Jungle

Where we bring events to you

CS 307 Design Document - Team 17

Youssuf Elshall

Sutan Alahmadi

Albert Ibekwe

David Fuqua

Yaodong QIu

Table of contents

Purpose **3**

Functional requirements *4*

Non-functional requirements *7*

Design Outline **9**

High-level overview *9*

Components *10*

Design Issues **12**

Functional Issues *12*

Non-functional Issues *14*

Design Details **17**

Class Design *17*

Descriptions of the classes and the interactions between them *18*

Sequence Diagrams *20*

UI Mockups **23**

Purpose

College life is all about new experiences and activities. However, students have a hard time finding about what is happening in their college communities. Most events, clubs, and activities are scattered across many social media platforms and each one requires a different form or process in order to participate in. On the other hand, hosts such as student groups, clubs and university officials spend a lot of time posting and advertising on different platforms and dealing with each platform separately. Furthermore, the wide spread of their efforts ends up wasting time and resources as it does not reach a lot of students, and in some cases lowers student participation as the process can be confusing to many.

Our goal with this project is to develop an application that will connect the happenings around campus directly with students, and make it easier for hosts to add, manage, and connect with people. Current platform like Facebook do not provide the direct focus and connection between students and hosts, they are designed and focused on being an Ad supported marketing social-media platform rather than a way to connect college communities together. Furthermore, our application will provide machine learning recommendation based on users input and previous interactions using the app. In addition, it will provide you with event/club description, location, tracking, and a direct communication channel between hosts and students. Hosts will have the ability to add their events/clubs and manage them as well as manage their own event/club channels. On top of that, verification for both hosts and students will be implemented. So everyone can have more trust and authenticity.

Functional requirements

Student

A. Student Account

- As a student, I would like to sign up with my email
- As a student, I would like to sign up with my Google account (If time allows)
- As a student, I would like to sign with my Facebook account (If time allows)
- As a student, I want my personal information to be securely stored in the database and accessed through my login information only (email and password)
- As a student, I want to provide my personal details when I sign up (Gender, age, location, name)
- As a student, I want to choose from a menu the types of events I am interested in
- As a student, I want to have a settings page where I can change personal information and preferences (when swiping through events)

B. Swiping page

- As a student, I want to have a swipe page where I swipe left on events I am not interested and right on events I am interested in
- As a student, I want to see a photo, name, timing, location, and type of event in the swipe page
- As a student, I want to have a search tab where I can search for events using different methods (Search by general search, event/host name, hashtags)
- As a student, I want to have a navigation bar where I can move between tabs

- As a student, I want to click on an event to view more information on the events (Description, timings, locations, number of attendees, updates posted by hosts)
- As a student, I want to see more relevant events on my swiping page as the algorithm learns my preferences
- As a student, I want to filter events when swiping. Filter options are location, event types, age limits

C. Events and hosts details

- As a student, I want to see the events I swiped right on
- As a student, I want to chat to the event hosts through a chat button available on the event page
- As a student, I want to RSVP to an event through a button available on the event page
- As a student, I want to add the event to my calendar through a button available on the event page
- As a student, I want to be able to chat with a host through the event page button
- As a student, I want to be able to follow a hosts page
- As a student, I want to view a host's page through a button available on one of their hosted events page or by searching for them
- As a student, I want to be able to chat with a host through their host page
- As a student, I want to scroll through the hosts that I followed
- As a student, I want to see the trending events/searches inside the search section

D. Notifications

- As a student, I want to receive notification on events that I might be interested in from the recommendation algorithm
- As a student, I want to receive notifications on hosts/ organization/ clubs that I might be interested in from the recommendation algorithm (If time allows)
- As a student, I want to receive notifications on events that are coming up from the ones I am interested in
- As a student, I want to receive notifications on events that are coming up from the ones I RSVP'd to
- As a student, I want to receive notifications on updates to events I RSVP'd
- As a student, I want to be able to receive notifications from hosts that I followed about new events that they have
- As a student, I want to get notification on canceled events that I RSVP'd to
- As a student, I want to get notifications when I receive messages

Host

A. Host Account
   - As a host, I would like to sign up as a host and provide information about my organization or club
   - As a host, I want my profile to be verified and have it shown on my page (Like verified Twitter tick, for example) (If time allows)
   - As a host, I want my personal information to be securely stored in the database and accessed through my login information only (email and password)

B. Manage events

- As a host, I want to see my profile page where I can update and change information about my organization
- As a host, I want to be able to post events publicly and write all the relevant information (Name, location, description, photo, timing, event type)
- As a host, I want to change the information of previously posted events
- As a host, I want to push updates to events that are previously events
- As a host, I want to message attendees or message general messages (Separate into general and event specific messages)
- As a host, I want to cancel an event that was previously posted
- As a host, I want to see basic statistics about my events (Attendees per event, views on the events page, views on the host's page, follows, ratings)
- As a host, I want to limit the number of attendees

Non-functional requirements

A. Performance
   - Support 1000 concurrent users
   - Receive app startup information from server within 2 seconds
   - Get recommendations for event within 500ms
   - Provide 15 new recommended events from the recommendation algorithm every time the user logs on or runs out of recommended events
   - Support both iOS and Android (If time allows)

B. Server

- Provide cookies to the user as an authentication method (If time allows)
- Connect all the separate backend components into one coherent system

C. Security

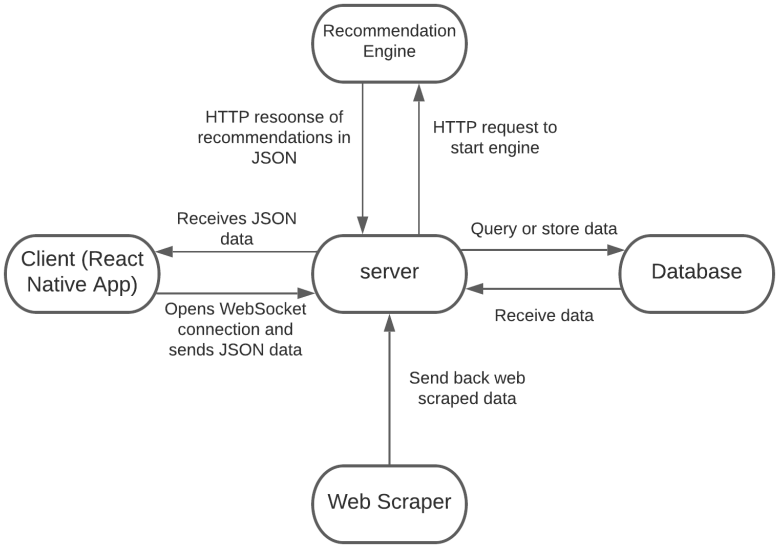
- Provide secure user authentication for logging in
- Authenticate every request from the user (If time allows)

Design Outline

Our project is mobile phone application that provides information to students about upcoming events based on their preferences, and allows hosts to promote and manage their events. This application uses an event-driven architecture to communicate between client and server. The server runs queries to the database, recommendation engine, and web scraper, in response to client requests.

High-level overview

The client forms a WebSocket connection to the server where they send and receive data. When the server needs to fulfill a request from the client it queries the database for the required information and sends it back through the socket. The client also requires the server to store information in the database. The recommendation engine is run when the client needs new events to be recommended to them, which is sent back to the client through the server. The web scraper is run periodically and stores the events collected from the internet inside the database. When the client logs out of the app, then the socket connection is closed.



Components

A. Client
    - Client includes the U/I where user's can interact with our system
    - Client opens a WebSocket connection with our server in order to send and receive events (requests) from the server
    - Data received from the server is presented to the client through the app
B. Server
    - Server receives requests from client and runs queries to other backend components
    - Server accepts WebSocket connection from client and begins to send and receive requests
    - Server will query the database for required information by the client or recommendation engine
    - Server will query the recommendation engine to run when needed by the client
C. Web Scraper
    - Web scraper gathers information from multiple websites about upcoming events
    - Web scraper filters the gathered information and stores it in the database
    - Web scraper is run periodically to make sure information is up-to-date
D. Recommendation Engine
    - Recommendation engine provides recommendation to users about events they might like and is sent back to them (through the server) to be displayed on the swipe page

- Recommendation queries database (through the server) to get up-to-date about the user's activities

E. Database
   - Database stores all information that is required by our application
   - Data stored in the database include client information (Student or Host) and event information

Design Issues

Functional Issues

A. Should we implement location-based event matching?

- **Option 1**: Build a scalable database for different regions and modify the recommendation algorithm
- **Option 2**: Maintain a central database that only contains Purdue and Lafayette events

We decided to keep the application on a small scale only local to Purdue, since it would require a much more complicated to implement location-based event matching anywhere. This would require an extremely complex web scraper that can extract information from any website. Also, the recommendation engine will need to modified to accommodate for different locations, and this is beyond our scope of knowledge and abilities. The application will only be limited to Purdue events.

B. How should we implement the chatting option for events?

- **Option 1**: Implement a group chatting functionality where you can chat with everyone that is interested in the event
- **Option 2**: Implement a chat where you only talk to the host regarding this specific event

We decided to go for the second option because we feel that will be a better user experience. Implement a group chat might be uncomfortable for some people and we don't think people will engage in the group chat. Also, it will be difficult for hosts to answer questions in a cluttered group chat. So, the best option is to only talk to the host about their event.

C. Where to show the interested events (Swiped right) and the events you RSVP'd to?

- **Option 1**: Add a page that contains the events you swiped right on and you can RSVP to events through that page
- **Option 2**: Add separate pages for interested and RSVP'd events

We decided to go for the second option because it provides a clear distinction to the user as to which events they are merely interested in and which ones they plan to go to. Moreover, this allows for the option to add more features to the RSVP page over the other, like showing updates from the hosts to events you are going to, which would be unnecessary for events you are only interested in.

D. How should we display recommended events to the user?

- **Option 1**: Use an Instagram/Facebook type feed where users can scroll through and find events
- **Option 2**: Use a tinder style page to swipe through events
- **Option 3**: Use a Pinterest type feed that has multiple events that users can scroll through

We decided to go for the second option because we believe it will provide the best user experience and will attract more users to the app. Additionally, we feel that this type of UI tends to be really engaging to the user and captivates their attention longer. Also, the swiping action provides an easier way to collect data for future recommendations because all we have to know is whether the user swiped left or right.

E. What if a host signs up for the app and finds their web scraped events?

- **Option 1**: Allow the host to claim those events

- **Option 2**: Letting the event pass and configuring the web scraper to not capture any events from that host again

We will be implementing the second option since it is easier and straightforward. Allowing the host to claim events would mean implementing a searching algorithm to try and match already existing events with a newly signed-up host, which is not guaranteed. Instead, once the host signs up, the web scraper will have the name of the host so that it does not capture its events from the web again. The host will have to upload their events now.

Non-functional Issues

A. Where should we host our different backend components?

- **Option 1**: Heroku
- **Option 2**: AWS Elastic Beanstalk

We decided to go for the second option because it offers better flexibility in configuring the environments meaning we can host both Python and a Node.js server, and both are needed for this project.

B. What type of database should we use?

- **Option 1**: MongoDB
- **Option 2**: MySQL
- **Option 3**: Firebase

We decided to go for the first option mostly to diverge and try to learn non-relational databases. However, one advantage for using a non-relational database is that it provides flexibility since we don't have to predefine the database with fields before use. Also, using the JSON document format in MongoDB is easier to use rather than using SQL queries to get data from the database.

C. How to integrate a recommendation algorithm in an efficient way?

- **Option 1**: Different matrices for different locations
- **Option 2**: One central matrix

Our team is unfamiliar with integrating machine learning models into production applications and we wouldn't know how to implement the first option. However, since we decided to have the application only local to Purdue we will not have to worry about having different matrices for different locations. We will only update the matrix as soon as a user or event is added, and remove an event when it's time has passed. This allows for the algorithm to run whenever a new user logs on and needs to be recommended events.

D. Which languages should we use for the server?

- **Option 1**: Java
- **Option 2**: Node.js
- **Option 3**: C#

Even though we collectively have more experience writing in Java, we decided to implement the server in Node.js. It is much easier to develop a backend server in Node.js and the hosting options are abundant. Also, there are many community-developed packages that can be used in the server if needed.

E. Which protocol should be used for client-serve communication?

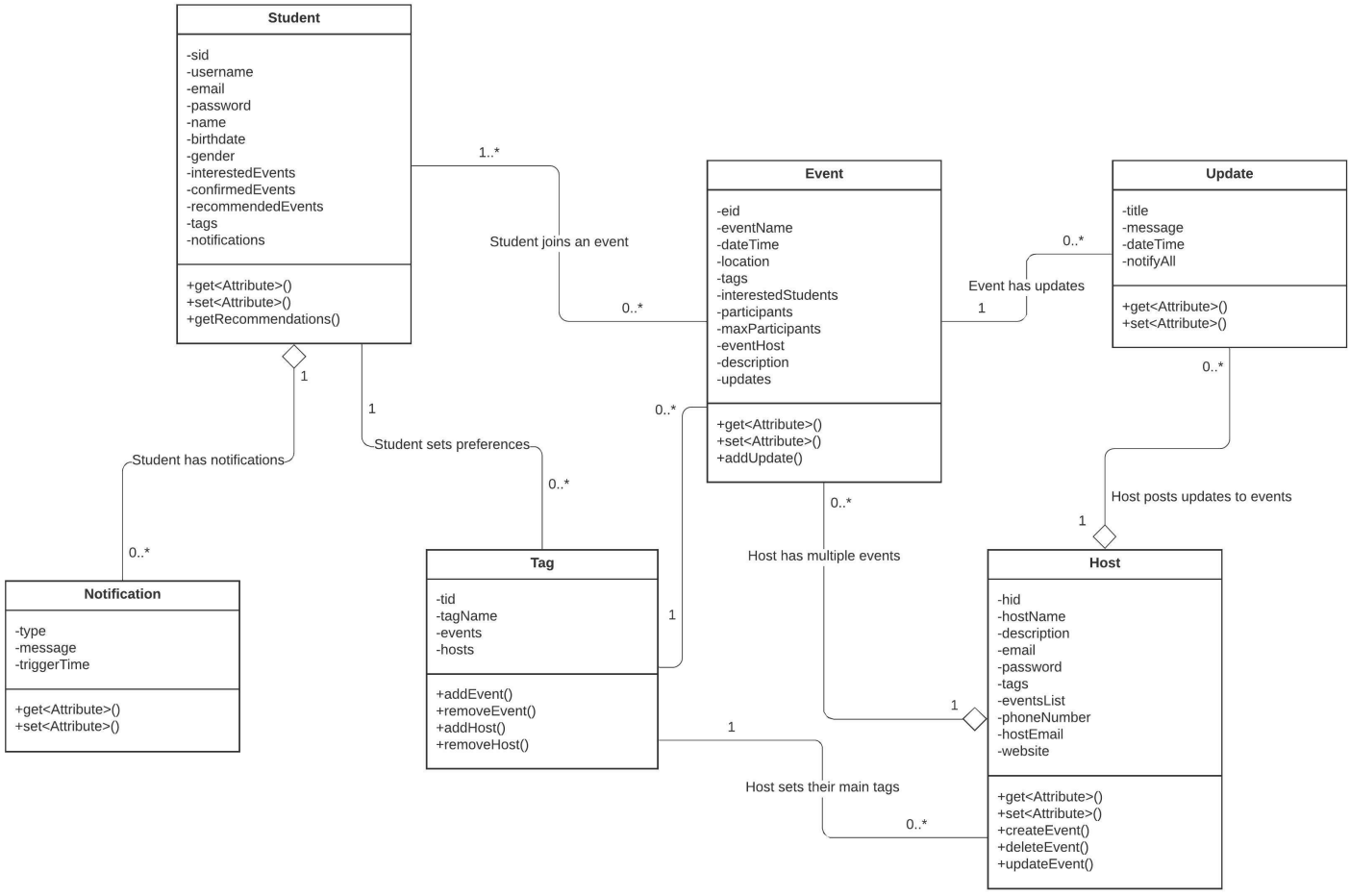
- **Option 1**: HTTP
- **Option 2**: HTTPS
- **Option 3**: WebSocket

Our choice was to use WebSocket as the main protocol for client-server requests since it provides much less overhead than using HTTP or HTTPS.

However, one issue is security which will be solved by using local cookies on the client side that will be sent with the request.

Design Details

Class Design



Descriptions of the classes and the interactions between them

A. Student
   - Student object is created when a user signs up to our application as a student
   - Each student is assigned a unique user ID
   - Each student has a username, email, and password for login
   - Each student fills in their gender, date of birth, and uploads a profile picture
   - Each student chooses their preferences for events by selecting tags
   - Each student has events they swiped right on (interestedEvents), events they RSVP'd to (confirmedEvents), and events they will be recommended on the swiping page
   - Each student can ask for more recommendations if they swipe through all their recommended events

B. Host
   - Host object is created when a user signs up to our application as a host
   - Each host is assigned a unique host ID
   - Each host has an email and password for login
   - Each host fills in their contact information and organization information
   - Each host chooses tags that their events mainly lie within
   - Each host can create, update, and delete their events

C. Event
   - Event object is created by a host
   - Each event is assigned a unique event ID
   - Each event has information about the event
   - Each event has a host that created the event
   - Each event has updates that are posted by the host

- Each event has a list of interested students and participating students
- Each event can have a maximum number of participants

D. Update
   - Update object is created by the host
   - Each update object has a title, description of the update, and date of creation
   - Each update object has the option to notify all participants

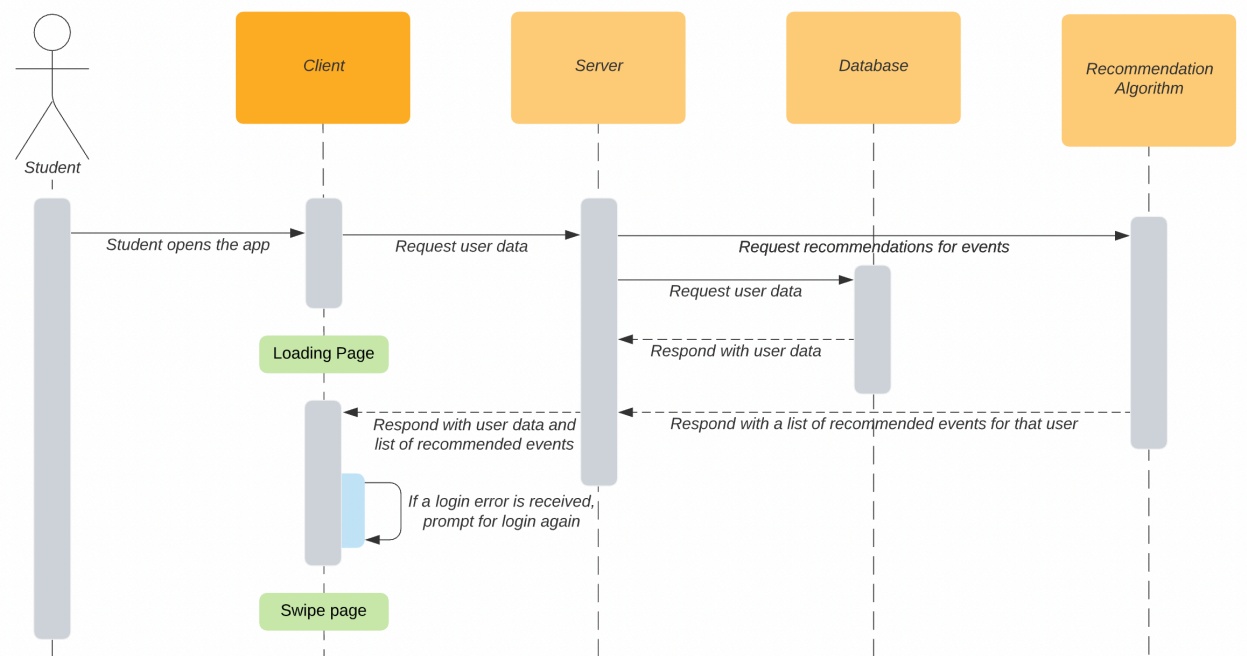
E. Tag
   - Each tag object has a unique tag ID and tag name
   - Each tag object has the list of events that contain this tag
   - Each tag object has the list of hosts that contain this tag

F. Notification
   - Each notification object has a type and message to be displayed
   - Each notification object has trigger time where it will be shown to the user

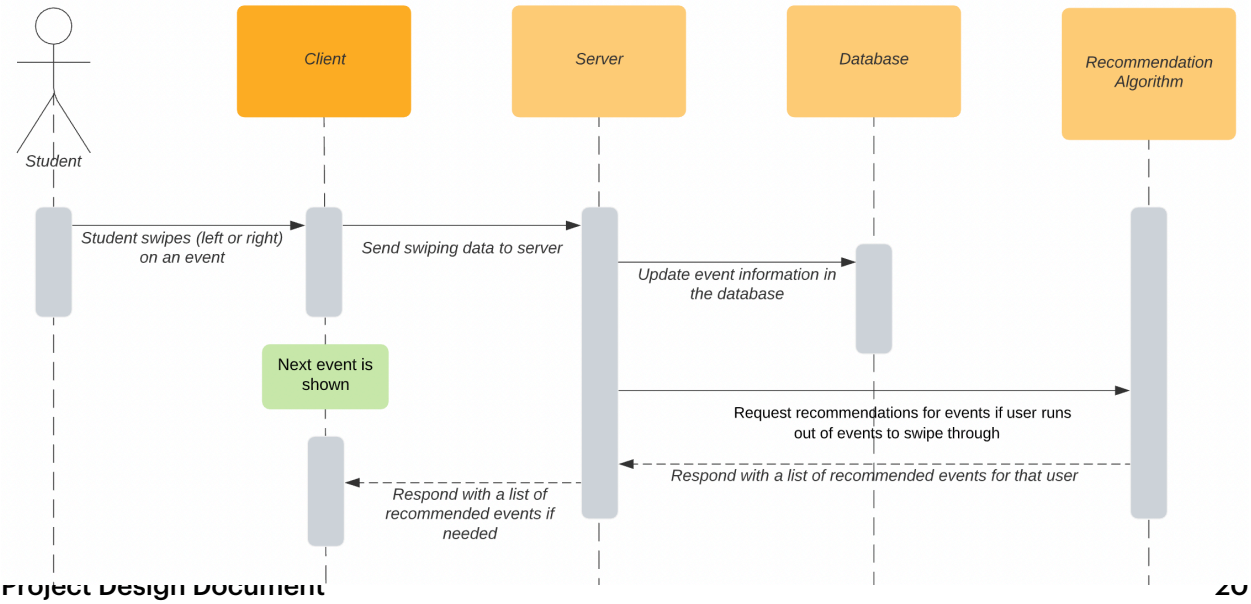
Sequence Diagrams

The following diagrams show the main actions that happen on the student and host side. They include the client, server, database, and the recommendation algorithm, and show the various components interact.

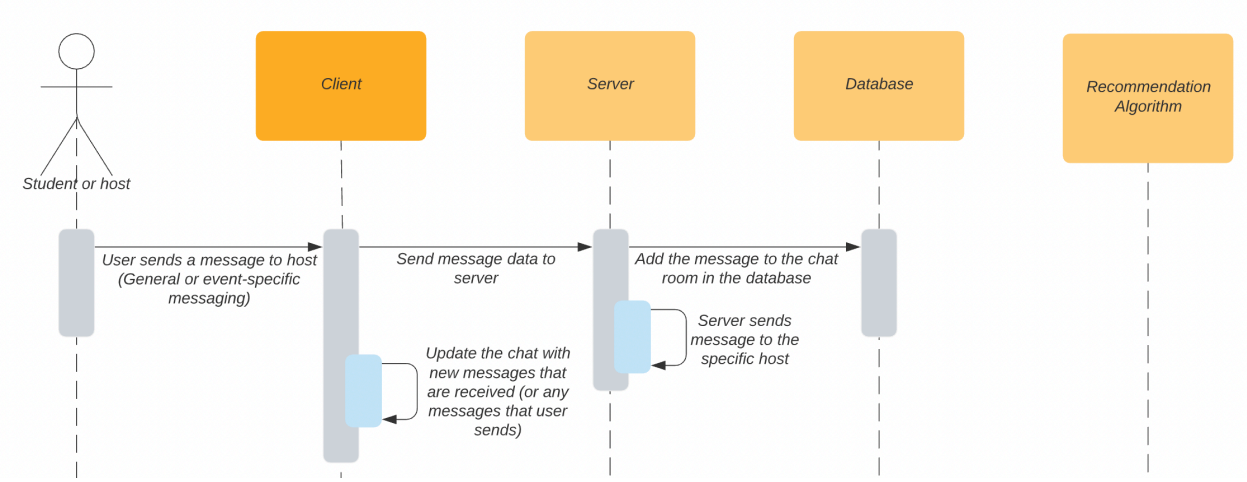
Student logs into the app



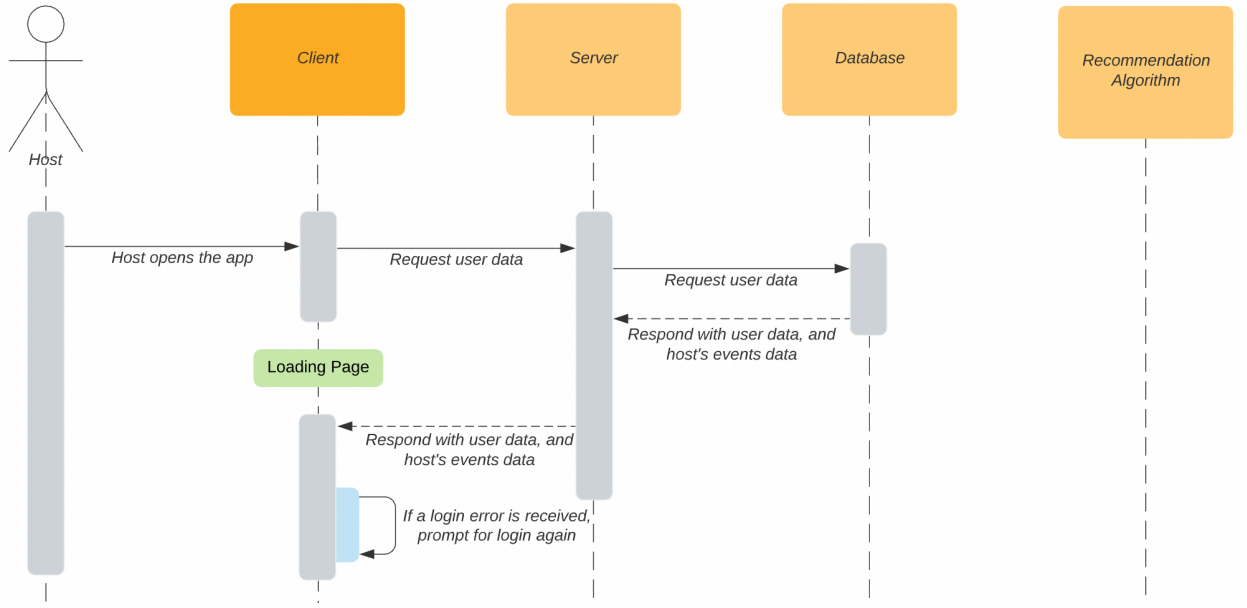
Student swipes on an event



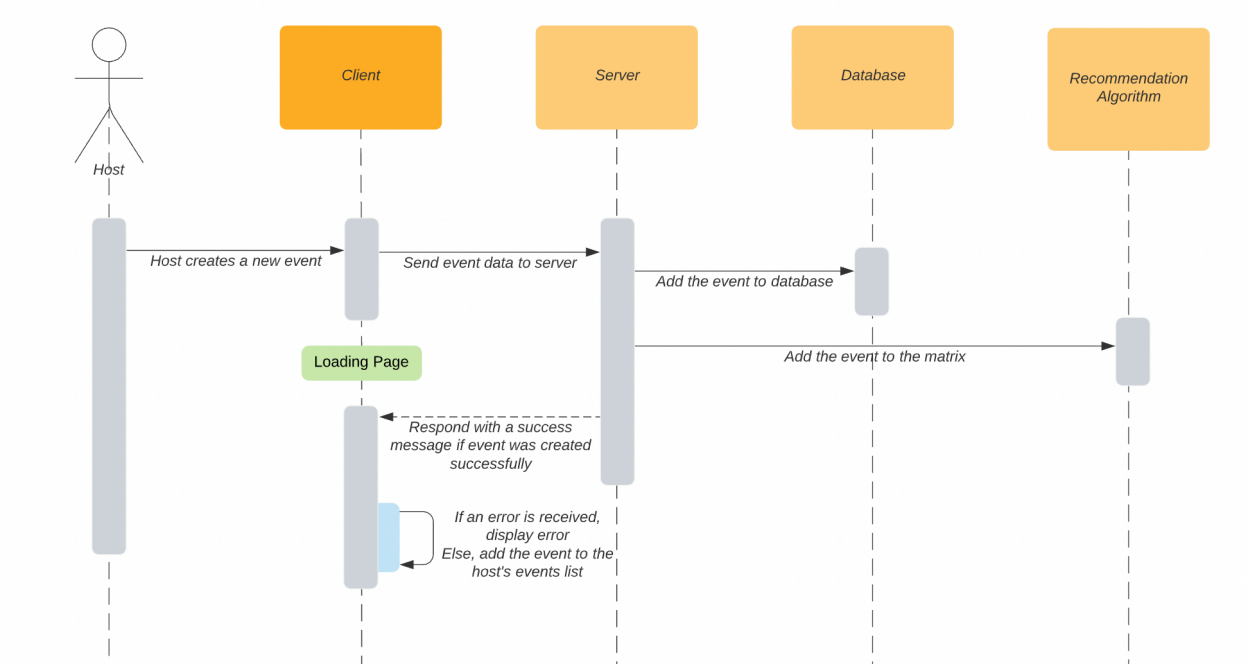
Student or host sends a message (event-specific or general message)



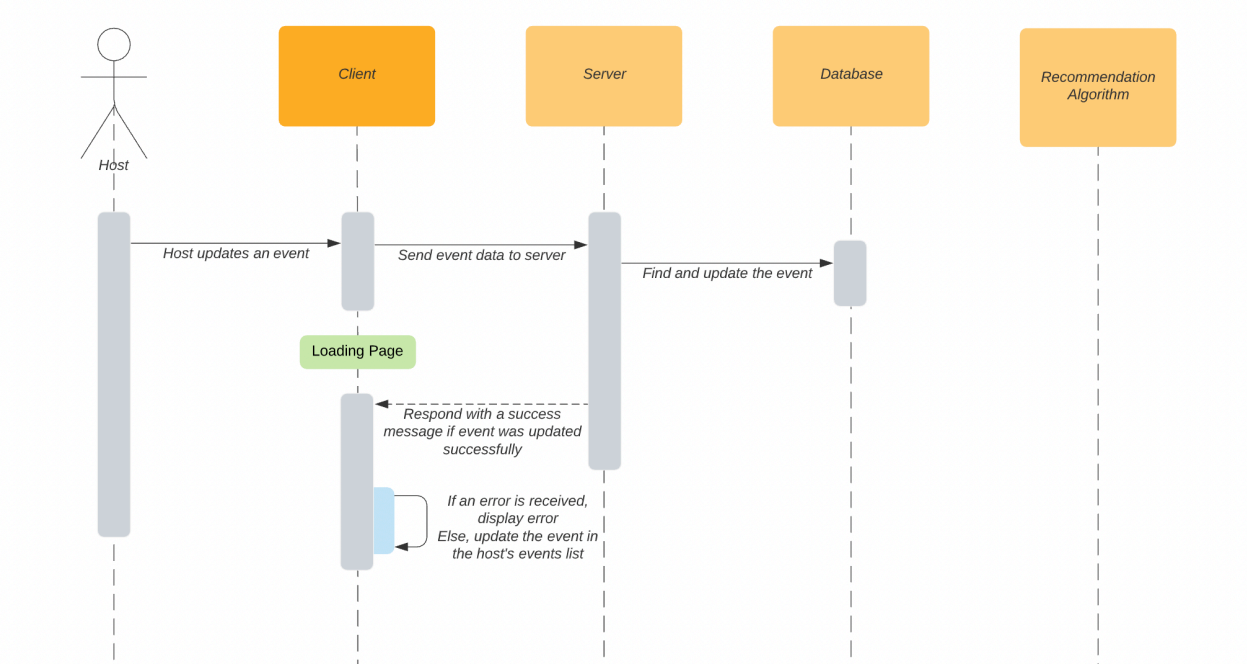
Hosts logs into the app



Host creates an event



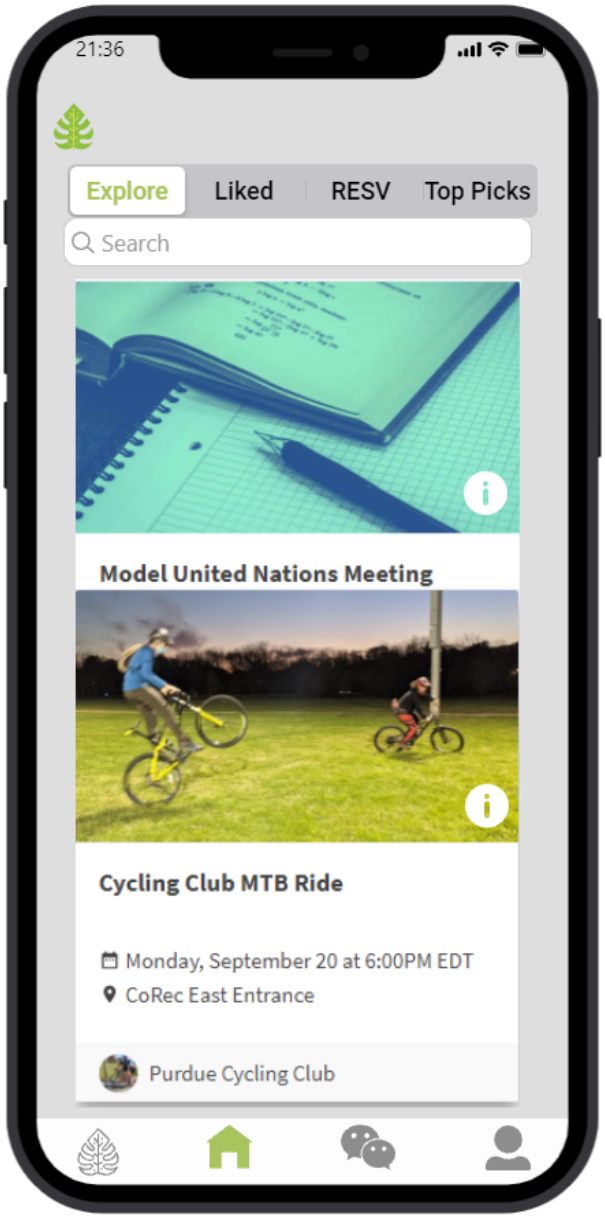
Host updates an event



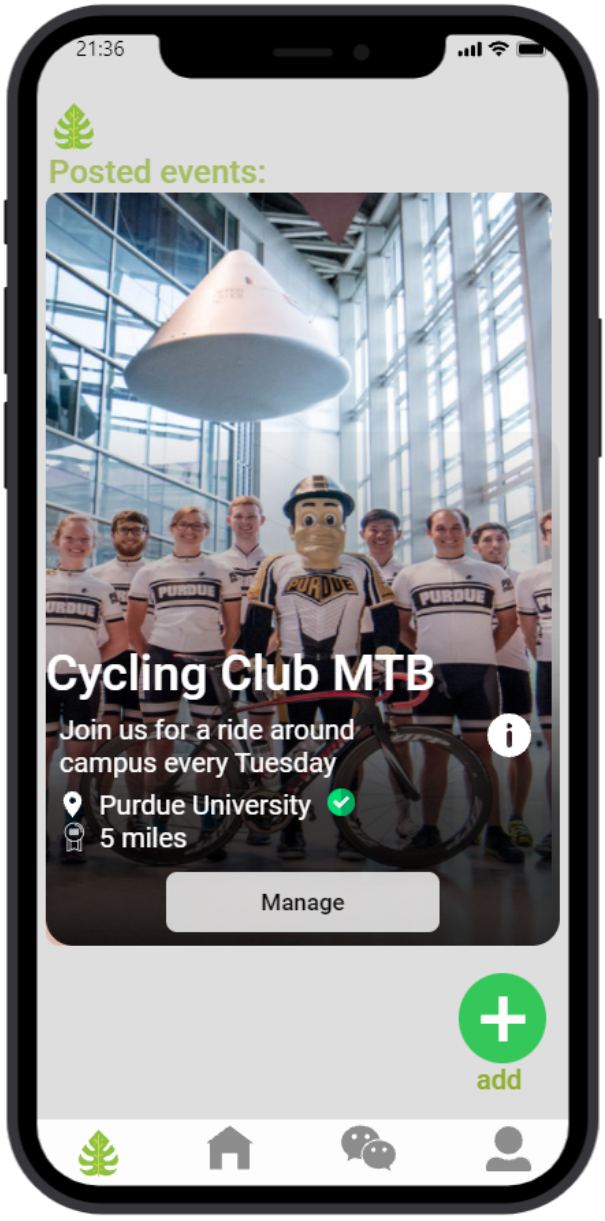
UI Mockups

Swiping page

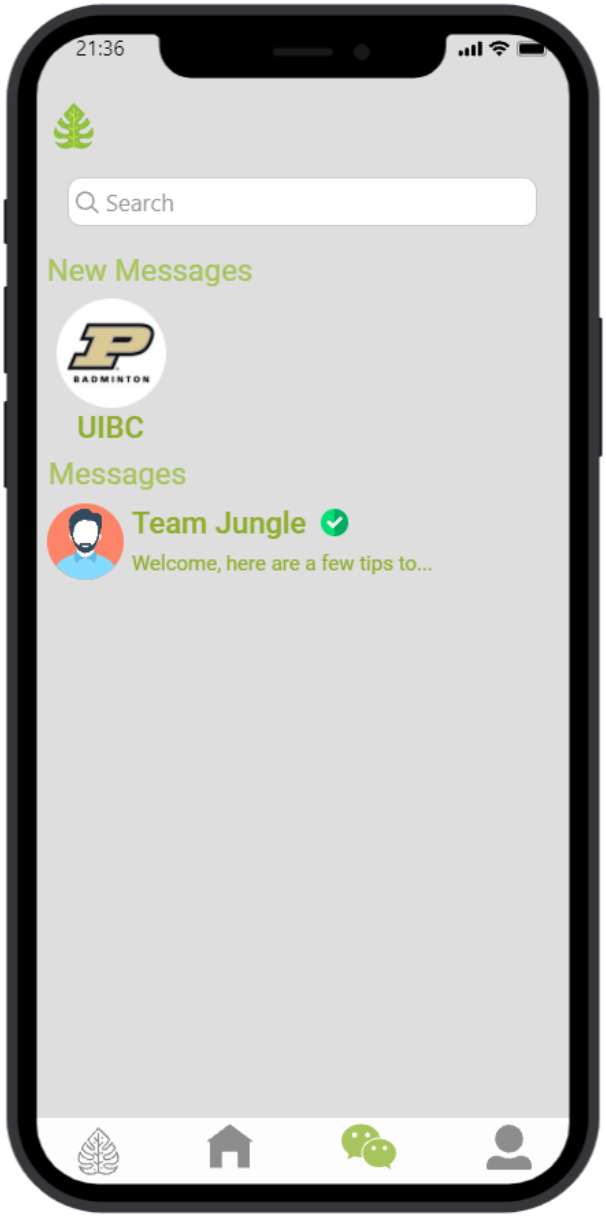
Interested, Confirmed, and Top picks



Host manage events



Chat page


21:36
Search
New Messages
BADMINTON
UIBC
Messages
Team Jungle
Welcome, here are a few tips to...

Event Info Page