Team 2 Design Document

Annalycia Melendez, Cody Schroeder, Dylan Mahan, Quinn Conrad

Table of Contents

Table of Contents 1
1. Purpose 3
1.1 Problem Statement 3
1.2 Requirements – Functional 3
1.2.1 Accounts 3
1.2.2 Events & planning 4
1.2.3 Interactive Map 4
1.2.4 Avatar customization 5
1.2.5 Functionality 5
1.3 Requirements – Nonfunctional 5
2. Design Outline 7
2.1 Design Decisions 7
2.1.1 Client 7
2.1.2 Server 8
2.1.3 Database 8
2.1.4 Testing 8
2.2 System Interactions 9
3. Design Issues 10
3.1 Functional 10
3.1.1 How Can We Let Friends See Each Other's Locations 10
3.1.2 How should pin removal be handled 10
3.1.3 What is the extent of avatar customization 10
3.1.4 What type of layers should be toggleable on the map 11
3.2 Non-functional 11
3.2.1 What language should the backend be created in 11

3.2.2 What database system should be utilized 12
3.2.3 How can we protect our system from spamming attacks 12
3.2.4 What type of software can be utilized for the map 12
4. Design Details 13
4.1 Class Diagram 13
4.2 Class Descriptions & Interactions 13
4.2.1 User: 13
4.2.2 Location: 14
4.2.3 Event (Extends Location): 14
4.2.4 Edge: 14
4.2.5 IconFeature: 15
4.2.6 Backend: 15
4.3 Sequence Diagrams 16
4.3.1 Log-in Sequence 16
4.3.2 Create an Event 17
4.3.3 Purchasing a Feature 18
4.3.4 Map Update Loop 19
4.4 Activity Diagrams and UI Mockups 19
4.4.1 Navigation Flow Map 19
4.4.2 Activity Diagrams 20
4.4.3 UI Mockup 20

1. Purpose

1.1 Problem Statement

With the ever-growing chaos of our university, planning around the unpredictability of campus activity has never been harder. Between finding adequate parking, getting to dining halls, and finding half-hour-long lines with no seating, or trying to get a quick gym session in before class, only to see what seems like everyone fighting for the machines at the same time, one must plan ahead or inevitably be late to something. It's safe to say students have better things to be thinking about, which is why we propose an all-in-one crowd-sourced application to keep in check with the current campus activity and make planning easier. Our application will provide relevant information all in one place so that students don't need to have apps like Waze, Calendar, Messages, and Purdue dining all open when trying to coordinate activities, and will also provide a shared calendar feature to make planning social events with others less of a headache.

1.2 Requirements – Functional

1.2.1 Accounts

1. As a general user, I would like to be able to make an account.
2. As a general user, I would like to be able to log in.
3. As a general user, I would like to be able to change account details from the Account page. (Username, Password, etc.)
4. As a general user, I would like to be able to delete my account.
5. As a general user, I would like to be able to change my username/password on the login page from an email if forgotten.
6. As a general user, once logged in, I would like to be able to access the Account, Schedule, and Interactive Maps pages.

7. As a general user, I would like to be able to send friend requests to other accounts.
8. As a general user, I would like to be able to accept or deny friend requests from other accounts, as well as be able to unfriend users.
9. As a user, I would like to be able to report accounts.
10. As an admin, I would like to be able to ban accounts for misconduct.

1.2.2 Events & planning

11. As a general user, I would like to be able to schedule events.
12. As a general user, I would like to be able to share events with friends.
13. As a general user, I would like to be able to add locations to events.
14. As an event coordinator, I would like to be able to communicate with event attendees.
15. As a user, I would like to be able to subscribe to and unsubscribe from events.

1.2.3 Interactive Map

16. As a general user, I would like to be able to see my location on the map.
17. As a general user, I would like to be able to select a location on the map and see details about it.
18. As a general user, I would like to be able to navigate the map and snap back to my current location.
19. As a general user, I would like to be able to search within the current location-base.
20. As a general user, I would like to be able to place a pin on the map at a specific location.
21. As a general user, I would like to get directions from my current location/a specified location to my current location/a specified location.

22. As a general user, I would like to be able to toggle different map layers (Tunnels, sidewalks, etc.).
23. As a general user, I would like to find nearby vending machines and restrooms.
24. As a commuter, I would like to be able to select my mode of transportation.
25. As a general user, I would like to report blocked paths and closed locations.
26. As a general user, I would like to report a busy area.
27. As a general user, I would like to be able to report temporary POIs (food trucks, events, etc.).
28. As an admin, I would like to be able to add and remove locations.
29. As an admin, I would like to be able to edit location details

1.2.4 Avatar customization

30. As a user, I would like to see my avatar on the map.
31. As a user. I would like to be able to customize my avatar.
32. As a user, I would like to earn currency from reports for customization purposes.
33. As a user, I would like to be able to spend currency for extra customization options.
34. As a user, I would like to be able to view the avatars of other users.
35. As a user, I would like to be able to toggle my location visibility for other users.

1.2.5 Functionality

36. As a user, I would like to be able to report bugs and/or crashes.
37. As a user, I would like to add suggestions for new locations and paths.

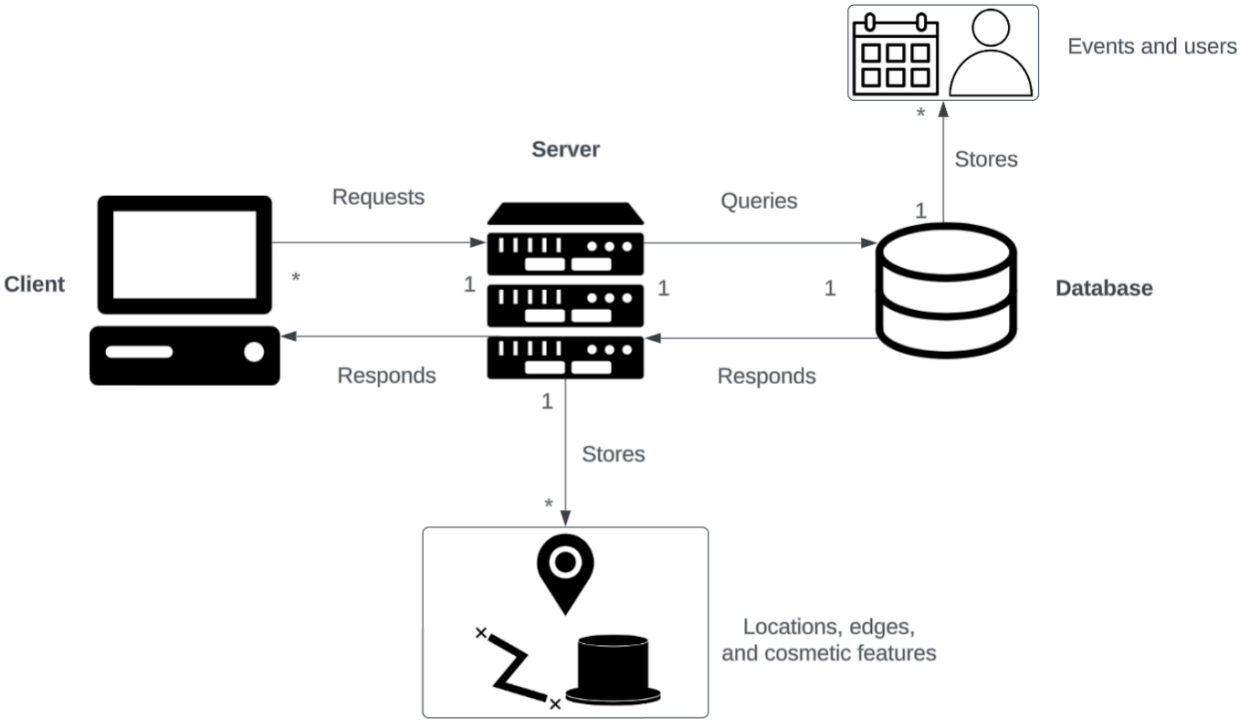
1.3 Requirements – Nonfunctional

1. As a developer, I would like to be able to securely store sensitive user data.
2. As a user, I would like pathfinding to be performed in a reasonable time.
3. As a user, I would like the app to have virtually no lag despite user counts.
4. As a user, I would like the app to be functional 24/7.
5. As a developer, I would like the app to be free from bot accounts.
6. As a developer, I would like the app to be safe from spam-based attacks
7. As a developer, I want database queries to be safe from injection.
8. As a developer, I want to maintain a separate front-end and back-end.

2. Design Outline

2.1 Design Decisions

Odyssey is a mapping & event planning application which follows the client-server model for centralized management. The server handles queries to the database and serves the data back to the client for display on the GUI, as well as implementing the algorithm for pathfinding.



2.1.1 Client

- Implemented in Python
- Uses the Folium library for map customization
- Interprets response from the server to display to GUI

2.1.2 Server

- Implemented in C++
- Receives & processes requests from the client
- Fetches data from database
- Uses pathfinding algorithm
- Authenticates logins

2.1.3 Database

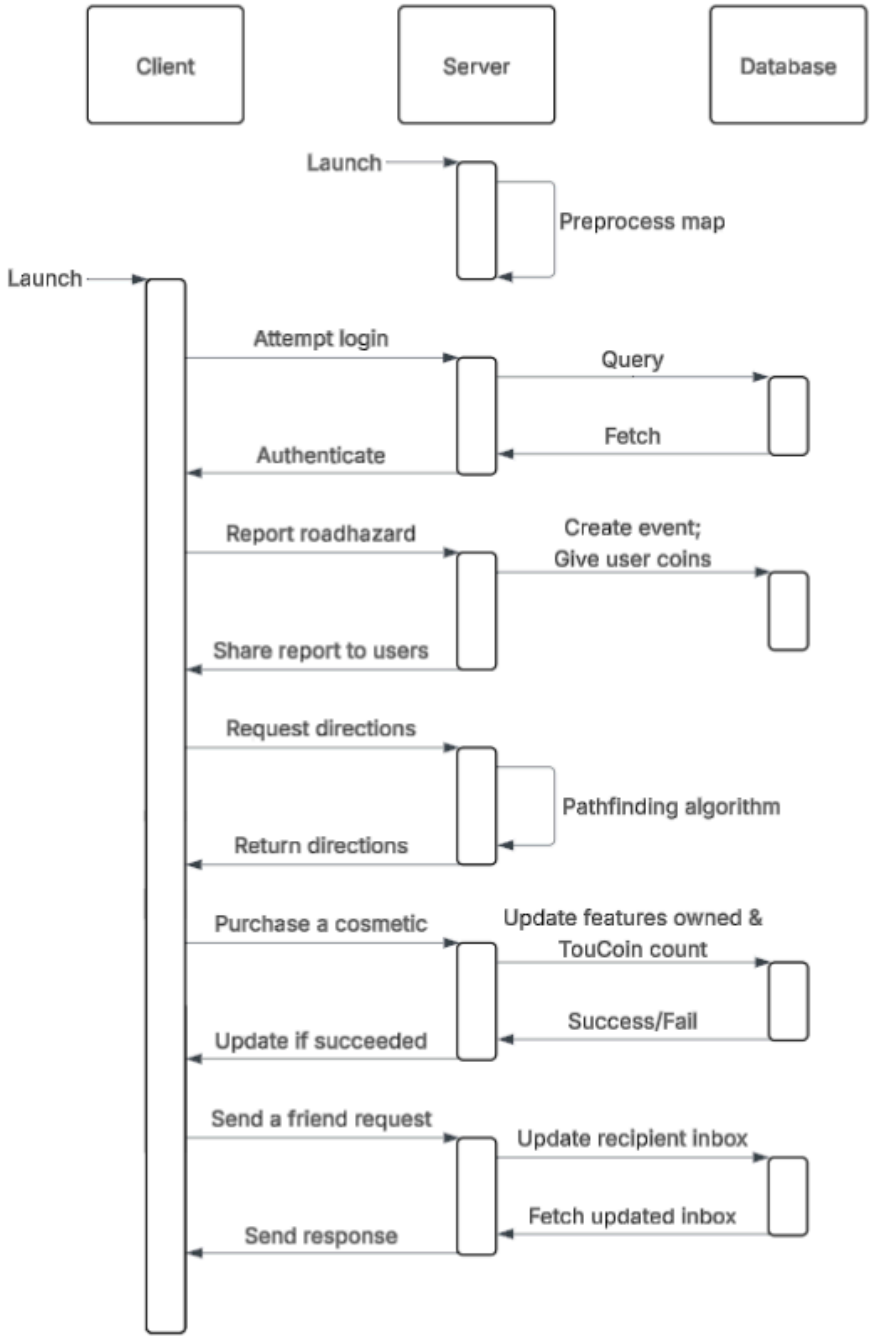
- Implemented in NoSQL
- Creates, reads, updates, & deletes data
- Responds to server

2.1.4 Testing

- Front-end testing implemented in Java using JUnit
- Back-end testing implemented in C++ with GoogleTest

2.2 System Interactions

Below is an example of how the client, server, and database communicate as a user goes through the application.



3. Design Issues

3.1 Functional

3.1.1 How Can We Let Friends See Each Other's Locations

- Option 1: Have a Friends Id list compared against online user ids and show any that match
- Option 2: Don’t
- Option 3: Tie user profiles together and show the corresponding location date to the tied profiles
- Choice: Option 1
- We chose Option 1 because it better integrates with our current design structure and is the most efficient way to make friends visible to each other. This also fits with our current planned database structure and how we hope to link together our objects.

3.1.2 How should pin removal be handled

- Option 1: Make pins temporary with a fixed expiry time
- Option 2: Allow users to remove pins at their own convenience
- Choice: Option 2
- We chose Option 2 because it seems to be the best way to improve the user experience, as users can make their pins as temporary or permanent as they please. This also creates a much more customizable environment and experience for the user.

3.1.3 What is the extent of avatar customization

- Option 1: Choose from a set of profile pictures
- Option 2: Allow personalizable rings surrounding the profile image

- Option 3: Allow the user to choose between a set of hats, a set of avatar bodies, and a set of items for the avatar to hold
- Choice: Option 3
- We chose Option 3 as we feel it gives the most choices of customization to the user without sacrificing the quality of our system, as well as adding a leading in for our shop where users can use Toucoins to buy new items, hats, or bodies.

3.1.4 What type of layers should be toggleable on the map

- Option 1: Events and Locations
- Option 2: Tunnels, Bus routes, Bike routes, Walking routes
- Choices: Option 2
- We chose option 2 because having locations and events in fixed non-togglable positions seemed like the best way to provide map details, as well as tunnels, bus routes, bike routes, and walking routes, being alternative paths to destinations, it seems best to have them be separately laid out on the map

3.2 Non-functional

3.2.1 What language should the backend be created in

- Option 1: C++
- Option 2: Python
- Option 3: Java
- Choices: Option 1
- We chose this option as we have the most experience with this language, and this option integrates well with the map software we are going to utilize. All we need to use the map software all we need to include is a Python to C API, which is a simple process.

3.2.2 What database system should be utilized

- Option 1: MySQL
- Option 2: Firebase
- Choices: Option 2
- We chose Firebase as we determined that a NoSQL database would be best for our data structures and our program design. After looking into Firebase in particular, it seems well structured to fit our application model.

3.2.3 How can we protect our system from spamming attacks

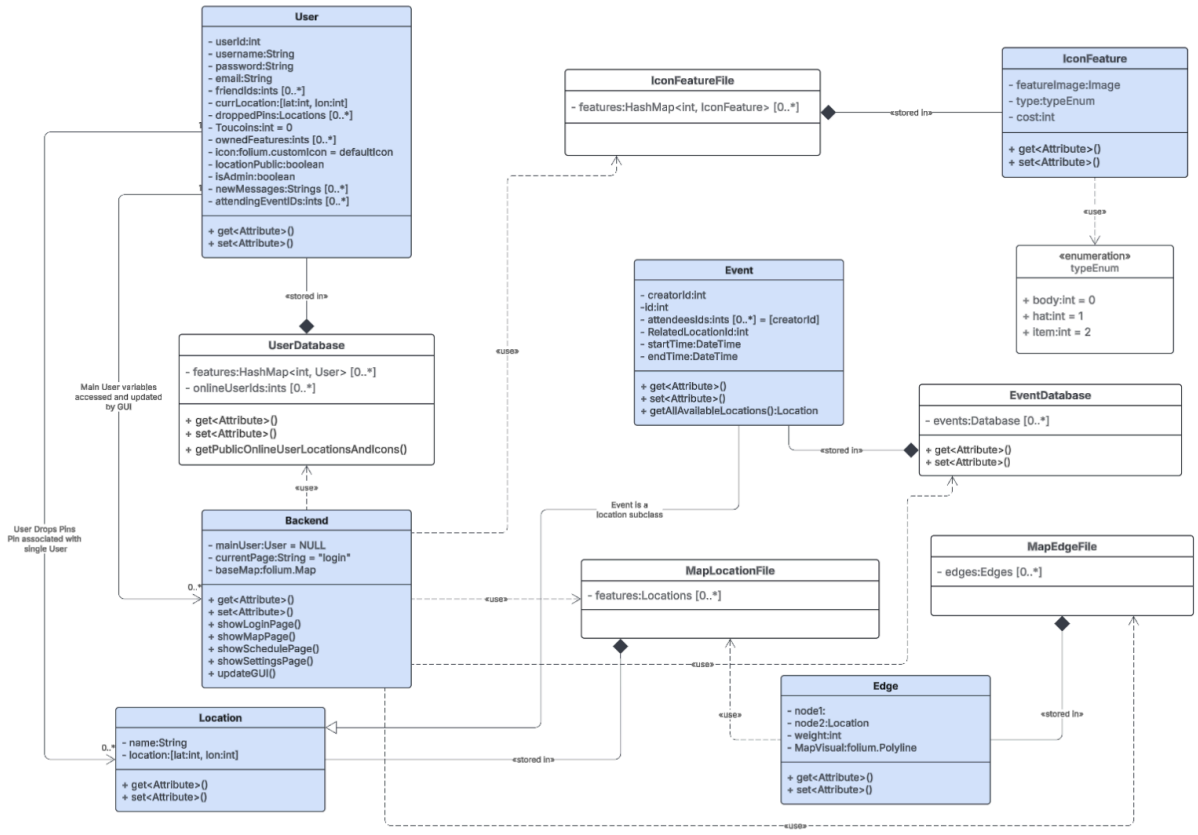
- Option 1: Use a counting system and suspend accounts that go over a set amount
- Option 2: Use an entry cooldown with a diminishing amount of Toucoins for each entry
- Choices: Option 2
- We chose this option because this seemed to be the most stable way to implement a spam protection system while still maintaining accurate hotspot tracking and while still maintaining a pleasant user experience.

3.2.4 What type of software can be utilized for the map

- Option 1: Folium
- Option 2: Google Map API
- Option 3: OpenLayers
- Choices: Option 1
- We chose to go with this option primarily due to its ability to communicate with C++ in a seamless fashion, and its pathing system worked well with our project design and our planned implementations.

4. Design Details

4.1 Class Diagram



4.2 Class Descriptions & Interactions

4.2.1 User:

- This is the basis for the program, each user of the program will have a corresponding User object with their information.
- User's current location is updated when online, and is used in the map
- Users can create events and drop pins on the map. These are stored in their User object.
- Users will also receive "Toucoins" which they can buy Icon customization features with. Owned features are referenced by IDs in the User object

4.2.2 Location:

- Locations store basic location data, such as a name and coordinates
- This is used mainly for Map locations that will be basically static and kept in a file
- Also used to store dropped pins from a user in the User's pins array to be loaded on a map update

4.2.3 Event (Extends Location):

- An Event is a subclass of Location and therefore also has a name and location to be shown to its attendees on the map
- When created, the eventID will be added to the current user's attendingEventIDs list, and any added friends will have a message added to their messages list containing the eventID so that if they accept, the id will also be added to their attendingEventIDs list. Events are then stored in a database.
- Events also have a start and end time that can determine if it should still be considered or if they should be deleted
- Additional functionality might be added to have public pop-up events on campus for food trucks, outdoor activities, etc.

4.2.4 Edge:

- Stores two basic locations and a weight between them, as well as a folium PolyLine to show on the map when the edge is referenced in a route
- Used for pathfinding from one location to another
- Also could be used to show map layers like the tunnel system, bus system, bike routes, etc.
- Should be stored in a file and stay mostly static

4.2.5 IconFeature:

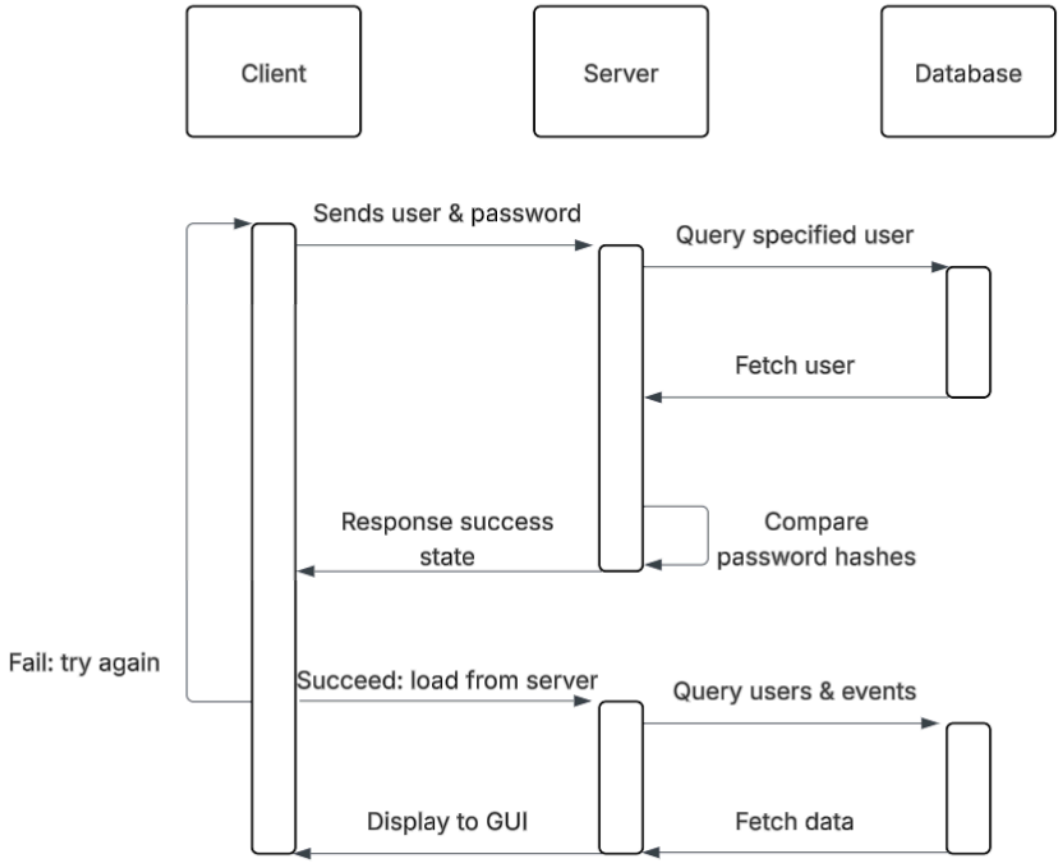
- IconFeatures are customizable add-ons to user icons. As users who have their location set to Public can show their customized icons, there's an incentive to get more of these features
- IconFeatures can be bought from a pop-up in the profile page, and will show up in the main user's owned features list
- IconFeature images should be able to be combined into an image, turned into a folium CustomIcon, and stored in the main User object when saving.
- IconFeatures are loaded in from a file on startup, and should be referencable by all users so that anyone can see them on their map

4.2.6 Backend:

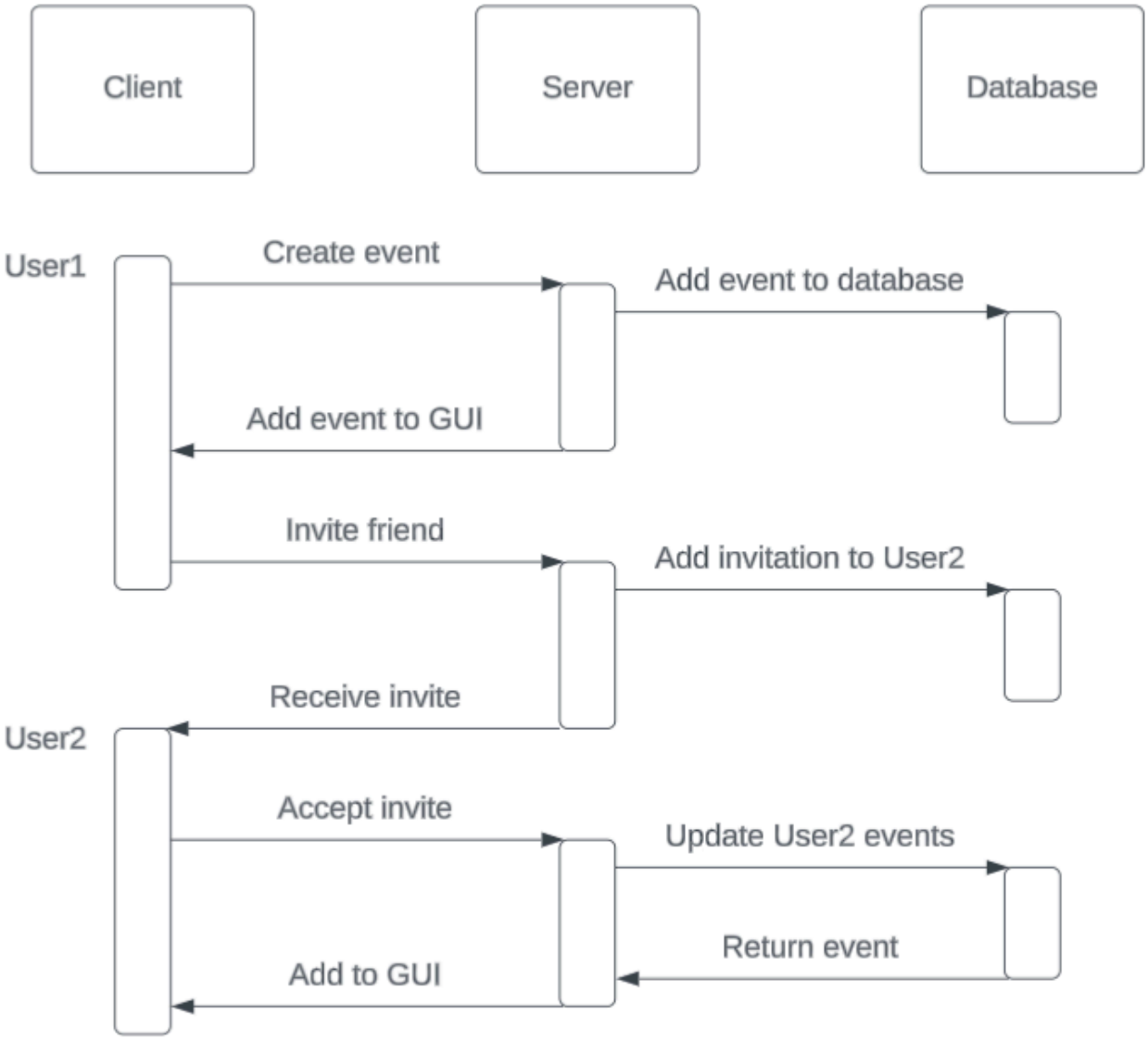
- Backend communicates with files and database(s) to save data and update the frontend
- On startup, Backend should generate a base folium Map from the mapLocations file and edge file. Then, after login (login credentials authenticated through the database), it updates the map to include the main user's pins and events.
- On the schedule page opened, it loads users' events and allows events owned by the current user to be edited, denoted by the eventOwnerID value in the event.
- On the Profile page opened, it loads the main user's profile details and writes changes to the user database on save

4.3 Sequence Diagrams

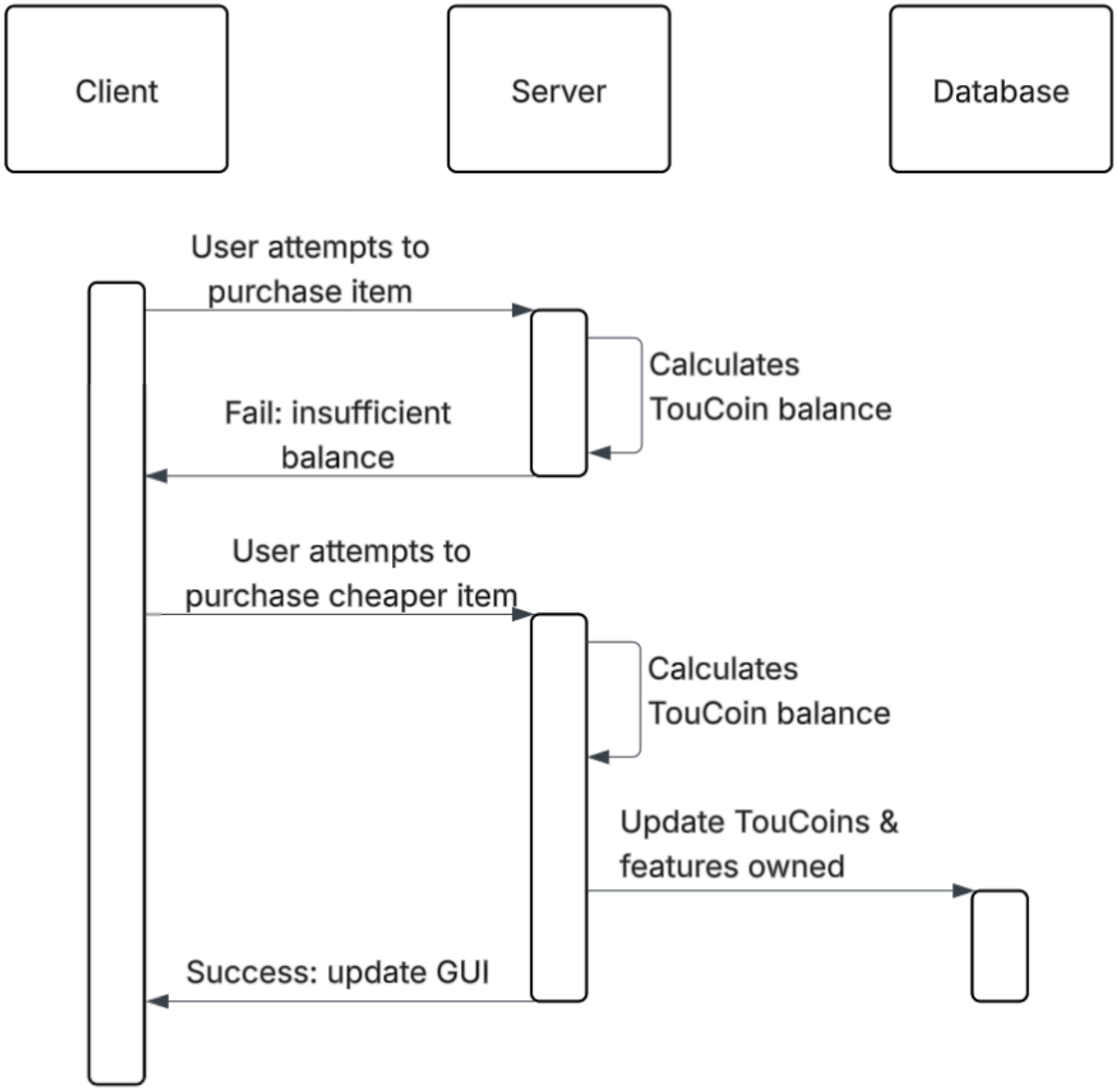
4.3.1 Log-in Sequence



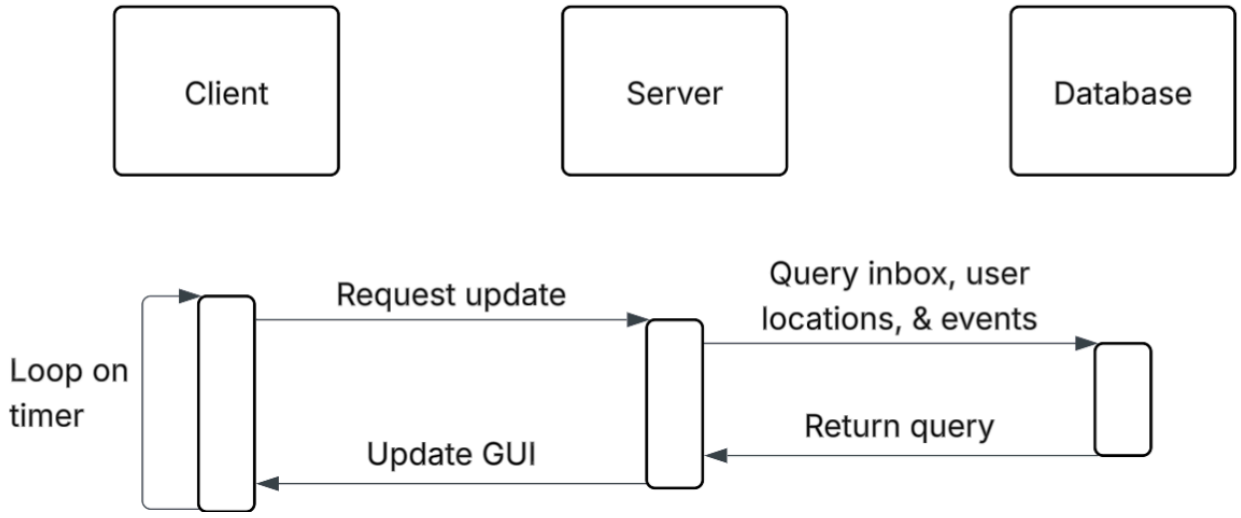
4.3.2 Create an Event



4.3.3 Purchasing a Feature

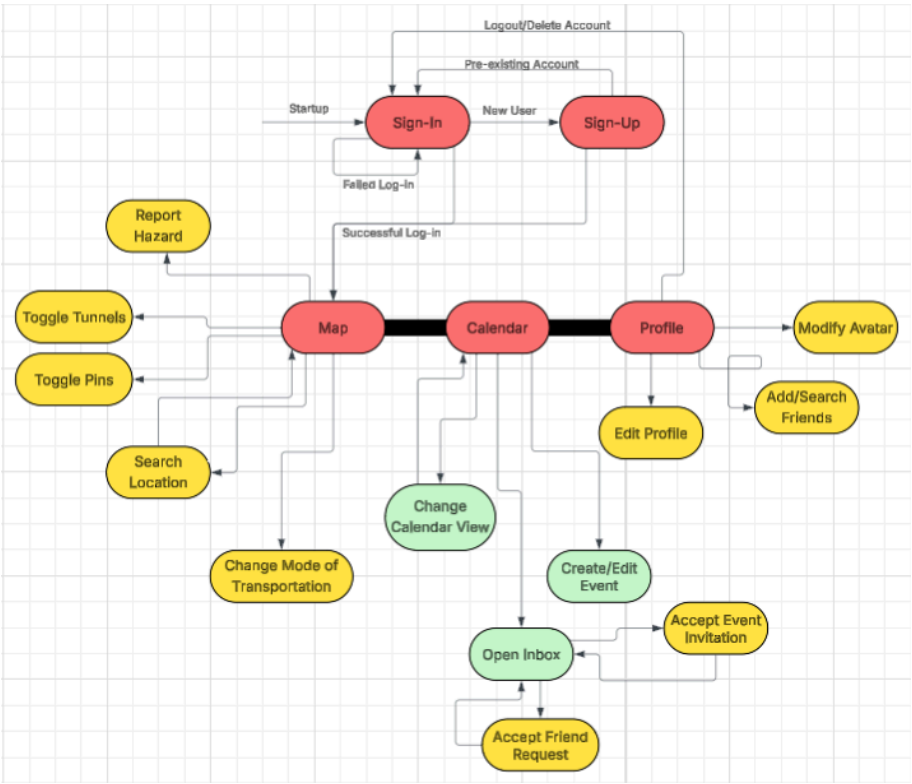


4.3.4 Map Update Loop

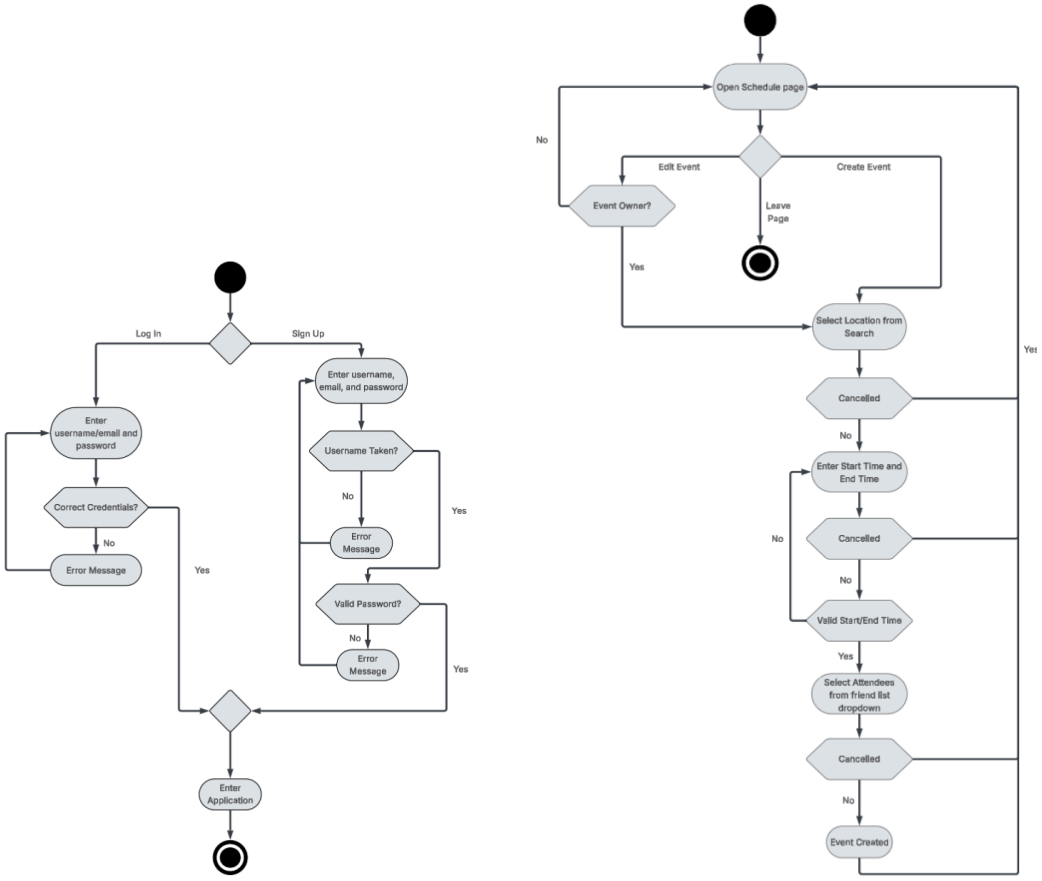


4.4 Activity Diagrams and UI Mockups

4.4.1 Navigation Flow Map

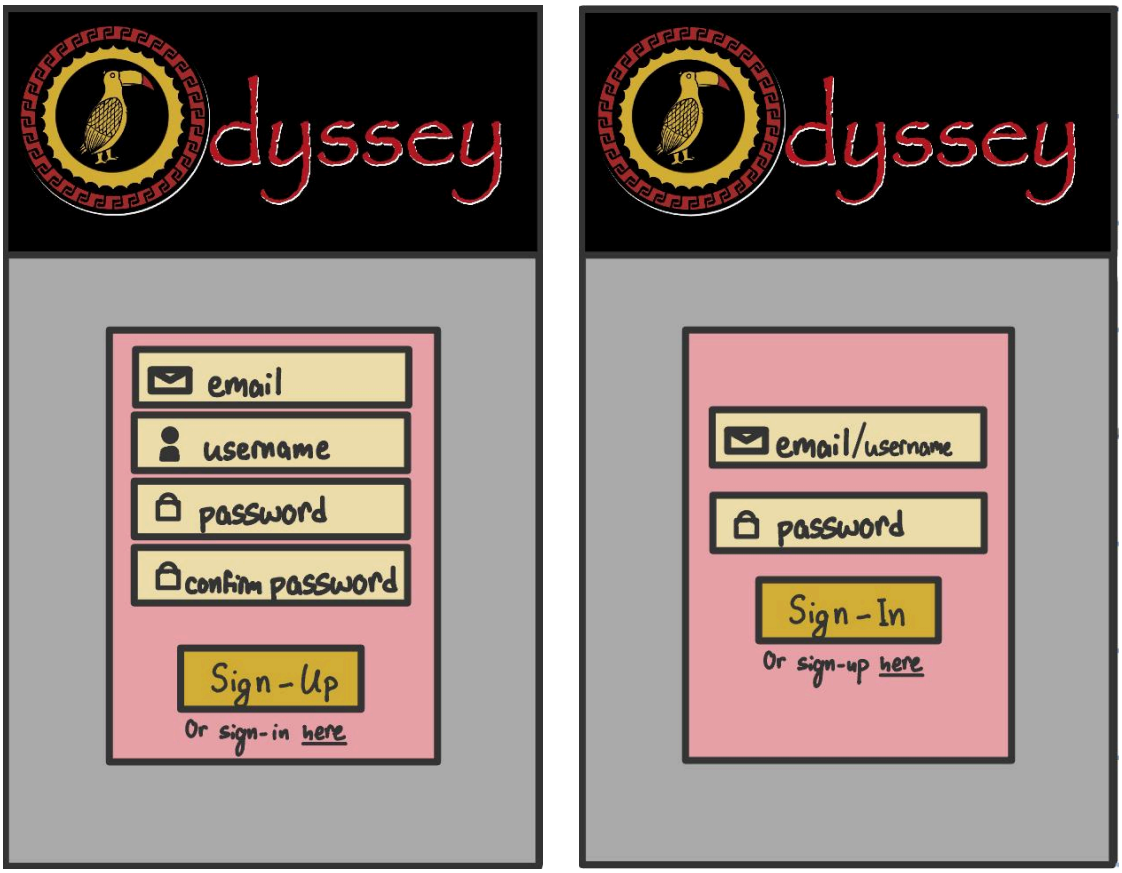


4.4.2 Activity Diagrams

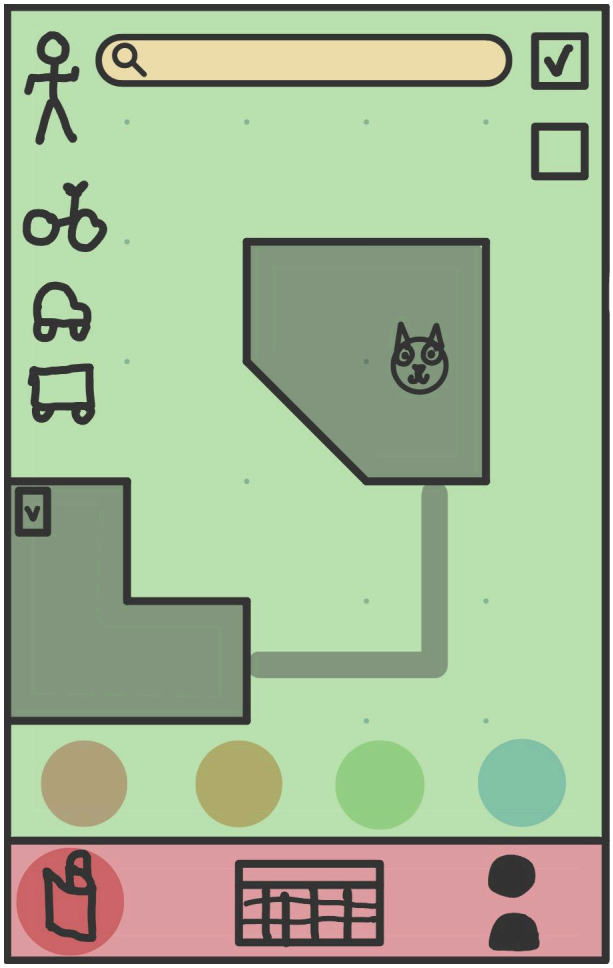


4.4.3 UI Mockup

We designed our application in a way to bring simplicity and ease to our user experience with Odyssey. The system for navigating between pages is on every page, allowing users to toggle between their map, calendar, and profile at will. We intend to have our platform be compatible with mobile devices through a progressive web application, so we have created a design that is calibrated towards the mobile experience.

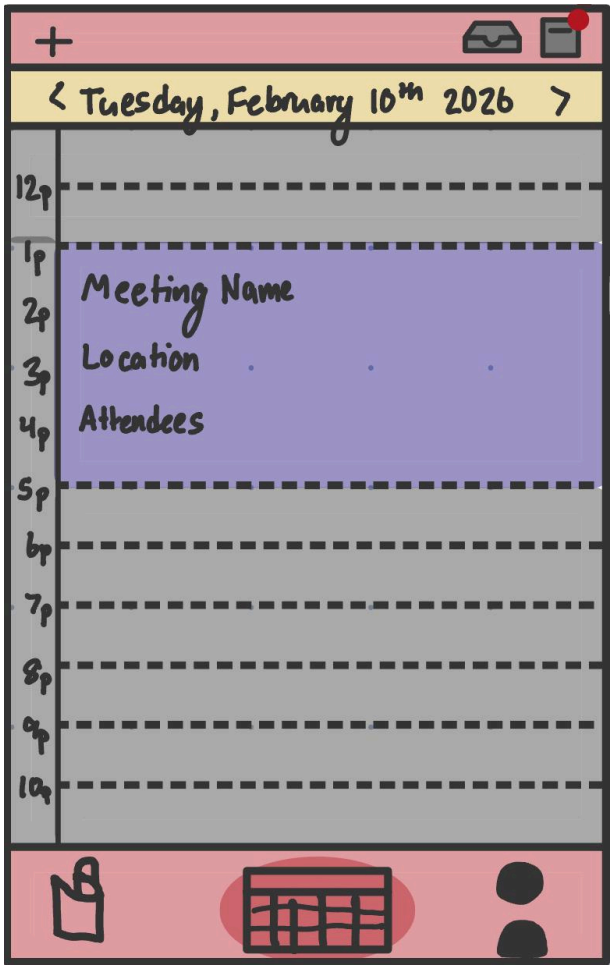


This is the first page that will come up when a user opens up the application, starting on the "Sign-In" specific page. The page displays the Odyssey app logo followed by a box to enter their email or username and one to enter their password. The user will be able to toggle which page they wish to be on by clicking the underlined "Here" text. If a user attempts to sign-in without an account, the app will prompt the user to sign-up instead. On the converse, if a user attempts to sign-up when an account already exists, they will be prompted to sign-in instead.



This is the map page, which is the main screen that the user will be on and the first one they will see when they log-in (or sign-in). The cat image in the center of the screen is the user's avatar, which shows their current location. This will be a customizable avatar that will be discussed more in the profile page section. The checkboxes in the upper righthand corner represent toggles for both the tunnel systems and pins. Currently, the tunnel system view is toggled, as can be seen by the tunnel between the two buildings on the map. The icons on the upper left side (person, bike, car, bus) represent the layers of the map, whichever mode of transport the user is looking for can be chosen and then seen on the map. The search bar at the top allows the user to search a location on campus and get directions to that location. The colorful buttons on the bottom of the screen allow users to make reports on locations, such as reporting that a place is busy or reporting a hazard. The bar at the bottom of the screen allows users to navigate

between the different pages, with highlighting on the map since they are currently on the map page.

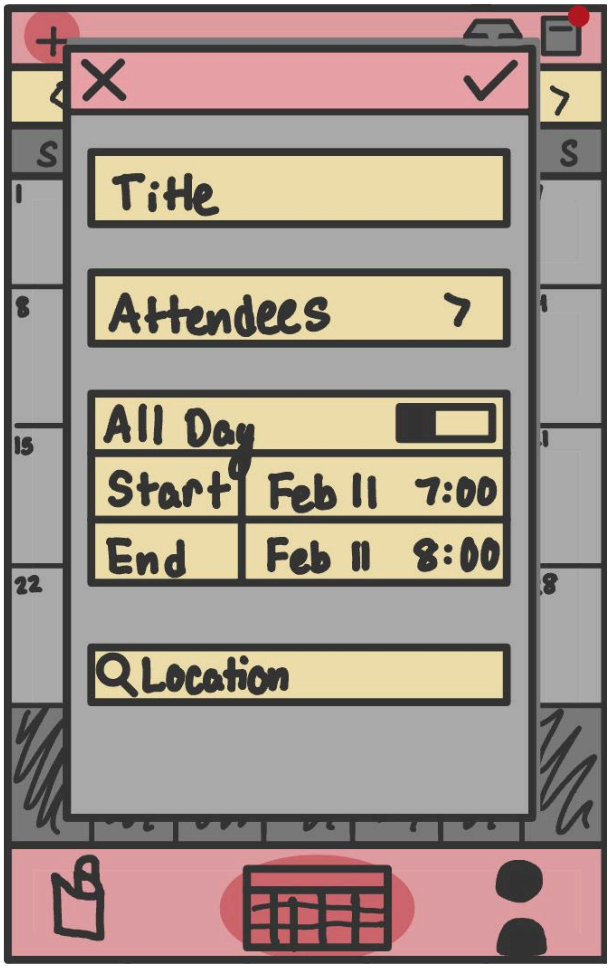


This is the main calendar page, which will show up when the user clicks on the calendar button in the bottom navigation bar. This is the day view of the calendar, if the user wishes to see the month or day view on the calendar, they can switch by clicking the button on the top bar, second from the right. The day view will show each hour of the day and any events that occur during that time. The events will show the meeting name, location, and attendees. The user can change days by clicking the arrows next to the day on the yellow bar. A new event can be created by clicking the plus sign on the top left corner of the screen. This will bring up an event creation pop-up, which will be described in more depth later. The user can check their inbox by clicking the top right button, this will bring up

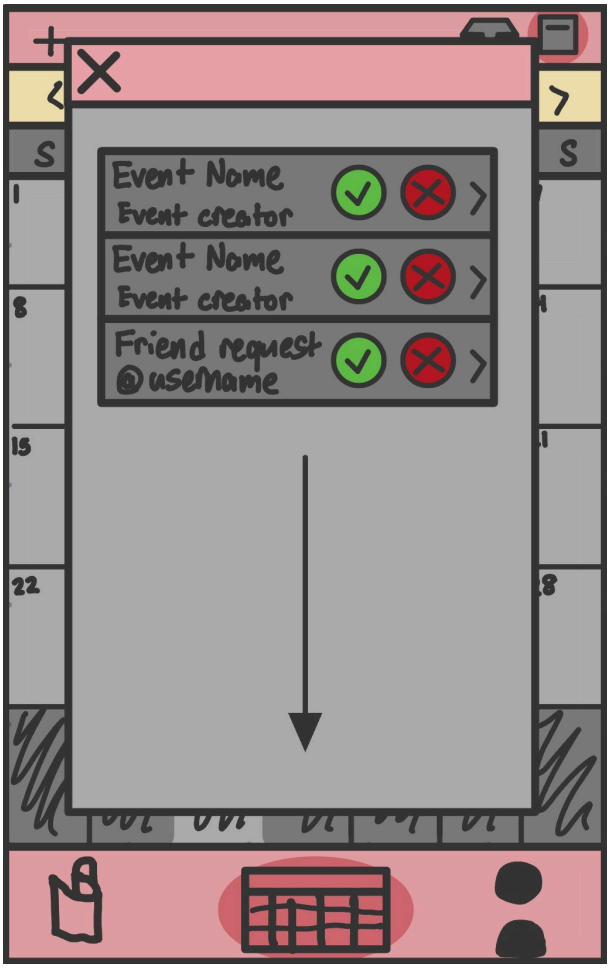
an inbox pop-up, which will also be discussed more later. The inbox will alert a user if it has anything in it with a red notification.

This is the month view of the calendar page. This view allows the user to see all events in the month, which only show the meeting name. The user can click on a specific day to see the day view of that day. The user can see different months by clicking on the arrows to the right and left of the month on the yellow bar.

This is the year view of the calendar page. This does not allow users to see events in the specific year, however the user can click on a month and change to that month's view. The user can switch to different years by clicking on the arrows next to the year on the yellow bar.



This is the event pop-up on the calendar page, which appears when the user clicks on the plus sign in the top left corner. This allows users to create a new event, specifying the name of the event, list of attendees, the date and time of the event, and the location of the event. The user has the ability to toggle whether the event is all-day or not. The user can confirm the event and send out invites by clicking the check mark on the right side of the pop-up, or they can scrap the event by clicking the X-mark on the left side. The user cannot click buttons outside of the pop-up while it is open.



This is the inbox pop-up of the calendar page, which appears when the user clicks the top right inbox button. The inbox will contain event invites and friend requests. The user can easily accept or deny these requests by either clicking the green checkmark or the red x-mark. The arrow represents the user's ability to scroll if they get too many invites. The user can click on an event invite to see more information about the event, such as the location, date/time, and other attendees. The user can exit their inbox by clicking the x-mark in the top left corner. On this screen, the user also cannot click buttons outside of the pop-up while it is open.

This is the user information page, which can be toggled by clicking on the person icon in the bottom navigation bar. This page allows for users to see and edit their information. The user is able to edit all items with a pencil next to them (e.g username, password, email, and profile picture), and they must save these changes by clicking the "Save" button on the bottom of the page. The user can also toggle their location settings to either public or private, allowing or disallowing friends from seeing their location. The user can search up friends to add using the search bar in the friends sidebar. This bar shows their current friends and allows them to request new friends. The top right corner of the page has the user's "TouCoins", which are the currency used in the Odyssey app to purchase new cosmetics. The user can change their avatar's outfit and buy new outfits by using the "dressing room" in the bottom right corner of the page (above the navigation bar). The arrows on the dressing room allow users to click through outfits that they own or ones that they wish

to buy. If a user clicks on the locks over the un-bought cosmetics, they are able to buy them using their TouCoins.