papaya

Team 6

Ben Maxfield, Christian Lock, Adam Johnston, Caleb Flynn, Scott Hanberg

Table of Contents:

Purpose:

Presently, there are no apps dedicated to using college social life to aid studying. Even though there are so many students at each university, these students have no way to easily meet up with other students for study sessions. Additionally, professors have no way of knowing which students are meeting outside of class, or what they are doing if they do meet. Our goal is to mitigate these problems by developing a cross-platform application that will access location services for users to plan impromptu study sessions. Professors will also be able to see active user logs to gauge common study session topics to develop further.

Functional

Students can create and manage their study session for a class.
As a student:

1. I want to create a new study session.
2. I want to display to other students what topics will be studied at the study session and other session specific data (a session description).
3. I want to set a time frame that the study session will be active.
4. I want to have my study session's location displayed to my other classmates.
5. I want to automatically end my study session when I leave the nearby area.
6. I want to transfer ownership (transfer host) of a study session.

Students can view and join study sessions started by other classmates.
As a student:

1. ~~I want to filter what study sessions appear on my map based on class.~~
2. ~~I want to be able to view study sessions that are near me on a map.~~
3. I want to be able to join a study session created by a classmate.
4. I want to see the other students that are attending an active study session.
5. I want to show I joined a study session to my classmates in the session.
6. I want to invite friends to a study session.
7. I want to post to any active study session to communicate with those who are a part of it.
8. I want to remove my post from public view at an active study session.

Students can also:

1. ~~Sign up and login to the app.~~
2. Add friends to a friends list and have them appear differently on maps.

Professor can view past information about students and sessions.

As a professor:

1. I want to see who has been to past study sessions.
2. I want to see the dates of past study sessions.
3. I want to view the topics discussed from past study sessions.
4. I want to see posts from active study sessions.
5. I want to view active study sessions on a map.

Professors have more power to manage and create sessions than students.
As a professor:

1. I want to post and view comments on an active study session.
2. I want to create professor sponsored study sessions without having to be located nearby.
3. I want to be able to remove any post on any active study session.

Teaching Assistants can view and join study sessions created by other students.
As a Teaching Assistant:

1. ~~I want to filter what study sessions appear on my map based on class.~~
2. ~~I want to be able to view study sessions that are near me on a map.~~
3. I want to be able to join a study session created by a classmate.
4. I want to see the other students that are attending an active study session.
5. I want to show I joined a study session to my classmates in the session.
6. I want to invite friends to a study session.
7. I want to post to any active study session to communicate with those who are a part of it.
8. I want to remove my post from public view at an active study session.

Like students, Teaching Assistants can create their own study sessions, but they can also manage study sessions created by other students.
As a Teaching Assistant:

1. I want to create TA sponsored study sessions without having to be located nearby.
2. I want to display to other students what topics will be studied at the study session and other session specific data (a session description).
3. I want to set a time frame that the study session will be active.
4. I want to have my study session's location displayed to my other classmates.
5. I want to automatically end my study session when I leave the nearby area.
6. I want to transfer ownership (transfer host) of a study session.
7. I want to remove any student post on any active study session.

Non-functional

Client

1. Users must be able to use the app on an android device.
2. The interface will be simple and user friendly.
3. The app will be accessible on all screen sizes and resolutions but will be built primarily for late generation Android, iPhone and large browser screens.

Backend

1. The app must implement location (and GPS) services.
2. The app will integrate a back-end web service using AWS, to manage all user data.
3. The app will have fast response times, with no back-end call delay above a second.
4. The app and backend service should support at least 100 concurrent users during this development phase.

Security

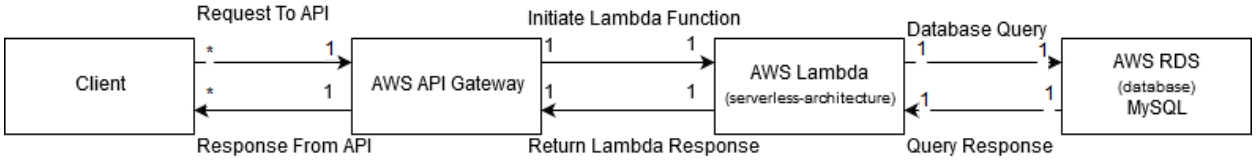
1. Our app will prevent security and privacy breaches. All data will be encrypted during transit between our service and our client (HTTPS). And all necessary client encryption keys and passwords will be stored as securely as possible on the client devices (Android Keystore System on Android devices).
2. Our app will enforce user roles (student, professor, teaching assistant) and their associated permitted actions.

Design Outline:

High Level Overview:

The goal of our project is to create a multi-platform app with a shared database. This type of relationship is often conducive to the client-server model, which is why we chose it as our structure.

More specifically however, in terms of the model itself, the clients will be mobile apps on either Android or iPhone and web browsers, and the server will be built around Amazon Web Services, most significantly using AWS Lambda. Our client-server architecture does not use a dedicated server, which would be typical in this situation. Rather, we will be taking advantage of a new paradigm, the server-less architecture. Using this architecture, developers no longer have to worry about maintaining servers, only the logical segments of code that run. Furthermore, server-less costs less, as you only pay for the resources you actually use. However, this does require our code to be 'stateless' and independent of each other and lack continued client communication (i.e. web sockets). To circumvent the statelessness of the code, we save all important records in a universal AWS RDS database. Finally, once a request from the client is processed and the query finished in the database, the result will be returned in the same JSON format back to the client connection.



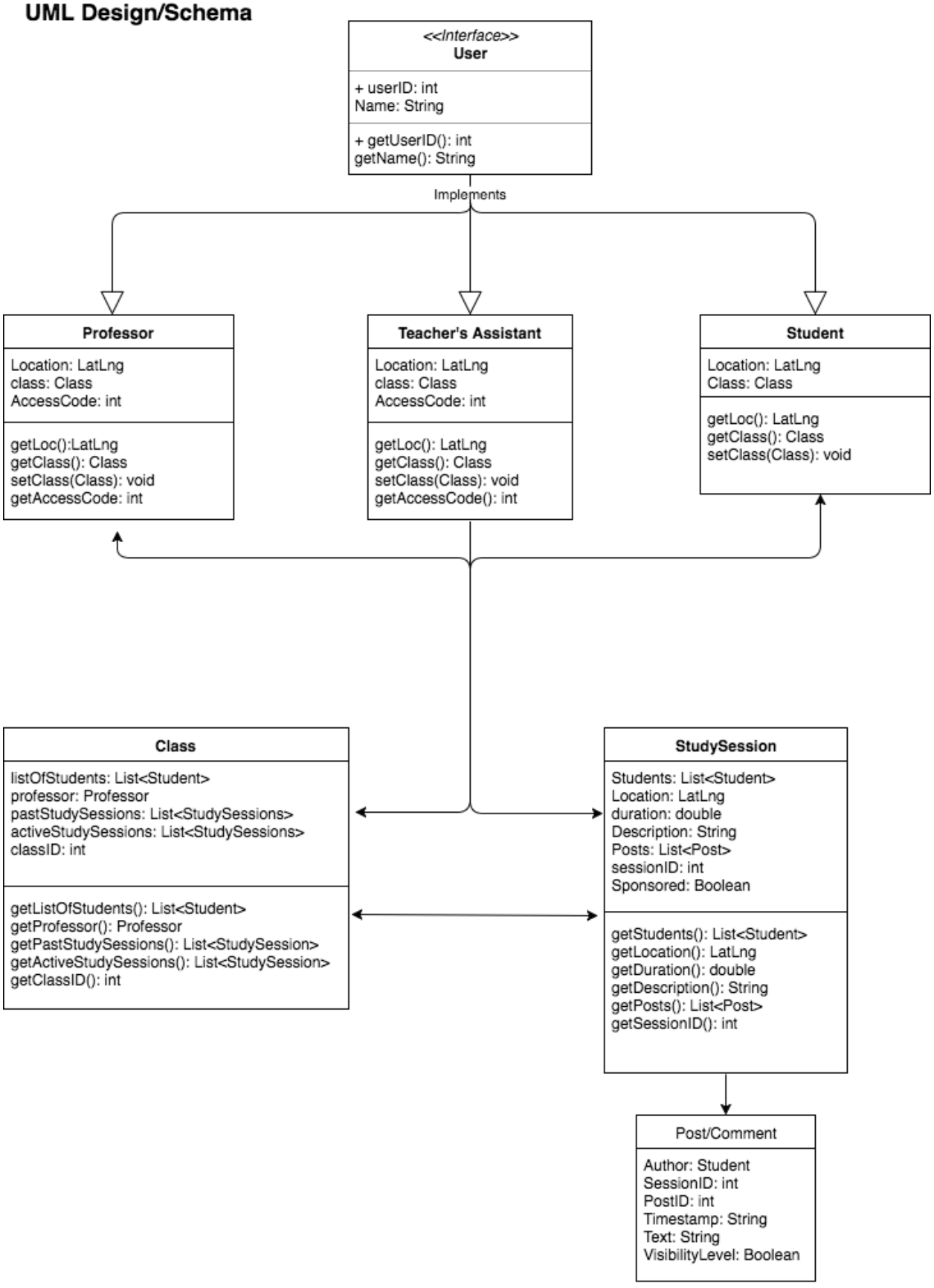
Class Diagram:

Below is an in App Design/Schema. Although things may change with the development of the app, the current schema will satisfy the standard design and functional requirements for a prototype version of our app.

1. User
    a. All users are implemented in this design. Users implement the professor, TA, and student classes.
    b. Contains login information of user, such as email, name, and password
2. Professor & TA's
    a. Professors and TA's will be identified by name and an access code after authenticating for special permissions.

    b. Professors and TA's will have location and class tags to view when new study sessions are made
3. Students
    a. Students will be identified by name after authenticating
    b. Students will have location and class tags to view when new study sessions are made
4. Class
    a. Classes will have a list of students to see who is enrolled in the class
    b. Classes will have a professor and a class ID to see what class it is and who teaches it
    c. Classes will have a list of active and past study sessions to save old sessions and record new sessions
5. Study Sessions
    a. Study sessions will have a list of students attending the study session
    b. Study sessions will have a location and a duration to see who is attending the study session and how long it will last
    c. Study sessions will have a description and a sponsored tag to see if it was created by a professor/TA or a student
    d. Study sessions will have a posts section where users can view active comments about the study session or topic of study.

UML Design/Schema



Design Issues:

Functional:

1. How should we design our main UI?
    - Option 1: Have a list of all available study sessions based on distance away from the user.
    - **Option 2: Display a map marking all of the study sessions nearby**
    - Option 3: Include a smaller map with a list of study sessions below

We felt having a map would be the best way to display the data on the main page, as it provides an instant visual for the user, allowing them to easily see study sessions nearby and catch their eye better than a list would. While we could include both, given that this is a mobile application and screen space is limited, it seems that it would become overly cluttered for a main page.

2. What development platform should we use for our application?
    - **Option 1: Native android and iOS**
    - Option 2: Web application

We have decided to go with using the native software of android/iOS rather than using a web application. While this reduces the portability of our application code, we felt the increase in quality was worth it. This will reduce load times for the main UI element, the map, since the web platform for Google Maps is their Javascript API, which requires about 2-3x more data to be transferred than the native API.

3. Should we target android or iOS first?
    - **Option 1: Android**
    - Option 2: iOS

We decided to go with android because for the practical development purposes. We are more proficient in java than Objective C or Swift and none of us have done any iOS development, we decided that we would be able to produce the best possible result by targeting android first for this prototype project.

4. Should we include push notifications in our design or not?
    - **Option 1: Yes**
    - Option 2: No

While adding push notifications adds a layer of complexity to our design, we felt it was necessary in order to notify users about updates to their study sessions. These notifications will trigger in situations such as when there are new posts on a session or new users joining a session.

5. Should we include private or public messaging?

- **Option 1: Public**
- Option 2: Private

We opted to include a public “post based” messaging system. One of the primary reasons for this is that we believe it will facilitate more on-topic discussion. It will also give the professor insight as to which issues/topics the students are struggling with and can adjust the class to accommodate that.

6. Should we create our own authentication system or integrate with Google and/or Facebook?
    - Option 1: Facebook
    - Option 2: Google
    - **Option 3: Facebook and Google**
    - Option 4: Create our own authentication system

The benefits of integrating with a system designed for universal app authentication is that, users are more inclined to ‘Login with Facebook’ or Google since it is far easier than to create a new account and remember a new username and password. For us, it is easier to use Facebook or Google authentication because we don’t have to develop a secure password management system, which is also an increased liability issue. Choosing either Facebook or Google individually have their detriments. The majority of the students will have Facebook accounts, but much less for the professors. However, Facebook is the primary authentication service for the majority of popular apps. Therefore, to satisfy both of our user demographics we chose to integrate both systems.

Non-Functional:

1. What type of database should we use for our app?
    - **Option 1: SQL**
    - Option 2: NoSQL

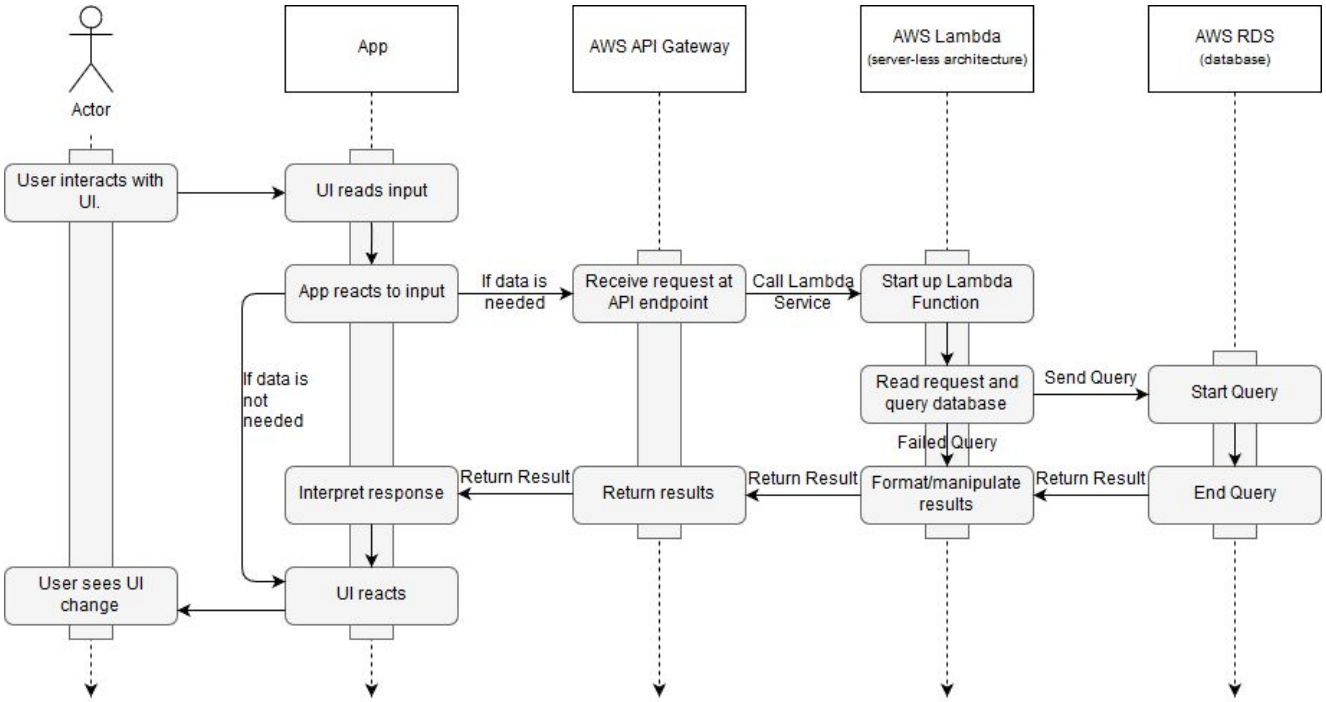
We have chosen to go with using SQL, since our data is fairly consistent and would do well in a relational database. We felt NoSQL would be unnecessary, as some of its core strength is in having data fields that can vary across objects.

2. Should we have multiple tiers of user access/permissions?
    - **Option 1: Yes**
    - Option 2: No

We opted to have multiple tiers of permissions for professors, TAs, and students. We are going to market this app to professors to use in their classes, so it makes sense that they should have extra privileges such as viewing study sessions their students have had over the course of the class, seeing deleted comments, and potentially creating study sessions with more options.

Design Details:

Typical Event Sequence:



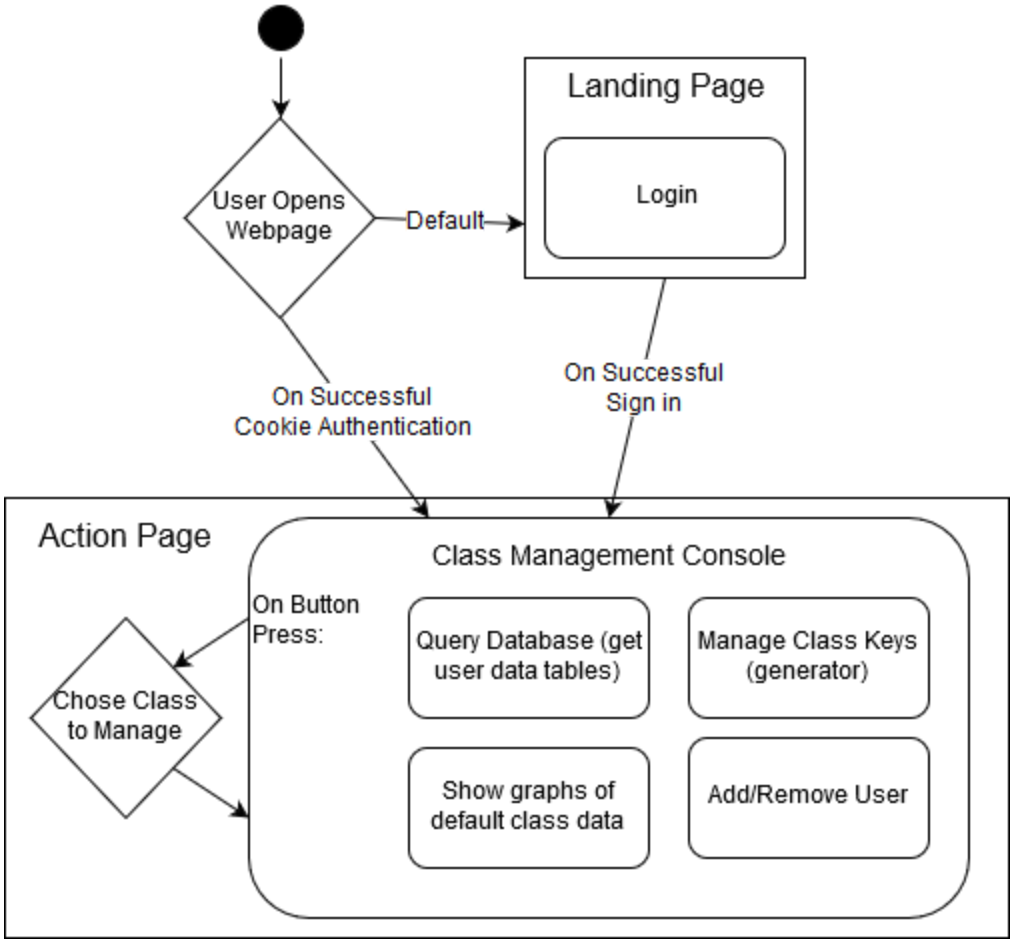
The diagram above outlines the typical event sequence, starting with a user interaction with the UI and ending with a UI response and, if external data is needed, querying the backend and database.

Whenever a client needs to obtain information about other users or needs to update other users about itself, the app must communicate with the backend. It does this by forming a JSON object containing necessary information about the request or data to be submitted and writing that to a RESTful API endpoint specified in *API Routes*. Developers must also make sure they are using HTTP headers correctly to ensure proper API execution. HTTP header use descriptions can be found in the *API Routes* section of this document.

The receiving of the API call will be handled by AWS API Gateway which will spawn a AWS Lambda Function (loading code stored on AWS S3) that will read the request JSON data and manipulate it correctly to retrieve or push data to or from the AWS RDS database. Once all necessary database requests were accomplished, the Lambda function will create a response JSON object and close itself (no more code running), returning the JSON, through API Gateway down through the connection created by the client, who is then able to read the response JSON and ensure data was written correctly and they received the data they

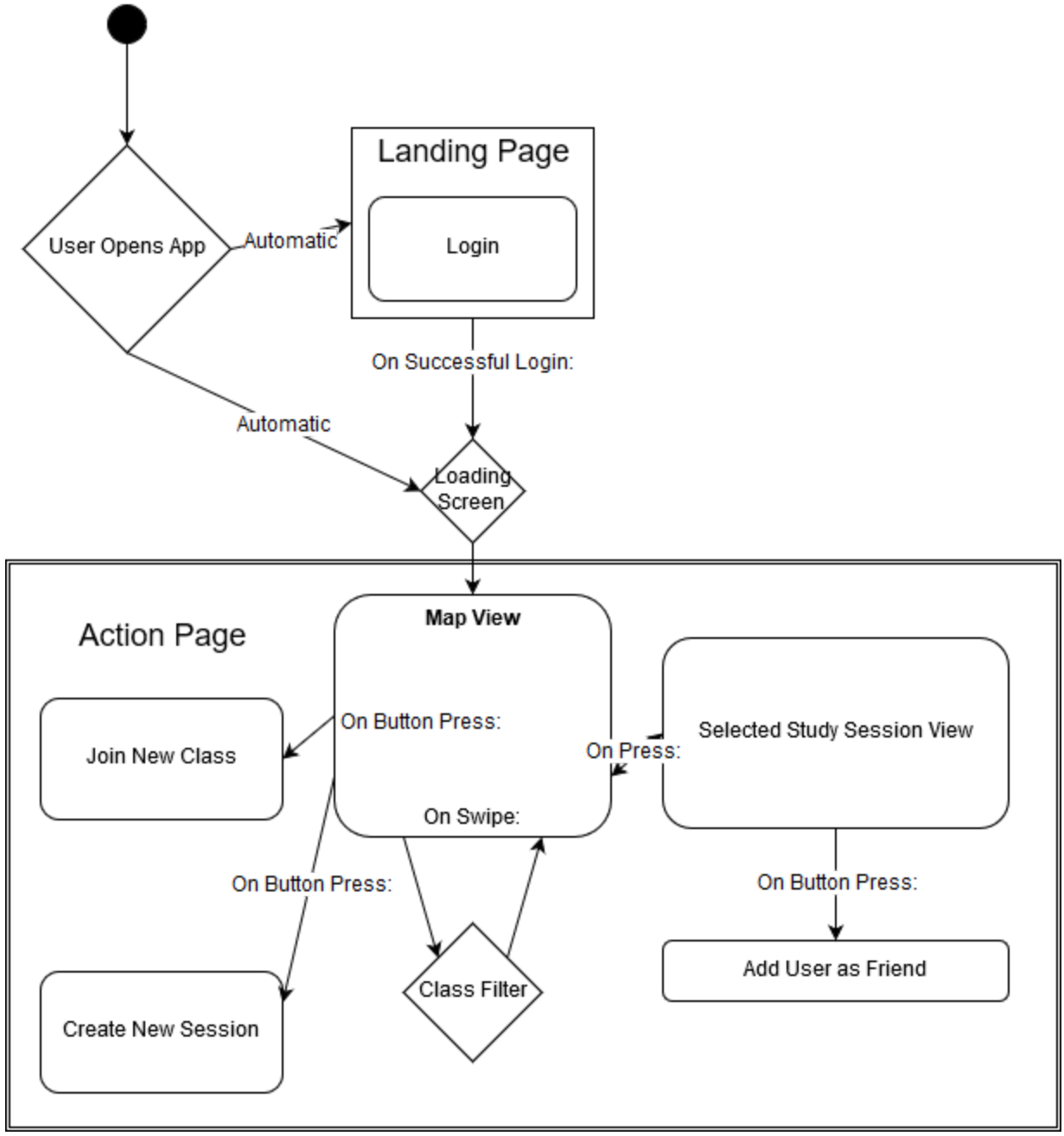
requested. If a database query fails, the returning JSON object will notify the client of its failure and the client can notify the user of a failure and attempt to restart the request.

Website User Interface Flow:



This is a short flow of the UI states necessary for the website/browser interface. The website will simply be a management console for professors. Here they can generate keys for their class (the tokens professors distribute to their students and TA's). Very little backend work will have to be made to accommodate for the browser functions as they will use much of the same API calls as the app.

App User Interface Flow:



This diagram outlines the UI states and flow of the client app. There are only two major pages: the landing page, where users can log in or sign up an account, and the action or activity page, where users can view the map, create sessions, and manage their account (classes, sessions, friends). We view that the success of our app will be tied in part to the simplicity and easiness of navigation of our app. Therefore, almost everything on our app

will be accessible from the Map View, and all other pages will just be overlays of the map (so the map is still partially visible so users know where their ‘center’ of the app is, and that they can easily get back to it).

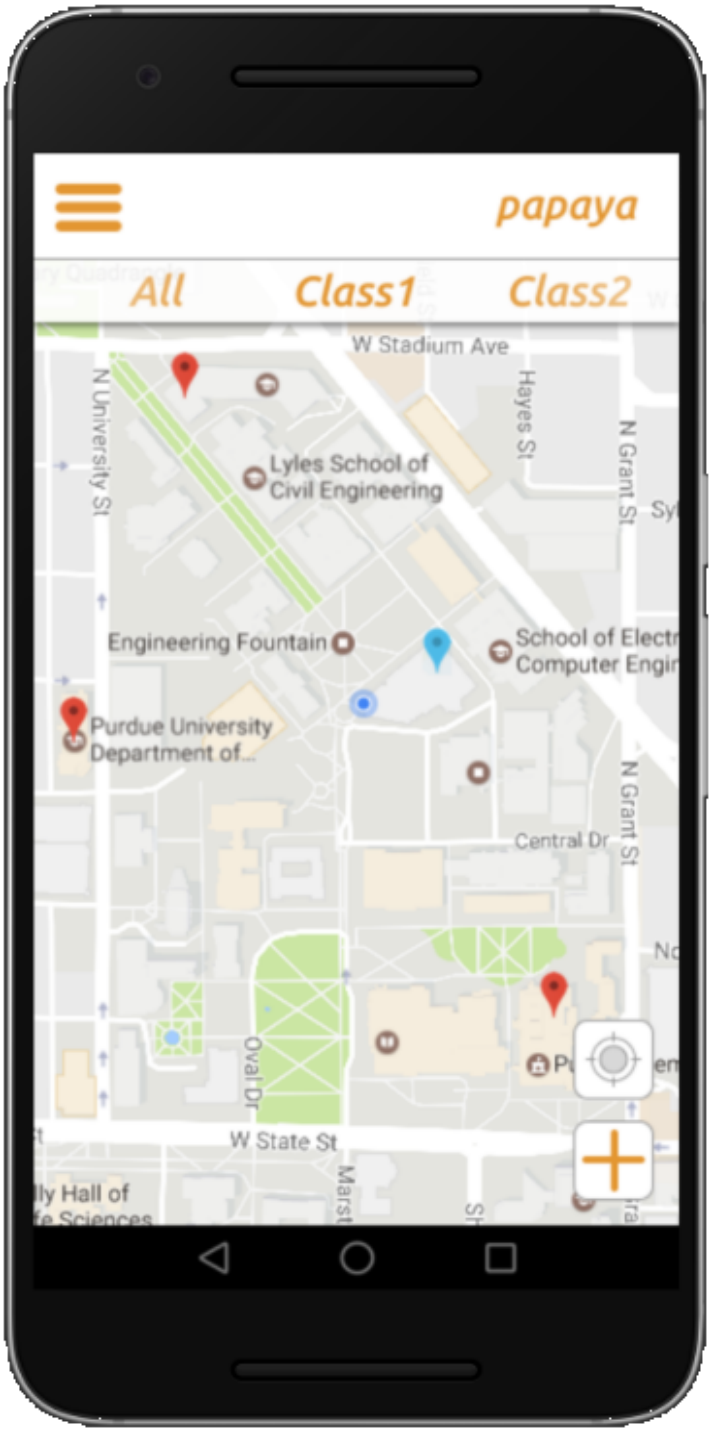
UI Mockups:

Login:

- This is the first page that you will see the first time you launch our app. It simply allows the user to choose a login method from which we can retrieve information from.

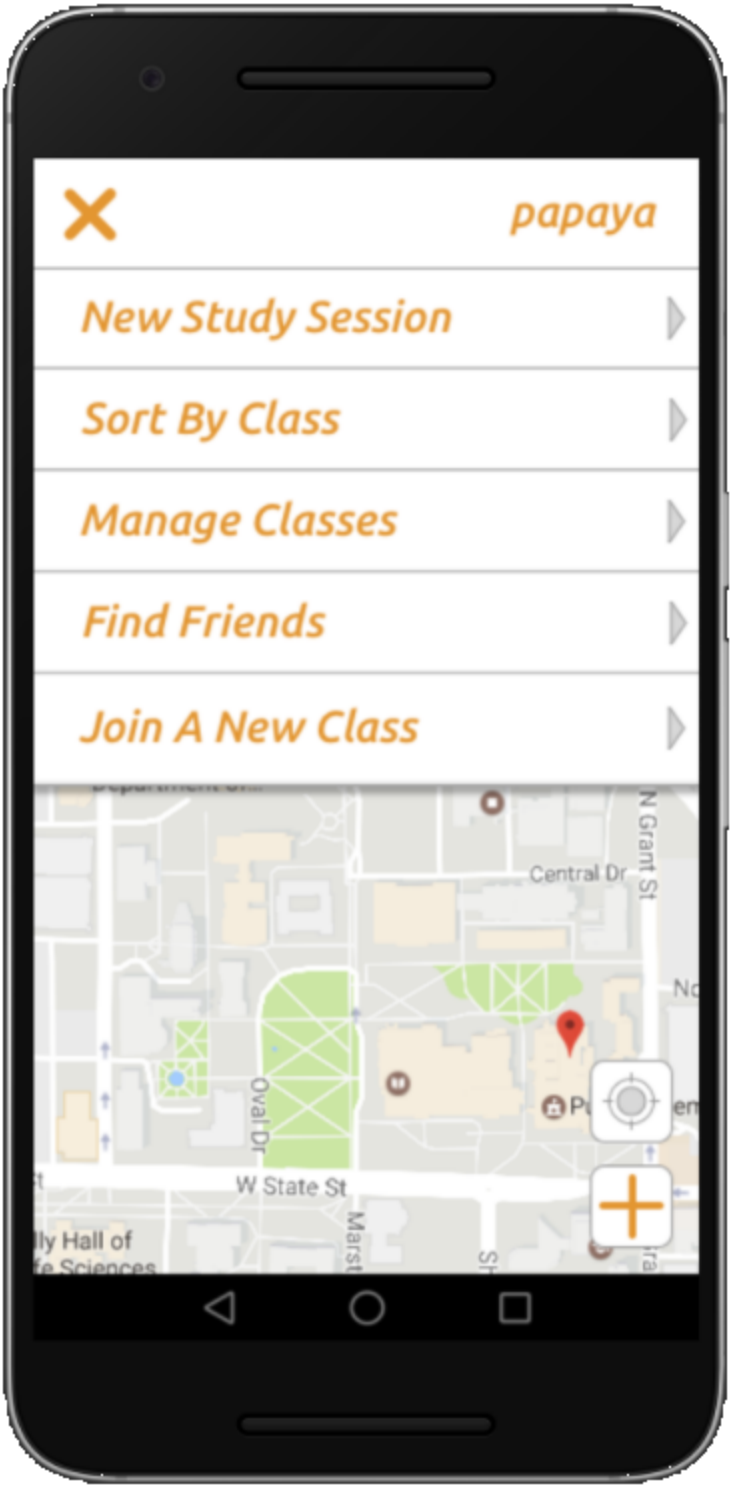
Map View:

- This is the main page a user will see after being logged in. It shows the user's current location, all active study session in the surrounding area, and easy buttons to find the current location and create new study sessions. At the top, there is also a menu bar that will be addressed later and a easy way to sort available study sessions by class.



Menu View:

- This drop down menu gives the user lots of options and ways to find other people to study with. They can create a new study sessions, sort the map by class, manage classes, find friends to study with or join a new class.



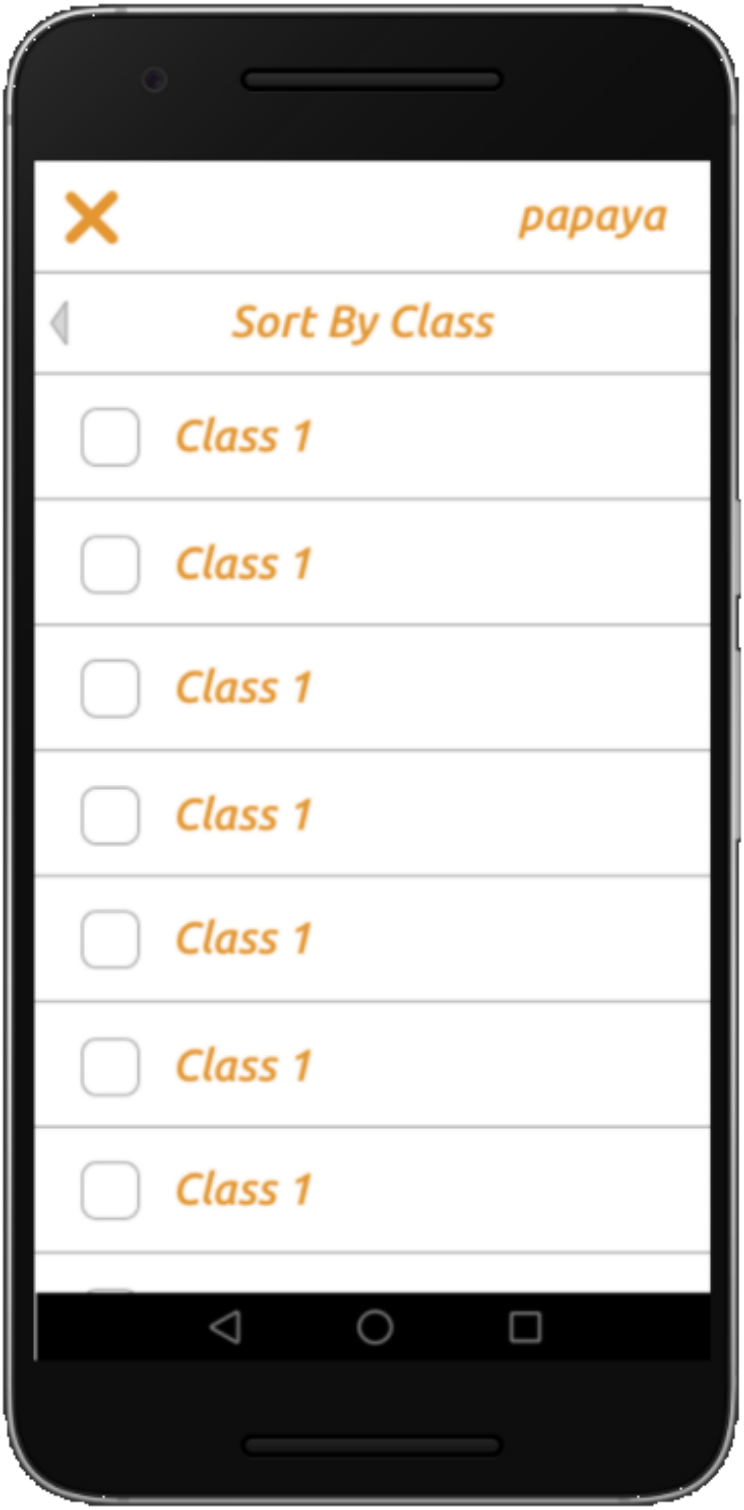
New Study Session:

- After tapping on New Study Session, the app pulls this page up which asks the user for information regarding the new study session being created. After entering the planned duration of the study session and a description of the study session, the app

will create the session and display it on the map for the other users of the app to see.

Sort By Class:

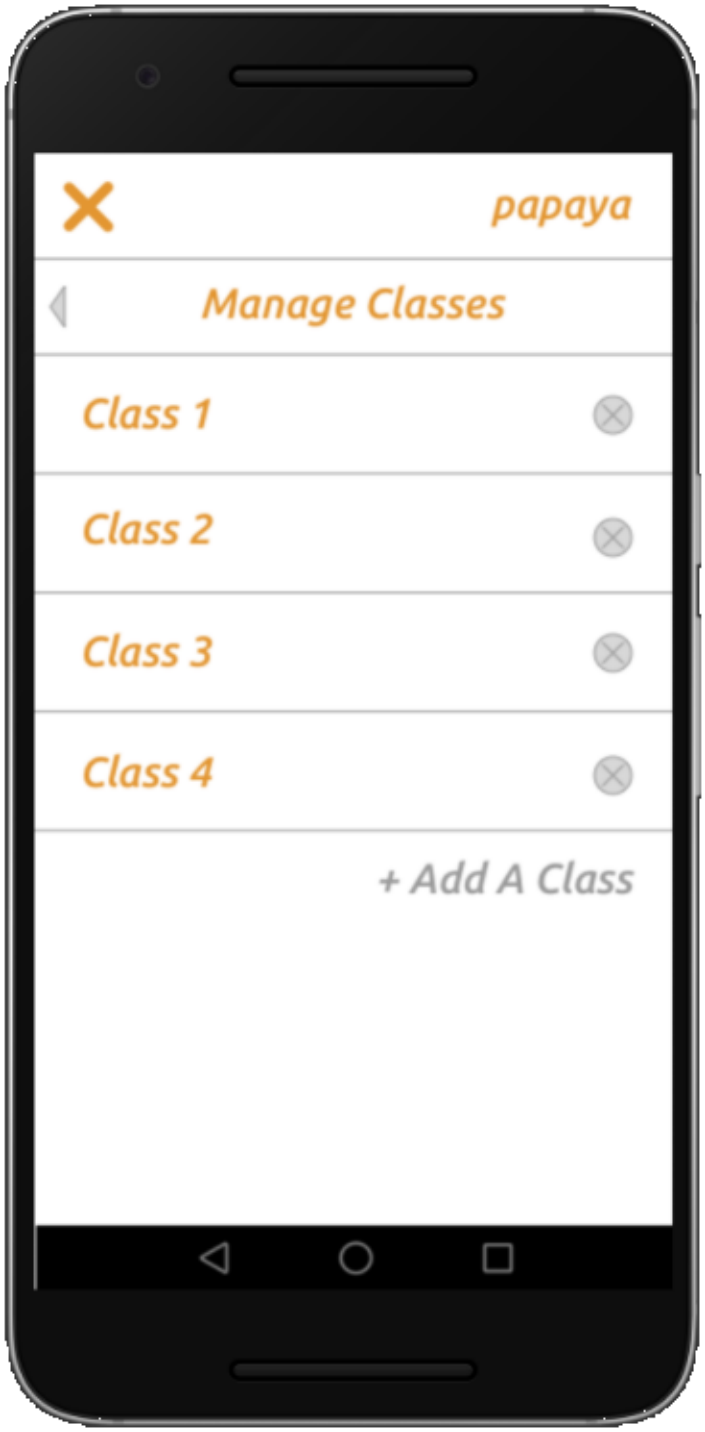
- This page allows the User to choose zero or more classes to display on the map to help sort through the different study sessions.



Manage Classes:

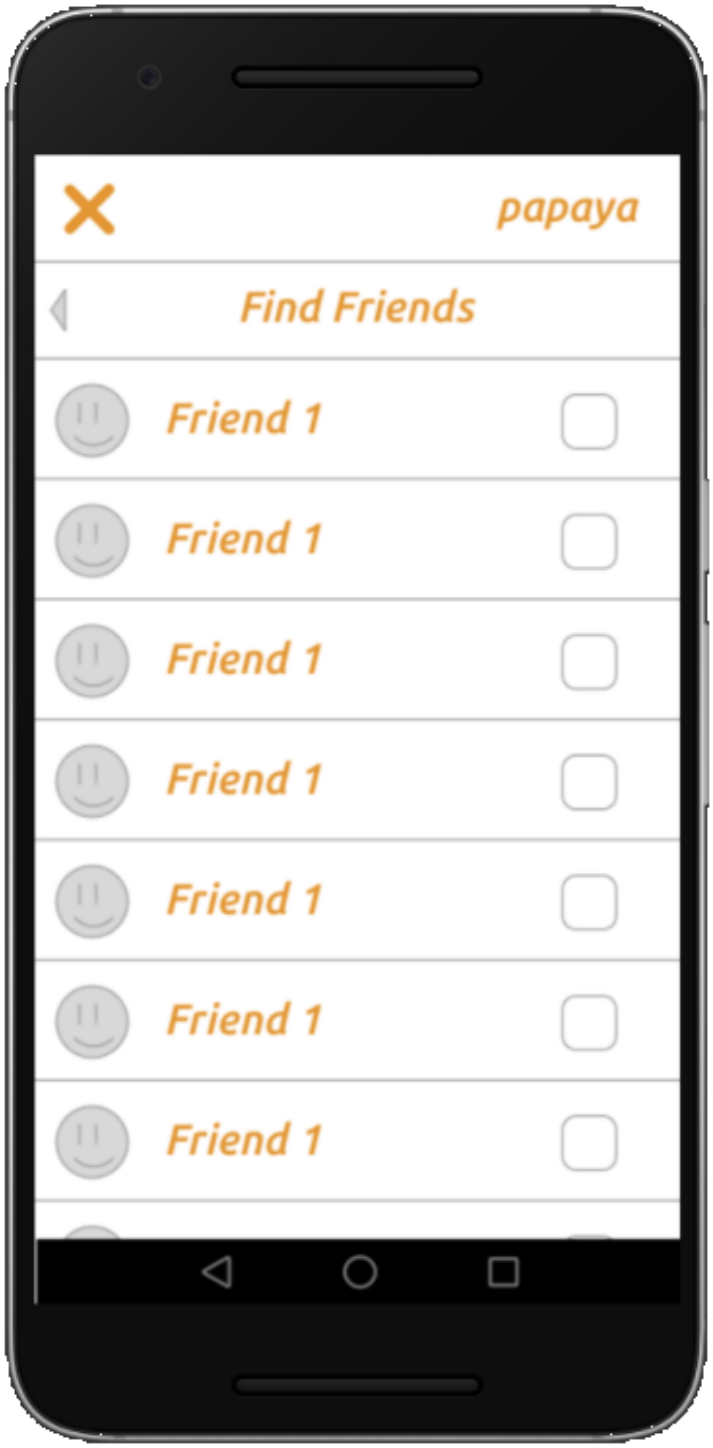
- This page allows the User to manage their classes. If the user no longer wants to be part of a class' study group, they simply tap on the 'x' corresponding to the class.

There’s also an add class button at the bottom if the user is looking to add another class.



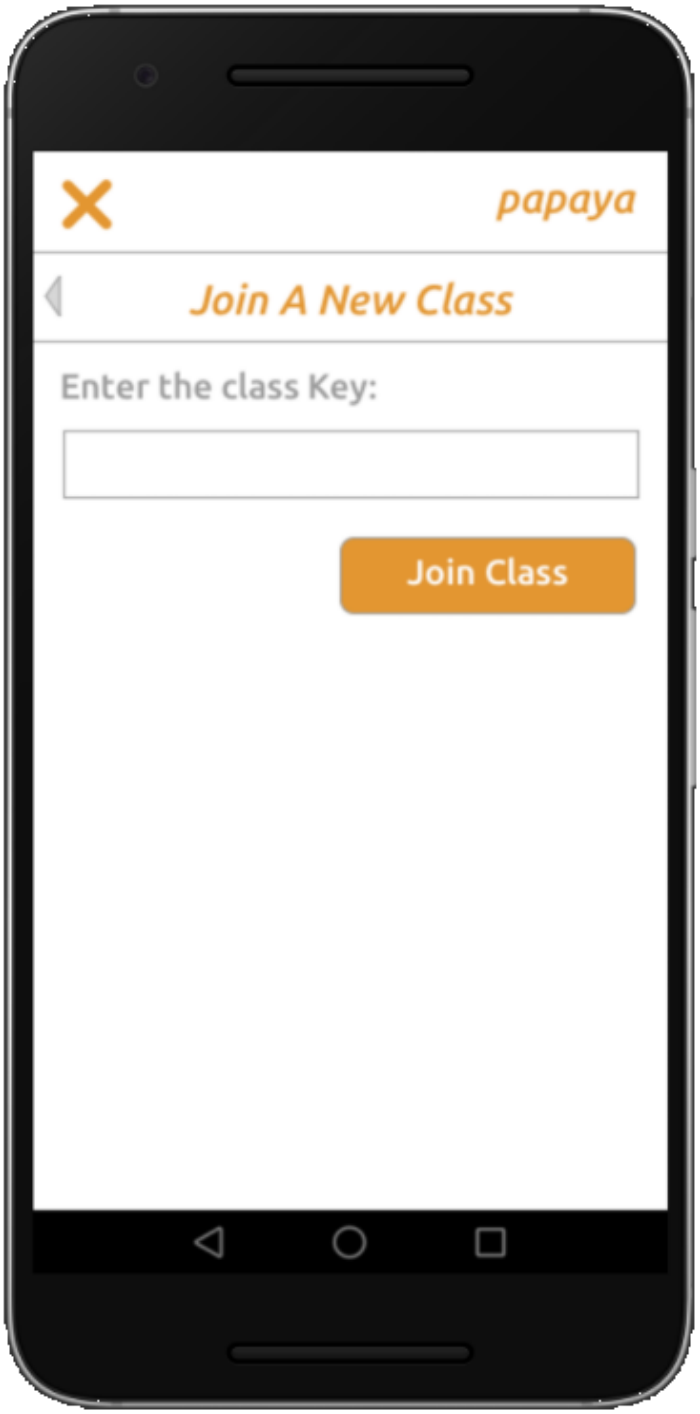
Find Friends:

- The “Find Friends” page allows you to ‘friend’ anyone in any of your current classes. These friends will make the sessions they have joined a different color, so users can more easily find sessions with their friend.

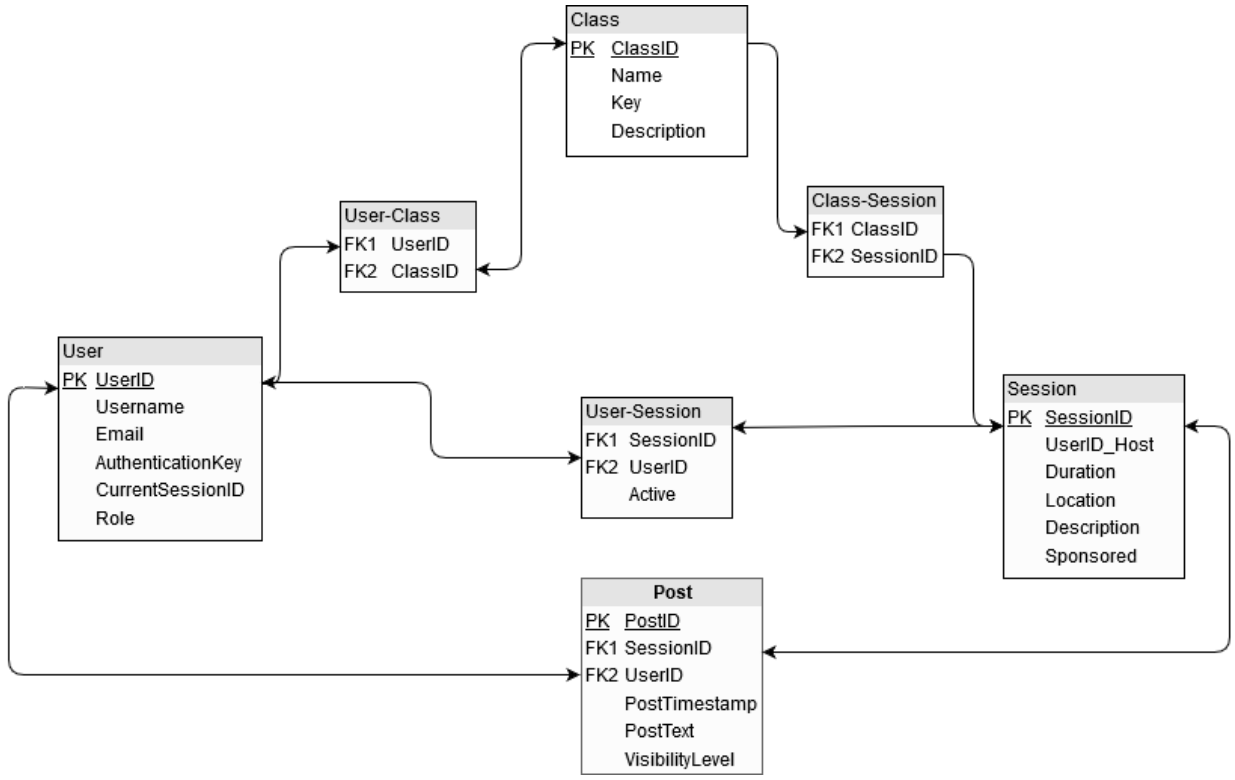


Join A New Class:

- This page allows the user to join a new class once they are given an access code from their professor. After joining the class, they can find other classmates within the class and sort by the class.



Database Design:



Database Description

1) User
    a) This table will represent Students, TAs, and Professors and different statuses will be given based on their corresponding UserLevel value
    b) Contains information about the user such as their username, email, and authentication key in. If a user is given the key for a class, they will be able to join that class, in which case the User-Class table is updated
    c) Sessions order to verify the user
2) Class
    a) This will contain a list of classes, containing their name, an identification key, and a description
    b) This table will contain a list of all sessions of each class. This table will contain the session durations, locations and descriptions of the study sessions.
    c) This table does not hold which users are in a session or which class a session belongs to. This will be handled in the User-Session table and the Class-Session table respectively.
3) Post

    a) This table holds posts that users make to study sessions. It contains the session ID that the message was posted in, the userID of the user who posted it, and information about the post, such as visibility level, which will determine who can see the post. This will be useful for when a professor would like to see deleted posts.
4) User-Class
    a) As stated above, this will maintain which users are in which classes. This allows our class and session tables to be much smaller.
5) User-Session
    a) Similar to the User-Class table, this will allow our user and session tables to be smaller by maintaining which users joined which sessions. Their current session however, will also be held in the user table.
6) Class-Session
    a) This table will maintain which class each session belongs to. Again, it will allow us to have fewer rows in the class and session tables.

API Routes:

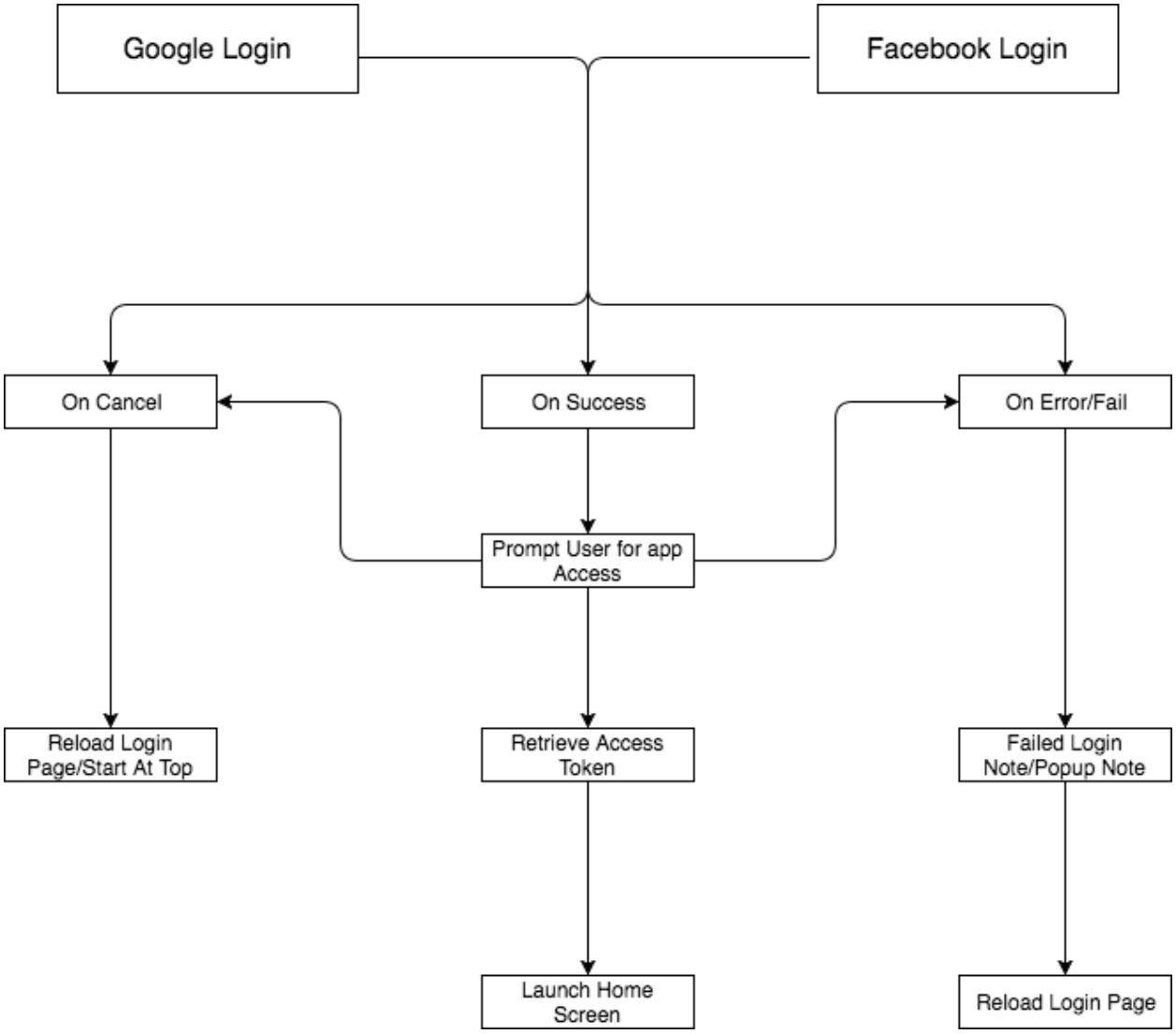
Public API routes are made available for all devices to interact with our universal backend. The API is designed and will be implemented according to REST standards. Our API will be serviced by Amazon Web Services' API Gateway Service, which will route requests to be handled by its proper backend service. See each route for a description of its purpose:

Route	Accepted HTTP Methods	Description
user	PUT, POST	PUT - Authorizes 3rd party account and saves 'cookie hash' in DB. POST - Used to create a record in our database for a new user and checks the contained Google/FB account keys.
user/classes	GET, POST	GET - Returns a list of all a user's acquired classes. POST - Used when a user joins a class, creating student-to-class link in the database.
user/friends	GET, POST	GET - Returns a list of the user's friends' user ids from the database. POST - Adds a new friend to a user's friends list in database.
class	POST	POST - Creates a new class.
classes/[id]/sessions	GET, POST	GET - Retrieves the active sessions for a specified class id.

		POST - Create a new study session under this class id.
classes/[id]/sessions/[id]	GET, POST, PUT, DELETE	GET - Retrieves data about a specific study session given its id. POST - Adds user to a study session (user joins study session). PUT - Updates study session using data given in body (may update description of study session or even transfer ownership of study session to another user). DELETE - Either deletes study session, if the user owns the study session, or user leaves study session, if he/she does not own study session.
classes/[id]/sessions/[id]/invite	POST	POST - Notifies other users of invites to a study session and its location.
classes/[id]/sessions/[id]/posts	GET, POST, DELETE	GET - Gets all posts for a session. POST - Creates a new post to a session. DELETE - Removes a post from public view.

User Authentication Flowchart:

Authentication will work on a three way system that prompts the user to login before launching the home screen of the app. If the user has logged in before on a certain device, settings will be saved and they will not have to login again. If the user is using the app for the first time on their device, then authentication will force the user to login. In the case of logging in, three cases will be handled. If the user successfully authenticates through Facebook or Google, they will be asked for permissions to use their account and receive an access key for the app to identify the user. If the user chooses to not login, then they can cancel login and reset back to the authentication screen where they will be prompted to login before launching the home screen again. In the case of an error or failure when logging in, the user will be notified with an error message and told to try again. On the server side, once our user sends us an API call with an authentication token, we will have to verify its authenticity using one of Facebook's multiple debugging endpoints to check if it's a valid access token. This will prevent malicious users from token hijacking and protect our backend from security breaches.



App and API Diagram:

This diagram is an in depth diagram combining the *UI Flow* diagram and *API Routes* table. Solid arrows show user flow or information travel. Diamonds are transient statuses (a long term transition effect, not interactable like a loading screen). Dotted lines show data travel across the Internet (not locally). The labels on the dotted lines signify the API HTTP method used and the API Gateway items are the API resource endpoints.

Android & API:

By: Ben Maxfield