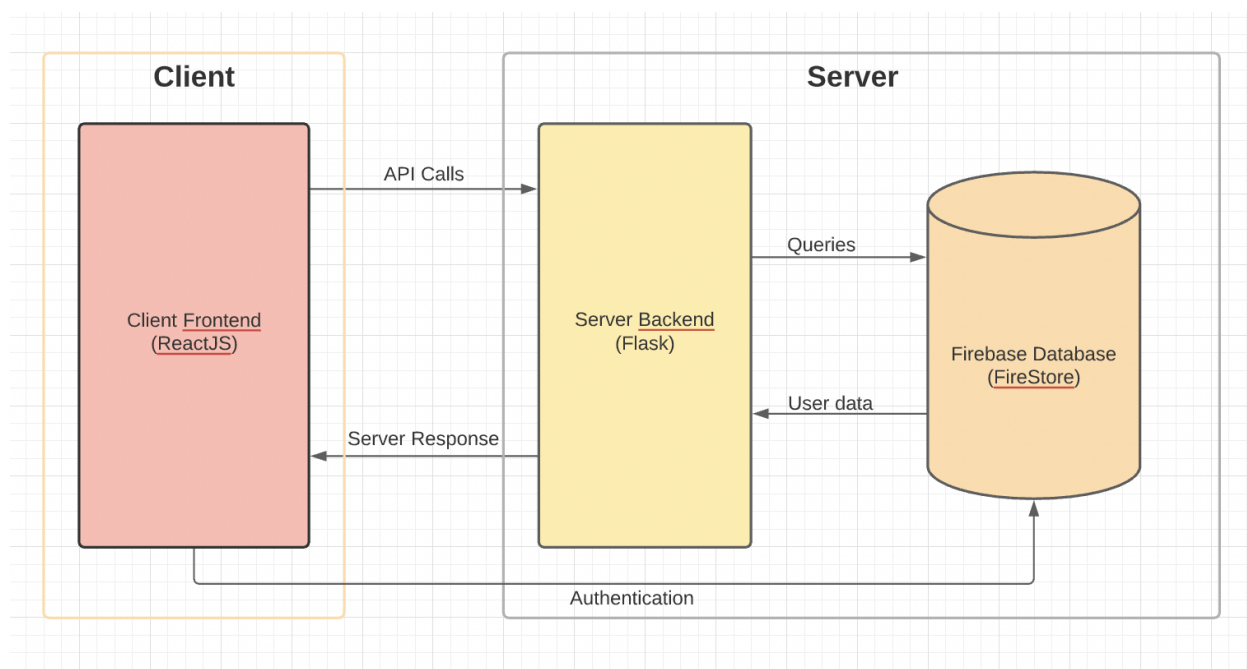
- Front End Architecture
 - As developers, we would like to create a responsive web app in ReactJs using Firebase so that we can authenticate and authorize users
 - As developers, we would like to request read permissions from the user's gmail
- Back End Architecture
 - As developers, we would like to store the user's access/refresh token in the Firebase's Nosql database
 - As developers, we would like to extract emails from the user's gmail
 - As developers, we would like to parse emails that have been extracted
 - As developers, we would like to store the master itineraries for each user
- Security
 - As developers, we would like to store user authentication credentials securely
 - As developers, we would like users to verify their data in sign up and sign in forms
 - As developers, we would like to prevent XSS and CSRF
- Usability
 - As developers, we would like an intuitive user interface

- As developers, we would like for users to easily be able to dig deeper than their itinerary
- Performance
 - As developers, we would like to consistently and quickly parse the emails of users

Design Outline

High-Level Design of Architecture

We are using the standard client-server architecture for MergeTrip. Users will be interacting with a frontend built with ReactJS. Operations completed on the frontend will be relayed to the Flask backend, which is a framework that can implement backend API operations. The backend will integrate with a Firebase server that will mostly be utilized for user authentication and information storage.



Frontend

Our frontend will be integrated with ReactJS, which is a JavaScript library used to build user interfaces. This will be the first thing that users see, and will serve as the face of the web application. All operations that require transferring or pulling travel

data from the user database will be done through the backend. The frontend will also be responsible for calling API's that we will use in our code. Because we are extracting travel data through email, we will be calling the Gmail API.

Backend

Our backend will consist of a Flask backend, which is written in Python. The extraction of emails will be done by the Flask backend that will communicate with the Gmail API to get access to the emails. The front end will request read permissions from the user's Gmail account and will hold the access/refresh token for the user's Gmail account and store it in Firebase's NoSQL Database (Firestore). Along with the access/refresh token for a user's Gmail account, we will also be storing the parsed travel data and travel itineraries for each user.

Design Issues

Functional Issues

- How will the user sign in?
 - Option 1: Firebase - using email and password
 - Option 2: Firebase - using gmail
 - Option 3: Manually managing JWT Authentication
 - Choice: Option 1
 - Discussion:

We chose option 1, to login with email and password using Firebase.

Firebase has a secure serverless authentication service that can be setup very quickly and has already built-in features that we do not have to implement from scratch, such as resetting passwords, changing passwords and email verification. On the free plan, we can support an unlimited amount of accounts and about 1000 concurrent requests, but we can upscale our usage plan at any time without any need for extra code for load balancing or other scalability issues.

We chose to use email and password through firebase rather than signing in with gmail through firebase, because we want to choose whether they want to link or unlink their gmail account from our service, in case they want to manually create Itineraries. Moreover, it will allow for flexibility on our part in the future, as we can link multiple email providers rather than being stuck with just gmails.

- How will we filter out the correct confirmation emails?
 - Option 1: Popular company emails
 - Option 2: Popular company emails, subject lines and dates
 - Option 3: Extract information from all emails
 - Choice: Option 2
 - Discussion:

We chose to filter out the correct confirmation emails using popular company emails (AirBnb, United, American Airlines, etc...), as well as the subject lines by checking whether the emails are confirmation emails and not advertisements or policy change emails. The date will also help limit the number of emails we have to scan to speed up the process.

- How do we extract the confirmation numbers, due dates and locations from emails?
 - Option 1: Regex
 - Option 2: NLP/RNN Model
 - Choice: Sprint 1: Option 1, Others; Option 2
 - Discussion:

For the first sprint, we will extract the confirmation numbers, due dates and locations from emails using regex based on the format of the popular company emails, however this limits extracting to only popular emails such as AirBnb, United, etc. For the other sprints, we plan to build a Natural Language Processing model for extraction for all confirmation emails and not just popular emails.

Non Functional Issues

- Should we use a web or mobile app?

- Option 1: Web
- Option 2: Mobile/iOS
- Choice: Option 1
- Discussion: We chose option 1, to develop a web app, because more team members had prior more experience with web development compared to iOS. Additionally, a web app will be able to be accessed by users on their phones and they have the ability to pin the web app to their home screen.
- What language and framework should we use for the front end?
 - Option 1: InfernoJs
 - Option 2: ReactJs
 - Choice: Option 2
 - Discussion: Our group has more experience working in React than any of the other alternatives that we considered. Due to this, we decided that it would be in the best interest of our time that we use the React library rather than all try and learn an alternative.
- What platform should we use for the back end database?
 - Option 1: Firebase
 - Option 2: AWS
 - Choice: Option 1
 - Discussion: We chose to use Firebase because it just fit our needs better than AWS. While they both offer similar services, Firebase offers a real time database which means that user information and their itineraries would update more quickly. Additionally, Firebase is free and the server

can handle up to 1000 concurrent requests before we need to pay to upgrade the service. Overall, Firebase seems like the cheaper and easier to manage option.

- What should we use for user authentication?
 - Option 1: Azure
 - Option 2: Firebase
 - Choice: Option 2
 - Discussion: Since we are using Firebase for our databases, it should be easier to just also use it for user authentication. Additionally, since user accounts only require an email and password to login, it will be easy to implement Firebase for user authentication.
- What machine learning framework should we use?
 - Option 1: Pytorch
 - Option 2: TensorFlow
 - Choice: Option 1
 - Discussion: While you are able to do more on TensorFlow compared to PyTorch, PyTorch has a shorter learning curve compared to TensorFlow. One of the main reasons we considered using TensorFlow was that there is a larger community around TensorFlow because PyTorch is a newer framework. However, the PyTorch community is sizable and growing rapidly so we feel that this is a small consideration and have decided that the easier to learn framework is the better option.

Classes:

Users

- Contains user information including username, password, email, **etc**
- Able to create different Trips
- Connected to a Trip by default

Trips

- Default object given to the user to plan Trips within
- Contains different reservation objects
- The user can interact with the Trip to manually add reservation objects

Reservations

- Contains within the Trip Object
- Object contains information regarding individual reservations in a calendar
- Includes type of reservation. There are two types of reservations: Flight and Non-Flight.

Flight Object

- Connected to the Reservation object
- This object contains all the information necessary for flights

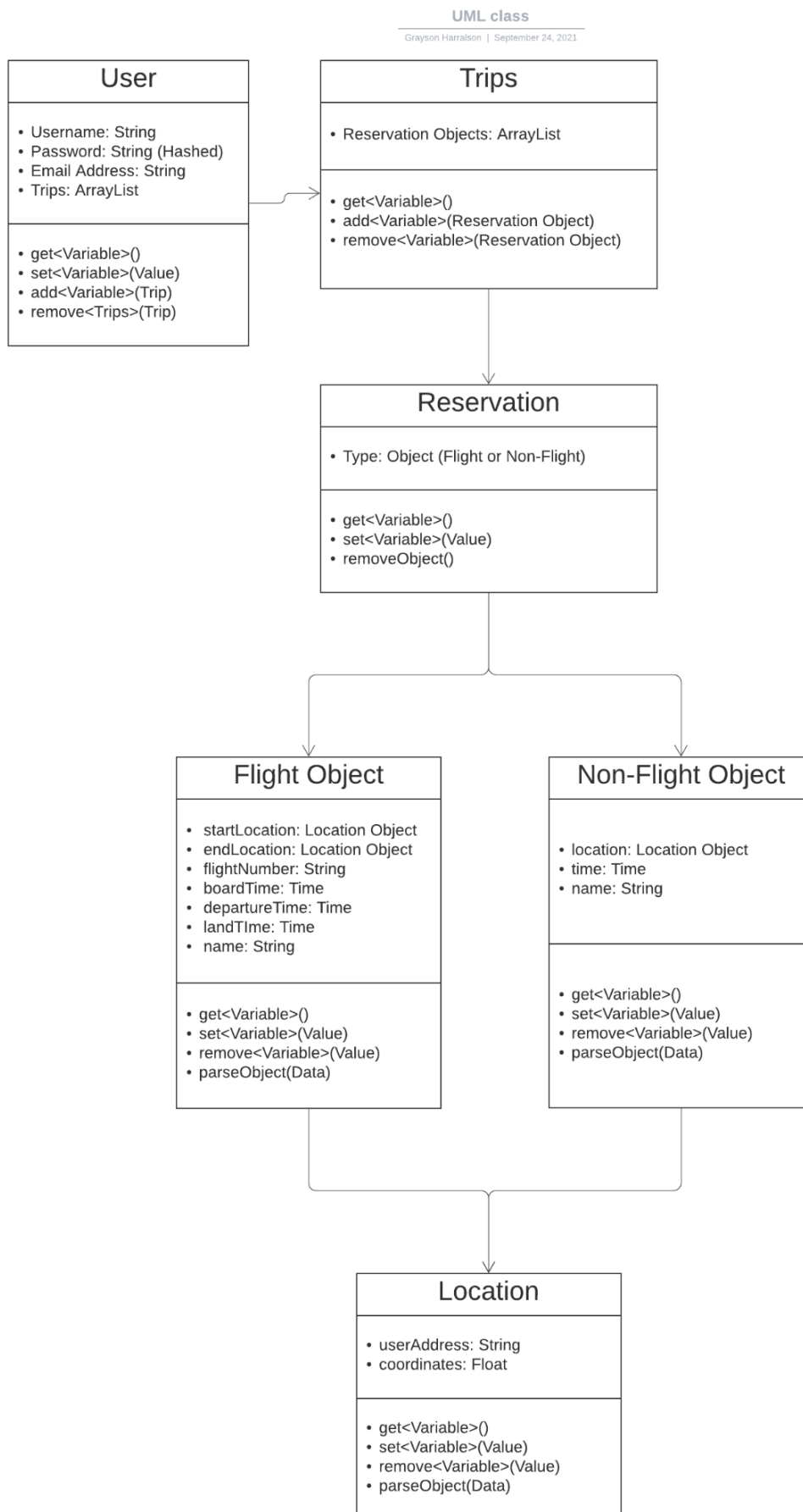
- Contains flight numbers
- Has two location objects attached
 - Starting airport
 - Ending airport
- Contains boarding time, departure time, and landing time
- Name of flight company

Non-Flight Object

- Connected to the Reservation object
- This object contains all the information for a typical reservation
- Has at a location object attached
 - Location of the reservation
- Contains time of reservation
- Name of company/reservation

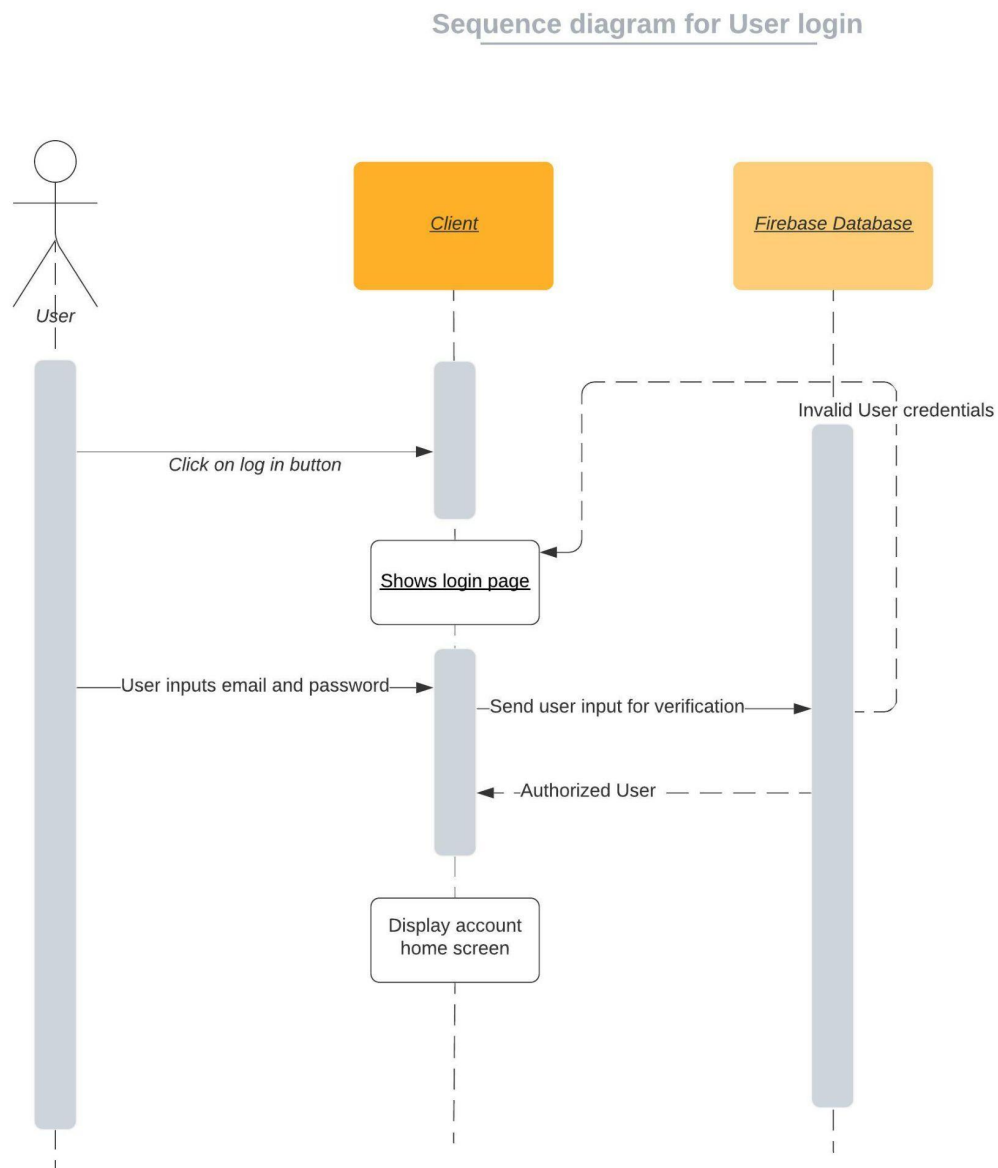
Location

- Connected to the reservation Object
- Contains a readable by user address and precise GPS coordinates

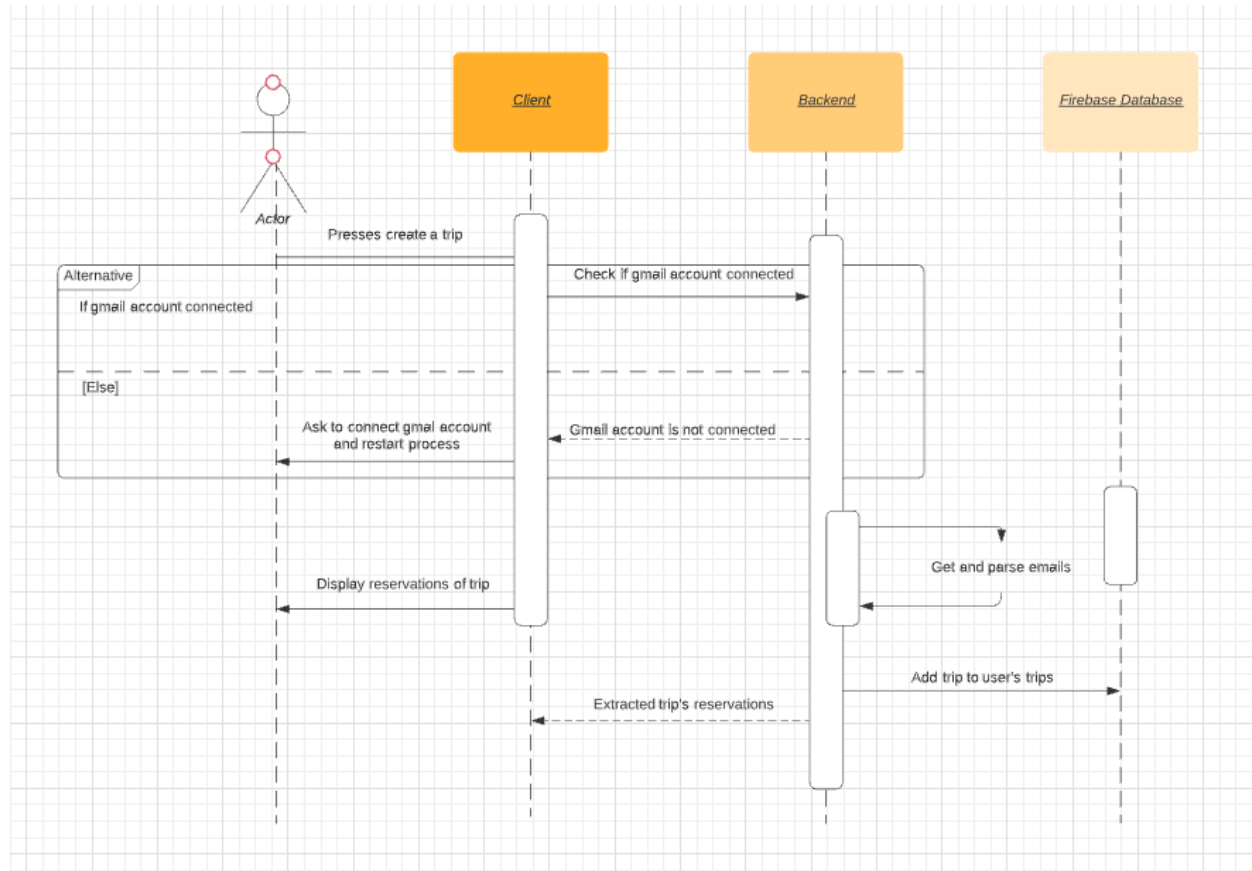


Sequence Diagrams

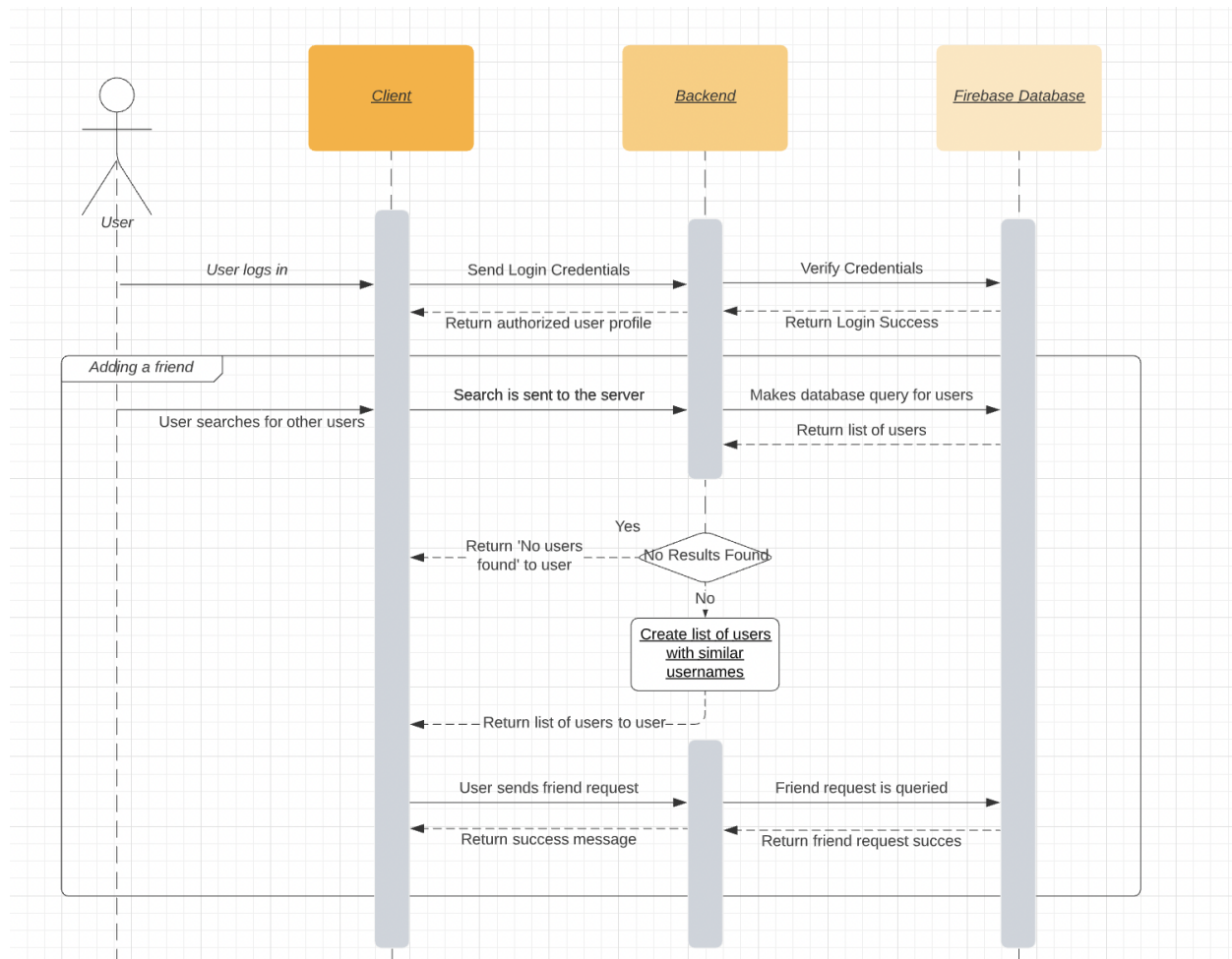
1 - Sequence Diagram for User Login



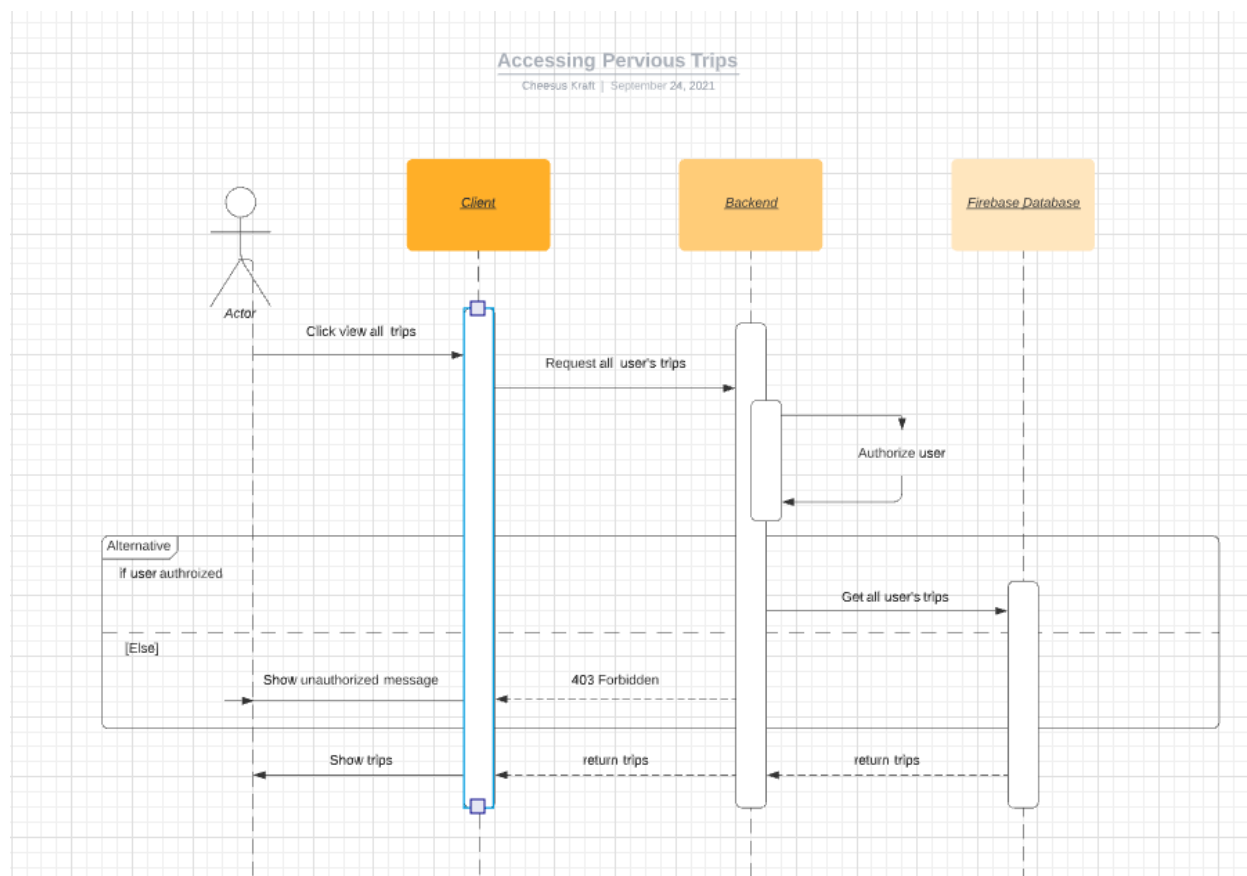
2 - Sequence Diagram for creating a trip



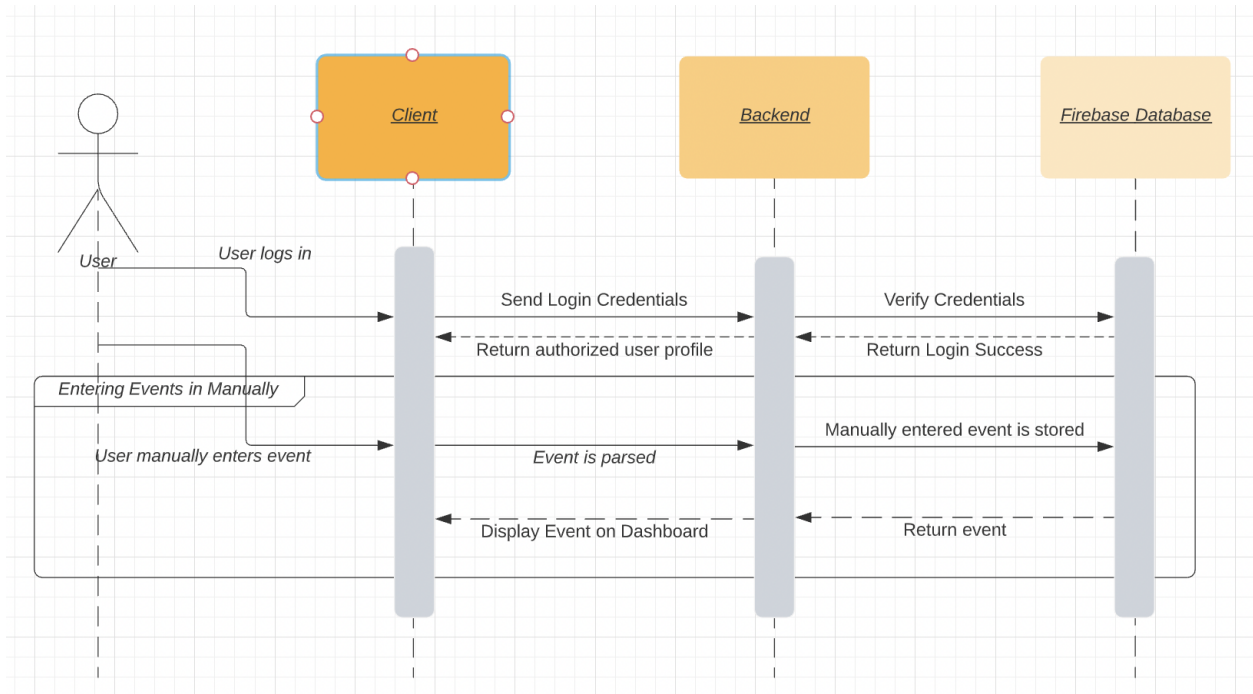
3 - Sequence Diagram for Adding a Friend



4- Sequence Diagram for accessing previously created trips




5 - Sequence Diagram for manually entering events





UI Mockups

Home Screen




 **MERGETRIP**

Login 



MergeTrip can seamlessly create travel itineraries just by linking to your email

**See all of your travel details in one place.
Hotels, Flights, Rental Cars, Parks,
and much more**



Usage Flow



Account 

To find all of your confirmation emails
MERGETRIP needs access to your Gmail.

Login with Gmail 

We will never sell or share your data.



Account 

Create a New Trip

Trip Title

Location

Browser Date
Picker



Create



Account

Trip to Washington DC

09/21 - 09/28



United Airlines - Flight UA255

Indianapolis to Washington
Departs at 3:50 PM, Boarding Begins 3:00 PM

[Boarding Pass](#)

[Directions](#)



Hertz Car Rental - SUV

Washington (IAD) Airport
Pickup at 7:00 PM

[Confirmation](#)

[Directions](#)



Hilton Hotel

Washington (IAD) Airport
Check-In at 2:00 PM

[Confirmation](#)

[Directions](#)



Six Flags - Amusement Parks

Richmond, Virginia
Park Opens at 10:00 AM

[Tickets](#)

[Directions](#)



Account

Trip to Washington DC

09/21 - 09/28



United Airlines - Flight UA255

Indianapolis to Washington
Departs at 3:50 PM, Boarding Begins 3:00 PM

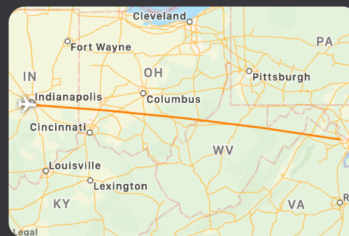
[Boarding Pass](#)

[Directions](#)


IND
Terminal B, Gate 10
3:50 PM EST


IAD
Terminal D, Gate 32
5:20 PM EST


Emraer 170
600 Miles
Duration: 1hr, 34 min
Baggage Claim: D



Account/Settings Page

 **MERGETRIP**

Account 



Kunwar Sahni
Account Created
on 09/21/21

Change Email

Change Password

☒ Some Check Mark Setting

☒ Some Check Mark Setting

Download Data

Delete Account