SW Engineering CSC648/848 Milestone 2

Project title: **Momentus**
Team 4
Matt Bolles: Team Lead
Raza Ali: Github Master
Natalie Christie: Scrum Master
Jo Ey Chong: Front End Lead
Eric Chie: Back End Lead
Adam Belaid: Back End

05/20/2021

Version History

Milestone/Version	Date
M2V1	03/18/2021
M2V2	05/20/2021

Contents

1 Data Definitions V2

- **Post** - an image, video, or text. A post is attributed to the user that originally created it. When displayed inline in the gallery or stream, the post content may be resized or truncated in order to save space and avoid taking up the whole screen. Upon viewing the post individually, content will be displayed in full when possible, only resizing to fit content fully into view of screen. Users will also be able to add a short description for each post.
  - Image post: max size: 10mb, max resolution: 5000px x 5000px, file format: JPG, PNG
  - Video post: max size: 250mb max resolution: 1920x1080 (1080p) file format: AVI, MP4, MOV (will convert if possible for player)
  - Text post: max size: 2mb

  In database as table: Post
- **Collection** - User-defined teaming of user content. Unique to that individual user.
  In database as tables: Collections, Collection_Content
- **Stream** - Where new content from users you follow appears. By default, all content is posted here. Content appears in chronological order. Content in the stream is not added to the gallery by default, but can be added when the post is created, or added to the gallery later.
  Not specifically in database, pulls content from table Post
- **Following** - Subscribing to a user's content. User content will appear in stream of those whole follow them.
  In database as table: Follow
- **Favorites** - a list of content that a user has favorited for later viewing, and to show the content creator they like and appreciate the post. Public by default.
  In database as table: Favorites
- **Comment** - a public message left on a post that appears under the post.
  In database as table: Comments

2 Functional Requirements V2

1. **Posting** - allow users to add photo, video, or text content. Users can decide whether to post to stream, gallery, or both. Users can decide who can see the post, and can add tags and a description to it.
   Priority: 1

2. **Collection** - make collection, add posts to collection.
   Priority: 1

3. **Stream** - this includes text posts, video, and photos. Chronological order for real-time interaction to simulate day-to-day socializing.
   Priority: 2

4. **Search** - User-friendly search function throughout site or specific profile to easily find your and other users' posts, and filter by tags, author, location, date, and other criteria.
   Priority: 1

5. **Commenting** - users can publicly comment on posts made by other users.
   Priority: 1

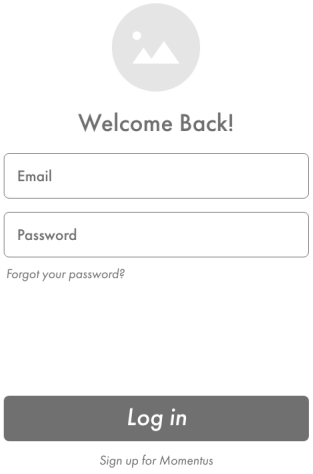
6. **Favoriting** - users can "favorite" a post, which shows the poster than this user likes said post. Favorites are saved and public by default.
   Priority: 2

3 High Level UI Mockups and Storyboards

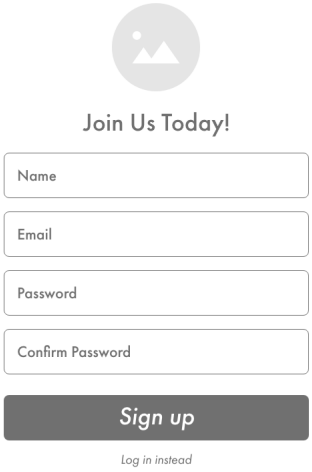
3.1 Mockups and Storyboards

Landing Page - displayed when unregistered user visits site

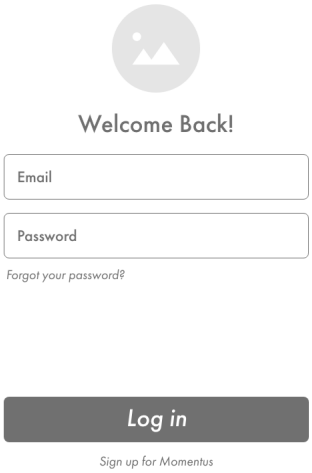
Landing Page - Log In Overlay



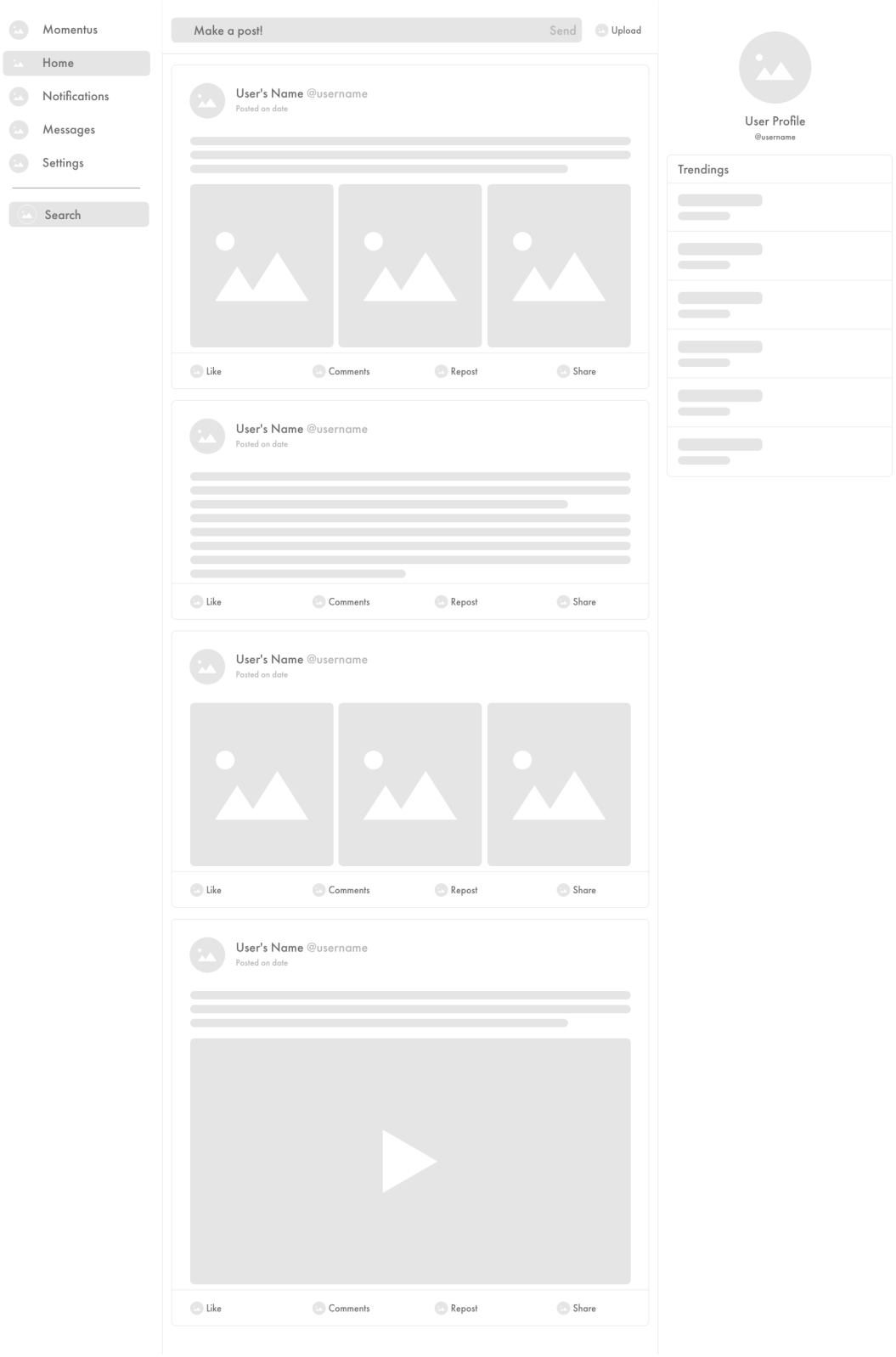
Landing Page - Register Overlay



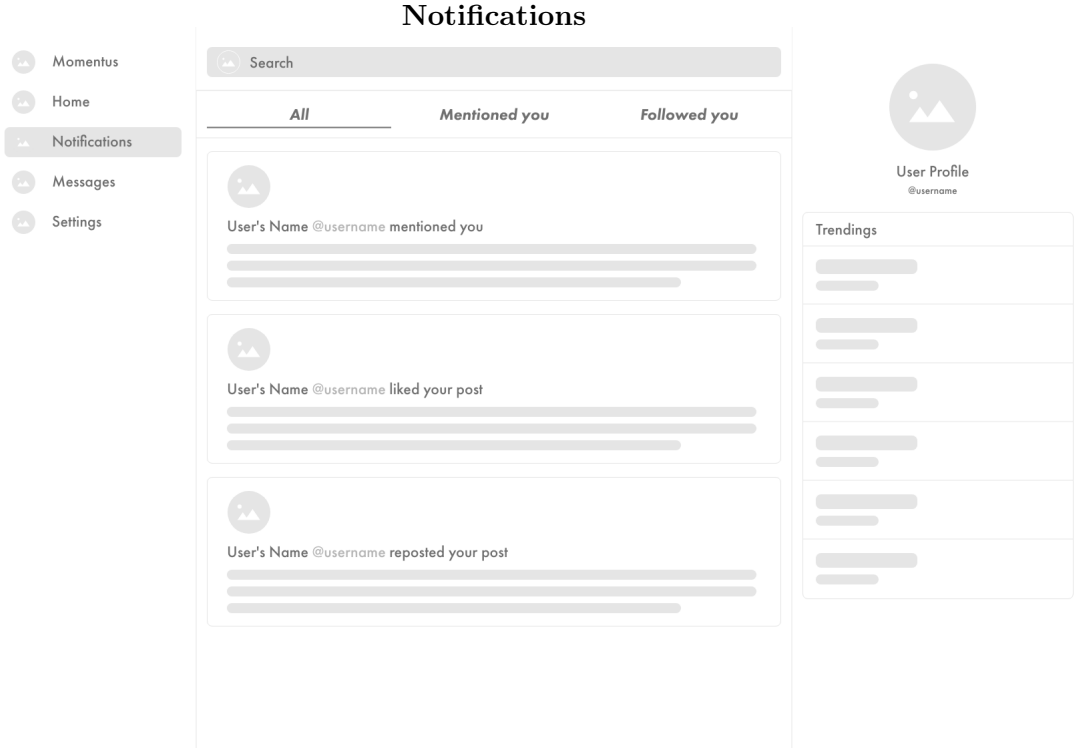
Landing Page - Log In Overlay



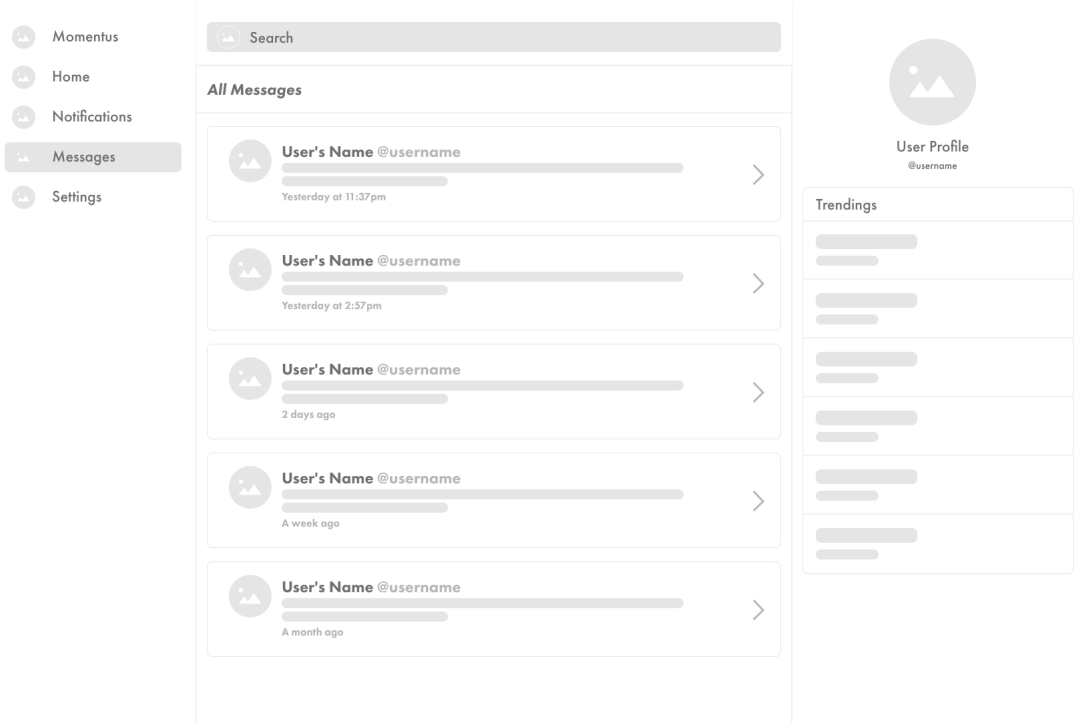
Stream


Momentus
Home
Notifications
Messages
Settings
Search
Make a post!
Send
Upload
User's Name @username
Posted on date
Like
Comments
Repost
Share
User's Name @username
Posted on date
Like
Comments
Repost
Share
User's Name @username
Posted on date
Like
Comments
Repost
Share
User's Name @username
Posted on date
Like
Comments
Repost
Share
User Profile
@username
Trendings

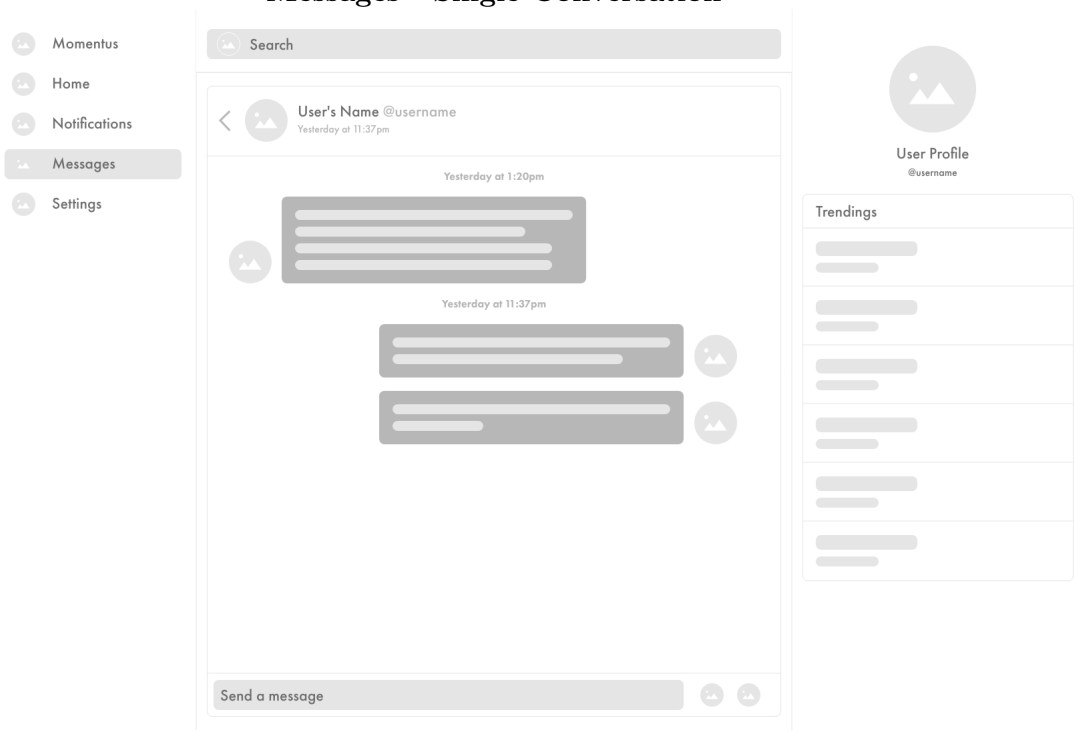
Notifications



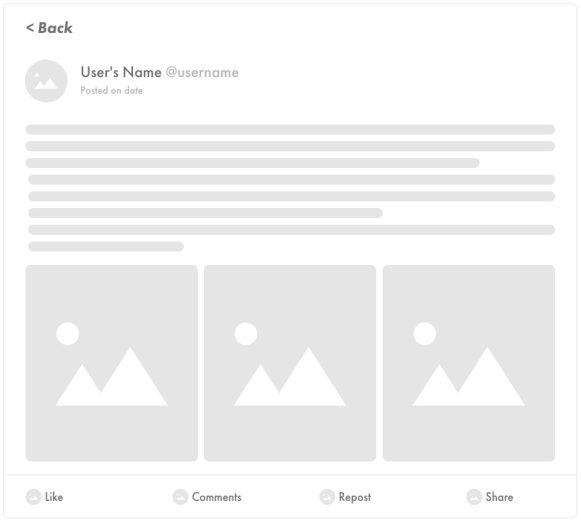
Messages - Overview



Messages - Single Conversation

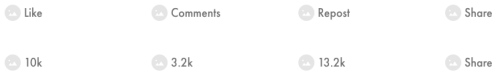


Single Post

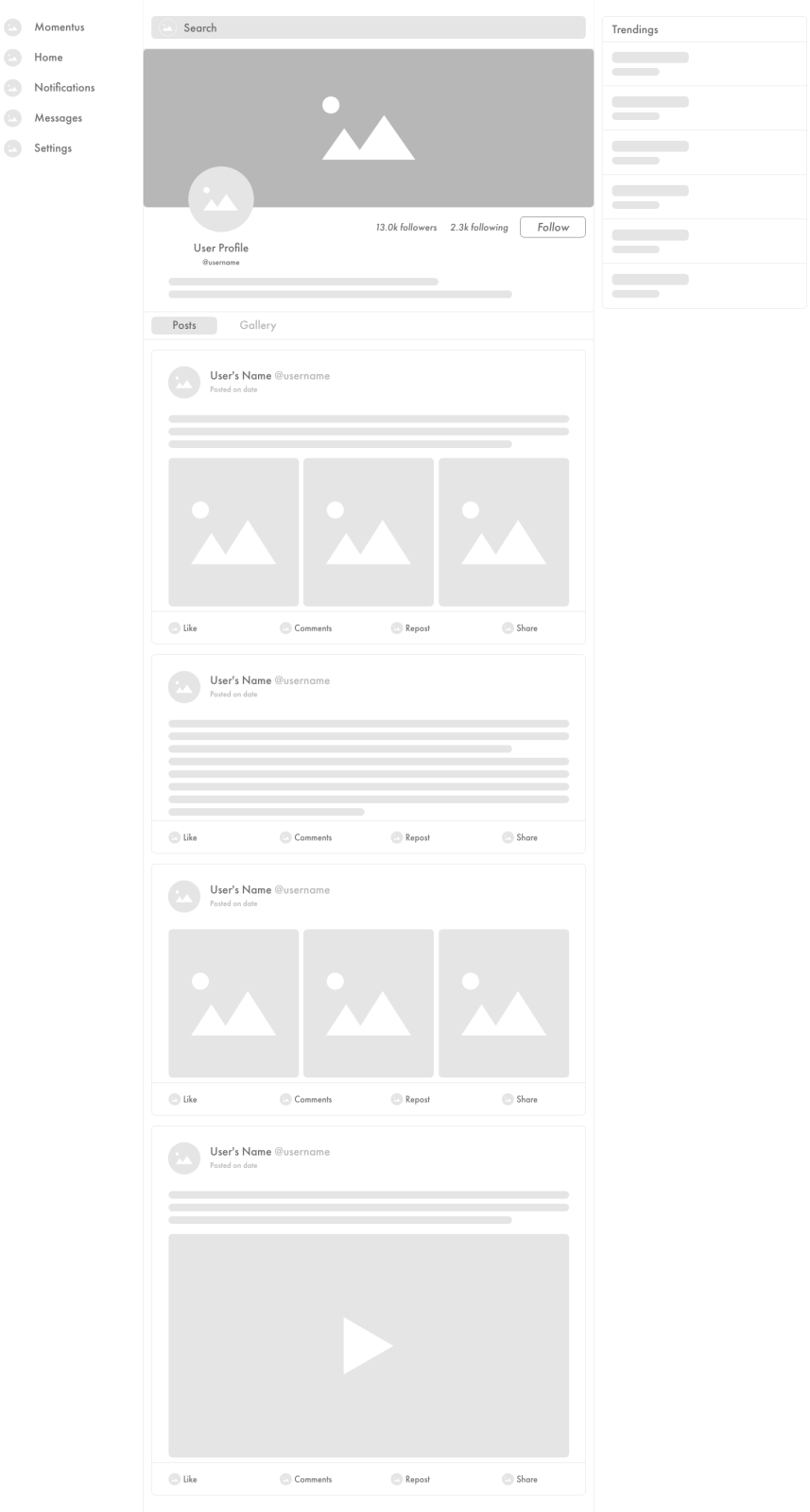


Change to full page

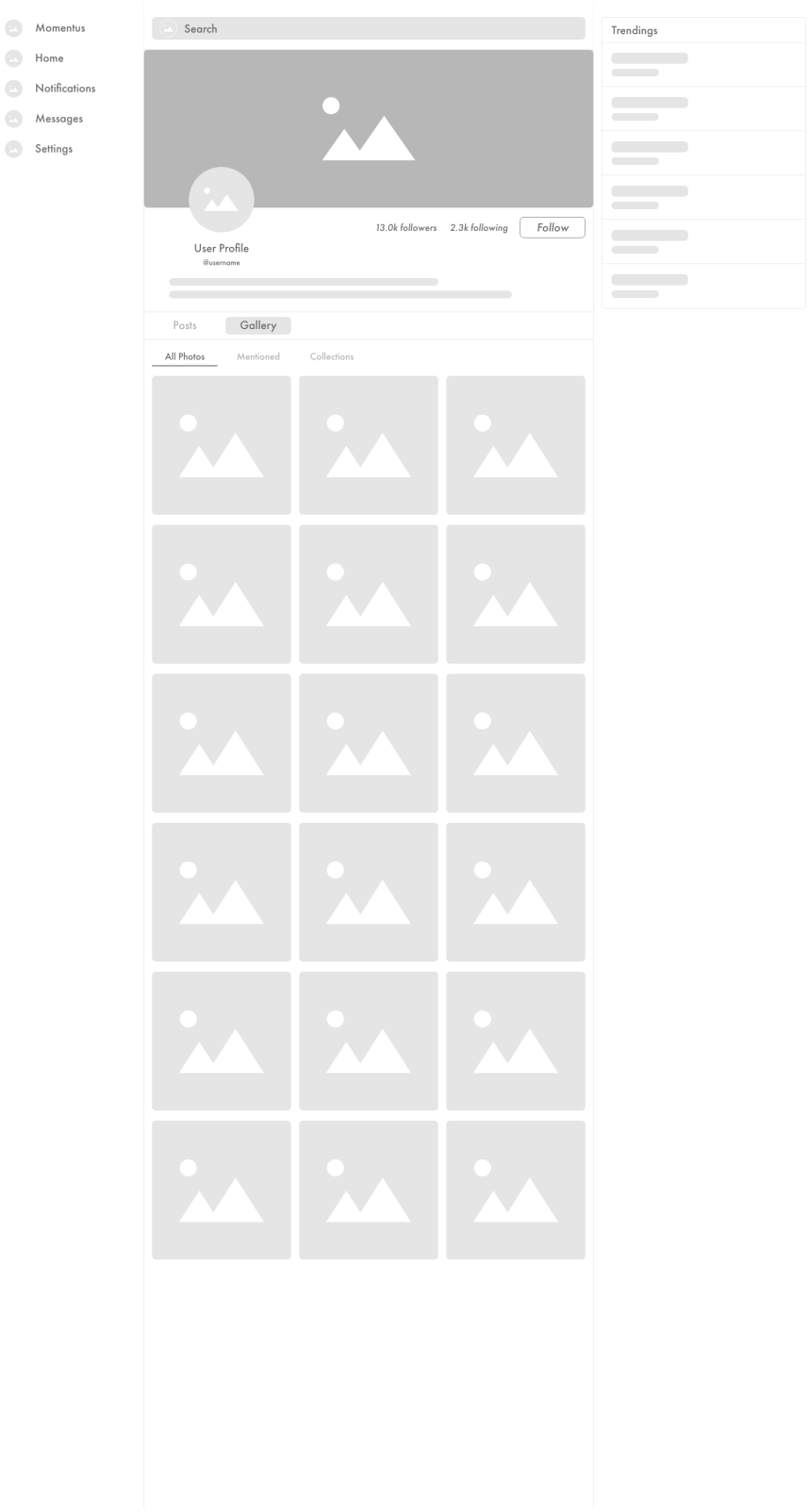
Post - Like / Comment / Repost / Share Panel



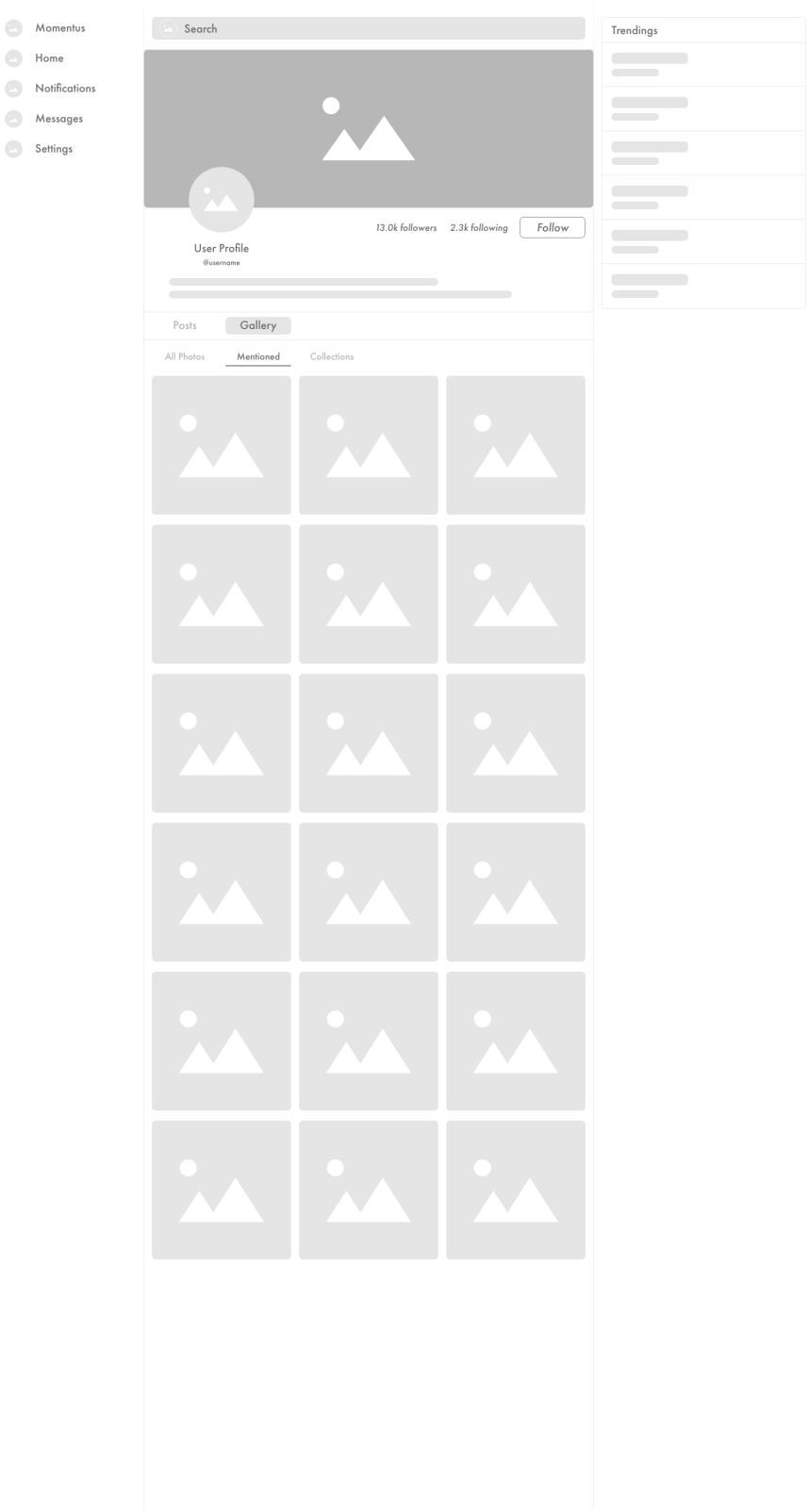
Profile - Most Recent Posts



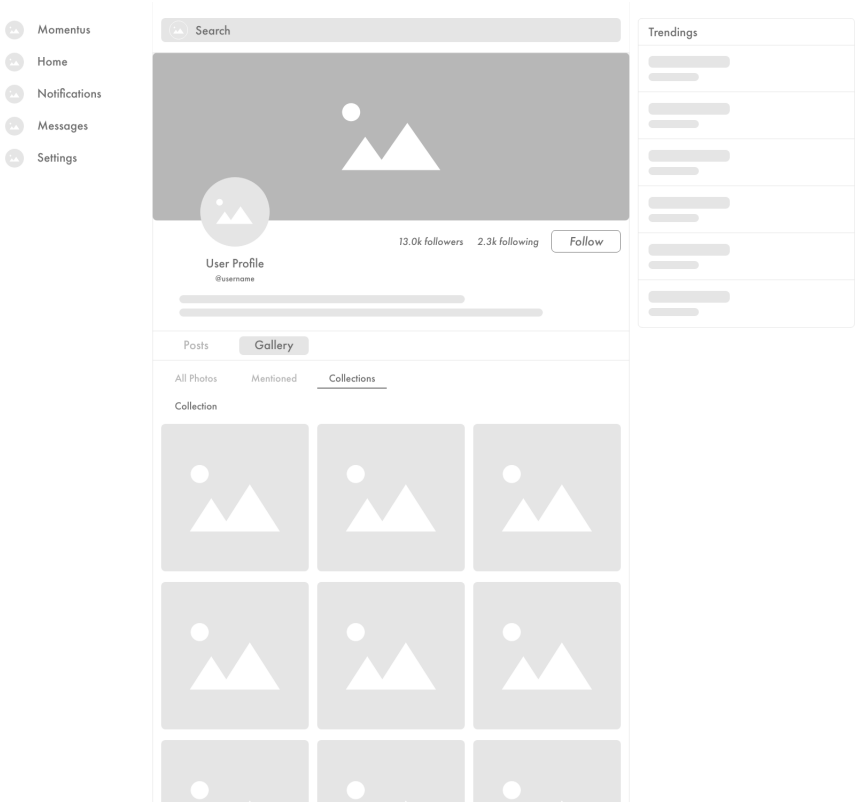
Profile - Gallery - All Posts



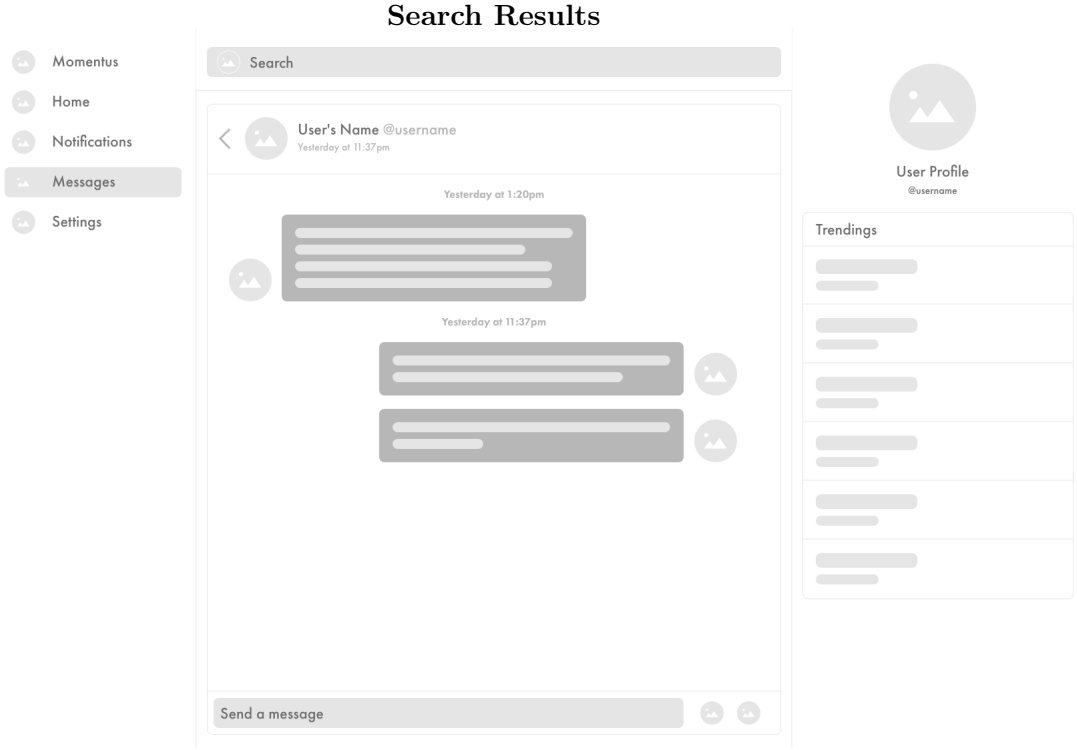
Profile - Gallery - Mentioned Posts



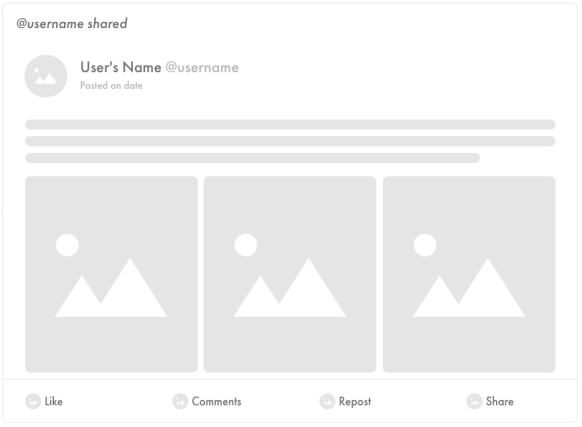
Profile - Gallery - Collections



Search Results



Reposted Post



3.2 UX Review Meeting Minutes

Updates made:

- New adaptations to posts functions (like, comments, share)
- Reviewed mockups of shared post formatting
- Added trending feature
- Updated to add posting feature

To do:

- Update future parameters on search feature
- Update notification page to be more minimal (summary of posts to keep things coherent)
- Updated direct message feature to show text previews
- Direct message landing page?
- Brainstorm setting page
- Set up database fetch
- Attempt to revamp trending to possibly be featured as a "popular post" gallery that will reflect the message of our website
- Add easy direct symbol/link to fellow users' galleries on their individual posts

4 High Level Architecture and Database Organization

4.1 Database Structure

Our database schema and organization is intended to break down each main component that will need data to be passed into it for making our social media platform. So far we have our tables organized to handle users, posts, likes, comments, profile and also collections. Each will be made sure to have their corresponding IDs as well as the important information for each table as outlined below.
Our finalized database structure is as follows:

- **Table: Users**
  - userID - int NOT NULL AUTO_INCREMENT
  - email - varchar(50) NOT NULL
  - username - varchar(45) UNIQUE NOT NULL
  - password - varchar(45) NOT NULL
  - firstName - varchar(50) NOT NULL
  - lastName - varchar(50) NOT NULL
  - city - varchar(45) DEFAULT NULL
  - state - varchar(45) NOT NULL
  - DOB - date NOT NULL
  - privacy - int NOT NULL DEFAULT 0
  - pfpURL - varchar(225) NOT NULL
  - PRIMARY KEY (userID)
- **Table: Post**
  - postID - int NOT NULL AUTO_INCREMENT
  - userID - int NOT NULL
  - title - varchar(127) NOT NULL
  - caption - varchar(255)
  - type - enum('text', 'photo', 'video') NOT NULL
  - contentURL - varchar(225)
  - content - text(65535)
  - dateCreated - BIGINT NOT NULL
  - PRIMARY KEY (postID)
  - FOREIGN KEY (userID) REFERENCES Users(userID)
- **Table: Comments**
  - commentID int NOT NULL AUTO_INCREMENT
  - postID - int NOT NULL
  - cuID - int NOT NULL
  - comment - varchar(255) NOT NULL
  - dateCommented - BIGINT NOT NULL

  - PRIMARY KEY (commentID)
  - FOREIGN KEY (cuID) REFERENCES Users(userID)
  - FOREIGN KEY (postID) REFERENCES Post(postID)

- **Table: Favorites**
  - postID - int NOT NULL
  - fuID int - NOT NULL
  - dateFavorite - BIGINT NOT NULL
  - FOREIGN KEY (fuID) REFERENCES Users(userID)
  - FOREIGN KEY (postID) REFERENCES Post(postID)

- **Table: Follow**
  - userIDFollowing - int NOT NULL
  - userIDFollowed - int NOT NULL
  - dateFollowed - BIGINT NOT NULL
  - FOREIGN KEY (userIDFollowing) REFERENCES Users(userID)
  - FOREIGN KEY (userIDFollowed) REFERENCES Users(userID)

- **Table: Collections**
  - collectionID - int NOT NULL AUTO_INCREMENT
  - userID - int NOT NULL
  - name - varchar(255) NOT NULL
  - iconURL - varchar(225)
  - dateCreated - BIGINT NOT NULL
  - lastUpdated - BIGINT NOT NULL
  - PRIMARY KEY (collectionID)
  - FOREIGN KEY (userID) REFERENCES Users(userID)

Table: Collection_Content

  - postID - int NOT NULL
  - collectionID - int NOT NULL
  - dateAdded - BIGINT NOT NULL
  - FOREIGN KEY (postID) REFERENCES Post(postID)
  - FOREIGN KEY (collectionID) REFERENCES Collections(collectionID)

4.2 Add / Search / Delete / Display Architecture

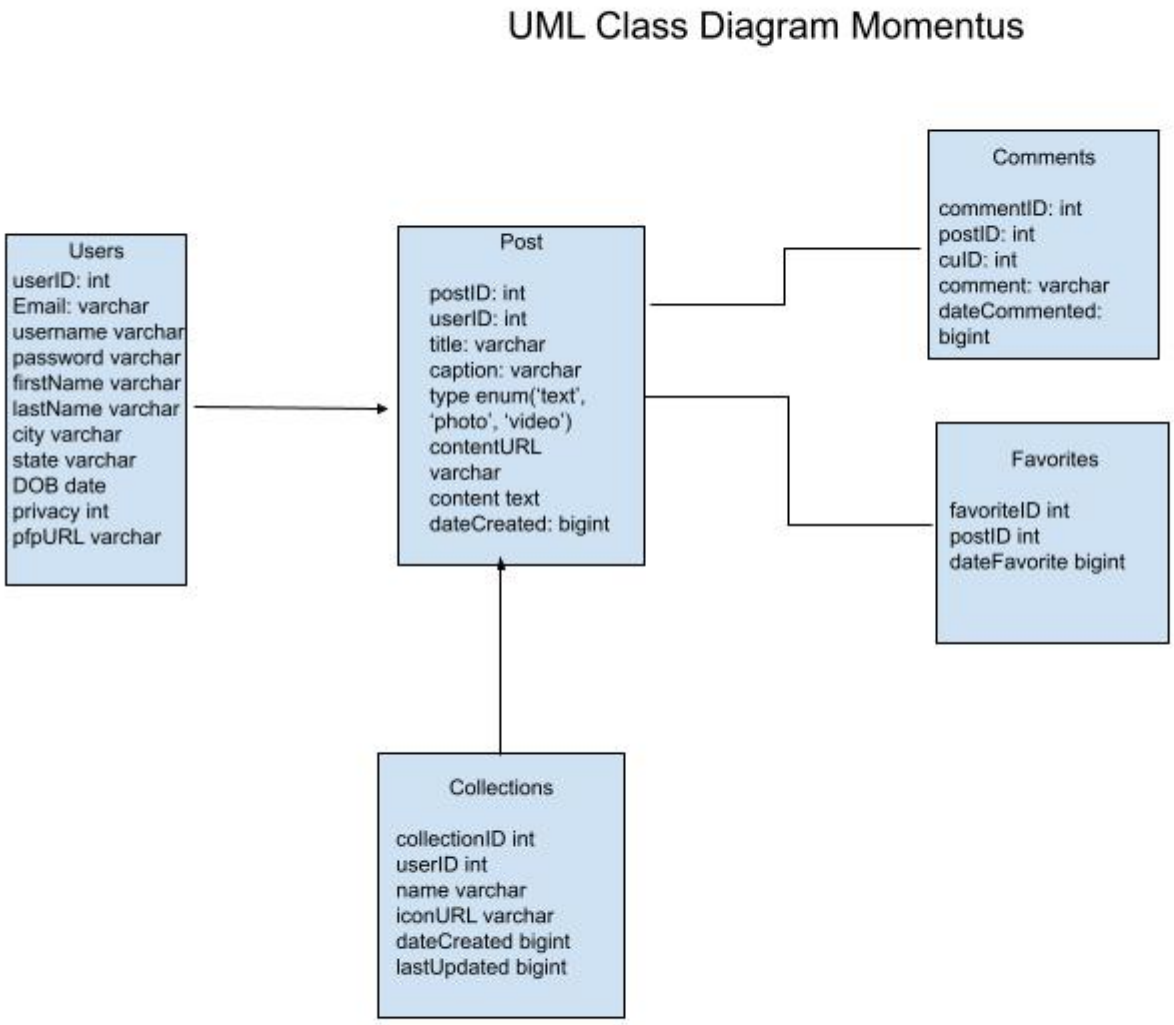
- **Add** - DB entries will be added for each specific table value, so depending on if a user or post is created we will be adding entries of data for each specific value. We also intend to add entries for likes and hold messages for each users' data so that it is saved upon reopening a direct message with someone.
- **Search** - when a search query is performed on the front-end the DB should be able to pull the data for the specific query depending upon the request, for example if a query is created regarding a post we can push the post to the user based on the postID, and the type (shown in the DB table) and make sure we have the appropriate logic to make sure it only shows results that are publicly accessible unless otherwise specified.
- **Delete** - our DB should have the ability to remove items such as users, collections, and posts. Ideally the DB entries that would be deleted would be dependent upon the users request, we want to have the ability to suspend a userID for a certain grace period before we delete the userID with all of its information included.
- **Display** - As of now we want to process all the inputted data for each user, as well as any form of data that they will post on our website. We intend to have our DB display our different values for each table, for example our user DB table will display all the information on the user to track their username, name, city, state, and followings list.

4.3 APIs

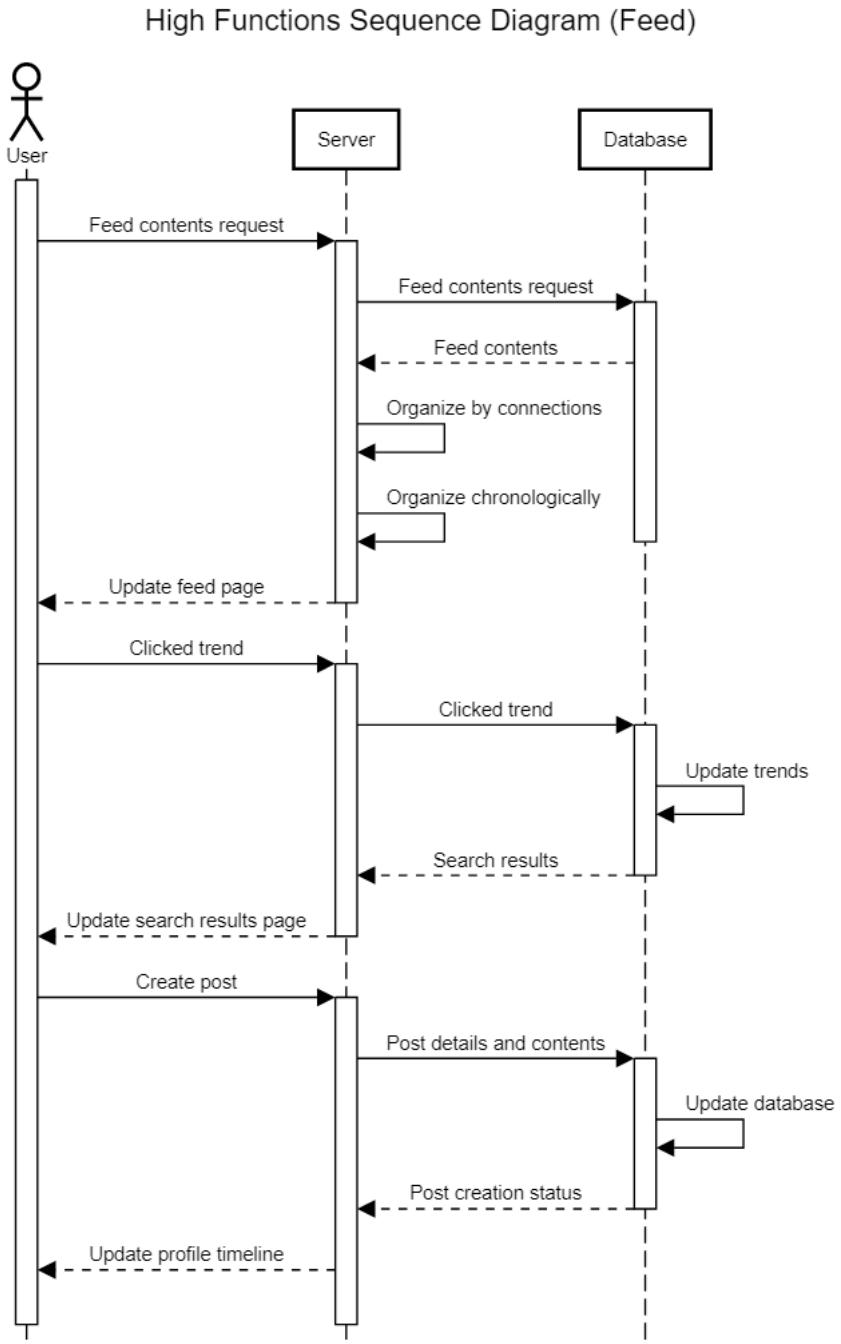
As of now, we have not yet reached the point where we are able to definitively say what APIs we will need, though we will likely be able to finalize our selection before the next prototype. We intend to implement APIs for posts, users, collections, followers, mutual followings, favorites, and saved posts.

5 High Level UML Diagrams

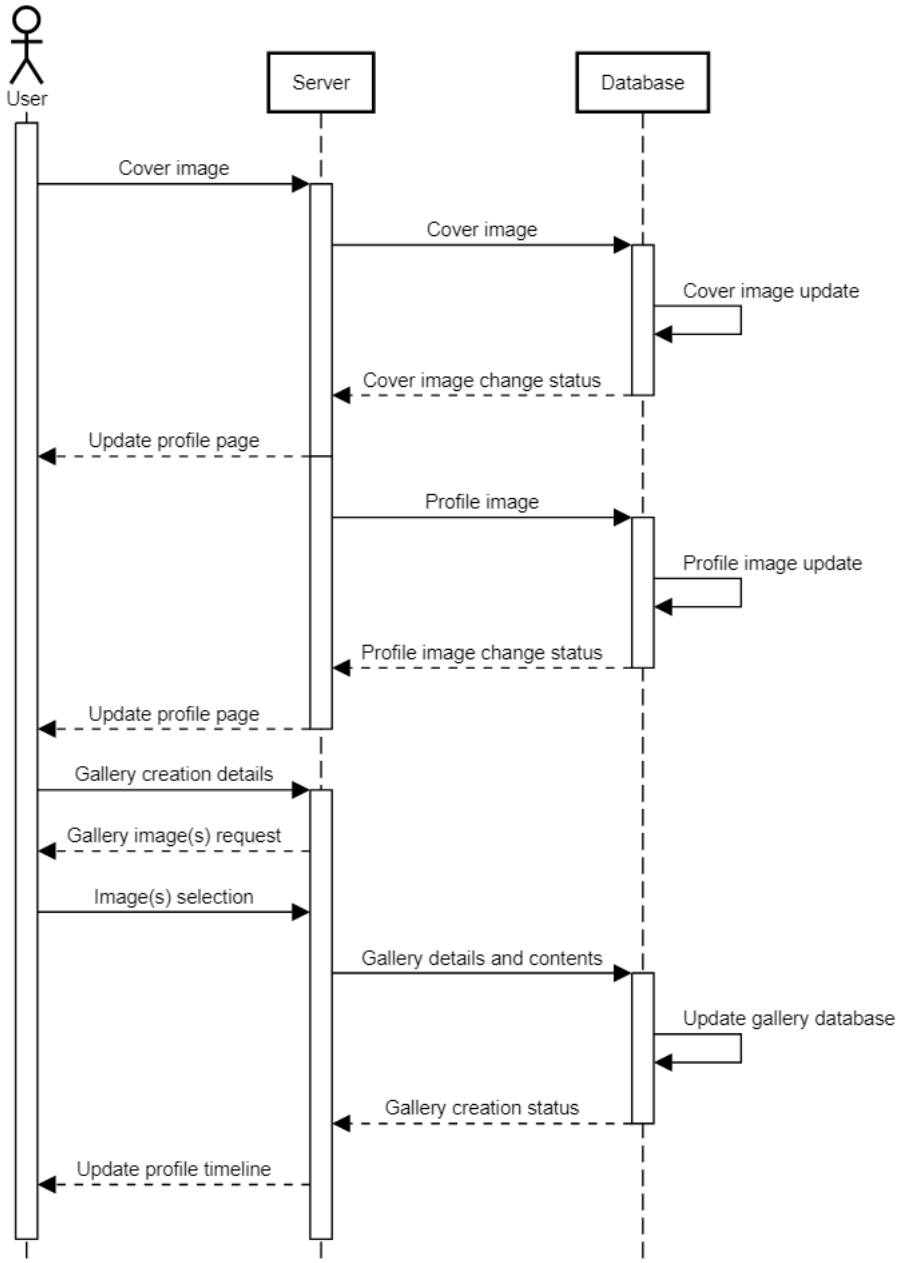
5.1 Class Diagrams



5.2 Sequence Diagrams



High Functions Sequence Diagram (Profile)



6 Key Risks At This Time

- Conflicting time zones / schedules - Due to the pandemic, one of our team members, Jo, is in another time zone (Malaysia time), which puts him 16 hours ahead of the rest of the team. Due to this, we must keep this in mind when scheduling meetings, and ensure that the meetings are not too late or too early for either Jo or the rest of the team. So far, scheduling meetings for later in the evening (8PM - 10PM PST) seems to be the best compromise that works for everyone.

- Skills / Knowledge of Technologies - Though not everyone has much experience in all technologies being used, at least one person, usually more has good knowledge of each technology. In addition to this, team member positions have been well chosen, allowing those with prior knowledge of their position's domain to work in said position. Our team stays in constant communication, and team members will often explain things to those that ask questions, or write small guides on certain technologies and features. In addition to this, team members will sometimes share links to guides and videos that summarize and explain technologies for team members who are not familiar with it.

- Deliberating Tasks: Currently for each component we have a person attempting to handle each task. Currently we have Adam and Eric working the Node.js, Raza and Adam are handling the DB and AWS transfer process, while Natalie and Jo Ey are handling the front-end design/React. One of the key risks we do run into is being able to work side by side regarding certain tasks due to risk of overlap in work. We need to figure out a method to allow work to be done side by side rather than one at a time for improved efficiency. The back-end team/entire group also needs to review data transfer technologies, specifically when running into issues using axios and having to switch over to using a fetch method for the data transfer and display process. Front-end team/entire group also continuing to study React documentation to improve the design of our overall project.

- Team work Risks: Currently each team member does know what they need to take care of, we also continue to have constant dialogue regarding this via discord to make sure that each team member is in the loop and on task. We do run into stretches where we are not too sure what each person may be doing, but this is something we are currently addressing and making sure each team member has a task to do and is capable of doing it.

7 Project Management

Throughout our project, our team stays in constant communication through a custom Discord server. This allows for communication between members at times that are convenient for each of them, and allows for messages to be stored for posterity and later reference. Different channels are set up for different topics, including general discussion, back end discussion, front end discussion, and links to guides and references needed for the project.

Our team regularly meets in addition to class meetings and assesses our progress, as well as discussing any potential problems, questions, or solutions that may come up. Team lead helps decide what tasks will go to each team member in conjunction with their feedback and prior knowledge and skill set.

Important documents are kept in a Google Drive folder, which is organize by milestone and topics. Here team members can collaboratively work on and view documents for the project.