SW Engineering CSC648/848 Summer 2022

GatorShare

Milestone 02

Team 01, Section 01

Team Members:

Donna Nguyen, Team Lead | unguyen3@mail.sfsu.edu

Estefanos Kebebew, Back-End Lead

Brianna Soukup, Front-End Lead

Mohamed Toure, Github Master

Brian Nguyen, Database Master

Aleksandr Gusev, Front-End

History:

M2 Submitted:	**July 07, 2022**
M2 Revised:	**July 18, 2022**
M1 Submitted:	**June 21, 2022**
M1 Revised:	**July 15, 2022**

Table of Contents

Data Definitions

User: Any person who uses the web application.

- **Unregistered User/General User:**

A User who has not signed up or logged into the application. This User may only browse our application, and this User is able to access all posts and comments. This User cannot create posts, post comments, like a post, send messages or receive notifications. This User is not required to log in/register, but must register in order to communicate with other Registered Users.

- **Registered User/Approved User:**

A Registered User is a User that has made an account. The User needs to register with the specifically required information, notably their name and email address. The User must sign up using the SFSU email address to verify their student status. A registered user can be a student, staff, faculty member, or alumni. This User must create a password and accept our application's terms of service. This User is allowed to fully utilize the platform (besides Admin privileges). This User can browse and search through the website. This User can also post, delete their posts, comment on posts, receive notifications, like/unlike posts (one button), and report something on the platform.

- **Admin:**

Admin Users are solely restricted to the creators of the web application. These Users are able to fully utilize the platform. They can monitor all Posts, Comments, and Registered Users. They have the privilege to delete posts, comments, and also registered users. They supervise the platform and make sure that the rules and conditions of utilization are respected. They have the possibility of communicating with the users via email when necessary.

- **Post:**

A submission made by a Registered User. A post is composed of a title, description, one or more images, and an optional media/link submission. All posts are required to have a title text describing what the post contains, although a description can be left blank. A Post can be used as plain text for an article, an uploaded image to promote a digital artwork or event flyer, or a discussion post.

- **Comment:**

A way for Users to communicate with other Users. A Registered User can leave a comment by clicking on another Registered User's post and using the box located underneath each post's description. A comment would be text only. The username, the text content, and the time it was submitted are all included in each comment. A comment may only be 250 characters long.

- **Message:**

A Registered User can message another Registered User. A subject field is required with a limit of 50 characters. A textbox must not be empty with a limit of 500 characters. A message body will be text-only. When a message is sent, the other registered user will receive a notification. A

Registered User can edit or delete their message after the message has been sent. When the message is updated, it will have an “Edited” note.

Prioritized Functional Requirements

Priority 1 - Must Have:

Requirements for Website:

1. Search Bar shall be present throughout all pages.
2. Footer shall be present throughout all pages.
3. Logo shall be present throughout all pages.
    - When the Logo is clicked, it shall redirect the User to the Home page.
4. Navigation Bar shall be accessible for all Users.
5. Navigation Bar shall have a Home page.
6. Navigation Bar shall have a Login page.
7. Navigation Bar shall have a Sign Up page.
8. Navigation Bar shall have a Posts page.
9. Navigation Bar shall have a Create a Post page.
10. Navigation Bar shall have an About Me page.
11. Posts shall be public for all Users.
12. Comments shall be public for all Users.
13. Images shall be accessible for all Users.

Requirements for Unregistered User:

14. Unregistered Users shall be able to view all Posts.
15. Unregistered Users shall be able to view all Comments.
16. Unregistered Users shall be able to view all Profiles.

17. Unregistered Users shall be able to view all Images.

Requirement for Registered Users:

18. Registered Users shall be able to add a Post on the platform.
19. Registered Users shall be able to edit their Post.
20. Registered Users shall be able to delete their Post.
21. Registered Users shall be able to like a Post.
22. Registered Users shall be able to dislike a Post.
23. Registered Users shall be able to add Comments on another Registered User's Posts.
24. Registered Users shall be able to receive notifications from other Registered Users.
25. Registered Users shall have an individual Profile.
26. Registered Users shall be able to send messages.
27. Registered Users shall be able to receive messages.
28. Registered Users shall be able to edit messages.
29. Registered Users shall be able to delete messages.
30. Registered Users shall be able to like a Comment.
31. Registered Users shall be able to dislike a Comment.
32. Registered Users shall be able to edit their Comments.
33. Registered Users shall be able to delete their Comments.

Requirements for Posts:

34. Posts shall have a character limit of 750 characters.
35. Posts shall have an image submission field.
    - Image upload shall not be a requirement when creating a post.

- Images shall only be accepted in jpg, png, or jpeg format.

36. Posts shall display the Registered User who created the post.
37. Posts shall include a comment box.
38. Posts shall include a submit button.
39. Posts shall display the time of the submission.
40. Posts shall display the number of likes it has received.

Requirements for Comments:

41. Comments shall not be empty.
42. Comments shall display the username of the Registered User who created the comment.
43. Comments shall display the time of the comment submission.

Requirements for Admin:

44. Admin shall be able to delete other Users.
45. Admin shall be able to delete other Posts.
46. Admin shall be able to delete other Comments.
47. Admin shall be able to Post.
48. Admin shall be able to delete their Post.
49. Admin shall be able to edit their Post.
50. Admin shall be able to Comment.
51. Admin shall be able to delete their comments.
52. Admin shall be able to edit their comments.
53. Admin shall be able to send messages.
54. Admin shall be able to receive messages.

55. Admin shall be able to delete messages.
56. Admin shall be able to like a Post.
57. Admin shall be able to dislike a Post.
58. Admin shall be able to like a Comment.
59. Admin shall be able to dislike a Comment.
60. Admin Users shall be able to receive notifications from Registered Users.
61. Admin users shall be able to post and edit the “About Me” section of the web application.

Priority 2 - Desired:

Desired Requirements for Registered Users:

1. Registered Users shall be able to ‘Flag’ a post.
2. Registered Users shall be able to ‘Flag’ a comment.

Desired Requirements for Admin:

3. Admin shall be able to review ‘Flagged’ posts.
4. Admin shall be able to review ‘Flagged’ comments.

Priority 3 - Opportunistic:

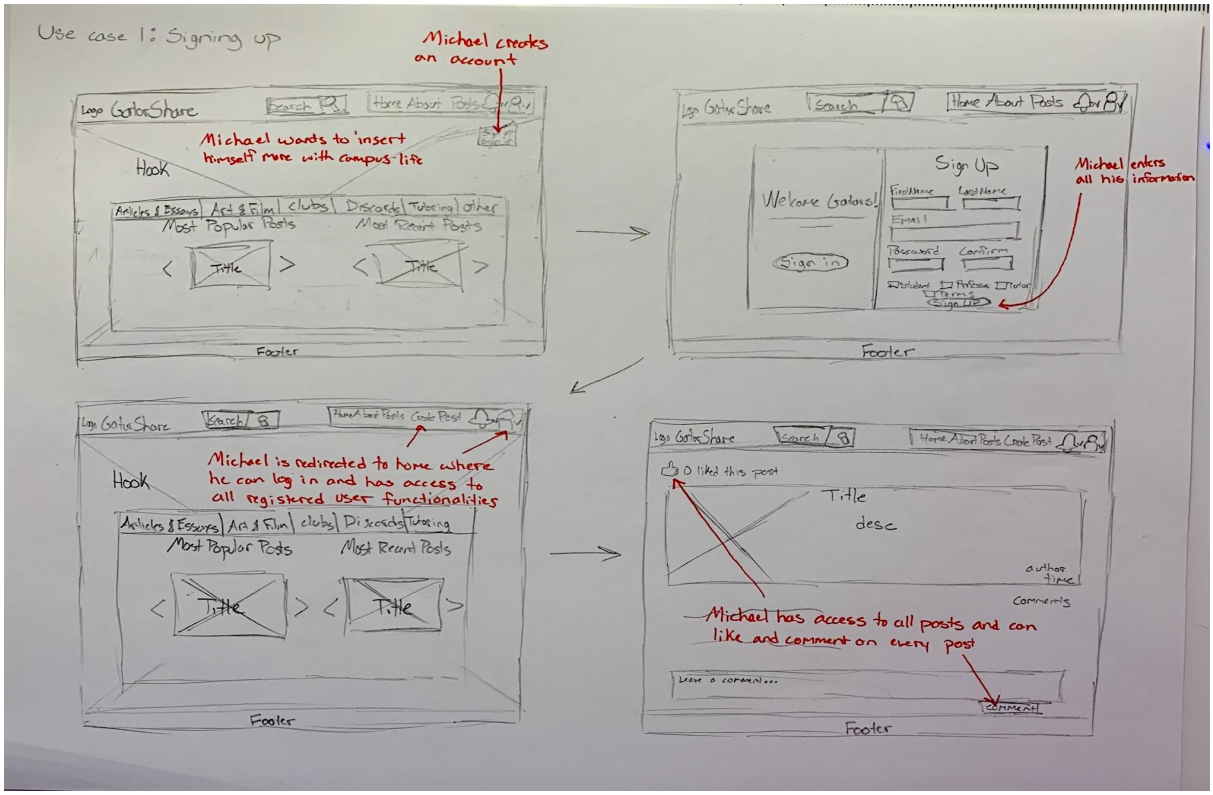
1. All liked posts from users shall be saved.
2. All posts shall have a ‘Report’ button for spam.
3. All comments shall have a ‘Report’ button for spam.
4. Registered Users shall be able to ‘Flag’ a post.
5. Registered Users shall be able to ‘Flag’ a comment.

6. Registered Users shall be able to receive notifications from Administrators if their Posts or Comments have been removed.
7. Admin shall be able to review ‘Reported’ posts.
8. Admin shall be able to review ‘Reported’ comments.
9. Admin shall be able to review ‘Flagged’ posts.
10. Admin shall be able to review ‘Flagged’ comments.

UI Mockups and Storyboards (High Level Only)

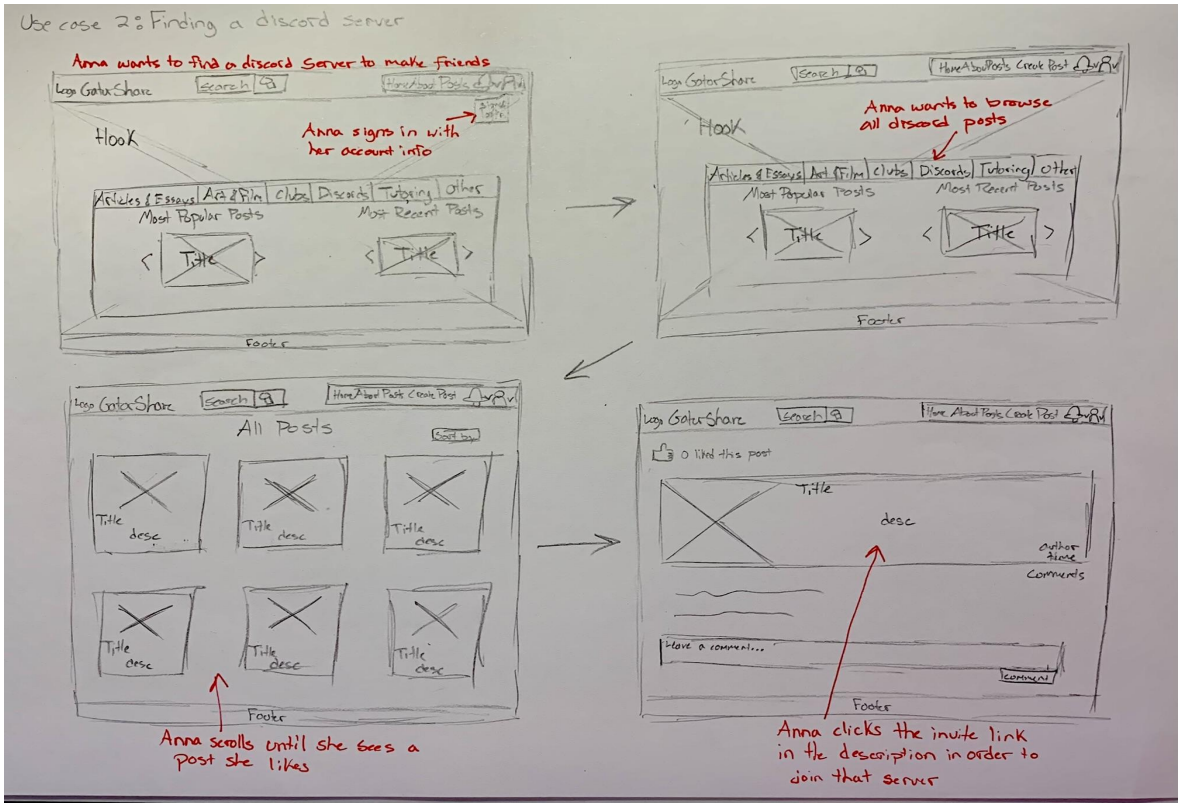
Use Case 1: Signing Up

Michael, an SFSU student, wants to insert himself more into campus life. Michael visits the platform's homepage to create an account. Michael navigates to the “Sign Up” button in the navigation bar and is redirected to the “Sign Up” page. The new page requires Michael to input certain information, such as his email, username, password, and terms and conditions. Once registered, Michael now has full access to the platform. Michael logs into his new account and continues to browse the platform's posts. While viewing a post, Michael learns he has the ability to like the post and comment on it.



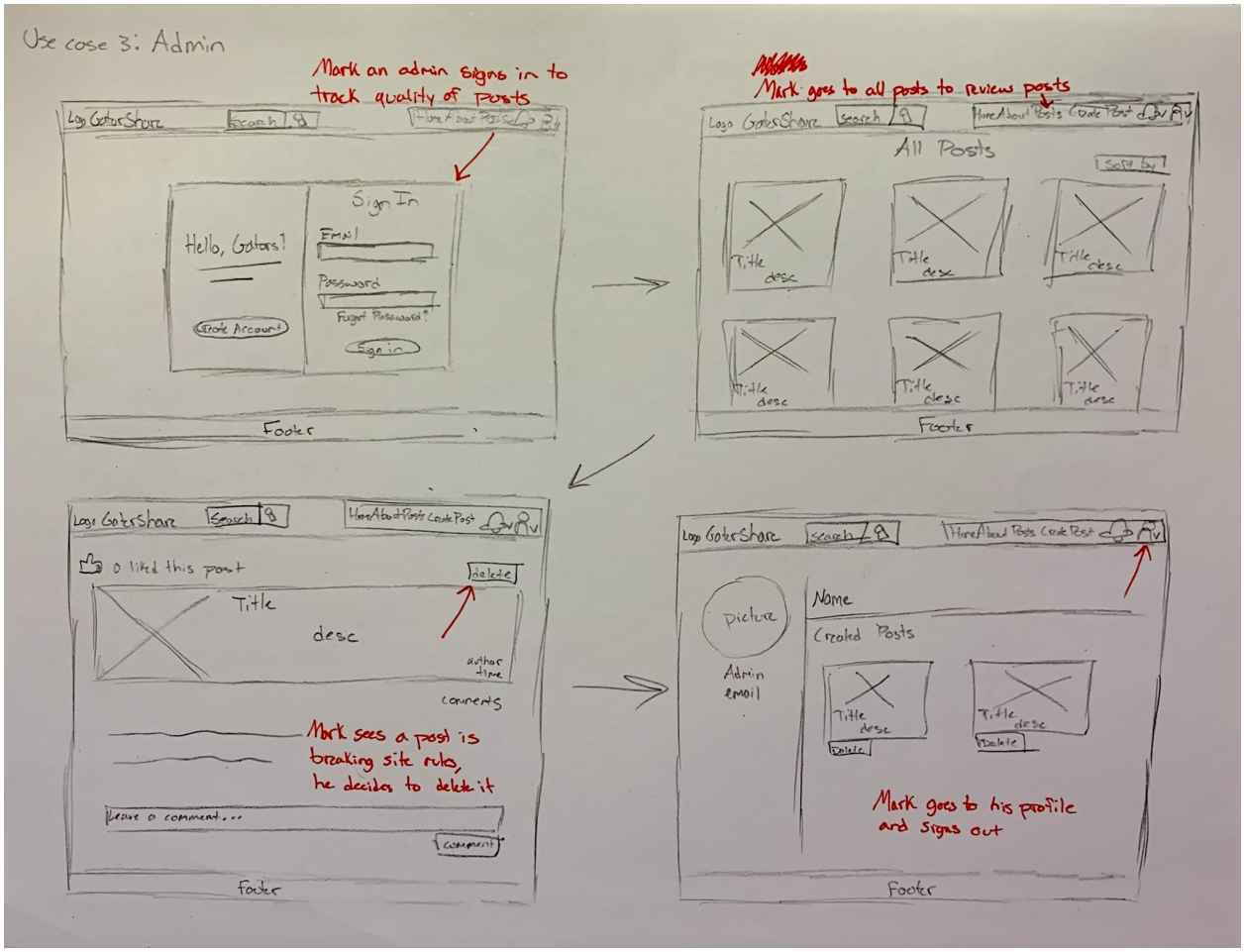
Use Case 2: Finding a discord server

Anna is a registered user who wants to join a Discord server so she can meet fellow SFSU freshmen like herself. She finds the “Log In” button and enters her account credentials. Anna chooses to explore all Discord postings rather than do a specific search. When she clicks the “Discord” button on the main page, she is taken to the “Discord Posts” page, where she can see all the posts that registered users have created relating to Discord. Anna clicks on a post called “Computer Science Discord” that captures her attention. While looking through the post, she is able to like the post, see the number of likes, read a short description about the Discord server, view comments posted by other registered users like herself, and an available invite link. Anna clicks the provided server invite link. There, she exits the platform and is connected to the Discord server.



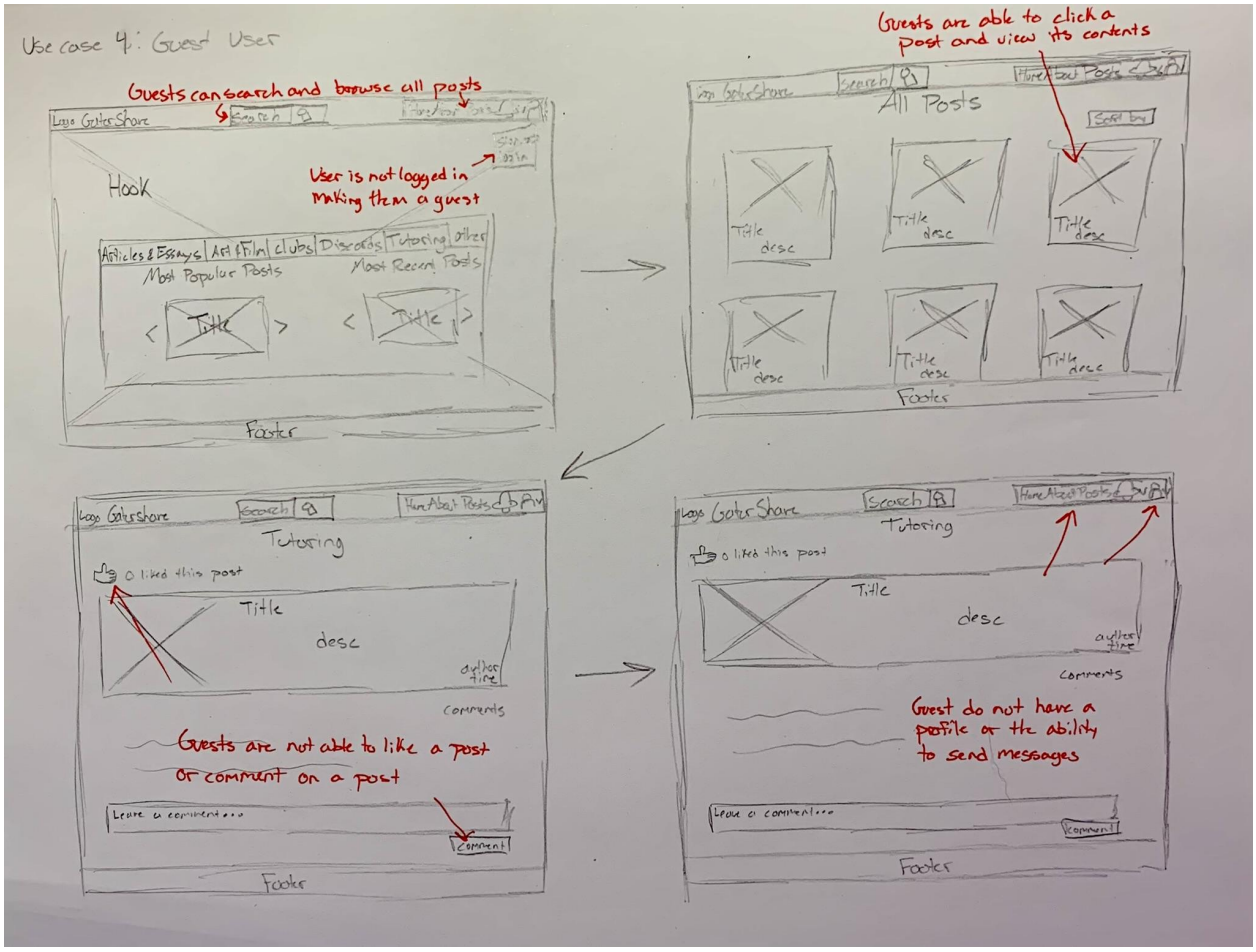
Use Case 3: Admin

Mark has an “Admin” role on the platform. When Mark logs in, he is able to view the entire application. Mark has more privileges than the standard Registered User, such as being allowed to see the login activities of the User. He wants to verify that some of the recent posts aligned with the website terms and conditions. When browsing through posts, they find a suspicious submission. He clicks the post and assesses its content. After some judgment, he decides that this post needs to be discarded. Mark clicks the “Delete” button in the top-right corner and permanently deletes the post.



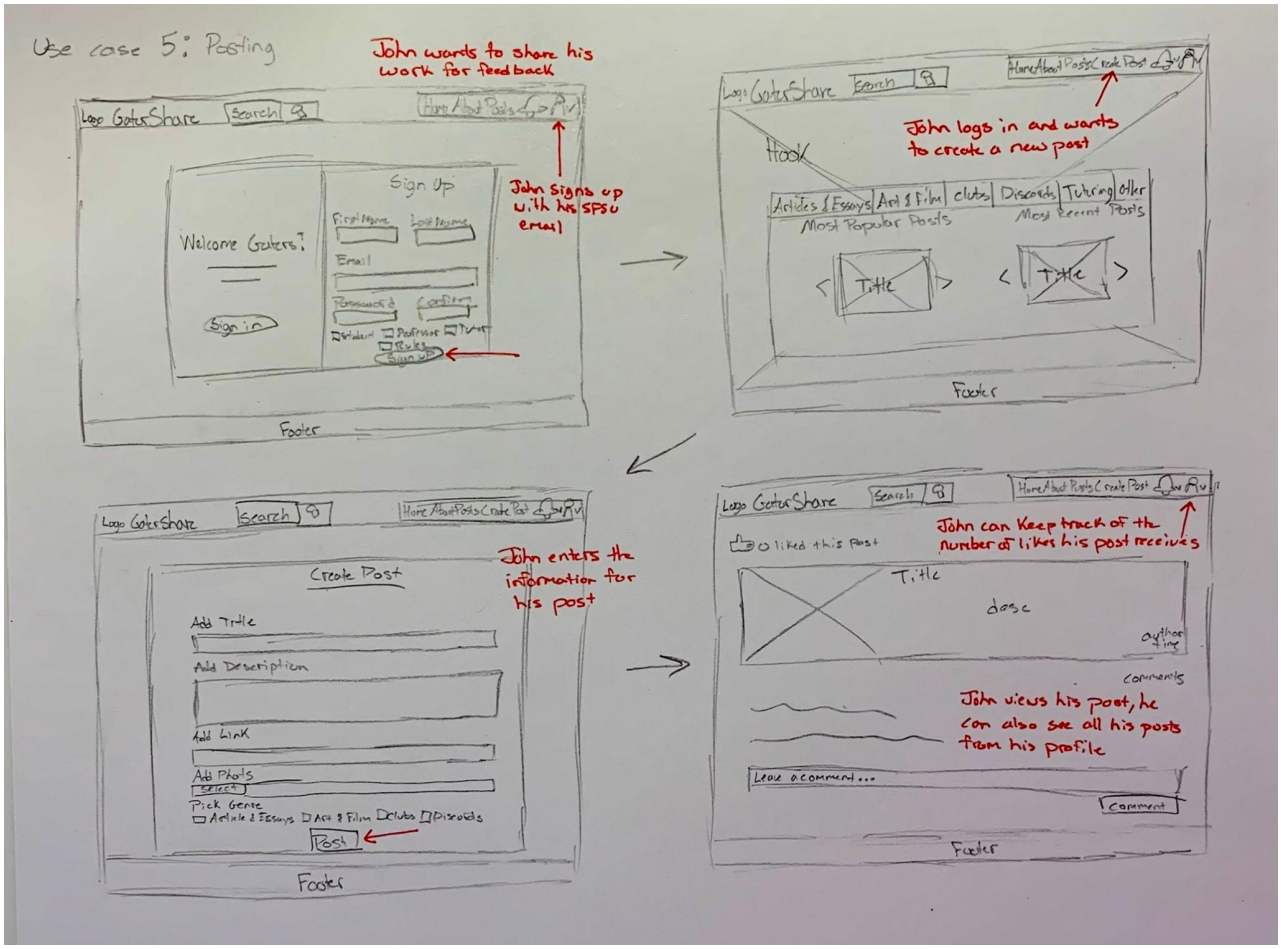
Use Case 4: Guest

Tom is a Guest. A “Guest” is someone who has not registered an account or is accessing the page while logged out. As a Guest, Tom has permission to browse through every post, view club events, access articles, etc. Tom can also utilize the search bar to search for his interests. Tom is interested in browsing the platform's tutoring posts, therefore he navigates to the “Tutoring" section of the platform. He views a specific post- in hopes to locate more information. While viewing the post, Tom tries to comment. As a Guest, Tom is then met with a popup that notifies him that says he cannot comment since he is not a registered user. In order to have full access to the platform and its functionalities, Tom must create an account and be logged in.



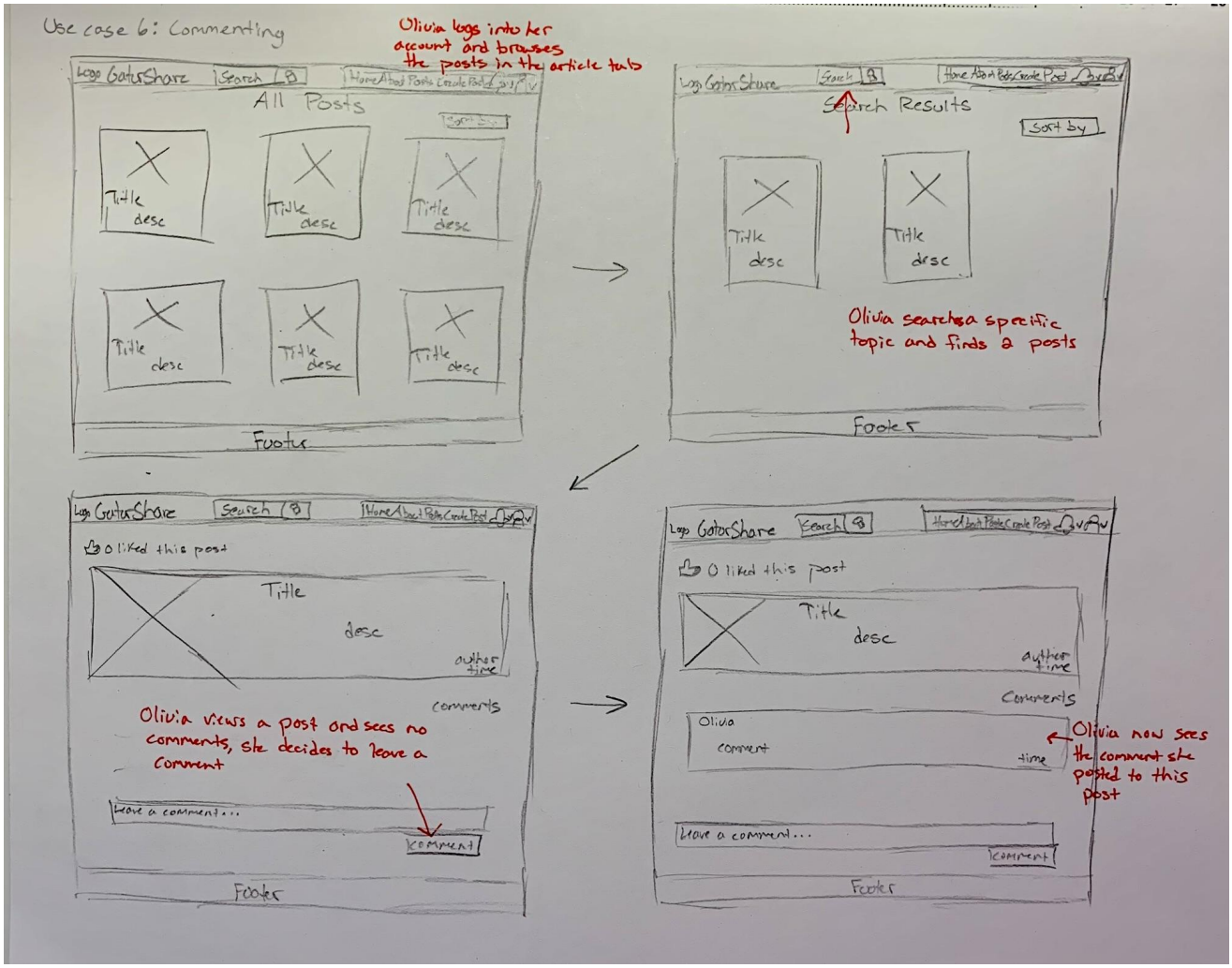
Use Case 5: Posting

John, a student at SFSU, wants to share his artwork to gain publicity and feedback. John decides to post his work on GatorShare. When he clicks the “Sign Up” button in the navigation bar, he is taken to the “Sign Up” page. John completes the registration process by providing all the necessary details. After signing in, John navigates to the “Create Post” button. John inputs a title, a description, and several pictures of his paintings. He selects the “Art & Film” category for his post and hits “Post”. Finally, John is redirected to his post, where he can view the number of likes and comments it has received. John can also access his posts from his profile.



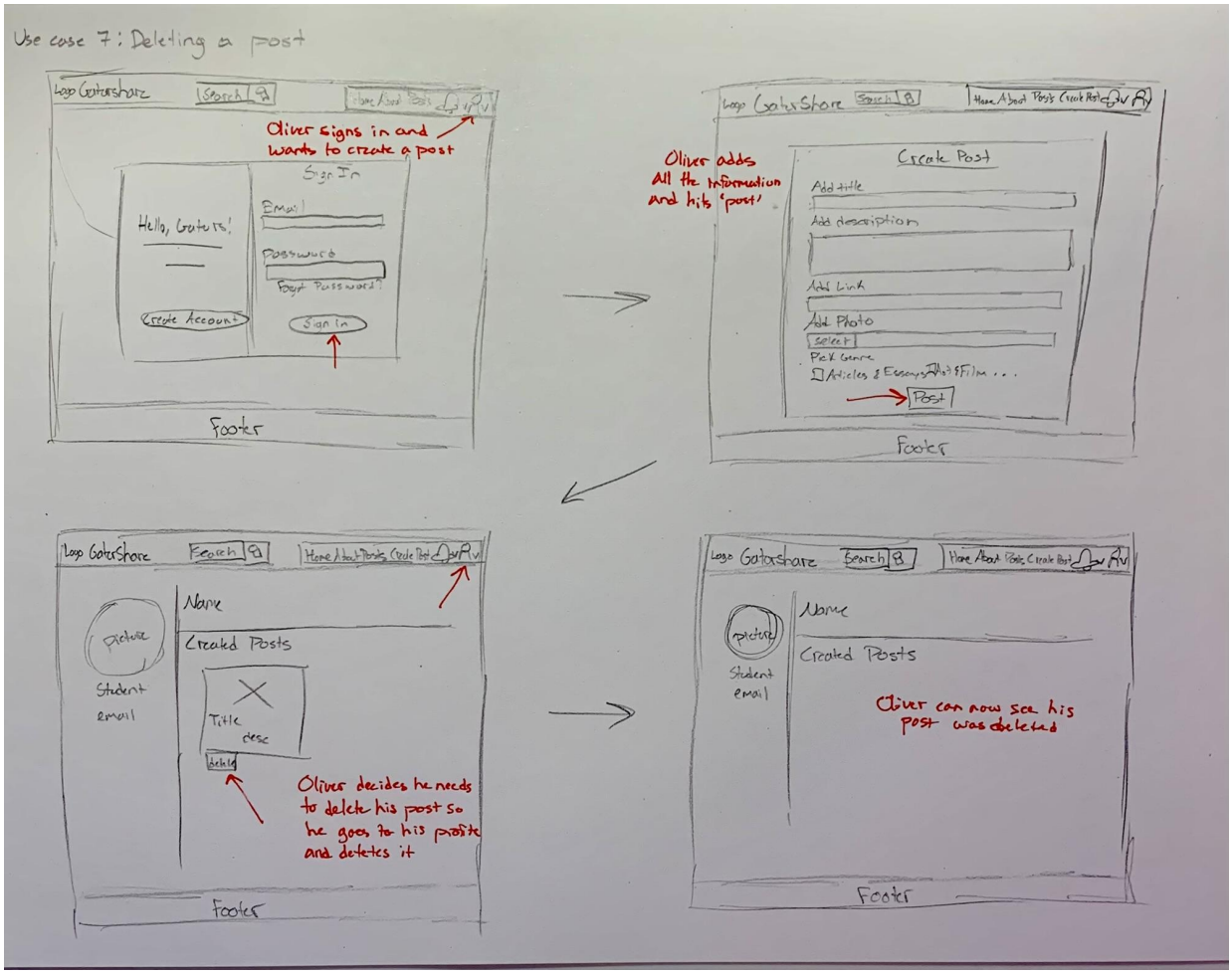
Use Case 6: Commenting

An SFSU student, Olivia, is a registered user on GatorShare. As she is browsing through the website, she stumbles upon several student-written articles. She clicks on one of the articles that has interested her. Olivia, being a registered user, is able to leave a comment on the post. She scrolls to the bottom of the article, and sees a comment box under the post. Here, she clicks a "Comment" button, and is able to write her comment.



Use Case 7: Deleting a post

Oliver, an SFSU student and a registered user at GatorShare.com, can create a new post. He logs into the website with his credentials and clicks the “Create Post” button. On the “Create a Post” page, he enters all the information and uploads a photo. He then clicks the “Post” button. Suddenly, he realizes he uploaded the wrong image and would like to delete it. Oliver clicks on the post and is then redirected to the posting page. He then clicks the “Delete” button at the top right of the post and confirms the deletion. The website brings him to the “Posts” page once deleted, with his submission cleared.



High Level Database Architecture and Organization

Business Rules

1. Account:
    1.1. Account can have one or many Users.
    1.2. Accounts can only be monitored by Admin.
2. Admin:
    2.1. Admin shall have at least one account.
    2.2. Admin will have the ability to see users credentials.
    2.3. Admin can remove posts.
    2.4. Admin can flag accounts.
3. Student Account:
    3.1. Students can only have one and one account.
    3.2. Students cannot be an Admin.
    3.3. Students will have access to the messaging feature.
4. Professor Account:
    4.1. Professors can have only one account.
    4.2. Professor can comment on a Post.
    4.3. Professor can flag accounts.
5. Tutor Account:
    5.1. Tutors can have only one account.
    5.2. Tutors can tutor one or more subjects.
    5.3. Tutors can only co to one and one Student.

6. Post:
    6.1. Posts can be done by Registered Users.
7. Unregistered Users:
    7.1. Unregistered Users shall be able to view posts.
    7.2. Unregistered Users shall be able to view comments.
    7.3. Unregistered Users shall be able to view Profiles

Identifying Entities and Attributes

Strong Entities:

1. Professor
    1.1. professor_id, key, numeric
    1.2. professor_name, alphanumeric, multivalue, composite
    1.3. professor_email, alphanumeric, multivalue, composite, unique
    1.4. professor_password, alphanumeric, multivalue, composite
2. Tutor
    2.1. tutor_id, key, numeric
    2.2. tutor_name, alphanumeric, multivalue, composite
3. Student
    3.1. idStudent Account, key, numeric
    3.2. username, alphanumeric, multivalue, composite
    3.3. email, alphanumeric, multivalue, composite, unique
    3.4. password, alphanumeric, multivalue, composite

4. Post
    4.1. idPost, key, numeric
    4.2. post_created, alphanumeric, multivalue, composite
    4.3. post_descirption, alphanumeric, multivalue, composite

Weak Entities:

1. Admin
    1.1. idAdmin, key, numeric
    1.2. username, alphanumeric, multivalue, composite
    1.3. password, alphanumeric, multivalue, composite
2. Like
    2.1. idlike, key, numeric
    2.2. idPost, key, numeric
    2.3. idStudent Acccount, key, numeric
    2.4. created_date, multivalue
3. Message
    3.1. idmessage, key, numeric
    3.2. idStudent Account, key, numeric
    3.3. to_userid,
    3.4. message, alphanumeric, multivalue, composite
    3.5. time_sent, multivalue
4. Spam

4.1. spam_id, key, numeric

4.2. idPost, key, numeric

4.3. idStudent Account, key, numeric

4.4. createdDate, multivalue

5. Comment

5.1. idcomment, key, numeric

5.2. idpost, key, numeric

5.3. idcomment, key, numeric

5.4. comment_text, alphanumeric, multivalue, composite

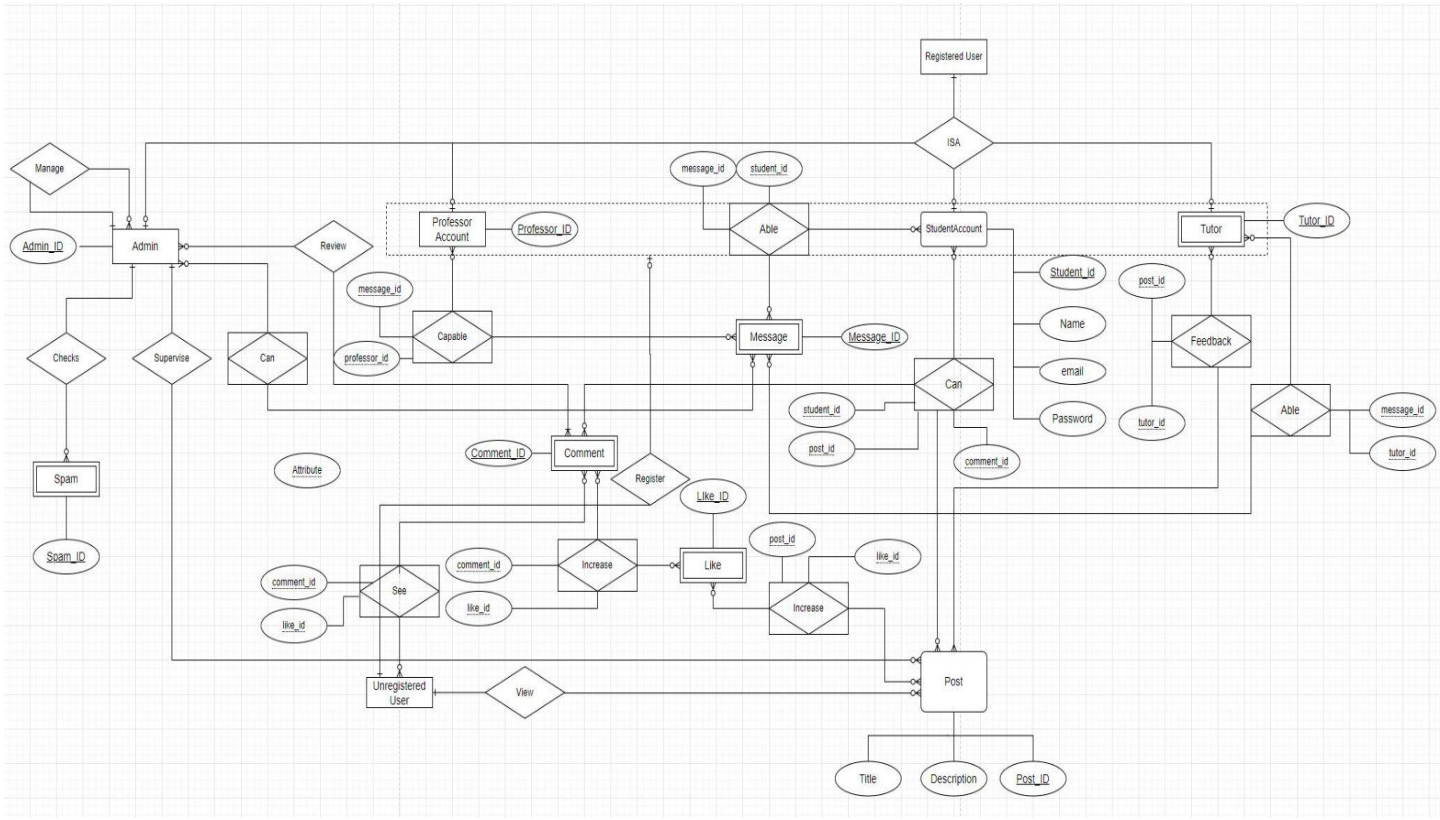
6. Tokens

6.1. token_id, key, numeric

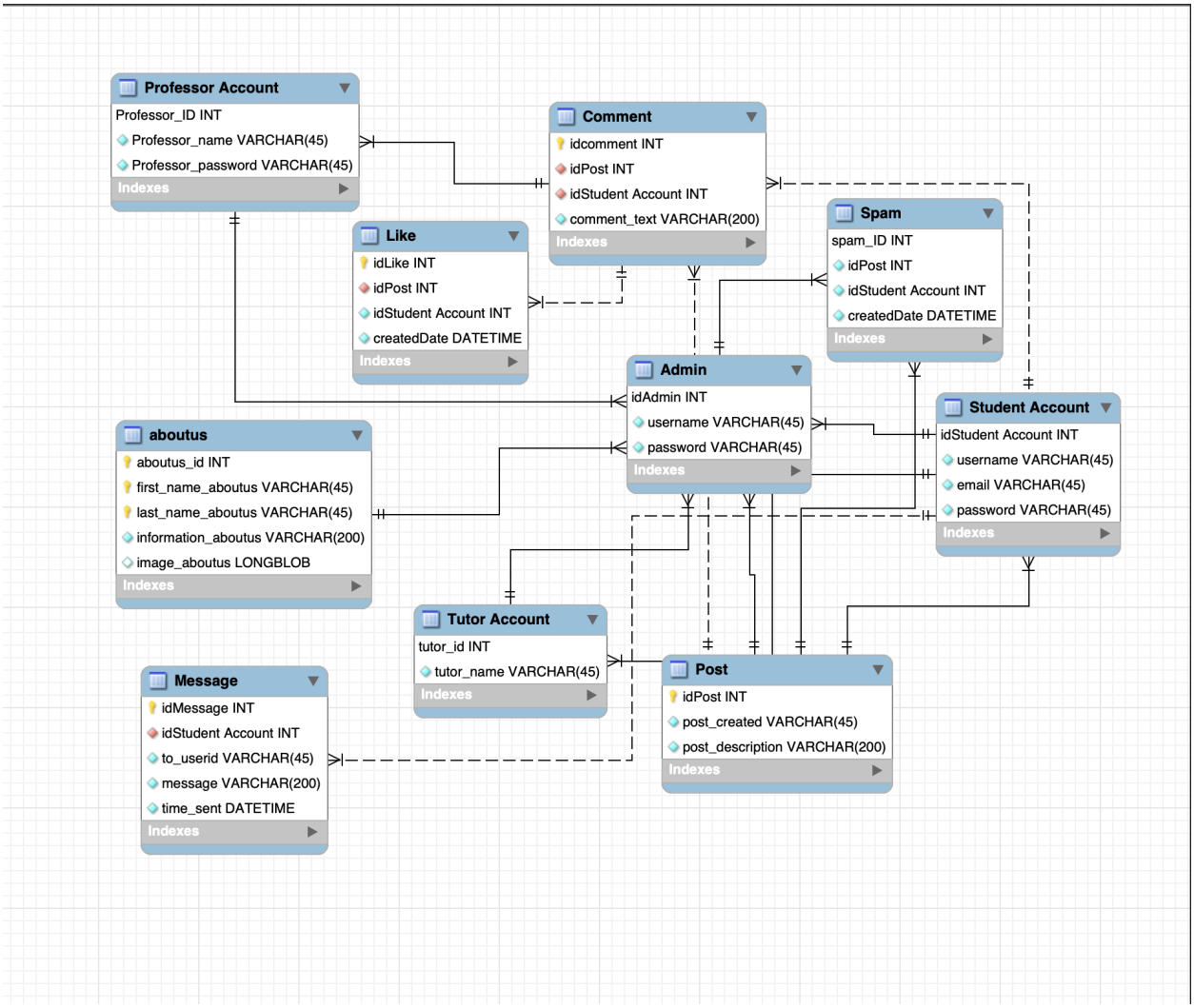
6.2. created_date, multivalue

6.3. token, alphanumeric, multivalue, composite

Entity Relationship Diagram:



Enhanced Entity Diagram:



Database Model:

Student Account:	Student_Id
	username
	email

	password
Messages:	from_userid
	to_userid
	message
	time_sent
Post:	Post_ID
	post_created
	post_description

Professor Account	
Professor_ID	Varchar(45)
Professor_name	Varchar(45)
Professor_password	Varchar(45)

Tutor Account	
Tutor_ID	Varchar(45)
Tutor_name	Varchar(45)

Spam	
Spam_ID	An int
Post_ID	ID number from the post_ID
Student_ID	ID number from idStudent
CreatedDate	DATETIME of the post

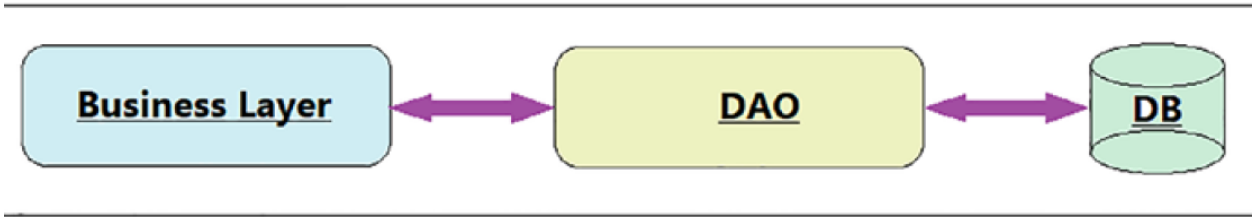
Like	

Like_ID	An int
Post_ID	ID number from the post_ID
Student_ID	ID number from idStudent
CreatedDate	DATETIME of the post

Comment	
Comment_ID	
Post_ID	
Student_ID	
Comment_text	

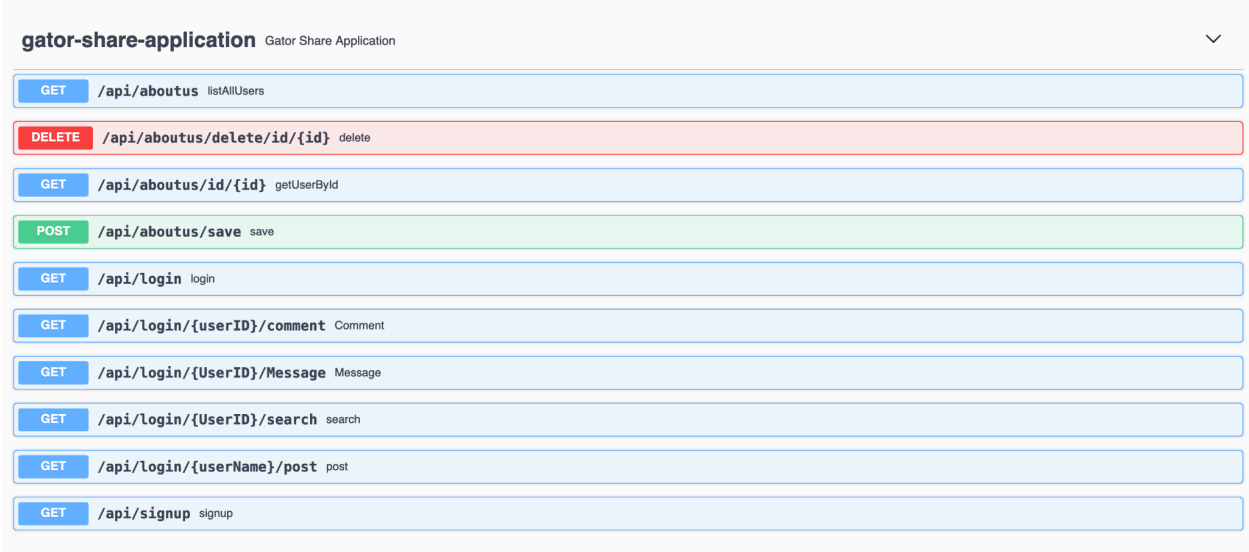
High Level APIs and Main Algorithms

Each RESTful web service will talk to the AWS RDS MySQL database to perform CRUD operation. The REST controller will define different REST endpoints. The Data Access Object diagram pattern is implemented to handle the separation of the business layer and data access operation. In these designs, the DAO layer performs the CRUD operation in the relational database.



The posts are sorted by day. In the Sort class, the by() function is used to sort the records based on the field. The records can be sorted in both ascending or descending order. For Example, /posts?field=date will display the posts in ascending order by day.

Screenshot of the EndPoints:

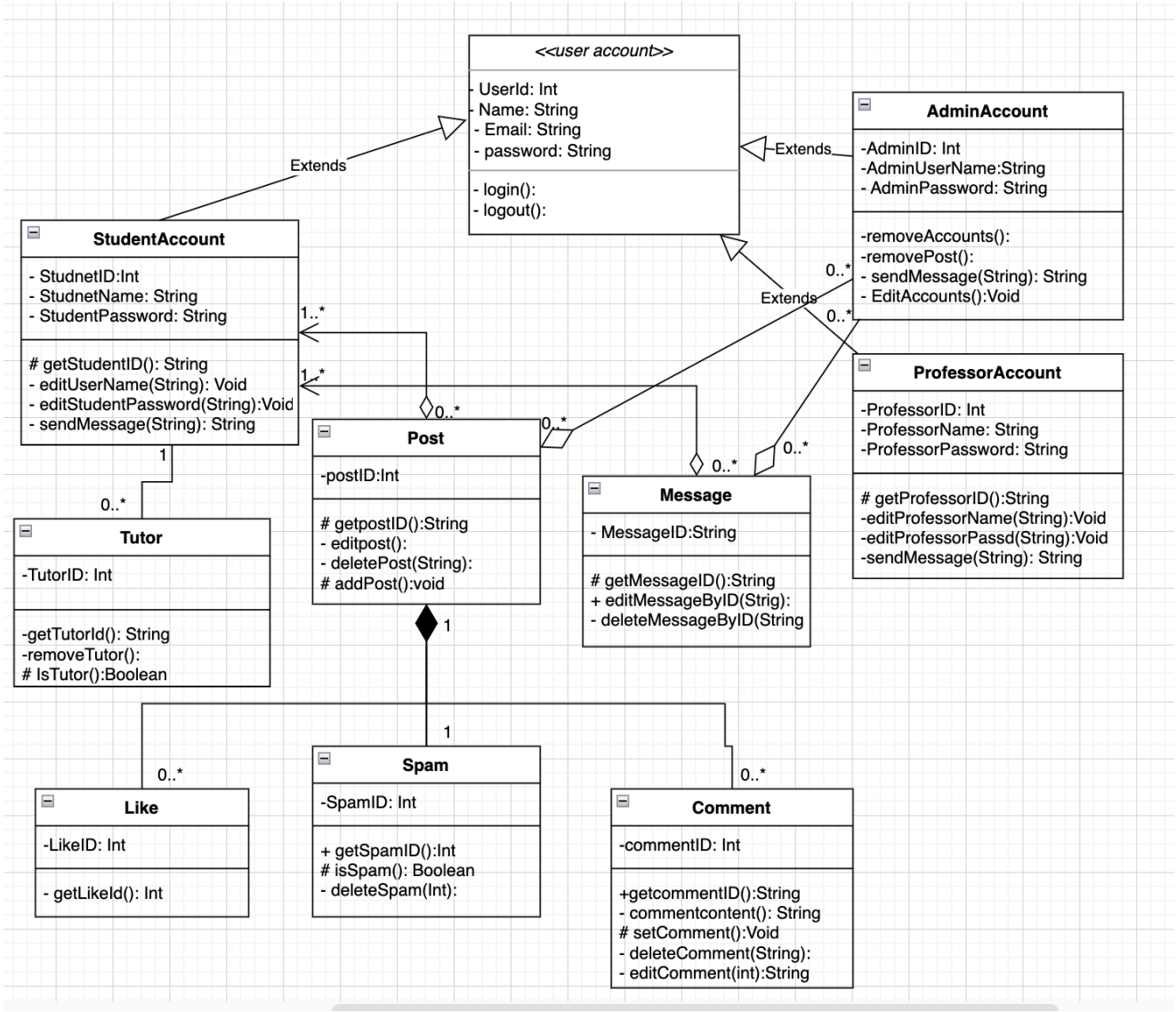


Each Routes Login, Signup, login/{userID}/commet, login/{userID}/message, login/{userID}/post and login/{userID}/search will have POST, DELTE and UPDATE Methods.

Cross-Origin Resource Sharing(CORS) is used to connect RESTFul web service to React. It enables cross-origin requests for a RESTFul web service. To allow these, the Controller class is annotated with @CrossOrigin annotation. And by default, all methods(GET, HEAD, and POST HTTP) will be accessible. All component calls from the front end will internally use the Axios HTTP library to make HTTP requests and receive responses.

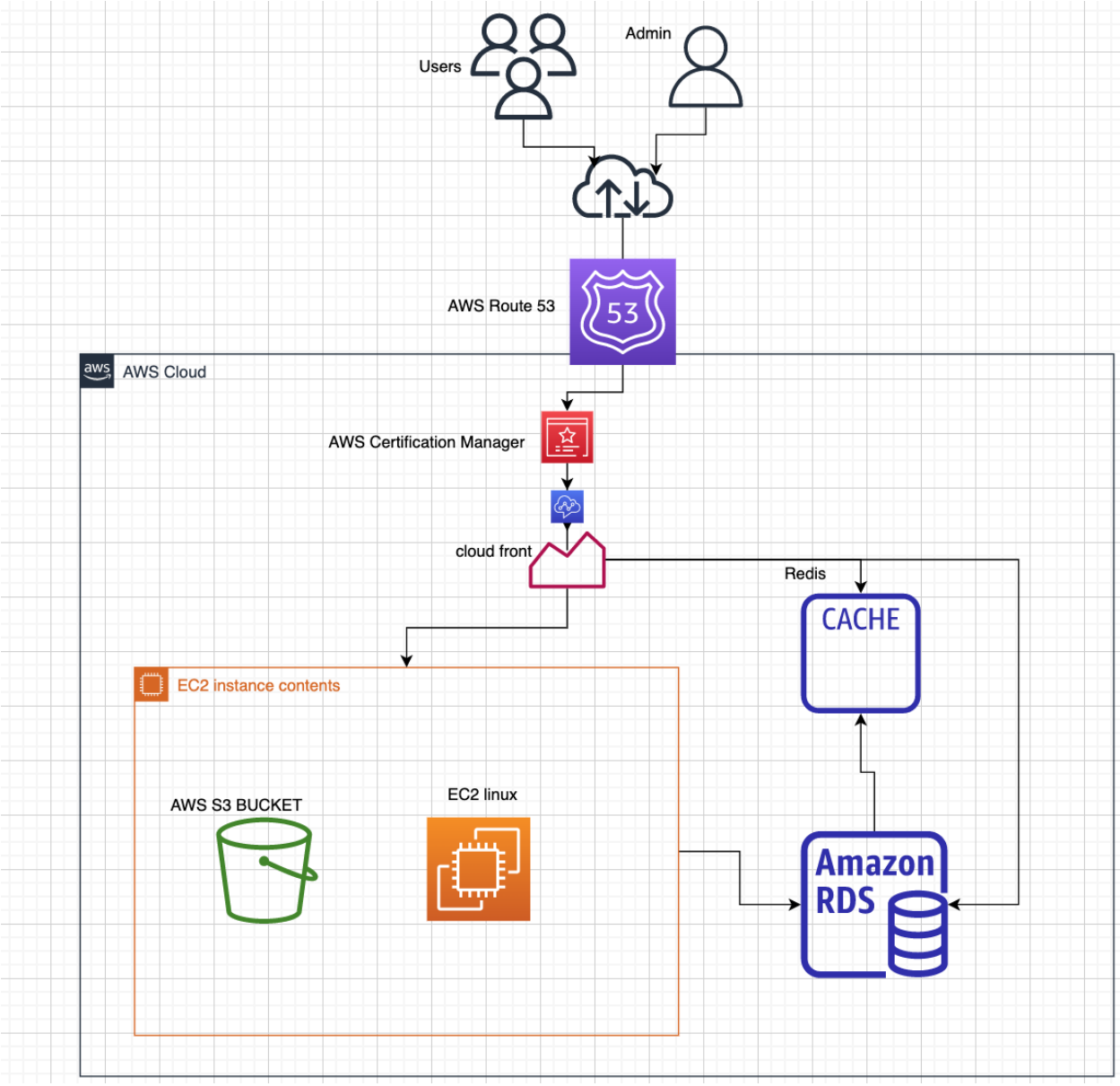
We chose Messaging as our superior feature. To implement this feature, STOMP and SockJS are used. STOMP envelops payloads and adds a header to the communication between the client and the server. SockJS will create low latency, full-duplex, and cross-domain communication channels between the client and the server. In addition, a long-lived TCP communication will be implemented between the client and the server.

High Level UML Diagrams

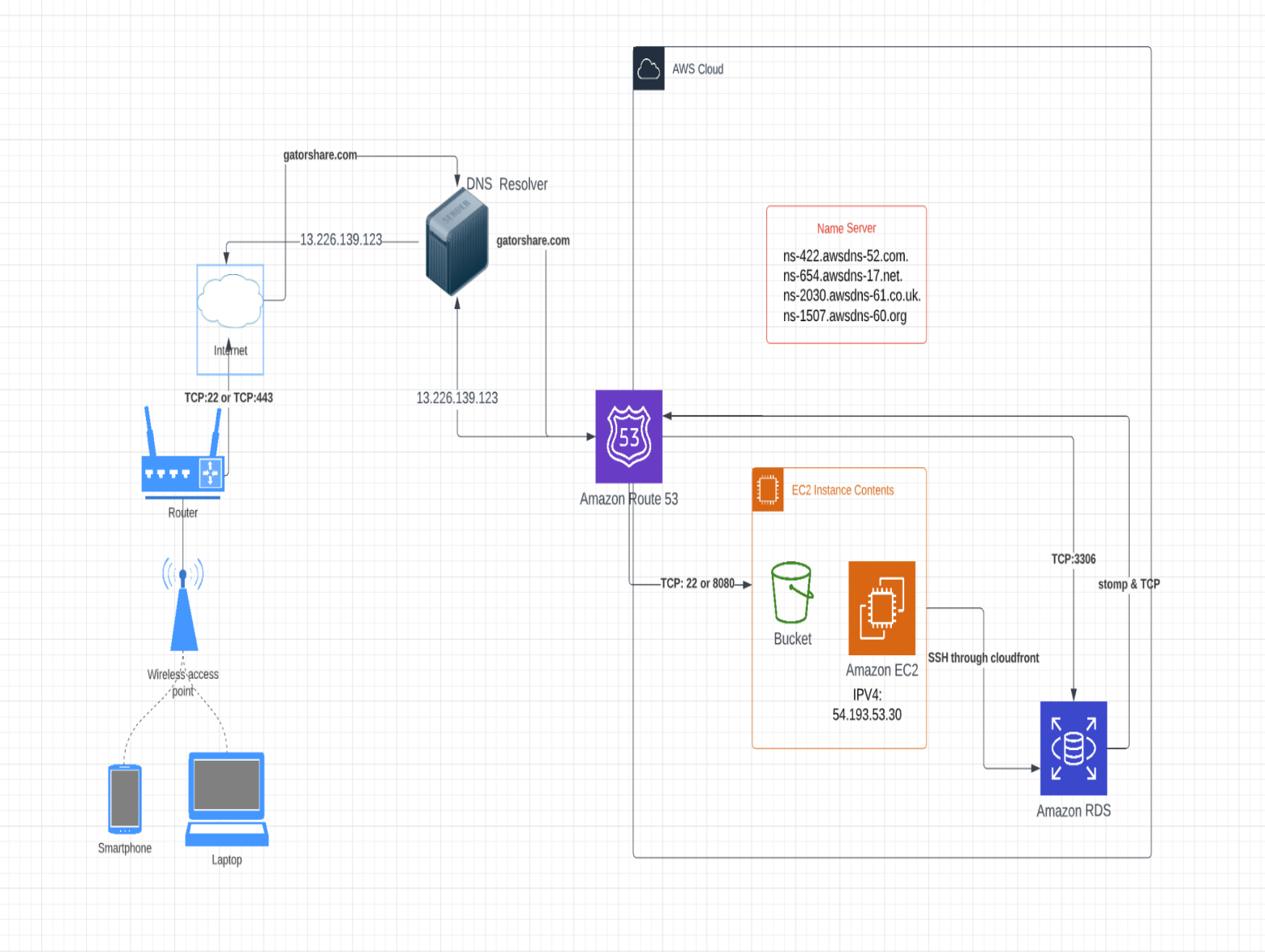


High Level Application Network and Deployment Diagrams

Deployment Diagram:



Application Network Diagram:



Identify Actual Key Risks for your project at this time

Skills Risks -

One of the biggest risks we have stumbled upon was that we had to learn new technologies. For instance, although the majority of us do have some degree of knowledge when it comes to SQL, we do not have much hands-on experience in the programming language itself. Therefore, we had to revise our ERD multiple times due to our lack of understanding attributes and entities. Our solution for this issue is to work together during team meetings and on Discord to help revise and finalize our ERD. In the future, we plan to continue to work together as a team in order to help one another who may need help.

Schedule Risks -

We do not have any drastic scheduling issues overall. Although, on very rare occasions, we have found that we need to reschedule some team meetings. Fortunately, all of us are usually very flexible with rescheduling, therefore there aren't many issues. One of our solutions when rescheduling is necessary, is that we try to make time directly after class (either Tuesday or Thursday after 4:25pm) to sync up, if our Monday meetings do not work. During our current meetings, we also remind one another about our next meeting(s) and we also add/move new meetings if we find it necessary.

Technical Risks -

One of the biggest technical risks for our team is how challenging implementing a messaging feature may be. Our idea to create a private messaging feature may be difficult to overcome due

to not only the short amount of time, but also the lack of experience coding such a feature. Securing our database poses another risk for our application. Due to our lack of experience with databases overall, we do not know how to secure our database to the highest standards. Our solution for this for now is to learn and to do as much research as possible. We also plan to reach out to the CTO for helpful pointers and advice if needed.

Teamwork Risks -

So far, our team has divided all the required work based on the members with the most suitable skills. For example, the Front End team worked on the Storyboard Mockups, while the Back End lead worked on API research and the Network diagram. When our members need help, other members will step in to give feedback, input, and a helping hand. There has been no overall teamwork risk at the time. Although our solution to solve any teamwork risk is to be open and communicative and allow one another to compromise if necessary.

Legal/Content Risks -

No legal or content risks at the time. Our logo is made independently without plagiarizing any other logo and all visual elements such as current (and future) buttons and icons will be obtained for free and will be properly cited if necessary. Any future additional framework, libraries, services, etc. will be free or an open-source to avoid any legal issues.

Project Management

Our team aims to meet on Zoom at least 1-2x a week. Our Zoom room is also open and available 24/7, which allows teammates to meet one-on-one if necessary. During every meeting, the Team Lead walks through each Milestone criteria, and everyone discusses each section requirement. This involves brainstorming, asking questions, and analyzing concepts. The Team Lead also notes questions that the team is stuck on, and circles these questions back to the CTO for further explanation. Walking through each Milestone criteria allows everyone to be on the same page, and everyone understands what still needs to be done. The Team Lead is able to track all of the team members' work this way as well. The Team Lead also makes sure to ask teammates to communicate their ideas, questions, and feedback. Most of the time, our team communicates through Discord. We have different channels for each team, that way everything is organized and members can work with one another independently. We also utilize Trello to help manage tasks. Tasks are organized based on Milestone Doc, Frontend, Backend, Database, and Github. When completed, each team member will write their names and move it to “Completed”. This allows for the Team Lead to track contributions. We plan to continue utilizing both Discord and Trello until the finalization of our application.

Detailed List of Contributions

Estefanos (Back End Lead) -

Researched, drafted and finalized the High Level API and Main Algorithm. Drafted and finalized the High Level UML diagram. Drafted and finalized the High Level Network and Deployment Diagram. Contributed to Data Definitions. Communicated consistently in Discord. Had open conversations and discussions with the team. Heavily contributed to the Vertical Software Prototype. Helped revise and finalized the Database model.

Brianna (Front End Lead) -

Contributed in discussions on Discord. Attended scheduled team meetings. Contributed to use case mockups and sketched the frontends ideas for basic UI wireframes such as the navigation bar, homepage, create post, post/search, etc. Created mockups for use cases no. 1, 2, 4, and 5. Contributed to data definitions. Reviewed and edited/added to data definitions. Reviewed and edited/added some priority functional requirements. Helped review and revise Milestone 2.

Brian (Database Master) -

Drafted and finalized the Database model. Drafted a table within MySQL. Created the Entity Relationship Diagram, and consistently updated it with new entities and attributes.

Communicated on Discord and team meetings. Reviewed, revised and finalized Milestone 2 as the Milestone 2 editor. Contributed to data definitions.

Mohamed (GitHub Master) -

Tested all code by pushing all updated code to the “testing” branch. Moved all tested code to the “master” branch for submission. Worked alongside the Database Master to draft the Entity Relationship Diagram. Helped revise and finalize the Entity Relationship Diagram. Reviewed and revised M1V2. Communicated on Discord and attended team meetings. Contributed to data definitions.

Aleksandr (Front End) -

Contributed to data definitions. Contributed ideas for the following layouts/components: navbar, footer, create a post, posts, homepage, search bar. Drafted and finalized no. 3, 6, and 7 of the Use Case mockups. Added a search bar to the application, and worked with Backend Lead to have a functional and working search bar. Added layouts to application’s Sign Up page, Sign In page, and Navigation Bar. Added icons and buttons to the application. Added notifications to the application. Contributed to M1V2 competitive analysis. Reviewed and revised M1V2. Helped review the Entity Relationship Diagram. Communicated on Discord and team meetings.