



Team 2 - Design Document

Alejandro Griffith, Jiwon Seo, Hojin Sohn, Sun Ahn, and Peter Kang

Index

Purpose.....	3
Functional Requirements.....	3
1. Profile Settings.....	3
2. Groups and Friend Settings.....	3
3. Daily Photo Prompts and Responses.....	4
4. Daily and Weekly Votes / Leaderboard.....	4
5. Comments.....	5
6. Battle Shop / Coins.....	5
7. Global Snaps.....	6
8. Message System.....	6
Non Functional Requirements.....	7
Architecture and Performance.....	7
Security.....	7
Usability.....	7
Hosting/Deployment.....	7
Design Outline.....	8
High Level Overview.....	8
1. Frontend Client.....	9
2. Backend server.....	9
3. Database.....	9
4. Cloud.....	9
Sequence of Events Overview.....	10
Design Issues.....	11
Functional Issues.....	11
What information do we need for users to create an account?.....	11
How will the prompts be generated across the app?.....	11
What group permissions are administrator-only?.....	11
How will the voting periods work?.....	12
How should prompts be created?.....	12
Non Functional Issues.....	13
What backend language/framework should we use?.....	13
What web service should we use?.....	13
What frontend language/framework should we use?.....	13

What database should we use?.....	14
Design Details.....	15
Class Design.....	15
Description of Data Classes and their Interactions.....	16
User.....	16
Group.....	16
Prompt.....	16
Post.....	16
Chat.....	17
Leaderboard.....	17
Comment.....	17
Sequence Diagrams.....	18
Sequence of events when users login.....	18
Sequence of events when users create a group.....	19
Sequence of events when user joins a group.....	20
Sequence of events when user posts a photo.....	21
Sequence of events when user votes for a photo.....	22
UI Mockup.....	23
Authentication.....	23
Main Home Page.....	24
Profile Pages.....	25
Friends Pages.....	26
Group Home Page + Misc.....	27
Voting Page.....	28
Error prompt.....	29

Purpose

Despite the plethora of social media apps available, staying connected with friends you don't often see is challenging, as popular social media platforms do not tend to promote close, intimate friendships. Our project, SnapBattle, aims to solve this problem by challenging friend groups to share photos of their daily lives that best respond to specific prompts.

Our mobile app will allow users to connect with a small group of friends in a fun and engaging way. Every day, each group member will receive the same prompt, such as "What was your favorite meal?" or "High-five your co-worker." By the end of the day, each person will need to respond to the prompt with an image they took.

Functional Requirements

1. Profile Settings

- a. As a user, I would like to create an account with an email and password.
- b. As a user, I would like to sign into my account.
- c. As a user, I would like to be able to change my username and password.
- d. As a user, I would like to delete my account and all the pictures associated with it.
- e. As a user, I would like to be able to choose a profile picture and change it later if wanted.
- f. As a user, I would like to be able to recover my password if I cannot sign in and forget my password (if time allows).
- g. As a user, I would like to be able to have a biography section viewable by other users to showcase my personality.

2. Groups and Friend Settings

- a. As a user, I would like to create a group of friends and become the group administrator.
- b. As a user, I would like to be able to join an existing group of friends when an invitation is sent.
- c. As a user, I would like to search for a user by username and view their profile information (username, profile picture, biography, achievements, etc.)
- d. As a user, I would like to be able to invite a user to a group of friends.
- e. As a user, I would like to add/remove friends from my account.
- f. As a user, I would like to view the list of friends in my group.
- g. As a user, I would like to be able to leave a group.
- h. As a user, I would like to be able to block users.

- i. As the administrator of a group, I would like to be able to transfer my permissions to someone else if I leave the group.
- j. As the administrator of a group, I would like to be able to kick users.
- k. As the administrator of a group, I can set the maximum number of users that can join my group.
- l. As a user, I would like to be a part of multiple groups.

3. Daily Photo Prompts and Responses

- a. As a user, I would like to be able to receive random prompts every day.
- b. As a user, I would like to receive push notifications when the prompts are released, and I would like to receive notification reminders if I have not responded to the prompt by the deadline (if time allows)
- c. As an administrative user, I would like to be able to set/edit a time for my group to receive prompts every day.
- d. As an administrative user, I would like to be able to set/edit a time for my group to send picture responses.
- e. As an administrative user, I would like to choose how long the daily and weekly voting periods are. Daily voting periods should start immediately after the daily submission deadline, and weekly voting periods should start immediately after the last daily voting period of the week.
- f. As a user, I would like to take a photo inside the app to submit to a daily prompt with two resubmissions.
- g. As a user, I would like the option to save every submission that I make to my local phone album.

4. Daily and Weekly Votes / Leaderboard

- a. As a user, I would like to participate in a daily vote to choose the best picture-prompt combination of the group for that day.
- b. As a user, I would like to see the results of the daily voting contest at the end of the daily voting period.
- c. As a user, I would like to receive 1 leaderboard point if my response to the daily prompt received the greatest number of votes within the group, and I would like to receive 3 leaderboard points if one of my prompt-response combinations wins a weekly contest with the greatest number of votes.
- d. As a user, I would like to participate in a weekly vote to choose the best picture-prompt combination of the week, where only picture-prompt combinations that won daily votes are possible options to vote for.
- e. As a user, I would like to see the results of the weekly voting contest at the end of the weekly voting period.

- f. As a user, I would like to see the leaderboard of the friend group, which is updated with the results of both daily and weekly votes and which resets every two weeks.
- g. As a user, I would like to be able to choose a date anytime within the past year and view/save the daily prompt-response combination that won that day.
- h. As a user, I would like to be able to choose a date that falls on a weekly voting deadline, anytime within the past year, to view/save the prompt-response combination that won that week.
- i. As a user who is at the top of the leaderboard when it resets, I would like to receive an achievement that I can show on my personal profile page.

5. Comments

- a. As a user, I would like to be able to leave comments on my friend's daily photo submissions.
- b. As a user, I would like to be able to view the comments on my daily photo submissions and my friends' daily photo submissions.
- c. As a user, I would like to be able to edit my comments on my friend's daily photo submissions.
- d. As a user, I would like to be able to reply to the comments that other users have commented on my post.
- e. As a user, I would like to be able to delete comments on my daily photo submissions, as well as my comments on my friend's daily photo submissions.
- f. As a user, I would like to block comments on my daily photo submissions, meaning that no one can comment on my photo submissions
- g. As a user, I would like to be able to like comments on my daily photo submission.

6. Battle Shop / Coins

- a. As a user, I want to earn coins (currency) and/or achievements for my consistent participation in daily submissions and daily/weekly votes. For example, I want to earn 1 coin every time I submit a photo before the daily deadline (if time allows)
- b. As a user, I would like to see the coins (currency) that I have earned (if time allows)
- c. As a user, I would like to open a shop where I can buy various items (buffs/nerfs) with the coins I have in order to get an edge in competition (for example, I could buy an item that resets the daily prompt for everyone to something I want) (if time allows)
- d. As a user, I would like to be able to see the items (buffs/nerfs) I have purchased and are ready to activate (if time allows)
- e. As a user, I would like to be able to activate the items (buffs/nerfs) I have whenever I want (if time allows)

7. Global Snaps

- a. As a user, I would like to be able to open the photo-response combination that won the week to the public. (if time allows)
- b. As a user, I would like to see a feed of various photo-response combinations that won weekly contests in different groups. (if time allows)
- c. As a user, I would like to be able to like any public photo-response combination anonymously. (if time allows)

8. Message System

- a. As a user, I would like to have automatically created a group chat for the group that I am in.
- b. As a user added to an existing group, I would like to see the history of the chat before I was added.
- c. As a user, I would like to be able to see if people are typing in chat.
- d. As a user, I would like to be able to send messages in the group chat that are viewable by all other users in the group. Other users should see my username as well as my profile picture along with my message.
- e. As a user, I would like to be able to reply directly to another person's message.

Non Functional Requirements

Architecture and Performance

We plan to use React Native for our front-end, Node.js/Express.js for our back-end, and MongoDB for our database. React Native will allow our application to be available to both Android and iOS users without having to rewrite code for each platform. Our backend will host a REST API, in order that the front-end software and back-end software can communicate in a consistent manner. Lastly, MongoDB is suitable for this project because it is inexpensive and can scale horizontally easily, which is needed to support many users. We hope to be able to support at least 500 users concurrently. Furthermore, it is likely that our data will change frequently, so MongoDB will be a better fit for the project than a relational database. Additionally, since we need to store images on the backend, we would likely need a Cloud Storage platform such as Google.

Security

As another social media platform, security is vital to ensure that users feel safe using our app. We hope to ensure that pictures uploaded by users to certain groups will be secure within the database and visible only to those within the chosen group. Requests to and from the server will be encrypted to ensure that stolen information will not expose sensitive information.

Usability

We want the interface to be simple and easy enough for average users to understand. For our app to be user-friendly, it is important to make the app navigation clear. This is especially true because this application will be accessible to both Android and iOS. Furthermore, it is often hard to fit a lot of things onto a mobile screen, meaning that we will have to have many screens for various purposes, with navigation between each one.

In addition, because our entire idea is based on interactions between friends in a group, we need to ensure that editing groups and group settings (such as prompt release times and deadlines) should be simple for group administrators (users that create a group).

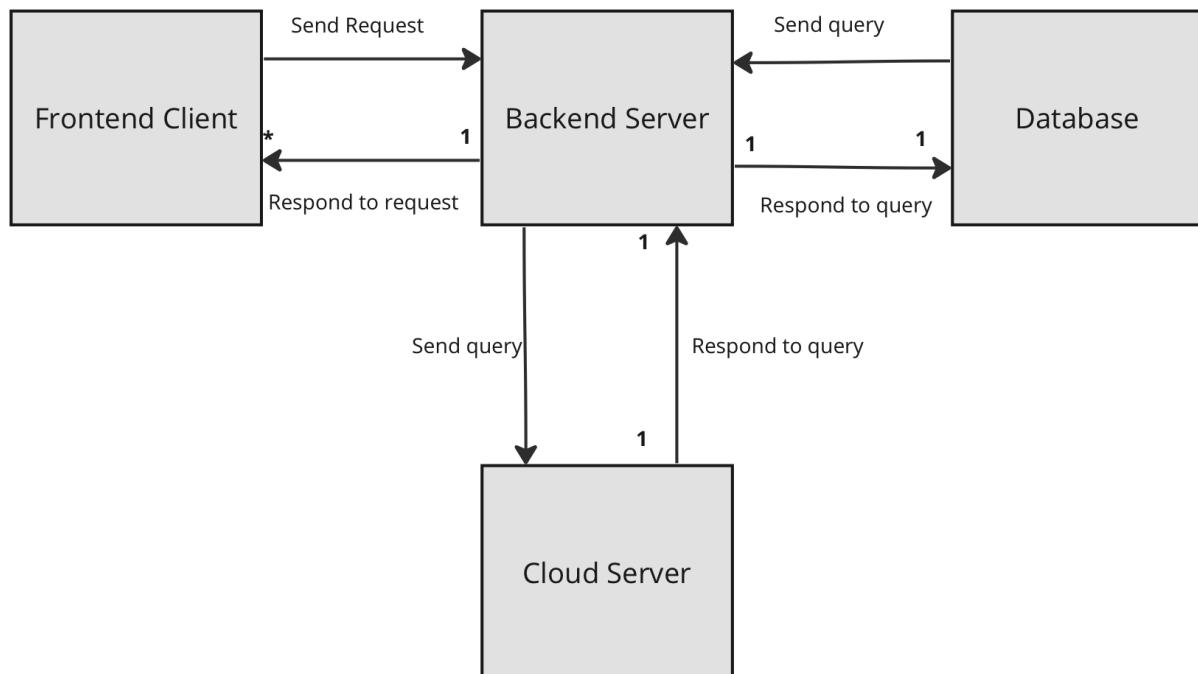
Hosting/Deployment

Because we have a separate front-end and back-end, we can deploy each on separate platforms. Since this is a cross-platform mobile app, we will attempt to deploy our app to both the Google Play Store and the App Store. Moreover, we will attempt to host our backend on Heroku, which will be able to handle a minimum of 500 requests per minute. For our pictures in particular, we hope to store all the pictures on Google Cloud.

Design Outline

High Level Overview

Our project aims to provide a mobile social media platform that promotes close, intimate friendships. Users will be able to post pictures to respond to prompts that are assigned daily, which will be viewable only to those in the group that the post was made in. Furthermore, users will be able to interact with the posts of their friends. This application will use the client-server model in which one server simultaneously handles access from a large number of clients using the MERN (MongoDB, Express.js, React Native, Node.js) stack framework in JavaScript. The server will accept client requests, handle database operations such as create, update, delete, and provide feedback back to the client. The client-server model works especially well for mobile apps, as it requires less processing power from the client since most of the processing is executed by the server. Furthermore, this ensures smooth data handling and adaptability for different user needs and allows for standardized communication between multiple devices simultaneously.



1. Frontend Client

- a. The mobile client, running on each user's phone, provides an interface for our system.
- b. The mobile client will retrieve and send data to our Express.js server via HTTP requests, passing data in JSON format.
- c. The mobile client will receive and interpret the HTTP responses from our server and render the changes in the user interface.

2. Backend server

- a. The server will receive and handle the HTTP requests from the mobile clients
- b. The server will validate requests and send queries to our MongoDB database when appropriate to add, modify, delete data.
- c. The server will generate responses with the appropriate status codes and data and send them to the targeted client.

3. Database

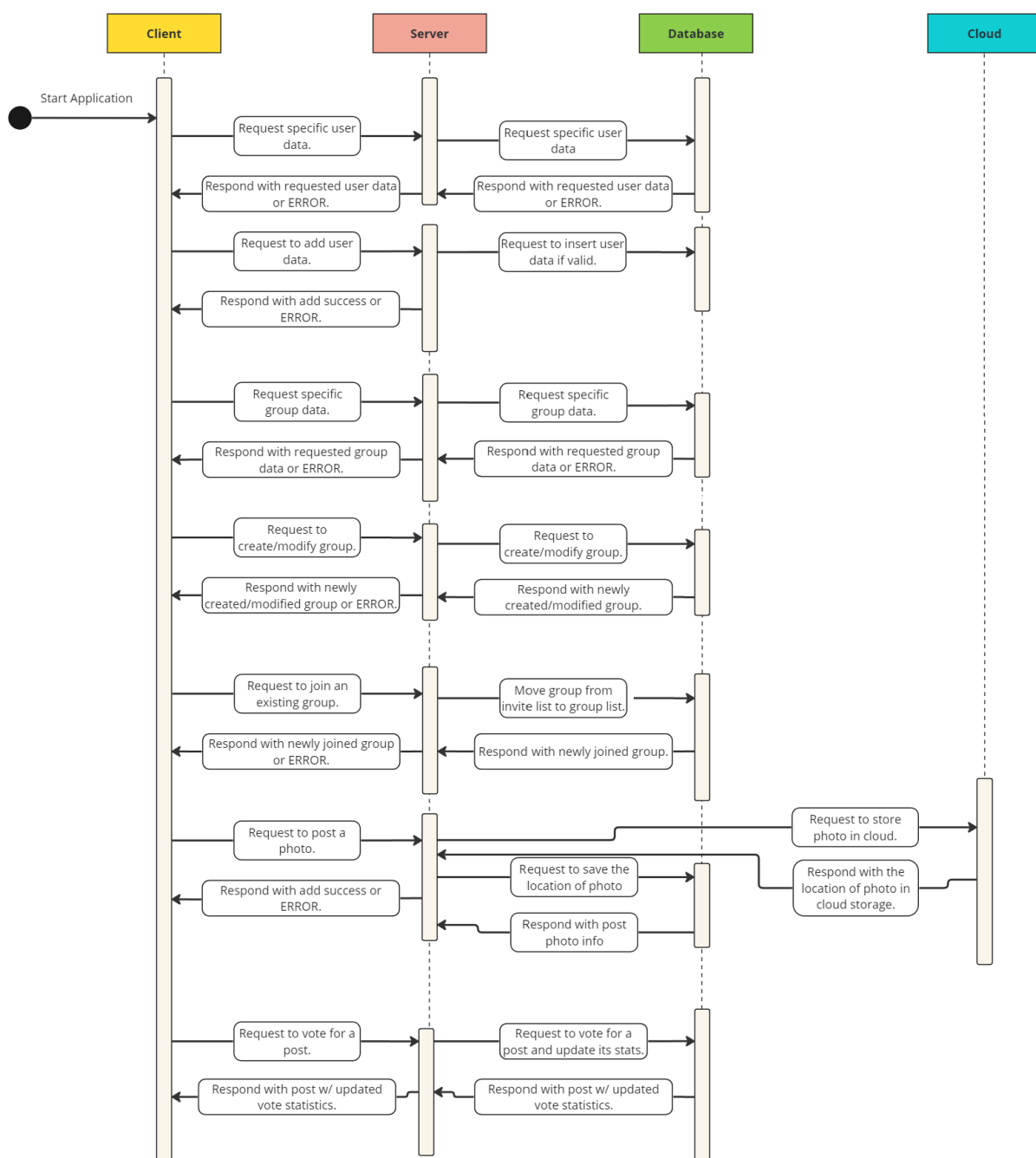
- a. MongoDB's non-relational database will store all of the data used for SnapBattles, such as user information, group information, etc.
- b. The database will respond to queries from the server and deliver the extracted data back to the server.
- c. Due to the persistent nature of the database, the server can be stopped, started, and duplicated to meet client demands without corruption of data.

4. Cloud

- a. The cloud will store all the pictures associated with each post; the location of these photos in relation to each post will be stored in the database.
- b. When the frontend wants to upload a photo, it first sends the photo to the backend server, and the backend server uploads the photo to our cloud server.
- c. When the frontend wants to retrieve a photo, it requests the photo from our backend server. Our backend photo gets the photo from the cloud and then sends the photo to the frontend.

Sequence of Events Overview

The sequence diagram below shows a high-level overview of how the client, server, database, and cloud interact. The sequence is initiated by a user starting the mobile app. If the user wants to retrieve information about other users, groups, posts, etc., it does so by requesting this information from the server. Vice versa, if the user wants to upload information about their groups, posts, comments, etc., it does so by sending this information to the server. The server then manipulates this information according to protocol and stores this information in the database or the cloud for image data.



Design Issues

Functional Issues

What information do we need for users to create an account?

- a. Unique username and password only
- b. Username, password, user ID
- c. Unique username, password, email
- d. Username, password, email, user ID
- e. Unique username, password, email, phone number

Decision: Unique username, password, email

For normal account creation, we need a username and a password for security and identification. Furthermore, we thought that we could potentially be adding some sort of recovery method if the user were to lose his/her password to their account, and we believed that sending an email would be easier than sending a text message. Having both would be overboard since email and phone numbers serve basically the same purpose. Moreover, unique usernames are not hard to require from users, and they are also not hard to store and maintain in databases. We decided not to use unique user IDs for each account since unique usernames can serve the same purpose.

How will the prompts be generated across the app?

- a. All groups will have the same prompt on the same days
- b. Each group will have a different prompt on the same days

Decision: Each group will have a different prompt on the same days

Since we have each group be able to choose what time the prompts are being released, if the prompts were the same for each group, the users in groups with the earliest time would see the prompts first, and for every other group they would be in after that, the excitement of seeing a new prompt would be gone. It would also be unfair to those who would see the prompts later since they would get less time to think about how to respond to the prompts.

What group permissions are administrator-only?

- a. Group settings, kicking users, inviting users
- b. Group settings, kicking users
- c. Group settings

Decision: Group settings, kicking users

Our app has it so that users can only be invited to groups by pre-existing friends. If administrators were the only ones that could invite users, they would have to go and friend every

single person that people would want in the group, so we decided to give all users permission to invite people to groups. For kicking users, we thought that it would be a dangerous permission for everyone to have, especially if immature users would attempt to kick everyone out for no apparent reason; therefore, we decided to limit it to administrator permissions.

How will the voting periods work?

- a. Opens when the submission period closes, open until the next prompt is released
- b. Opens when the submission period closes, open for a select number of hours
- c. Opens at a specific time, open until the next prompt is released
- d. Opens at a specific time, open for a select number of hours

Decision: Opens when the submission period closes for a select number of hours

We decided on opening the submission period for a select number of hours and not until the next prompt is released because of the weekly voting. Weekly voting must happen after the daily voting on Sunday (the last day of the week), so the daily voting cannot be open until the next prompt is released and must have some defined open period. We also thought it would be best to open the voting period right as the submission period closes, since this would create a sense of continuity between the submission period and the voting period.

How should prompts be created?

- a. ChatGPT creates a random prompt
- b. Scrape existing prompts from the internet
- c. User creates a prompt

Decision: ChatGPT creates a random prompt

Our app wants to give users a creative daily prompt that users can actually enjoy answering. As a powerful language model, ChatGPT can generate a wide range of creative prompts. By controlling the parameters for generating prompts, we can achieve prompts with diverse themes and topics. As integrating ChatGPT prompts into our app is straightforward, it will save our efforts and time to generate daily prompts.

Non Functional Issues

What backend language/framework should we use?

- a. Spring Boot (Java)
- b. Node.js (JavaScript)
- c. Django (Python)

Decision: Node.js (Javascript)

With our front-end being designed in React Native, which is based in Javascript, creating our backend using Node.js, which is also based in Javascript, will allow for greater code consistency and collaboration between both our front-end and back-end developers. In addition, Node.js has a huge set of third-party libraries that we can import and use so that we do not have to code various functionality from scratch. For example, the Axios package allows us to send and receive HTTP requests easily, as Axios has many useful built-in features, such as throwing errors on specific response codes and auto-parsing JSON. In addition, the majority of our team members are familiar with Javascript, meaning that there will be less time learning the language/framework and more time building our application.

What web service should we use?

- a. Google Cloud Services
- b. Amazon Web Services

Decision: Google Cloud Services

Both web services support almost equally as many features and are almost equally difficult to use, but we decided to use Google Cloud Services since we could potentially link its services to Google accounts, while Amazon Web Services does not have that functionality. This could be helpful while we test accounts.

What frontend language/framework should we use?

- a. raw HTML + JavaScript
- b. React Native
- c. Flutter
- d. SwiftUI

Decision: React Native

React Native is easy to learn and use. It also allows us to write a single codebase and deploy it across both IOS and Android platforms, which saves us a lot of time compared to developing separate code for each platform. React Native uses components that have their own rendering and logic, making it easier to implement a class-based architecture and also allowing us to better divide up workload on the client side. React Native has a Hot Reloading feature, which will

improve our work productivity by allowing us to see changes from code fixes immediately without needing to rebuild the entire app.

What database should we use?

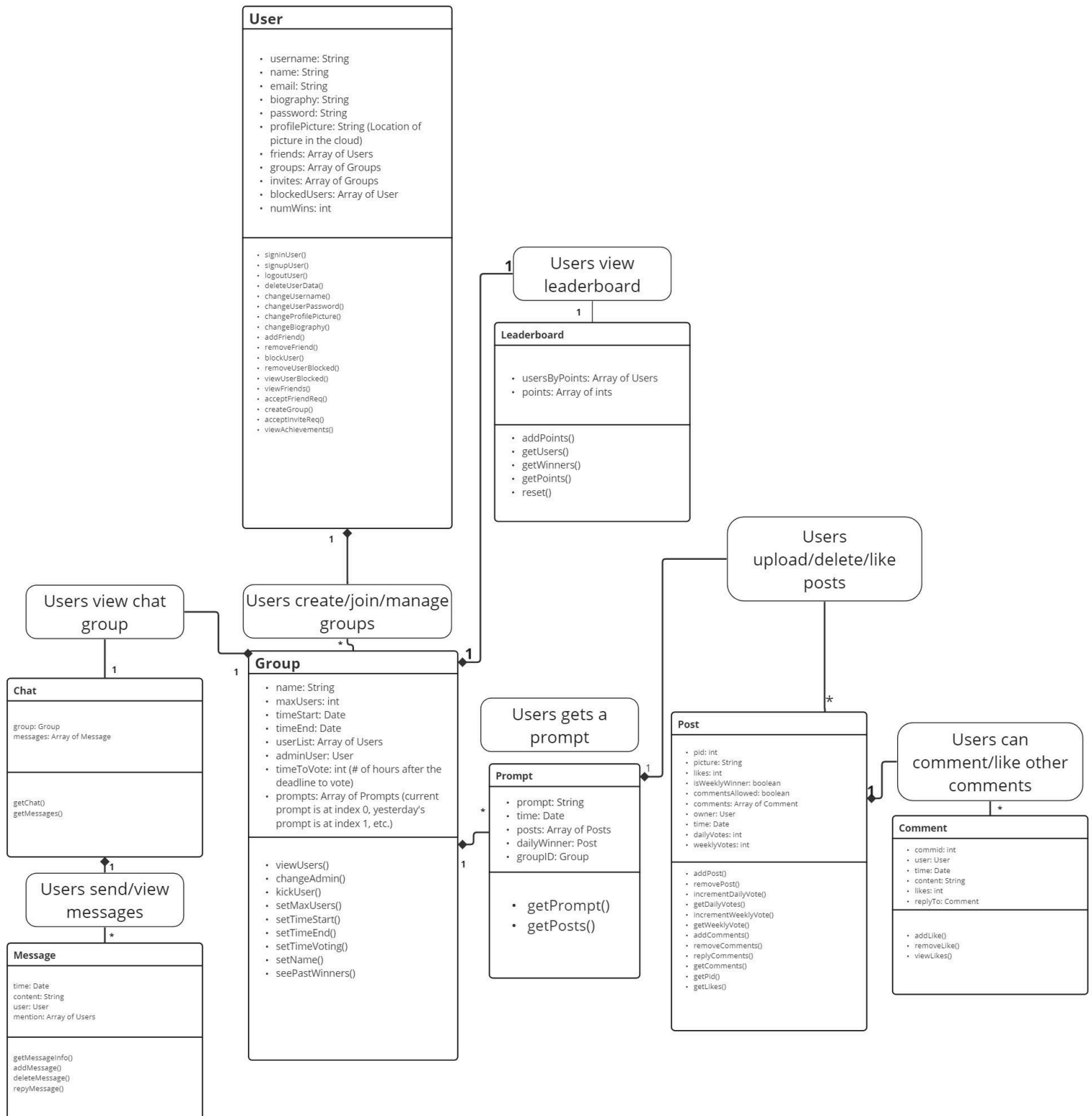
- a. MySQL
- b. MongoDB
- c. PostSQL

Decision: MongoDB

MongoDB is a NoSQL database, meaning that we can store complex, nested data structures without having to worry about designing a perfect “blueprint” or schema before working with our database. In addition, MongoDB stores information in JSON format, which is perfect for us since we are using Javascript. Objects in Javascript are stored in a format similar to JSON. Moreover, objects in Javascript can be easily transformed into JSON using the `JSON.stringify()` method, and JSON strings can be easily transformed into objects in Javascript using the `JSON.parse()` method.

Design Details

Class Design



Description of Data Classes and their Interactions

User

- Represents a user of the application.
- Created when a new user signs up to use the application.
- Each user will have a username, password, and email for login purposes.
- Each user must have a unique email to differentiate themselves from other users.
- Each user may either create a new group or join an existing group through an invitation.
- A user that creates a new group gains administrator privileges in the group they create (editing group settings and removing people from the group).

Group

- Represents a group of users in which all basic interactions between friends occur (posting photos, voting for photos, liking/commenting on photos).
- Created when a user decides to create a group.
- Each group has an array of users, where each user belongs to the group.
- Each group has an array (implemented as a queue) of prompts, where the prompt at index 0 correlates with the pseudo-randomly generated prompt for that day, index 1 correlates with the prompt for yesterday, index 2 correlates with the prompt for the day before yesterday, and so on.
- Each group has various settings, including the time window users have to post photos and the number of hours after the photo submission deadline in which users can vote for photos (weekly voting occurs directly after daily voting).
- Each group has a single administrator user who can edit the settings for the group they are in.

Prompt

- Each prompt contains an array of all the photos that respond to it
- The actual prompt itself is created daily pseudorandomly using the ChatGPT API for a given group
- Each prompt contains a post that had the most number of votes for the daily, a.k.a. the “daily winner” (“the weekly winner” information is stored differently; see Post section)

Post

- Represents a user’s post/response for a prompt in a group
- Created by a user in the group
- Users in the same group as the user who posts will be able to view and interact with the post through likes and comments

- Contains information about the number of daily votes it has (and the number of weekly votes it has, which is not always applicable)
- Contains a boolean to state whether or not the picture is a weekly winner

Chat

- Represents the sole chat room of a group
- Created automatically when a group is created
- Users in the group are able to send and view messages
- Includes an array of Message objects to show all the messages in a chat to users

Leaderboard

- Represents a leaderboard of a group
- Created when a group is created
- Keeps track of the number of points of all users in a group
- Keeps users in sorted order based on the number of points they have from most to least
- Users can view the rankings of users based on the points earned from daily and weekly voting.
- Leaderboard will reset every two weeks

Message

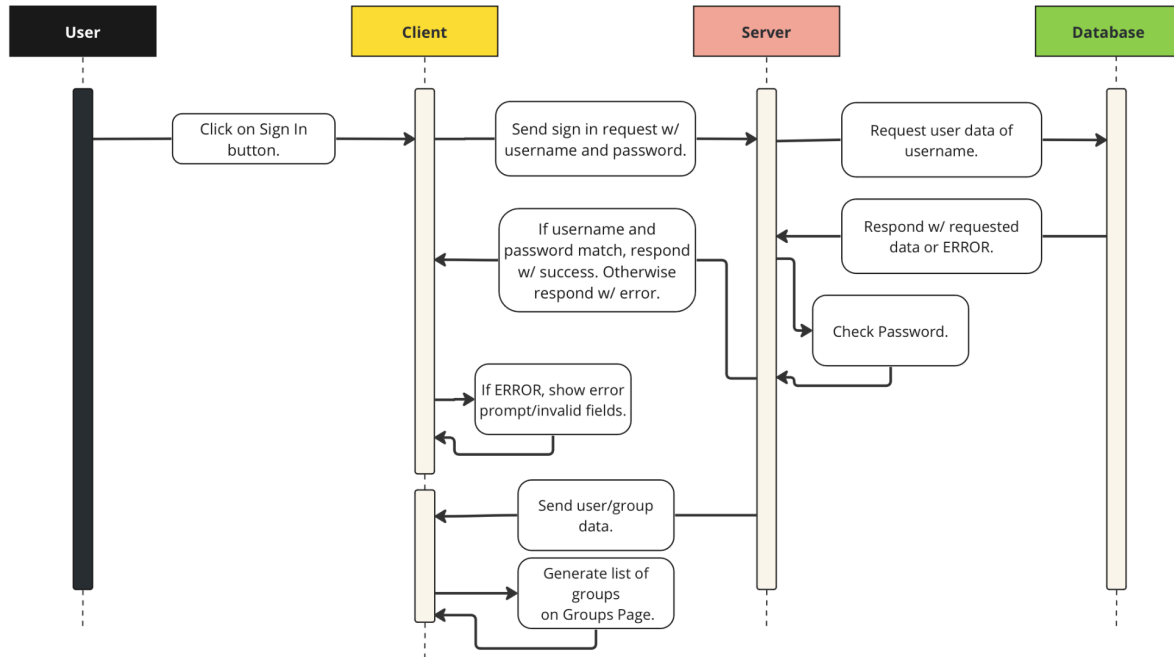
- Represents a message sent by a user
- Created when a user sends a message to a group chat
- Includes the date when it was created/sent
- A user can mention other users by using @

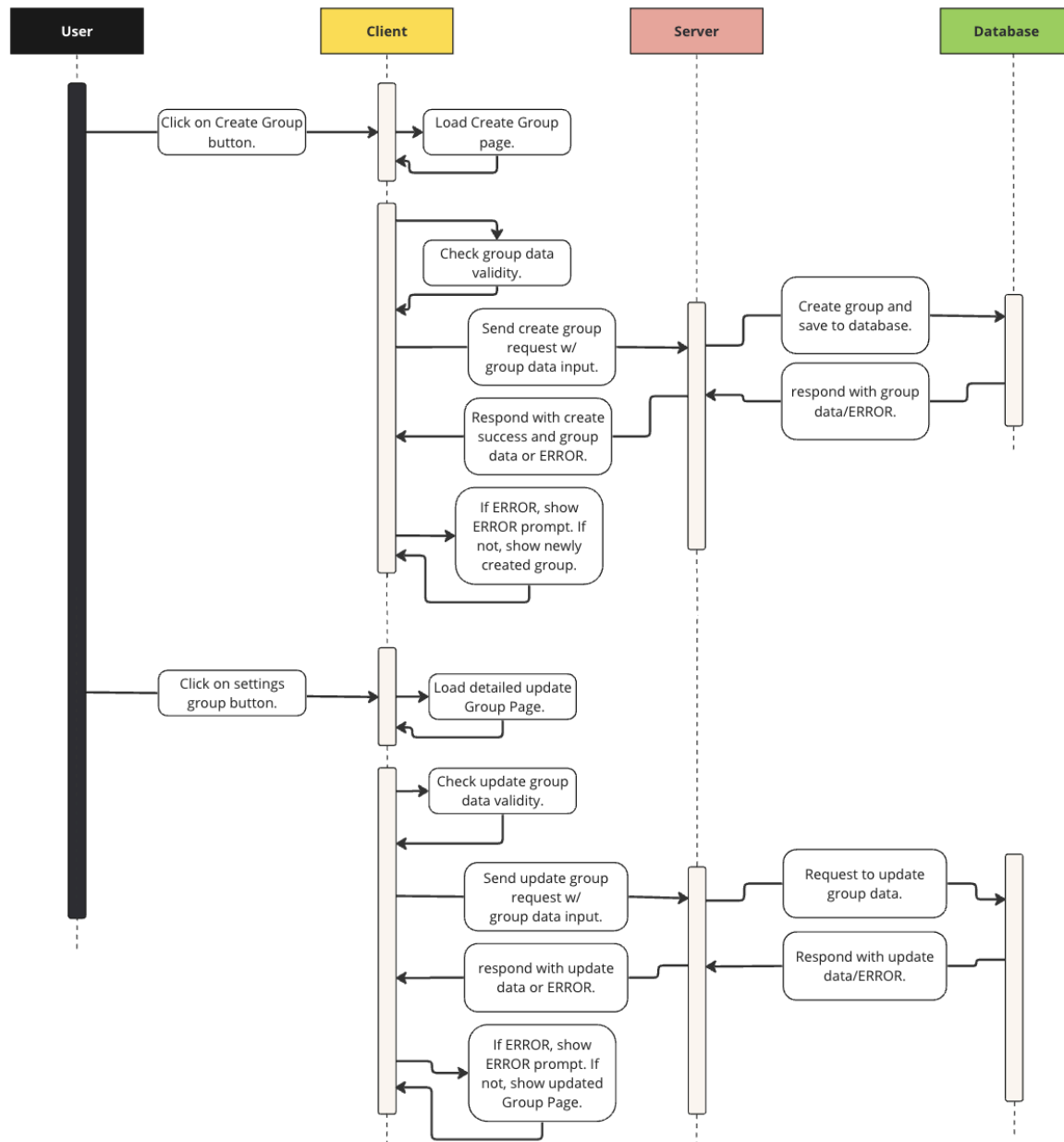
Comment

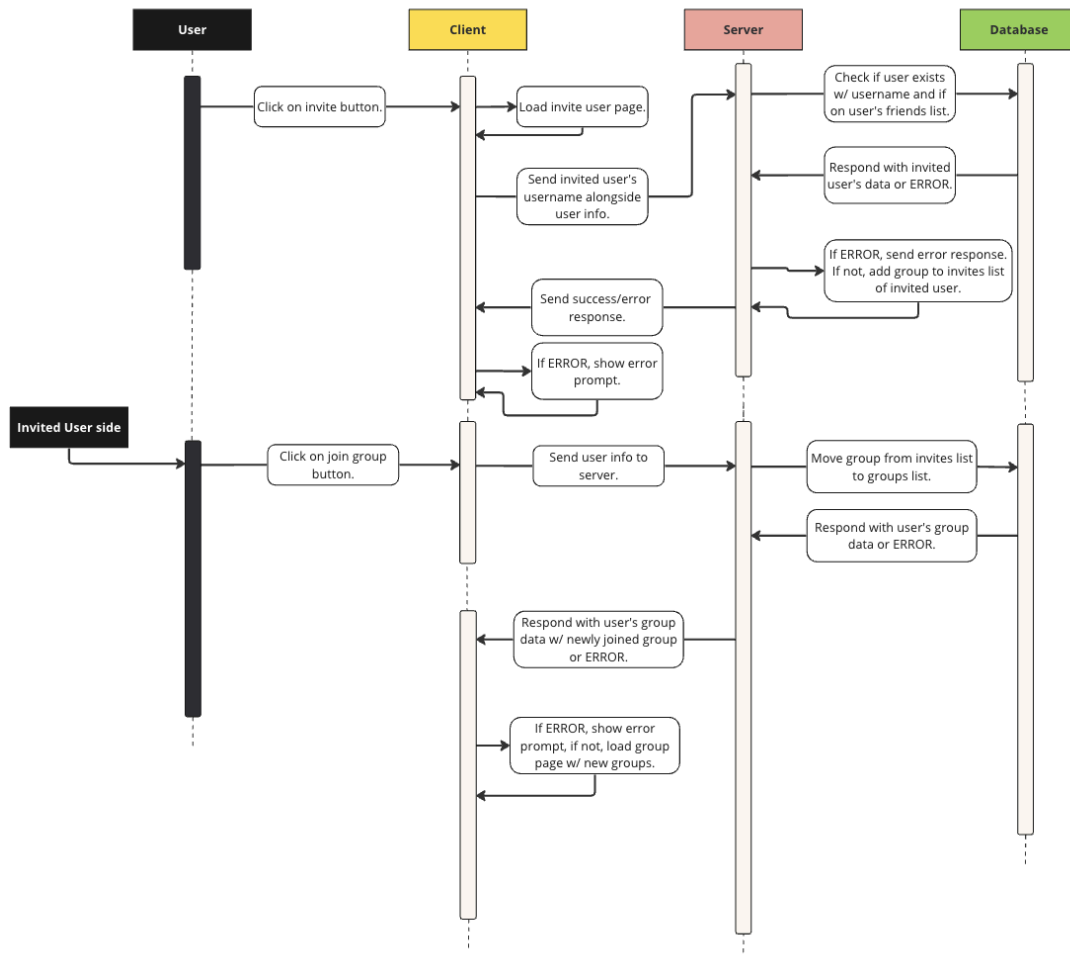
- Represents a comment on a post
- Created when a user writes a comment to a post
- All users in a group can view a comments and interact with the comment by liking and replying
- A user who created a post can delete comments

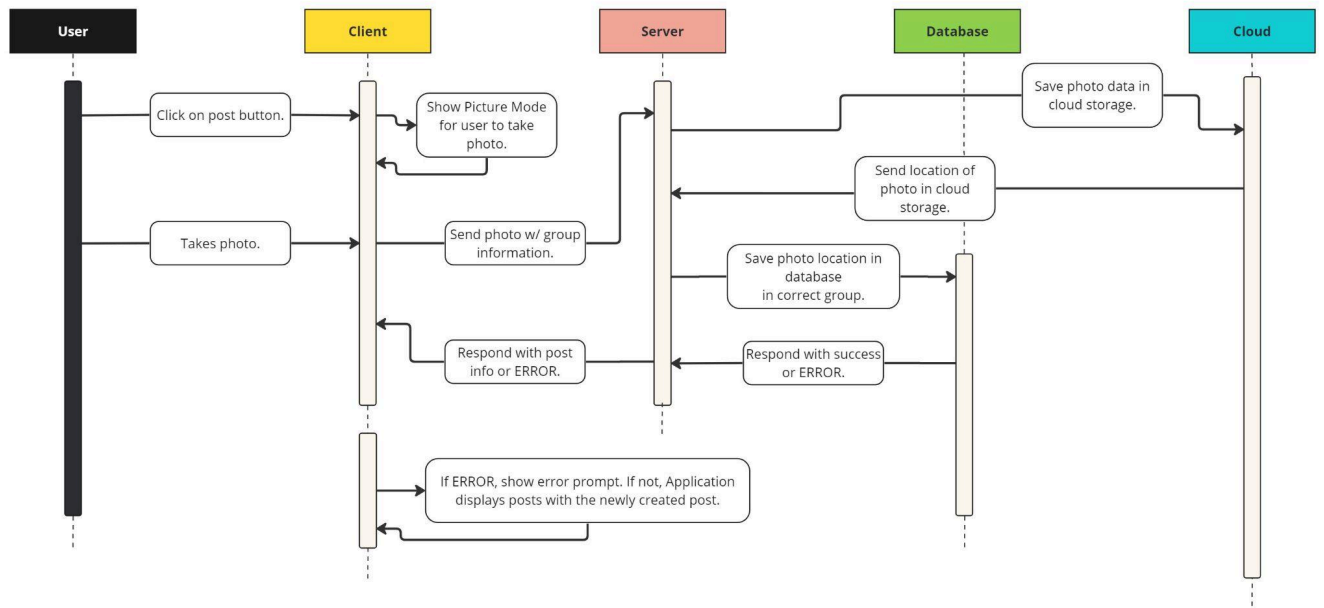
Sequence Diagrams

Sequence of events when users login

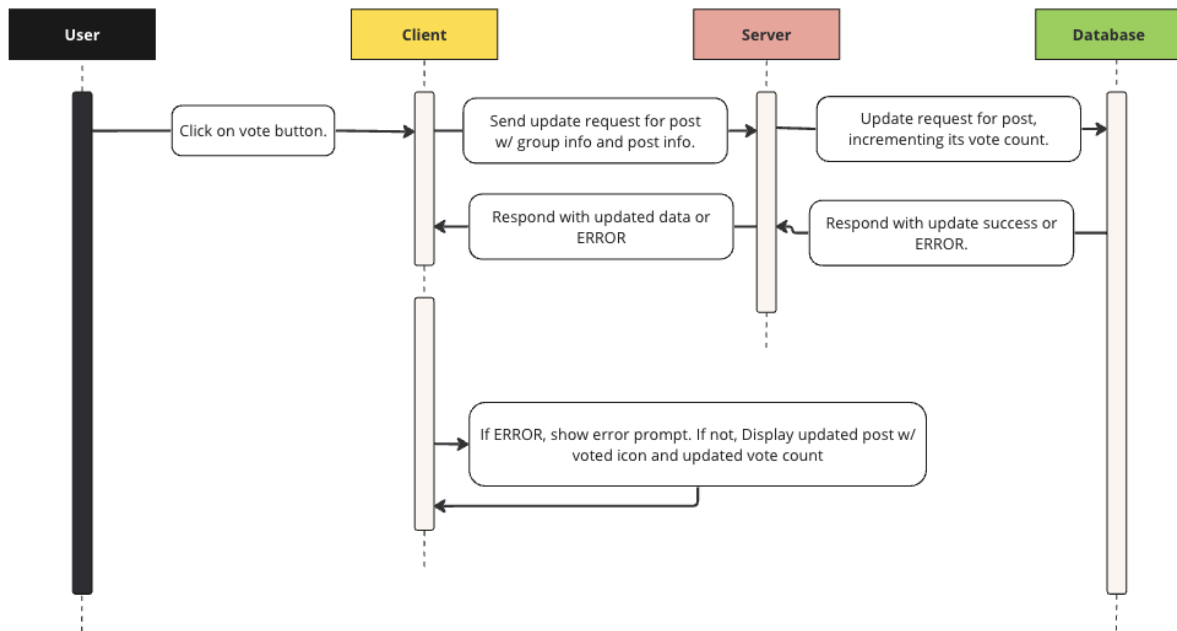


Sequence of events when users create a group

Sequence of events when user joins a group

Sequence of events when user posts a photo

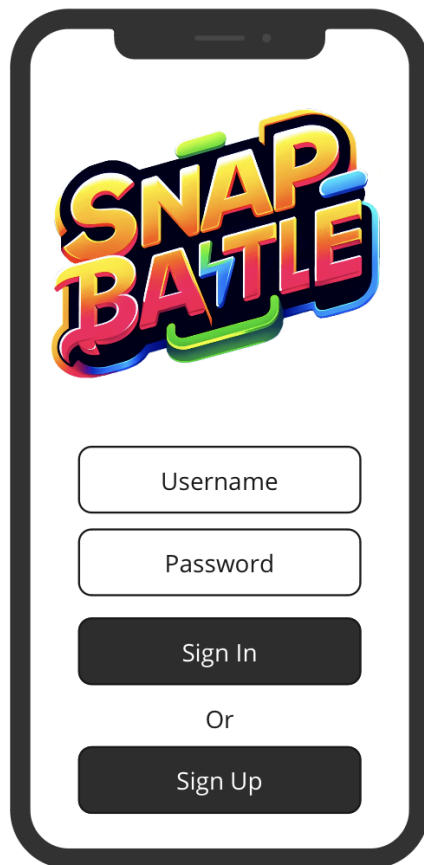
Sequence of events when user votes for a photo



UI Mockup

Our user interface (UI) is clean, stylish, and intuitive – providing a great visual and engaging user experience. We have organized the app into distinct pages where each page provides specific functionalities, such as creating an account, editing profile settings, or taking a picture within the app. The design ensures a user-friendly navigation system – improving the entire interaction with our app.

Authentication

A mobile app mockup for the Sign-In screen. At the top is a colorful, 3D-style logo that reads "SNAP BATTLE". Below the logo are two white input fields with rounded corners, labeled "Username" and "Password". Under these fields is a dark grey button with the text "Sign In". Below the button is the word "Or" in a small font, followed by another dark grey button with the text "Sign Up".

SNAP BATTLE

Username

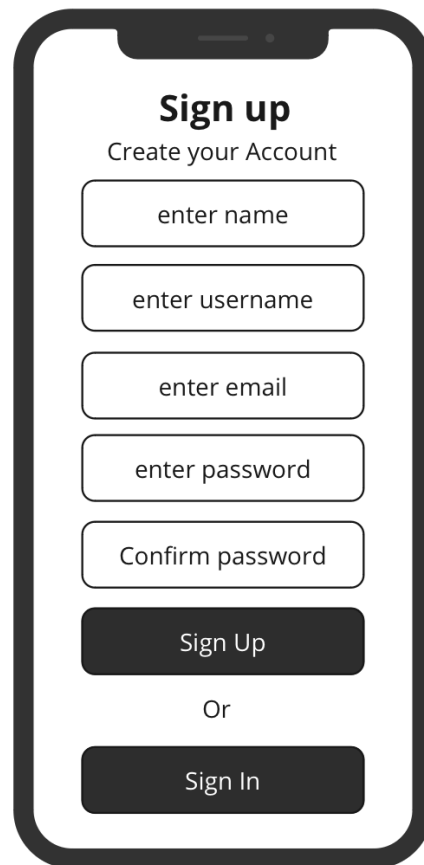
Password

Sign In

Or

Sign Up

Sign-In Screen

A mobile app mockup for the Sign-Up screen. At the top, the text "Sign up" is displayed in a bold font, followed by the subtitle "Create your Account". Below this are five white input fields with rounded corners, labeled "enter name", "enter username", "enter email", "enter password", and "Confirm password". Under the "enter password" field is a dark grey button with the text "Sign Up". Below the button is the word "Or" in a small font, followed by another dark grey button with the text "Sign In".

Sign up

Create your Account

enter name

enter username

enter email

enter password

Confirm password

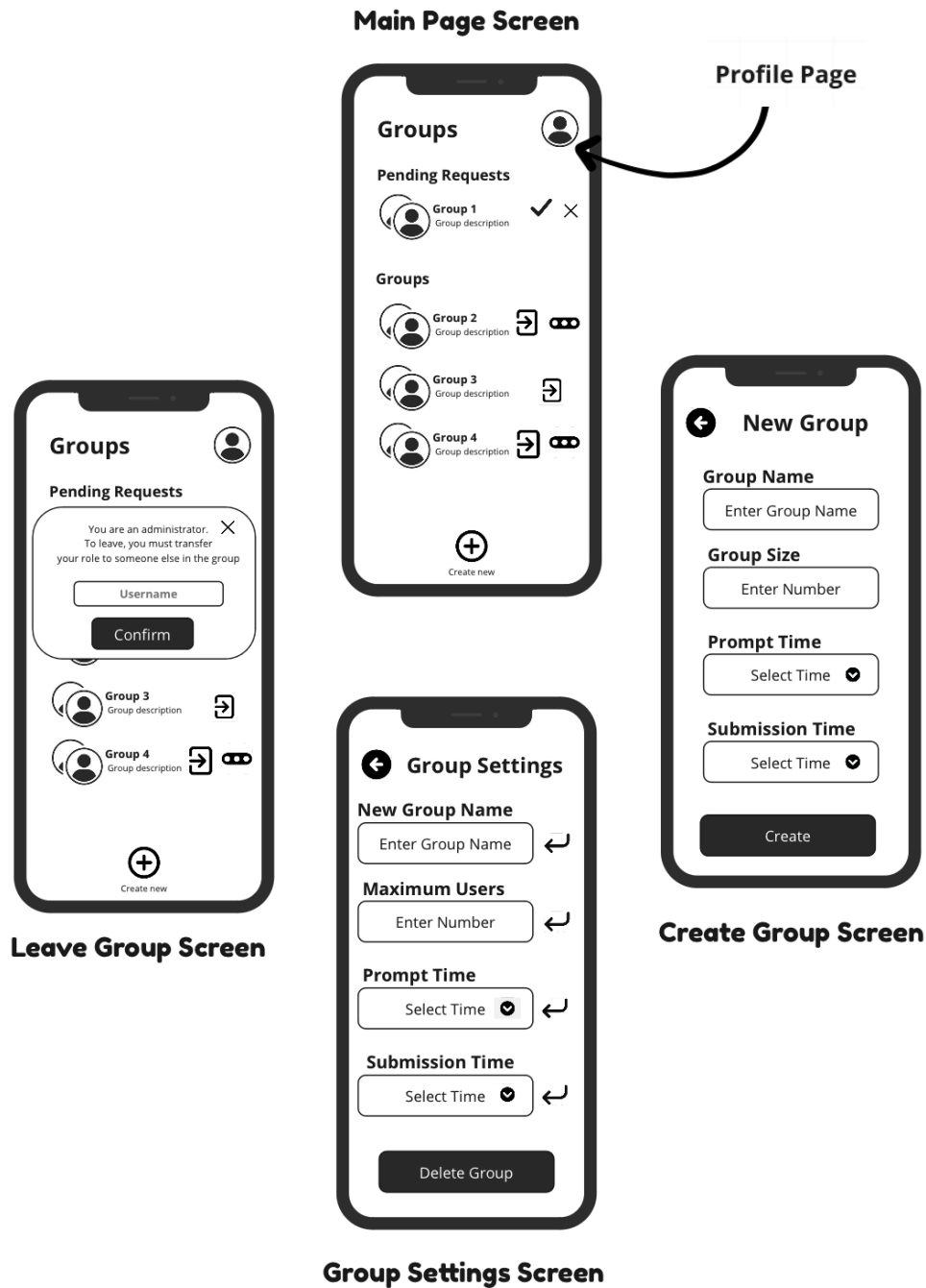
Sign Up

Or

Sign In

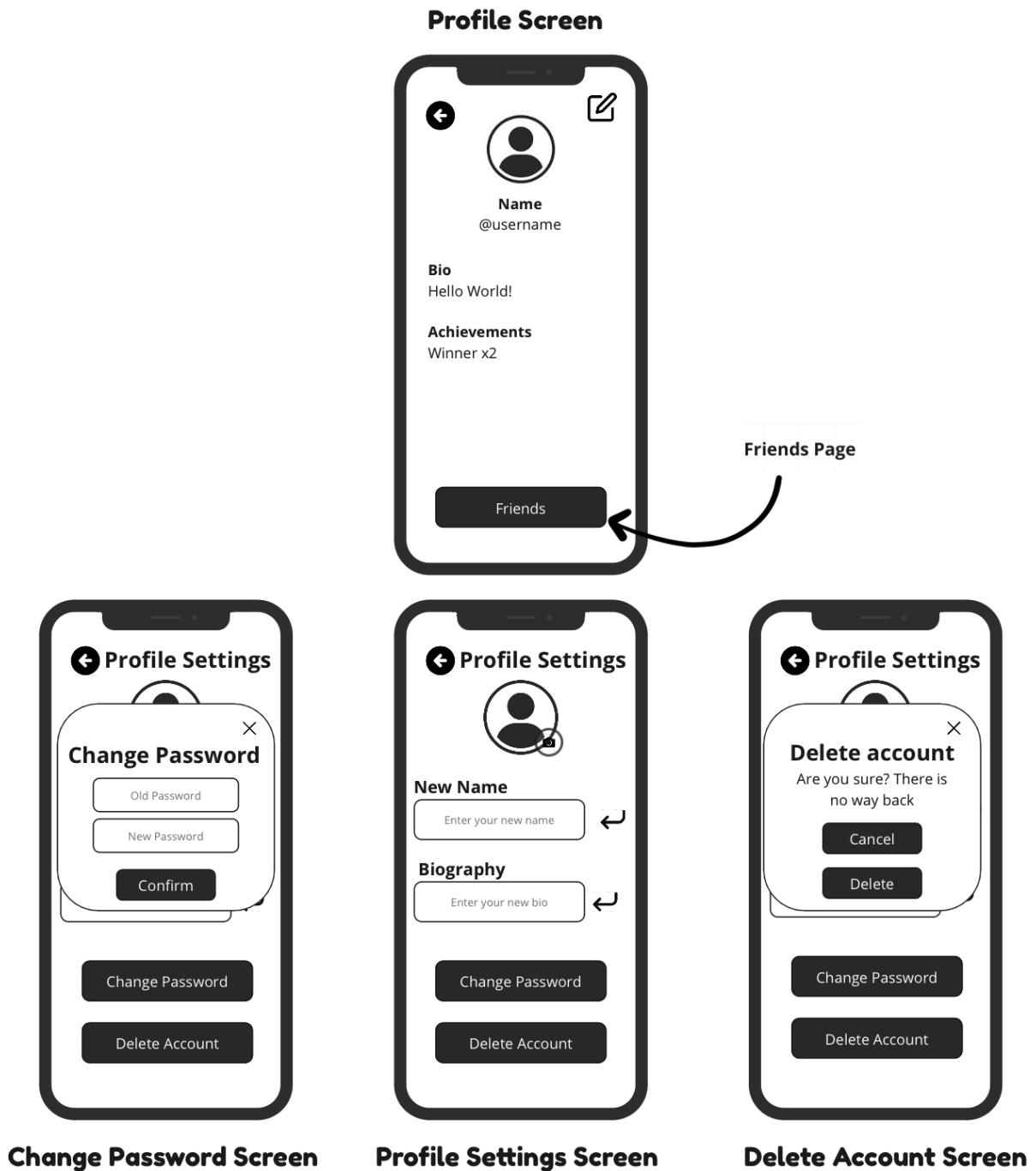
Sign-Up Screen

On the authentication page, users can create an account or sign in to our app and begin using its different functionalities.

Main Home Page

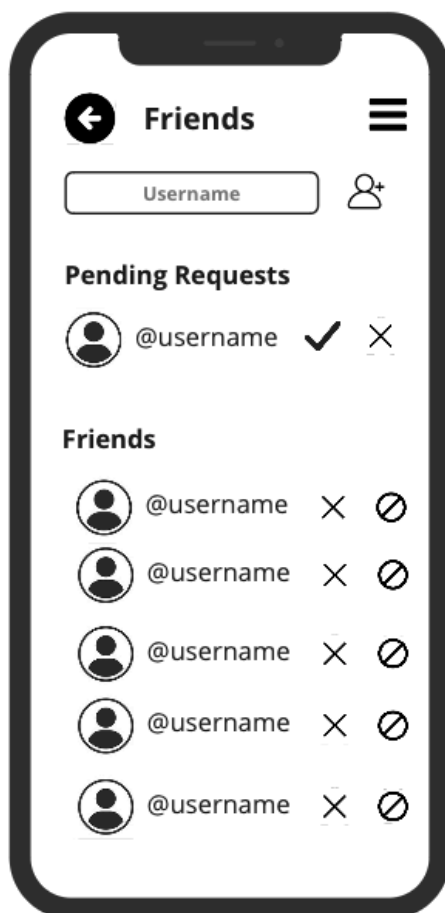
From the top, the main home page contains a list of pending group invites, if the user has any. This will be followed by the current list of groups that the user is already part of and each group's respective group settings; selecting a group proceeds the user into the respected group home page (elaborated on page 26). Furthermore, users can choose to create a new group. Clicking on the profile icon takes users to their personal profile page.

Profile Pages

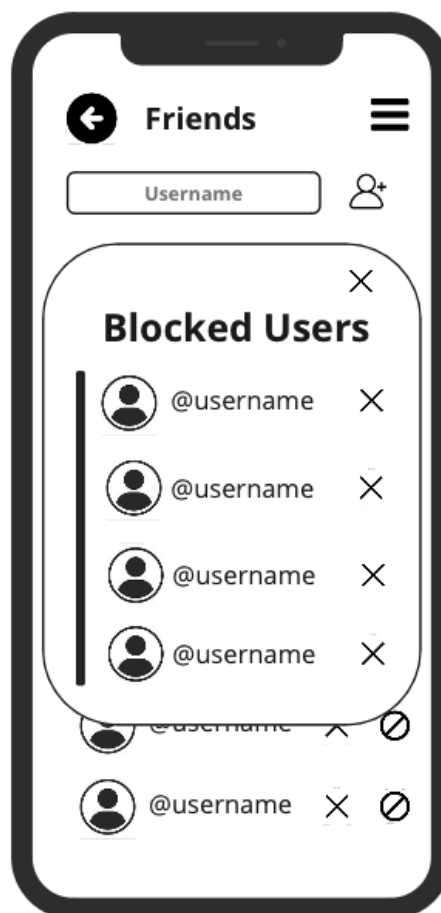


The profile page provides users with the ability to view, modify, and customize their profile. This includes but is not limited to their respective list of friends, account deletion, password and username modification, etc..

Friends Pages



Friends Screen

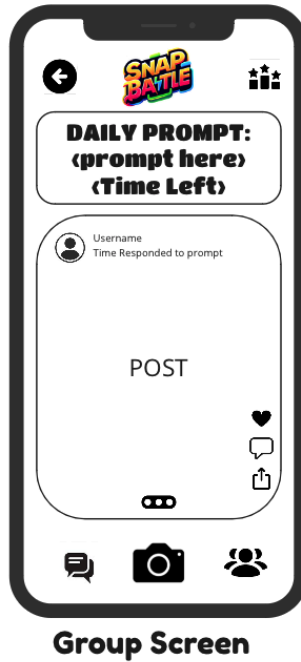


Blocked Friends Screen

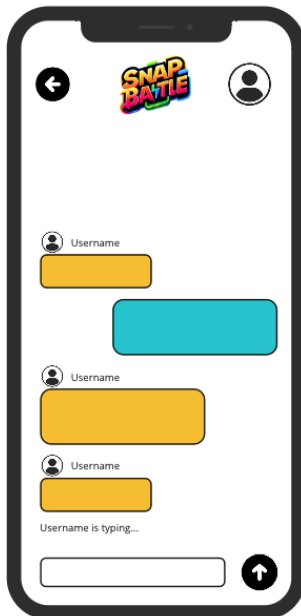
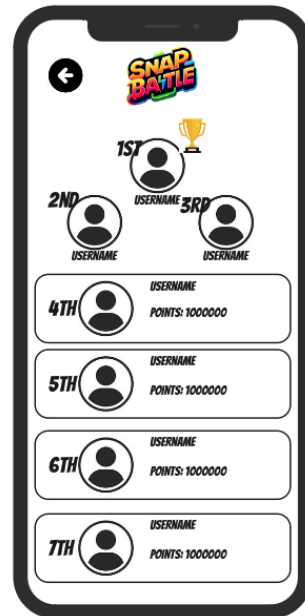
The friends page will allow users to search for, add, and remove friends. Additionally, users can block pre-existing users. Now, we will continue onto the group home page.

Group Home Page + Misc

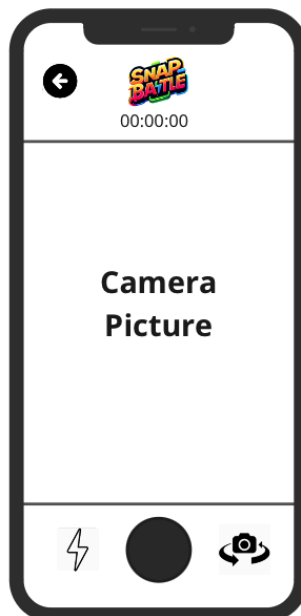
Invite Screen



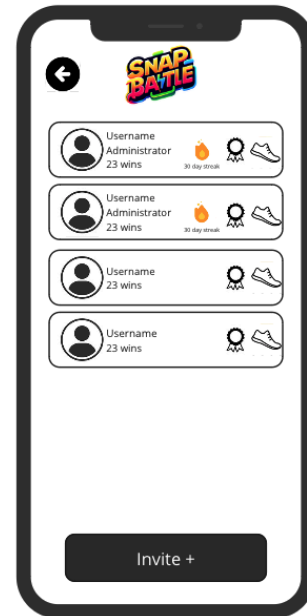
Leaderboard Screen



Chat Screen



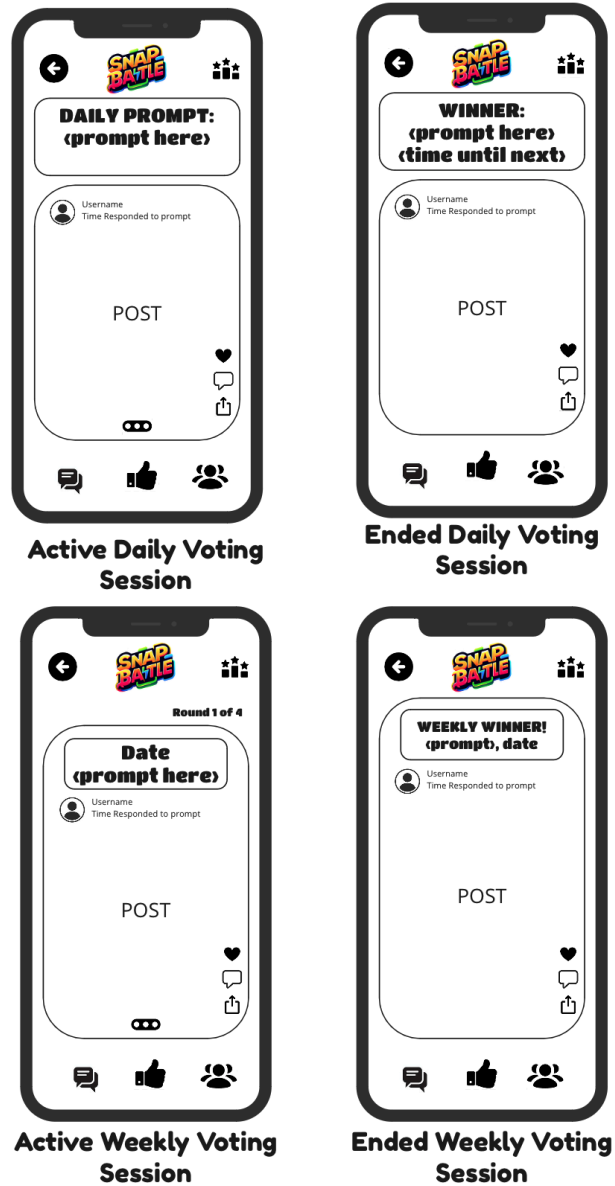
Camera Screen



Members Screen

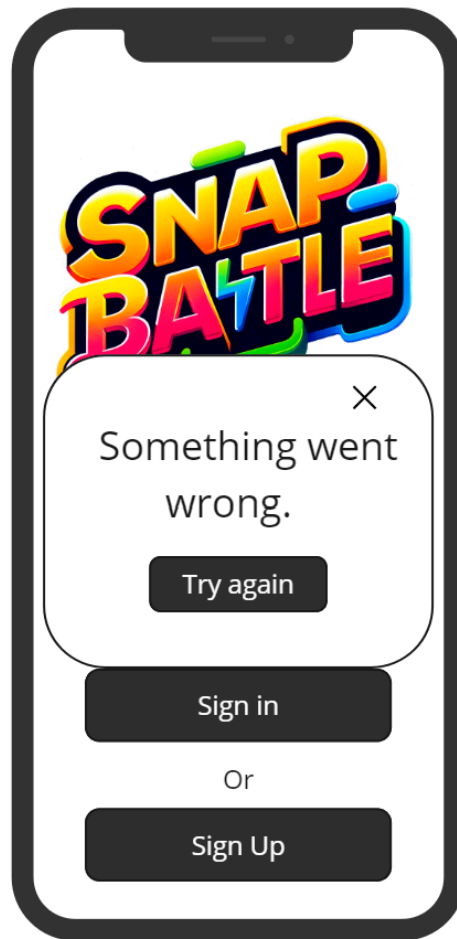
The group home page will offer users the ability to chat, capture their daily snaps, view the leaderboard, and see the members of the group.

Voting Page



Once the submission period ends, users will be able to vote for their favorite entries on the daily and weekly voting pages. Once the voting concludes, the submission with the most votes will be featured for all users to see until the start of the next subsequent submission period.

Error prompt



Error Prompt

The Error Prompt will appear every time an unexpected event occurs within the app, such as a lost connection between client-server and a failure to upload/retrieve image data.