Final Project for SW Engineering Class CSC 648-848 Section 01 Summer 2022

Team 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

URL: http://gatorshare.com/

Date: August 2, 2022

Table of Contents

Product Summary

Name of the product:

GatorShare

Product Summary:

Our application is a one-stop website designated for San Francisco State University students, faculty, alumni, and the general public. Our goal is to connect people, allow viewers to explore different social events SFSU has to offer, find Discord servers related to SFSU clubs and courses, and find Tutoring services. Our website also allows Users to post their work and become a part of a community. GatorShare is unique in that our objective is to combine everything into one. Just by accessing GatorShare alone, a Guest will be able to find resources, find Discord channels, view event flyers, read articles written by SFSU peers, and also find Tutors that fit their needs. What makes us different from our competitors is that our service allows our Users to communicate with one another all in one session. Registered Users can interact with other SFSU students by utilizing our commenting and messaging features, they can post their original articles and essays, tutors can be found, and most importantly, our Users can form bonds. Our product helps both new and returning students locate the tools they require to feel welcomed and a member of the SFSU community. We provide an atmosphere in which members of the SFSU community may be connected and discover all the tools they need to support their academic success and social fulfillment on and off campus.

Product URL:
http://gatorshare.com/

Itemized List: Priority 1

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 tab that redirects to the Home page.
6. Navigation Bar shall have a tab that redirects to the Login page.
7. Navigation Bar shall have a tab that redirects to the Sign Up page.
8. Navigation Bar shall have a tab that redirects to the Posts page.
9. Navigation Bar shall have a tab that redirects to the Create a Post page.
    - Create a Post page shall be visible only for Registered Users.
10. Navigation Bar shall have a tab that redirects to the Messages page.
    - Messages page shall be visible only for Registered Users.
11. Navigation Bar shall have a tab that redirects to the Profile Page.
    - Profile page shall be visible only for Registered Users.
12. Home page shall have a “Most Recent Posts” section.
13. Home page shall have a “Most Popular Posts” section.
14. Home page shall have six tabs, organized by content:
    - “Articles & Essays”, “Art & Film”. “Clubs”, “Discords”, “Tutoring” and “Other”.
15. Footer shall have an About Us link.
16. Footer shall have the Terms & Conditions link.

17. Footer shall have the address of San Francisco State University.
18. Posts shall be public for all Users.
19. Comments shall be public for all Users.
20. Images shall be accessible for all Users.

Requirements for Unregistered User:

1. Unregistered Users shall be able to view all Posts.
2. Unregistered Users shall be able to view all Comments.
3. Unregistered Users shall be able to utilize the Search bar.

Requirement for Registered Users:

21. Registered Users shall be able to add a Post onto the platform.
22. Registered Users shall be able to add Comments on another Registered User's Posts.
23. Registered Users shall have a Profile.
24. Registered Users shall be able to send messages.
25. Registered Users shall be able to receive messages.
26. Registered Users shall be able to like a Post.
27. Registered Users shall be able to delete their Post.

Requirements for Posts:

28. Posts shall display the Registered User who created the post.
29. Posts shall include a comment box.
30. Posts shall include a "Leave a Comment" submission button.
31. Posts shall display the time of the submission from present time.
32. Posts shall include a like button.

33. Posts shall display the number of likes it has received.

Requirements for Comments:

34. Comments shall display the username of the Registered User who created the comment.
35. Comments shall display the time of the comment submission from present time.
36. Comments shall be text only.

Requirements for Admins:

37. Admins shall be a Registered User.
38. Admins shall be able to Post.
39. Admins shall be able to Comment.
40. Admins shall be able to send messages.
41. Admins shall be able to receive messages.
42. Admins shall be able to like a Post.
43. Admins Users shall be able to receive notifications from Registered Users.

Requirements for Professors:

44. Professors shall be a Registered User.
45. Professors shall be able to receive notifications from other Registered Users.
46. Professors shall have a Profile.
47. Professors shall be able to Comment.
48. Professors shall be able to send messages.
49. Professors shall be able to receive messages.

Requirements for Tutors:

50. Tutors shall be a Registered User.

51. Tutors shall be able to receive notifications from other Registered Users.

52. Tutors shall have a Profile.

53. Tutors shall be able to Post.

54. Tutors shall be able to delete their Post.

55. Tutors shall be able to Comment.

56. Tutors shall be able to send messages.

57. Tutors shall be able to receive messages.

58. Tutors shall be able to like a Post

Milestone 01

Table of Contents

Executive Summary

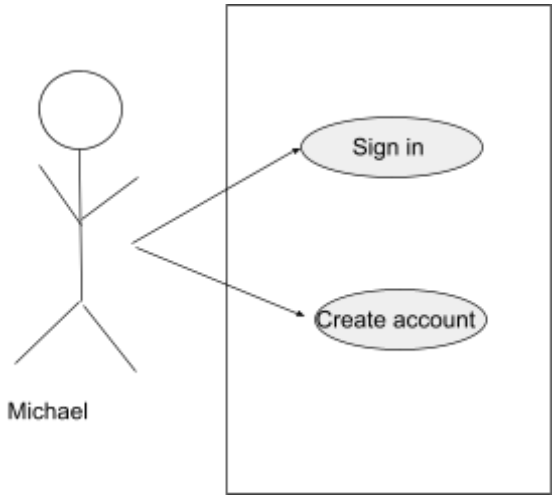
Throughout the years of online learning, San Francisco State University's social networks have become extremely limited for students. Today, students are restricted to only the school's unofficial Discord server and SFSU's homepage. Online learning has shown that many student's social platforms are reduced to Zoom calls, iLearn discussions and casual conversations on Discord channels. We see many students aren't as sociable or approachable due to online constraints.

Our motive is to create a platform where all students are welcomed to communicate with one another and to openly share their thoughts, hobbies, creative skills and social events. Our goal is to build a stronger community within SFSU's online medium by integrating everything into one. With our application, students can communicate with one another and aren't restricted to a few Discord channels. Students can post articles and essays to share their thoughts, post their art, photography and films for recognition and feedback, share club information and meetup times to the public, share Discord servers, and even find tutoring services— all available in one web application. Although our target group is aimed towards students, our platform welcomes faculty and alumni as well. Students do not need to roam around SFSU's website to search for a link, nor do they need to search for different Discord channels in hopes to find an event post. Our goal is to allow a platform where every student can welcome one another, with a focus on sharing content and building greater connections.

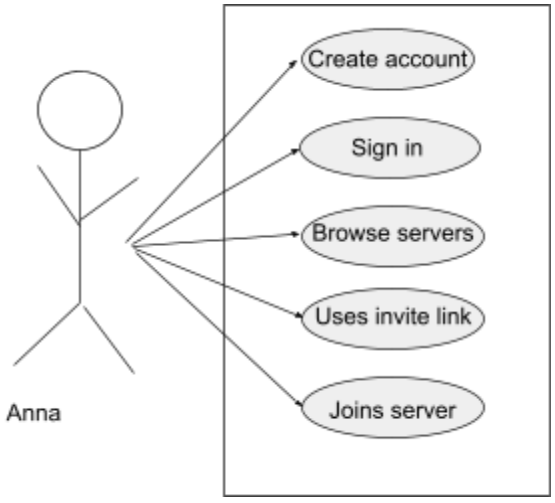
Our system is unique in a sense where nothing is scattered in several different channels or links. We want to create a community where students can share their ideas and receive positive exposure all in one platform.

Main Use Cases

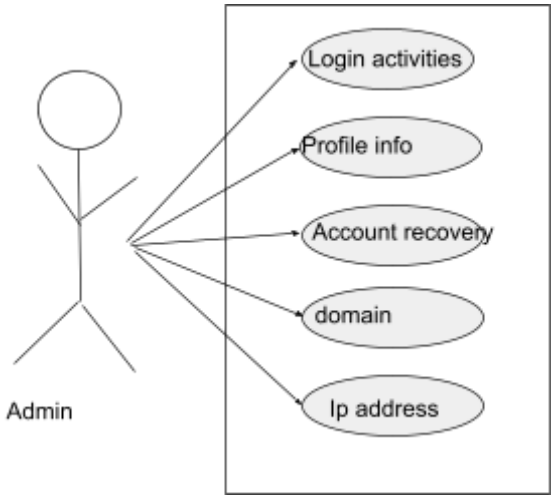
Use Case Title:	Signing up
Actors	Student
Description	Michael is a student of SFSU. He just started his first semester and would like to insert himself in the campus. His classmate tells him about the GatorShore platform where he can exchange with other students and more. So Michael goes on the platform and register so he can have fully access to the functionalities of the platform. He fills in his informations and now can share and see others' posts.



Use Case Title	Finding a Discord Server
Actor:	Registered User - Student
Description:	Anna is a first-year student at SFSU. She wants to join a Discord server and make some new friends. She uses her school email to create an account and log in to our website. Anna notices that she may either search for a specific Discord server with the search bar, or explore all of the servers that have been posted by other registered users. Anna chooses to browse all of the available servers, looking at which servers interest her. After finding a Discord server she likes, Anna clicks the given invite link to the server. There, she is connected to the Discord server and is able to make new friends.

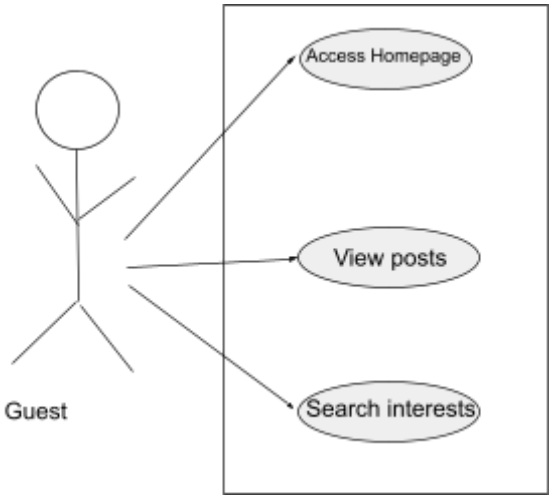


Use Case Title	Deleting Suspicious Activity
Actor:	Registered User - Admin
Description:	When logging in, the Admin are able to view the entire platform. They are allowed to see the login activities of the user. Such as changes of passwords, information, and account recovery. They can also review reports of malicious activities. The report is the composition of domain, login date and time, and Ip address; it can be the user's physical location, proxy server or virtual private network.



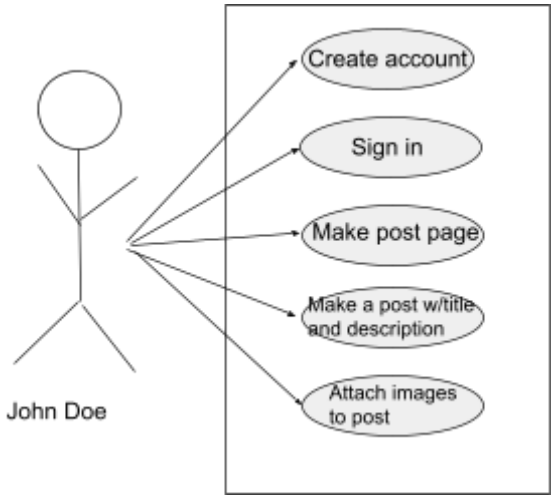
Use Case Title:	Guest
Actors	Unregistered User
Description	A user who has not registered an account or is accessing the page while logged out will be considered a Guest. A guest, when accessing our application, can only view posts. The guest will have full access to view

	every page within the platform, meaning that they can browse through all posts, view club events, access articles, etc. This user can also search for his/her interests. The only limitation an unregistered user has is that they cannot post to the platform, they cannot post comments, they cannot reply to other comments, and they cannot like/dislike any post. If the unregistered user wants to make a post, they must sign up and register an account.

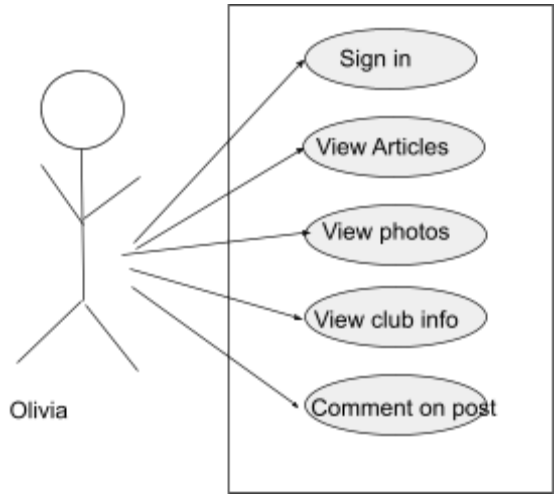


Use Case Title:	Posting
Actors	Registered User - Student
Description	An SFSU student, John Doe, is a Visual Communications major. Throughout his semesters at SFSU, he has built an extensive portfolio of his digital artwork. As John progresses in his academic career, he has decided that he would like to share his work for both publicity and feedback. John stumbles upon GatorShare and sees that other students are posting their original work as well. John decides to register an account, navigate to

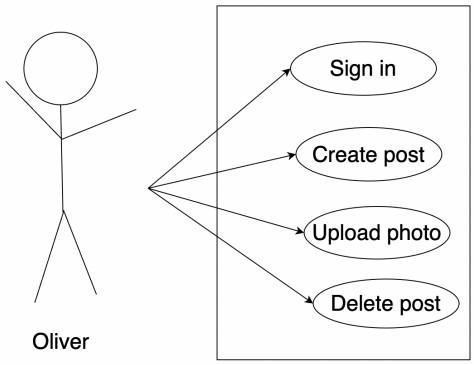
	student posts, and attach several images of his digital artwork. John is required to post a title, post an author name, attach at least one image, and add a small description.



Use Case Title:	Commenting
Actors	Registered User - Student
Description	An SFSU student, Olivia, is a registered user on GatorShare. As she is browsing throughout the website, she stumbles upon several student-written articles, original photography posted by her college peers, and even club flyovers about upcoming events. A student-written article talking about a new restaurant opening at Stonestown Galleria captures Olivia's attention. Olivia, being a registered user, decides that she wants to leave a comment on the post. She scrolls to the bottom of the article, and is able to see a comment box under the post. Here, she leaves a comment.



Use Case Title:	Deleting Post
Actors	Registered User - Student
Description	Oliver is a SFSU student and recently went on a hike with his roommate. He decides to upload a picture he took from Mount Diablo. His roommate, when logging into GatorShare saw Oliver's post and wasn't happy about the submission. Oliver signs back into his account, and is able to delete his post.



List of Main Data Items and Entities

User: Any person who uses the web application.

- **Unregistered User:** A User who has not signed up or logged into the application. This User is allowed to browse and search through the website, but cannot post or comment on the platform. He has the possibility to sign up if he has not yet an account or log in his account.
- **Registered User:** A User who has signed up or logged into the application. This User must sign up via their full name, school email (to verify SFSU student status), 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, comment, view messages, and like/dislike posts 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 and can delete Posts, Comments, and Registered Users.

Post: A submission made by a Registered User. A post is composed of a title, text box, and optional media content. A Post can be plain text used as an article, an uploaded image to promote a digital artwork or event flyer, or a discussion post.

Comment: A Registered User can click on another Registered User's post and comment underneath the Registered Users post. A comment would be text only.

Message: A Registered User can send a private message to another Registered User.

Initial List of Functional Requirements

Website:

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

Requirements for Unregistered Users:

12. Unregistered Users shall not be able to add a Post on the platform.
13. Unregistered Users shall not be able to add Comments to a Post.
14. Unregistered Users shall not be able to receive notifications.
15. Unregistered Users shall not be able to like a Post.
16. Unregistered Users shall not be able to dislike a Post.
17. Unregistered Users shall not be able to like a Comment.
18. Unregistered Users shall not be able to dislike a Comment.

19. Unregistered Users shall not be able to send Messages.
20. Unregistered Users shall not be able to communicate with Registered Users or Admin.

Requirements for Registered Users:

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

Requirements for Posts:

35. Posts shall have a title.
36. Posts shall have a description.
37. Posts shall not have a character limit.

38. Posts shall have an image submission field.
39. Images shall only be accepted in jpg, png, or jpeg format.
40. Posts shall display the Registered User who created the post.
41. Posts shall display the time of the submission.
42. Posts shall include a comment box.
43. Posts shall include a submit button.

Requirements for Comments:

44. Comments shall not be empty.
45. Comments shall display the username of the Registered User who created the comment.
46. Comments shall display the time of the comment submission.
47. Comments shall have a text limit of 500 characters.

Requirements for Admin:

48. Admin shall be able to delete other Users.
49. Admin shall be able to delete other Posts.
50. Admin shall be able to delete other Comments.
51. Admin shall be able to Post.
52. Admin shall be able to edit their Post.
53. Admin shall be able to delete their Post.
54. Admin shall be able to Comment.
55. Admin shall be able to edit their comments.
56. Admin shall be able to delete their comments.
57. Admin shall be able to send messages.
58. Admin shall be able to receive messages.

59. Admin shall be able to delete messages.

60. Admin shall be able to like a Post.

61. Admin shall be able to dislike a Post.

62. Admin shall be able to like a Comment.

63. Admin shall be able to dislike a Comment.

64. Admin Users shall be able to receive notifications from Registered Users.

65. Admin users shall be able to post and edit the “About Me” section of the web application.

List of Non-Functional Requirements

System requirements:

1. The website application shall be optimized for selected browsers such as Safari, Google Chrome, and Mozilla Firefox.
2. The website application shall display posted media in full width of the screen
3. Images posted on the application shall be in the format of jpg, png, or jpeg only.

Performance requirements:

4. The website application shall be optimized for desktop/laptop usage.
5. Loading time for the website shall be 5 seconds or less for any page or screen.

Storage, security, environmental requirements

6. The website shall handle at least 50 users simultaneously.
7. Data shall be stored in the team's SQL database.

Marketing, legal requirements (logos, branding, licensing)

8. Certain logos were created for this project only.
9. Logos from San Francisco State University were used on the website since this platform is for San Francisco State University.
10. All icons and frameworks used shall be from a free, open-source library.

Content (size, formats…)

11. Comments shall have a text limit of 250 characters.

Privacy (what data is collected, how is it used…)

12. The login shall be required for all Registered Users (Approved Users and Admin).
13. Passwords shall be encrypted in the database where it is stored.
14. The Registered Users session shall end once they click “Log Out”.

Optimization & Performance:

15. The website application language shall be in English.
16. The application will upload User’s posts and comments in real time.
17. The website application shall be deployed from the team's Amazon Web Services account.
18. The website application shall store the Users data in the team’s MySQL database.
19. User information shall not be duplicated in the database.

Security:

20. Users data shall be securely stored in the backend database.
21. Passwords shall be protected.
22. Passwords shall not be stored in the database as plain text.
23. When an Unregistered User signs up:
    - Passwords shall be at least 8 characters.
    - Passwords shall require at least one upper case letter.
    - Passwords shall require at least one lowercase letter.
    - Passwords shall require at least one number.
    - Passwords shall require at least one special character.

Competitive Analysis

Features	CourseHero	Chegg	SFSU Discord	SFSU Website	SFSU iLearn	SFSU Reddit	GatorShare
Search/Filter	+	+	+	++	-	+	+
User Profiles	++	++	++	++	++	+	+
Like/Dislike	+	++	+	-	-	++	+
Messaging	-	-	+	-	-	+	+
Commenting	-	+	+	+	+	+	+
Online Status	-	-	+	-	-	+	-
Notifications	+	+	+	-	++	+	+
Posting	+	+	+	+	+	+	+

Features	CourseHero	Chegg	SFSU Discord	SFSU Website	SFSU iLearn	SFSU Reddit	GatorShare
Strengths	Has an abundance of different courses from schools all over the United States.	Provides a wide range of services for students. Has a wide community of tutors, and contributors.	Widely used by SFSU faculty and students. One of the most popular online platforms for all SFSU members.	Has an abundance of information regarding SFSU. All information is accurate.	One of the main platforms used by students, faculty, etc. All learning material is centralized and consistent amongst all pages and courses.	No need to subscribe to the profile in order to see posts and interact.	No subscription necessary. Able to interact and find accurate information.
Weaknesses	Lots of reposted uploaded files and repetitive answers. Must have a monthly subscription in order to	Subscription is needed in order to see solutions. Very easy to be banned (particularly when using not Chegg search), Post	Has user limitations such as gifs and longer messages. Must subscribe to “Nitro” in order to maximize	Hard to navigate due to different tabs and pages. Many different links that redirect to pages with more links. Bad UI	Only used for course-related material. Private chat is messy and is not utilized amongst most users. User experience is	It is not an official platform of SFSU. Few people know about it. There are many pages. Spam posts can be found	Due to its new application, it does not have a lot of information or posts catered to SFSU students or

	access the platform.	quality is not guaranteed.	Discord benefits. Does not have a lot of memory space. Lack of safety in large servers and many spam accounts/posts .	design.	“boring”.	due to unmoderated web pages.	faculty. Lack of information and participants.
Social Media	No strong marketing strategy. Information is inconsistent at times.	Their marketing strategy is very poor. Lack of information especially about particular services and the information is not consistent. Poor customer support	Strong online presence. Discord servers are easily shared and posted around. Many different channels and servers that can be fitting for anyone and everyone.	No strong social media presence such as on Instagram and Twitter. Too many different social media accounts to keep up with. Not many things are centralized.	Because this platform is used strictly by registered SFSU students and faculty members, there is no strong social media presence for privacy reasons.	The platform is messy and not advertised. No strong social media presence.	Due to its new presence, it does not have a strong social media presence. Cannot share posts to outside platforms.
Onboardi	Messy user interface	Several types of	Very easy and straightforwar	Very messy experience		The lack of school	Able to navigate

ng Experienc e	with too many features which makes navigating throughout the website confusing.	subscriptions make the UX confusing. Often contributors copy/paste the information. Customer support is slow and takes some time for a response.	d to navigate around. Very user friendly with different features such as chat perks, gifs, emojis, server boosts, etc.	with too much information in one homepage. Web page design is messy and hard to navigate to find the correct pages. Some pages have information that has not been updated for years.		officials make it difficult to guarantee the received information.	throughout the platform very smoothly. The presence of Admins allows for a clean website without any spam posts. Accurate information is shared.

(++) Superior
(+) Same
(-) Does not exist

Summary:

Despite GatorShare being average overall compared to its competitors, our application is the only platform that aims to integrate everything into one. When assessing the chart, we can observe that some of our competitors may be superior to our application, or just about the same. When it comes to searching and filtering, GatorShare is the same as their competitors but serves different information. GatorShare allows users to search posts in order to find tutors, homework answers, events, discussion posts, etc. This can be seen as advantageous simply because CourseHero and Chegg strictly adhere to academic-related content, while Discord only filters comments and users from specific servers, and Reddit may have spam and unsupervised posts, and random discussions unrelated to SFSU. GatorShare is monitored by their Admins, where everything must be related to SFSU and students have a free range to start discussions, post events and flyers, and also search for tutors/get homework answers. SFSU's website has a search engine far more superior than GatorShare simply because of their implementation of Google's search engine, something we are not utilizing. It is worth noting that many of our other competitors are either far more superior with functional user profiles than GatorShare, or about the same. Our application does not have an emphasis on the functionality of user profiles, as our main focus is the capabilities of our overall platform instead. We can also see that SFSU's Website and SFSU's iLearn lacks a like/dislike feature, and our other competitors such as Chegg and Reddit may be more desirable. Our application does not have a system like Chegg, where users are compensated for having likes (on an answer), nor does it have a system like Reddit, where a post may be "hot" or "popular" based on the number of likes. Our like/dislike feature is mainly used as a means to support another User's content. We also pride ourselves in our messaging and commenting features, as our messaging system allows one SFSU user to connect with one another and our commenting system allows SFSU's to share compliments, give input, and admire other students for their original work. Overall, although our application may be seen as average, this should not be seen as a disadvantage, since GatorShare has a different goal in mind. Our platform aims to create both an online learning platform and a social website for our peers. It is made by students and is catered to students

High-Level System Architecture and Technologies Used

Software: Postman, Swagger Editor

Programming Language: Java, Javascript, SQL

Markup languages: HTML, CSS, JSON

Database System: MySQL.

Deployment Platform: Amazon Web Services, AWS Amazon Linux (Inferred)

Web Server: Apache Tomcat

Frameworks:

- Front End
    - Bootstrap, React
- Back End
    - Spring, Reddis

APIs:

- Spring Boot

IDE

- Microsoft Visual Studio Code
- Sublime Text
- Eclipse
- Intellij

Checklist

1. Team found a time slot to meet outside of the class

DONE

2. Github master chosen

DONE

3. Team decided and agreed together on using the listed SW tools and deployment server

DONE

4. Team ready and able to use the chosen back and front end frameworks and those who need to learn are working on learning and practicing

DONE

5. Team lead ensured that all team members read the final M1 and agree/ understand it before submission

DONE

6. Github organized as discussed in class (e.g. master branch, development branch, folder for milestone documents etc.)

DONE

List of Team Contributions

Donna Nguyen, Team Lead	Drafted the Executive Summary, where the team reviewed. Team Lead finalized the summary alongside M1 Editor in order for the whole team to be on the same page. Helped contribute to main use cases, and reviewed each case in order to make sure each implementation is logical and relevant to the applications purpose. Drafted the list of main data and entities with the help of teammates. Afterwards, Team Lead then reviewed and finalized alongside M1 Editor. Drafted the list of functional and nonfunctional requirements then had the team review each requirement. Team Lead finalized the requirements alongside the M1 editor to make sure each requirement is feasible. Led team meetings and checked in on team members. Kept up with questions, communication, and documentation. Reviewed and finalized high-level system architecture. Summarized the competitive analysis with the help of teammates research and assessment.
Estefanos Kebebew, Back-End Lead	Successfully registered the application's domain from AWS Cloud. Researched where to host the website that AWS provided. Tested several different types of cloud storages such as using amplify, S3 Bucket, and elastic beanstalk. Decided to go with S3 Bucket and used Cloudfront to access the bucket. Used Cloudfront to create SSL certification. Debugged back-end and server-side errors to help the application project run. Implemented AWS RDS and AWS Instance to host the database. Helped teammates connect to the AWS Cloud server. Helped teammates connect to MySQL database. Efficiently fixed bugs related to the backend. Helped with constructing main use case ideas. Reviewed and finalized functional and nonfunctional requirements to make sure all requirements are feasible. Reviewed and finalized high-level system architecture. Researched and contributed to competitive analysis.
Brianna Soukup, Front-End Lead	Contributed to many discussions and ideas in the team's Discord server. Proposed the idea of our web application. Provided ideas for the basic function of our application's purpose. Contributed to main use cases by coming up with main use case ideas. Created use case diagrams for each main use case. Helped come up with, and created the application's logo for our website. Created a shared Figma file for the frontend team to utilize the creation of user flow diagrams and wireframes in the near future. Reviewed and finalized functional and nonfunctional requirements to

	confirm that the requirements are feasible. Researched and contributed to competitive analysis.
Mohamed Toure, Github Master	Designated M1 editor. Moved code in the development branch into the testing branch. Helped test files and finalize code before moving to the master branch. Helped edit M1 documentation. Helped review and finalize executive summary to make sure it's legible. Reviewed main use cases. Researched and contributed to competitive analysis.
Brian Nguyen, Database Master	Helped brainstorm and create a table within MySQL for users with basic information (rough draft). Reviewed and finalized functional and nonfunctional requirements in order to make sure the requirements are feasible and implementable. Contributed to creating data items and entities by coming up with ideas and adding input. Helped review and finalize data items and entities. Helped provide documentation for back-end lead and also provided documentation about the database thus far. Researched and contributed to competitive analysis.
Aleksandr Gusev, Front-End	Created the front end codebase (React App) and created a rough outline of the web application. Created and designed the "About Us" page. Helped sketch a rough draft of the user flow diagram. Contributed to main use cases along with reviewing each use case. Gave input to the Front End Lead about the application's logo design in order to have a presentable logo for display. Reviewed and finalized functional and nonfunctional requirements to confirm the requirements are feasible. Helped manage and maintain the frontend branch of the GitHub repo. Helped with formatting the M1 document. Consistently updated the document's table of contents and reviewed M1 document for errors. Fixed push and overwriting errors on GitHub. Cleaned up repo. Researched and contributed to competitive analysis.

Milestone 02

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:

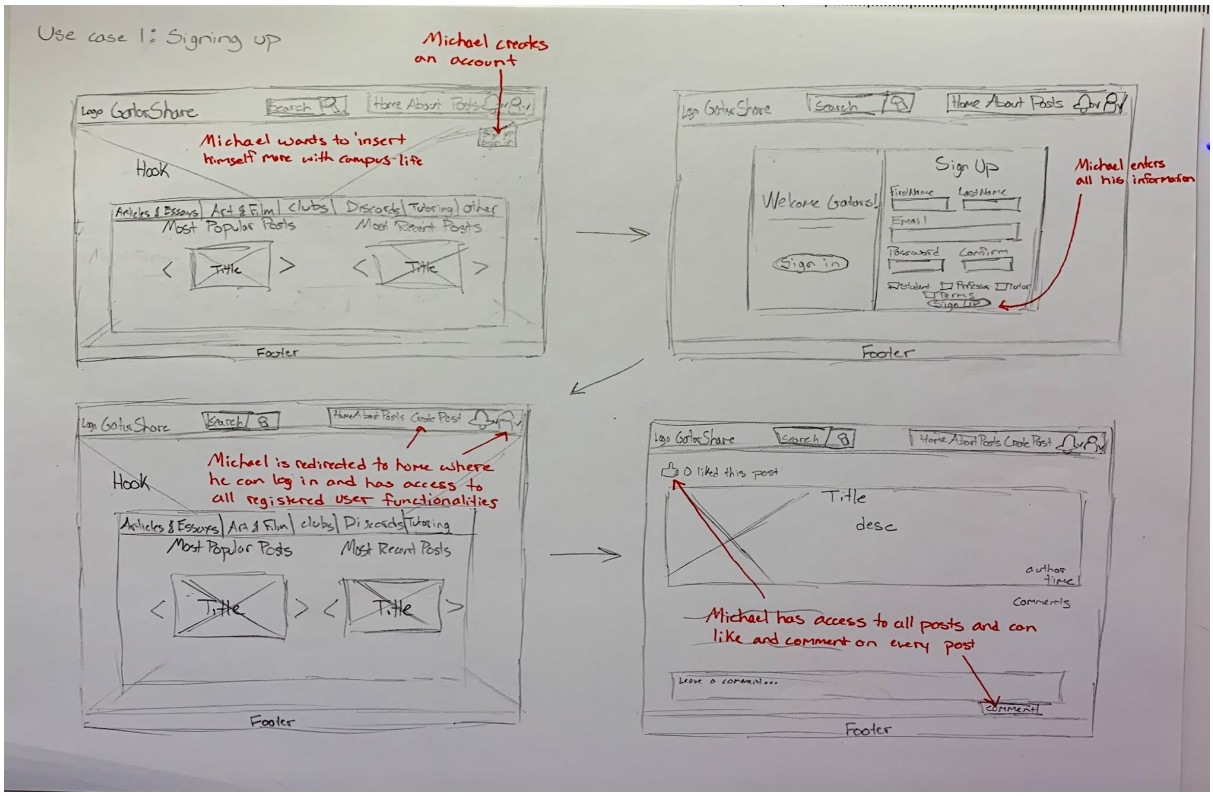
1. All liked posts from users shall be saved.
2. All posts shall have a ‘Report’ button for spam.
3. All comments shall have a ‘Report’ button for spam.
4. Registered Users shall be able to ‘Flag’ a post.
5. Registered Users shall be able to ‘Flag’ a comment.
6. Registered Users shall be able to receive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

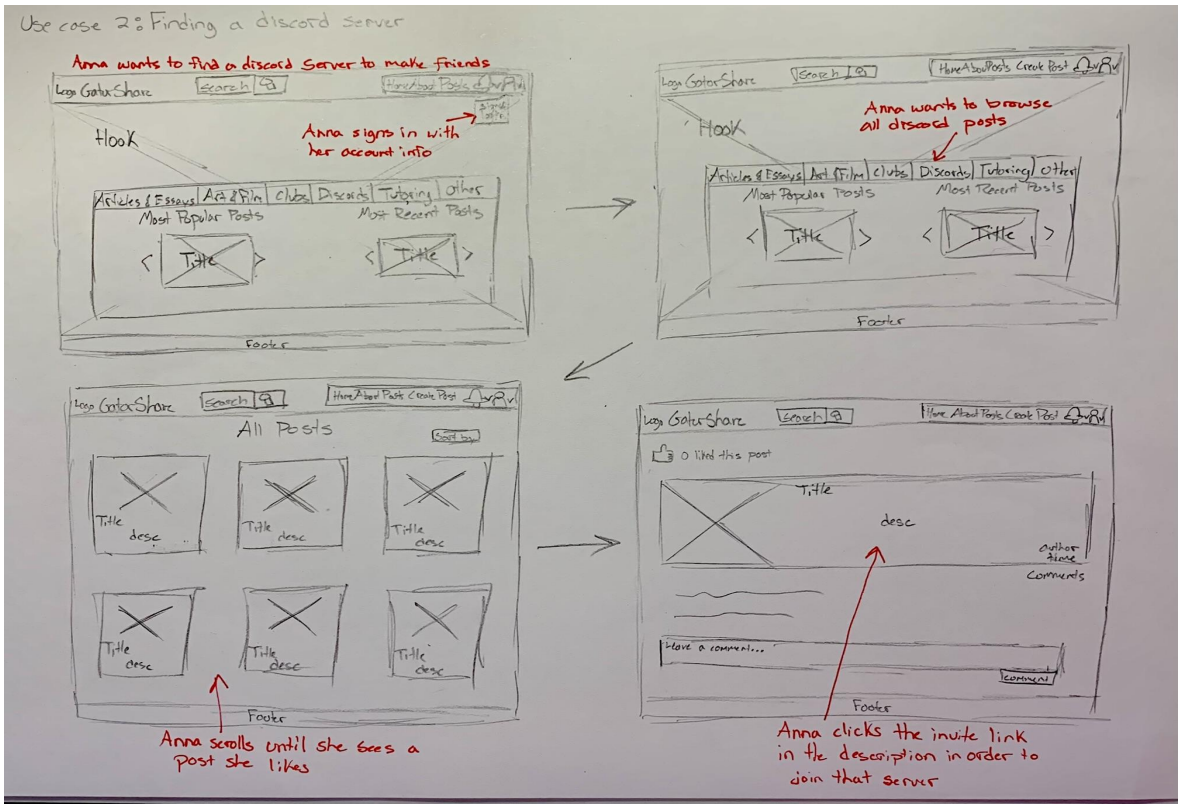
Use Case 1: Signing Up

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.



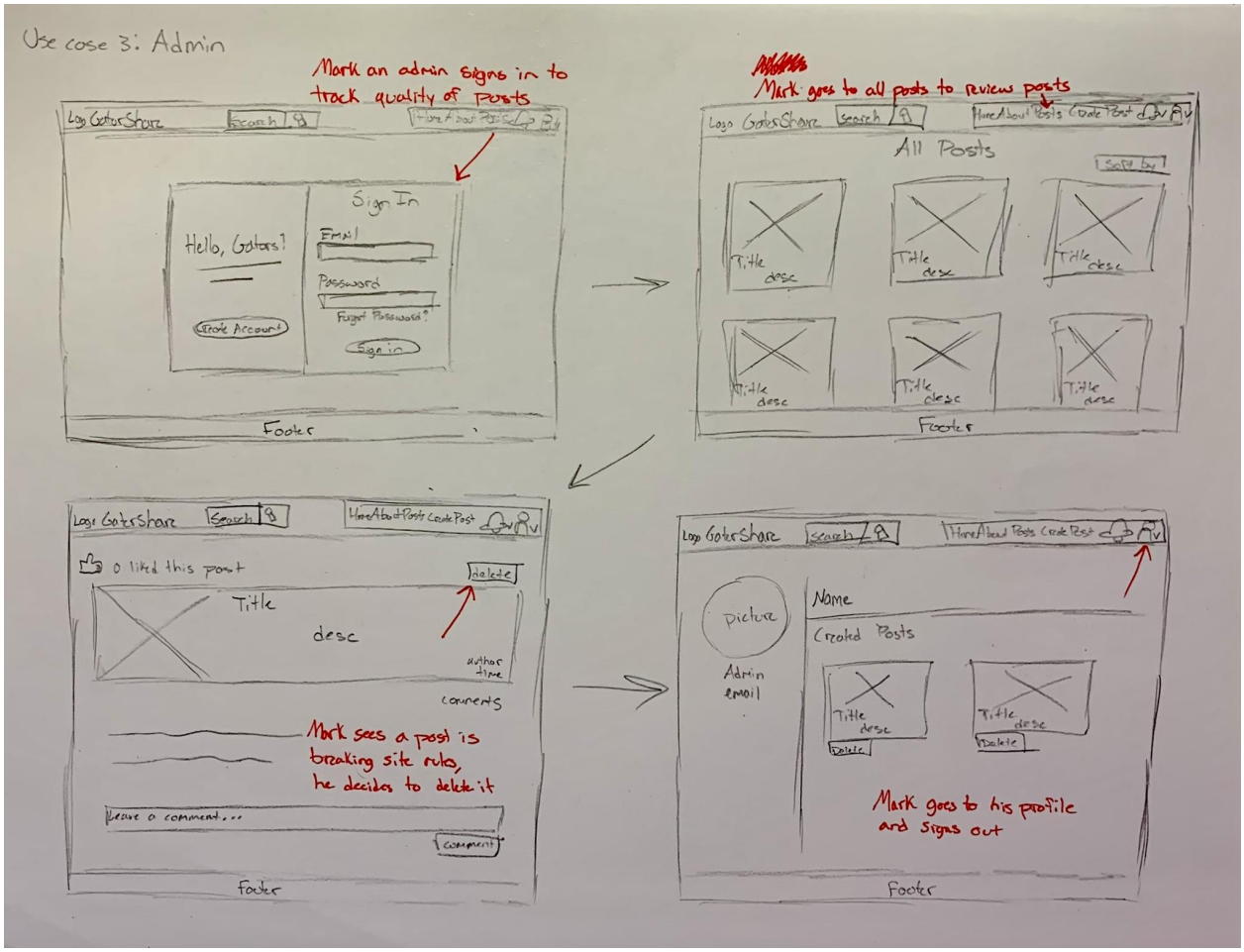
Use Case 2: Finding a discord server

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.



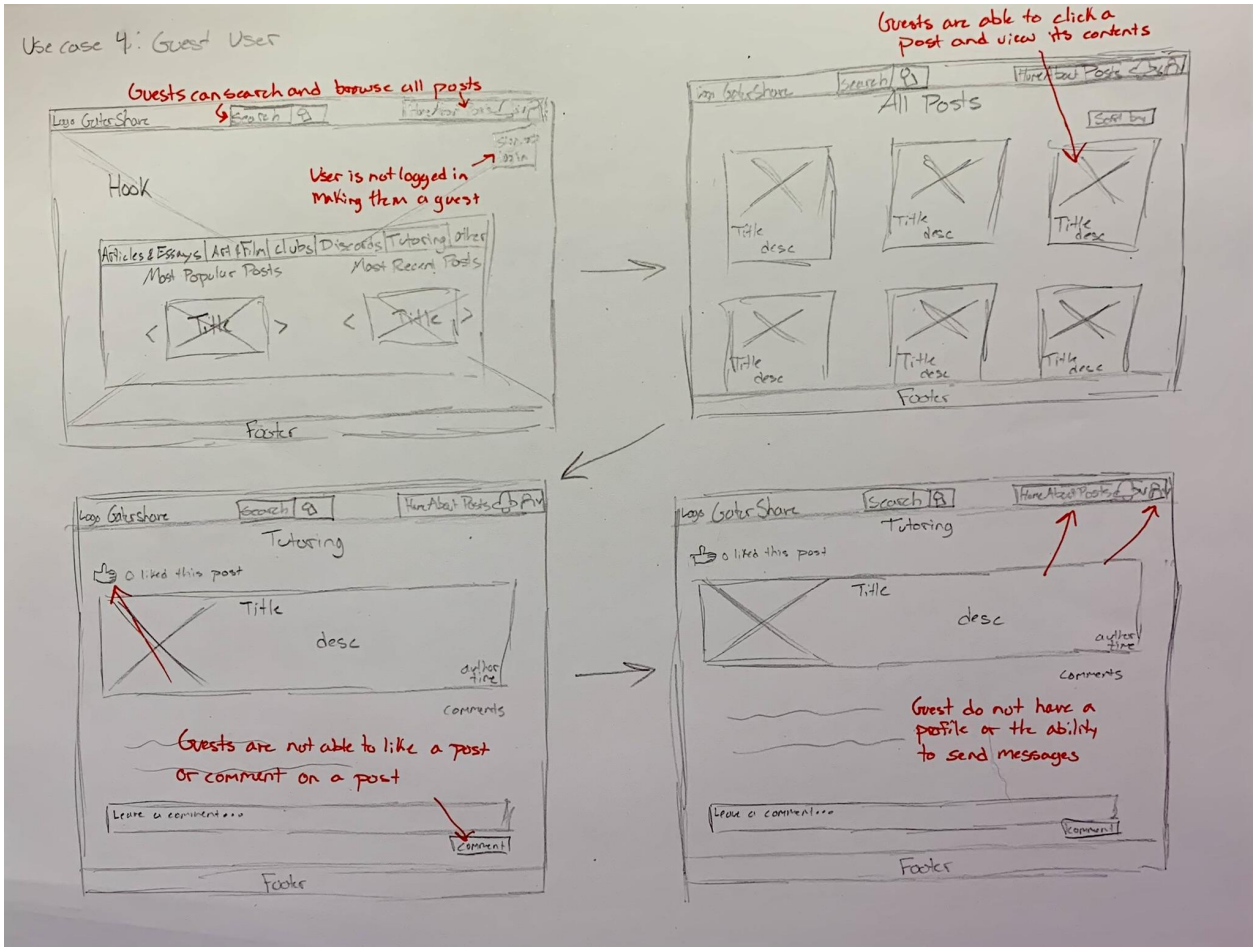
Use Case 3: Admin

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.



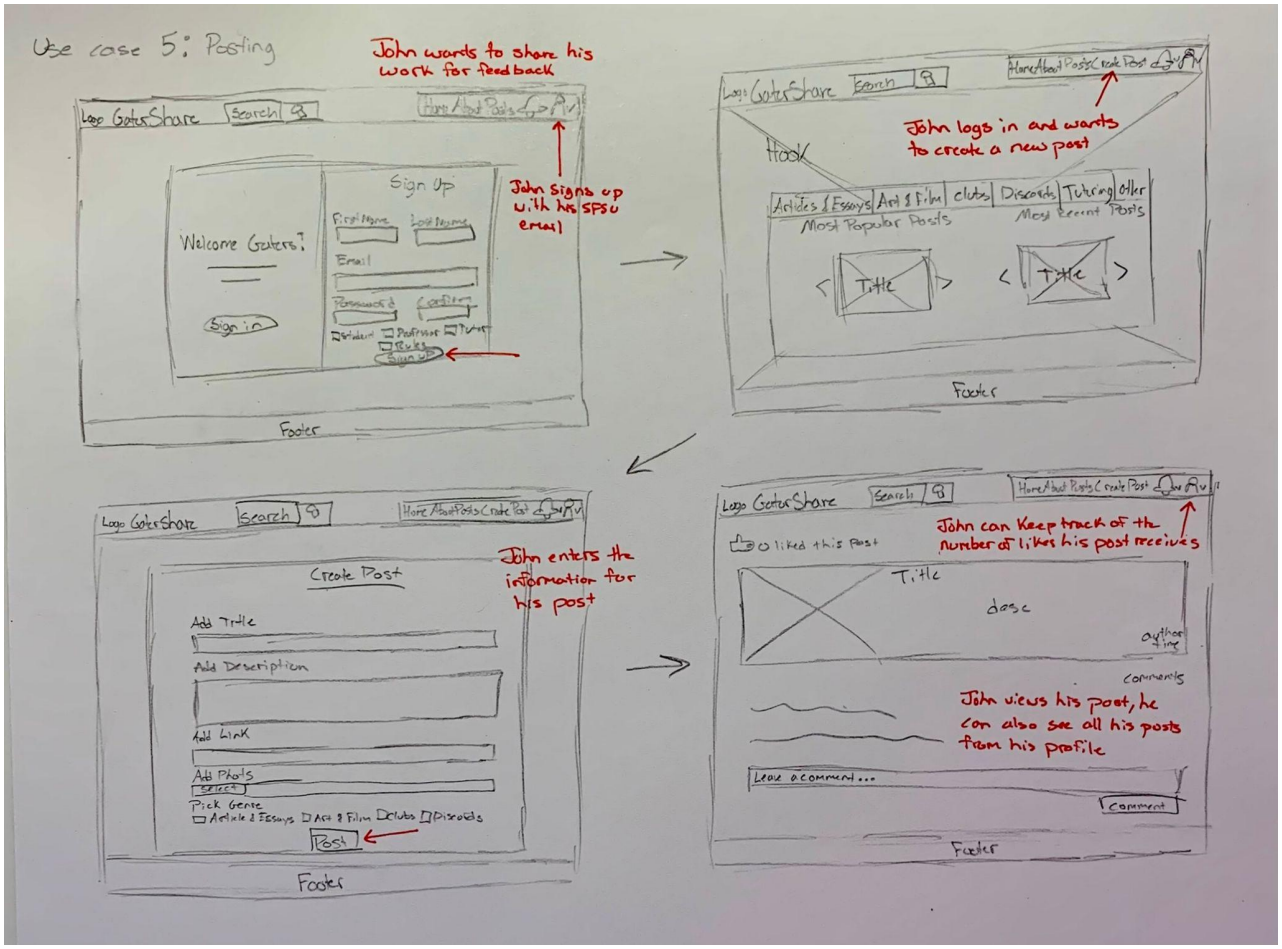
Use Case 4: Guest

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.



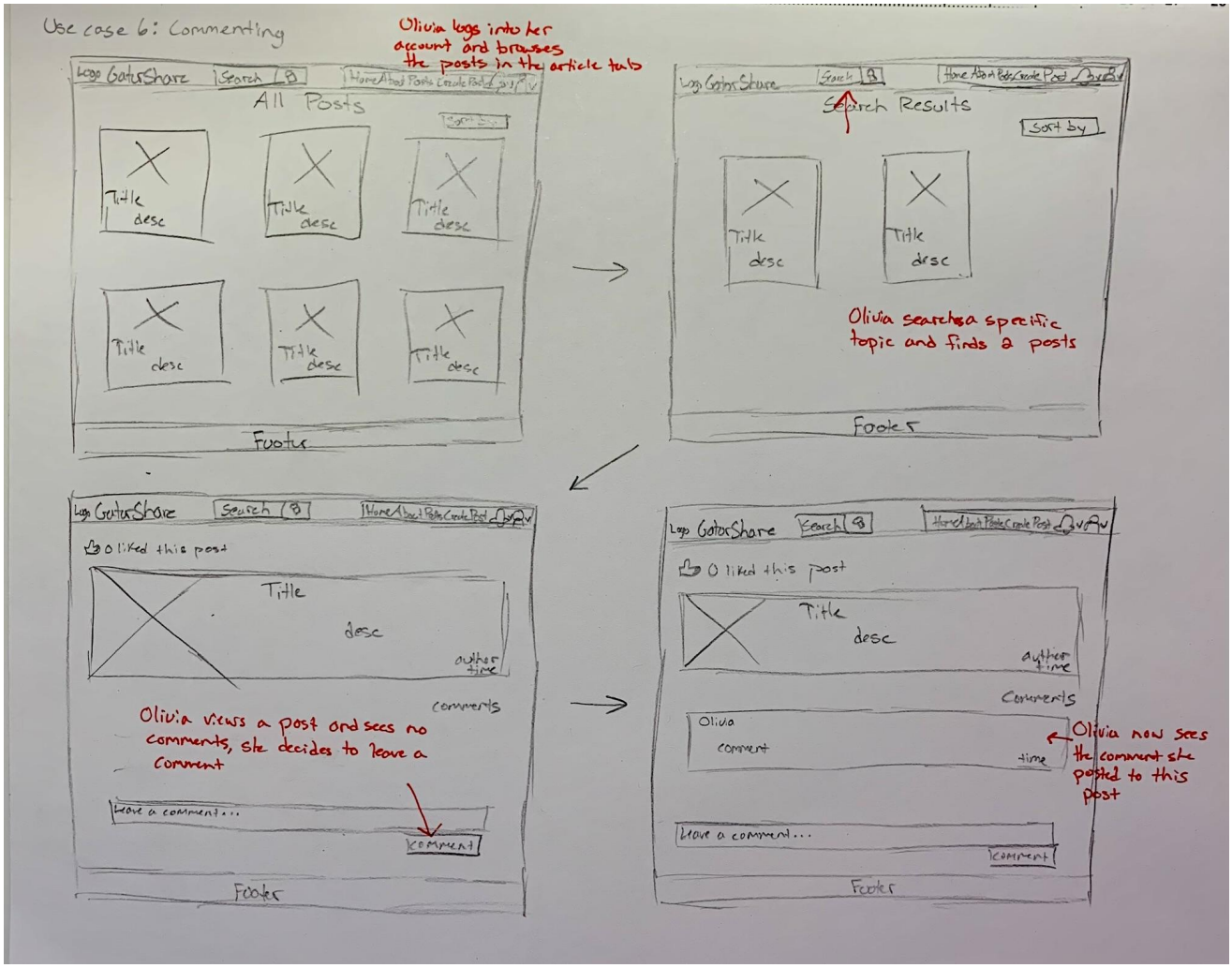
Use Case 5: Posting

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.



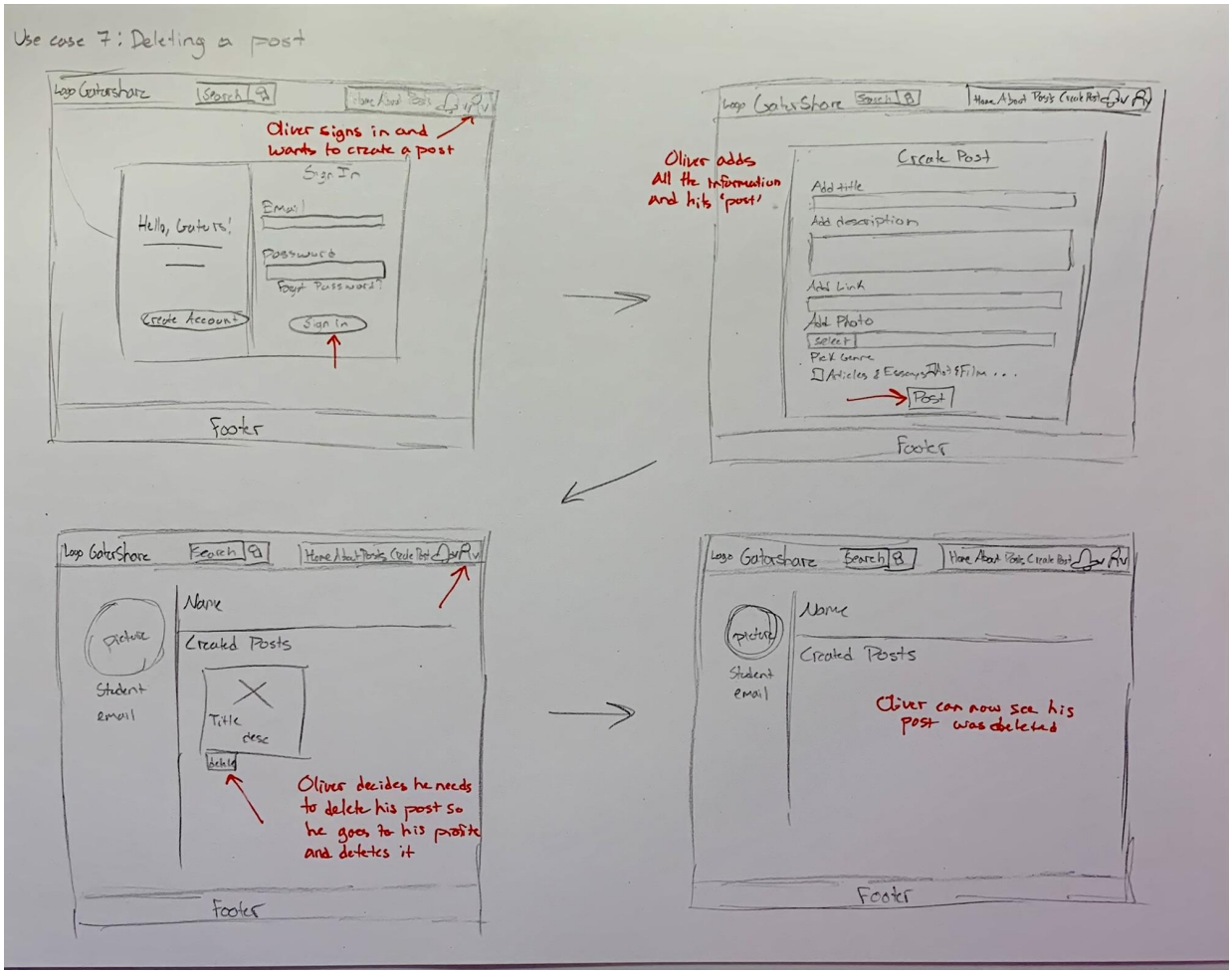
Use Case 6: Commenting

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

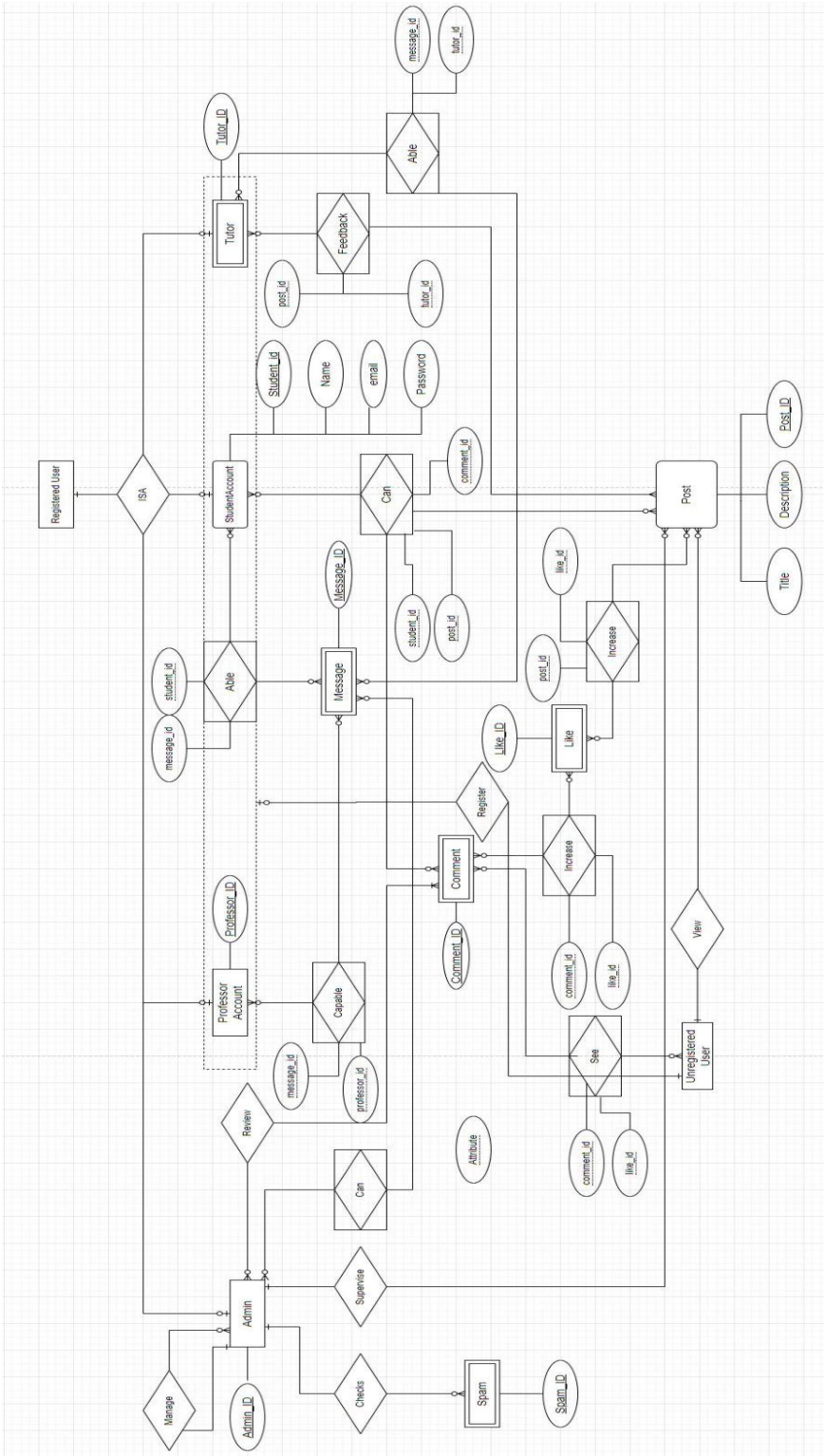
6. Tokens

6.1. token_id, key, numeric

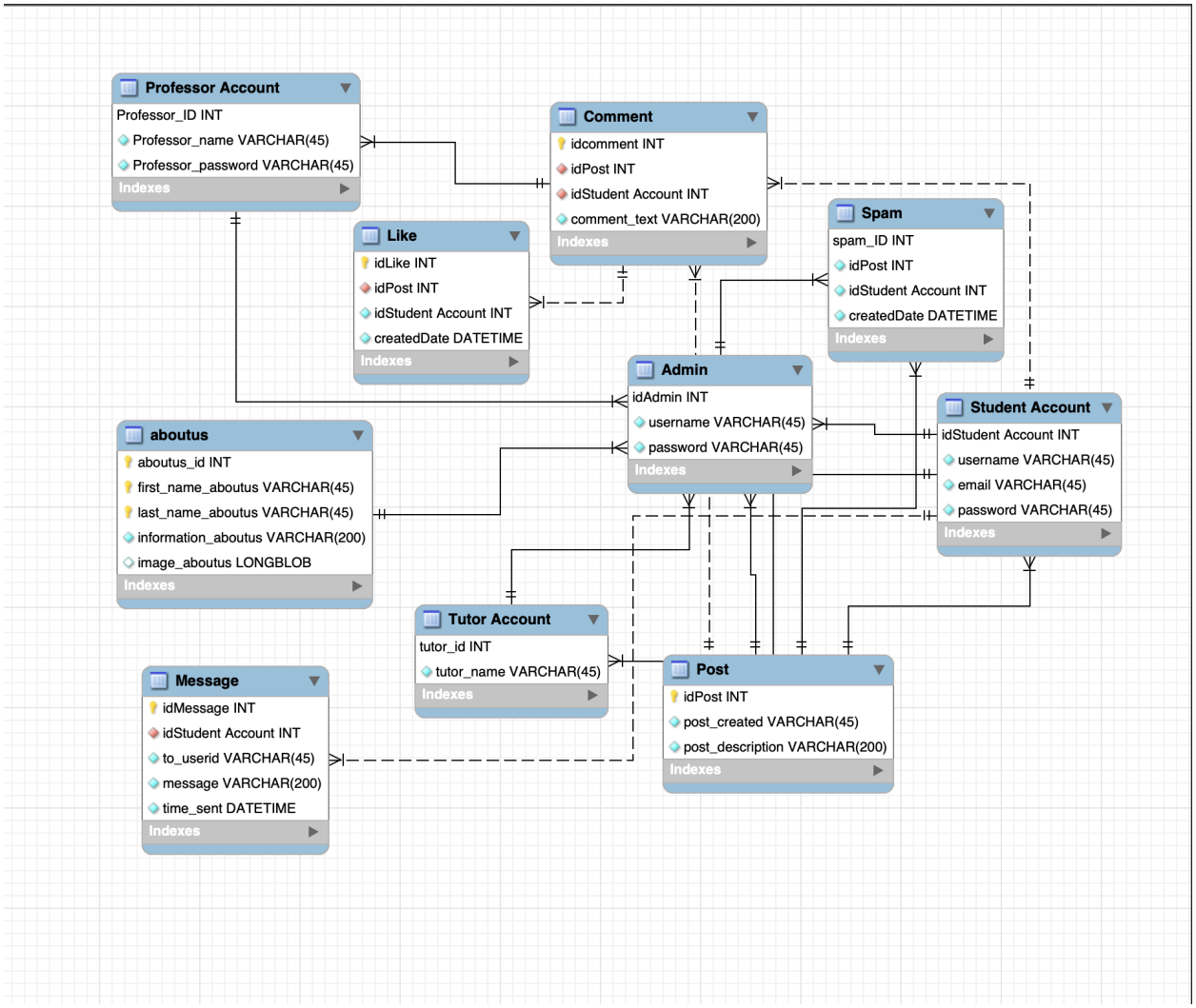
6.2. created_date, multivalue

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

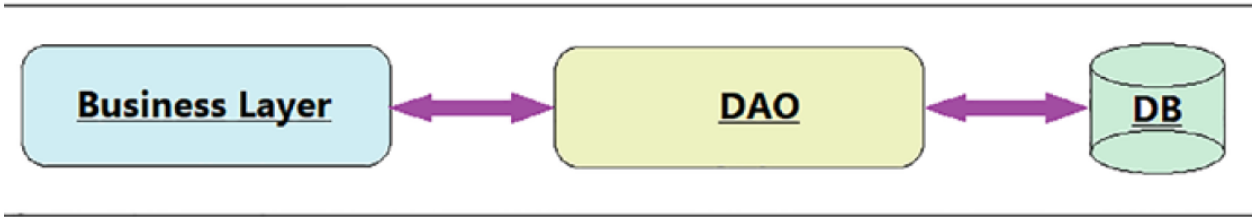
Like	

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	

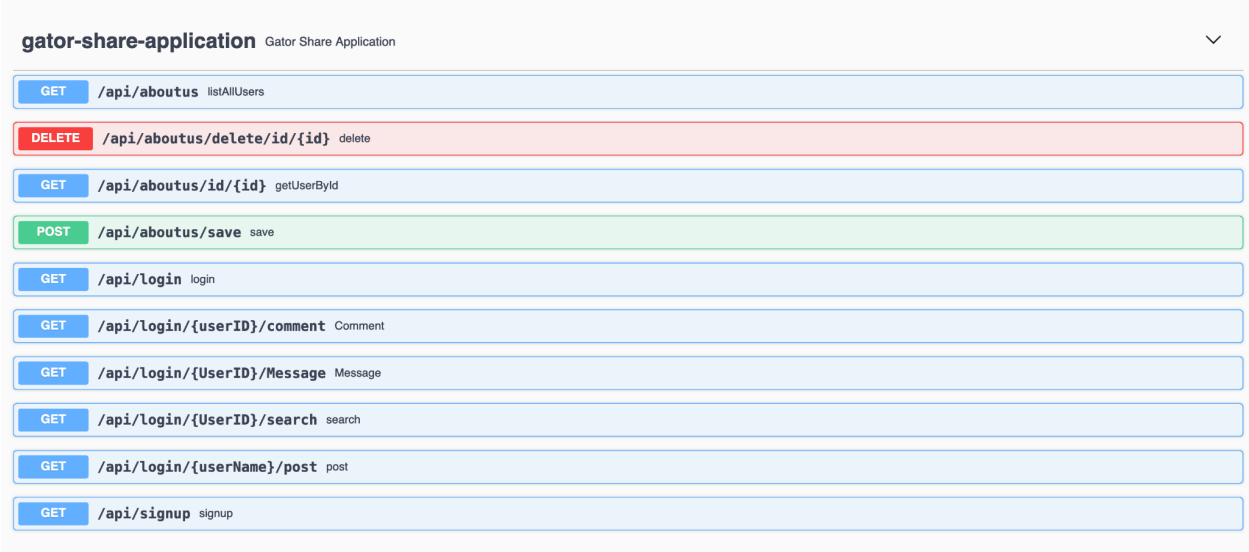
High Level APIs and Main Algorithms

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:

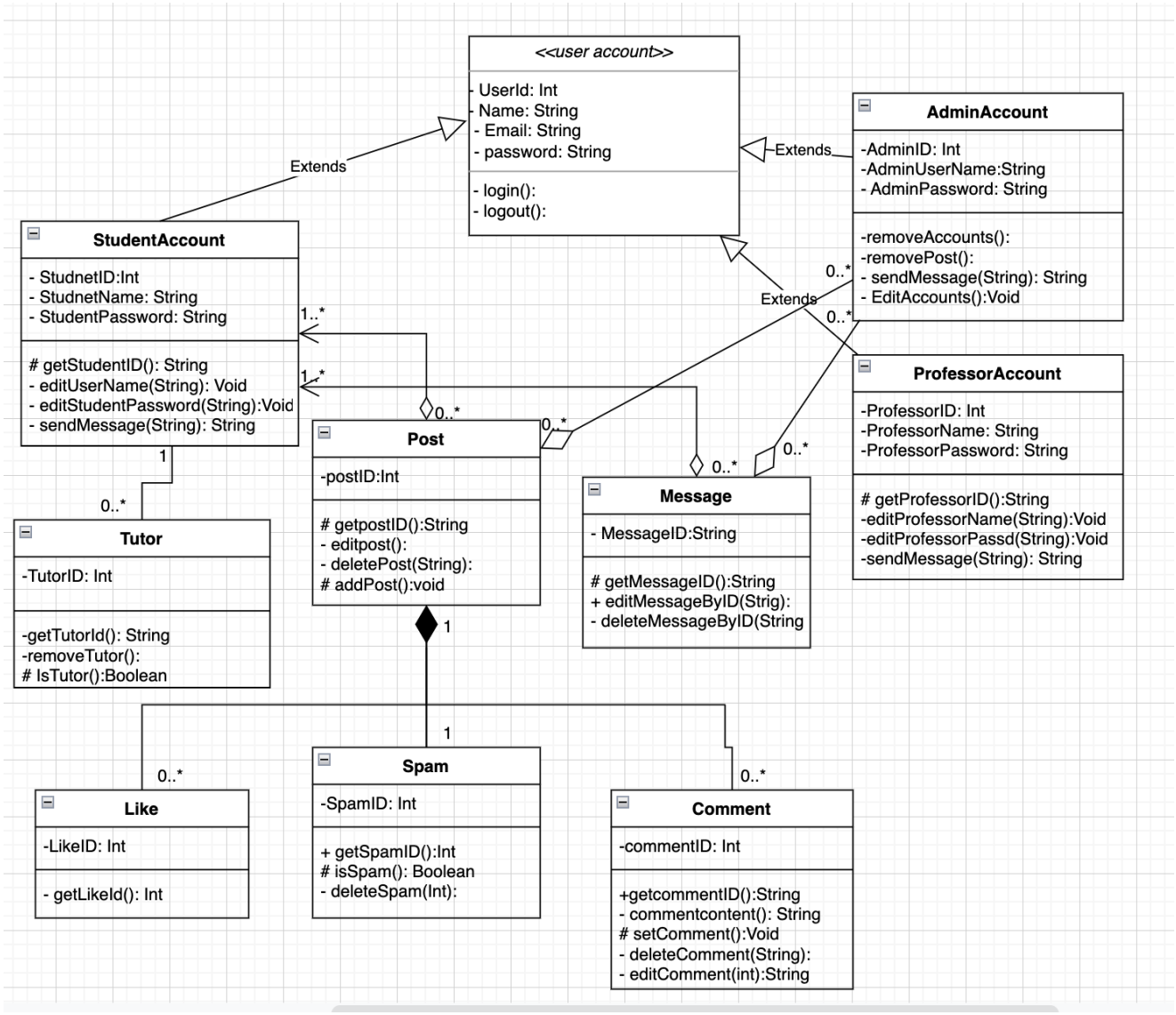


Each Routes Login, Signup, login/{userID}/commet, login/{userID}/message, login/{userID}/post and login/{userID}/search will have POST, DELTE and UPDATE Methods.

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.

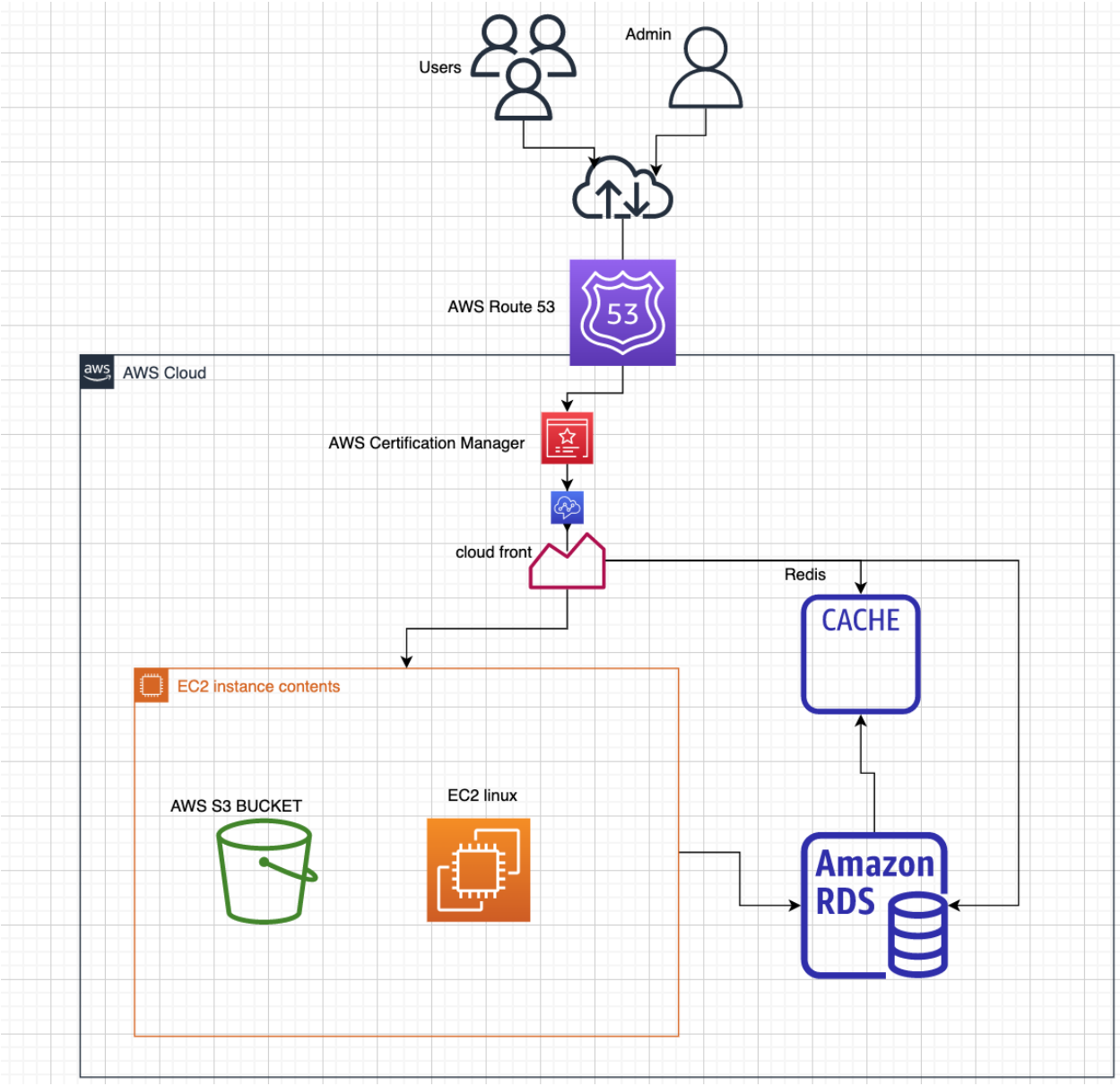
We chose Messaging as our superior feature. To implement this feature, STOMP and SockJS are used. STOMP envelops payloads and adds a header to the communication between the client and the server. SockJS will create low latency, full-duplex, and cross-domain communication channels between the client and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

High Level UML Diagrams

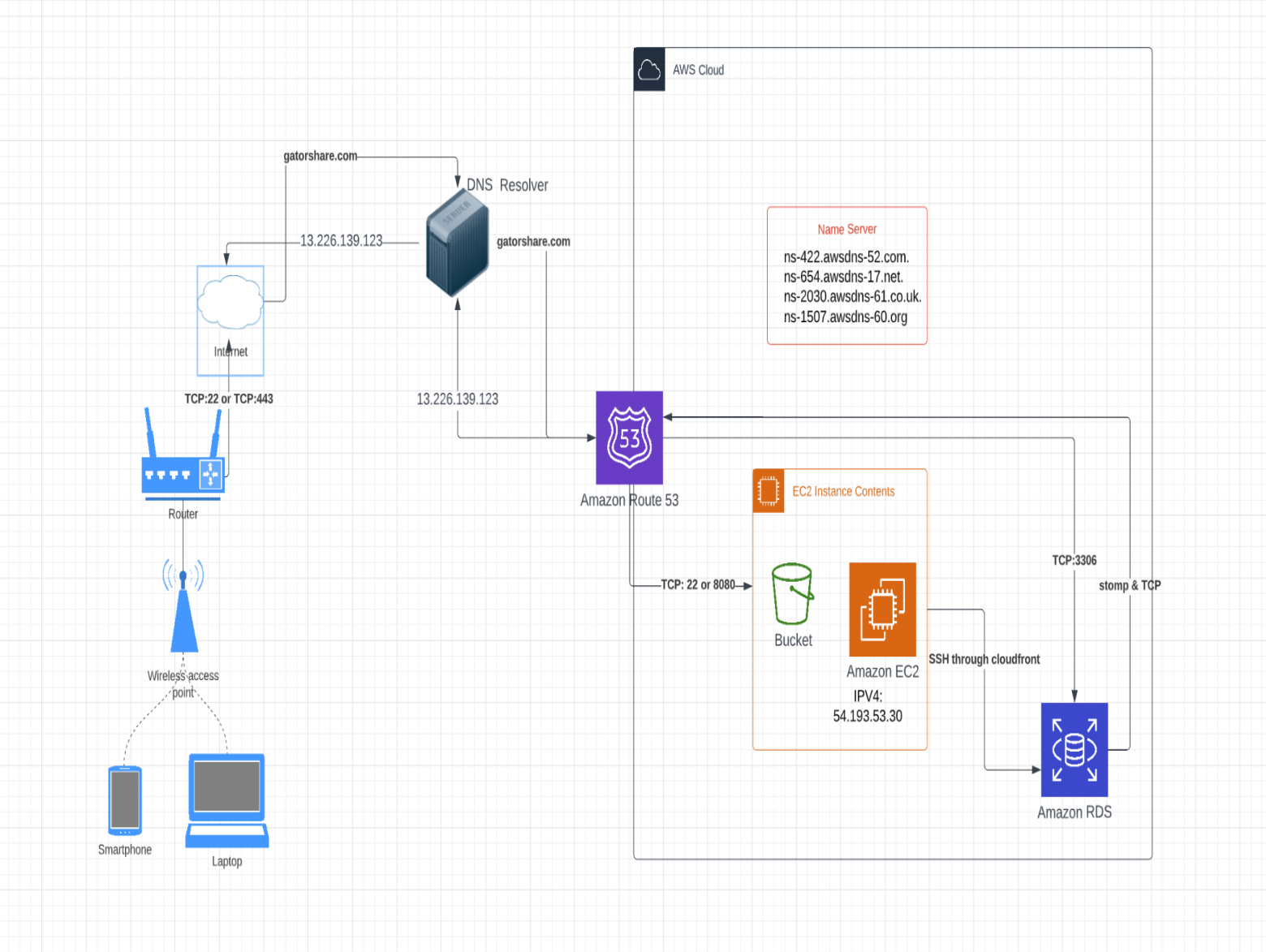


High Level Application Network and Deployment Diagrams

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.

Horizontal Prototype Feedback

Team 01: 0

1. This is a really simple UI and it is not attractive enough to let user stay at your website for 5 seconds. -2
2. It did not look professional to me. Even a simple google sites tool can do better than your website.
3. Not have any function yet. Even log in page and sign up page. Your group is way behind other groups.
4. I cannot update your database with webpage. -1

Milestone 03

Table of Contents

Data Definitions V3

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. When browsing, 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 does not have a personal profile. 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 created an account and has signed in. The User needs to register with the specifically required information, notably their name and email address. A registered user can be a Student, Tutor or Professor. 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 create posts, delete posts, comment on posts, receive notifications, and like/unlike posts (one button). This User has access to a personal profile where they can view all their created posts, and delete a post.

- **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 image, topic category, and an optional media/link submission. All posts are required to have a title text describing what the post contains, although a description and image 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 image and description. A comment would be text only. The username, the text content, and the time it was submitted are all included in each comment.

- **Message:**

A Registered User can message another Registered User. A textbox must not be empty. A message body will be text-only. When a message is sent, the other registered user will receive a notification.

Functional Requirements V3

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 tab that redirects to the Home page.
6. Navigation Bar shall have a tab that redirects to the Login page.
7. Navigation Bar shall have a tab that redirects to the Sign Up page.
8. Navigation Bar shall have a tab that redirects to the Posts page.
9. Navigation Bar shall have a Notifications tab.
10. Navigation Bar shall have a tab that redirects to the Create a Post page.
    - Create a Post page shall be visible only for Registered Users.
11. Navigation Bar shall have a tab that redirects to the Messages page.
    - Messages page shall be visible only for Registered Users.
12. Navigation Bar shall have a tab that redirects to the Profile Page.
    - Profile page shall be visible only for Registered Users.
13. Home page shall have a “Most Recent Posts” section.
14. Home page shall have a “Most Popular Posts” section.
15. Home page shall have six tabs, organized by content:

- “Articles and Essays”, “Art & Film”. “Clubs”, “Discords”, “Tutoring” and “Other”.

16. Footer shall have an About Us link.
17. Footer shall have the Terms & Conditions link.
18. Footer shall have the address of San Francisco State University.
19. Posts shall be public for all Users.
20. Comments shall be public for all Users.
21. Images shall be accessible for all Users.

Requirements for Unregistered User:

1. Unregistered Users shall be able to view all Posts.
2. Unregistered Users shall be able to view all Comments.
3. Unregistered Users shall be able to view all Images.
4. Unregistered Users shall be able to utilize the Search bar.

Requirement for Registered Users:

22. Registered Users shall be able to add a Post onto the platform.
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 a 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 like a Post.
29. Registered Users shall be able to delete their Post.

Requirements for Posts:

30. Posts shall have an image submission field.
    - Posts shall have a default image if no image is uploaded.
31. Images shall only be accepted in jpg, png, or jpeg format.
32. Posts shall display the Registered User who created the post.
33. Posts shall include a comment box.
34. Posts shall include a “Leave a Comment” submission button.
35. Posts shall display the time of the submission.
36. Posts shall include a like button.
37. Posts shall display the number of likes it has received.

Requirements for Comments:

38. Comments shall display the username of the Registered User who created the comment.
39. Comments shall display the time of the comment submission.
40. Comments shall be text only.

Requirements for Admins:

41. Admins shall be a Registered User.
42. Admins shall be able to Post.
43. Admins shall be able to Comment.
44. Admins shall be able to send messages.
45. Admins shall be able to receive messages.
46. Admins shall be able to like a Post.

47. Admins Users shall be able to receive notifications from Registered Users.

Requirements for Professors:

48. Professors shall be a Registered User.
49. Professors shall be able to receive notifications from other Registered Users.
50. Professors shall have a Profile.
51. Professors shall be able to Comment.
52. Professors shall be able to send messages.
53. Professors shall be able to receive messages.
54. Professors shall be able to like a Comment.
55. Professors shall be able to undo the like on a Comment.

Requirements for Tutors:

56. Tutors shall be a Registered User.
57. Tutors shall be able to receive notifications from other Registered Users.
58. Tutors shall have a Profile.
59. Tutors shall be able to Post.
60. Tutors shall be able to delete their Post.
61. Tutors shall be able to Comment.
62. Tutors shall be able to send messages.
63. Tutors shall be able to receive messages.
64. Tutors shall be able to like a Post.

Priority 2 - Desired:

Desired Requirements for Registered Users:

1. Registered Users must register using their San Francisco State University email.
2. Registered Users shall be able to upload a Profile Picture.
3. Registered Users shall be able to like a Comment.
4. Registered Users shall be able to undo a like on a Comment.
5. Registered Users shall be able to undo a like on a Post.
6. Registered Users shall be able to edit their Comments.
7. Registered Users shall be able to delete their Comments.
8. Registered Users shall be able to ‘Flag’ a post.
9. Registered Users shall be able to ‘Flag’ a comment.
10. Registered Users shall be able to edit messages.
11. Registered Users shall be able to delete messages.
12. Registered Users shall be able to edit their Post.

Desired Requirements for Admin:

13. Admin shall be able to review ‘Flagged’ posts.
14. Admin shall be able to review ‘Flagged’ comments.
15. Admin shall be able to upload a Profile Picture.
16. Admins shall be able to edit their Post.
17. Admins shall be able to delete their Post.
18. Admins shall be able to undo a like on a Post.

19. Admins shall be able to edit their comments.
20. Admins shall be able to delete their comments.
21. Admin shall be able to edit messages.
22. Admin shall be able to delete messages.
23. Admins shall be able to like a Comment.
24. Admins shall be able to undo a like on a Comment.
25. Admins shall be able to delete other Users.
26. Admins shall be able to delete other Posts.
27. Admins shall be able to delete other Comments.
28. Admins users shall be able to post to and edit the “About Me” section of the web application.

Desired Requirements for Tutor:

29. Tutors shall be able to upload a Profile Picture.
30. Tutors shall be able to edit their Post.
31. Tutors shall be able to undo the like on a Post.
32. Tutors shall be able to edit messages.
33. Tutors shall be able to delete messages.
34. Tutors shall be able to edit their comments.
35. Tutors shall be able to delete their comments.
36. Tutors shall be able to like a Comment.
37. Tutors shall be able to undo the like on a Comment.

Desired Requirements for Professor:

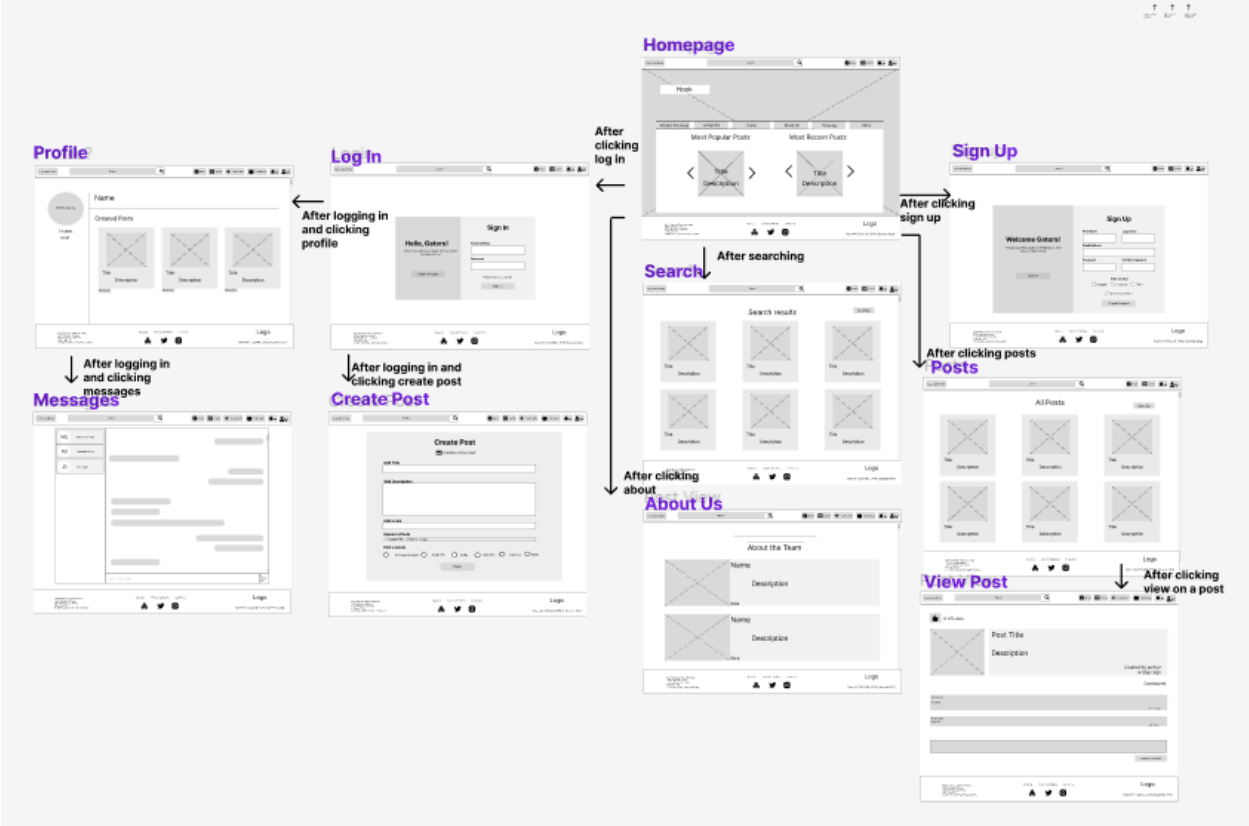
38. Professors shall be able to upload a Profile Picture.

39. Professors shall be able to edit messages.
40. Professors shall be able to delete messages.

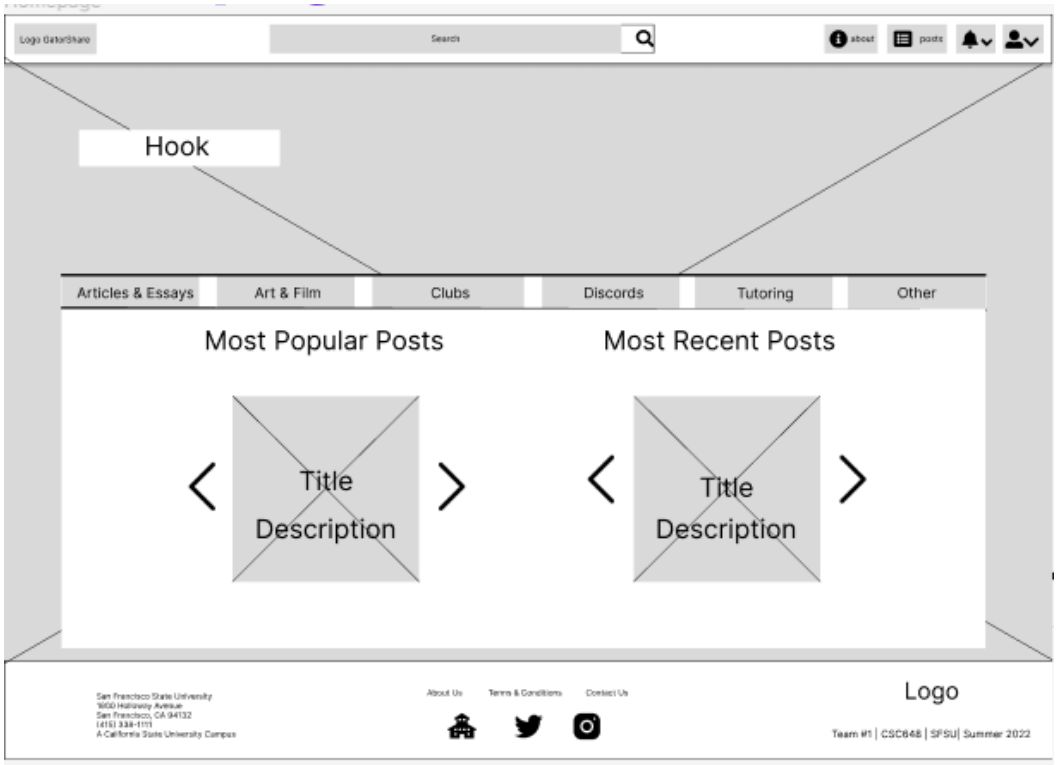
Priority 3 - Opportunistic:

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 receive notifications from Administrators if their Posts or Comments have been removed.
5. Admins shall be able to review ‘Reported’ posts.
6. Admins shall be able to review ‘Reported’ comments.

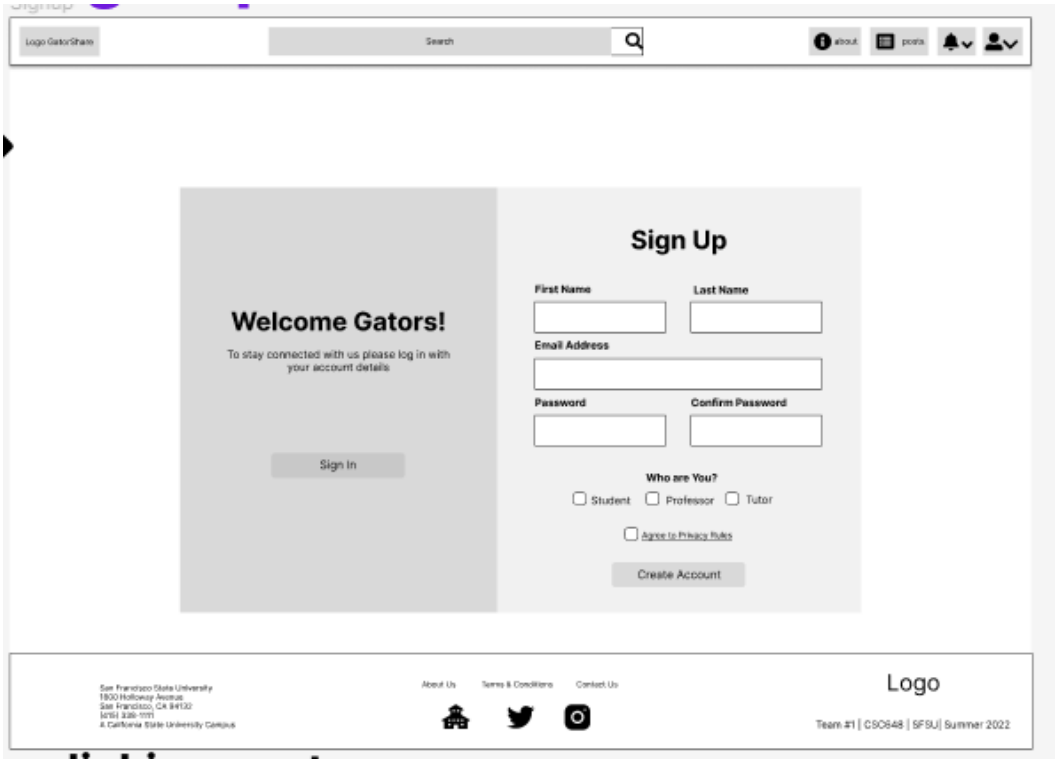
Wireframes based on Mockups/Storyboards



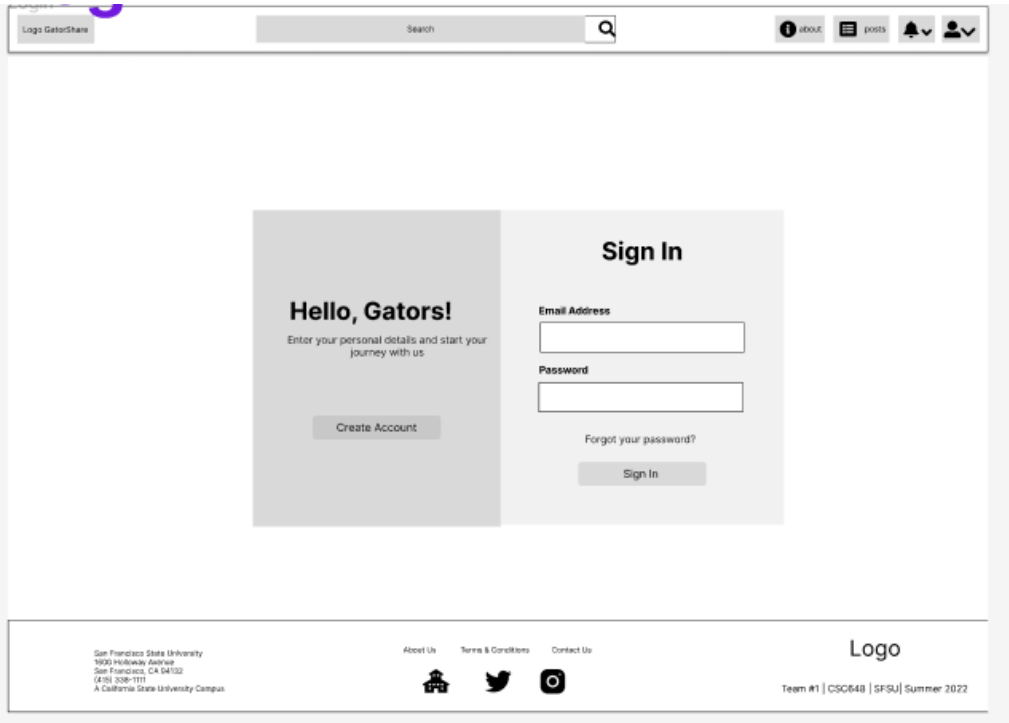
Homepage:



Sign Up page: After clicking "Sign Up" from the navbar tab or clicking "Create Account" from the Sign In page

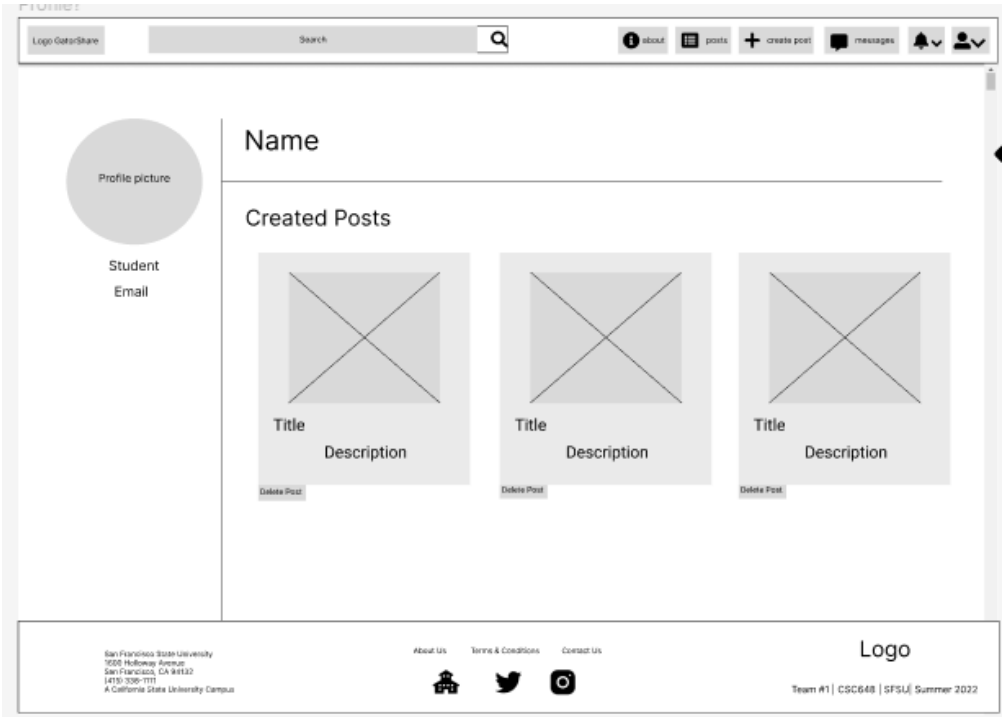


Sign In: After clicking “Log In” on the navbar tab, or “Sign In” from the Sign Up page

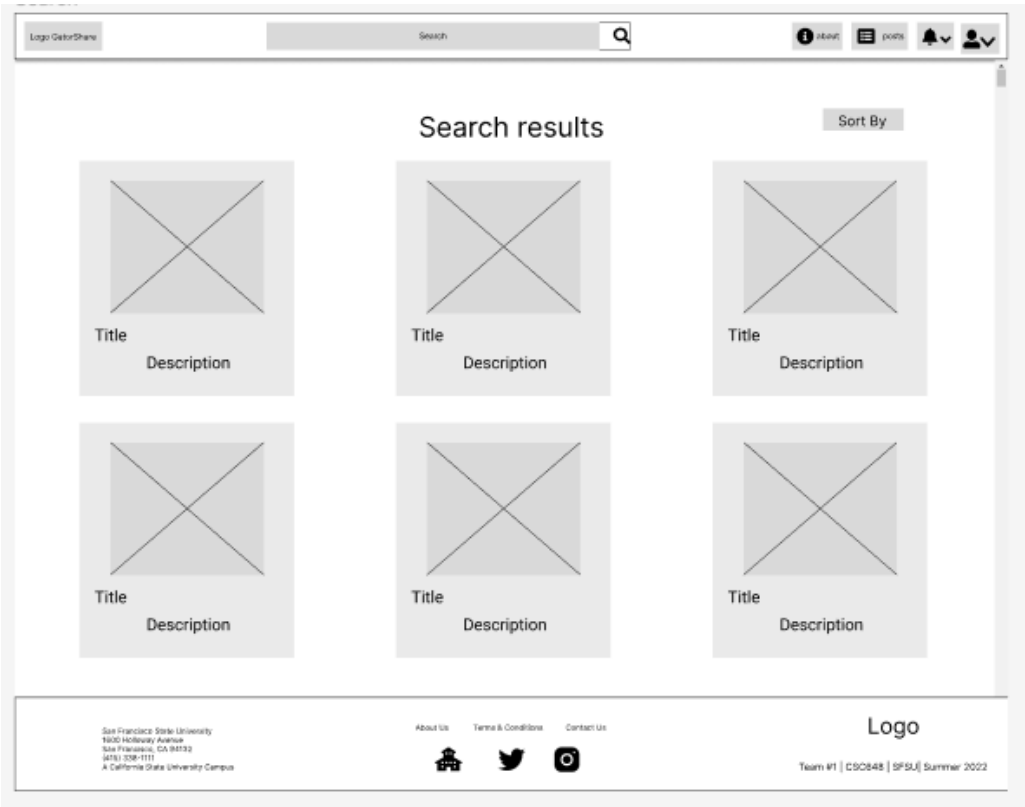


Create Post: After logging in, click the “+ Create Post” tab in the navbar

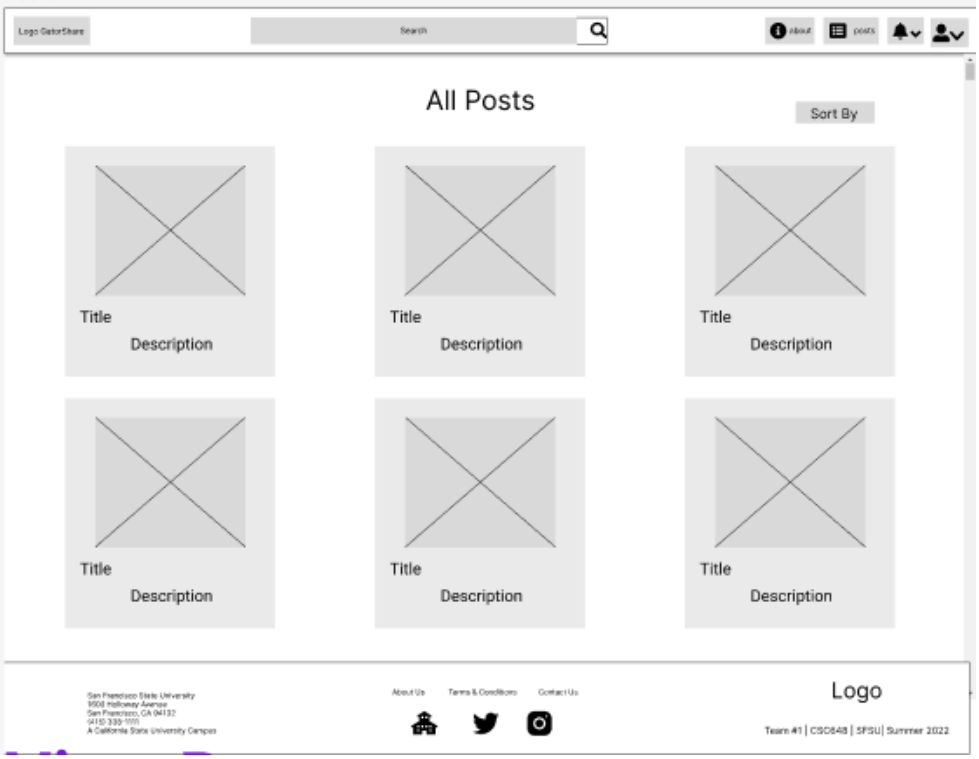
Profile: After logging in, and clicking “Profile” in the navbar



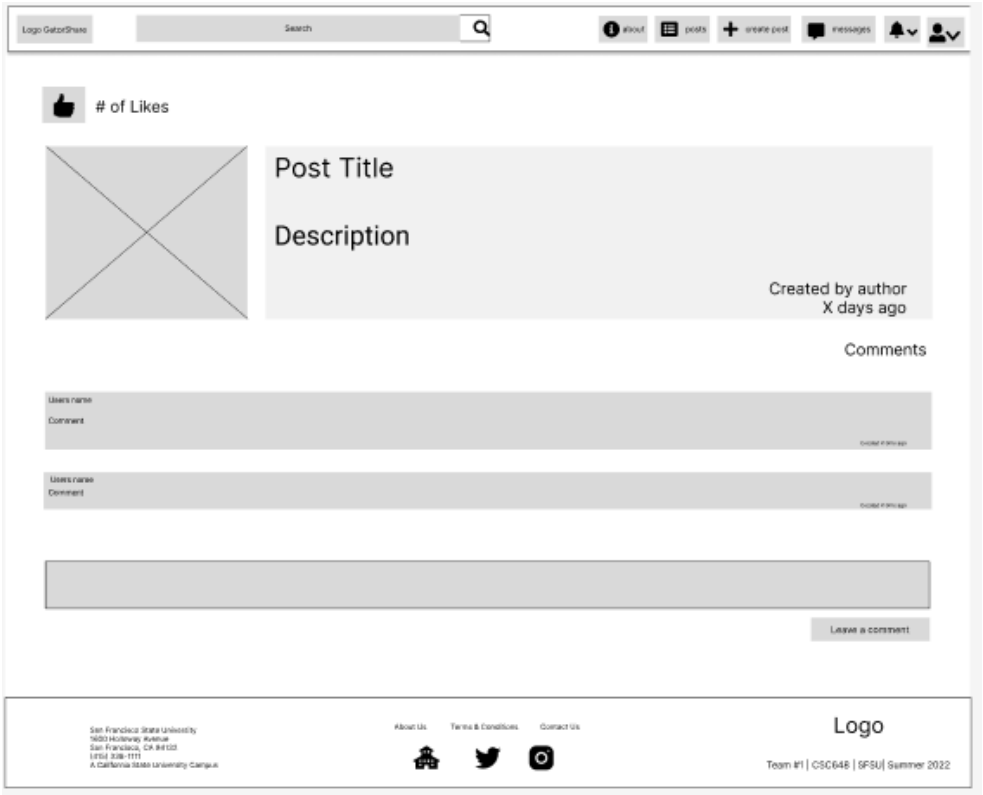
Search: After using the search bar in the navbar



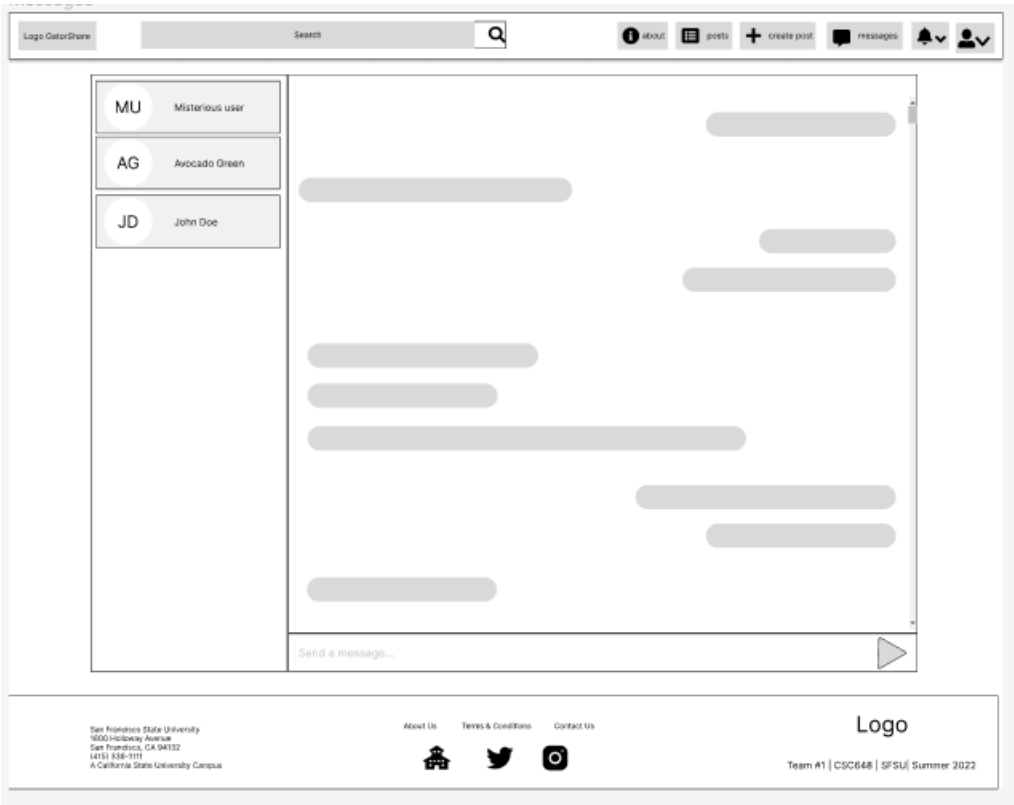
Posts: After clicking “Posts” in the navbar



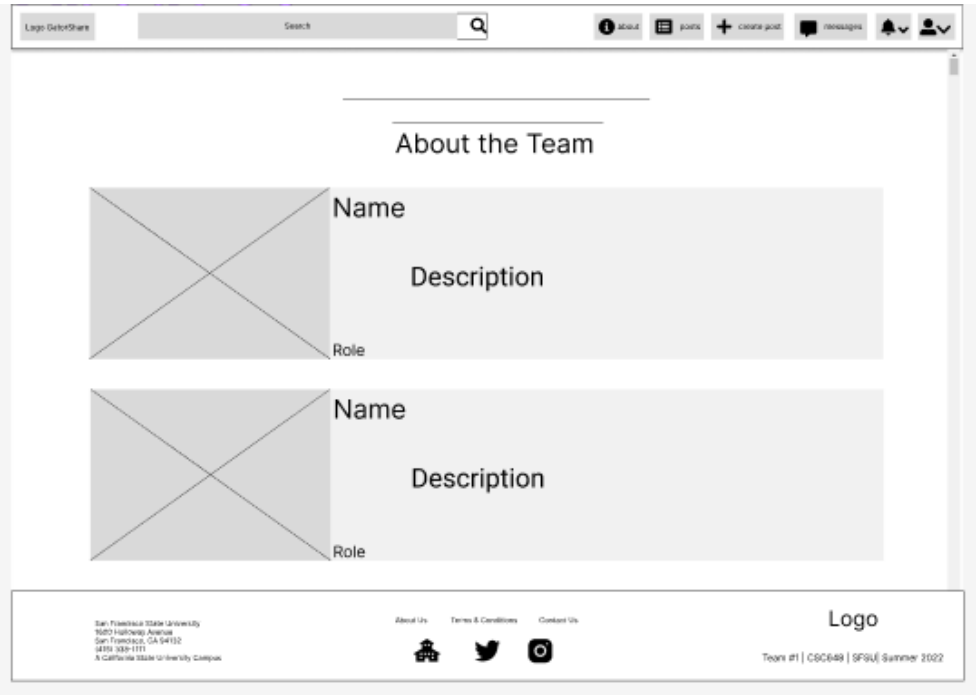
Post View: After clicking a post to view from the Homepage, Posts, Search Results, or Profile



Messages: After logging in and clicking “Messages” in the navbar

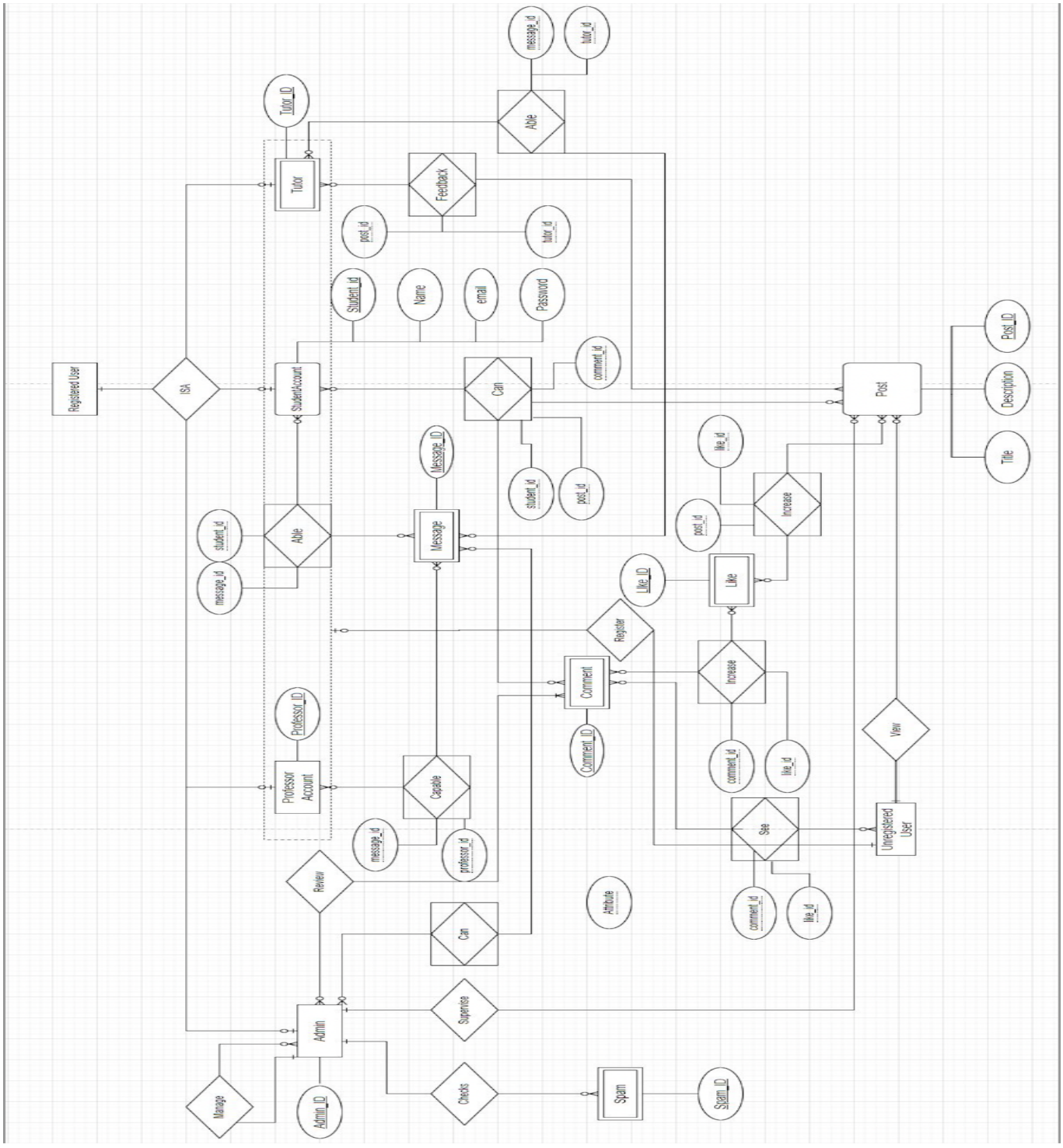


About Us: After clicking “About Us” in the footer

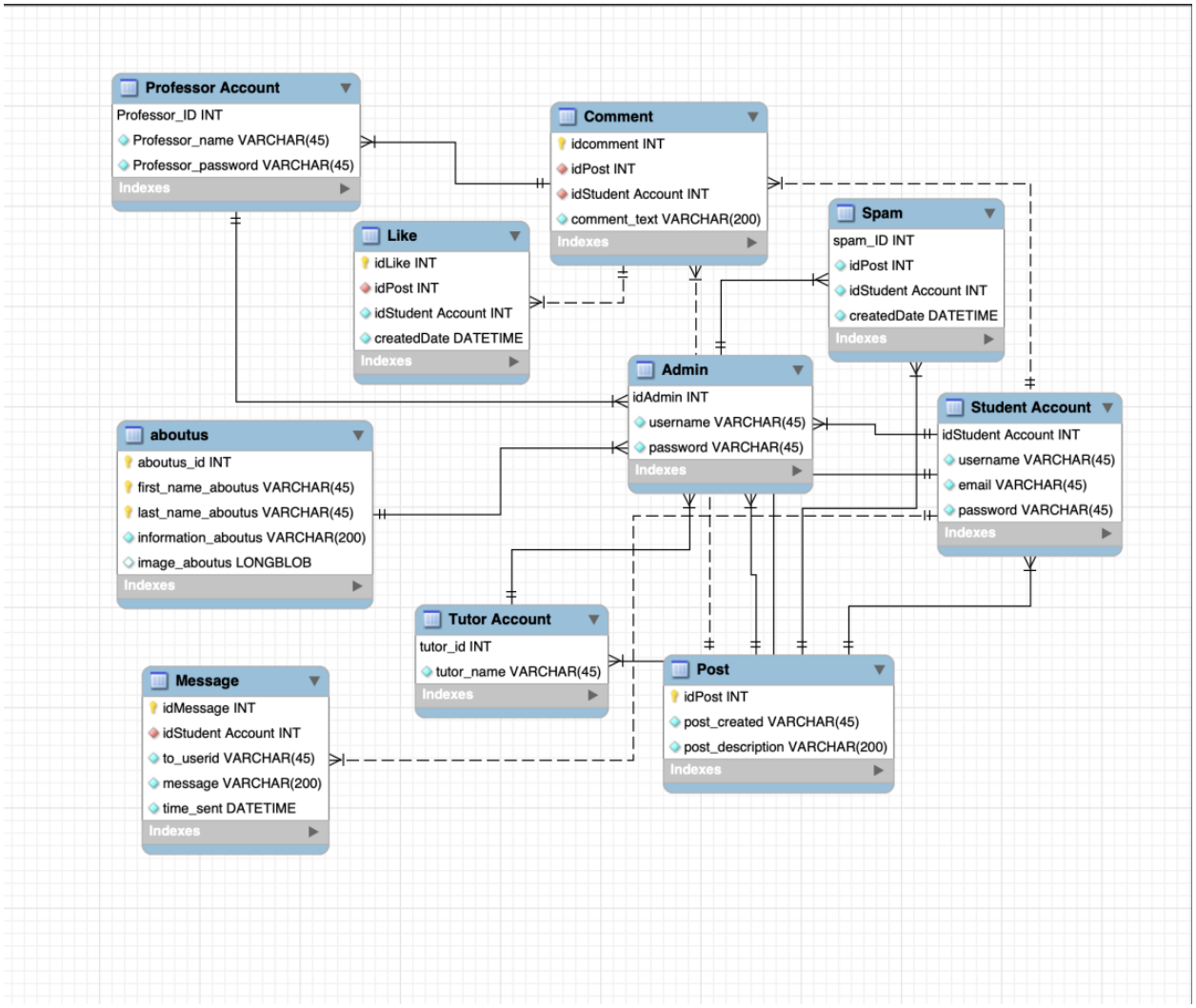


High Level Database Architecture and Organization V2

Entity Relationship Diagram:

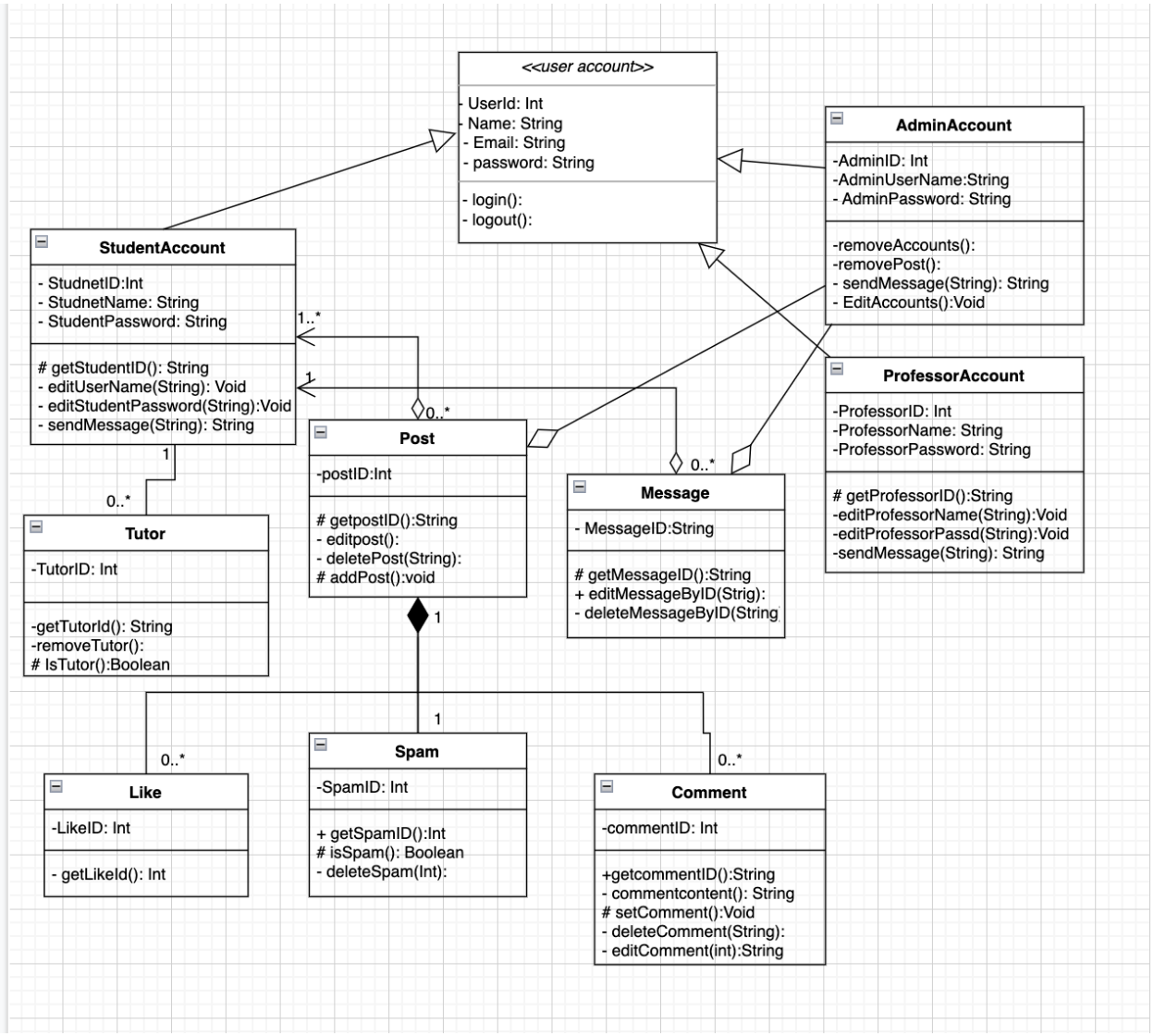


Database Model:

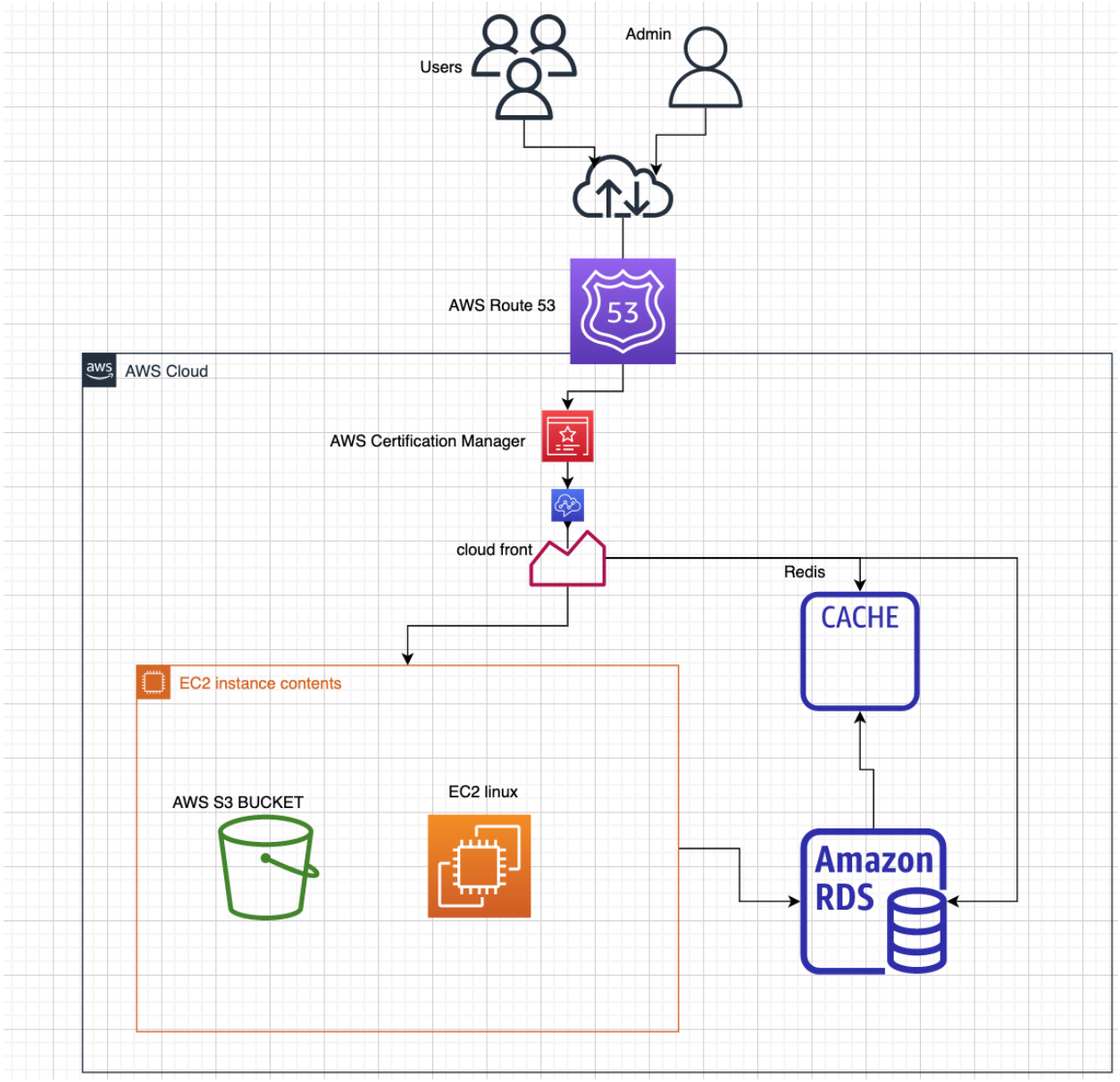


High Level Diagrams V2

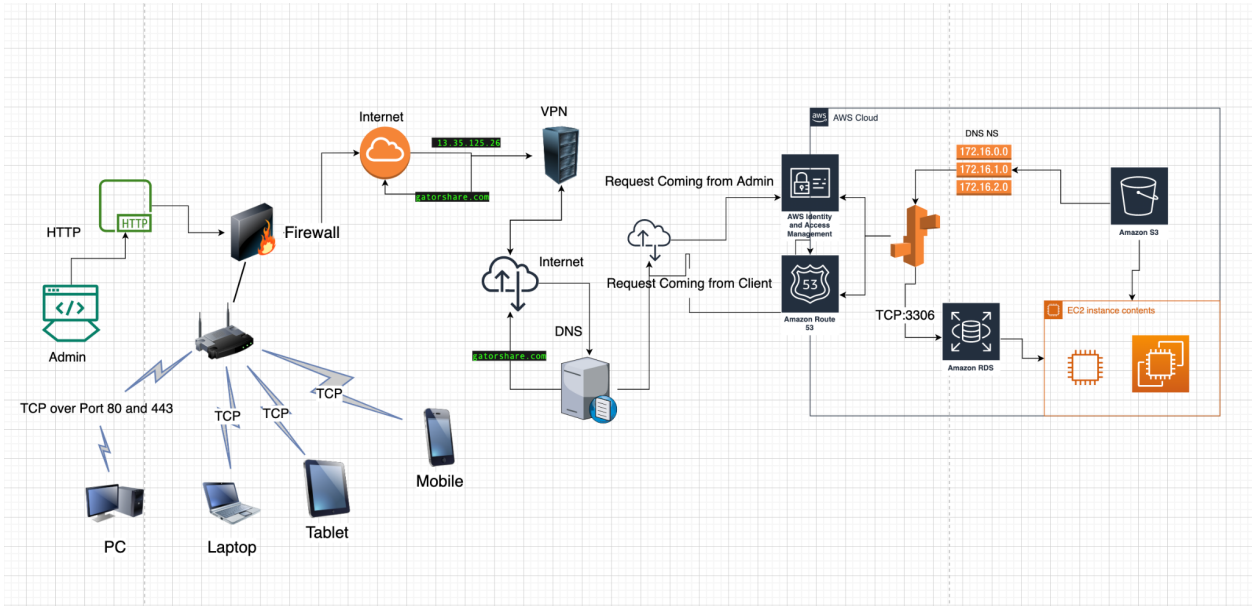
UML Diagram:



Deployment Diagram:



Application Network Diagram:



List of Contributions

Estefanos (Back End Lead) -

Contributed in discussions on Discord. Attended scheduled team meetings. Researched, revised and finalized high level API for Milestone 2 revision. Revised and finalized the UML diagram for both Milestone 2 revision and Milestone 3 documentation. Revised and finalized the High Level Application Network and Deployment Diagrams. Helped work on and revised High Level Database Architecture and Organization. Worked alongside the front-end team to integrate several functionalities such as creating a working Search bar, saving the Sign Up and Log In data, saving the Create Post data, and much more. Consistently tested alongside the front-end team to make sure the application is deployed correctly and ready to be used.

Brianna (Front End Lead) -

Contributed in discussions on Discord. Attended scheduled team meetings. Made the effort to ensure everyone was aware of deadlines. Collaborated with the front-end team on the Horizontal Prototype. Helped check-in with team mates about their progress with specific tasks. Contributed to the creation of UI/UX wireframes. Helped revise Use Case mockups in Milestone 2. Helped code the Log In, Sign Up, Home and Create Post UI/UX. Created and finalized the Wireframe diagram. Added all screenshots and data related to wireframes into the Milestone 3 documentation. Helped revise all front-end portions of Milestone 2's revision document. Designated Milestone 3 editor. Helped revise and finalize Milestone 3 documentation.

Brian (Database Master) -

Contributed in discussions on Discord. Attended scheduled team meetings. Helped revise Main Data Entities and Functional Requirements. Revised and finalized High Level Database Architecture and Organization for Milestone 2 revision. Revised and finalized the Entity Relationship Diagram for Milestone 2 revision and Milestone 3 documentation. Revised and Finalized the Enhanced Entity Relationship diagram for both Milestone 2 revision and Milestone 3 documentation. Designated Scribe for the Horizontal Prototype meeting. Wrote notes and collected feedback for the team.

Mohamed (GitHub Master) -

Contributed in discussions on Discord. Attended scheduled team meetings. Created, revised, and finalized the application's Terms and Agreements. Helped design the applications Footer. Helped revise Main Data Entities and Functional Requirements. Helped brainstorm the Entity Relationship Diagram design. Helped brainstorm with the front-end team about specific page designs. Helped revise the Competitive Analysis for Milestone 2 revision. Helped revise the Non-Functional Requirements for Milestone 2 revision. Pushed M1V2 and M2V2 into the team's Github Repo.

Aleksandr (Front End) -

Contributed in discussions on Discord. Attended scheduled team meetings. Spent countless hours working with the front-end team and back-end lead to have a working, user-friendly, and

appealing Horizontal Prototype. Helped alongside the front-end lead to create UI/UX wireframes. Coded most of the Horizontal prototype. Helped revise and finalize the Entity Relationship Diagram. Helped revise and implement the “Agreement” portion for the website Terms and Agreements. Helped review and update Functional Requirements and Wireframes portion of the Milestone 3 document.

Horizontal Prototype Feedback

Post-

- If he clicks on a page, it is supposed to be implemented already
- It is confusing when clicking on every tab, it goes back to the posts.
- It is supposed to be something related to the tabs, it cannot be going back to the posts
-

Making account- It took him 10 seconds to find where to make an account. If the user is not familiar with websites, they should be able to find it right away. Don't assume that users can figure it out on their own.

Frontpage:

Example: "let's get it started" for the hook

Sign up Page-

- Password: he does not know if there is supposed to be a number, symbol or what
- He is glad about terms & conditions
- He says everything else about the page is good

Notifications-

- This needs to be implemented

Wireframe

- What was implement does not look the same as the wireframe

Create post:

- Creating a post for the UI is good and easy to follow

Messages

His main concern is supposed to have different pages for the home page when you are clicking on the links. It is not implemented at least for the frontend portion. He will not mark us down but we have to have it implemented by the next class. We are not meeting in class please send him an email and he will again go over this. The front page has to be implemented. Send the prototype again before the next class. He does not want to be seeing the same page over and over. You were supposed to complete the whole front end.

Milestone 04

Table of Contents

Product Summary

Name of the product:

GatorShare

Product Summary:

Our application is a one-stop website designated for San Francisco State University students, faculty, alumni, and the general public. Our goal is to connect people, allow viewers to explore different social events SFSU has to offer, find Discord servers related to SFSU clubs and courses, and find Tutoring services. Our website also allows Users to post their work and become a part of a community. GatorShare is unique in that our objective is to combine everything into one. Just by accessing GatorShare alone, a Guest will be able to find resources, find Discord channels, view event flyers, read articles written by SFSU peers, and also find Tutors that fit their needs. What makes us different from our competitors is that our service allows our Users to communicate with one another all in one session. Registered Users can interact with other SFSU students by utilizing our commenting and messaging features, they can post their original articles and essays, tutors can be found, and most importantly, our Users can form bonds. Our product helps both new and returning students locate the tools they require to feel welcomed and a member of the SFSU community. We provide an atmosphere in which members of the SFSU community may be connected and discover all the tools they need to support their academic success and social fulfillment on and off campus.

Product URL:
http://gatorshare.com

Itemized List: Priority 1

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 tab that redirects to the Home page.
6. Navigation Bar shall have a tab that redirects to the Login page.
7. Navigation Bar shall have a tab that redirects to the Sign Up page.
8. Navigation Bar shall have a tab that redirects to the Posts page.
9. Navigation Bar shall have a tab that redirects to the Create a Post page.
    - Create a Post page shall be visible only for Registered Users.
10. Navigation Bar shall have a tab that redirects to the Messages page.
    - Messages page shall be visible only for Registered Users.
11. Navigation Bar shall have a tab that redirects to the Profile Page.
    - Profile page shall be visible only for Registered Users.
12. Home page shall have a “Most Recent Posts” section.
13. Home page shall have a “Most Popular Posts” section.
14. Home page shall have six tabs, organized by content:
    - “Articles & Essays”, “Art & Film”. “Clubs”, “Discords”, “Tutoring” and “Other”.

15. Footer shall have an About Us link.
16. Footer shall have the Terms & Conditions link.
17. Footer shall have the address of San Francisco State University.
18. Posts shall be public for all Users.
19. Comments shall be public for all Users.
20. Images shall be accessible for all Users.

Requirements for Unregistered User:

1. Unregistered Users shall be able to view all Posts.
2. Unregistered Users shall be able to view all Comments.
3. Unregistered Users shall be able to view all Images.
4. Unregistered Users shall be able to utilize the Search bar.

Requirement for Registered Users:

21. Registered Users shall be able to add a Post onto the platform.
22. Registered Users shall be able to add Comments on another Registered User's Posts.
23. Registered Users shall have a Profile.
24. Registered Users shall be able to send messages.
25. Registered Users shall be able to receive messages.
26. Registered Users shall be able to like a Post.
27. Registered Users shall be able to delete their Post.

Requirements for Posts:

28. Posts shall have an image submission field.
29. Images shall only be accepted in jpg, png, or jpeg format.

30. Posts shall display the Registered User who created the post.
31. Posts shall include a comment box.
32. Posts shall include a “Leave a Comment” submission button.
33. Posts shall display the time of the submission from present time.
34. Posts shall include a like button.
35. Posts shall display the number of likes it has received.

Requirements for Comments:

36. Comments shall display the username of the Registered User who created the comment.
37. Comments shall display the time of the comment submission from present time.
38. Comments shall be text only.

Requirements for Admins:

39. Admins shall be a Registered User.
40. Admins shall be able to Post.
41. Admins shall be able to Comment.
42. Admins shall be able to send messages.
43. Admins shall be able to receive messages.
44. Admins shall be able to like a Post.
45. Admins Users shall be able to receive notifications from Registered Users.

Requirements for Professors:

46. Professors shall be a Registered User.
47. Professors shall be able to receive notifications from other Registered Users.
48. Professors shall have a Profile.

49. Professors shall be able to Comment.
50. Professors shall be able to send messages.
51. Professors shall be able to receive messages.
52. Professors shall be able to like a Comment.
53. Professors shall be able to undo the like on a Comment.

Requirements for Tutors:

54. Tutors shall be a Registered User.
55. Tutors shall be able to receive notifications from other Registered Users.
56. Tutors shall have a Profile.
57. Tutors shall be able to Post.
58. Tutors shall be able to delete their Post.
59. Tutors shall be able to Comment.
60. Tutors shall be able to send messages.
61. Tutors shall be able to receive messages.
62. Tutors shall be able to like a Post.

Usability Test Plan

Purpose -

This test is intended to identify any problems Users may experience while attempting to use our posting feature. This feature includes creating, viewing, searching for, commenting on, and deleting a post.

Problem Statement and Objective -

We have chosen to test these post-related functionalities to ensure that a User is able to navigate around our application effortlessly and efficiently. As we test for creating a post, we want to ensure that a User's post submits correctly and accurately. We want to test to confirm their post is saved into our database which in turn allows their post to be shown on our public feed. Within our public feed, we want to test that posts can be accessed by all Users within our application. When a User searches for a post, we want to guarantee the correct subject is being filtered. Upon clicking on a post, we will be testing to see if each post has a comment box and comments may be posted. Lastly, we want to test if a User can delete their post, and if this deleted post erases from our data system.

Method -

Our method to test our functionalities is to have a new User access our web application. Our User must navigate around our homepage where they can utilize our search bar, click on different tabs, and click on different posts. Our User must also create an account with us in order to test how to create, delete, and comment on a post.

Task list -

We'd want to start by testing our search bar to see if a User can search and find a post. We'd also like to check if a User can click on the post they've searched for. If these two tests are successful, we can confirm that any User can publicly access all postings on our platform. Next, we want to test if a User can create a post, post comments, and delete a post. Because these features are only available to Registered Users, we must create an account in order to test them. If a User can successfully create a post, post a comment, and delete a post, it confirms that all data is being stored in our database, the User can use our platform, and data can be deleted from our database.

Description:

Test Environment/System Set Up -

In order to test our website, our server must first be set up and running. Our database must also be connected to the server in order to store a User's data. When a User first accesses our application, they are able to view and search for a post. These two functionalities are open to both an Unregistered and Registered user. When a User sign's up with our application, their registration data is saved into our database. Upon logging in, a User is now free to create a post, comment, and delete an existing post.

User Profile -

We are testing our application for both an Unregistered User (A User who has not signed up or logged in) and a Registered User (A User who has created an account and has logged in).

Starting Point -

Upon entering our application, our search bar and feed is open to the general public. The User can click on any "Recent Post", "Popular Post", search for a subject in our search bar, and also click any of the category tabs to navigate to different Posts. These features are not limited to any

User. For other components- such as commenting on a post, creating a post, and deleting a post, a User must be registered and logged in. Once logged in, in order to create a post, a User must navigate to the navigation bar and click “+Create a Post”. To comment, a User may click on any submitted Post whether, its by accessing “Recent Posts” or “Popular Posts” from the homepage, opening a category page such as “Discord”, “Tutoring”, “Art & Film”, etc. or searching for a post using our search bar. To delete a post, a user must navigate to their Profile using the navigation bar. From there, a User can delete any of their created posts.

Intended Users -

For students of SFSU to connect with other students, faculty, tutors, alumni, and the general public. Our application is designed to assist SFSU students in communicating with others and finding conveniently available resources that meet their specific needs. They may use our website to acquire tutoring services, locate a discord server to meet new people, join clubs, provide feedback for arts & films, or read articles about SFSU.

URL, What Is Being Measured -

http://gatorshare.com/

Report Content/Evaluation Measures:

Usability Test Table - Measuring Effectiveness -

# of Pages	Test/Use Case	% Completed	Errors	Comments	% of Time to Complete the Task (in seconds)
3	Create a Post	50	Not able to post	You can start the post creation process but nothing happens when you click “Post”	–
	View a Post	100	No error detected	The user can view a post completely with no hassle	06.52
2	Search a Post	90	No error detected	It shows the posts containing the searched words. Though, no recommendation is made yet if it does not find anything.	01.50
3-4	Comment on a Post	100	No error detected	The user can comment on any post after logging in	33.11
3	Delete a Post	0	Not able to test yet	This functionality does not work yet	0

Task Description -

Task	Description
Create a Post	From the homepage, the User must first log in. If they are not registered yet, they must create an account and then log in. Once logged in, the User can click "Create Post"

	+ Create Post from the navbar. Once redirected to the Create a Post form, the User must fill out the various fields for the post and, if wanted, upload any desired images or media. When the User is finished, they must click "Post" POST. Once posted, the User is redirected to "All Posts" where they can see their post amongst all other posts.
View a Post	From the homepage, the User can either click on "Posts" Posts in the navbar to view the "All Posts" page or if the User is interested in a certain category of post, they can click on one of the six categories located under the Homepage's hook.
Search a Post	From the homepage, the User can click on the "search bar" and type in what they are looking for. The User can also narrow their search by clicking the "Category" Category dropdown menu next to the bar and selecting a category. Once the User submits their search, the results will be displayed.
Comment on a Post	While viewing a post, a User can leave a comment for that post by entering their comment in the text box below the post's image and description. To post the comment, the User must click "Leave a comment" LEAVE A COMMENT. Once clicked, the User can then see their comment, along with any others, in the provided comment box.
Delete a Post	From the homepage, the User must first log in. Once logged in, the User can click the "Account" Account dropdown from the navbar. In the dropdown, the User must click "Profile" in order to view all their created posts. Once redirected to their Profile, the User can click "Delete" on any of their posts they wish to delete.

User Satisfaction Table

	Strongly Disagree	**Disagree**	**Neutral**	**Agree**	**Strongly Agree**
Create a Post					
Creating a post is fast and requires little effort			X		
The Create a Post form is organized and easy to understand		X			
My post was successfully created with no errors	X				
View a Post					
It is easy to view a post					X
The view a post page contains a title, description, image and comment box					X
I can view any post on the site					X
Search a Post					
The search bar is visible					X
The search function is easy to use					X

I received search results related to what I searched			X		
Comment on a Post					
The comment box is easy to find					X
It is quick and easy to post a comment	X				
I can see all comments posted on a post	X				
Delete a Post (Not implemented yet)					
I have the ability to delete any of my created posts	X				
It is quick and easy to delete a post	X				
I am provided with a popup to confirm deletion	X				

QA Test Plan

Test Objectives:

We would like to test the following components: Comments must have a text limit of 250 characters, passwords shall be encrypted in the database where it's stored, passwords must have very specific criteria when a User signs up, all data shall be stored in the team's SQL database, and a Registered User's session will end once they click log out. We have chosen these components because we want to make sure comments have a text limit to prevent long and unnecessary potential spam. We also want to make sure passwords are encrypted to guarantee a User's password is safe and secure. We require passwords to meet certain criteria to make sure their passwords are not easily guessed. This will prevent hacking measures. Lastly, we want to test if a User's data is stored into our team's database to make sure all posts and comments are properly being collected and saved.

HW and SW setup (including URL):

http://gatorshare.com/

- **Setup of HW:**

In order to have our application accessible, the live version of our website must be deployed to our Cloud services, then a User can access our website from any device of the User's choice. In order to test our website, a User must have a computer, smartphone, or any device that has connection to the internet.

- **Setup of SW:**

We have chosen to deploy our website using Amazon Web Services (AWS) where we have connected our MySQL database to the cloud. A User must have a working version of any web

browser in order to access and test our live application. We have tested our application on both a Mac and Windows device. We have also tested our application on the following web browsers: Google Chrome, Mozilla Firefox, and Safari.

Test Environment, Human Resources Needs, Expected Time to Complete:

In order to test our objectives, we need to access our database. Some of the physical resources we require include having our Database Master guarantee that our database is operational and that all passwords are encrypted for privacy and security.

Risks, Contingencies:

One danger we may face is if our database fails. A User's data cannot be stored if our database fails for any reason, such as when they try to sign up or create a post. Another danger we may face is a large number of spam and/or 'bots' posting on our application. We may need to evaluate how to approach such an issue, such as screening what is spam or how to delete a large amount of spam in one session.

Actual Test Sequence, Test Input and Test Expected Results:

In order for our tests to be successful, our Tester must follow the instructions clearly. In order to create a password that fulfills our requirements, our Tester must have a password that is 8-characters long, must have one uppercase letter and one lower case letter, and must contain a number. To make sure our Tester’s password is encrypted, we have hashed their passwords in order to ensure data integrity.

Feature to Be Tested:

We would like to test the following features: Creating a password that fits our requirements, verifying that passwords are encrypted, and posting a comment with a text limit.

QA Test Plan:

Test Plan #1: Entering a Valid Password

- **Input:** Enter “Abcd123!” as a password when creating an account
- **Output:** Password requirements should turn green. This will indicate that the password created is considered valid.

Test Plan #2: Saving an Encrypted Password

- **Input:** Enter any passwords that fulfills the password requirements. (UserTest1234!)
- **Output:** Password is considered valid and the user’s information is now stored in our database system. The User’s password is encrypted with a series of letters and numbers.

Test Plan #3: Commenting more than 250 Characters

- **Input:** Type or copy and paste any paragraph that is 250 characters or more.
- **Output:** Comment will still submit, regardless of length.

QA Test Plan Table:

Test #	Test Title	Test Description	Test Input	Expected Correct Output	Test Results (Pass/Fail)
1	Create a password that fulfills requirements	Type “Abcd123!”	Abcd123!	Abcd123! All password requirements should turn green, thus confirming that the password is valid	Pass
2	Password is	Type any	UserTest12	Password is in the	Pass

	encrypted in database	password (that fulfills requirements)	34!	database with a series of scrambled letters and numbers	
3	Posting a comment more than 250 characters	Type or copy and paste any paragraph	Text that has 250 characters or more	Comment will still submit, regardless of length. Our goal is to have a pop up with a text limit error, or a limit that stops the user from continuing to type.	Fail

Code Review

Coding Style:

Our primary goal is to have a code that is clear, organized, well-documented, and clutter-free. PascalCase is used for the names of React UI components (e.g. UserProfile.js). For any other assistance files, CamelCase is utilized (e.g. userAction.js). CSS file names are written in kebab-case (e.g. user-action-button.js). The React naming conventions must be followed for components containing React classes (e.g. className instead of class, marginBottom instead of margin-bottom). Make multiple tiny files rather than one big one. All CSS files are in a single, shared folder (like "./CSS"). Avoided !important and inline css as much as possible. We utilized a prettier plugin to make the code easier to understand and review. Files are named sensibly based on the function they carry out. Organized imports in order by React import, Library imports (Alphabetical order), Absolute imports from the project (Alphabetical order), Relative imports (Alphabetical order), Import * as.

Code Review Feedback from members of Team 1 and Team 3:

Team 1 Feedback:

- Short description about each part of the code makes it very direct and straight-forward.
- The white space keeps each part very organized.
- export const login and export const logout are not inline. One is tabbed and one isn't, consider aligning these two.
- Good use of naming conventions.
- Code is very organized and easy to read.

- Handling search results is easy to understand.
- There are alerts when errors occur which is good to be able to keep track
- Good indentation which made it easy to follow along with the code
- Remove the //Todo once the section is done
- The readability for both files are proficient and clear

Team 3 Feedback:

- While you log errors to the console, also make sure to render an error page for the user.
- Good consistent code for what is given
- A bit too simple and brief and it only focuses on a basic functionality that is ubiquitous amongst a lot of web applications(basic search and authentication are common functionalities).
- Maybe use better url parameters as opposed to Articles&Essays and Art&Films? I think the ampersand is generally a safe character to use, so it's not a significant issue. But I personally think its better to follow existing uri naming standards.
- The code here is good code, it is organized well and optimized for readability. It is modular enough to be easily inserted into other functionalities.
- There are good comments that make it easier for a non-js developer to read and understand.
- Variable names follow styling guidelines and are pragmatic.

Code for Search and User Session -> Auth.js & SearchResults.js

auth.js M

application > client > src > js > auth.js > login > then() callback

```
/*************************************************************
 * Class:  CSC-648 Summer 2022
 * Author: Aleksandr Gusev, Brianna Soukup
 * Project: Gatorshare website
 * File: auth.js
 * Description: this file includes all functions required for authentification
 *************************************************************/
import { ReactSession } from 'react-client-session';

import http from '../http-common';
import { alert } from './alert';

//Review code #1 -> Non functional requirement: The Registered Users session shall end once they logout

// login function handles data for the Login page.
export const login = (formData) => {
    // the data for the axios request is stored here
    const { email, password } = formData;
    let id;

    // post request to the backend
    http.post('/login', {
        email: email,
        password: password,
    })
        .then((response) => {
            if (response.data.token) {
                const token = response.data.token;
                // user's id
                id = response.data.id;

                // store token and logged in user id
                ReactSession.set('token', token);
                ReactSession.set('currentUserId', id);
                // append success alert message
                alert('success', 'successfully logged in');

                // post request to get user's role, e.g. 'Student'
                http.get('/login/id/' + id)
                    .then((response) => {
                        ReactSession.set(
                            'userRole',
                            response.data.role_name[0].role_name
```

```
//Review code #2 -> Non functional requirement: The Registered Users session shall end once they logout

// logout function handles signing out
export const logout = () => {
    // get current user's token
    const token = ReactSession.get('token');

    // if user logged in then clear the token
    if (token) {

        alert('warning', 'successfully logged out');

        localStorage.clear();
    } else {
        // if can't log out, append and error message
        alert('danger', 'error logging out');
        console.log('token error');
    }
};
```

```
/*************************************************************
 * Class:  CSC-648 Summer 2022
 * Author: Aleksandr Gusev, Brianna Soukup
 * Project: Gatorshare website
 * File: SearchResults.js
 * Description: this file includes all functions and components
 * required for search functionality
 *************************************************************/
import React, { useEffect, useState } from 'react';
import { useNavigate } from 'react-router-dom';
import http from '../../http-common';
import noImage from '../../img/noImage.jpeg';
import Spinner from '../misc/Spinner';
import moment from 'moment';
import { ReactSession } from 'react-client-session';

// Review code #3 -> Searching

// the main component that handles search results
const SearchResults = () => {
    // flag to indicate when results are fetched
    const [isLoaded, setIsLoaded] = useState(false);
    const [posts, setPosts] = useState([]);
    // redirects to another component within react component
    const navigate = useNavigate();
    // what we search for
    let searchTerm = ReactSession.get('searchTerm');
    // get the searched category
    let category = ReactSession.get('category');

    // define the endpoint for the search depending on the category
    let searchURL = `/search?query=${searchTerm}`;
    if (category == 'Articles&Essays') {
        searchURL = `/search/{Article}?query=${searchTerm}`;
    } else if (category == 'Art&Films') {
        searchURL = `/search/{ArtAndFilm}?query=${searchTerm}`;
    } else if (category == 'Clubs') {
        searchURL = `/search/{Clubs}?query=${searchTerm}`;
    } else if (category == 'Discords') {
        searchURL = `/search/{Discord}?query=${searchTerm}`;
    } else if (category == 'Other') {
        searchURL = `/search/{Other}?query=${searchTerm}`;
    } else if (category == 'Tutoring') {
        searchURL = `/search/{Tutoring}?query=${searchTerm}`;
    }

    // set pager header to indicate category
    if (category) {
        if (category === '') {
            document.getElementById('search-button-1').innerHTML = 'Category';
        } else {
            document.getElementById('search-button-1').value = category;
            document.getElementById('search-button-1').innerHTML = category;
        }
    }
```

```
// Review code #4 -> Searching


// the search request.
useEffect(() => {
    http.get(searchURL)
        .then((res) => {
            console.log(searchURL);
            // set results when fetched
            setPosts(res.data);
            // indicate that the page can be loaded
            setIsLoaded(true);
        })
        // handle errors
        .catch((e) => {
            setIsLoaded(false);
            console.log(e);
        });
}, []);
```

Why this code:

The code in auth.js and SearchResults.js was chosen for code review as it relates to the search feature tested in the Usability Test and the non functional requirement for User sessions in the QA Test Plan. These two files effectively show how the team uses proper naming conventions, maintains clean, well-organized code, and provides helpful comments and headers.

Self-Check on Best Practices for Security

Major Assets that are Protected:

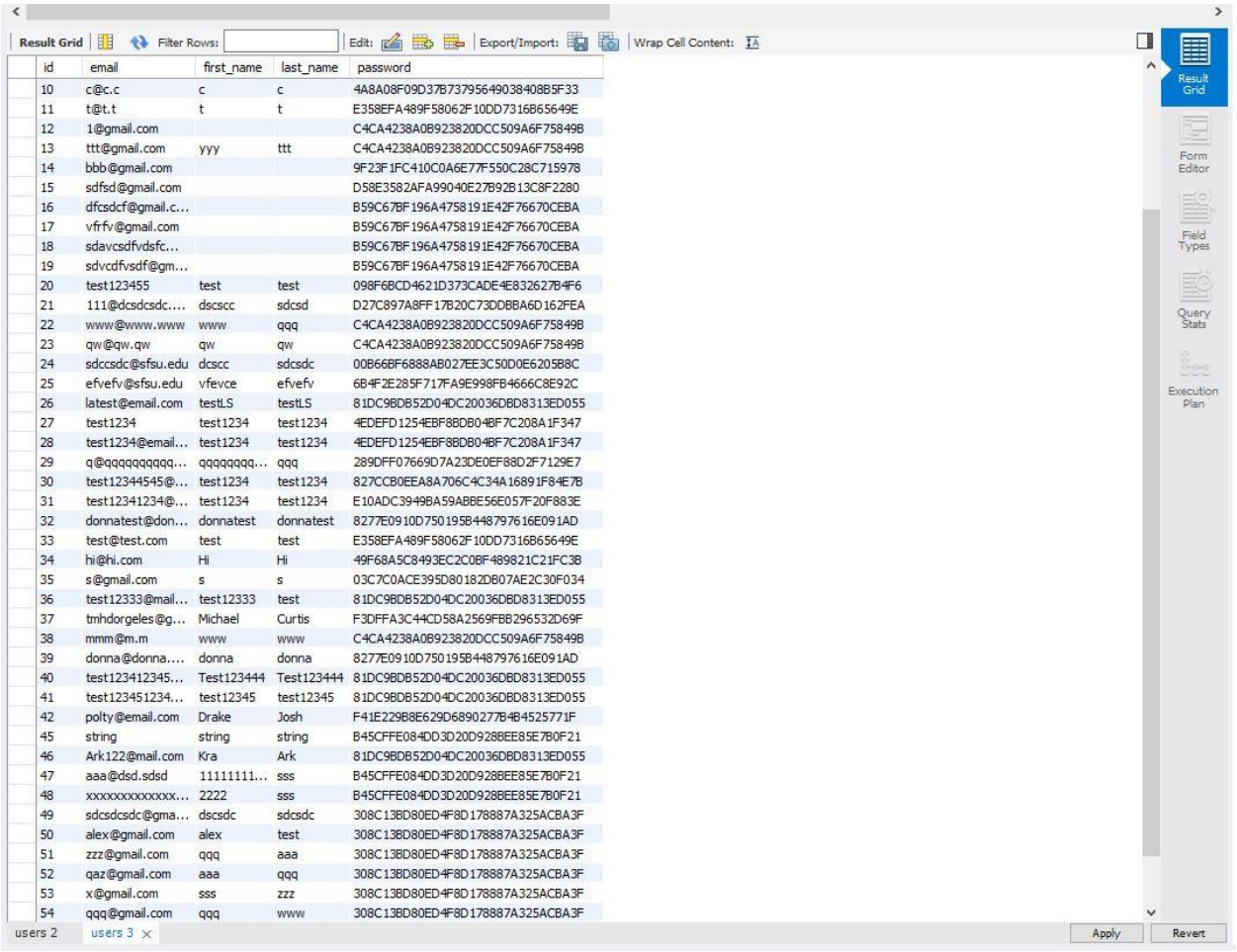
- User's personal information
- User's email address
- User's password

Confirm that you encrypt PW in the DB (with process and screenshots):

The User password is, indeed, encrypted. Without an RSA private key, unauthorized individuals cannot ssh into our database. When our Users create an account, they must enter their created password twice. When they input their password the second time, we check to see if it is the same as the first password input. Afterward, it is stored in our database. MD5 hashing is used to encrypt data in our database. We employ tokens (JWT) in addition to the hashing process to ensure that the user's information is valid and matches our database.

Here's is the method we use to hash the user password:

```
public String hashPassword(String password) throws NoSuchAlgorithmException {
    MessageDigest md = MessageDigest.getInstance("MD5");
    md.update(password.getBytes());
    byte[] digest = md.digest();
    String myHash = DatatypeConverter.printHexBinary(digest).toUpperCase();
    return myHash;
}
```



id	email	first_name	last_name	password
10	c@c.c	c	c	4A8A08F09D37B73795649038408B5F33
11	t@t.t	t	t	E358EFA489F58062F10DD7316B65649E
12	1@gmail.com			C4CA4238A0B923820DCC509A6F75849B
13	ttt@gmail.com	yyy	ttt	C4CA4238A0B923820DCC509A6F75849B
14	bbb@gmail.com			9F23F1FC410C0A6E77F550C28C715978
15	sdfsd@gmail.com			D58E3582AFA99040E27B92B13C8F2280
16	dfcsdcf@gmail.c...			B59C67BF196A4758191E42F76670CEBA
17	vfrfv@gmail.com			B59C67BF196A4758191E42F76670CEBA
18	sdavcsdfvdsfc...			B59C67BF196A4758191E42F76670CEBA
19	sdvcdfvsdf@gm...			B59C67BF196A4758191E42F76670CEBA
20	test123455	test	test	098F6BCD4621D373CADE4E832627B4F6
21	111@dcsdcsdc....	dscscc	sdcsd	D27C897A8FF17B20C73DDBBA6D162FEA
22	www@www.www	www	qqq	C4CA4238A0B923820DCC509A6F75849B
23	qw@qw.qw	qw	qw	C4CA4238A0B923820DCC509A6F75849B
24	sdccsdc@sfsu.edu	dcscc	sdcsdc	00B66BF6888AB027EE3C50D0E6205B8C
25	efvefv@sfsu.edu	vfevce	efvefv	6B4F2E285F717FA9E998FB4666C8E92C
26	latest@email.com	testLS	testLS	81DC9BDB52D04DC20036DBD8313ED055
27	test1234	test1234	test1234	4EDEFD1254EBF8BDB04BF7C208A1F347
28	test1234@email...	test1234	test1234	4EDEFD1254EBF8BDB04BF7C208A1F347
29	q@qqqqqqqqqq...	qqqqqqqq...	qqq	289DFF07669D7A23DE0EF88D2F7129E7
30	test12344545@...	test1234	test1234	827CCB0EEA8A706C4C34A16891F84E7B
31	test12341234@...	test1234	test1234	E10ADC3949BA59ABBE56E057F20F883E
32	donnatest@don...	donnatest	donnatest	8277E0910D750195B448797616E091AD
33	test@test.com	test	test	E358EFA489F58062F10DD7316B65649E
34	hi@hi.com	Hi	Hi	49F68A5C8493EC2C0BF489821C21FC3B
35	s@gmail.com	s	s	03C7C0ACE395D80182DB07AE2C30F034
36	test12333@mail...	test12333	test	81DC9BDB52D04DC20036DBD8313ED055
37	tmhdorgeles@g...	Michael	Curtis	F3DFFA3C44CD58A2569FBB296532D69F
38	mmm@m.m	www	www	C4CA4238A0B923820DCC509A6F75849B
39	donna@donna....	donna	donna	8277E0910D750195B448797616E091AD
40	test123412345...	Test123444	Test123444	81DC9BDB52D04DC20036DBD8313ED055
41	test123451234...	test12345	test12345	81DC9BDB52D04DC20036DBD8313ED055
42	polty@email.com	Drake	Josh	F41E229B8E629D6890277B4B4525771F
45	string	string	string	B45CFFE084DD3D20D928BEE85E7B0F21
46	Ark122@mail.com	Kra	Ark	81DC9BDB52D04DC20036DBD8313ED055
47	aaa@dsd.sdsd	11111111...	sss	B45CFFE084DD3D20D928BEE85E7B0F21
48	xxxxxxxxxxxxx...	2222	sss	B45CFFE084DD3D20D928BEE85E7B0F21
49	sdcsdcsdc@gma...	dscsdc	sdcsdc	308C13BD80ED4F8D178887A325ACBA3F
50	alex@gmail.com	alex	test	308C13BD80ED4F8D178887A325ACBA3F
51	zzz@gmail.com	qqq	aaa	308C13BD80ED4F8D178887A325ACBA3F
52	qaz@gmail.com	aaa	qqq	308C13BD80ED4F8D178887A325ACBA3F
53	x@gmail.com	sss	zzz	308C13BD80ED4F8D178887A325ACBA3F
54	qqq@gmail.com	qqq	www	308C13BD80ED4F8D178887A325ACBA3F

Confirm Input data validation (list what is being validated and what code you used):

Articles & Essays, Art & Film, Clubs, Discords, Tutoring, Posts, and Other are what we look for in the search box input. To do a search, we utilize SQL query. Each search, including the criteria, is not hard programmed.

Here's is the query that was used to do filter and search from the post:

```
("SELECT p FROM Post p WHERE p.Title LIKE CONCAT('%',:query, '%') Or p.Description LIKE CONCAT('%', :query, '%')")
```

```
@Query("SELECT p FROM Post p where p.Tag LIke '%Article%'")

@Query("SELECT p FROM Post p where p.Tag LIke '%Essay%'")

@Query("SELECT p FROM Post p where p.Tag LIke '%ArtAndFilm%'")

@Query("SELECT p FROM Post p where p.Tag LIke '%Discords%'")

@Query("SELECT p FROM Post p where p.Tag LIke '%Tutoring%'")

@Query("SELECT p FROM Post p where p.Tag LIke '%Clubs%'")

@Query("SELECT p FROM Post p where p.Tag LIke '%others%'")

@Query("SELECT p FROM Post p ORDER BY p ASC")

@Query("SELECT p FROM Post p ORDER BY p.photo_Like DESC")
```

Self-Check: Adherence to Original Non-Functional Specs

System requirements:

1. The website application shall be optimized for selected browsers such as Safari, Google Chrome, and Mozilla Firefox. **-DONE**
2. The website application shall display posted media in full width of the screen. **-DONE**
3. Images posted on the application shall be in the format of jpg, png, or jpeg only. **-DONE**

Performance requirements:

4. The website application shall be optimized for desktop/laptop usage. **-DONE**
5. Loading time for the website shall be 5 seconds or less for any page or screen. **-IN PROGRESS**

Storage, security, environmental requirements

6. The website shall handle at least 50 users simultaneously. **-IN PROGRESS**
7. Data shall be stored in the team's SQL database. **-DONE**
8. React UI component's names shall be PascalCase. **-DONE**
9. All other helper files shall be camelCase. **-DONE**
10. Inline styles shall be camelCase. **-DONE**
11. CSS files shall be named using kebab-case. **-DONE**
12. React classes within components shall follow react naming conventions. **-DONE**
13. Multiple files of code shall be written. **-DONE**
14. All CSS files shall be in one common folder. **-DONE**
15. All files shall be named logically according to the job that they perform. **-DONE**

16. Props shall be destructured in order to make the code cleaner and more maintainable. **-DONE**
17. All imports shall be in an order: React import, Library imports (Alphabetical order), Absolute imports (Alphabetical order), Relative imports (Alphabetical order), Import * as. **-IN PROGRESS**
18. There shall be no unneeded comments in code. **-DONE**
19. All debugging console.log() shall be removed in code. **-DONE**
20. .js extensions shall be used for React components. **-DONE**

Marketing, legal requirements (logos, branding, licensing)

21. Specific logos shall be created for this project only. **-DONE**
22. Some images and logos shall be used from San Francisco State University's Website. **-DONE**
23. All icons and frameworks used shall be from a free, open-source library. **-DONE**

Content (size, formats…)

24. Comments shall have a text limit of 250 characters. **-IN PROGRESS**

Privacy (what data is collected, how is it used…)

25. The login shall be required for all Registered Users (Approved Users and Admin). **-DONE**
26. Passwords shall be encrypted in the database where it is stored. **-DONE**
27. The Registered Users session shall end once they click "Log Out". **-DONE**

Optimization & Performance:

28. The website application language shall be in English. **-DONE**

29. The application will upload User’s posts and comments in real time. **-DONE**
30. The website application shall be deployed from the team's Amazon Web Services account. **-DONE**
31. The website application shall store the Users data in the team’s MySQL database. **-DONE**
32. User information shall not be duplicated in the database. **-DONE**

Security:

33. Users data shall be securely stored in the backend database. **-DONE**
34. Passwords shall be protected. **-DONE**
35. Passwords shall not be stored in the database as plain text. **-DONE**
36. When an Unregistered User signs up: **-DONE**
    - Passwords shall be at least 8 characters. **-DONE**
    - Passwords shall require at least one uppercase letter. **-DONE**
    - Passwords shall require at least one lowercase letter. **-DONE**
    - Passwords shall require at least one number. **-DONE**
    - Passwords shall require at least one special character. **-DONE**

List of Contributions to the Document

Estefanos (Backend Lead) -

Participated in team meetings and discussions. Worked on the comment section. Helped implement and finalize messaging API. Worked on posts. Implemented group chat. Completed all features besides the comment section. Created new tables to have working back end and front end. Worked on group messaging, post, and comment end points. Wrote chatroom.js, group message dto, dao and controllers. Helped Database team with Best Practices for Security. Helped draft Best Practices for Security. Introduced sockjs and STOMP for web-socket purposes.

Brianna (Frontend Lead) -

Contributed to discussions on Discord. Attended all team meetings. Drafted, reviewed, and finalized Code Review. Provided Team 3 with the code for Code Review. Reviewed Team 3 code. Designated M4 editor. Added last bits of CSS styling. Reviewed and revised Usability Test Plan. Reviewed and Revised QA Test Plan. Redid User Satisfaction Table for posting. Finalized all sections of the M4 Doc before submission.

Brian (Database Master) -

Participated in team meetings and discussions. Helped contribute to Product Summary by drafting. Helped create and finalize user test plan questionnaires (user satisfaction table). Drafted, reviewed, and finalized Best Practices for Security. Helped with Code Review by giving feedback. Reviewed all sections of the Doc.

Mohamed (GitHub Master) -

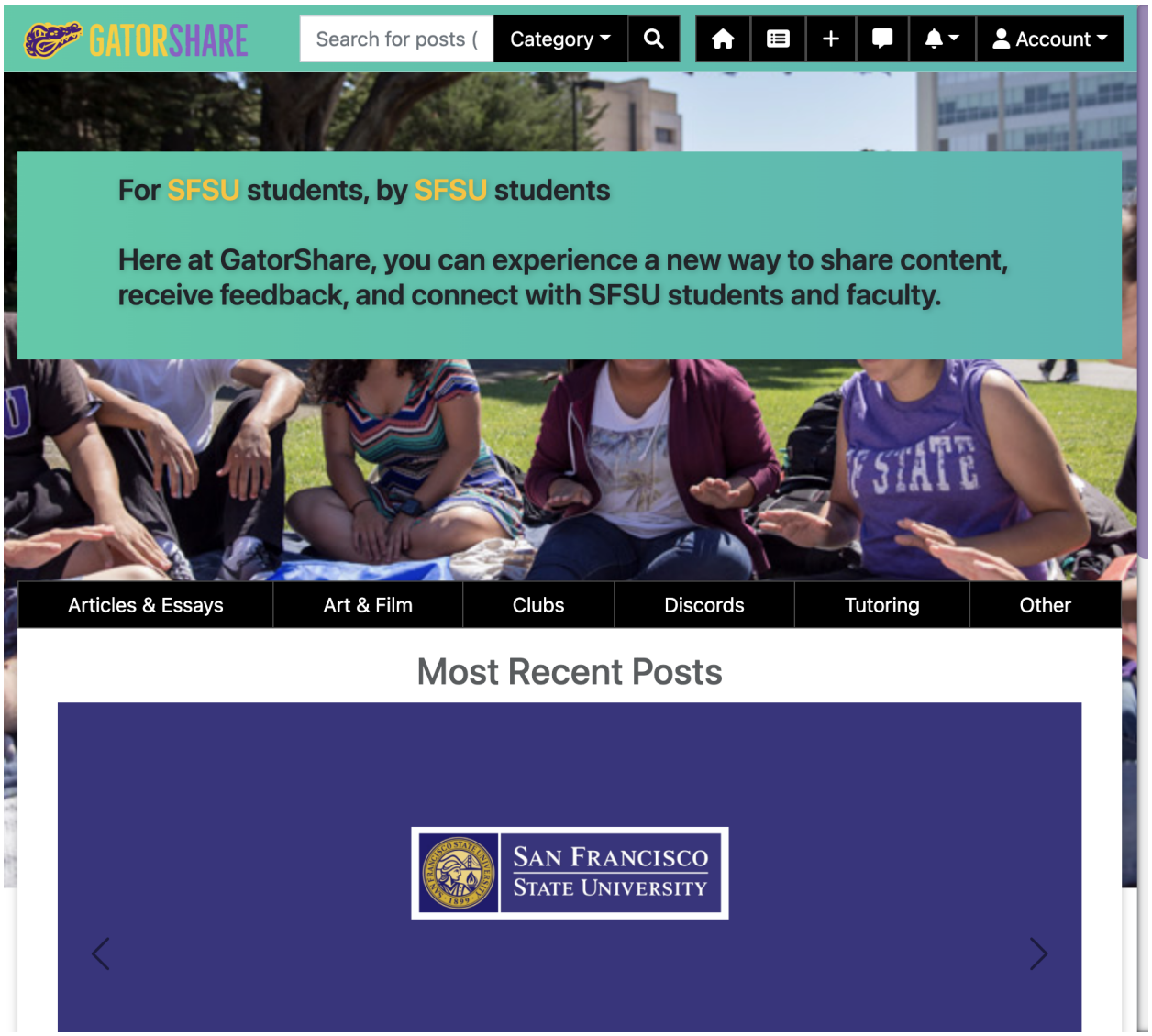
Participated in team meetings and discussions. Brainstormed on the final modifications and input ideas for improvement on the platform. Performed a series of testing for the platform and submitted feedback describing the task performed. Worked on the project descriptions. Had a Tester for our application. Drafted and filled out our Usability Test Table. Drafted and finalized the Task Description. Finalized and tested our User Satisfaction Table.

Aleksandr (Front End) -

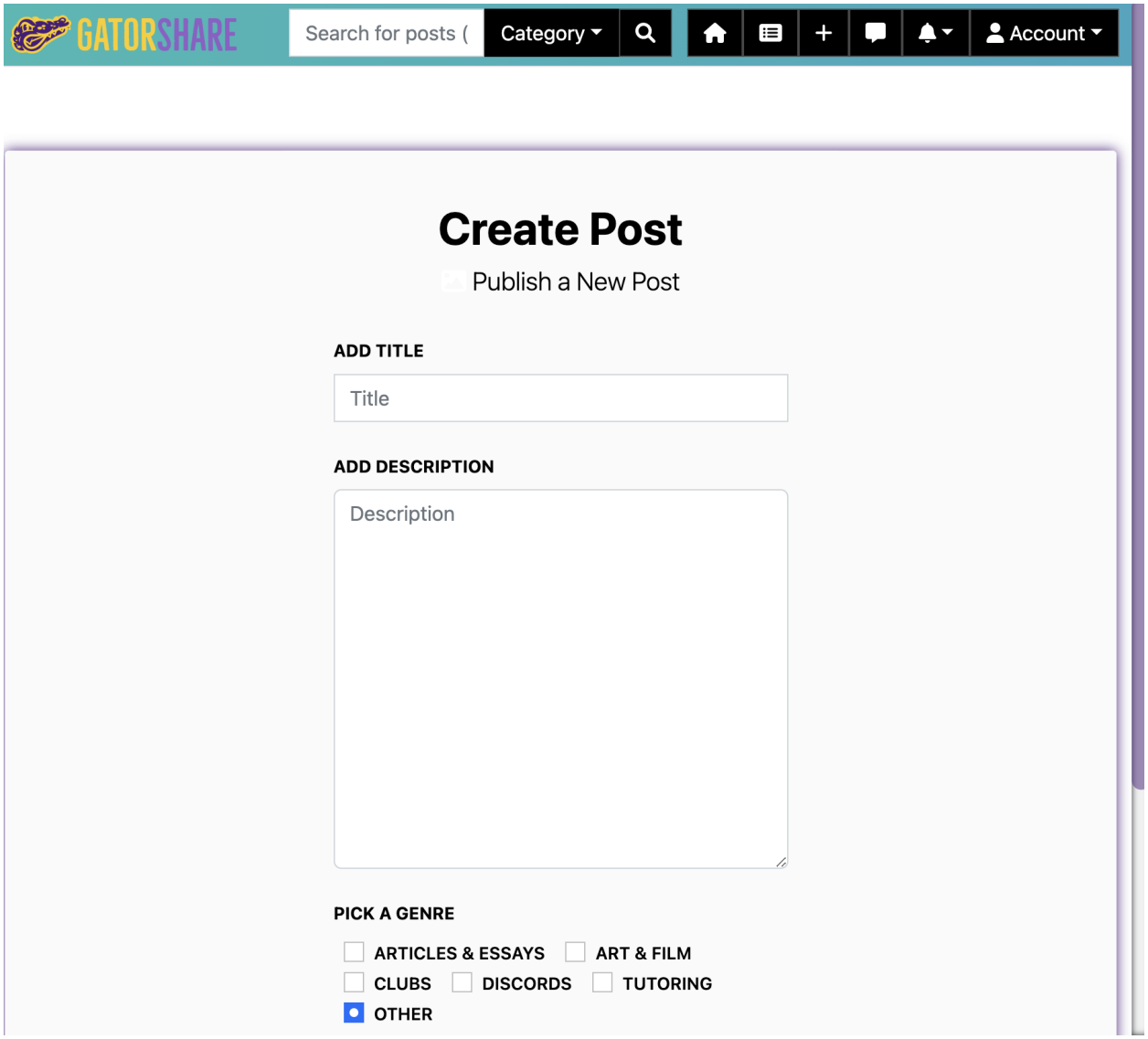
Contributed to discussions on Discord. Attended all team meetings. Contributed to code review. Created ‘edit’ post functions. Created access to other user’s profile pages. Edited the ‘messages’ text box. Fixed search bar functionalities. Added categories to the search bar. Added sorting to the posts. Refactored the code to send fewer requests. Updated links to search bar. Prepared code for review and slightly helped review the code for the other team. Reviewed requirements priorities

Product Screenshots

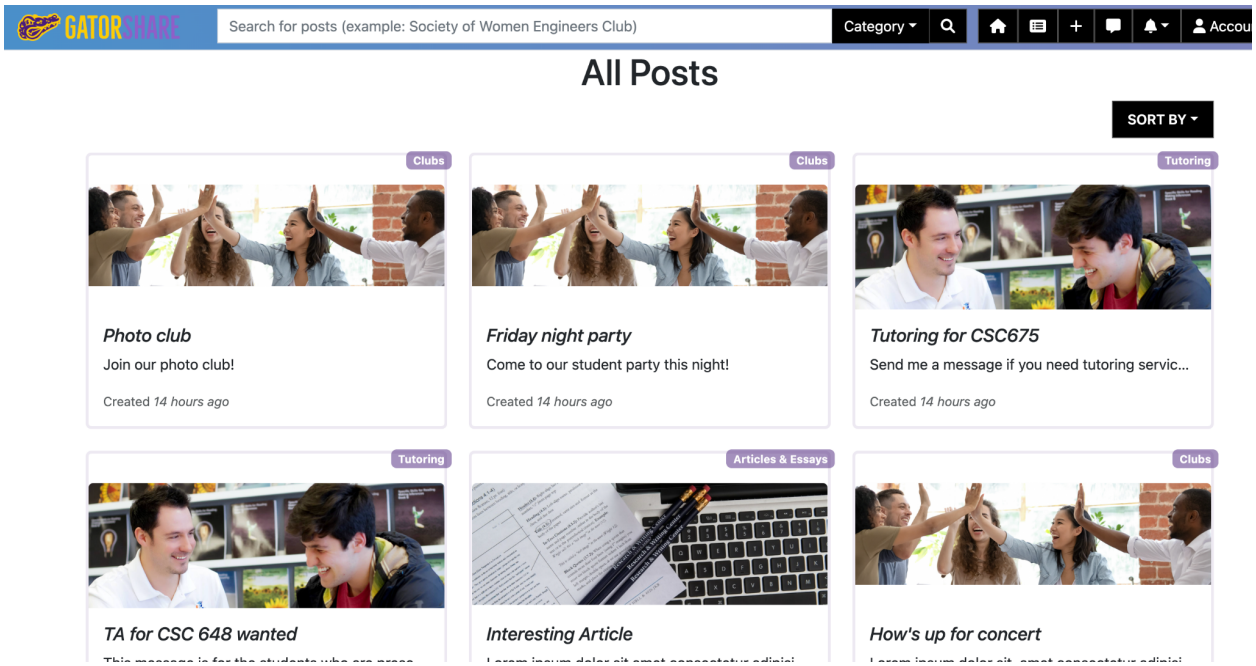
Home Page



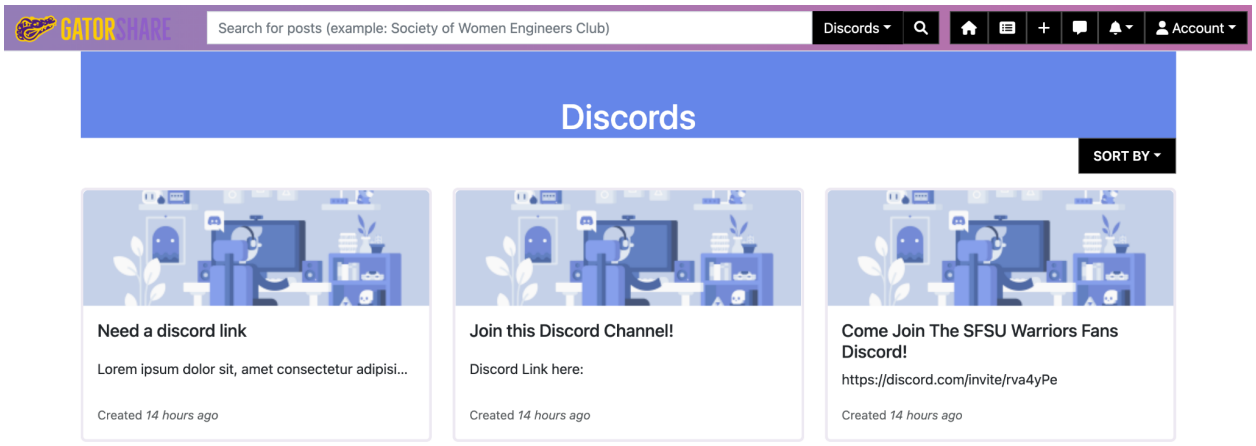
Create a Post Page



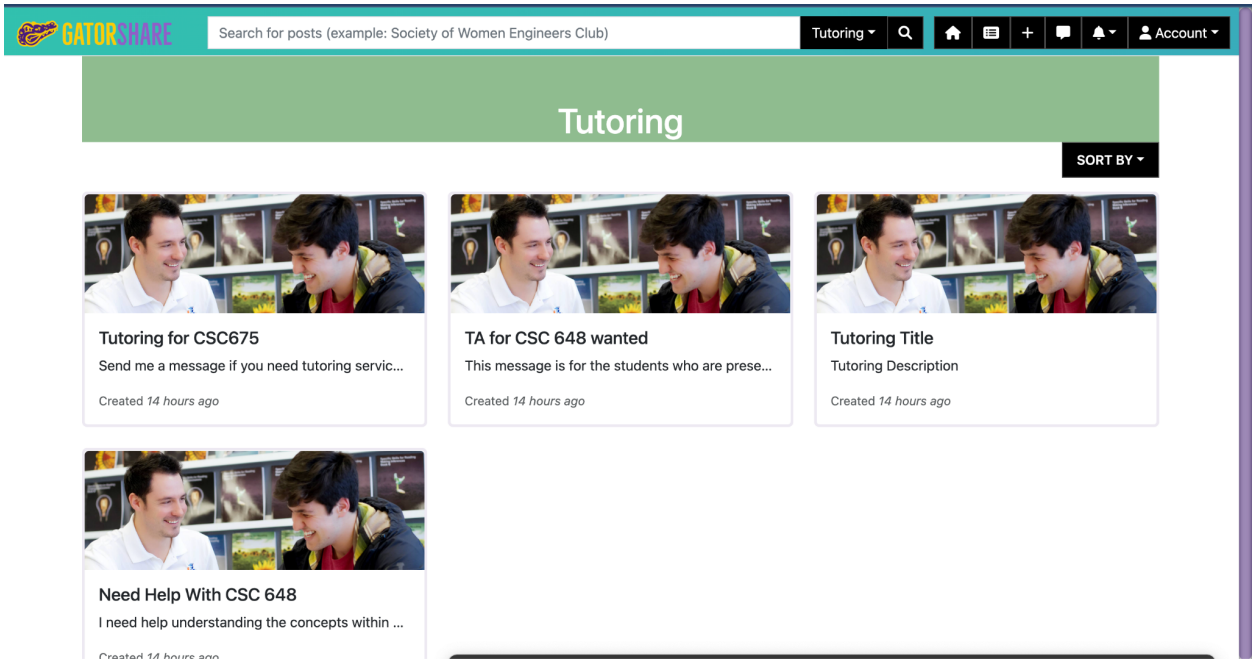
Search Results Page



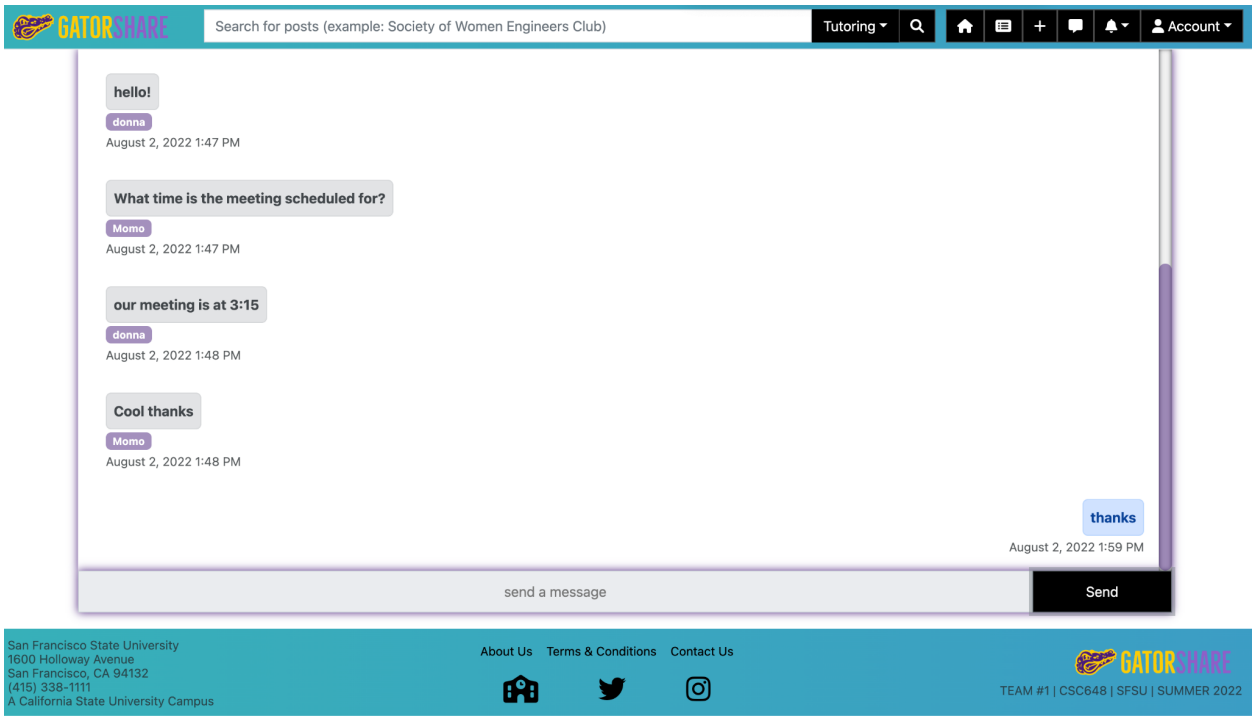
Discords Page



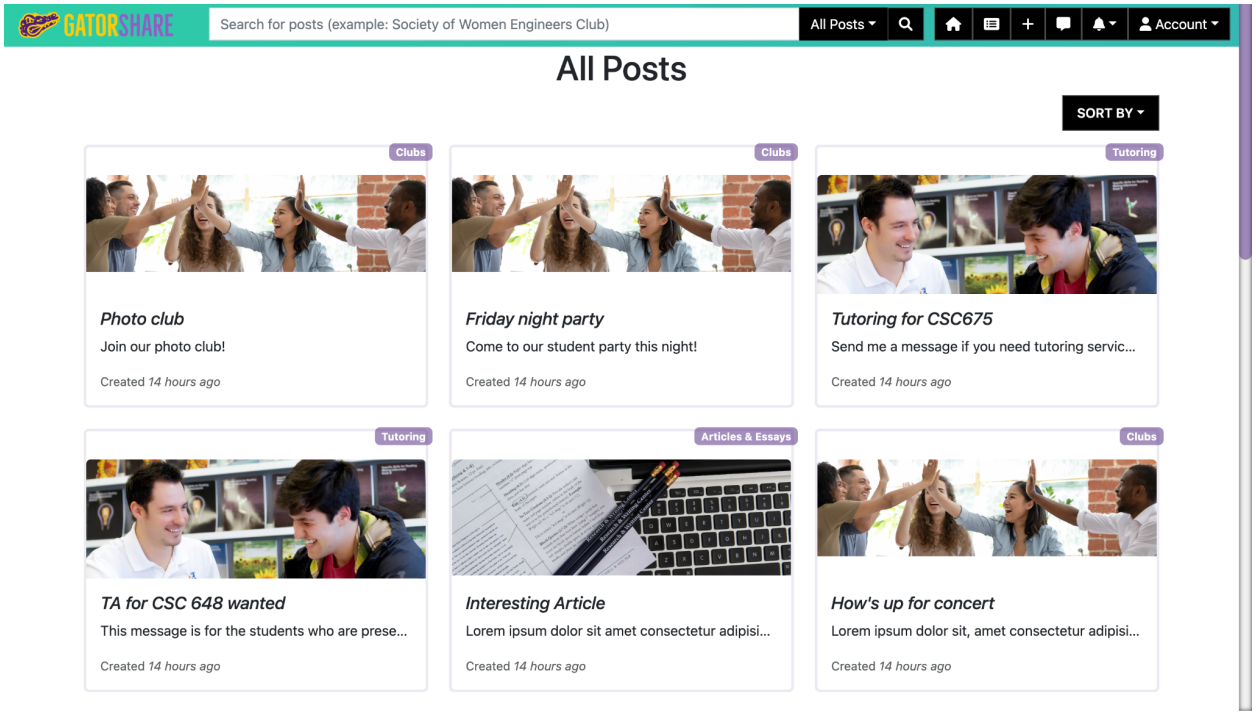
Tutoring Page



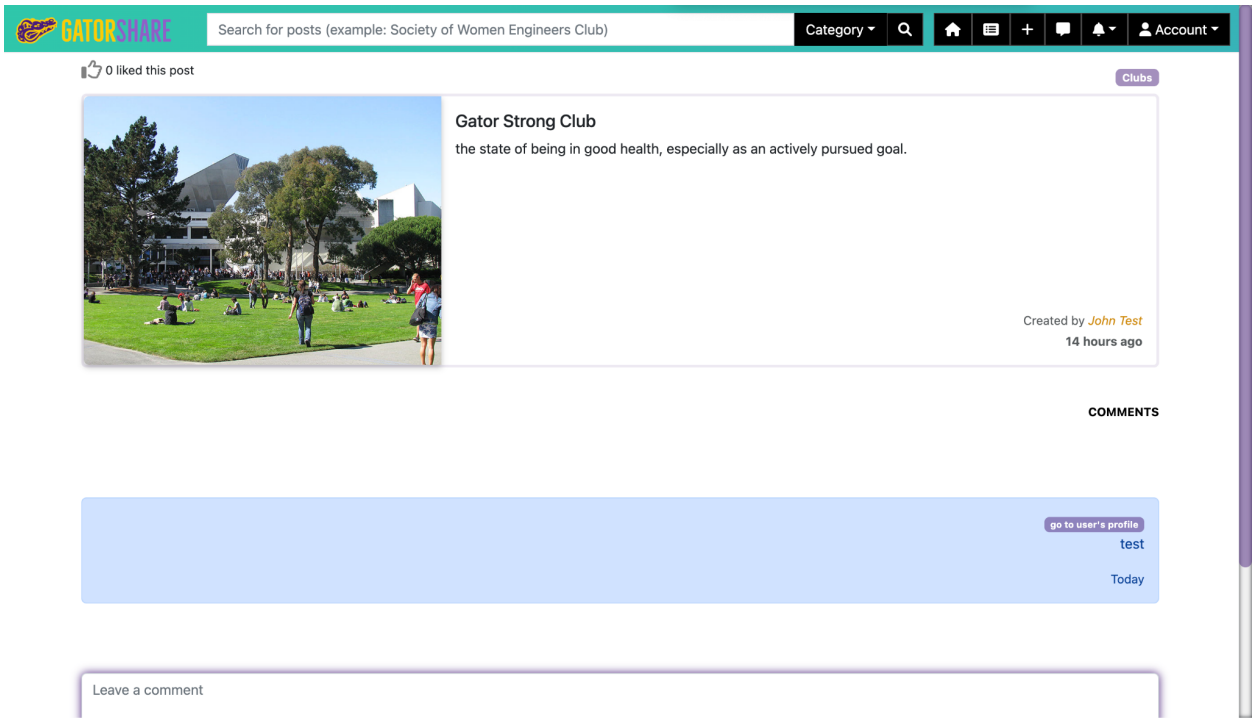
Messages


GATORSHARE
Search for posts (example: Society of Women Engineers Club)
Tutoring
Account
hello!
donna
August 2, 2022 1:47 PM
What time is the meeting scheduled for?
Momo
August 2, 2022 1:47 PM
our meeting is at 3:15
donna
August 2, 2022 1:48 PM
Cool thanks
Momo
August 2, 2022 1:48 PM
thanks
August 2, 2022 1:59 PM
send a message
Send
San Francisco State University
1600 Holloway Avenue
San Francisco, CA 94132
(415) 338-1111
A California State University Campus
About Us
Terms & Conditions
Contact Us
GATORSHARE
TEAM #1 | CSC648 | SFSU | SUMMER 2022

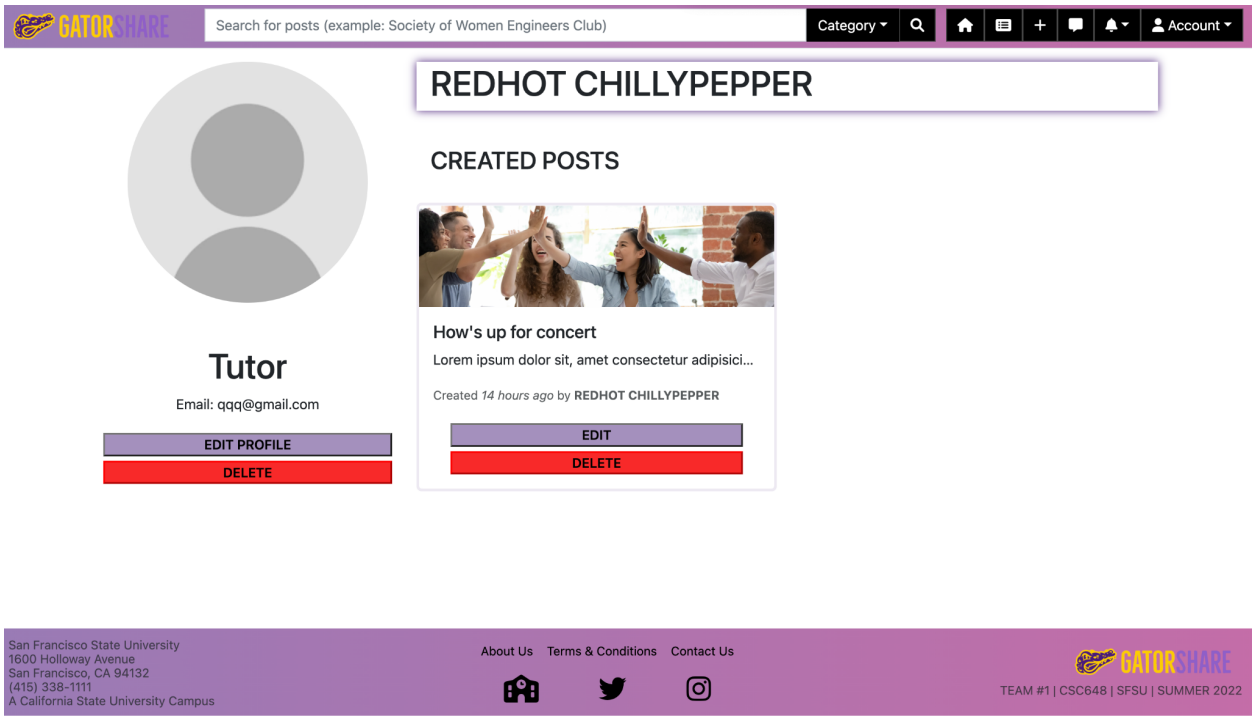
All Posts


GATORSHARE
Search for posts (example: Society of Women Engineers Club)
All Posts
Account
All Posts
SORT BY
Clubs
Photo club
Join our photo club!
Created 14 hours ago
Clubs
Friday night party
Come to our student party this night!
Created 14 hours ago
Tutoring
Tutoring for CSC675
Send me a message if you need tutoring servic...
Created 14 hours ago
Tutoring
TA for CSC 648 wanted
This message is for the students who are prese...
Created 14 hours ago
Articles & Essays
Interesting Article
Lorem ipsum dolor sit amet consectetur adipisi...
Created 14 hours ago
Clubs
How's up for concert
Lorem ipsum dolor sit, amet consectetur adipisi...
Created 14 hours ago

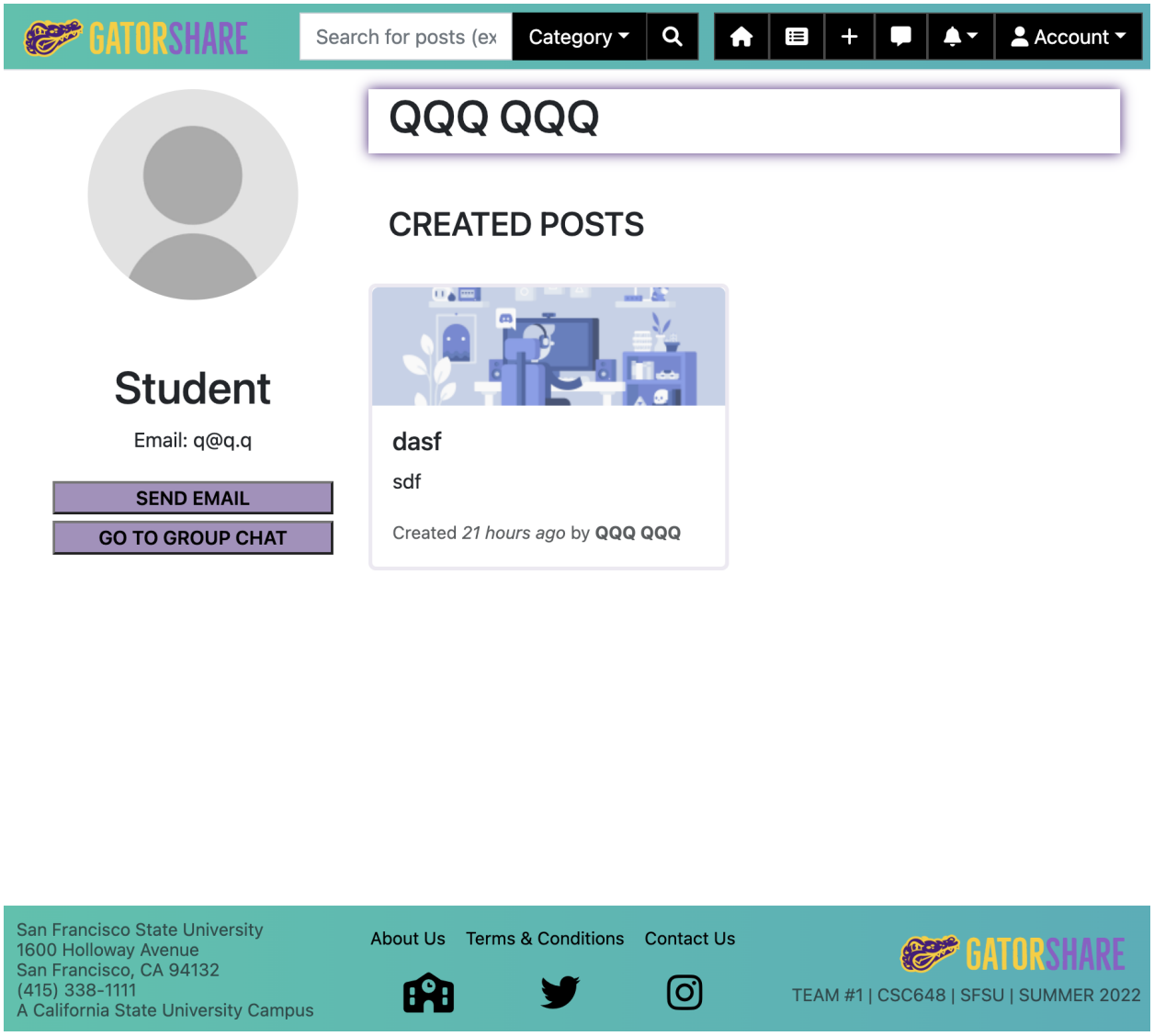
View Post Page w/ Comments and Likes



Profile Page w/ Delete Post

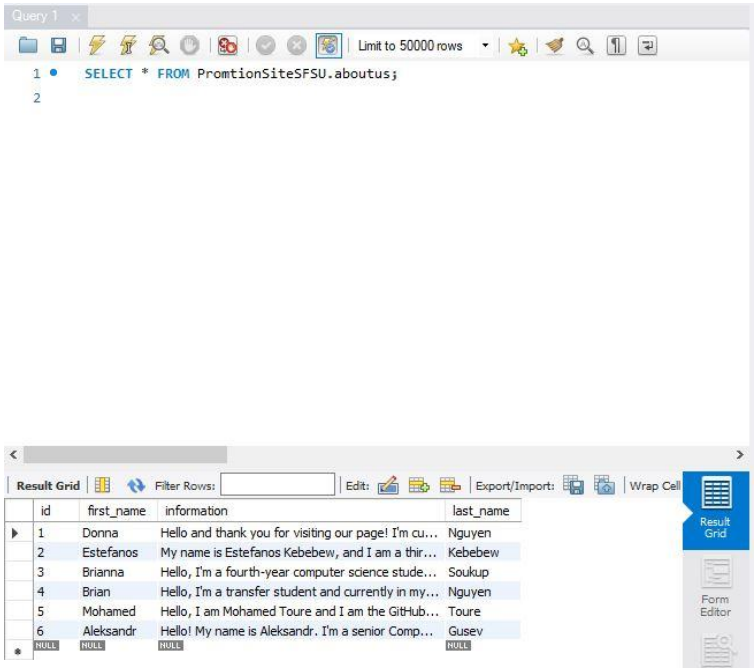


User Profile



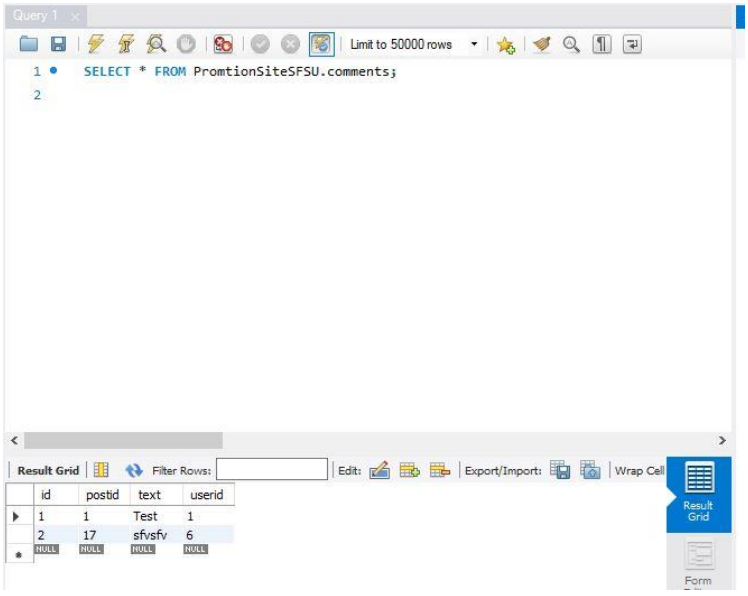
Database Table Screenshots

About Us



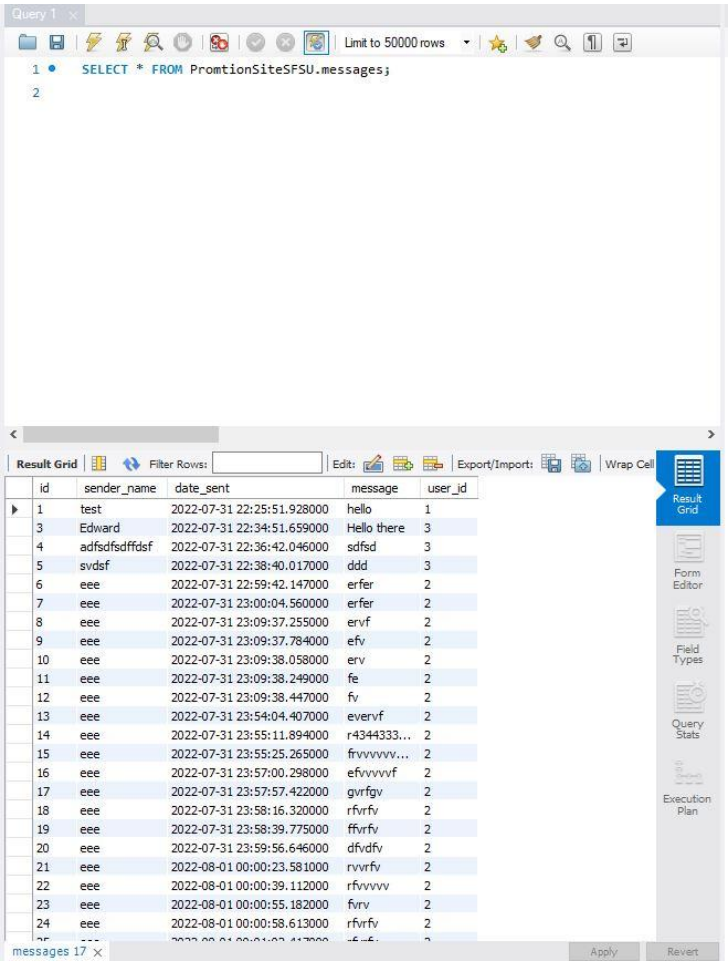
id	first_name	information	last_name
1	Donna	Hello and thank you for visiting our page! I'm cu...	Nguyen
2	Estefanos	My name is Estefanos Kebebew, and I am a thir...	Kebebew
3	Brianna	Hello, I'm a fourth-year computer science stude...	Soukup
4	Brian	Hello, I'm a transfer student and currently in my...	Nguyen
5	Mohamed	Hello, I am Mohamed Toure and I am the GitHub...	Toure
6	Aleksandr	Hello! My name is Aleksandr. I'm a senior Comp...	Gusev

Comments



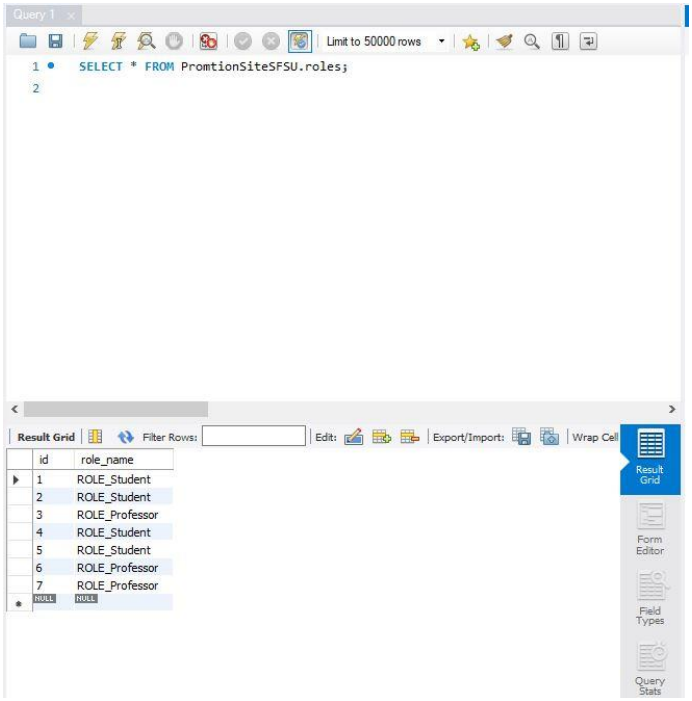
id	postid	text	userid
1	1	Test	1
2	17	sfvsfv	6

Messages



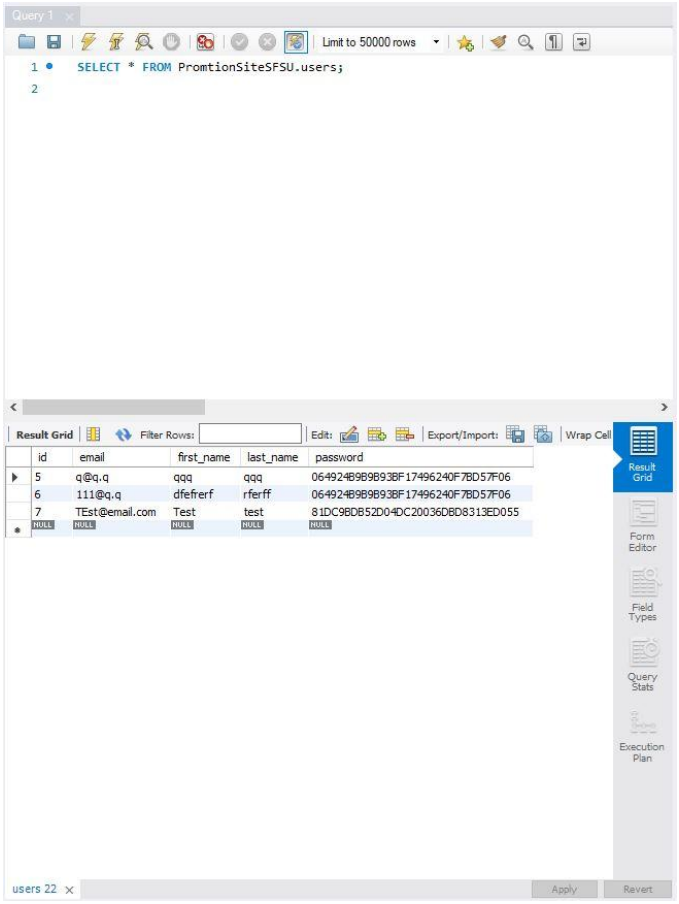
id	sender_name	date_sent	message	user_id
1	test	2022-07-31 22:25:51.928000	hello	1
3	Edward	2022-07-31 22:34:51.659000	Hello there	3
4	adfsdfsdffdsf	2022-07-31 22:36:42.046000	sdfsd	3
5	svdsf	2022-07-31 22:38:40.017000	ddd	3
6	eee	2022-07-31 22:59:42.147000	erfer	2
7	eee	2022-07-31 23:00:04.560000	erfer	2
8	eee	2022-07-31 23:09:37.255000	ervf	2
9	eee	2022-07-31 23:09:37.784000	efv	2
10	eee	2022-07-31 23:09:38.058000	erv	2
11	eee	2022-07-31 23:09:38.249000	fe	2
12	eee	2022-07-31 23:09:38.447000	fv	2
13	eee	2022-07-31 23:54:04.407000	evervf	2
14	eee	2022-07-31 23:55:11.894000	r4344333...	2
15	eee	2022-07-31 23:55:25.265000	frvvvvvv...	2
16	eee	2022-07-31 23:57:00.298000	efvvvvvf	2
17	eee	2022-07-31 23:57:57.422000	gvrfgv	2
18	eee	2022-07-31 23:58:16.320000	rfvrfv	2
19	eee	2022-07-31 23:58:39.775000	ffvrfv	2
20	eee	2022-07-31 23:59:56.646000	dfvdfv	2
21	eee	2022-08-01 00:00:23.581000	rvvrfv	2
22	eee	2022-08-01 00:00:39.112000	rfvvvvv	2
23	eee	2022-08-01 00:00:55.182000	fvrv	2
24	eee	2022-08-01 00:00:58.613000	rfvrfv	2

Roles

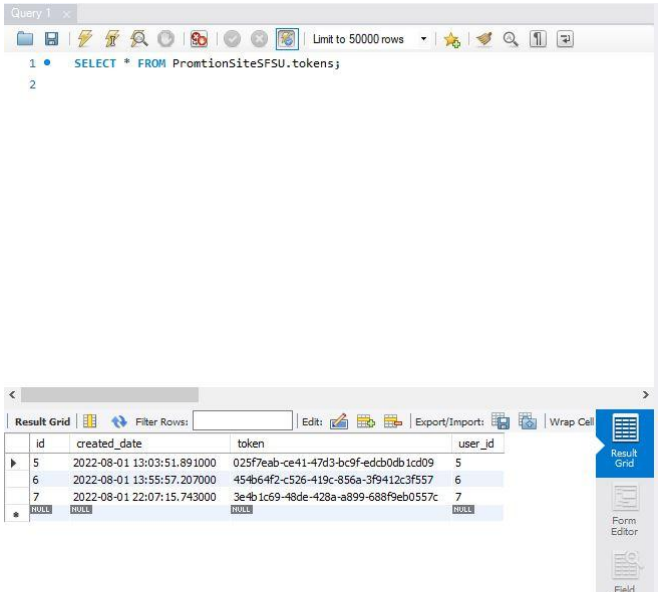


id	role_name
1	ROLE_Student
2	ROLE_Student
3	ROLE_Professor
4	ROLE_Student
5	ROLE_Student
6	ROLE_Professor
7	ROLE_Professor
NULL	NULL

Users

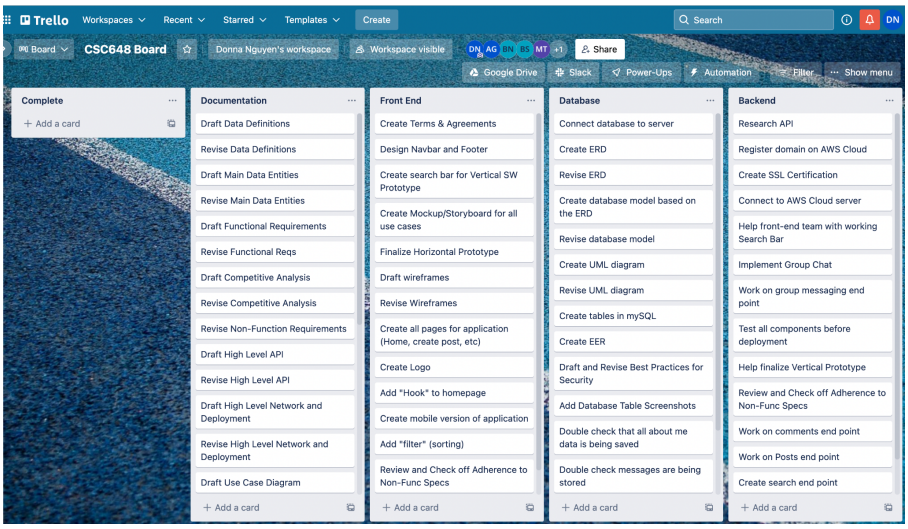


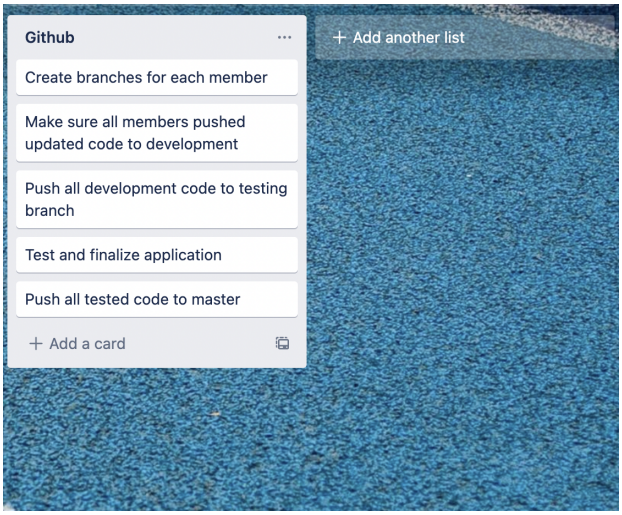
Tokens



Task Management Screenshots

Note: Tasks were moved back to their designated boards to show how the tasks are organized. Tasks are separated by “Documentation”, “Front End”, “Database”, “Backend”, and “Github”. All tasks are sorted based on a team member's role. “Documentation” is for all members to complete. When a teammate finishes a task, they will move the task to “Complete”. Depending on the Milestone and revision needed, these tasks may sometimes be moved back to their designated board for further revision.





Team Member Contributions

Estefanos (Backend Lead) - 10

Consistently communicated in Discord channels. Had open conversations and discussions with the team. Attended all team Zoom meetings. Successfully registered the application's domain from AWS Cloud. Researched where to host the website that AWS provided. Tested several different types of cloud storages such as using amplify, S3 Bucket, and elastic beanstalk. Decided to go with S3 Bucket and used Cloudfront to access the bucket. Used Cloudfront to create SSL certification. Debugged back-end and server-side errors to help the application project run. Implemented AWS RDS and AWS Instance to host the database. Helped teammates connect to the AWS Cloud server. Helped teammates connect to MySQL database. Efficiently fixed bugs related to the backend. Helped with constructing main use case ideas. Reviewed and finalized functional and nonfunctional requirements to make sure all requirements are feasible. Reviewed and finalized high-level system architecture. Researched and contributed to Competitive Analysis. 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. Heavily contributed to the Vertical Software Prototype. Helped revise and finalized the Database model. Helped work on and revised High Level Database Architecture and Organization. Worked alongside the front-end team to integrate several functionalities such as creating a working Search bar, saving the Sign Up and Log In data, saving the Create Post data, and much more. Consistently tested alongside the front-end team to make sure the application is deployed correctly and ready to be used. Helped implement and finalize messaging API. Worked on posts. Implemented group chat.

Completed all features such as Creating Post, Deleting Post, Commenting, etc. Created new tables to have working back end and front end. Tested all components. Worked on group messaging, post, and comment end points. Wrote chatroom.js, group message dto, dao and controllers. Helped Database team with Best Practices for Security. Introduced Sockjs and STOMP for web-socket purposes. Added new endpoints to the backend. Modified the database. Deployed the application. Worked with the front end team to have the final product of the website. Fixed all errors.

Brianna (Frontend Lead) - 10

Consistently communicated in Discord channels. Contributed to many ideas. Consistently updated the team with work done. Attended all team Zoom meetings. Proposed the idea of our web application. Provided ideas for the basic function of our application's purpose. Contributed to main use cases by coming up with main use case ideas. Created Use Case Diagrams for each main use case. Helped come up with, and created the application's logo for our website. Created a shared Figma file for the Frontend team to utilize the creation of user flow diagrams and wireframes in the near future. Reviewed and finalized Functional and Nonfunctional requirements to confirm that the requirements are feasible. Researched and contributed to Competitive Analysis. 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. Made the effort to ensure everyone was aware of deadlines. Collaborated with the front-end team on the Horizontal Prototype. Helped

check-in with team mates about their progress with specific tasks. Contributed to the creation of UI/UX wireframes. Helped revise Use Case mockups in Milestone 2. Helped code the Log In, Sign Up, Home and Create Post UI/UX. Created and finalized the Wireframe diagram. Added all screenshots and data related to wireframes into the Milestone 3 documentation. Helped revise all front-end portions of Milestone 2's revision document. Designated Milestone 3 editor. Helped revise and finalize Milestone 3 documentation. Drafted, reviewed, and finalized Code Review. Provided Team 03 with the code for Code Review. Reviewed Team 03 code. Designated M4 editor. Added last bits of CSS styling. Reviewed and revised Usability Test Plan. Reviewed and Revised QA Test Plan. Redid User Satisfaction Table for posting. Finalized all sections of the M4 Doc before submission. Created the draft for M5. Reviewed and revised all P1 requirements. Created list of Product Screenshots. Organized database screenshots. Reviewed, revised and finalized all areas of M5 doc.

Brian (Database Master) - 7

Attended all team Zoom meetings. Communicated on Discord. Helped brainstorm and create a table within MySQL for users with basic information (rough draft). Reviewed and finalized functional and nonfunctional requirements in order to make sure the requirements are feasible and implementable. Contributed to creating data items and entities by coming up with ideas and adding input. Helped review and finalize data items and entities. Helped provide documentation for back-end lead and also provided documentation about the database thus far. Researched and contributed to competitive analysis. 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. Reviewed, revised and finalized Milestone 2 as the Milestone 2 editor.

Contributed to data definitions. Helped revise Main Data Entities and Functional Requirements. Revised and finalized High Level Database Architecture and Organization. Revised and finalized the Entity Relationship Diagram. Revised and Finalized the Enhanced Entity Relationship diagram. Designated Scribe for the Horizontal Prototype meeting. Wrote notes and collected feedback for the team. Helped contribute to Product Summary by drafting. Helped create User Test Plan Questionnaires (user satisfaction table). Drafted, reviewed, and finalized Best Practices for Security. Helped with Code Review by giving feedback. Reviewed all sections of the Doc. Provided screenshots of database tables.

Mohamed (GitHub Master) - 7

Communicated on Discord and attended scheduled team Zoom meetings. Designated M1 editor. Moved all code in the development branch into the testing branch. Helped test files and finalize code before moving to the master branch. Helped edit M1 documentation. Helped review and finalize executive summary to make sure it's legible. Reviewed main use cases. Researched and contributed to competitive analysis. Reviewed and revised M1V2. Contributed to data definitions. Created, revised, and finalized the application's Terms and Agreements. Helped design parts of the applications Footer. Helped revise Main Data Entities and Functional Requirements. Helped brainstorm the Entity Relationship Diagram design. Helped brainstorm with the front-end team about specific page designs. Helped revise the Competitive Analysis for Milestone 2 revision. Helped revise the Non-Functional Requirement. Pushed M1V2 and M2V2 into the team's Github Repo. Brainstormed on the final modifications and input ideas for improvement on the platform. Performed a series of testing for the platform and submitted feedback describing the task performed. Worked on the project descriptions. Had a Tester for our

application. Drafted and filled out our Usability Test Table. Drafted the Task Descriptions. Added testers feedback to our User Satisfaction Table. Appended all V2 Milestone documents. Designated M5 editor. Revised and finalized M5.

Aleksandr (Front End) - 10

Consistently communicated in Discord channels. Attended all scheduled team Zoom meetings. Created the front end codebase (React App) and created a rough outline of the web application. Created and designed the “About Us" page. Helped sketch a rough draft of the user flow diagram. Contributed to main use cases along with reviewing each use case. Gave input to the Front End Lead about the application’s logo design in order to have a presentable logo for display. Reviewed and finalized functional and nonfunctional requirements to confirm the requirements are feasible. Helped manage and maintain the frontend branch of the GitHub repo. Helped with formatting the M1 document. Consistently updated the document’s table of contents and reviewed M1 document for errors. Fixed push and overwriting errors on GitHub. Cleaned up repo. Researched and contributed to competitive analysis. 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. Spent countless hours working with the front-end team and back-end lead to have a working, user-friendly, and appealing Horizontal Prototype. Helped alongside the

front-end lead to create UI/UX wireframes. Coded most of the Horizontal prototype. Helped revise and finalize the Entity Relationship Diagram. Helped revise and implement the "Agreement" portion for the website Terms and Agreements. Helped review and update Functional Requirements and Wireframes portion of the Milestone 3 document. Contributed to code review. Created 'edit' post functions. Created access to other user's profile pages. Edited the 'messages' text box. Fixed search bar functionalities. Added categories to the search bar. Added sorting to the posts. Refactored the code to send fewer requests. Updated links to search bar. Prepared code for review and slightly helped review the code for the other team. Reviewed requirements priorities. Revised and contributed to the M5 document. Worked alongside Backend Lead to deliver the website in time. Added new components to our application. Refactored and organized a lot of code. Made our application mobile friendly. Added messages, comments, and profile pages. Made sure that all requests used the correct endpoints. Spent over 15 hours on Zoom with Estefanos in order to complete the project. Offered new solutions in order to fix existing errors. Added delete post and account functionality. Debugged and fixed application for demoing.

Post Analysis

This project has been a great learning experience for all of us. We have all run into errors, had a hard time completing tasks in time, bumped heads, had differing opinions, and also learned from one another. Each milestone we completed was more knowledge gained week by week, and despite the accelerated pace, we all enjoyed this project.

One of the biggest challenges our team faced was time management. During the beginning of this course, we all had a vision in mind. Although we were able to create our application, it took a lot of time, patience, and coffee breaks! In the end, we also had to compromise some features that we wanted, such as allowing a User to upload images.

To balance coding our application, having weekly team meetings, attending classes, along with writing our documentation biweekly was very difficult for us. At one point, we had hit a wall where we submitted one of our Milestones at the very last minute. This was a moment where we all decided that we needed to pause and gather ourselves. We needed to reflect and set better internal deadlines, assign tasks earlier, and spend much more time on both our application and the documentation itself.

Another challenge we faced was also communication issues. During the beginning of this course, some of us were not very communicative. It was a bit stressful trying to have teammates open up. As time went on, we also faced challenges such as how sometimes our ideas would clash, we interpreted instructions differently, and also how Discord messages may come off unprofessional. At the end of the day, we were all able to talk things out and get along simply by communicating to one another better. We learned to be patient with one another and to compromise by respecting one another's ideas and thought processes.

If there is one thing I as a Team Lead could change after doing this project, it is that I wish my team and I talked about having a "helper" or "assistant", that way we can have extra teammates work on either the front or backend. I wish I also set stricter internal deadlines and prioritized more tasks. If I could do this project again, I would lay out all tasks to do and assign them in a more balanced and prioritized manner, that way one person would not be doing "more work" than the other.

One of the biggest lessons we've learned was learning how to give the "client what they want". For example, our team put in the utmost effort in our Horizontal Prototype. We finished the documentation, demoed our application, and double checked our work all in a timely manner. Despite some deployment issues, we thought we did a great job. But, when presenting our prototype, we were extremely disappointed when we heard our "Client" (Professor) wasn't satisfied with our execution. Despite our time and effort, this was our biggest lesson learned—how we can deliver what we want to give, and how the client can be satisfied in our delivery. This challenge allowed us to fix our mistakes, learn about our Client, and execute our application better the second time around.