Final Project for SW Engineering Class CSC 648-848 Section 05 Fall 2022

Team 7

Gatormmunity

Anthony Zhang (Team Lead),
azhang12@mail.sfsu.edu

Marwan Alnounou

Mohamed Sharif

Jose Lopez

Florian Cartozo

Product URL: http://54.241.101.69/

Date: 12/15/22

Table of Contents

2) Product Summary

Name of the Product: Gatormmunity

All Major Committed Functions:

1. All users shall be able to contact the developers via the Contact Us page.
2. All users shall be able to view the home page.
3. All users shall be able to view the about page to learn about Gatormmunity.
4. A guest user shall be able to view the registration page.
5. A guest user shall be able to view the login page.
6. A guest user shall be able to register for an account.
7. An approved user shall be able to log in to their account.
8. An approved user shall be able to reset their password by contacting the admins via the Contact Us page.
9. An approved user shall be able to report other users via the Contact Us page.
10. An approved user shall be able to sort forum threads.
11. An approved user shall be able to create a forum thread.
12. An approved user shall be able to assign their forum thread to a category.
13. An approved user shall be able to create a forum post.
14. An approved user shall be able to attach an image when creating a thread. The image will be displayed in the first post of the thread.
15. An approved user shall have a profile page.
16. An approved user's profile page shall show their details, e.g. their full name.
17. An approved user's profile page shall show their recent activities, e.g. their recently created listings or threads.
18. An approved user shall be able to view a user's profile page.
19. An approved user shall have a profile picture.
20. An approved user shall be able to change their profile picture.
21. An approved user shall be able to create a user group.
22. An approved user shall be able to join a user group after being invited to it, via the link provided in the invite message.
23. An approved user shall be able to send direct messages to other users.
24. An approved user shall be able to receive direct messages from other approved users.
25. An approved user shall be able to send messages in Gator Chat.
26. An approved user shall be able to see marketplace listings in the marketplace.
27. An approved user shall be able to select marketplace listings in the marketplace.
28. An approved user shall be able to report marketplace listings via the Contact Us page.
29. An approved user shall be able to buy items from the marketplace by contacting a listing's seller.
30. An approved user shall be able to sell items on the marketplace by creating a listing.
31. An approved user who is selling an item shall be able to assign their listing to a category.

32. An approved user shall be able to upload a picture of an item they are selling.
33. An approved user shall be able to delete their own listings from the marketplace.
34. An approved user shall be able to search for users.
35. An approved user shall be able to search for marketplace listings.
36. An approved user shall be able to search for forum threads.
37. An approved user shall be able to apply filters to their user search results.
38. An approved user shall be able to apply filters to their marketplace search results.
39. An approved user shall be able to apply filters to their forum thread search results.
40. A moderator shall be able to approve unapproved users via the Admin page.
41. A moderator shall be able to reject unapproved users via the Admin page.
42. A moderator shall be able to ban unapproved and approved users from Gatormmunity via the Admin page.
43. A moderator shall be able to delete forum threads.
44. A moderator shall be able to delete forum posts.
45. A moderator shall be able to delete listings in the marketplace.
46. An administrator shall be able to appoint moderators via the Admin page.
47. An administrator shall be able to unappoint moderators via the Admin page.
48. An administrator shall be able to ban moderators from Gatormmunity after first unappointing the moderator via the Admin page.
49. An administrator shall be able to change a user's password via the Admin page.
50. A group member shall be able to invite other users to their group.
51. A group member shall be able to leave their group.
52. A group member shall be able to create forum threads in their group's forum.
53. A group member shall be able to create forum posts in their group's forum.
54. A group member shall be able to send messages in their group's chat.
55. A group moderator shall be able to kick group members from their group via the Group Members page.
56. A group moderator shall be able to delete forum threads in their group's forum.
57. A group moderator shall be able to delete forum posts in their group's forum.
58. A group administrator shall be able to delete their group.
59. A group administrator shall be able to appoint group moderators via the Group Members page.
60. A group administrator shall be able to unappoint group moderators via the Group Members page.
61. A group administrator shall be able to kick group moderators from the group after first unappointing the moderator via the Group Members page.
62. A group administrator shall be able to edit the announcement on their group's home page.

What is Unique about our Project:

Gatormmunity distinguishes itself from its competitors like Craigslist and OfferUp by having mandatory identity verification by requiring a picture of one's SFSU ID card in order to register. This makes our platform safer because all registered users will know every other user is a member of SFSU and has been vetted by Gatormmunity's moderators. Also, we have a Forums page that allows members of Gatormmunity to talk to each other and plan events.
Furthermore, users can form groups with others and have their own private group chat and forum. This lets friends or people who share interests talk to each other in private. Gatormmunity also has a live chat (Gator Chat) for all of its members, where users can talk to one another in real time.

Product URL: http://54.241.101.69/

Since new accounts must be approved by a moderator or administrator in order to log in, we provided some credentials to some already approved accounts at the bottom of the Credentials README on GitHub. There is one account for each permission level: one for Approved Users, one for Moderators, and one for Administrators.

3) Milestone Documents (M1 – M4)

SW Engineering CSC648/848 Fall 2022
Gatormmunity
Team 7

Anthony Zhang (Team Lead),
azhang12@mail.sfsu.edu

Marwan Alnounou (Front End Lead)

Mohamed Sharif (Back End Lead)

Jose Lopez (GitHub Master)

Florian Cartozo

Milestone 1
12/12/2022

History Table

Version	Submission Date
M1V1	09/21/2022
M1V2	12/12/2022

Table of Contents

1. Executive Summary

Many are afraid of using Craigslist and other similar sites because of the danger associated with meeting up with a stranger, and because of the risk of being scammed. Hence, the need for Gatormmunity. Gatormmunity aims to create a safe environment where San Francisco State University's (SFSU's) students, alumni, faculty, and staff can exchange goods and services. Gatormmunity is also a tool for connecting with others in the university. SFSU has been criticized for its lacklustre community, and Gatormmunity aspires to revitalize it by providing its members with the tools to connect with their peers.

Gatormmunity will provide a safe environment for SFSU's community to trade and socialize in. Only people who are confirmed to be members of SFSU are allowed to partake in Gatormmunity, so Gatormmunity's users can feel safe knowing that they will not be dealing with a dangerous person. Furthermore, trades can take place on campus, giving further assurance to anyone buying or selling on Gatormmunity. The convenience of doing trades on campus cannot go unstated, as well. Those who were previously reluctant to part with their unneeded belongings due to fear can now start making money with Gatormmunity. To elaborate on the community aspect of Gatormmunity: Gatormmunity will have forums and chat rooms so that its users can make new friends and meet others with similar interests. Again, since every user is a verified SFSU attendee, there will be more honest interactions and less unsavoury behaviour between users.

What is special about Gatormmunity is that it is closed to everyone but SFSU attendees, similar to how Facebook started out. Having a closed community is crucial for Gatormmunity's mission: we are developing an application that makes selling one's belongings safe and accessible, as well as making a place where SFSU attendees can connect with others from the university. By doing away with anonymity, users will be heavily discouraged from being immoral or malicious. Another unique feature of Gatormmunity is that users can form groups, so people with similar interests can more easily talk to each other. The addition of forums and chat rooms is to make Gatormmunity a direct upgrade over existing buying and selling sites because they do not focus on the community aspect of buying and selling within one's community.

2. Main Use Cases

Key Categories of Users/Actors:

Katla (Marketplace Seller):
Characteristics and Skills:

- A student who has a lot of stuff to sell
- Prefers to sell items for cheap so she can offload her huge inventory quickly
- Web skills are not that great

Goals and Pain Points:

- Wants to make some money by selling her ample belongings
- She has had bad experiences on other selling websites because people kept scamming her.

Rachell (Marketplace Buyer, Marketplace Seller, and Approved User):
Characteristics and Skills:

- A compassionate Computer Science alumni
- Prefers to use their phone for browsing the web
- Great with technology

Goals and Pain Points:

- Wants to make some side income
- Wants to buy a new phone from an SFSU student rather than a store
- She could not find a good way to advertise her services to only SFSU students.
- She keeps getting spam emails and harassment whenever she advertises her services on other sites.

Jane (Marketplace Buyer):
Characteristics and Skills:

- A stingy and risk-averse student
- Can find her way around most websites without difficulty

Goals and Pain Points:

- Wants to buy a laptop for cheap
- She is afraid of meeting up with potentially dangerous people, yet she really wants to save money by buying locally.
- She hates websites that make it hard for her to find what she is looking for. She loves websites that have a search and filter functionality.

Haruhi (Group Administrator):
Characteristics and Skills:

- An ambitious student who is president of the chess club
- Excited to meet new potential chess club members
- An expert with websites

Goals and Pain Points:

- Wants to expand her chess club and make it more active
- She wants to promote her chess club but people keep taking down her flyers that she places around campus.

Victor (Group Member and Approved User):
Characteristics and Skills:

- A student who loves chess
- Unfamiliar with Macs, but had to buy a Mac laptop for his classes
- Not great at navigating websites

Goals and Pain Points:

- Wants to play chess with their club members
- Wants to get help with opening a file on his Mac laptop
- He needs urgent help with his laptop but finding someone online to help him would take too long, and he wants someone else to fix his problem for him because he is not good with Macs.

Ficus (Approved User)
Characteristics and Skills:

- A student on the verge of failing their Computer Science classes
- Would never cheat to pass their classes
- Not great at navigating websites

Goals and Pain Points:

- Wants to find someone to help him understand his course material
- When he tried to find people to help him online, they just gave him the solutions to his questions without explanations, which is not what he wanted. He wanted help understanding the material.

Judie (Approved User):
Characteristics and Skills:

- A senior student in Mechanical Engineering
- Does not have much money
- Loves trains
- High GPA with good technical and social skills
  Goals and Pain Points:
- Wants to network and find a job
- When she uploaded her resume online, she received a lot of spam calls and fake job offers.

Keith (Approved User):
Characteristics and Skills:

- A Civil Engineering professor who likes to help their students
- Used to work as a Senior Civil Engineer and knows a lot of people
- Good with technology

Goals and Pain Points:

- Wants to message all of their students at once
- Canvas and iLearn only lets him message all of his students one class at a time.

Anna (Approved User):
Characteristics and Skills:

- A Russian student who loves to have many friends
- Is always on social media, including Gatormmunity
- Familiar with Apple devices, especially their laptops

Goals and Pain Points:

- Wants to make even more friends
- She likes to talk in chat rooms and make posts in forums, but she does not feel comfortable knowing the whole world could be reading her messages.

Vladislav (Approved User):
Characteristics and Skills:

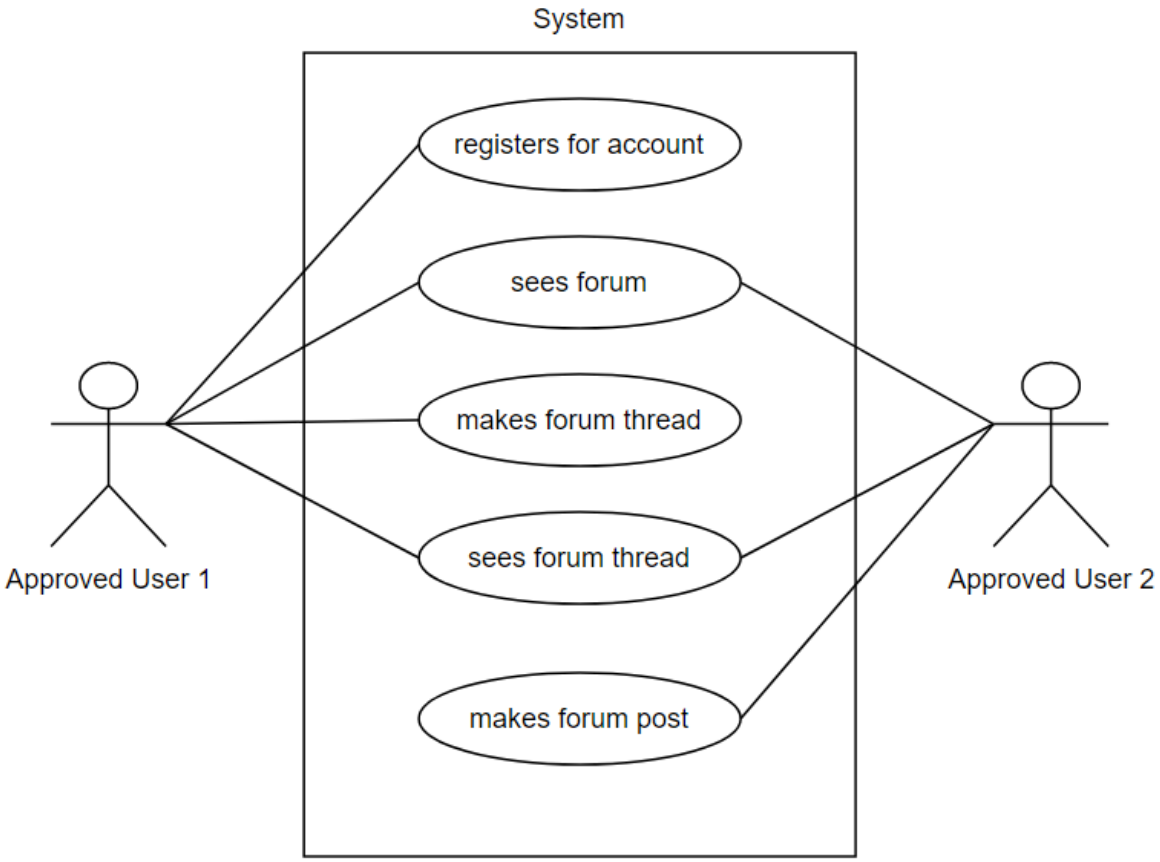
- A new Russian student at SFSU who is lonely because they have no friends yet
- Understanding
- A chess fanatic
- Can only use simple websites
- A fluent reader in English, but not a fluent speaker

Goals and Pain Points:

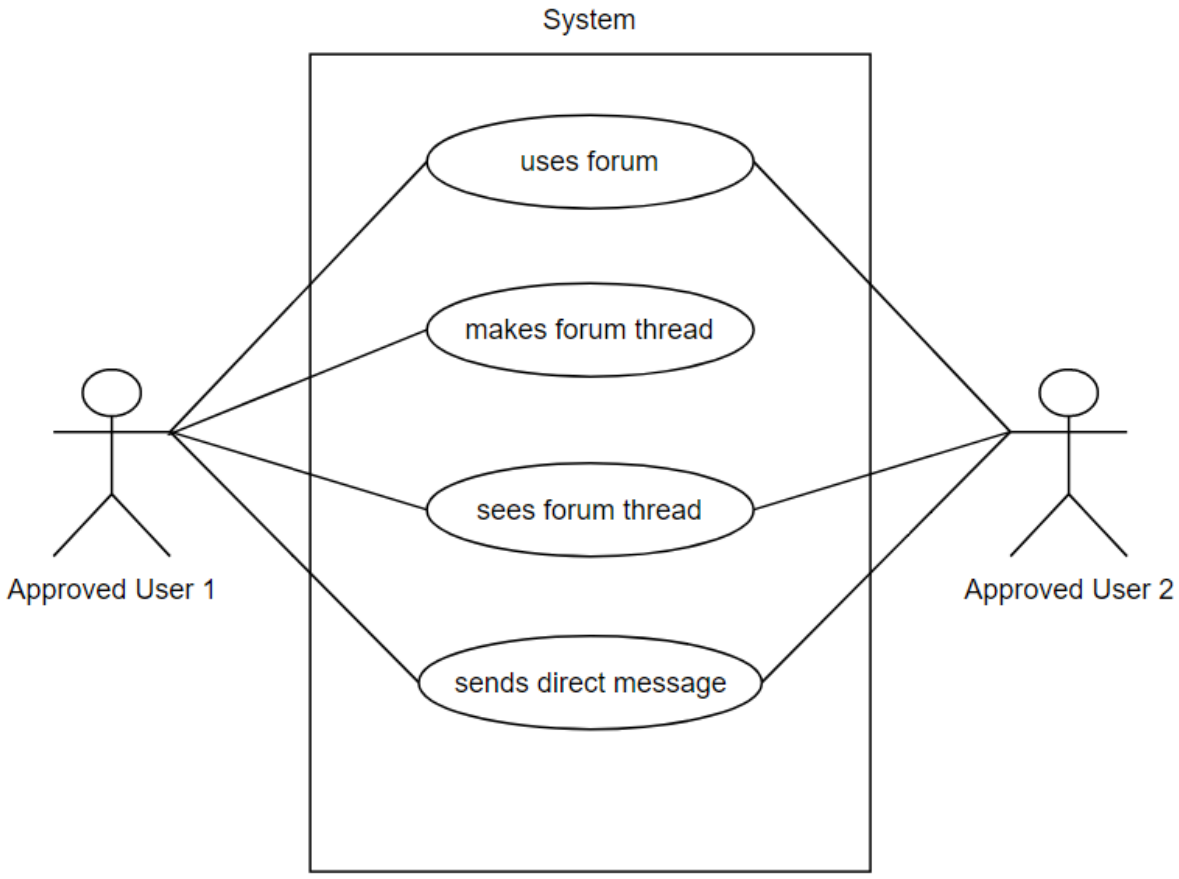
- Wants to make friends and meet fellow Russians at SFSU
- He does not know where the Russian SFSU students are and could not find any information online to help him with this very specific problem.

Main Use Cases:

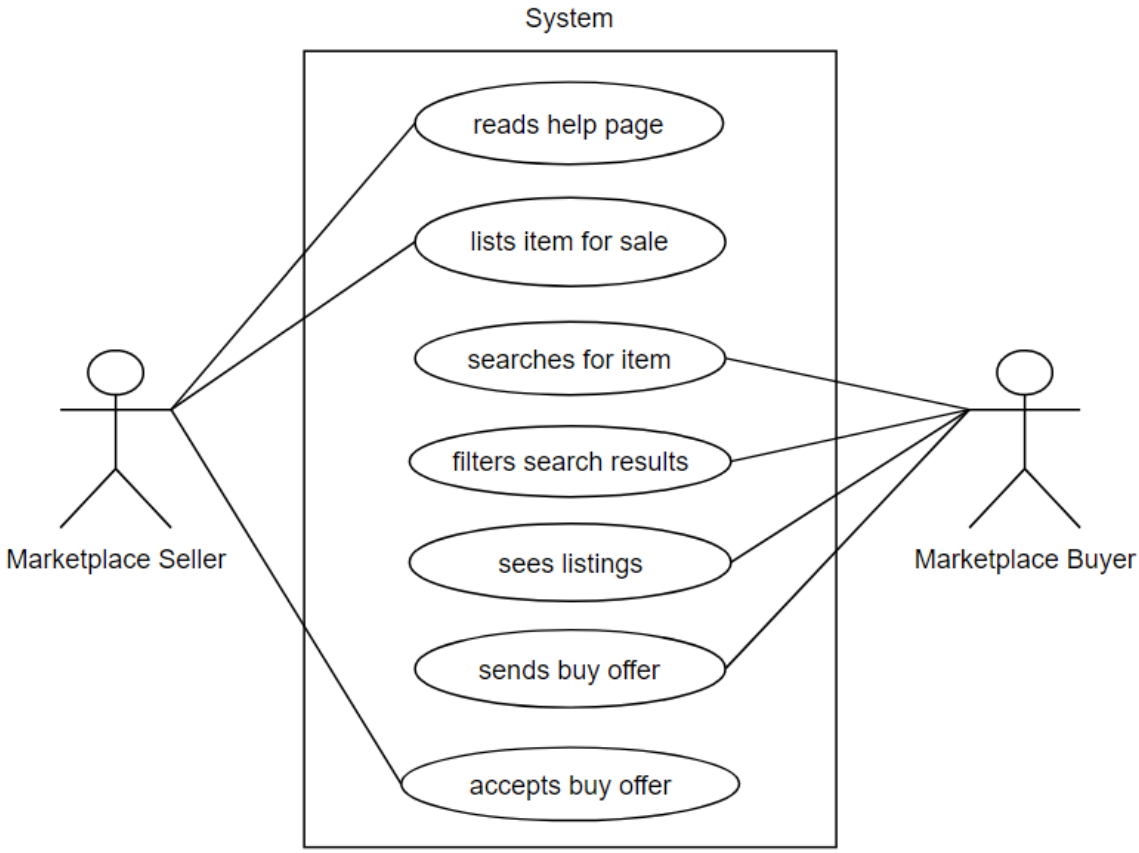
1. Use Case: Russian Friends
Actors: Vladislav (Approved User 1), Anna (Approved User 2)
Description: Vladislav is a new student from Russia, so he has no friends and is lonely. He wants to befriend Russians, but he does not know how to find other Russians in SFSU. Luckily, he spots a Gatormmunity ad on campus, so he goes to make an account. Unfortunately, he does not yet have the SFSU identification required to register. He is disappointed but he understands why this requirement exists. He waits until his SFSU identification arrives and finally registers. Vladislav easily finds the Gatormmunity forum and just as easily makes a forum thread saying that he is looking for Russian friends. Anna, who is Russian, sees his thread and offers her friendship in her reply post. Vladislav is elated and becomes fast friends with her. Now, Vladislav is not so lonely anymore, while Anna, who is already popular, has just become even more popular.



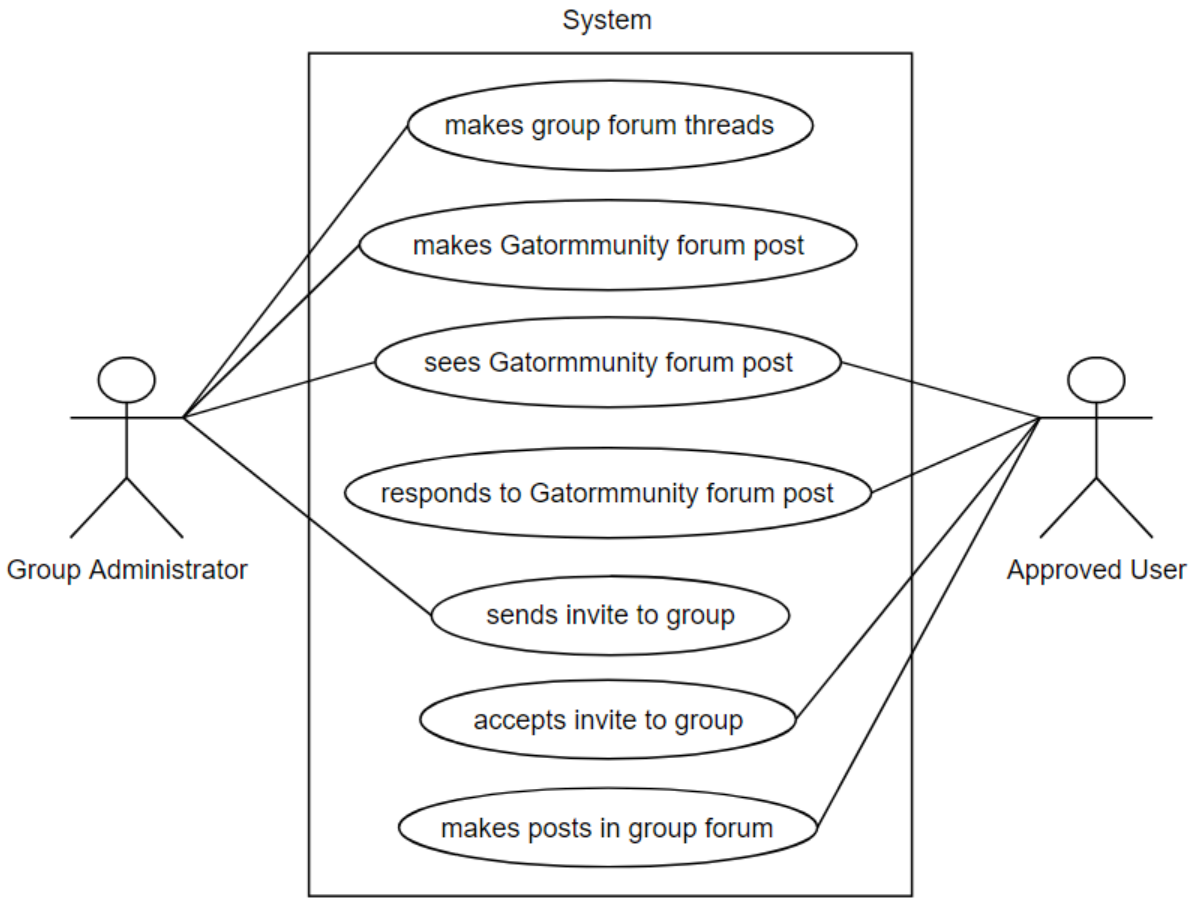
2. Use Case: Struggling Student
Actors: Ficus (Approved User 1), Rachell (Approved User 2)
Description: Ficus is sad because he needs help understanding the material in his computer science class, and he is not able to attend office hours with his professor nor does SFSU's tutoring have anyone that can help him with the topic he is struggling with. Ficus does not want to spend money on paid tutoring, so he uses Gatormmunity's free forum and makes a thread stating his circumstances. Rachell, a Computer Science alumni, sees his thread and pities Ficus, so she messages him, offering him a tutoring session free of charge. Ficus happily accepts, so Rachell tutors him over the internet. Ficus finally understands the material and is hopeful that he will pass his classes now. Rachell is thankful that she could help an SFSU student.



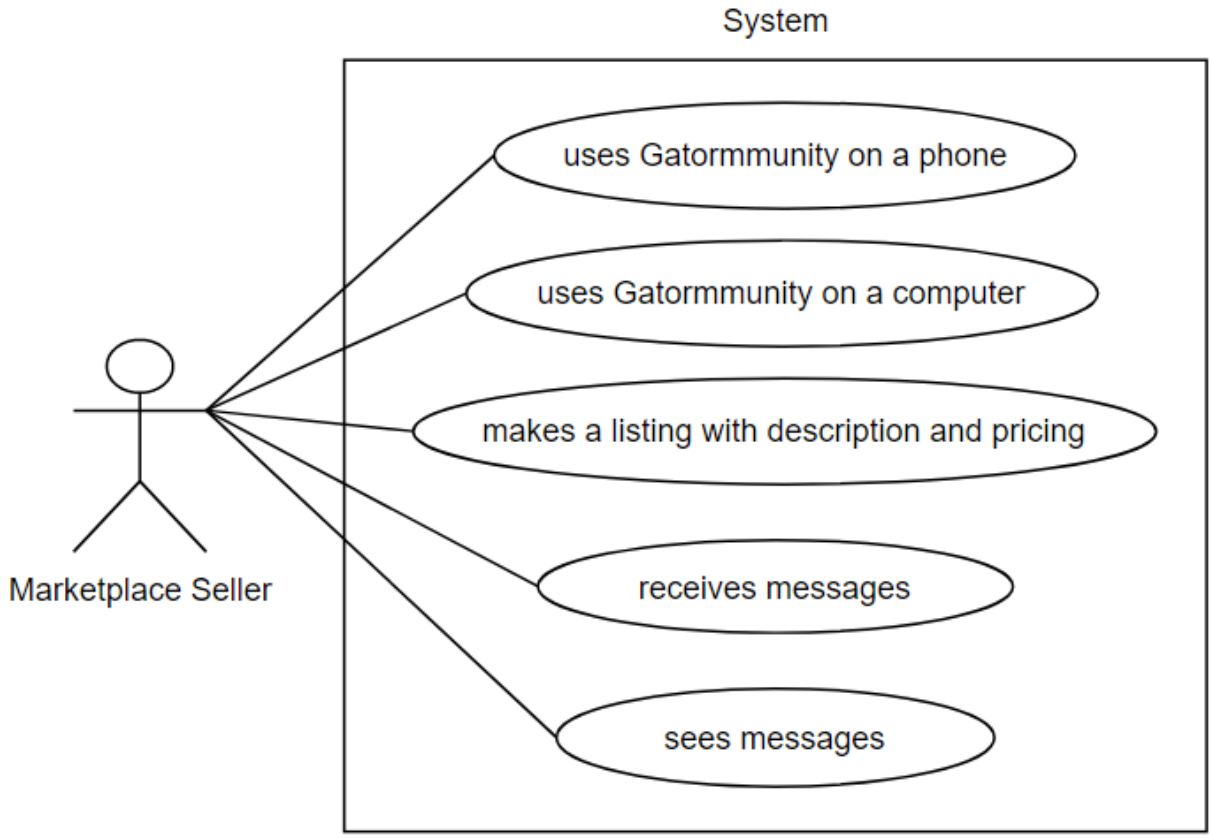
3. Use Case: Wary Traders
Actors: Katla (Marketplace Seller), Jane (Marketplace Buyer)
Description: Katla wants to make some money by selling stuff, but she is hesitant to sell online because people keep scamming her. When she heard about Gatormmunity she was elated because every user is a verified SFSU attendee and would not risk doing anything malicious. Katla initially could not figure out how to sell things on Gatormmunity, but after reading its help page, she was able to list her old laptop for sale. Jane, an SFSU student, wants to buy a laptop for cheap, so she uses Gatormmunity instead of its competitors because she does not want to risk getting robbed when meeting the seller. She searches for a laptop in the marketplace and filters the results so that she only sees listings that fit her budget. Only Katla's listing remained, so Jane sends her an offer and they agree to make the trade on campus since they both feel safe there. Jane is happy she got an inexpensive laptop, while Katla is happy she made some money.



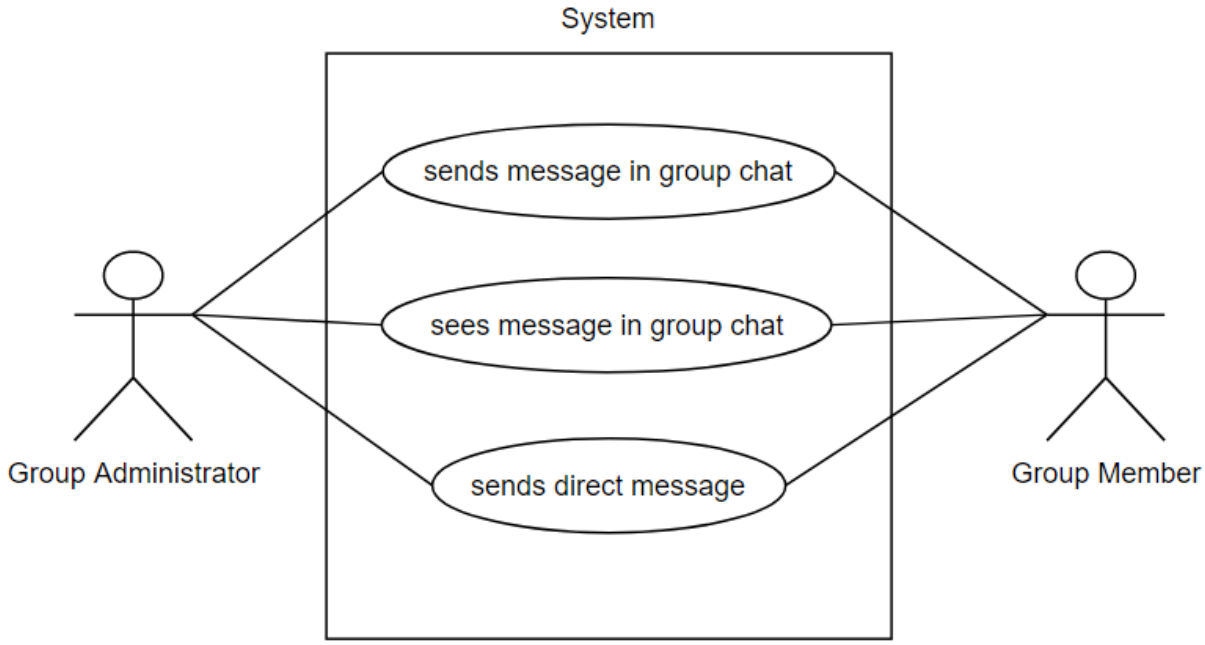
4. Use Case: Inactive Chess Club
Actors: Haruhi (Group Administrator), Vladislav (Approved User)
Description: Haruhi is trying to make her Gatormmunity chess group more active but is not having much luck despite making weekly forum threads encouraging discussion in her group's forum. Haruhi figured she just needed more members so she made a Gatormmunity forum post. Vladislav, a new student who wants to make friends, saw Haruhi's post and asked to join because he loves chess. Haruhi gladly accepted and invited Vladislav to the chess group. Vladislav immediately started posting in the group's forum because he has a lot to talk about when it comes to chess. Vladislav is glad to have found people with similar interests, while Haruhi is pleased her chess group has become more active.



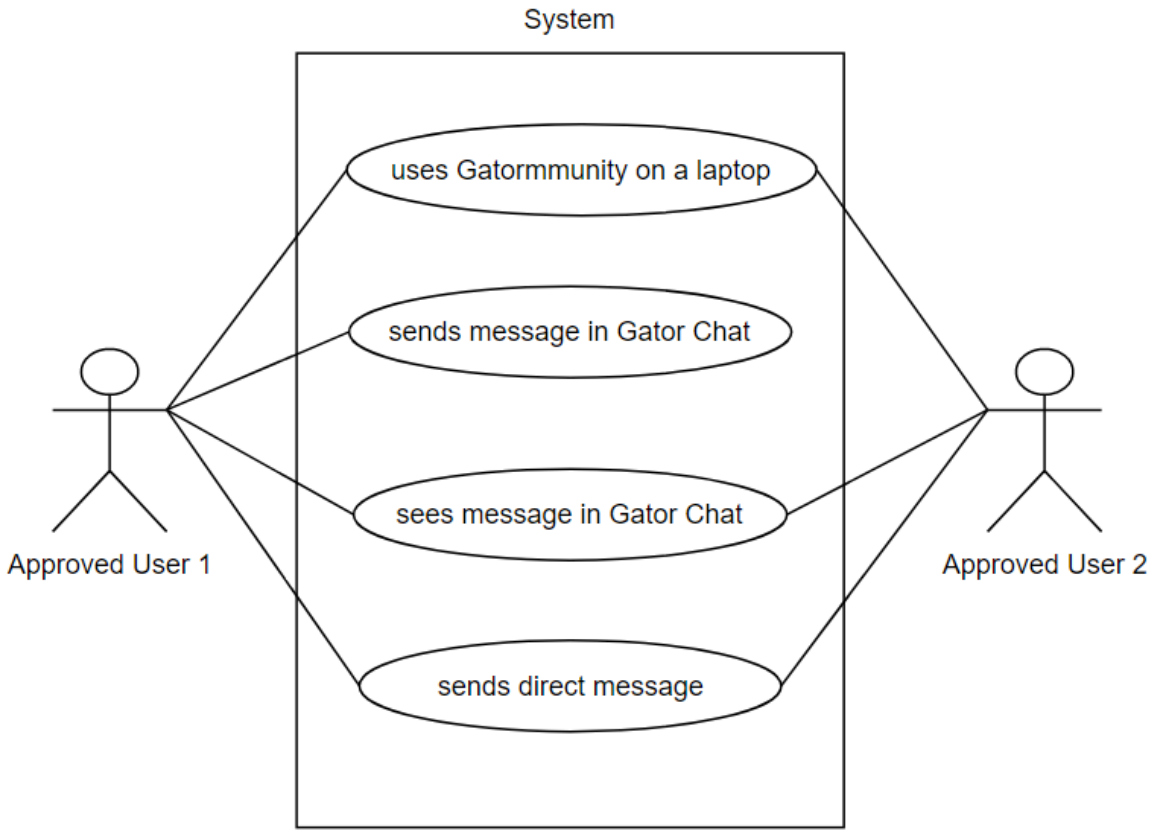
5. Use Case: Tired of Spam
Actors: Rachell (Marketplace Seller)
Description: Rachell is a Computer Science graduate who wants to make some cash by tutoring struggling computer science students. She has tried Gatormmunity's competitors before but it always resulted in her email being filled with spam and harassment. She did not try Gatormmunity first because it has a smaller user base, but the time has finally come. She was disappointed with how bad Gatormmunity looked on her phone, but she really wanted to help people from her alma mater. She bit the bullet and used her computer to make a listing describing her services and pricing on the marketplace. Soon, she was inundated with messages from students who wanted to pay for her services, and received no spam at all. Rachell loved seeing all of these potential clients and not a drop of spam or offensive messages.



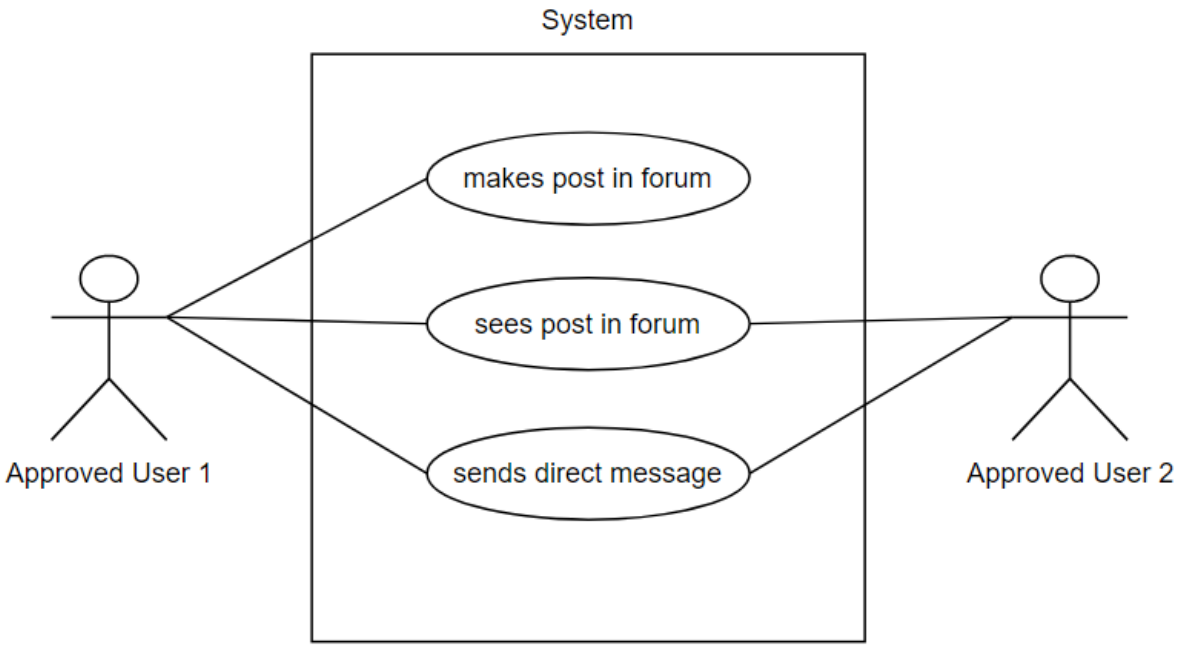
6. Use Case: A Game of Chess
Actors: Haruhi (Group Administrator), Victor (Group Member)
Description: Victor is bored and wants to play some online chess with his Gatormmunity chess group, so he sends a message in their group's chat asking if anybody wants to play. The group chat is a chat room where only members of the group can send messages. He did not ask in the group forum because he wanted an immediate response from whomever is online. Haruhi happened to be online and saw Victor's request for a game, so she gladly accepted by responding to his offer in the group chat. After the game, they felt like reviewing the game they just played. They messaged each other to talk about it rather than use the group chat because they did not want everyone in the group to read the conversation about their private chess game. Victor is not bored anymore; Haruhi is happy to have bonded with her group members.



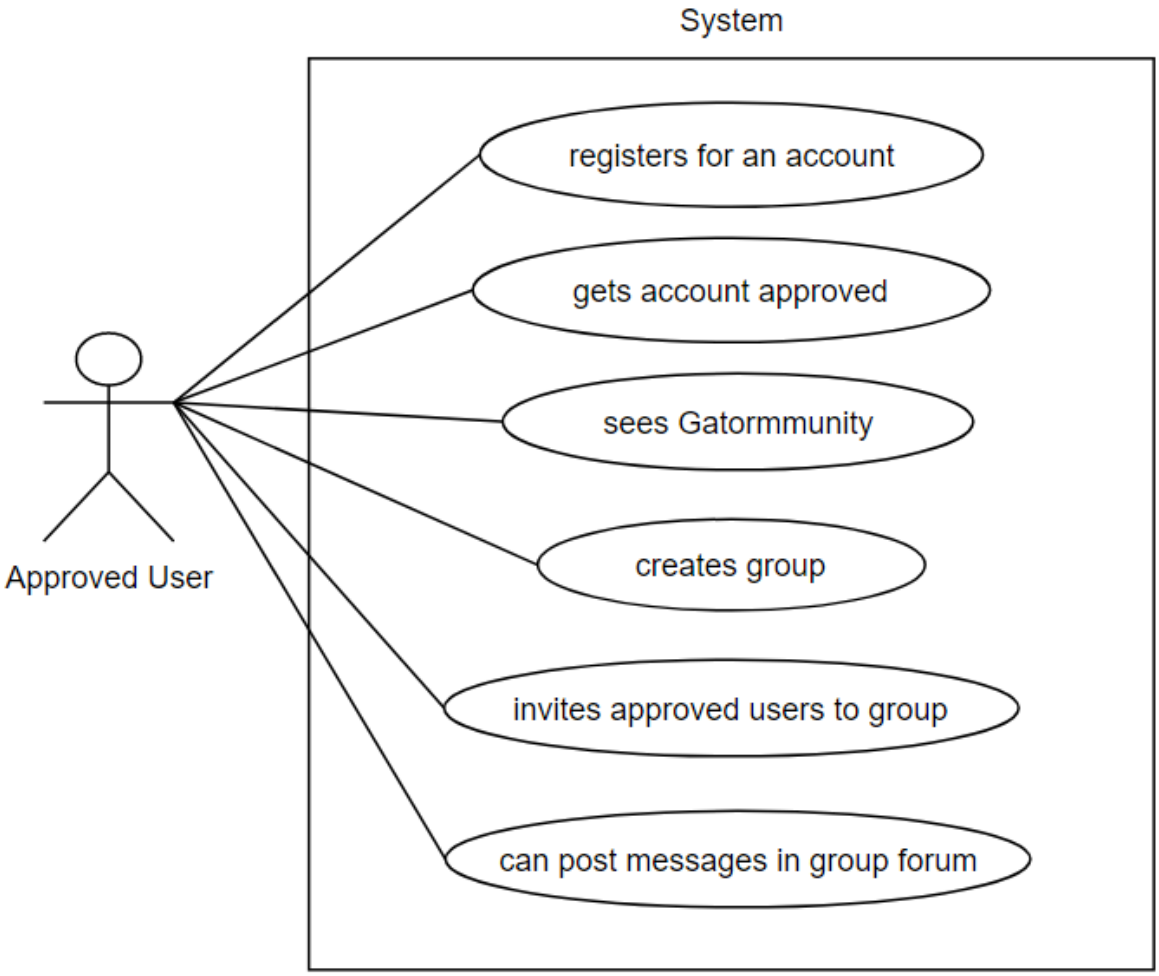
7. Use Case: Powerpoint Problems
Actors: Victor (Approved User 1), Anna (Approved User 2)
Description: Victor is on campus and he needs to open his powerpoint file for his class in an hour, but his Mac laptop refuses to open it. Victor starts to panic but remembers that he is a proud member of Gatormmunity. Even though he is not good with Macs, he at least knows how to use the web browser. He goes to Gatormmunity, and asks in Gator Chat if anyone is on campus that can help him with his dilemma. Anna, who is always on Gatormmunity via her laptop, sees the call for help and messages Victor saying she is on campus and can help him. They met up and she solved Victor's problem immediately owing to her expertise with Mac laptops. Victor is jubilant and promises to pay Anna back one day. Anna is satisfied enough to have made another friend.



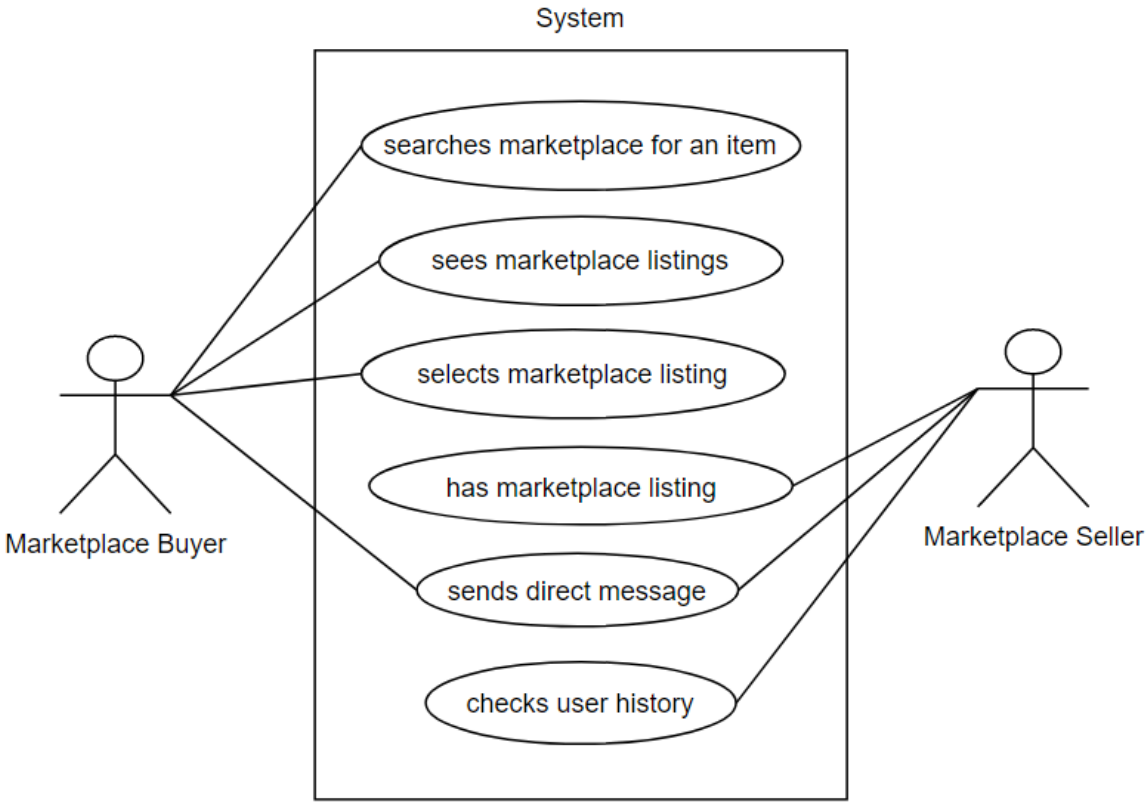
8. Use Case: An Aimless Student
Actors: Judie (Approved User 1), Keith (Approved User 2)
Description: Judie is in her last semester in Mechanical Engineering and is a hard worker with good grades, but she does not know how to apply for a job, nor has she thought about which company she wants to work for. All she knows is that she wants to work as soon as possible after graduation. Thus, she turned to Gatormmunity for help. She posted in the forum saying that she is looking for a job and listed her competencies and grades. Keith, a civil engineering professor and former senior engineer, saw her post and was impressed with Judie's skills. He sent her a direct message and told her he can get her an interview at BART since his friend works as the hiring manager there. Judie loves BART and is flabbergasted by Keith's offer. Naturally, she accepts. Keith is glad to have helped an SFSU student, and Judie is truly grateful for having chosen to post on Gatormmunity.



9. Use Case: A Professor's Pains
Actors: Keith (Approved User)
Description: Keith is a civil engineering professor who teaches 5 different classes this semester. He wants to make announcements to all of his students in all of his classes at once, instead of having to painstakingly make a separate announcement for each class. A colleague tells him about Gatormmunity and how people can create groups where members can post messages in the forum for everyone in the group to see. Keith signs up right away. After his account was approved, he saw how clean and quick Gatormmunity's pages were and wanted to tell his students all about Gatormmunity. He created a group and invited all of his students to it. Finally, he could post an announcement on a single page for all of his students to see. Thus, Keith's problems were solved and he was a happy man.



10. Use Case: Supporting the Students
Actors: Rachell (Marketplace Buyer), Katla (Marketplace Seller)
Description: Rachell needs a new phone but she does not want to buy it from a store. She wants to help her alma mater out by buying from its students. She knows the best way to buy from SFSU students is through Gatormmunity, so she searches the marketplace for new phones and picks the first one, which happens to be Katla's listing. Rachell messages Katla saying that she would like to buy the phone and have it shipped instead of picking it up in person. Katla checks Rachell's long Gatormmunity history and believes she is trustworthy, therefore she agrees to mail it to her. Katla is jubilant that she can make money, while Rachell is delighted that she got a new phone and helped an SFSU student.



3. List of Main Data Items and Entities

Types of Users and their Privileges:

All Users / Users: This category includes everyone who visits Gatormmunity's website, whether they have an approved account or not.

Guest User: A person that does not have an account. Has no access to almost everything in Gatormmunity.

Unapproved User: A person that registered for an account and has not yet had their account approved. Has the same privileges as a Guest User.

Approved User: A person that registered for an account and has had their account approved. Has access to almost everything in Gatormmunity, except for moderation tools and group-exclusive features.

Group Member: An approved user who is a member of a group. Has the same privileges as an Approved User, and has access to group-exclusive features.

Group Moderator / Group Mod: An approved user who is a moderator of a group. Has more privileges than a group member, including access to group moderation tools.

Group Administrator / Group Admin: An approved user who is an administrator of a group. Has more privileges than a group moderator, and has access to group administrative tools, including the management of group moderators.

Moderator / Mod: An approved user that has moderation powers in Gatormmunity. Has more privileges than an approved user, including access to moderation tools.

Administrator / Admin: An approved user who has access to Gatormmunity's server, database, and GitHub repository. Has more privileges than a moderator, including the management of moderators.

Main Terms:

Gatormmunity Forum / Forum: A forum where all approved users can start their own forum threads and make posts in existing threads.

Forum Category: The forum will have categories so that approved users can find others with similar interests.

Forum Thread / Thread: A collection of posts. Threads have a subject which is determined by the thread starter. Forum threads are comparable to discussion topics on iLearn, which other students can reply to.

Forum Post / Post: A message that approved users can post in a thread. The first post that starts a thread is also considered a post, and is referred to as the “original post”. Forum posts are comparable to the replies in a discussion topic on iLearn, including the post that started the discussion topic.

Pin: A pinned forum thread will always appear first in a list of threads.

Bookmark: Approved users can bookmark forum threads, which saves the thread into the user’s bookmarks. Bookmarks exist to help the user easily find threads that they want to go back to in the future.

Gatormmunity Marketplace: A place within the application where approved users can buy and sell goods and services.

Marketplace Buyer / Buyer: An approved user who is buying something from Gatormmunity’s marketplace.

Marketplace Seller / Seller: An approved user who is selling something on Gatormmunity’s marketplace.

Marketplace Listing / Listing: An item or service a seller is trying to sell on the marketplace. It can have photos, a description, a title, a price, and so on. The listing will not need to be approved before being listed, but approved users can report listings and moderators can delete listings that break the rules.

Gator Chat: A chat room for all approved users of Gatormmunity. Approved users can send messages and receive messages from other approved users in the chat room.

Direct Message / DM: A private method of communication between two approved users.

User Groups / Groups: Approved users can form their own groups which come with exclusive features such as a group-exclusive chat and forum.

Group Chat: A chat room for all members of a group. All group members can send messages and receive messages from other group members in the group chat room.

4. Initial List of Functional Requirements

1. All Users
    1.1. All users shall be able to donate money to the developers via PayPal. PayPal has a donate button that prompts the user to specify how much they want to donate, and that donation shall go to the developers.
    1.2. All users shall be able to donate money to the developers via Venmo. The developers have a Venmo username that the user can direct their Venmo donation to.
    1.3. All users shall be able to contact the developers.
    1.4. All users shall be able to view the home page.
    1.5. All users shall be able to view the registration page.
    1.6. All users shall be able to view the login page.
    1.7. All users shall be able to view the about page to learn about the application.

2. Guest User
    2.1. A guest user shall be able to register for an account.

3. Approved User
    3.1. An approved user shall be able to log in to their account.
    3.2. An approved user shall be able to reset their password.
    3.3. An approved user shall be able to change their password.
    3.4. An approved user shall be able to delete their account.
    3.5. An approved user shall be able to make a forum thread.
    3.6. An approved user shall be able to make a forum post.
    3.7. An approved user shall be able to attach images to their forum posts.
    3.8. An approved user shall be able to attach images to their chat messages.
    3.9. An approved user shall be able to sort forum threads.
    3.10. An approved user shall be able to bookmark forum threads.
    3.11. An approved user shall be able to like forum posts.
    3.12. An approved user shall be able to report forum posts to a moderator.
    3.13. An approved user shall be able to report other users to a moderator.
    3.14. An approved user shall be able to block other users.
    3.15. An approved user shall have a profile page.
    3.16. An approved user shall be able to edit their own profile page.
    3.17. An approved user shall be able to view the profile page of another approved user.
    3.18. An approved user shall have a profile picture.
    3.19. An approved user shall be able to change their profile picture.
    3.20. An approved user shall be able to create a user group.
    3.21. An approved user shall be able to join a user group.
    3.22. An approved user shall be able to send direct messages to other approved users.
    3.23. An approved user shall be able to receive direct messages from other approved users.
    3.24. An approved user shall be able to send messages in Gator Chat.
    3.25. An approved user shall be able to see marketplace listings in the marketplace.

3.26. An approved user shall be able to select marketplace listings in the marketplace.
3.27. An approved user shall be able to report marketplace listings to a moderator.
3.28. An approved user shall be able to compare different marketplace listings against each other.
3.29. An approved user shall be able to add items to a wishlist.
3.30. An approved user shall be able to buy items from the marketplace.
3.31. An approved user shall be able to sell items on the marketplace.
3.32. An approved user shall be able to edit the details of an item they are selling.
3.33. An approved user shall be able to upload pictures of an item they are selling.
3.34. An approved user who is selling an item shall be able to list their accepted payment methods.
3.35. An approved user who is selling an item shall be able to list their possible delivery methods.
3.36. An approved user who is selling an item shall be able to list their preferred contact methods.
3.37. An approved user who is selling an item shall be able to assign their listing to a category.
3.38. An approved user shall be able to leave feedback about the seller they purchased an item from.
3.39. An approved user shall be able to search for users.
3.40. An approved user shall be able to search for marketplace listings.
3.41. An approved user shall be able to search for forum posts.
3.42. An approved user shall be able to apply filters to their user search results.
3.43. An approved user shall be able to apply filters to their marketplace search results.
3.44. An approved user shall be able to apply filters to their forum post search results.

4. Moderator
   4.1. A moderator shall be able to approve unapproved users.
   4.2. A moderator shall be able to reject unapproved users.
   4.3. A moderator shall be able to ban approved users from Gatormmunity.
   4.4. A moderator shall be able to delete forum threads.
   4.5. A moderator shall be able to delete forum posts.
   4.6. A moderator shall be able to pin forum threads.
   4.7. A moderator shall be able to delete messages in Gator Chat.
   4.8. A moderator shall be able to delete listings in the marketplace.

5. Administrator
   5.1. An administrator shall be able to appoint moderators.
   5.2. An administrator shall be able to unappoint moderators.
   5.3. An administrator shall be able to ban moderators from Gatormmunity.
   5.4. An administrator shall be able to create new categories in the forum.
   5.5. An administrator shall be able to delete categories in the forum.

6. Group Member
    6.1. A group member shall be able to invite other approved users to their group.
    6.2. A group member shall be able to leave their group.
    6.3. A group member shall be able to create forum threads in their group's forum.
    6.4. A group member shall be able to create forum posts in their group's forum.
    6.5. A group member shall be able to send messages in their group's chat.

7. Group Moderator
    7.1. A group moderator shall be able to kick group members from their group.
    7.2. A group moderator shall be able to delete forum threads in their group's forum.
    7.3. A group moderator shall be able to delete forum posts in their group's forum.
    7.4. A group moderator shall be able to pin forum threads in their group's forum.
    7.5. A group moderator shall be able to delete messages in their group's chat.

8. Group Administrator
    8.1. A group administrator shall be able to appoint group moderators.
    8.2. A group administrator shall be able to unappoint group moderators.
    8.3. A group administrator shall be able to kick group moderators from the group.
    8.4. A group administrator shall be able to resign their position as group administrator and give the position to another member of the group.
    8.5. A group administrator shall be able to delete their group.
    8.6. A group administrator shall be able to add a description about their group.
    8.7. A group administrator shall be able to edit their group's description.
    8.8. A group administrator shall be able to create new categories in their group's forum.
    8.9. A group administrator shall be able to delete categories in their group's forum.

5. List of Non-Functional Requirements

1. System Requirements
    1.1. The AWS server's region shall be North California, United States.
    1.2. The database shall be stored on the AWS server.

2. Performance Requirements
    2.1. The application should be available for at least 23 hours a day.
    2.2. The application's homepage should load in at least 7 seconds.
    2.3. The application should be accessible from anywhere in the world.

3. Storage, Security, and Environmental Requirements
    3.1. The server's storage shall be a 30 GiB volume.
    3.2. The database shall be secured with a password.
    3.3. Passwords shall be hashed before being stored in the database. In other words, there shall be no plaintext passwords stored in the database.

4. Usability Requirements
    4.1. Guest users shall accept the privacy policy and terms of service before they can register for an account.
    4.2. Approved users should abide by the terms of service.

5. Marketing and Legal Requirements
    5.1. The application shall be known as Gatormmunity in its marketing.
    5.2. The application shall be known as Gatormmunity in its licensing.
    5.3. The application shall have a logo.
    5.4. The application shall have a privacy policy.
    5.5. The application shall have terms of service.

6. Content Requirements
    6.1. The application shall support the English language.
    6.2. The application shall support US Dollars as currency.
    6.3. Approved users shall only upload files less than or equal to 5 MB in size.

7. Privacy Requirements
    7.1. The application shall collect and store personal data from its users.
    7.2. The application shall use the personal data of its users for verifying their identity.
    7.3. The application shall store and use its users' credentials for logging them in.
    7.4. The application shall not reveal any of a user's private data to any other user.

8. Compatibility Requirements
    8.1. The application shall support the Windows 10, Windows 11, and macOS Monterey 12.5 and 12.6 operating systems.
    8.2. The application shall support the Google Chrome 105, Microsoft Edge 105, and Safari 15.5 web browsers.
    8.3. The application should not be designed for mobile devices.
    8.4. The application should be designed for and work on laptops.
    8.5. The application should be designed for and work on desktop computers.

9. Organizational Requirements
    9.1. The AWS server should not incur any costs to the developers.
    9.2. The developers should use semicolons in their JavaScript code.
    9.3. The developers should use tabs in their code, and not spaces.
    9.4. The developers should not have lines of code that exceed 130 characters in length.
    9.5. The developers should document their code with comments.
    9.6. The application's code should not be messy.
    9.7. The application shall use working and tested code. In other words, faulty and untested code shall not be served from the remote server.
    9.8. The master branch of the application's GitHub repository shall contain only working and tested code.
    9.9. The documentation and technical reports shall have a table of contents.
    9.10. The documentation and technical reports shall have page numbers.
    9.11. The documentation and technical reports shall start each section on a new page.

6. Competitive Analysis

Gatormmunity's Competitors:

1. VarageSale (www.varagesale.com)
2. Craigslist (www.craigslist.org)
3. OfferUp (www.offerup.com)
4. eBay (www.ebay.com)
5. 5miles (www.5miles.com)

Important Features

Feature/ Company	VarageSale	Craigslist	OfferUp	eBay	5miles
Strengths	Emphasis on local community, has an item reservation feature, simple website	Many item categories, worldwide availability, quick and easy to use	Easy to use, simple design, has an app, services the entire US	Large variety of items, Has large user base, has buyer/seller protection	Easy to use, has a detailed FAQ and help page, has an app
Weaknesses	Trading limited to local community, has reputation of admin abuse	Many spammers and scams, hard to get listings noticed, overwhelming UI	Steep learning curve, misleading advertising, hard to find some information	Users cannot sell services, has scammers	Messy front page, website has limited functionality, full of scammers
Pricing	Entirely free, no fees on sales	Has fees on certain services ranging from $3 to $75	Sales fee of 12.9%, premium option adds 7.9% sales fee	Sales fee of 12.9%	Sales fee of 10%
Social Media	Facebook, Twitter	Facebook	Facebook, Twitter, Pinterest, YouTube promoters	Facebook, Twitter, TikTok	Blog posts, Facebook, Instagram, Twitter
Onboarding Experience	Easy and simple to use, has a help page	Easy to sign up and sell items, has a help and FAQ page	Hard to learn, some things are not intuitive, lack of instructions	Easy to use, intuitive steps, has a help page	Easy to learn, intuitive steps, has a help page

Competitive Features

Feature	VarageSale	Craigslist	OfferUp	eBay	5miles	Gatormmunity
Community Forum	-	+	-	+	+	+
Website-wide General Chat	-	-	-	-	-	++
Group Communication	-	-	-	-	-	++
Direct Messaging	+	+	+	+	+	+
User Safety Measures	+	-	+	+	-	++
Scam Listing Prevention	+	-	-	-	-	+

Legend:
- Feature does not exist
+ Feature exists
++ Feature is superior

Research Summary:

Gatormmunity fills a niche that no other competitor fills: it is a marketplace and social networking service for the SFSU community. Gatormmunity's main advantage is that it better ensures user safety. Every user will be a SFSU member, and there will be no anonymity, so users would not do anything that would get them reported to the university or the police. Furthermore, Gatormmunity's content will only be visible to its members, which stops outsiders from snooping on them. Unlike its competitors, Gatormmunity will have a website-wide general chat, meaning there will be a chat room where every member can send messages that will be seen by all members. There will also be group communication: users can create user groups and send messages or make forum posts that are only visible to the group's members. Gatormmunity's competitors have no such comparable feature. The community forum and direct messaging features will be just as good as its competitors' versions, though it will look cleaner. Finally, Gatormmunity has scam listing prevention because its users cannot create an anonymous account and post scams without consequence.

7. High-level System Architecture and Technologies Used

System Architecture and Technologies:

- Server Host: Amazon AWS t2.micro 1vCPU 1 GiB RAM
- Operating System: Ubuntu Server 22.04 LTS (HVM)
- Database: MySQL 8.0.30
- Web Server: NGINX 1.23.1
- Server-Side Language: JavaScript

Additional Technologies:

- JavaScript Runtime Environment: Node.js 18.8.0
- Back End Framework: Express.js 4.18.1
- Front End Library: React.js 18.2.0
- IDE: Visual Studio Code 1.71
- MySQL Workbench 8.0.30

Supported Web Browsers:

- Google Chrome 105
- Microsoft Edge 105
- Safari 15.5

8. Checklist

- Team found a time slot to meet outside of the class
  DONE

- GitHub master chosen
  DONE

- Team decided and agreed together on using the listed SW tools and deployment server
  DONE

- 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

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

- GitHub organized as discussed in class (e.g. master branch, development branch, folder for milestone documents etc.)
  DONE

9. List of Team Contributions

List of Detailed Contributions Made by Each Team Member:

Anthony Zhang:

1. Assigned tasks with internal deadlines to the team
2. Sent email to Ortiz for tech stack approval
3. Set up GitHub for team SW development
4. Added server credentials to GitHub
5. Brainstormed ideas for the project
6. Launched AWS instance and installed some of our tech stack onto it
7. Uploaded team home page
8. Created template for the About Me page for the team
9. Uploaded his About Me page
10. Was editor for Checkpoint 1
11. Did Part 0 of Checkpoint 1: Title Page
12. Proposed ideas for the executive summary
13. Did Part 1 of Checkpoint 1: Executive Summary
14. Contributed to and did the editing for the use case actors
15. Contributed to 1 use case and did the editing for 9 other use cases
16. Made 10 use case diagrams
17. Did Part 3 of Checkpoint 1: List of Main Data Items and Entities
18. Contributed to the list of functional requirements
19. Contributed to the list of non-functional requirements
20. Researched the competitor 5miles
21. Contributed to the competitive analysis and wrote the research summary
22. Did Part 8 of Checkpoint 1: Checklist
23. Did Part 9.1 of Checkpoint 1: List of Team Contributions
24. Did Part 9.2 of Checkpoint 1: Email Ortiz
25. Active in Discord group and meetings, and facilitated discussion

Marwan Alnounou:

1. Brainstormed ideas for the project
2. Uploaded his About Me page
3. Proposed ideas for the executive summary
4. Made 3 use cases
5. Contributed to the list of functional requirements
6. Contributed to the list of non-functional requirements
7. Researched the competitor Craigslist
8. Contributed to the competitive analysis
9. Did Part 9.2 of Checkpoint 1: Email Ortiz

Mohamed Sharif:

1. Brainstormed ideas for the project
2. Installed and configured the tech stack on our AWS server
3. Uploaded his About Me page
4. Proposed ideas for the executive summary
5. Contributed to the use case actors
6. Contributed to the list of functional requirements
7. Contributed to the list of non-functional requirements
8. Researched the competitor OfferUp
9. Contributed to the competitive analysis
10. Did Part 7 of Checkpoint 1: High-level System Architecture and Technologies Used
11. Did Part 9.2 of Checkpoint 1: Email Ortiz
12. Active in Discord group and meetings, and facilitated discussion

Jose Lopez:

1. Uploaded his About Me page
2. Contributed to the use case actors
3. Made 3 use cases, and contributed to 1 other use case
4. Contributed to the list of functional requirements
5. Contributed to the list of non-functional requirements
6. Researched the competitor VarageSale
7. Contributed to the competitive analysis
8. Did Part 9.2 of Checkpoint 1: Email Ortiz
9. Active in meetings and facilitated discussion

Florian Cartozo:

1. Uploaded his About Me page
2. Contributed to the use case actors
3. Made 3 use cases
4. Contributed to the list of functional requirements
5. Contributed to the list of non-functional requirements
6. Researched the competitor eBay
7. Contributed to the competitive analysis

Contribution Scores:

Anthony Zhang: 9
Marwan Alnounou: 6

Mohamed Sharif: 9 *(I learned about Nginx and had React running on Nginx. I helped Marwan with Nginx. I Researched a lot on Offer Up. I gave the correct technical use of components and destructors for the homepage for a cleaner code. I contributed my opinion to the non functional and functional. I participated in every meeting and spoke in every meeting. I turned in all the work on time. I installed node and made updates on the server OS. I consulted with the team lead every time I submitted something and asked for feedback. I gave constructive feedback during meetings. I asked the team leader if he needed any help. In regards to the meetings, I was never told that it would be a problem to be on zoom while I'm picking my daughter from school 4 blocks away from my apartment. I was still able to actively participate.)*

Jose Lopez: 7.5
Florian Cartozo: 5

SW Engineering CSC648/848 Fall 2022
Gatormmunity

Team 7: Team 7

Anthony Zhang (Team Lead),
azhang12@mail.sfsu.edu

Marwan Alnounou

Mohamed Sharif

Jose Lopez

Florian Cartozo

Milestone 2
12/12/2022

History Table

Version	Submission Date
M2V1	10/20/2022
M2V2	12/12/2022

Table of Contents

1. Data Definitions

Types of Users:

All Users: Any person who visits Gatormmunity's website, whether they are logged in or not.

- All Users can see the home page, login page, and register page, but what they can do on each of these pages depends on whether they are logged in or not.

Guest User: A person that does not have an account.

- Guest Users can only see the pages that do not require one to be logged in for, e.g. the home page, registration page, login page, and about us page.
- Guest Users can register for an account by providing a full name, SFSU email address, SFSU ID number, SFSU ID picture, and password.
    - The picture must be of an image format. Supported formats are: JPEG, PNG, WebP, GIF, and AVIF. The picture must be at most 5 MB in size.

Unapproved User: A person that registered for an account and has not yet had their account approved by a moderator or admin.

- Unapproved Users have the same privileges as Guest Users, meaning they can only see the pages that do not require one to be logged in for.
- Unapproved Users cannot login until their account has been approved by a moderator or admin.
- Unapproved Users have the same attributes as any other user stored in the user table in the database: a first name, last name, email, SFSU ID number, SFSU ID picture, profile picture, role, and join date.
    - For unapproved users, role = 0.

Approved User: A person that registered for an account and has had their account approved by a moderator or admin. All logged in users are approved users, since unapproved users cannot login.

- Approved Users have access to almost every page and feature in our application, except for moderation tools and group-exclusive features. They can create forum threads and posts and create marketplace listings, for example.
- Approved Users can log in to their account by providing their SFSU ID number and password.
- Approved Users have the same attributes as any other user stored in the user table in the database: a first name, last name, email, SFSU ID number, SFSU ID picture, profile picture, role, and join date.
    - For approved users, role = 1.

Moderator / Mod: An approved user that has moderation powers in Gatormmunity.

- Moderators have more privileges than Approved Users, and have access to moderation tools, including the ability to approve Unapproved Users and ban Approved Users from Gatormmunity via the UI.

- The role and its associated privileges are determined by the role attribute stored in the database for each user. To be able to access the moderation tools in the UI, a user needs to have the correct role attribute value in the database. The back end will check the user ID of the user to ensure that they are of the correct role before executing any moderation actions.
- Moderators are appointed and unappointed by Server Administrators using either the Gatormmunity website or through directly modifying a user's role in the database.
- No users will be able to determine their role upon registering like they could in the Vertical Prototype. Instead, all newly registered users will be an Unapproved User by default.
- Moderators have the same attributes as any other user stored in the user table in the database: a first name, last name, email, SFSU ID number, SFSU ID picture, profile picture, role, and join date.
    - For moderators, role = 2.

Administrator / Admin: An approved user who has access to Gatormmunity's server, database, and GitHub repository.

- Administrators have more privileges than Moderators, and have access to administrative tools, including the ability to appoint moderators and ban them.
- Similar to moderators, the back end will check if the user performing the administrative actions is of the correct role before executing any administrative actions.
- Administrators are appointed by the developers (the team).
- Administrators have the same attributes as any other user stored in the user table in the database: a first name, last name, email, SFSU ID number, SFSU ID picture, profile picture, role, and join date.
    - For administrators, role = 3.

Group Member: An approved user who is a member of a group. Group Members are stored in the Group User table, an associative table between Group and User.

- Group Members have the same privileges as Approved Users, and have access to group-exclusive features such as creating forum threads in their group's forum or sending messages in their group's chat.
- Approved Users become Group Members when they join a group via an invite from another Group Member.
- Group Members are differentiated from the other Group User entities by means of their role attribute in the Group User table. For group members, role = 1.

Group Moderator: An approved user who is a moderator of a group. Group Moderators are stored in the Group User table, also.

- Group Moderators have more privileges than Group Members, and have access to group moderation tools, including the ability to kick group members from the group via the UI.
- Group Moderators are appointed and unappointed by Group Administrators in the UI.
- Group Moderators are differentiated from the other Group User entities by means of their role attribute in the Group User table. For group moderators, role = 2.

Group Administrator / Group Admin: An approved user who is an administrator of a group. Group Administrators are stored in the Group User table, also.

- Group Administrators have more privileges than Group Moderators, and have access to group administrative tools, including the ability to delete their group and appoint/unappoint Group Moderators via the UI.
- Group Administrators are differentiated from the other Group User entities by means of their role attribute in the Group User table. For group administrators, role = 3.

User Interface Terms:

Navigation Bar / Navbar: The navigation bar will be at the top of every page and will contain buttons that link to different pages within our application and a search bar.

- Only logged in users can see these buttons in the navigation bar. Users who are not logged in can only see the logo, buttons that link to the login or registration page, and the search bar.
- The navbar makes GET/POST requests that query the database. When a user uses the search bar in the navbar, the back end queries the database for the search results and the navbar redirects the user to the search page to display those search results.
- The navbar displays the profile picture of the user if they are logged in. The profile picture's URL is retrieved from the database which the client then uses to display the image.

Dashboard: The page that approved users will see after logging in, similar to a home page for only logged in users. The dashboard page is also accessible by clicking the logo in the navbar if the user is logged in. If the user is not logged in, the logo will take the user to the home page.

- The dashboard will show the user's profile picture, name, the newest forum threads in the Gatormmunity forum, and the newest marketplace listings. It is intended to catch the user up on what is happening right now in Gatormmunity and the threads/listings displayed may catch the user's eye.
- This page is only visible to logged in users. The code does this by checking the back end if the user is logged in before rendering the content in the page.

Gatormmunity Forum:

Gatormmunity Forum / Forum: A forum where all approved users can start their own forum threads and make forum posts in existing threads. The page will show a list of threads that are of the category the user specified. The threads are retrieved from the database, and the filtering will be done on the server's side.

- Only logged in users may see this page.

Forum Category: Threads belong to categories which are intended to help approved users find threads that they are interested in. Approved users can specify which category of threads they would like to see on the Gatormmunity Forums page. By default, no category filter is specified.

- Example categories: General, Gaming, Social

Forum Thread / Thread: A collection of posts. Threads have a title which is determined by the person starting the thread. Threads may belong to a group, and if they are, they will only be visible in that group's forum. Forum threads are comparable to discussion topics on iLearn, which other students can reply to.

- Forum threads can be clicked on, which will lead to the View Thread page, which shows the posts other approved users have made in reply to the thread.

Forum Post / Post: A message that approved users can post in a thread. Forum posts are comparable to the replies in a discussion topic on iLearn, including the post that started the discussion topic. Every forum post must have content and is associated with a thread and a user (the author) by means of a thread id and user id.

- The first post that starts a thread may also be referred to as the "original post". The original post is automatically created when a user creates a thread, since the create thread form asks for the body of the first post as well as the thread's title.
- Only the original (first) post of a thread may contain an image. All other posts in a thread will not be able to have images.

Post Attachment: Approved users can attach an image to the first post of the thread when creating a thread. No other post in a thread may contain an image. The image's filename on the server will be randomized in order to avoid name clashes, but the filename the user gave their file will be saved in the database and displayed next to the image in the post.

- The image must be of an image format. Supported formats are: JPEG, PNG, WebP, GIF, and AVIF.
- The image must be at most 5 MB in size.

Pin: A pinned forum thread will always appear first when displaying the list of threads. This is not a priority 1 item.

Bookmark: Approved users can bookmark forum threads, which saves the thread into the user's bookmarks. Bookmarks exist to help the user easily find threads that they want to go back to in the future. This is not a priority 1 item.

Gatormmunity Marketplace:

Gatormmunity Marketplace / Marketplace: A page within the application where approved users can buy and sell goods and services. Each good/service is displayed in a card, called a listing.

- Only logged in users may see this page.

- Approved users may filter what they see in the marketplace page by inputting a category filter and max price filter. The category filter, when applied, shows only listings that match the category specified, while the max price filter only shows listings that do not exceed the price entered. Both filters may be used at the same time.
- The listings shown on this page are retrieved from the Listing table in the database.

Marketplace Buyer / Buyer: An approved user who is buying something from Gatormmunity's marketplace. There is nothing special about a buyer compared to an approved user, it is just terminology that refers to the person making the purchase.

Marketplace Seller / Seller: An approved user who is selling something on Gatormmunity's marketplace. The listing that the seller makes will show the seller's name and email, which the buyer may use to contact them with.

Marketplace Listing / Listing: An item or service a seller is trying to sell on the marketplace. It must have a photo of the item being sold, a description, a title, a price, and a category. The seller's contact information (email and direct message option) will be automatically included in the listing.

- The listing will not need to be approved before being listed, but approved users can report listings and moderators can delete listings that break the rules.
- The photo must be of an image format. Supported formats are: JPEG, PNG, WebP, GIF, and AVIF. The image must be at most 5 MB in size.

Community Features:

Gator Chat: A chat room for all approved users of Gatormmunity. Approved users can send messages and receive messages from other approved users in the chat room. No images can be attached to the messages.

- The chat room is accessible from the Chat button in the navbar, and is only accessible to logged in, approved users. Every message must contain some text in its body.

Direct Message / DM: A private method of communication between two approved users. No images can be attached to the messages.

- An approved user's direct messages are accessible from the Inbox button in the navbar, and an approved user must be logged in to see their direct messages. Every message must contain some text in its body and every message is associated with a pair of users: the sender and recipient.

User Groups / Groups: Approved users can form their own groups which come with exclusive features such as a group-exclusive chat and forum.

- Members of a group can have 3 roles: Group Member, Group Moderator, or Group Administrator.

- Groups can have a picture which must be of an image format. Supported formats are: JPEG, PNG, WebP, GIF, and AVIF. The group picture must be at most 5 MB in size.
- Groups can have a description and announcement, which are displayed on the group's home page. It serves to tell new group members about the group's purpose and rules, as well as show a message the group admin would like all of its members to see on the home page.

Group Chat: A chat room for all members of a group. All group members can send messages and receive messages from other group members in the group chat room. No images can be attached to these messages.

- The group's chat is accessible in the navbar, via the Chat button. Only logged in users who are members of the group can see a group's chat and send messages in it.
- Each message must have content, and each message is associated with a sender (a user) via their user id.

2. Prioritized Functional Requirements

Priority 1:

1. All Users
   - 1.1. All users shall be able to contact the developers.
   - 1.2. All users shall be able to view the home page.
   - 1.3. All users shall be able to view the registration page.
   - 1.4. All users shall be able to view the login page.
   - 1.5. All users shall be able to view the about page to learn about the application.

2. Guest User
   - 2.1. A guest user shall be able to register for an account.

3. Approved User
   - 3.1. An approved user shall be able to log in to their account.
   - 3.2. An approved user shall be able to change their password.
   - 3.3. An approved user shall be able to reset their password.
   - 3.4. An approved user shall be able to delete their account.
   - 3.5. An approved user shall be able to make a forum thread.
   - 3.6. An approved user shall be able to make a forum post.
   - 3.7. An approved user shall be able to attach images to their forum posts.
   - 3.8. An approved user shall have a profile page.
   - 3.9. An approved user shall be able to edit their own profile page.
   - 3.10. An approved user shall be able to view the profile page of another approved user.
   - 3.11. An approved user shall have a profile picture.
   - 3.12. An approved user shall be able to change their profile picture.
   - 3.13. An approved user shall be able to create a user group.
   - 3.14. An approved user shall be able to join a user group.
   - 3.15. An approved user shall be able to send direct messages to other approved users.
   - 3.16. An approved user shall be able to receive direct messages from other approved users.
   - 3.17. An approved user shall be able to send messages in Gator Chat.
   - 3.18. An approved user shall be able to see marketplace listings in the marketplace.
   - 3.19. An approved user shall be able to select marketplace listings in the marketplace.
   - 3.20. An approved user shall be able to report marketplace listings to a moderator.
   - 3.21. An approved user shall be able to buy items from the marketplace.
   - 3.22. An approved user shall be able to sell items on the marketplace.
   - 3.23. An approved user shall be able to edit the details of an item they are selling.
   - 3.24. An approved user shall be able to upload pictures of an item they are selling.
   - 3.25. An approved user who is selling an item shall be able to list their accepted payment methods.
   - 3.26. An approved user who is selling an item shall be able to list their possible delivery methods.

3.27. An approved user who is selling an item shall be able to list their preferred contact methods.
3.28. An approved user who is selling an item shall be able to assign their listing to a category.
3.29. An approved user shall be able to search for users.
3.30. An approved user shall be able to search for marketplace listings.
3.31. An approved user shall be able to search for forum posts.
3.32. An approved user shall be able to apply filters to their user search results.
3.33. An approved user shall be able to apply filters to their marketplace search results.
3.34. An approved user shall be able to apply filters to their forum post search results.

4. Moderator
   4.1. A moderator shall be able to approve unapproved users.
   4.2. A moderator shall be able to reject unapproved users.
   4.3. A moderator shall be able to ban approved users from Gatormmunity.
   4.4. A moderator shall be able to delete forum threads.
   4.5. A moderator shall be able to delete forum posts.
   4.6. A moderator shall be able to delete messages in Gator Chat.
   4.7. A moderator shall be able to delete listings in the marketplace.

5. Administrator
   5.1. An administrator shall be able to appoint moderators.
   5.2. An administrator shall be able to unappoint moderators.
   5.3. An administrator shall be able to ban moderators from Gatormmunity.

6. Group Member
   6.1. A group member shall be able to invite other approved users to their group.
   6.2. A group member shall be able to leave their group.
   6.3. A group member shall be able to create forum threads in their group's forum.
   6.4. A group member shall be able to create forum posts in their group's forum.
   6.5. A group member shall be able to send messages in their group's chat.

7. Group Moderator
   7.1. A group moderator shall be able to kick group members from their group.
   7.2. A group moderator shall be able to delete forum threads in their group's forum.
   7.3. A group moderator shall be able to delete forum posts in their group's forum.
   7.4. A group moderator shall be able to delete messages in their group's chat.

8. Group Administrator
   8.1. A group administrator shall be able to delete their group.
   8.2. A group administrator shall be able to add a description about their group.
   8.3. A group administrator shall be able to edit their group's description.
   8.4. A group administrator shall be able to appoint group moderators.

8.5. A group administrator shall be able to unappoint group moderators.
8.6. A group administrator shall be able to kick group moderators from the group.
8.7. A group administrator shall be able to resign their position as group administrator and give the position to another member of the group.

Priority 2:

9. All Users
    9.1. All users shall be able to donate money to the developers via PayPal. PayPal has a donate button that prompts the user to specify how much they want to donate, and that donation shall go to the developers.
    9.2. All users shall be able to donate money to the developers via Venmo. The developers have a Venmo username that the user can direct their Venmo donation to.

10. Approved User
    10.1. An approved user shall be able to attach images to their chat messages.
    10.2. An approved user shall be able to sort forum threads.
    10.3. An approved user shall be able to bookmark forum threads.
    10.4. An approved user shall be able to like forum posts.
    10.5. An approved user shall be able to report forum posts to a moderator.
    10.6. An approved user shall be able to report other users to a moderator.
    10.7. An approved user shall be able to block other users.
    10.8. An approved user shall be able to add items to a wishlist.
    10.9. An approved user shall be able to leave feedback about the seller they purchased an item from.

Priority 3:

11. Approved User
    11.1. An approved user shall be able to compare different marketplace listings against each other.

12. Moderator
    12.1. A moderator shall be able to pin forum threads.

13. Administrator
    13.1. An administrator shall be able to create new categories in the forum.
    13.2. An administrator shall be able to delete categories in the forum.

14. Group Moderator
    14.1. A group moderator shall be able to pin forum threads in their group’s forum.

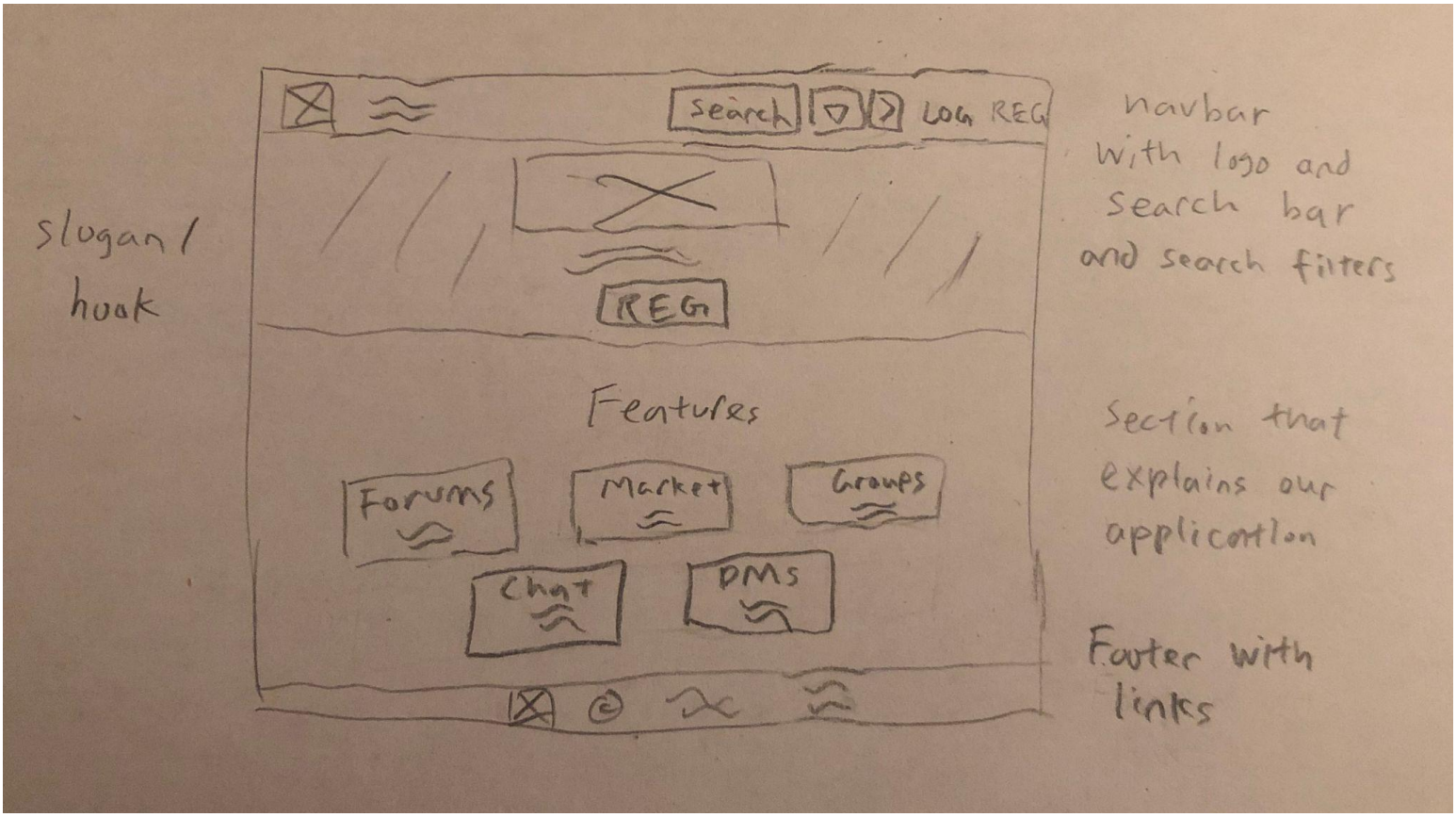
15. Group Administrator
    15.1. A group administrator shall be able to create new categories in their group's forum.
    15.2. A group administrator shall be able to delete categories in their group's forum.

3. UI Mockups and Storyboards

UI Mockups of the Main Areas of the GUI:

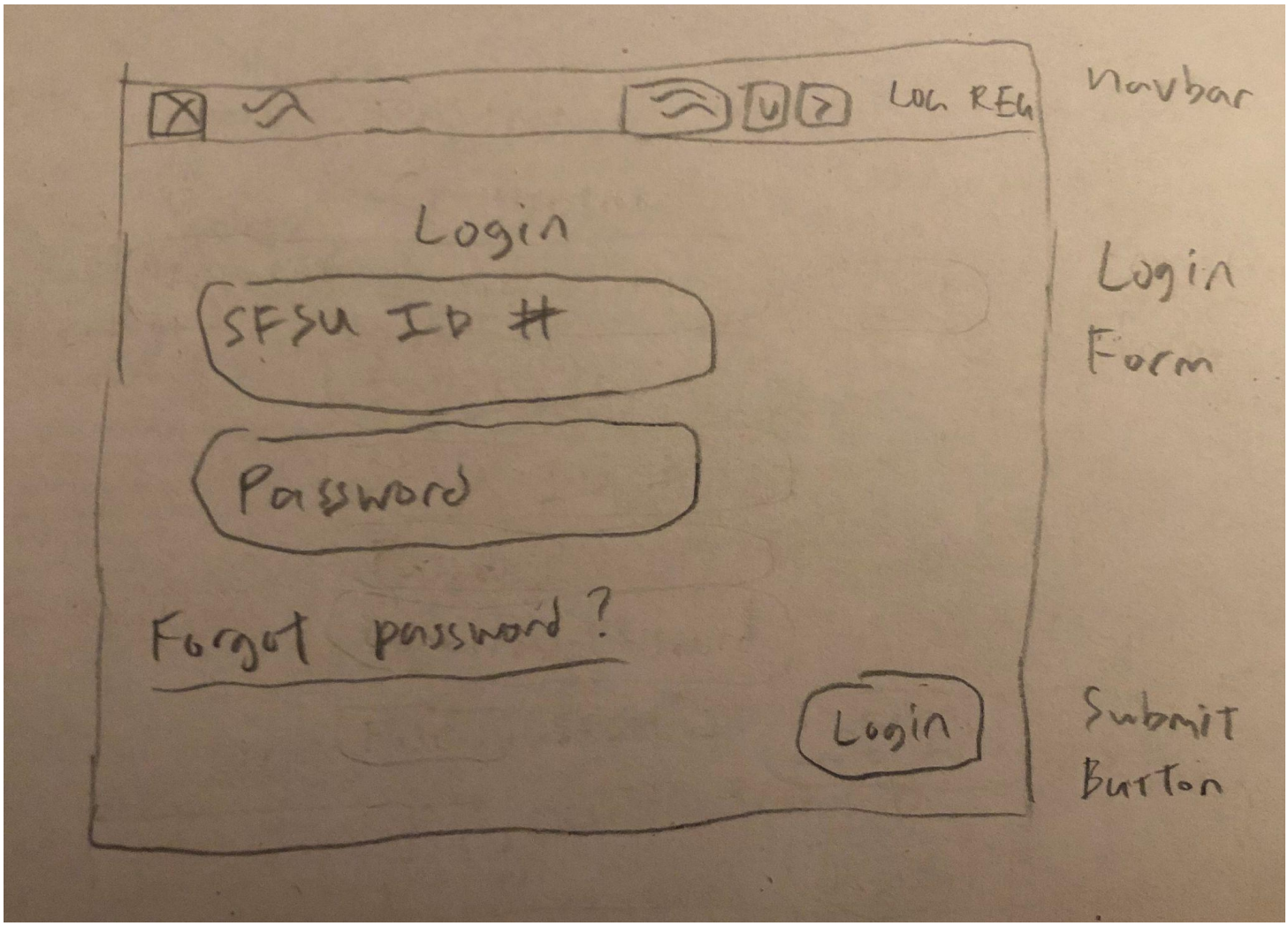
Home Page:

This page is the first page a user sees when going to Gatormmunity. It may also be accessed by users when they click on the logo in the navbar.



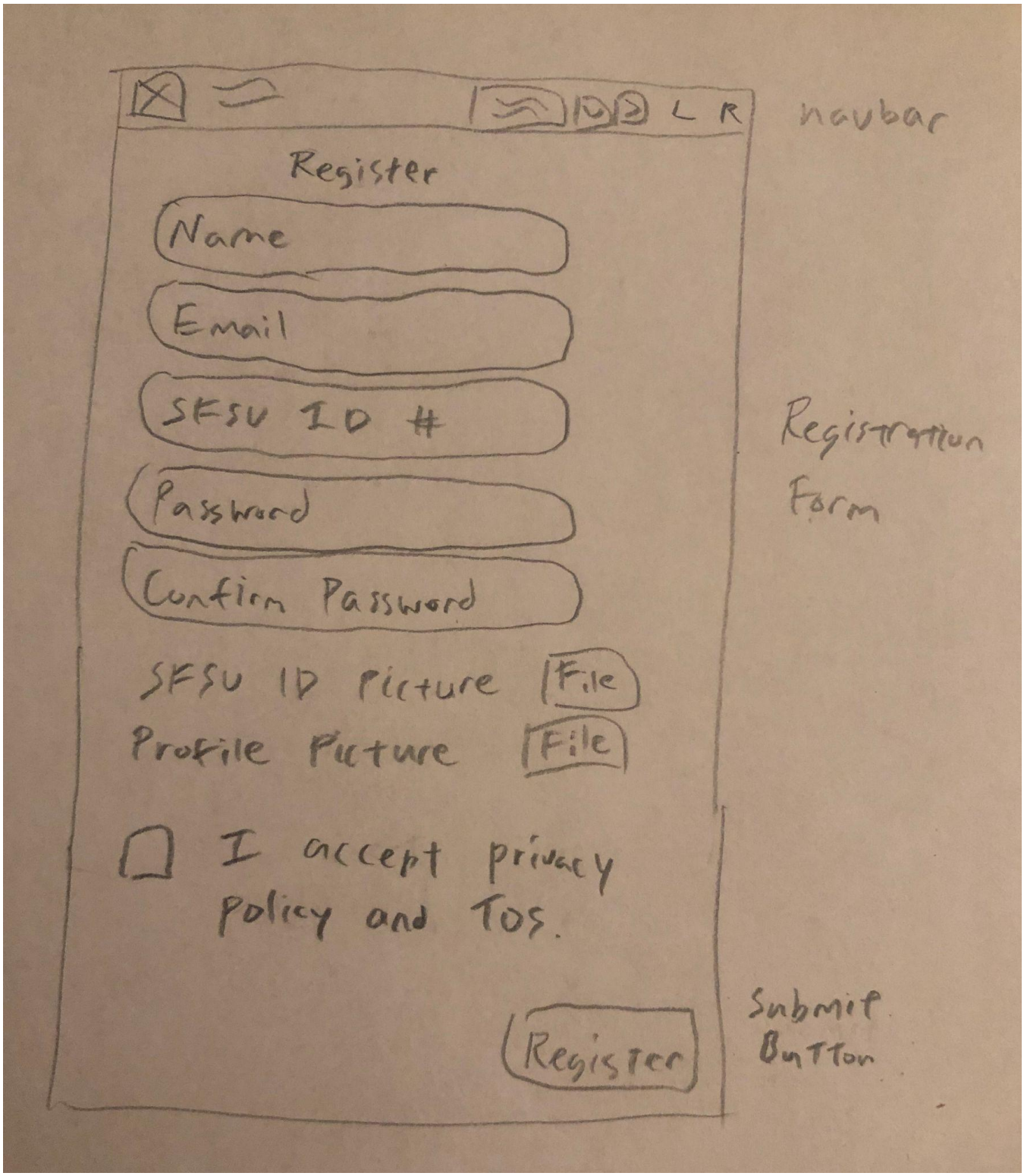
Login Page:

This page is accessed via the login button in the navbar.



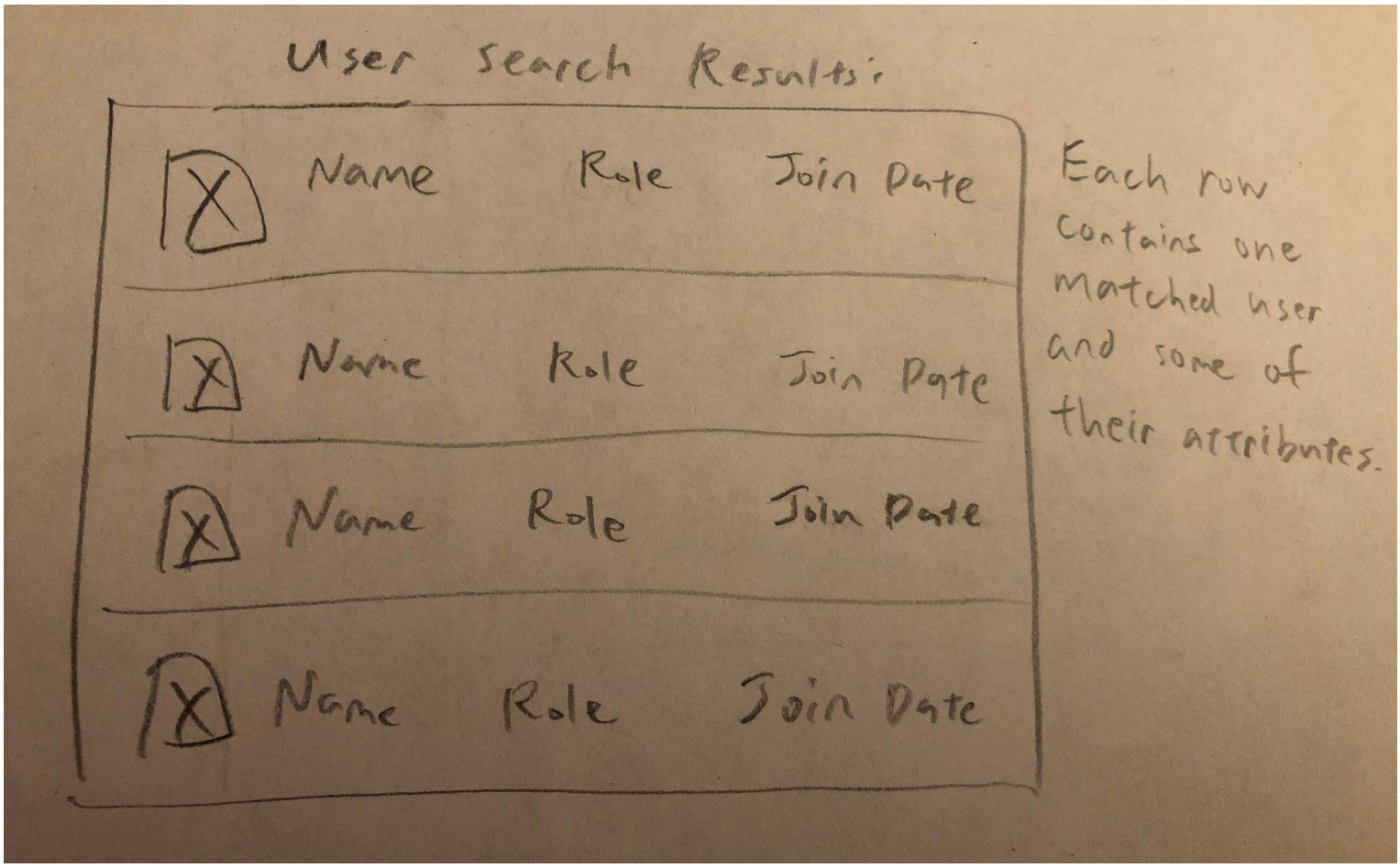
Registration Page:

This page is accessed via the Register button in the navbar.



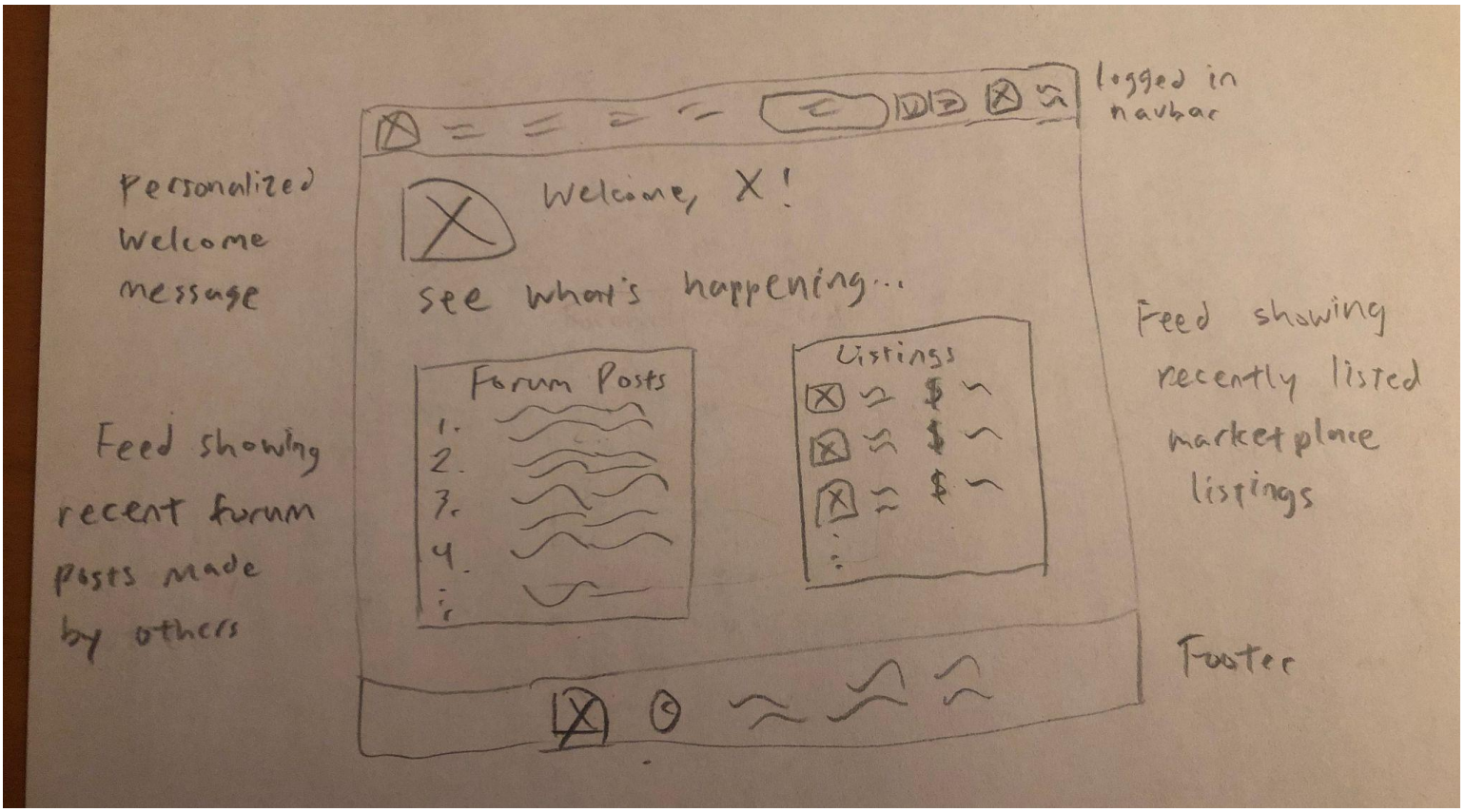
User Search Results:

The user search is the default search setting for our search feature. Searching for listings or threads requires clicking the “Search Listings” or “Search Threads” button on the Search page.



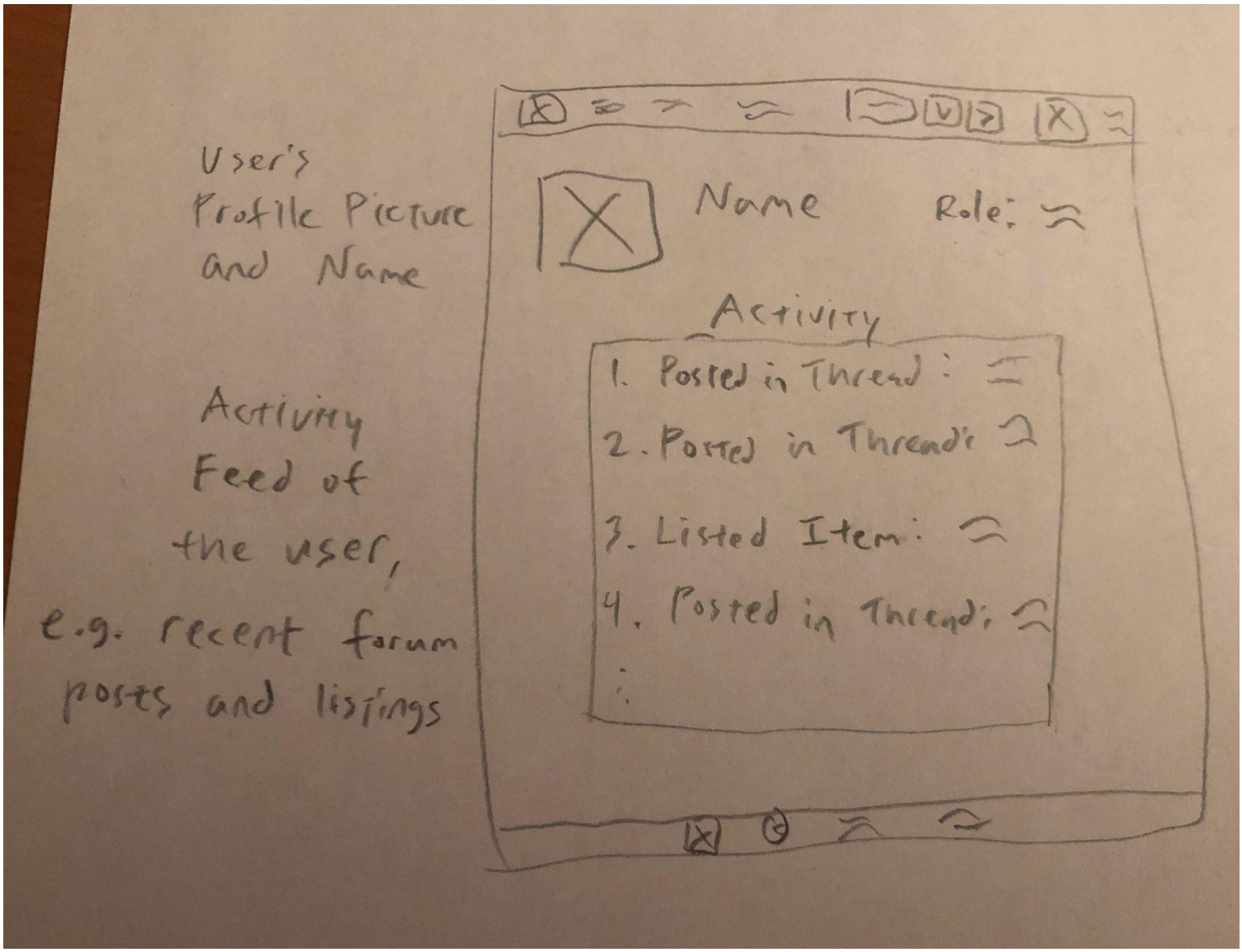
Dashboard Page:

This page is shown after a user logs in, and is also accessible from the navbar's dropdown when clicking one's profile picture.



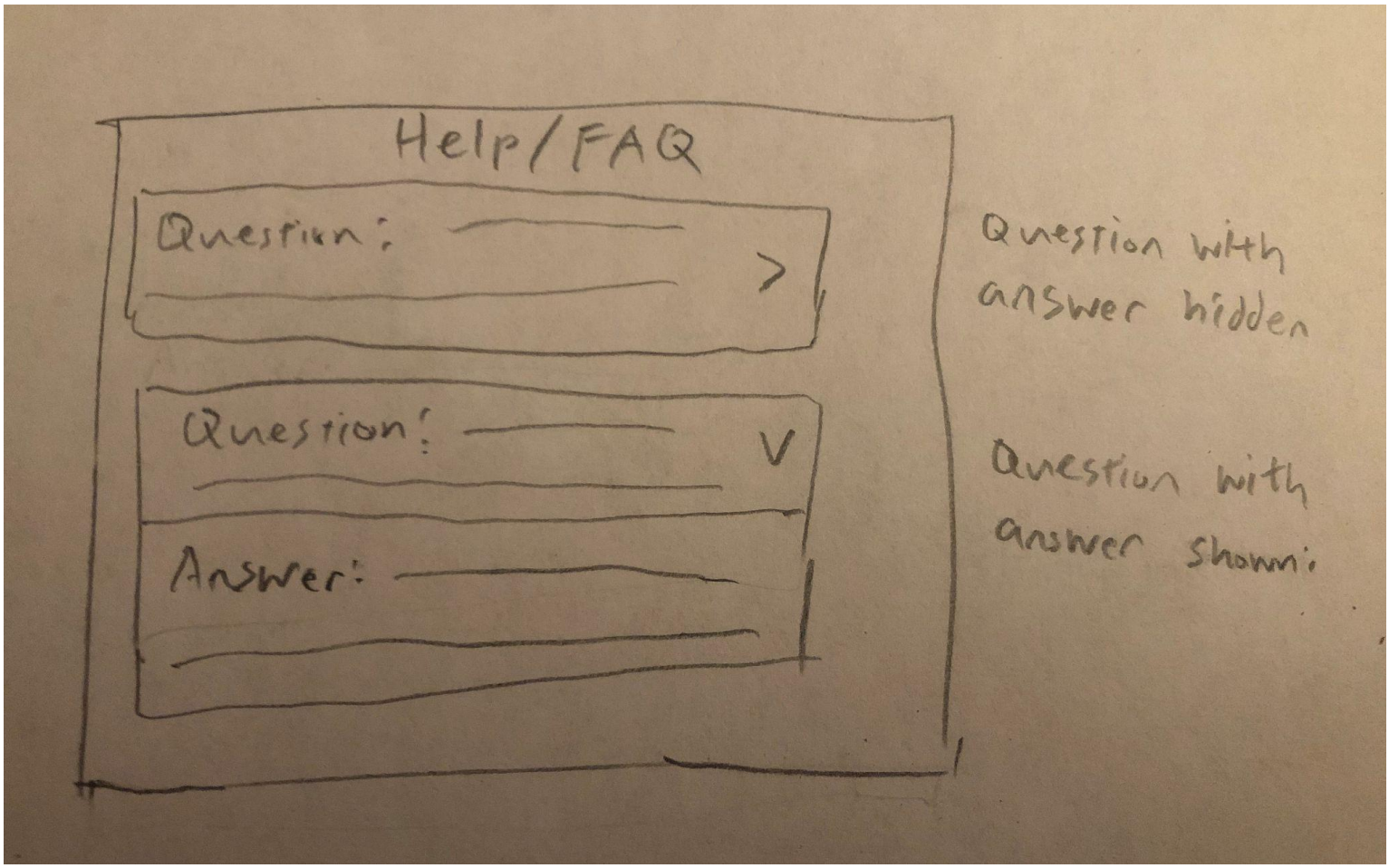
User Profile Page:

This page can be accessed by clicking on your own profile picture in the navbar, or by clicking someone else's profile picture or name in many other pages.



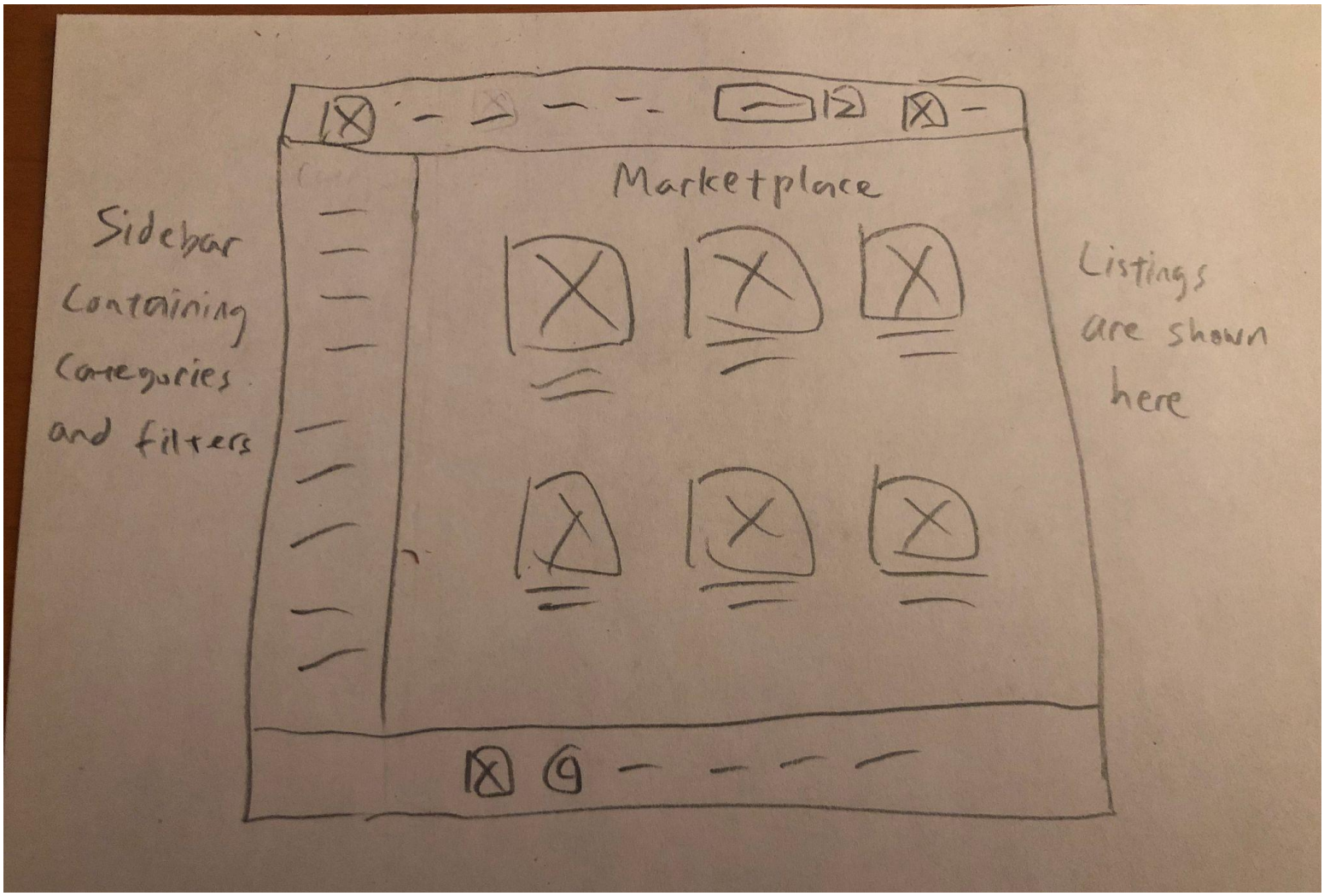
Help/FAQ Page:

This page can be accessed from the Help button in the footer.



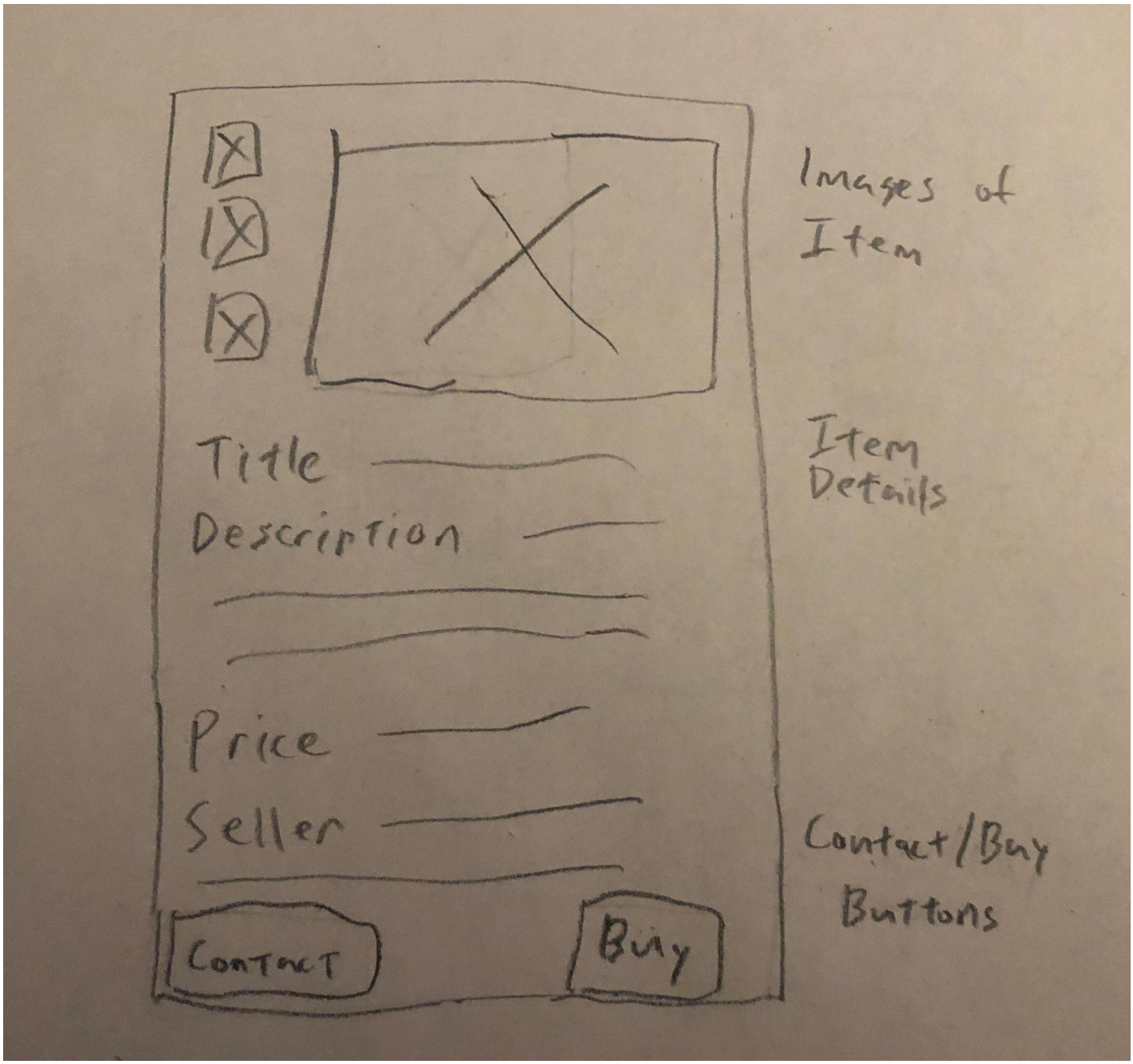
Marketplace Page:

This page can be accessed via the Marketplace button in the navbar.



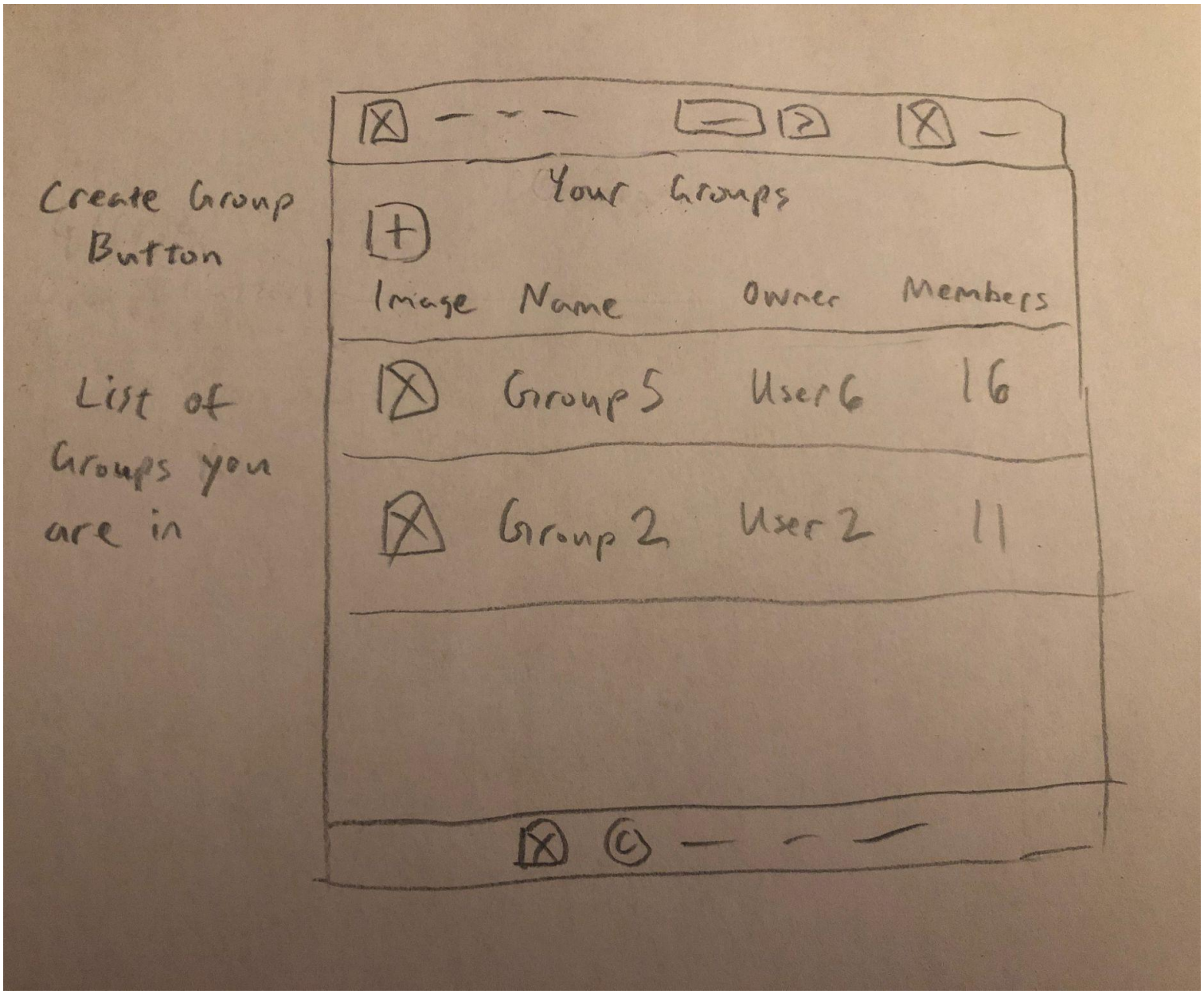
Marketplace Listing:

This page can be accessed via the Search page when clicking on a matched listing or by clicking on a listing in the Marketplace page.



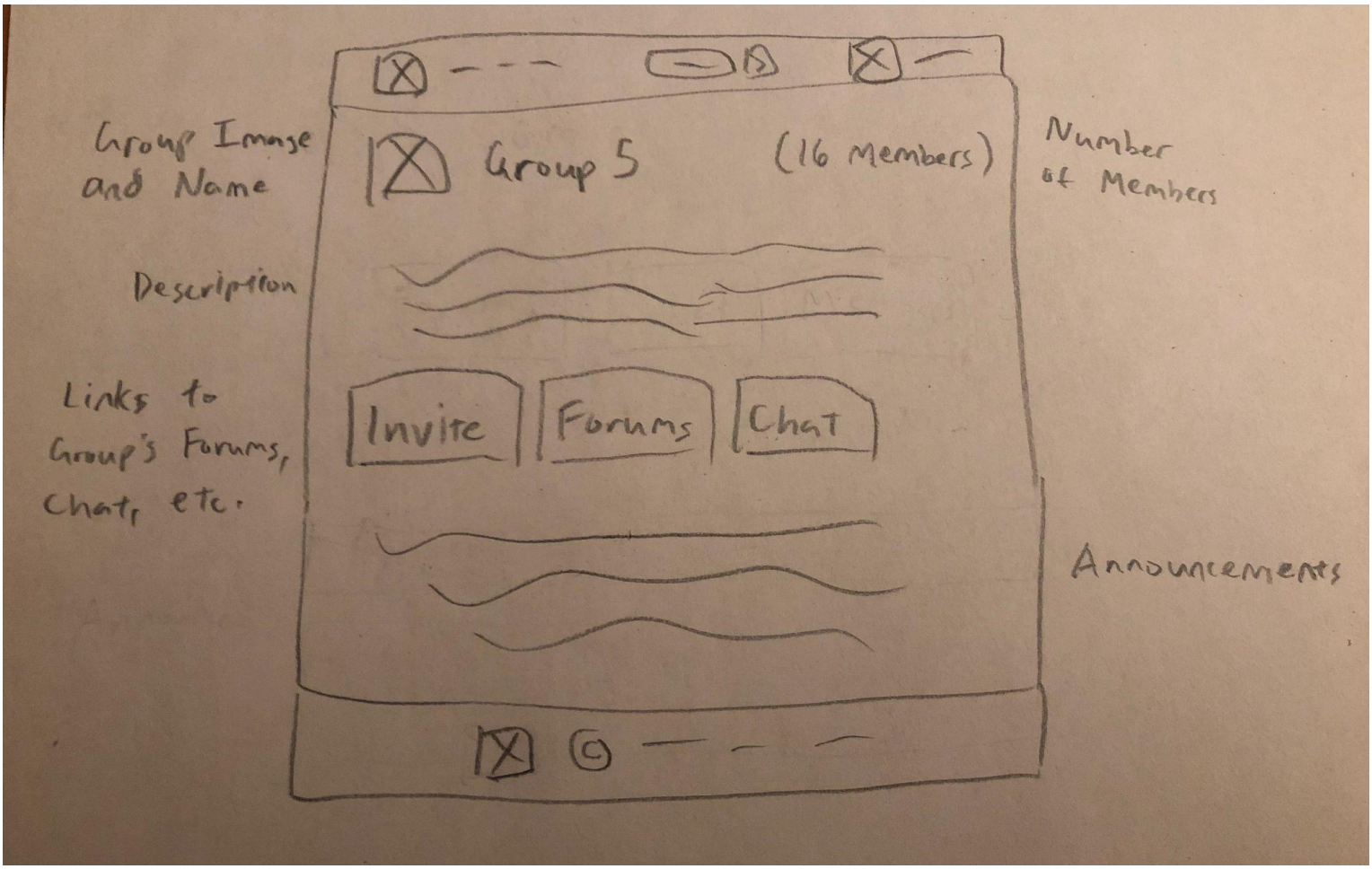
User's Groups:

This page is accessible via the Groups button in the navbar.



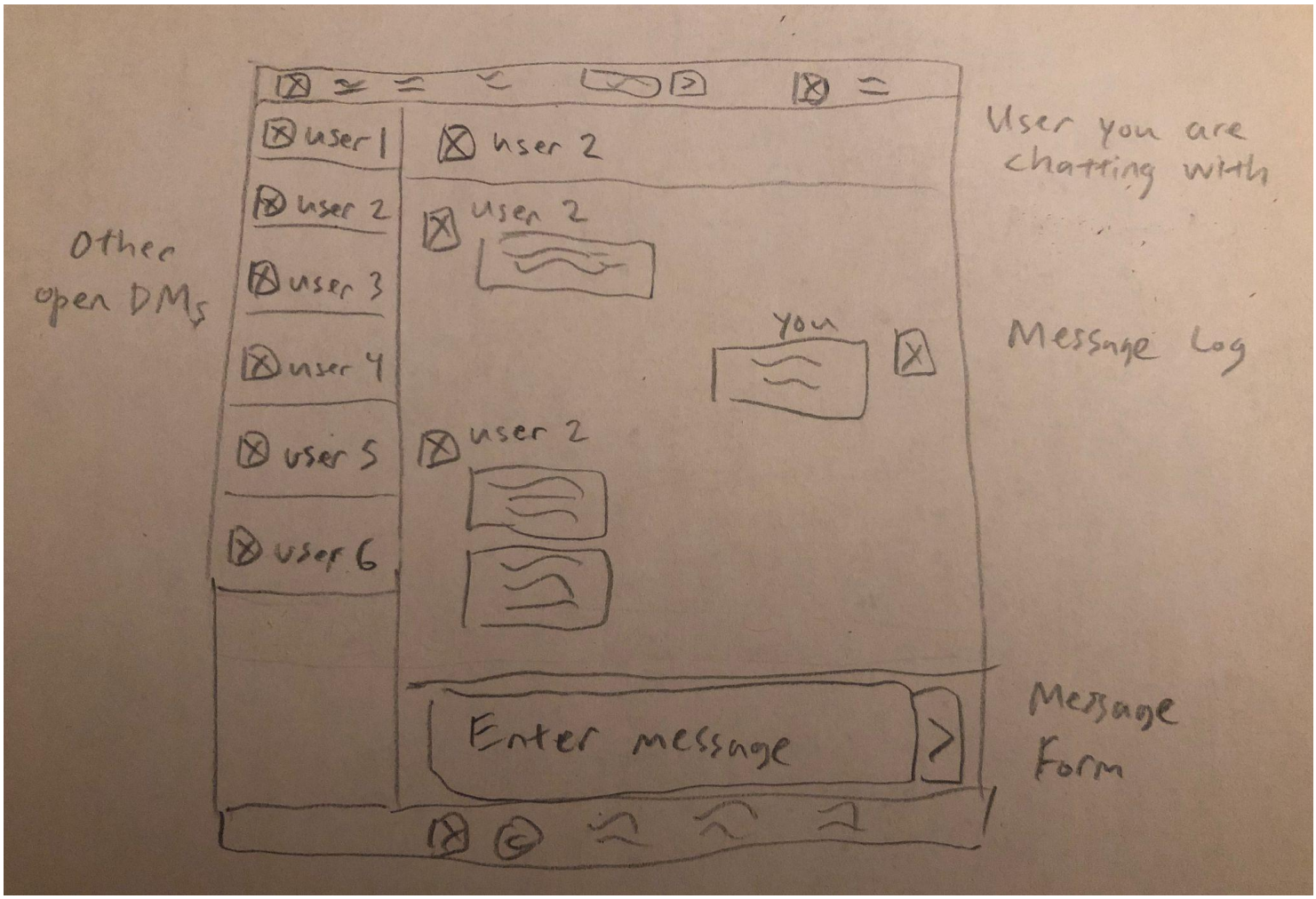
Group Home Page:

This page can be accessed by clicking on one of the groups from the User's Groups page, for instance.



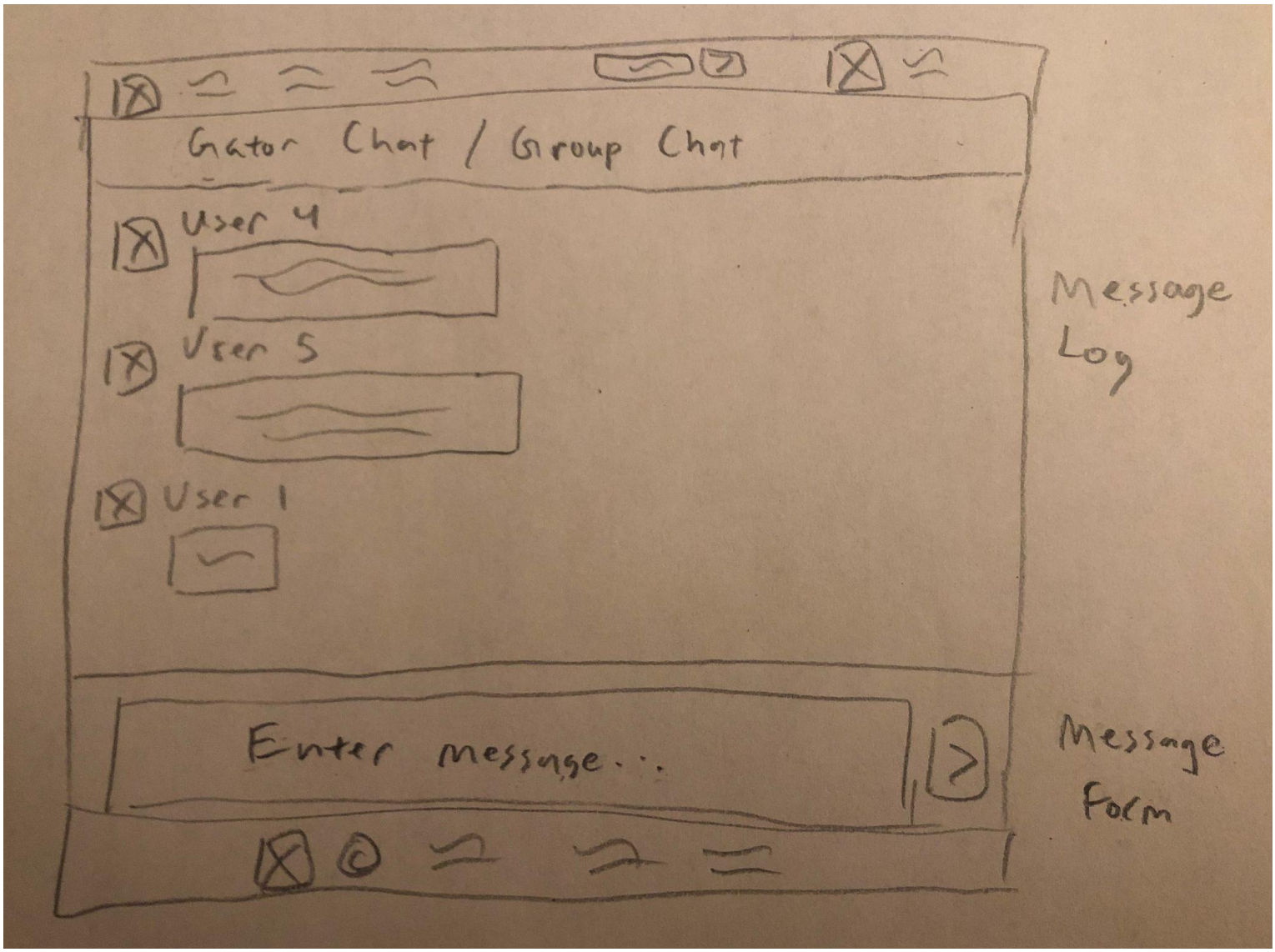
Direct Message (DM):

This page is accessible via the Inbox button in the navbar.



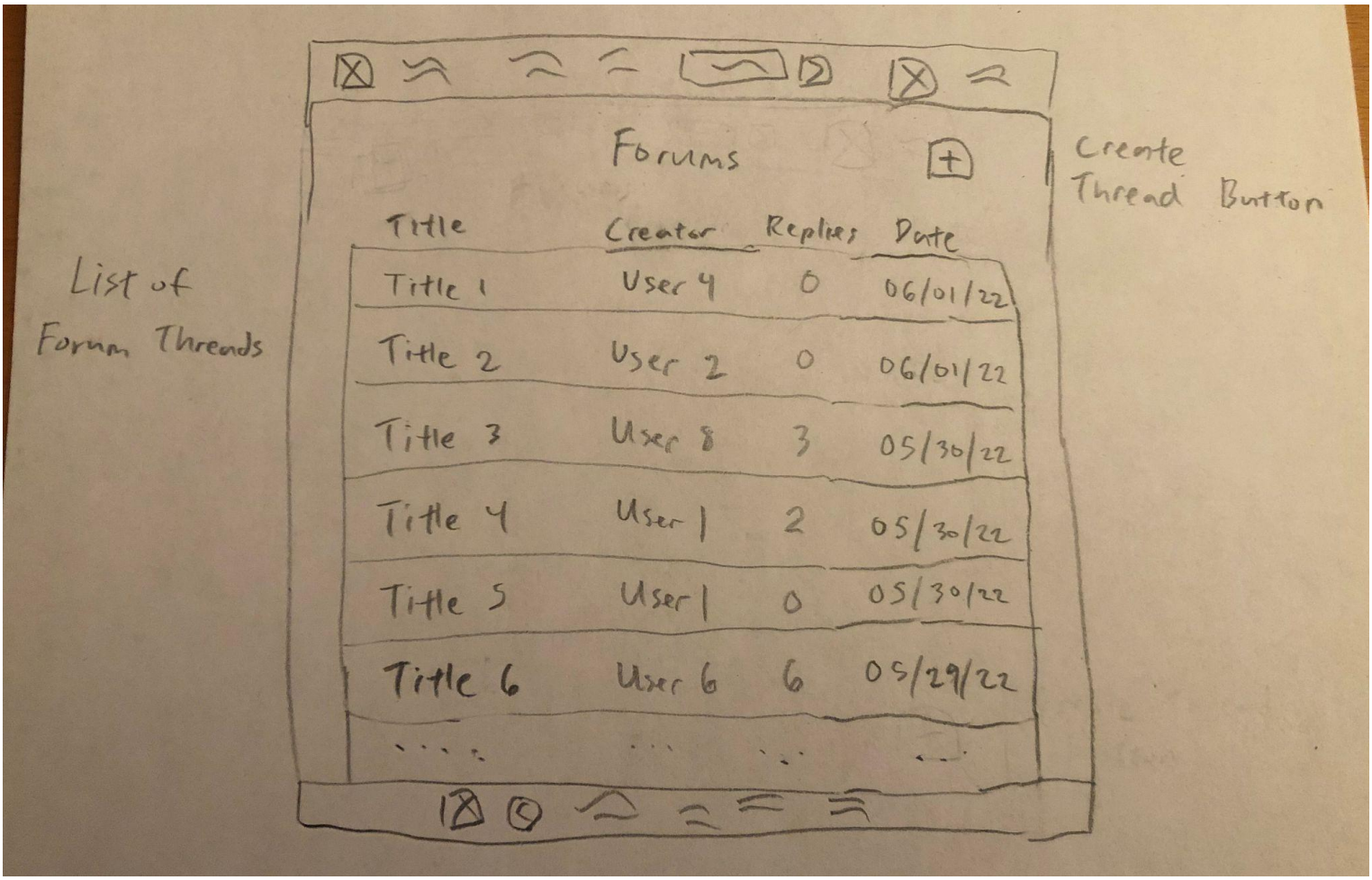
Gator/Group Chat:

This page is accessible via the Chat button in the navbar.



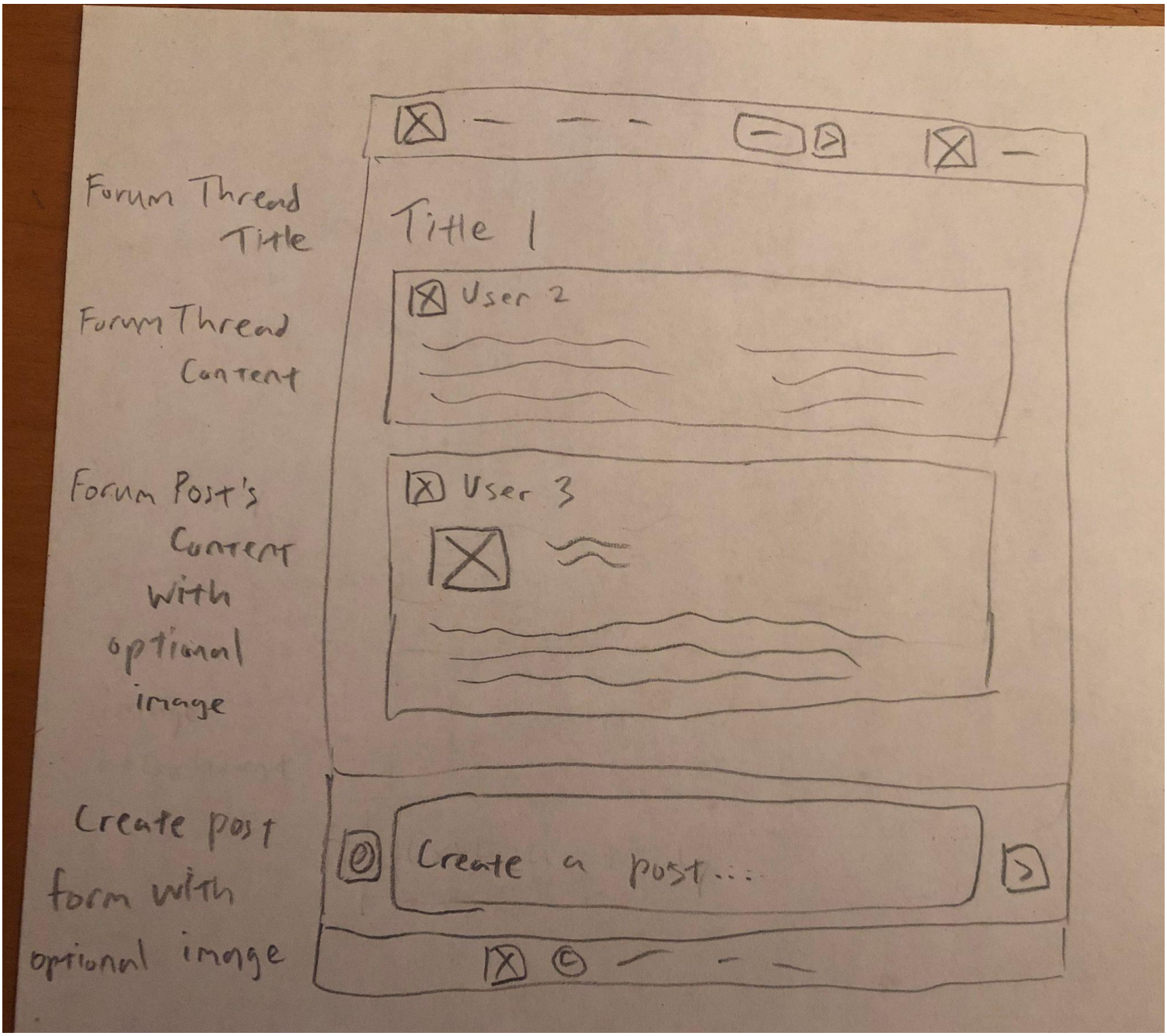
Gatormmunity/Group Forum:

This page is accessible via the Forums button in the navbar.



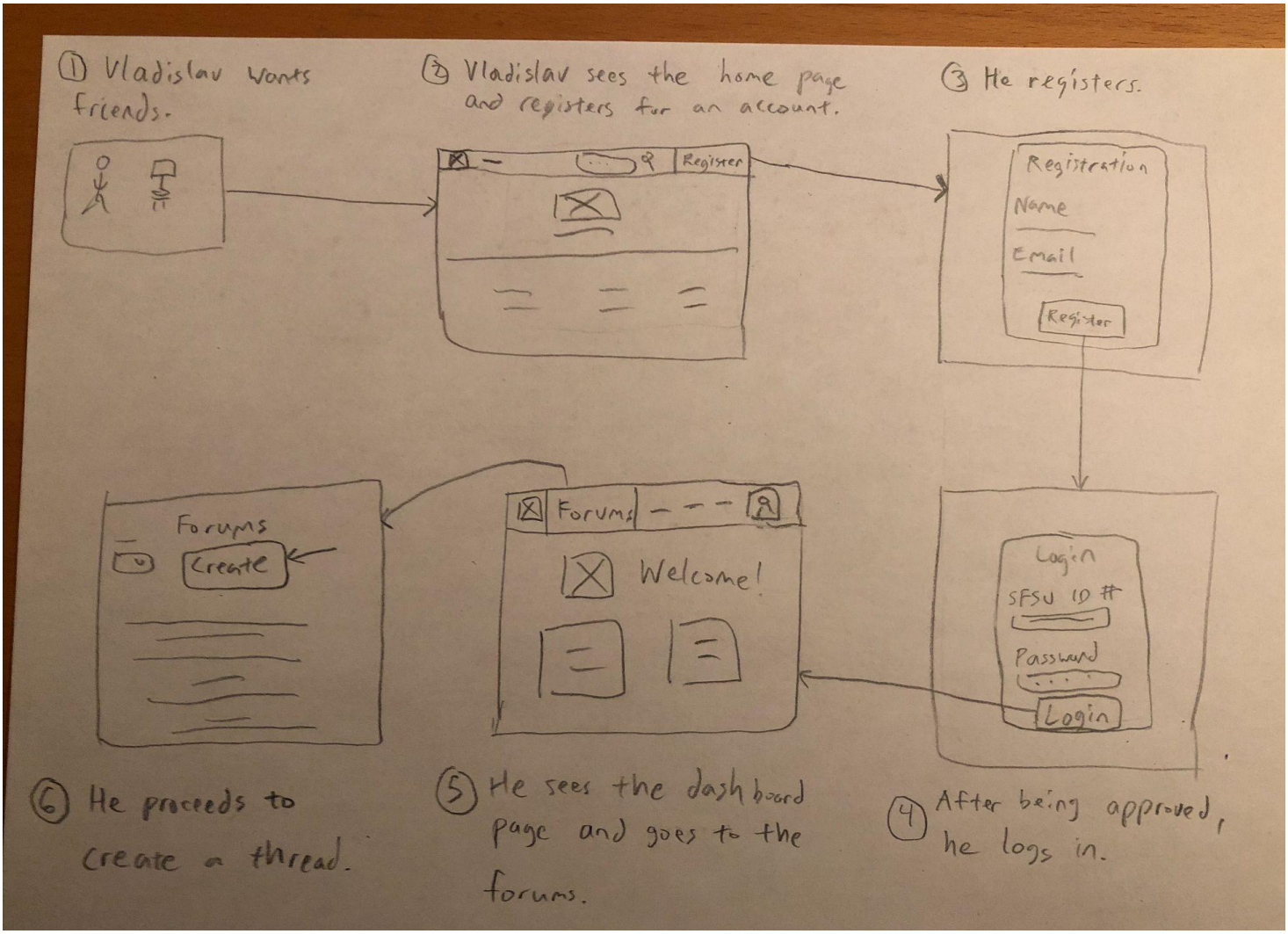
Viewing a Gatormmunity/Group Forum Thread:

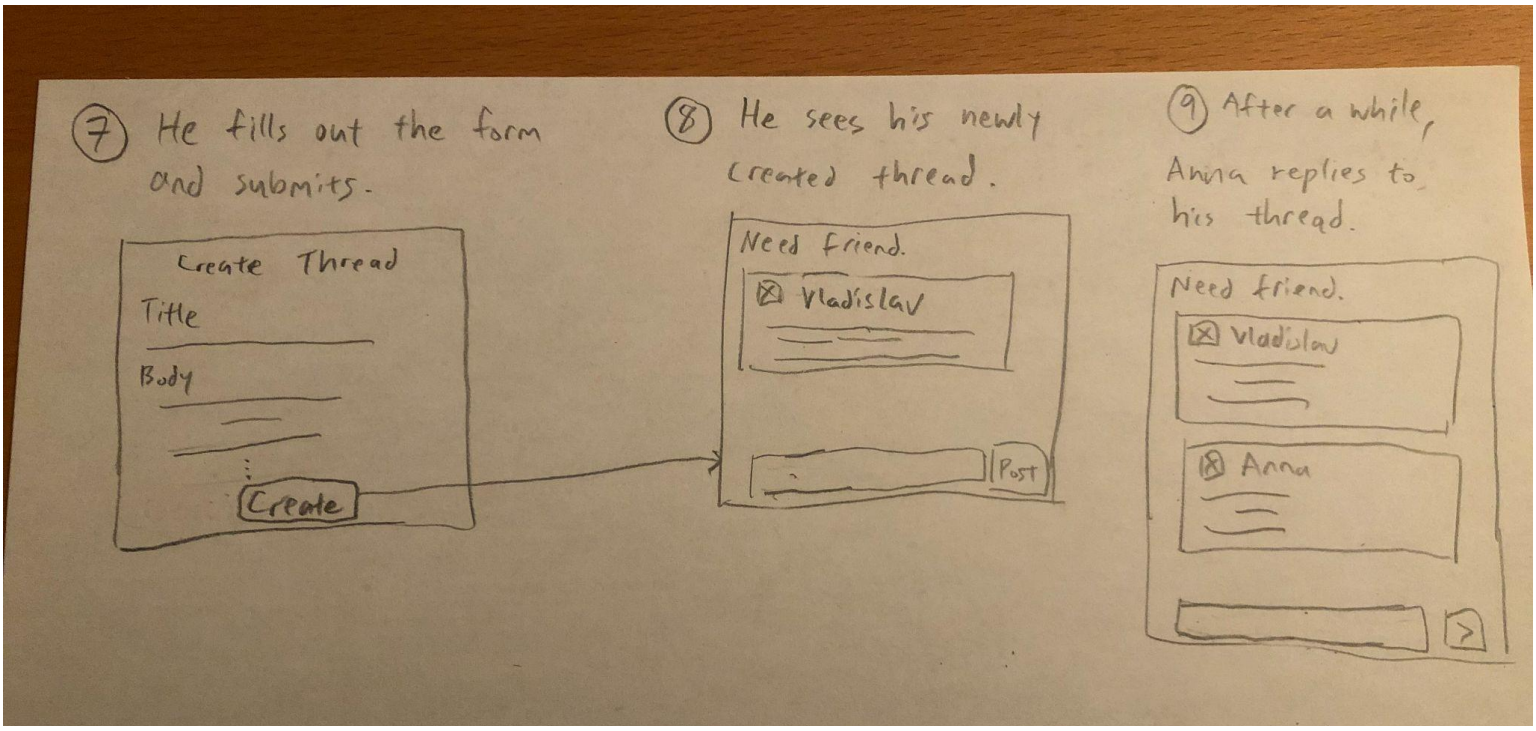
This page is accessible by clicking on a thread in the forums, for instance.



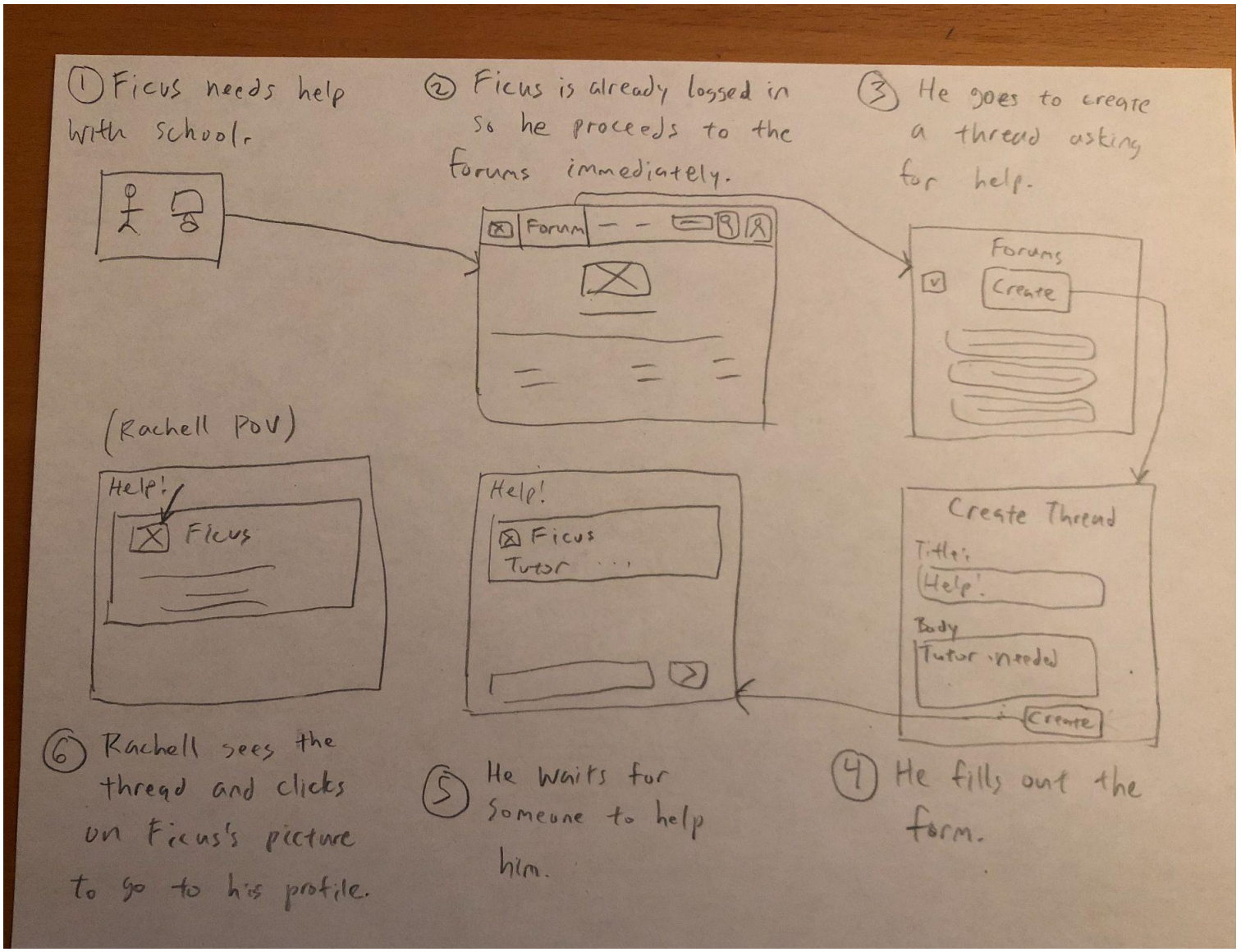
Storyboards for Use Cases:

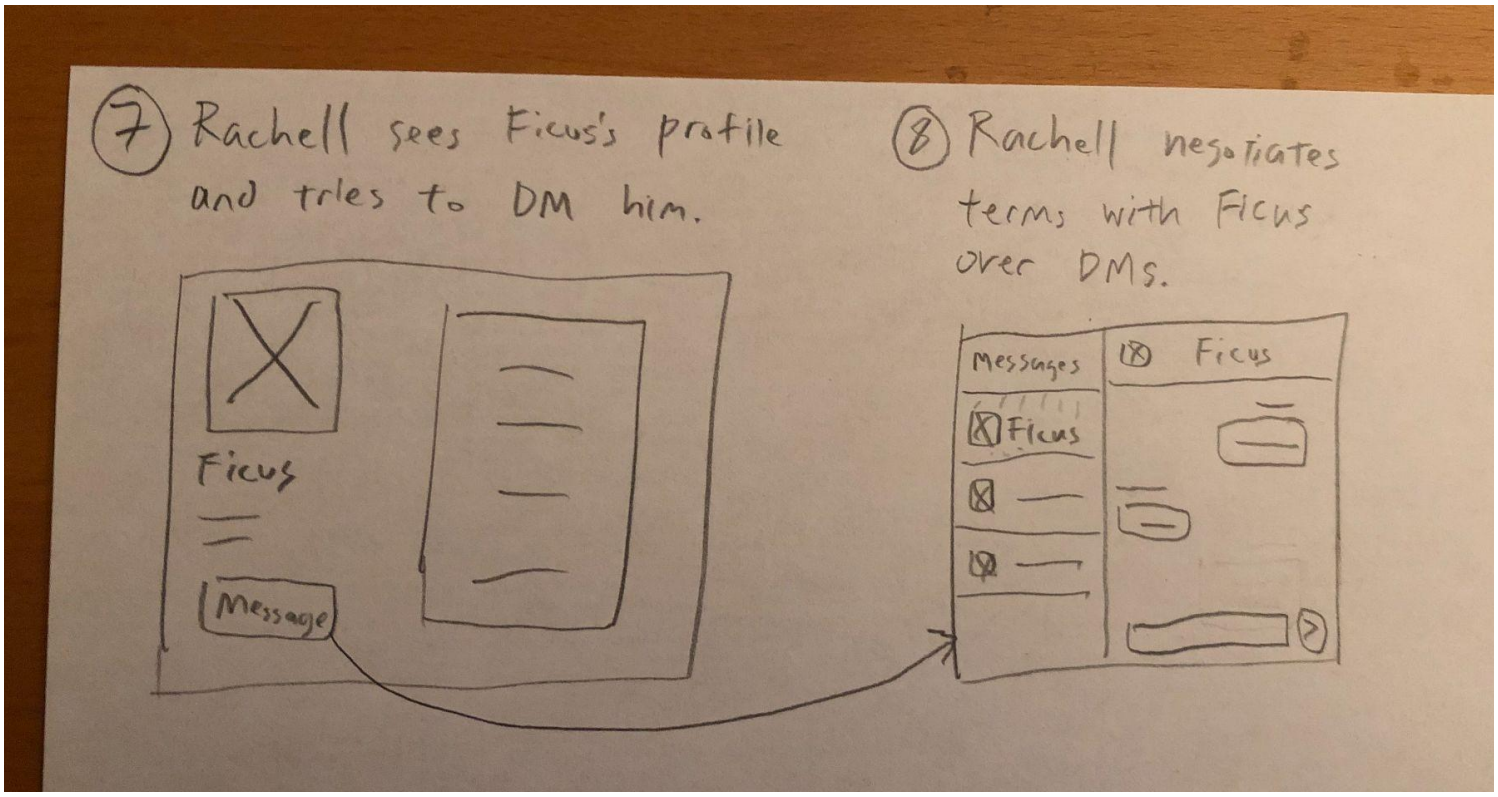
Use Case #1: Russian Friends
Vladislav is trying to make friends in SFSU so he proceeds to make an account on Gatormmunity. After registering and logging in, he goes to the forums and makes a thread stating that he is looking for friends. Anna sees his thread and replies saying she would like to be his friend.



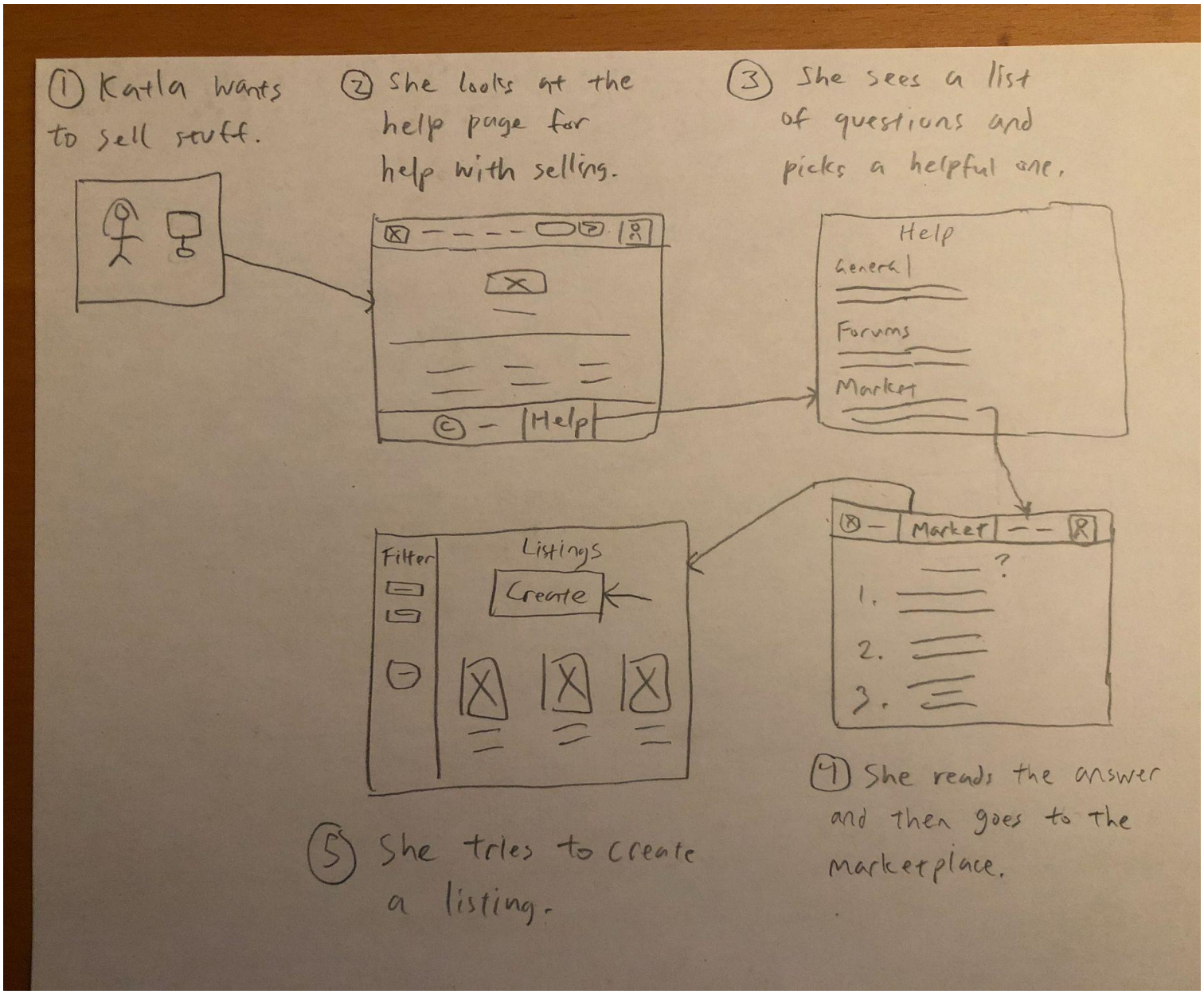

7 He fills out the form and submits.
Create Thread
Title
Body
Create
8 He sees his newly created thread.
Need Friend.
Vladislav
Post
9 After a while, Anna replies to his thread.
Need friend.
Vladislav
Anna

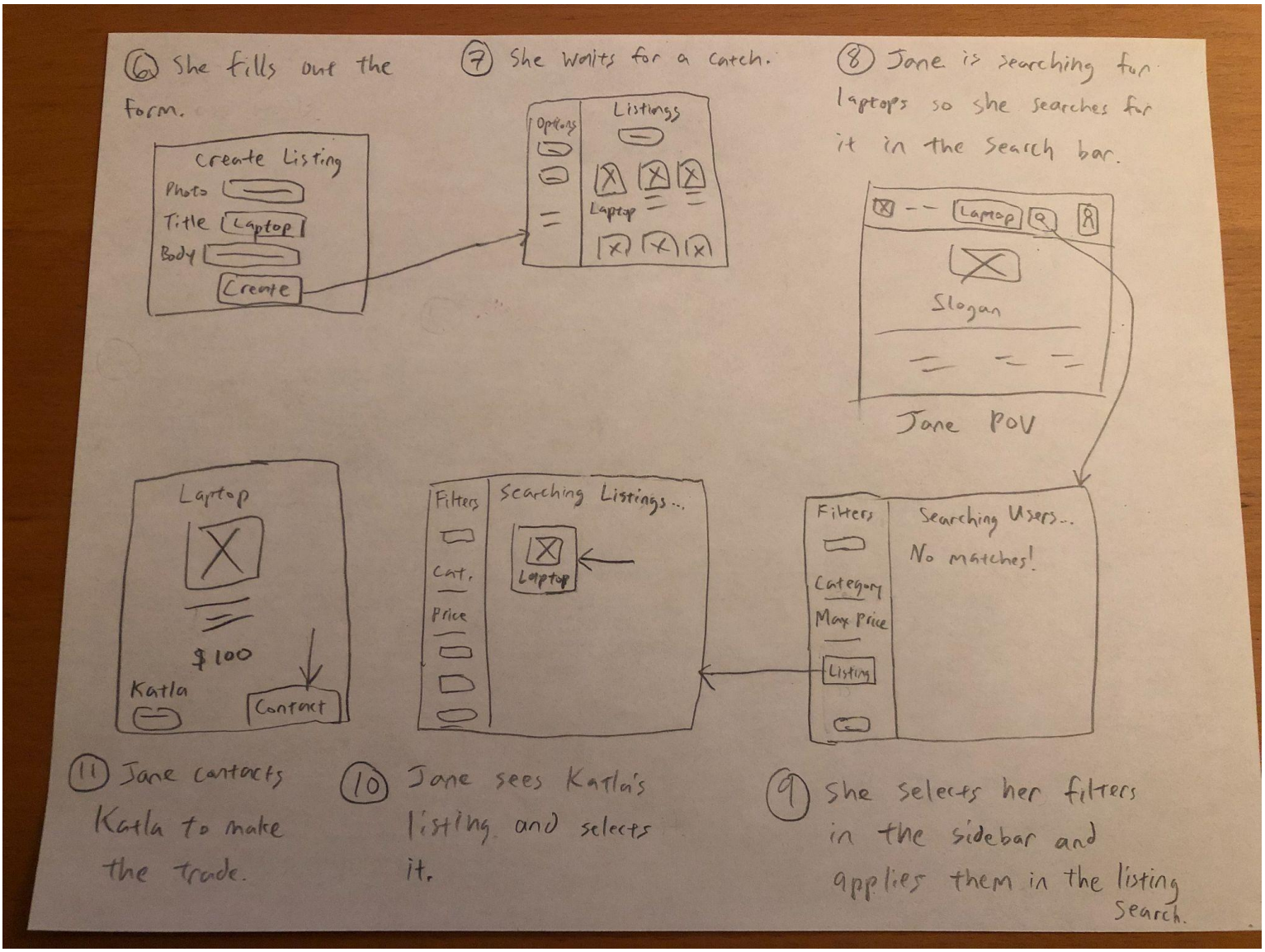
Use Case #2: Struggling Student
Ficus is looking for someone to help him with his coursework, thus he creates a thread in the forums asking for help. Rachell sees his thread and provides him with help after contacting him through a direct message.



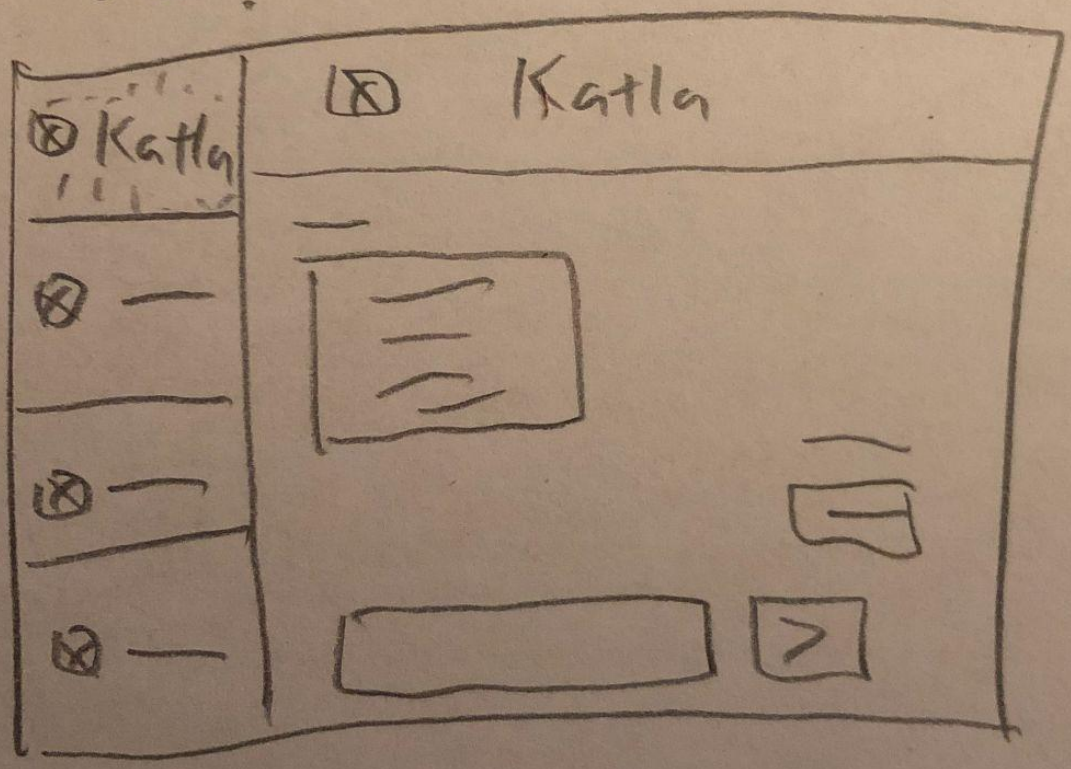

7 Rachell sees Ficus's profile and tries to DM him.
Ficus
Message
8 Rachell negotiates terms with Ficus over DMs.
Messages
Ficus
Ficus

Use Case #3: Wary Traders
Katla wants to sell her stuff and so she uses Gatormmunity. She was not sure how to sell things at first, until she looked at the help page. She was then able to list her laptop for sale. Coincidentally, Jane was looking for a cheap laptop and saw Katla's listing. They agree to make the sale.



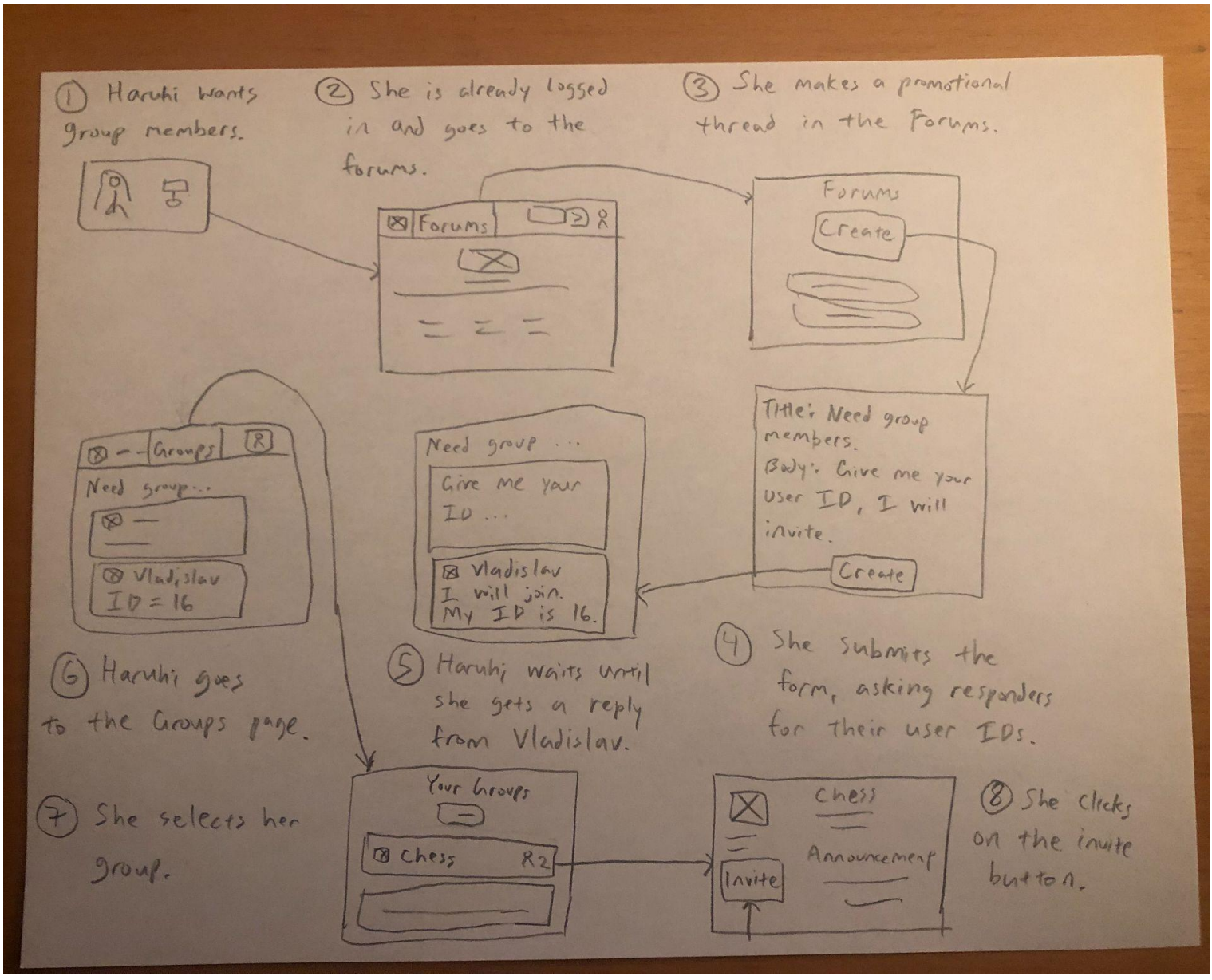

(6) She fills out the form.
Create Listing
Photo
Title Laptop
Body
Create
(7) She waits for a catch.
Options
Listings
Laptop
(8) Jane is searching for laptops so she searches for it in the search bar.
Laptop
Slogan
Jane POV
Laptop
$100
Katla
Contact
Filters
Searching Listings...
Cat.
Price
Laptop
Filters
Searching Users...
No matches!
Category
Max Price
Listing
(11) Jane contacts Katla to make the trade.
(10) Jane sees Katla's listing and selects it.
(9) She selects her filters in the sidebar and applies them in the listing search.

(12) They negotiate and finalize the deal.

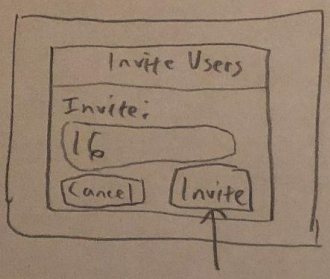


Use Case #4: Inactive Chess Club

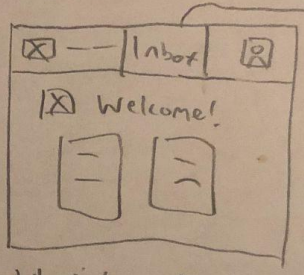
Haruhi wants to grow her Gatormmunity group, thus she makes a thread in the Gatormmunity forums hoping to attract some new members. Vladislav sees the thread and gives Haruhi his user ID, which lets Haruhi invite him. He accepts the invitation.



(9) She enters Vladislav's ID and submits.

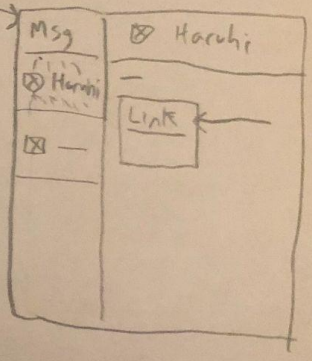


(10) Vladislav checks his inbox to see an invite link from Haruhi.

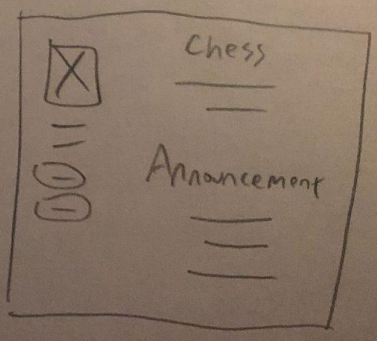


Vladislav POV

(11) Vladislav accepts the invite by clicking on the link.

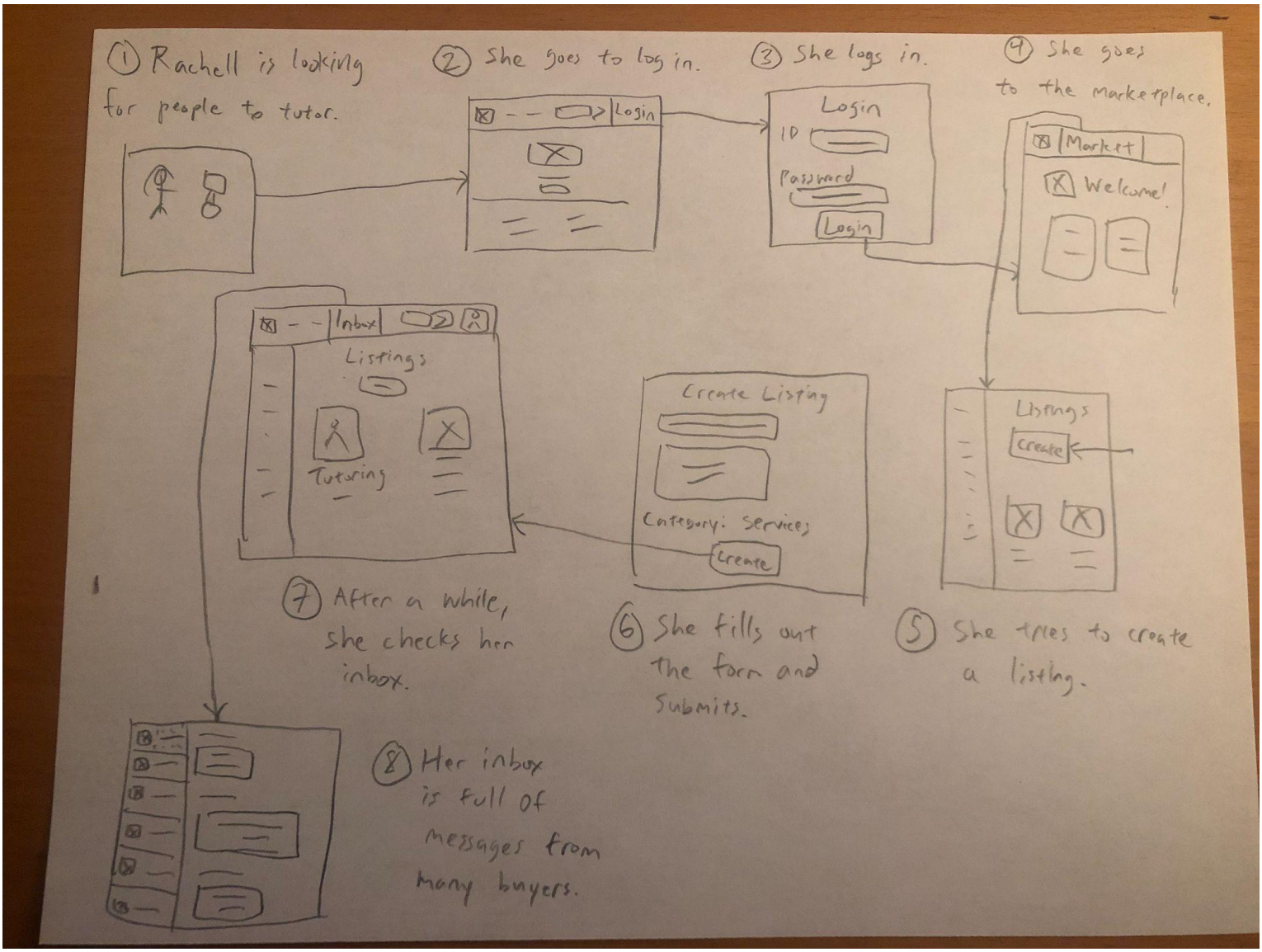


(12) Vladislav has joined Haruhi's group and sees the group's home page.

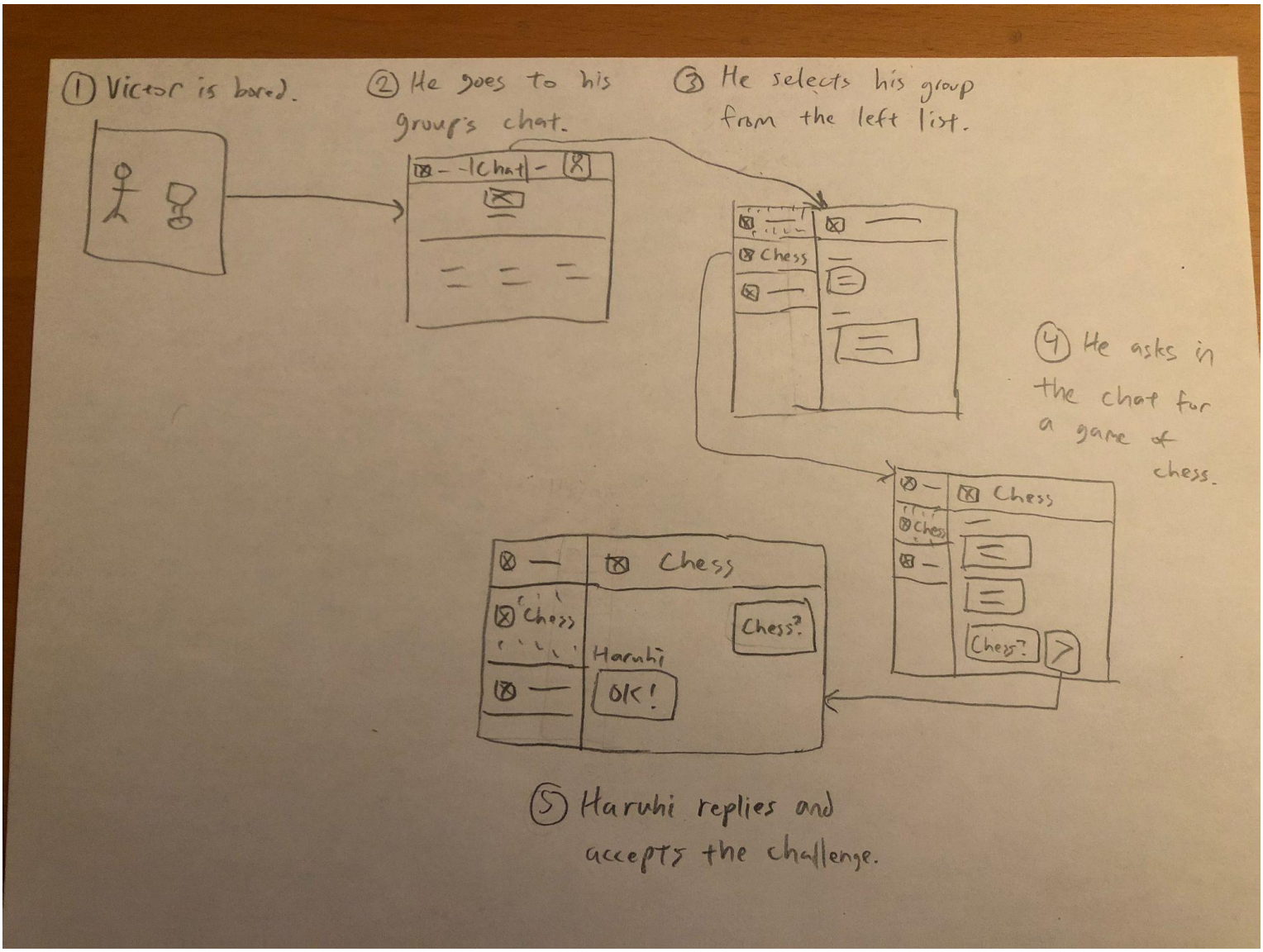


Use Case #5: Tired of Spam

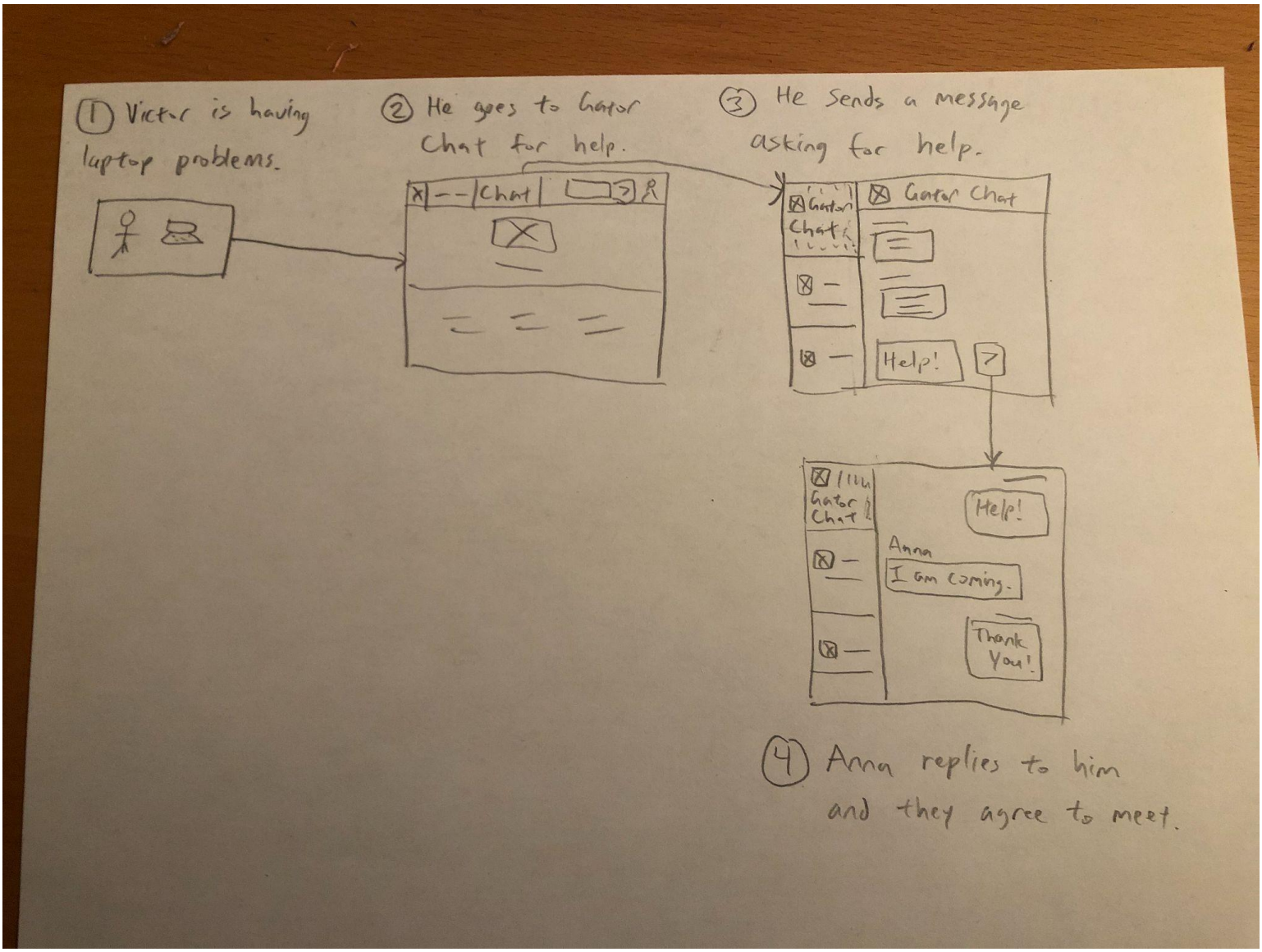
Rachell is trying to find students to tutor on Gatormmunity. She makes a listing in the marketplace offering her services and she is quickly inundated with messages from buyers who wish to pay for her services.



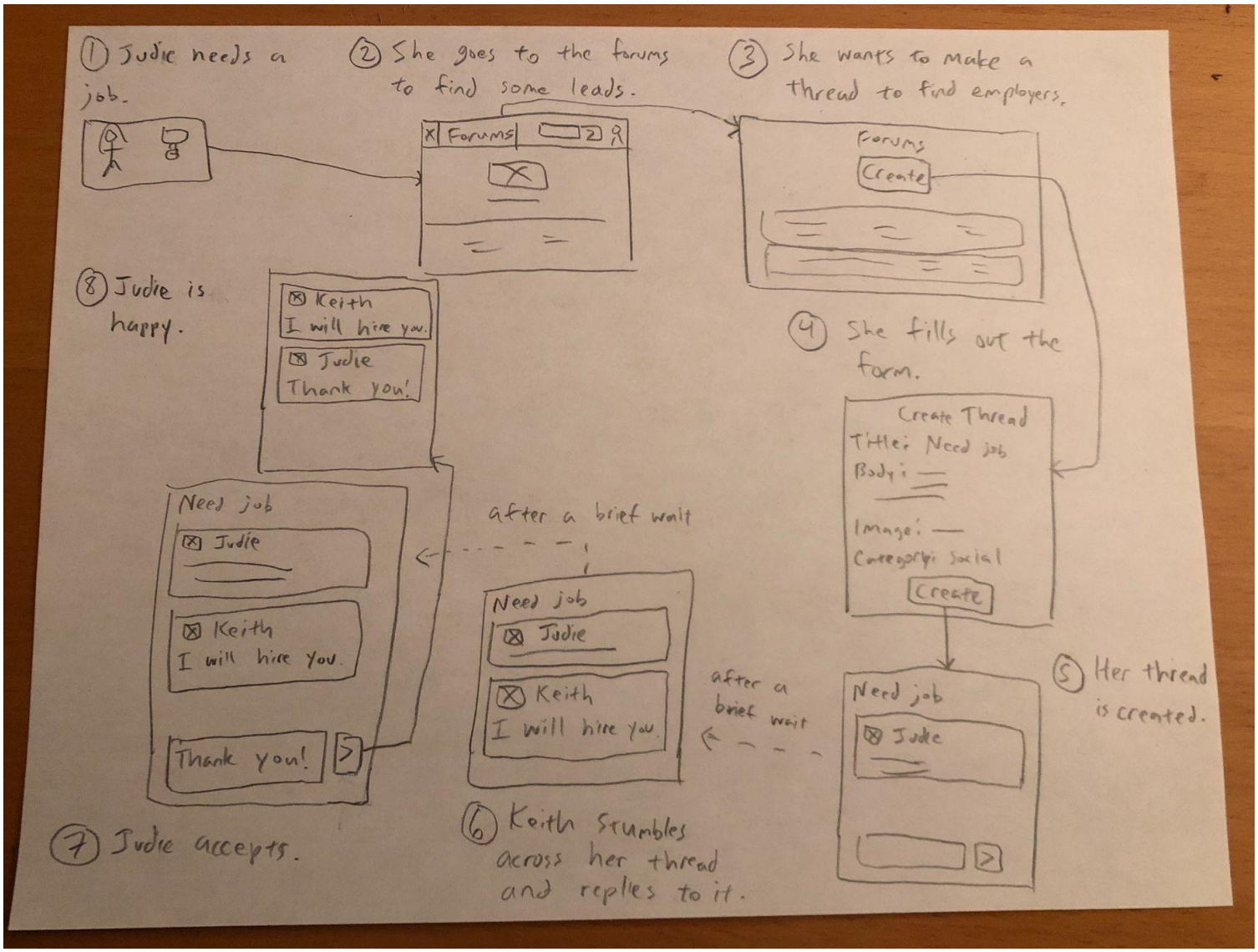
Use Case #6: A Game of Chess
Victor is bored and decides to ask in the Chess group's chat if anyone wants to play some chess. Haruhi accepts the challenge in the group chat.



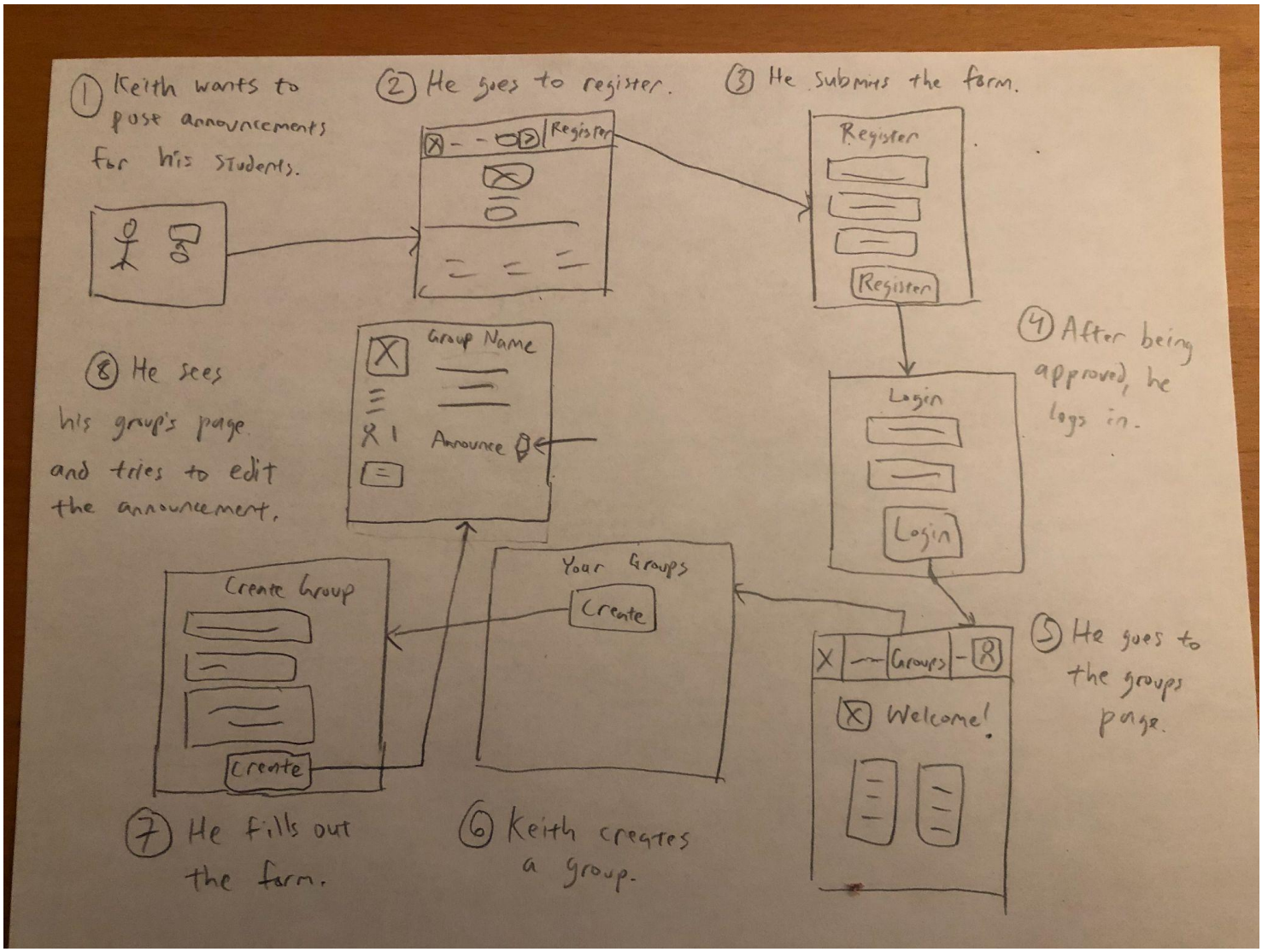
Use Case #7: Powerpoint Problems
Victor is having problems with his laptop, so he asks in Gator Chat if anyone can help. Anna notices the request and replies in the chat that she is coming to help.



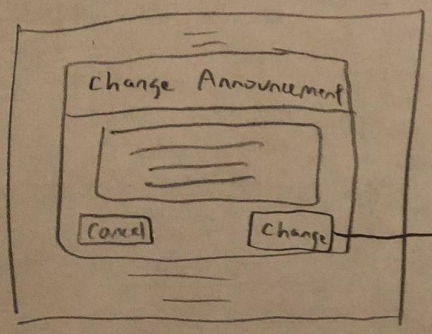
Use Case #8: An Aimless Student
Judie needs a job, therefore she creates a thread in the forums looking for work. Keith sees the thread and replies in the thread saying he can help Judie. Judie is elated.



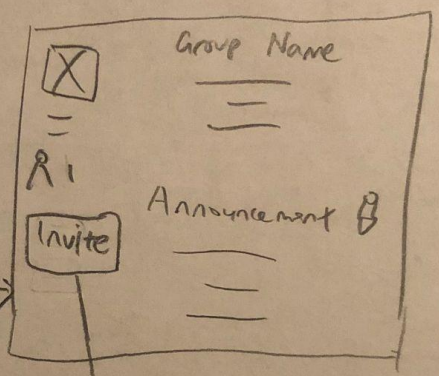
Use Case #9: A Professor's Pains
Keith heard Gatormmunity can meet his desire to post an announcement to all of his students at once, hence he registers for an account and logs in after being approved. He creates a group and posts an announcement, and then he invites his students to his group.

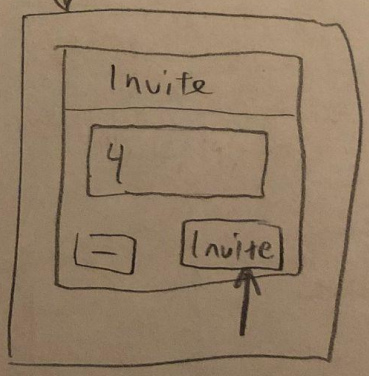


(9) He fills out the form to change the announcement.



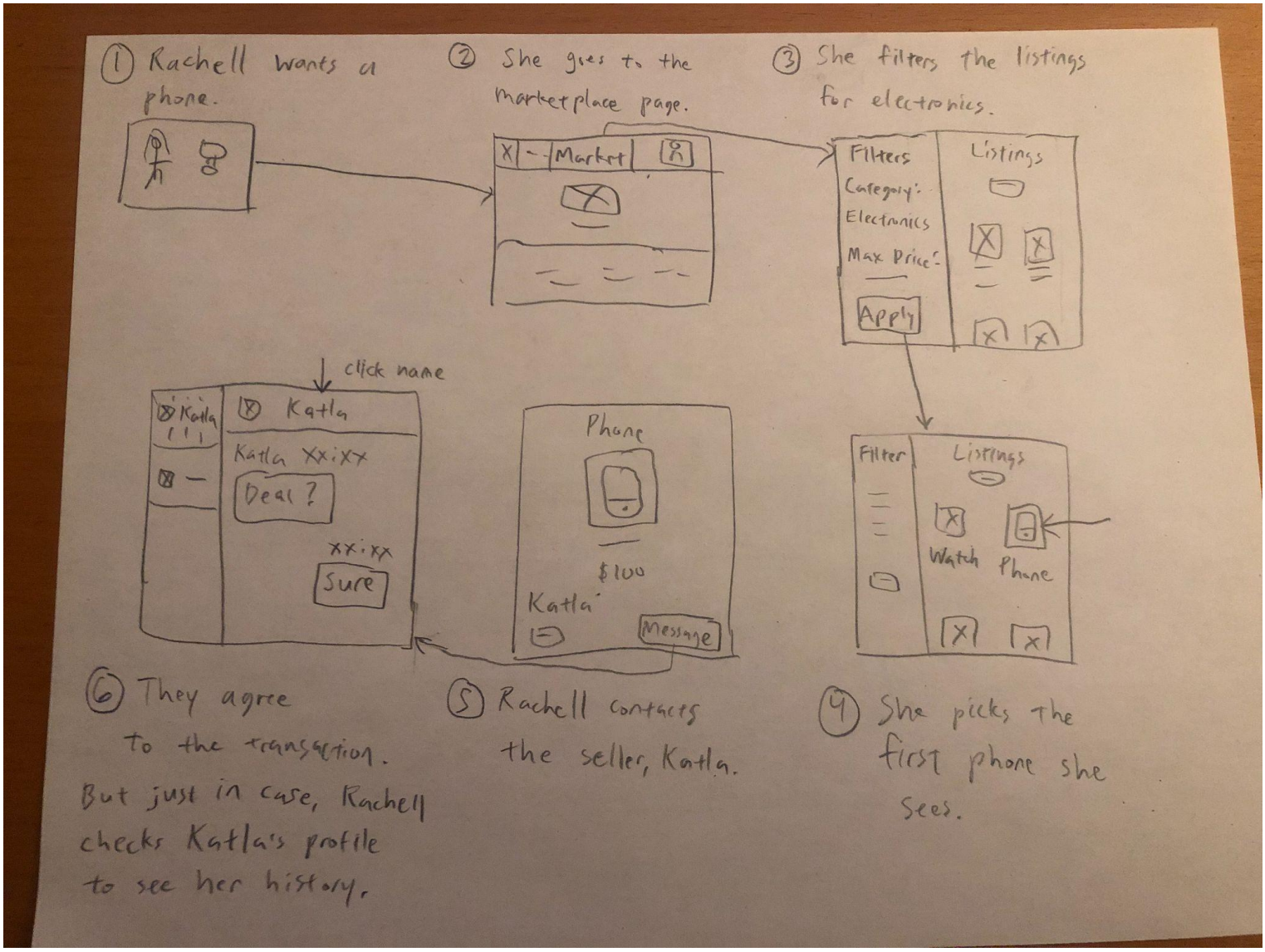
(10) He sees his announcement on the home page, then tries to invite his students to his group.



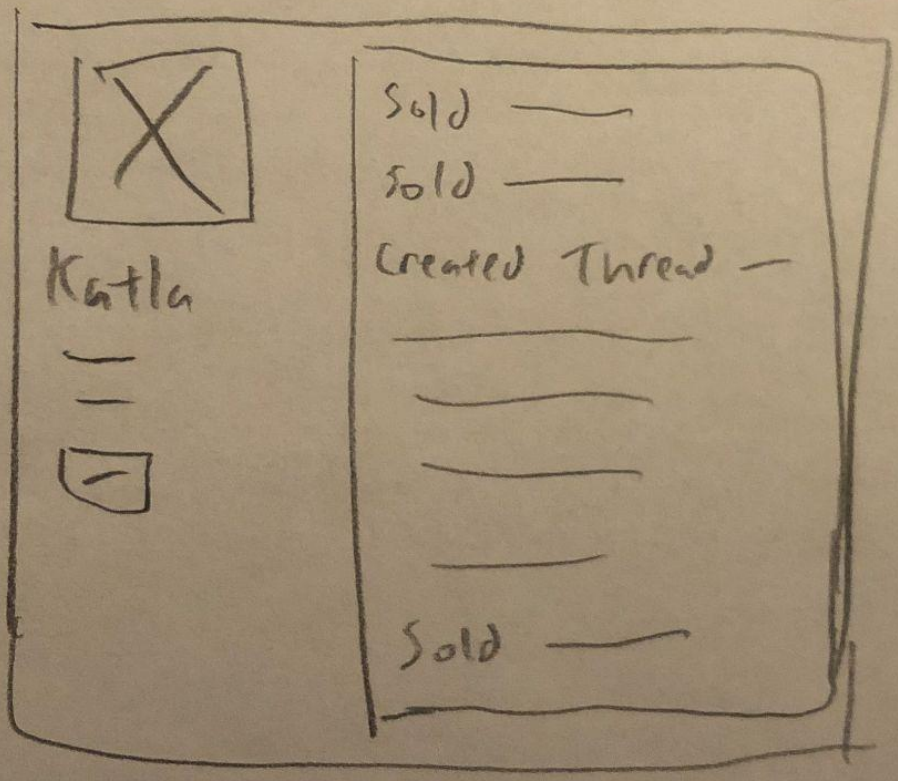


(11) He sends the invites. Soon, his students will see his announcements.

Use Case #10: Supporting the Students
Rachell wants a new phone and looks in the marketplace for electronics. Rachell sees Katla's listing and tries to buy it from her. The deal is made, but just in case, Rachell checks Katla's profile to see how often she uses Gatormmunity to gauge how trustworthy she is. Rachell finds Katla to be very trustworthy.



(7) Rachell sees Katla has an outstanding history of selling, so Rachell trusts her:



4. High Level Database Architecture and Organization

DB Organization:

1. Database Requirements:

1. A user must have one and only one account.
2. A user can ban many users.
3. A user can write many chat messages.
4. A forum can have many threads.
5. A thread must have at least one post.
6. A post can have zero to one attachments.
7. A post can be created by one and only one user.
8. A group can have many users.
9. A group can have many threads.
10. A group can have many chat messages.

2. High Level Description of our Entities, Attributes, Relationships, and Domains:

1. User (Strong)
    - 1.1. user_id: primary key, numeric
    - 1.2. first_name: alphanumeric
    - 1.3. last_name: alphanumeric
    - 1.4. email: alphanumeric
    - 1.5. sfsu_id_number: numeric
    - 1.6. sfsu_id_picture_path: alphanumeric
    - 1.7. profile_picture_path: alphanumeric
    - 1.8. profile_picture_thumbnail_path: alphanumeric
    - 1.9. role: numeric
    - 1.10. join_date: alphanumeric
    - 1.11. banned_by_id: foreign key, numeric

2. Account (Weak)
    - 2.1. account_id: primary key, numeric
    - 2.2. password: alphanumeric
    - 2.3. user_id: foreign key, numeric

3. Thread (Strong)
    - 3.1. thread_id: primary key, numeric
    - 3.2. title: alphanumeric
    - 3.3. category: alphanumeric
    - 3.4. creation_date: alphanumeric

3.5. group_id: foreign key, numeric
3.6. creator_id: foreign key, numeric

4. Post (Weak)
   4.1. post_id: primary key, numeric
   4.2. body: alphanumeric
   4.3. is_original_post: numeric
   4.4. creation_date: alphanumeric
   4.5. thread_id: foreign key, numeric
   4.6. author_id: foreign key, numeric

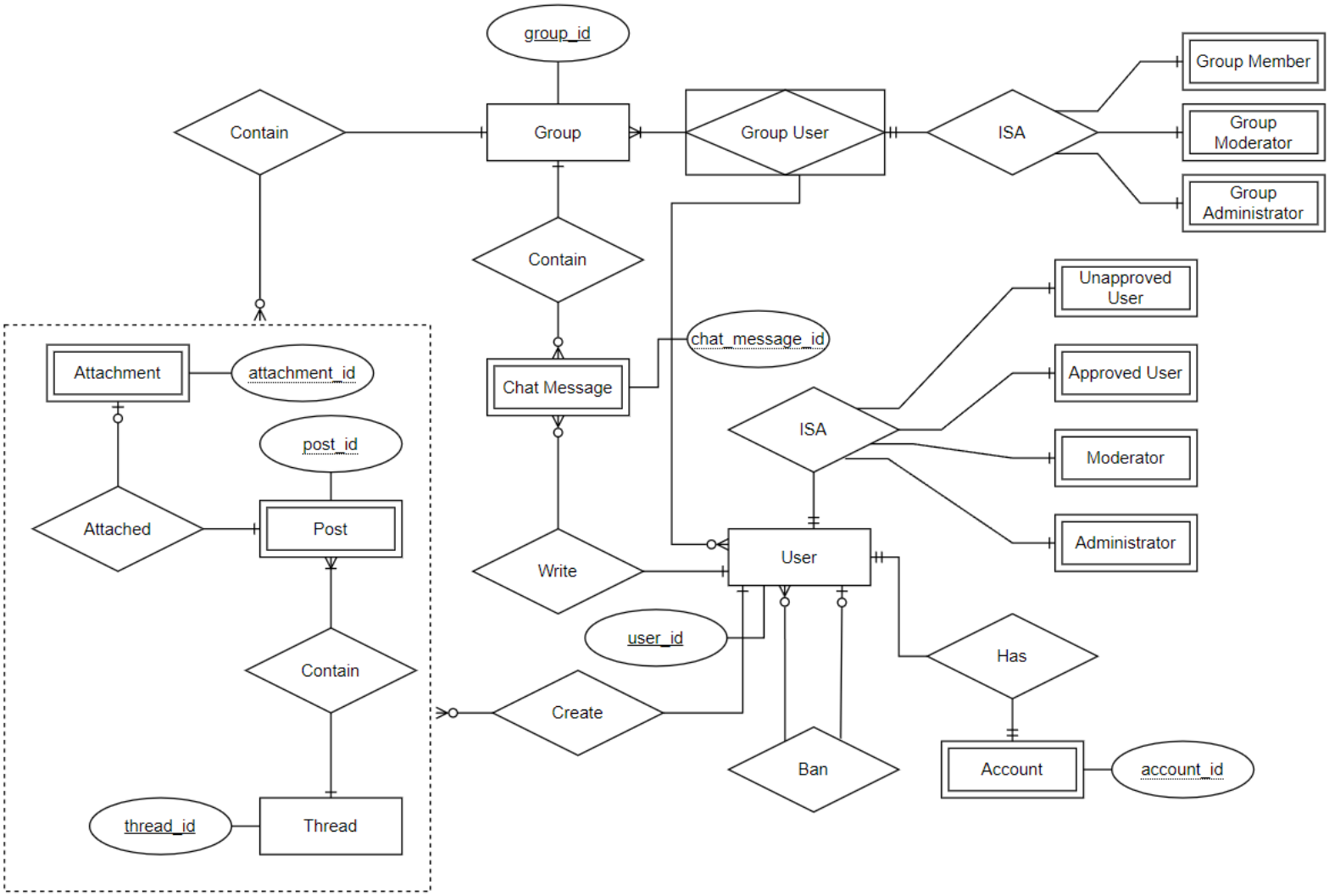
5. Attachment (Weak)
   5.1. attachment_id: primary key, numeric
   5.2. filename: alphanumeric
   5.3. image_path: alphanumeric
   5.4. thumbnail_path: alphanumeric
   5.5. post_id: foreign key, numeric

6. Group (Strong)
   6.1. group_id: primary key, numeric
   6.2. name: alphanumeric
   6.3. description: alphanumeric
   6.4. announcement: alphanumeric
   6.5. picture_path: alphanumeric
   6.6. picture_thumbnail_path: alphanumeric
   6.7. creation_date: alphanumeric

7. Group User (Weak):
   7.1. group_id: primary key, foreign key, numeric
   7.2. user_id: primary key, foreign key, numeric
   7.3. role: numeric

8. Chat Message (Weak)
   8.1. chat_message_id: primary key, numeric
   8.2. body: alphanumeric
   8.3. creation_date: alphanumeric
   8.4. group_id: foreign key, numeric
   8.5. author_id: foreign key, numeric

3. Entity Relationship Diagram (ERD):



4. Chosen DBMS:

We will use MySQL to create the database because it is the RDBMS we are most familiar with, and because we can use MySQL Workbench to check the database remotely over SSH.

Media Storage:

User-uploaded images will be stored on the remote server's file system. The database will store the path to the image, not the image file itself.

User-uploaded images will need to be of the following image formats: JPEG, PNG, WebP, GIF, and AVIF. This restriction exists because sharp, a Node.js module that creates thumbnails from images, only supports these image formats.

In addition, according to non-functional requirement 6.3, user-uploaded images can be at most 5 MB in size.

Search/Filter Architecture and Implementation:
The search algorithm will use SQL's %LIKE% operator. The search algorithm will match any users whose first or last name contains the search term that was entered into the search bar. The user may filter the search results such that it only searches for users who have a specified role (e.g. only Moderators).

We will use MySQL for querying the database, ExpressJS and JavaScript for sending the results to the front end, and ReactJS for displaying the results to the user.
We will use JavaScript to substitute any variables in the SQL query with the values the user provided, i.e. *search_term*, *X,* and *N*.

The search items will be organized in rows, similar to Facebook. Each row will display the first and last names of the matched user, their registration date, their profile picture, and their role.

The database searches the first name and last name of each user in the users table, and if the user specifies a role to filter by, the database will check the roles of the users and return only the users who are of the desired role.

Query 1) The complete SQL query for our search function. Certain parts of the query are removed depending on if a variable was provided, e.g. the role *X*.

```
SELECT CONCAT_WS(' ', first_name, last_name) AS full_name,
       profile_picture_path, profile_picture_thumbnail_path, role, join_date
FROM user
WHERE role = X
HAVING full_name LIKE "%search_terms%"
ORDER BY user_id DESC
LIMIT N
```

The line with `WHERE role = X` is removed from the query if a role filter is not selected. This means users can search for other users of any role if they do not provide a role filter, while applying the role filter means the results will consist of only users who match the role filter.
The variable X is an integer representing the role the user is looking for.
(0 = Unapproved User, 1 = Approved User, 2 = Moderator, 3 = Administrator)

The line with `HAVING full_name LIKE "%search_terms%"` is removed from the query if no search terms are provided. This means users can search for all users regardless of their name if no search terms are provided, while providing search terms means the results will consist only of users who contain *search_terms* in their full name.
The variable *search_terms* is a string representing the search terms the user provided.

Query 1 returns the *N* most recently created users who are of role *X* and whose full name contains the string *search_terms*.

Query 2) The SQL query for our search function that is called when no users were matched in Query 1. In other words, these are the “suggested/recommended” users that are shown when a user’s search results do not match anyone.

```
SELECT CONCAT_WS(' ', first_name, last_name) AS full_name,
       profile_picture_path, profile_picture_thumbnail_path, role, join_date
FROM user
ORDER BY user_id DESC
LIMIT N
```

Query 2 returns the *N* most recently created users.

5. High Level APIs and Main Algorithms

High Level APIs:

Register: The client provides a full name, SFSU email address, SFSU ID number, password, and SFSU ID card picture to create an account. The client will be notified with a message saying whether the registration was successful or not.

Login: The client provides their SFSU ID number and password to login. The server will create a cookie/session for the client to remember the client. The client will be notified if the login was unsuccessful.

Search: The client provides search terms, and the server will send back users which match the search terms at first. If there are no matches, the client will receive some of the newest registered users. The client can then specify filters or what exactly they wish to search for (users, listings, or threads) after the initial search for users. The client must be logged in to search for items other than users.

Create Forum Thread: The logged in client provides a thread title, a thread body (which will be used to automatically create the first post in the thread), an optional image, and a category. The client will be taken to their newly created thread's page on successful thread creation.

Create Forum Post: The logged in client provides a post body, and a post will be created on the thread the client is viewing. The client will see the newly created post on success, or receive a notification on failure.

Create Marketplace Listing: The logged in client provides an item photo, listing title, description, price, and category to create a listing. Upon success, the client will see their listing in the marketplace. On error, the client will be notified.

Create Group: The logged in client provides a picture, name, and an optional description to create a group. On success, the client will see their group's home page, otherwise an error message will be displayed.

Send Chat/Direct Message: The logged in client provides a message, and the message will be sent to the chat room or to the recipient. Upon success, the client will see their message; on error, the client will be notified.

Third Party APIs:

Send Email using Nodemailer: Nodemailer allows clients to send emails to our team's email address using the Contact Us form on Gatormmunity. The client provides their name, email address, an optional email subject, and email body, and an email will be sent to our team's email upon submission. Nodemailer uses the SMTP protocol to send the email.

Main Algorithms:

User Search: We will create a search algorithm for users. The search will return all the users with an account whose first or last name contains the search term the user entered. The user can search for users of any role, or a specific role using a role filter.

Marketplace Listing Search: We will create a search algorithm for marketplace listings. The search will return all marketplace listings whose title or description contains the search term the user entered. The user can search for a specific category of items using a category filter. The user can also filter the listings by specifying a max price.

Forum Thread Search: We will create a search algorithm for forum threads. The search will return all forum threads whose title contains the search terms the user entered. The user can filter the forum threads by category using a category filter.

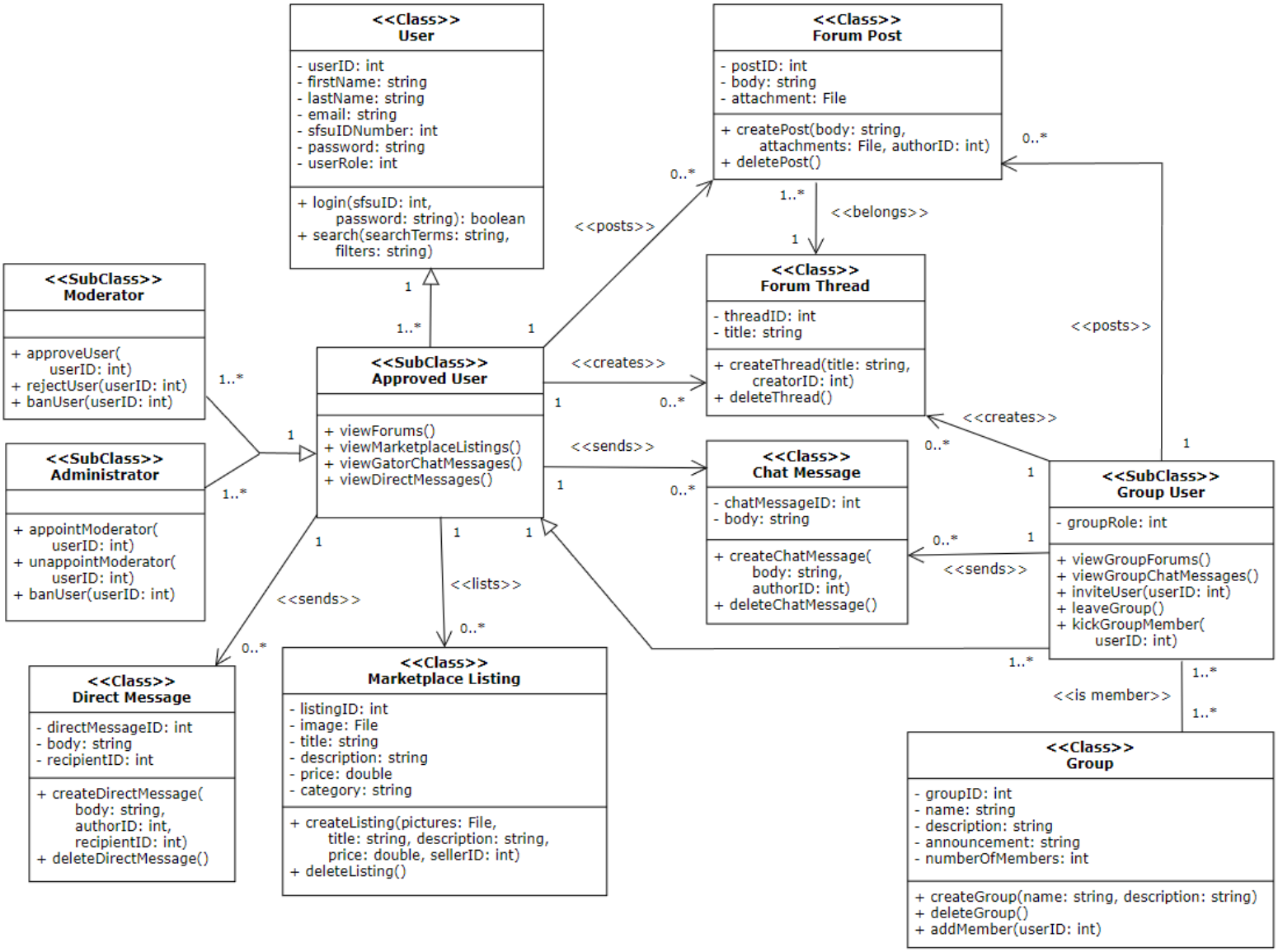
Forum Thread Sort: We may create a sorting algorithm for forum threads so that the user can see forum threads sorted by creation date, last reply date, or by thread title. The creation date and last reply date sort will be from newest to oldest or oldest to newest, while the thread title sort will be based on the alphabetical order of the title: from A to Z or from Z to A. This is not a priority 1 item.

Forum Thread Pins: We may create a process that allows pinned forum threads to display at the top of the list of forum threads, regardless of the forum thread sort option used. This is not a priority 1 item.

Seller Feedback/Rating: We may create an algorithm for determining the rating of a seller. Approved users may leave ratings or reviews on sellers they have purchased items from. Sellers will have ratings that are calculated by taking the average of the rating scores they have received from their buyers' feedback. This is not a priority 1 item.

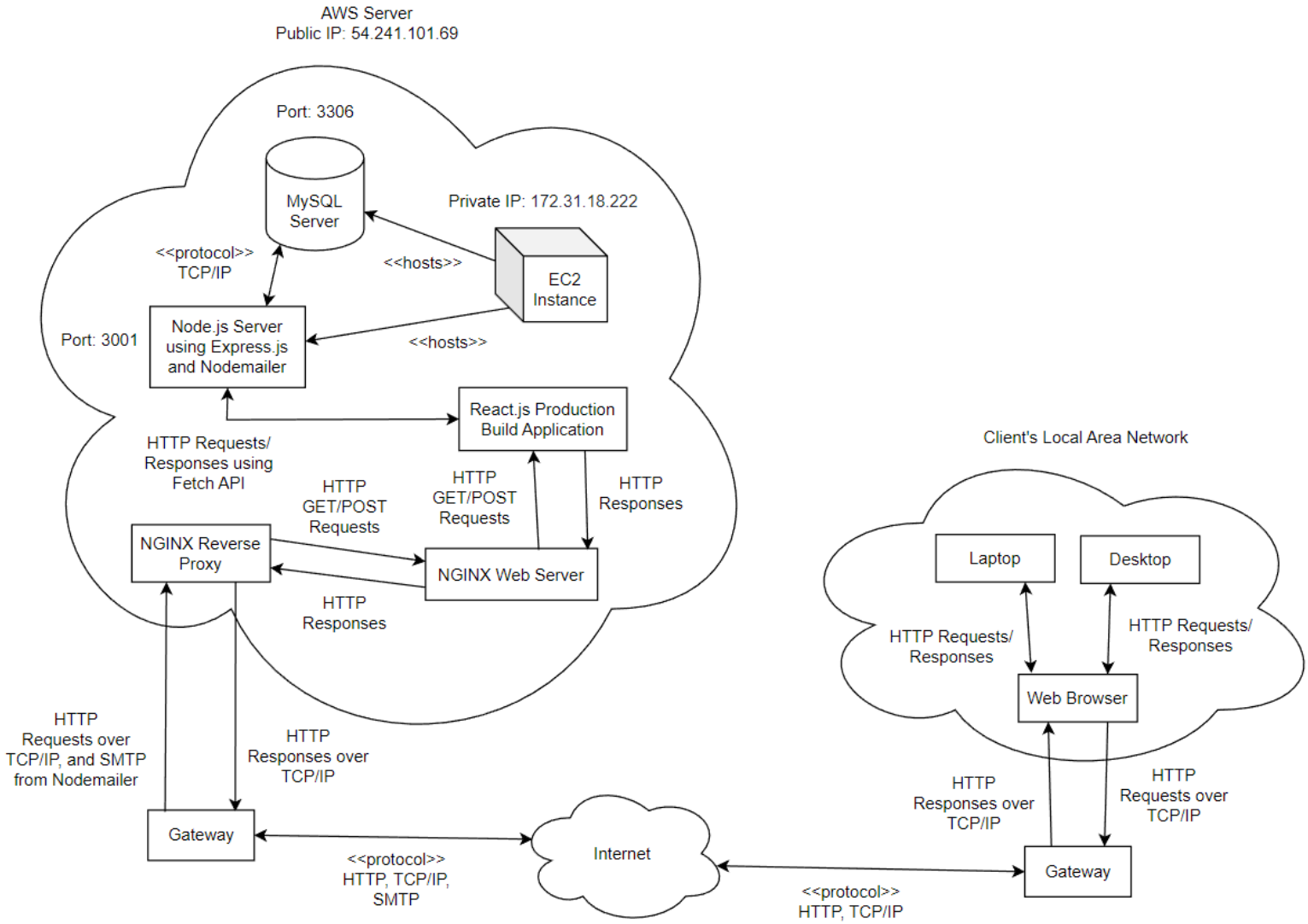
6. High Level UML Diagram

UML Class Diagram:

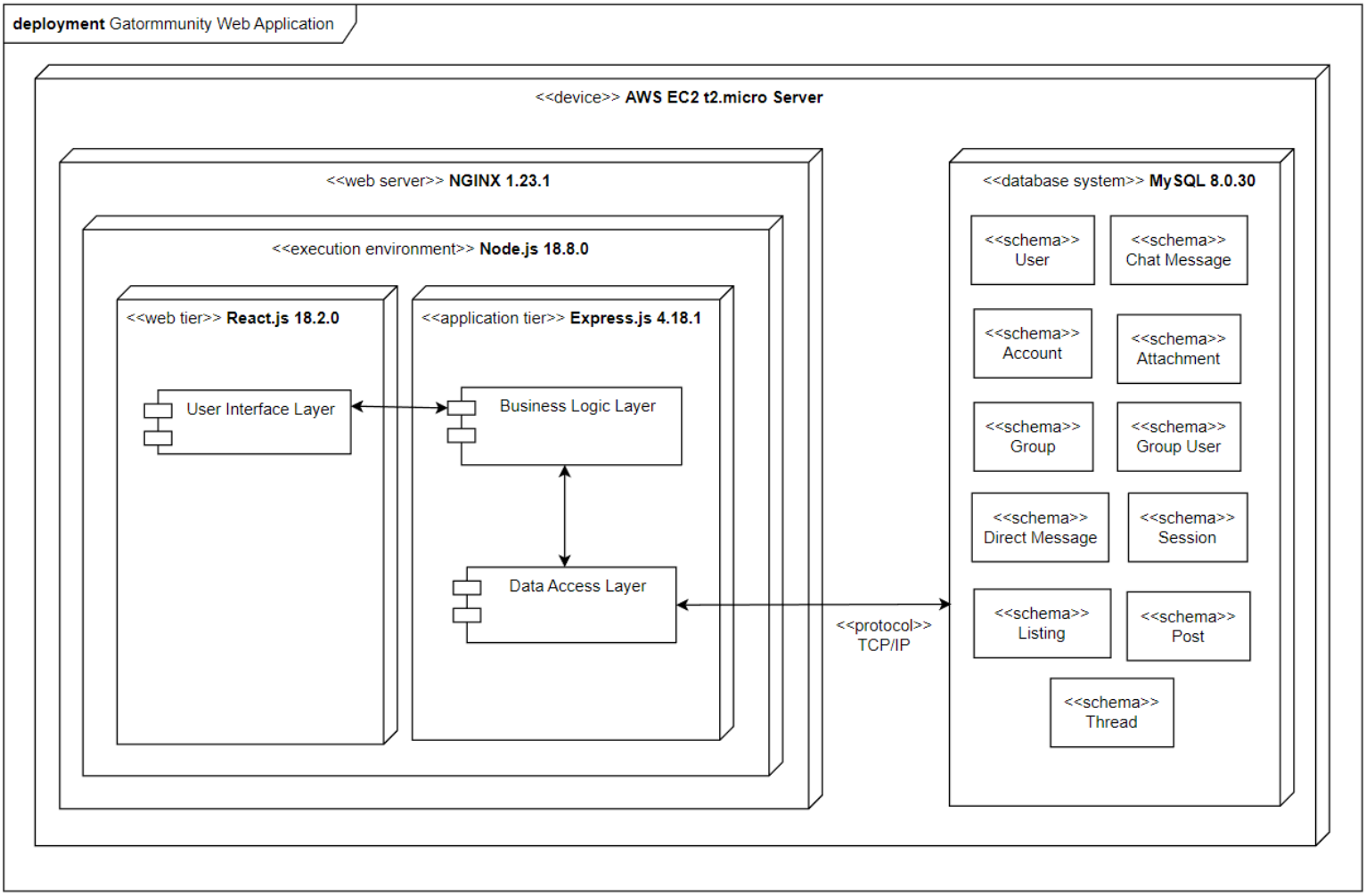


7. High Level Application Network and Deployment Diagrams

Application Network Diagram:



Deployment Diagram:



8. Identify Actual Key Risks for Your Project at This Time

Skills:

- The front end team is not completely confident in their abilities in React and they are rusty in JavaScript, which will slow their progress. To solve this, they will refresh themselves in React and JavaScript so that they can work unimpeded.

- Most of the team has not completed a course in databases. We will struggle with creating complex queries and our database model may be incorrect. To solve this, team members who have taken or are currently taking databases should help members who are not as familiar with databases.

Schedule:

- We are halfway through the semester, yet we only have a home page so far. We may not be able to implement every desired functional requirement before the semester ends. To solve this, we will focus mainly on our priority 1 functional requirements so that we can create a good application before the deadline.

Technical:

- We will face setbacks developing our application because of its complexity. We will have bugs and other unexpected problems. To solve this, we will have free team members assist struggling team members with debugging and by providing technical support.

Teamwork:

- We have had conflicts within the team and we will have more because it is inevitable that a team will have disagreements. To solve this, we will try to negotiate to solve the conflict, but if no solution can be reached, then we will escalate to the CTO.

Legal/Content Risks:

- We want to have pictures on our website but we also do not want to be sued for copyright infringement. To solve this, we will look online for copyright-free images that are safe for us to use, or we can draw our own images.

9. Project Management

We used Trello and Discord to keep track of and manage our assigned tasks. We used Trello to store our task descriptions, deadlines, task categories, as well as to show who was assigned to which task. We used Discord to notify the team of when a task was assigned since Trello does not seem to send a notification when a new task has been assigned.

We categorized our tasks by main tasks, sub tasks, and pending tasks. Our main tasks are high-level and will require the team's effort to complete, e.g. complete the vertical prototype's front end. Sub tasks are short-term and contribute to the completion of a main task, e.g. complete the registration feature. Pending tasks are ones that are finished and are awaiting feedback or require further changes, or are tasks that are unable to be completed due to us missing prerequisite information.

We will manage future tasks in the same way as we are now. We will continue to use Trello and Discord to manage our tasks, though we may introduce a new category for tasks, so that we can distinguish a task that will be assigned in the future from one that is pending feedback or changes.

10. Detailed List of Contributions

List of Detailed Contributions Made by Each Team Member in M2:

Anthony Zhang:

- Left feedback on the team's work using Discord, Google Doc comments, and GitHub comments
- Attended every scheduled team meeting in M2, and attended the unscheduled team meeting
- Active participant in team meetings: asks questions, answers questions, proposes ideas, and gives feedback
- Active participant in the team's Discord group: answers questions and gives feedback
- Helped apply the instructor's feedback to M1V2's Document
- Helped expand our data definitions for Part 1 of M2's Document
- Helped prioritize our functional requirements for Part 2 of M2's Document
- Drew the UI mockups for Part 3 of M2's Document
- Edited some of the storyboards for Part 3 of M2's document
- Justified the DBMS we would use for Part 4 of M2's document
- Agreed with the decision to keep images/files in file system for Part 4 of M2's document
- Helped define the search/filter architecture and implementation for Part 4 of M2's document
- Described APIs we have and will create for Part 5 of M2's document
- Described algorithms and processes for Part 5 of M2's document
- Created the UML class diagram for Part 6 of M2's document
- Created the applications network diagram for Part 7 of M2's document
- Created the deployment diagram for Part 7 of M2's document
- Helped determine our key risks for Part 8 of M2's Document
- Described how we managed M2's tasks for Part 9 of M2's document
- Listed the team's contributions for Part 10 of M2's document
- Sent the email to Ortiz for Part 10 of M2's Document
- Set up the front end infrastructure, allowing the team to start front end development
- Created the navbar and footer
- Created the login, registration, and search forms for the front end
- Created the login and registration modals for the front end
- Helped create the search results modal for the front end
- Created the authentication code that checks if a user is logged in for the front end
- Helped create and style the home page
- Added comments to the front end code, and added documentation for the front end endpoint API
- Created the database schema based on the ERD
- Adjusted the login route and related functions for the back end
- Created the authenticate route for the back end, which checks if a user is logged in
- Adjusted the server-side input validation code for the registration form
- Added comments to the back end code, and added documentation for the back end endpoint API

Marwan Alnounou:

- Attended every scheduled team meeting in M2
- Active participant in team meetings: asks questions, answers questions, and proposes ideas
- Agreed with the team's revisions to M1V2's document
- Helped expand our data definitions for Part 1 of M2's Document
- Helped prioritize our functional requirements for Part 2 of M2's Document
- Created 5 storyboards for Part 3 of M2's Document, and redid them multiple times to apply feedback
- Agreed with the decision to keep images/files in file system for Part 4 of M2's document
- Helped determine our key risks for Part 8 of M2's Document
- Sent the email to Ortiz for Part 10 of M2's Document
- Helped create the search results modal for the front end
- Fixed the home page's HTML and CSS

Mohamed Sharif:

- Attended almost every scheduled team meeting in M2, and attended the unscheduled team meeting
- Active participant in team meetings: asks questions, answers questions, proposes ideas, and gives feedback
- Active participant in the team's Discord group: asks questions, answers questions, proposes ideas, and gives feedback
- Helped apply the instructor's feedback to M1V2's Document
- Helped expand our data definitions for Part 1 of M2's Document
- Helped prioritize our functional requirements for Part 2 of M2's Document
- Agreed with the decision to keep images/files in file system for Part 4 of M2's document
- Helped determine our key risks for Part 8 of M2's Document
- Sent the email to Ortiz for Part 10 of M2's Document
- Set up the back end infrastructure, allowing the team to start back end development
- Connected the Express back end to the MySQL database
- Set up sessions for the back end, which allows users to stay logged in
- Created the registration route and related functions for the back end
- Helped create the server-side input validation code for the registration form
- Created the login route and related functions for the back end
- Created the logout route for the back end
- Did debugging for the back end

Jose Lopez:

- Attended every scheduled team meeting in M2, and attended the unscheduled team meeting
- Active participant in team meetings: asks questions, answers questions, proposes ideas, and gives feedback
- Helped apply the instructor's feedback to M1V2's Document

- Helped expand our data definitions for Part 1 of M2's Document
- Helped prioritize our functional requirements for Part 2 of M2's Document
- Created 5 storyboards for Part 3 of M2's Document, and redid them multiple times to apply feedback
- Agreed with the decision to keep images/files in file system for Part 4 of M2's document
- Helped determine our key risks for Part 8 of M2's Document
- Created numerous images for our application, including logos, a background image, and a slogan
- Created the home page
- Helped style the home page
- Helped style the navbar and footer

Florian Cartozo:

- Taught the team how to follow GitHub's best practices, e.g. creating feature branches and creating pull requests into the development branch
- Attended almost every scheduled team meeting in M2
- Agreed with the team's revisions to M1V2's document
- Helped expand our data definitions for Part 1 of M2's Document
- Helped prioritize our functional requirements for Part 2 of M2's Document
- Defined the database requirements for Part 4 of M2's Document
- Described the database entities for Part 4 of M2's Document
- Created the ERD for Part 4 of M2's Document, and redid it multiple times to apply feedback
- Agreed with the decision to keep images/files in file system for Part 4 of M2's document
- Helped define the search/filter architecture and implementation for Part 4 of M2's document
- Helped determine our key risks for Part 8 of M2's Document
- Sent the email to Ortiz for Part 10 of M2's Document
- Helped create the server-side input validation code for the registration form
- Created the search route and related functions for the back end

Contribution Scores:

Anthony Zhang: 10
Marwan Alnounou: 5
Mohamed Sharif: 10
Jose Lopez: 7
Florian Cartozo: 7

SW Engineering CSC648/848 Fall 2022
Gatormmunity

Team 7: Team 7

Anthony Zhang (Team Lead),
azhang12@mail.sfsu.edu

Marwan Alnounou

Mohamed Sharif

Jose Lopez

Florian Cartozo

Milestone 3
12/14/2022

History Table

Version	Submission Date
M3V1	11/10/2022
M3V2	12/14/2022

Table of Contents

1. Data Definitions V3

Types of Users:

All Users: Any person who visits Gatormmunity's website, whether they are logged in or not.

- All Users can see the home page, login page, and register page, but what they can do on each of these pages depends on whether they are logged in or not.

Guest User: A person that does not have an account.

- Guest Users can only see the pages that do not require one to be logged in for, e.g. the home page, registration page, login page, and about us page.
- Guest Users can register for an account by providing a full name, SFSU email address, SFSU ID number, SFSU ID picture, and password.
    - The picture must be of an image format. Supported formats are: JPEG, PNG, WebP, GIF, and AVIF. The picture must be at most 5 MB in size.

Unapproved User: A person that registered for an account and has not yet had their account approved by a moderator or admin.

- Unapproved Users have the same privileges as Guest Users, meaning they can only see the pages that do not require one to be logged in for.
- Unapproved Users cannot login until their account has been approved by a moderator or admin.
- Unapproved Users have the same attributes as any other user stored in the user table in the database: a first name, last name, email, SFSU ID number, SFSU ID picture, profile picture, role, and join date.
    - For unapproved users, role = 0.

Approved User: A person that registered for an account and has had their account approved by a moderator or admin. All logged in users are approved users, since unapproved users cannot login.

- Approved Users have access to almost every page and feature in our application, except for moderation tools and group-exclusive features. They can create forum threads and posts and post marketplace listings, for example.
- Approved Users can log in to their account by providing their SFSU ID number and password.
- Approved Users have the same attributes as any other user stored in the user table in the database: a first name, last name, email, SFSU ID number, SFSU ID picture, profile picture, role, and join date.
    - For approved users, role = 1.

Moderator / Mod: An approved user that has moderation powers in Gatormmunity.

- Moderators have more privileges than Approved Users, and have access to moderation tools, including the ability to approve/reject Unapproved Users, and ban Unapproved and Approved Users from Gatormmunity via the UI.

- The role and its associated privileges are determined by the role attribute stored in the database for each user. To be able to access the moderation tools in the UI, a user needs to have the correct role attribute value in the database. The back end will check the user id of the user to ensure that they are of the correct role before executing any moderation actions.
- Moderators are appointed and unappointed by Server Administrators using either the Gatormmunity website or through directly modifying a user's role in the database.
- No users will be able to determine their role upon registering like they could in the Vertical Prototype.
- Moderators have the same attributes as any other user stored in the user table in the database: a first name, last name, email, SFSU ID number, SFSU ID picture, profile picture, role, and join date.
    - For moderators, role = 2.

Administrator / Admin: An approved user who has access to Gatormmunity's server, database, and GitHub repository.

- Administrators have more privileges than Moderators, and have access to administrative tools, including the ability to appoint moderators and ban them.
- Similar to moderators, the back end will check if the user performing the administrative actions is of the correct role before executing any administrative actions.
- Administrators are appointed by the developers (the team).
- Administrators have the same attributes as any other user stored in the user table in the database: a first name, last name, email, SFSU ID number, SFSU ID picture, profile picture, role, and join date.
    - For administrators, role = 3.

Group Member: An approved user who is a member of a group. Group Members are stored in the Group User table, an associative table between Group and User.

- Group Members have the same privileges as Approved Users, and have access to group-exclusive features such as creating forum threads in their group's forum or sending messages in their group's chat.
- Approved Users become Group Members when they join a group via an invite from another Group Member.
- Group Members are differentiated from the other Group User entities by means of their role attribute in the Group User table. For group members, role = 1.

Group Moderator / Group Mod: An approved user who is a moderator of a group. Group Moderators are stored in the Group User table, also.

- Group Moderators have more privileges than Group Members, and have access to group moderation tools, including the ability to kick group members from the group via the UI.
- Group Moderators are appointed and unappointed by Group Administrators in the UI.
- Group Moderators are differentiated from the other Group User entities by means of their role attribute in the Group User table. For group moderators, role = 2.

Group Administrator / Group Admin: An approved user who is an administrator of a group. Group Administrators are stored in the Group User table, also.

- Group Administrators have more privileges than Group Moderators, and have access to group administrative tools, including the ability to delete their group and appoint/unappoint Group Moderators via the UI.
- Group Administrators are differentiated from the other Group User entities by means of their role attribute in the Group User table. For group administrators, role = 3.

User Interface Terms:

Navigation Bar / Navbar: The navigation bar will be at the top of every page and will contain buttons that link to different pages within our application and a search bar.

- Only logged in users can see these buttons in the navigation bar. Users who are not logged in can only see the logo, buttons that link to the login or registration page, and the search bar.
- The navbar no longer queries the database as it did in the Horizontal Prototype. When a user uses the search bar in the navbar, the navbar redirects the user to the search page with the provided search terms as parameters in the URL. The search page then performs the search by fetching from the back end.
- The navbar displays the profile picture of the user if they are logged in. The profile picture's URL is retrieved from the back end which the client then uses to display the image.
- The navbar shows an additional link to the Administration page which is visible and accessible only to moderators and administrators.

Dashboard: The page that approved users will see after logging in, similar to a home page for only logged in users. The dashboard page is also accessible via a dropdown from the profile picture in the navbar if the user is logged in.

- The dashboard will show the user's profile picture, name, the newest forum threads in the Gatormmunity forum, and the newest marketplace listings. It is intended to catch the user up on what is happening right now in Gatormmunity and the threads/listings displayed may catch the user's eye.
- This page is only visible to logged in users. The code does this by checking the back end if the user is logged in before rendering the content in the page.

Gatormmunity Forum:

Gatormmunity Forum / Forum: A forum where all approved users can start their own forum threads and make forum posts in existing threads. The Gatormmunity Forums page will show a list of threads that are of the category the user specified. The threads are retrieved from the database, and the filtering will be done on the server's side.

- Only logged in users may see this page.

Forum Category: Threads belong to categories which are intended to help approved users find threads that they are interested in. Approved users can specify which category of threads they would like to see on the Gatormmunity Forums page. By default, no category filter is specified.

- Example categories: General, Social, Questions

Forum Thread / Thread: A collection of posts. Threads have a title which is determined by the person starting the thread. Threads may belong to a group, and if they are, they will only be visible in that group's forum. Forum threads are comparable to discussion topics on iLearn, which other students can reply to.

- Forum threads can be clicked on, which will lead to the View Thread page, which shows the posts other approved users have made in reply to the thread.

Forum Post / Post: A message that approved users can post in a thread. Forum posts are comparable to the replies in a discussion topic on iLearn, including the post that started the discussion topic. Every forum post must have content and is associated with a thread and a user (the author) by means of a thread id and user id.

- The first post that starts a thread may also be referred to as the "original post". The original post is automatically created when a user creates a thread, since the create thread form asks for the body of the first post as well as the thread's title.
- Only the original (first) post of a thread may contain an image. All other posts in a thread will not be able to have images.

Post Attachment: Approved users can attach an image to the first post of the thread when creating a thread. No other post in a thread may contain an image. The image's filename on the server will be randomized in order to avoid name clashes, but the filename the user gave their file will be saved in the database and displayed next to the image in the post.

- The image must be of an image format. Supported formats are: JPEG, PNG, WebP, GIF, and AVIF.
- The image must be at most 5 MB in size.

Pin: A pinned forum thread will always appear first when displaying the list of threads. This is not a priority 1 item.

Bookmark: Approved users can bookmark forum threads, which saves the thread into the user's bookmarks. Bookmarks exist to help the user easily find threads that they want to go back to in the future. This is not a priority 1 item.

Gatormmunity Marketplace:

Gatormmunity Marketplace / Marketplace: A page within the application where approved users can buy and sell goods and services. An individual good/service being sold is called a listing.

- Only logged in users may see this page.
- Approved users may filter what they see in the marketplace page by inputting a category filter and max price filter. The category filter, when applied, shows only listings that match the category specified, while the max price filter only shows listings that do not exceed the price entered. Both filters may be used at the same time.
- The listings shown on this page are retrieved from the Listing table in the database.

Marketplace Buyer / Buyer: An approved user who is buying something from Gatormmunity's marketplace. There is nothing special about a buyer compared to an approved user, it is just terminology that refers to the person making the purchase.

Marketplace Seller / Seller: An approved user who is selling something on Gatormmunity's marketplace. The listing that the seller makes will show the seller's name and email, which the buyer may use to contact them with.

Marketplace Listing / Listing: An item or service a seller is trying to sell on the marketplace. It must have a photo of the item being sold, a description, a title, a price, and a category. The seller's contact information (email and direct message option) will be automatically included in the listing.

- The listing will not need to be approved before being listed, but approved users can report listings and moderators can delete listings that break the rules.
- The photo must be of an image format. Supported formats are: JPEG, PNG, WebP, GIF, and AVIF. The image must be at most 5 MB in size.

Community Features:

Gator Chat: A chat room for all approved users of Gatormmunity. Approved users can send messages and receive messages from other approved users in the chat room. No images can be attached to the messages.

- The chat room is accessible from the Chat button in the navbar, and is only accessible to logged in, approved users. Every message must contain some text in its body.

Direct Message / DM: A private method of communication between two approved users. No images can be attached to the messages.

- An approved user's direct messages are accessible from the Inbox button in the navbar, and an approved user must be logged in to see their direct messages. Every message must contain some text in its body and every message is associated with a pair of users: the sender and recipient. The pair of users is known as a "conversation" in the database.

User Groups / Groups: Approved users can form their own groups which come with exclusive features such as a group-exclusive chat and forum.

- Members of a group can have 3 roles: Group Member, Group Moderator, or Group Administrator.
- Groups must have a picture that is of an accepted image format. Supported formats are: JPEG, PNG, WebP, GIF, and AVIF. The group picture must be at most 5 MB in size.
- Groups can have a description and announcement, which are displayed on the group's home page. It serves to tell new group members about the group's purpose and rules, as well as show a message the group admin would like all of its members to see on the home page.

Group Chat: A chat room for all members of a group. All group members can send messages and receive messages from other group members in the group chat room. No images can be attached to these messages.

- The group's chat is accessible in the navbar, via the Chat button. Only logged in users who are members of the group can see a group's chat and send messages in it.
- Each message must have content, and each message is associated with a sender (a user) via their user id.

2. Functional Requirements V3

Priority 1:

1. All Users
    - 1.1. All users shall be able to contact the developers via the Contact Us page.
    - 1.2. All users shall be able to view the home page.
    - 1.3. All users shall be able to view the registration page.
    - 1.4. All users shall be able to view the login page.
    - 1.5. All users shall be able to view the about page to learn about Gatormmunity.

2. Guest User
    - 2.1. A guest user shall be able to view the registration page.
    - 2.2. A guest user shall be able to view the login page.
    - 2.3. A guest user shall be able to register for an account.

3. Approved User
    - 3.1. An approved user shall be able to log in to their account.
    - 3.2. An approved user shall be able to reset their password.
    - 3.3. An approved user shall be able to report other users via the Contact Us page.
    - 3.4. An approved user shall be able to create a forum thread.
    - 3.5. An approved user shall be able to assign their forum thread to a category.
    - 3.6. An approved user shall be able to create a forum post.
    - 3.7. An approved user shall be able to attach an image when creating a thread. The image will be displayed in the first post of the thread.
    - 3.8. An approved user shall have a profile page.
    - 3.9. An approved user's profile page shall show their details, e.g. their full name.
    - 3.10. An approved user's profile page shall show their recent activities, e.g. their recently created listings or threads.
    - 3.11. An approved user shall be able to view a user's profile page.
    - 3.12. An approved user shall have a profile picture.
    - 3.13. An approved user shall be able to change their profile picture.
    - 3.14. An approved user shall be able to create a user group.
    - 3.15. An approved user shall be able to join a user group after being invited to it.
    - 3.16. An approved user shall be able to send direct messages to other users.
    - 3.17. An approved user shall be able to receive direct messages from other approved users.
    - 3.18. An approved user shall be able to send messages in Gator Chat.
    - 3.19. An approved user shall be able to see marketplace listings in the marketplace.
    - 3.20. An approved user shall be able to select marketplace listings in the marketplace.
    - 3.21. An approved user shall be able to report marketplace listings via the Contact Us page.
    - 3.22. An approved user shall be able to buy items from the marketplace.
    - 3.23. An approved user shall be able to sell items on the marketplace by creating a listing.
    - 3.24. An approved user who is selling an item shall be able to assign their listing to a category.

3.25. An approved user shall be able to upload a picture of an item they are selling.
3.26. An approved user shall be able to delete their own listings from the marketplace.
3.27. An approved user shall be able to search for users.
3.28. An approved user shall be able to search for marketplace listings.
3.29. An approved user shall be able to search for forum threads.
3.30. An approved user shall be able to apply filters to their user search results.
3.31. An approved user shall be able to apply filters to their marketplace search results.
3.32. An approved user shall be able to apply filters to their forum thread search results.

4. Moderator
   4.1. A moderator shall be able to approve unapproved users.
   4.2. A moderator shall be able to reject unapproved users.
   4.3. A moderator shall be able to ban unapproved and approved users from Gatormmunity.
   4.4. A moderator shall be able to delete forum threads.
   4.5. A moderator shall be able to delete forum posts.
   4.6. A moderator shall be able to delete listings in the marketplace.

5. Administrator
   5.1. An administrator shall be able to appoint moderators.
   5.2. An administrator shall be able to unappoint moderators.
   5.3. An administrator shall be able to ban moderators from Gatormmunity by first unappointing the moderator, then banning them.
   5.4. An administrator shall be able to change a user's password, if the user is not an administrator.

6. Group Member
   6.1. A group member shall be able to invite other users to their group.
   6.2. A group member shall be able to leave their group.
   6.3. A group member shall be able to create forum threads in their group's forum.
   6.4. A group member shall be able to create forum posts in their group's forum.
   6.5. A group member shall be able to send messages in their group's chat.

7. Group Moderator
   7.1. A group moderator shall be able to kick group members from their group.
   7.2. A group moderator shall be able to delete forum threads in their group's forum.
   7.3. A group moderator shall be able to delete forum posts in their group's forum.

8. Group Administrator
   8.1. A group administrator shall be able to delete their group.
   8.2. A group administrator shall be able to appoint group moderators.
   8.3. A group administrator shall be able to unappoint group moderators.
   8.4. A group administrator shall be able to kick group moderators from the group by first unappointing the moderator, then kicking them.

8.5. A group administrator shall be able to edit the announcement on their group's home page.

Priority 2:

9. Approved User
   - 9.1. An approved user shall be able to delete their account.
   - 9.2. An approved user shall be able to change their password.
   - 9.3. An approved user shall be able to attach images to their chat messages.
   - 9.4. An approved user shall be able to sort forum threads.
   - 9.5. An approved user shall be able to report forum posts to a moderator.
   - 9.6. An approved user shall be able to block other users.
   - 9.7. An approved user who is selling an item shall be able to list their accepted payment methods.
   - 9.8. An approved user who is selling an item shall be able to list their possible delivery methods.
   - 9.9. An approved user who is selling an item shall be able to list their preferred contact methods.
   - 9.10. An approved user shall be able to edit the details of an item they are selling.

10. Moderator
    - 10.1. A moderator shall be able to delete messages in Gator Chat.

11. Group Moderator
    - 11.1. A group moderator shall be able to delete messages in their group's chat.

12. Group Administrator
    - 12.1. A group administrator shall be able to edit their group's description.

Priority 3:

13. All Users
    - 13.1. All users shall be able to donate money to the developers via PayPal. PayPal has a donate button that prompts the user to specify how much they want to donate, and that donation shall go to the developers.
    - 13.2. All users shall be able to donate money to the developers via Venmo. The developers have a Venmo username that the user can direct their Venmo donation to.

14. Approved User
    - 14.1. An approved user shall be able to compare different marketplace listings against each other.
    - 14.2. An approved user shall be able to add items to a wishlist.
    - 14.3. An approved user shall be able to bookmark forum threads.

14.4. An approved user shall be able to like forum posts.
14.5. An approved user shall be able to leave feedback about the seller they purchased an item from.

15. Moderator
    15.1. A moderator shall be able to pin forum threads.

16. Administrator
    16.1. An administrator shall be able to create new categories in the forum.
    16.2. An administrator shall be able to delete categories in the forum.

17. Group Moderator
    17.1. A group moderator shall be able to pin forum threads in their group's forum.

18. Group Administrator
    18.1. A group administrator shall be able to resign their position as group administrator and give the position to another member of the group.
    18.2. A group administrator shall be able to create new categories in their group's forum.
    18.3. A group administrator shall be able to delete categories in their group's forum.

3. Wireframes Based on Our Mockups/Storyboards V2

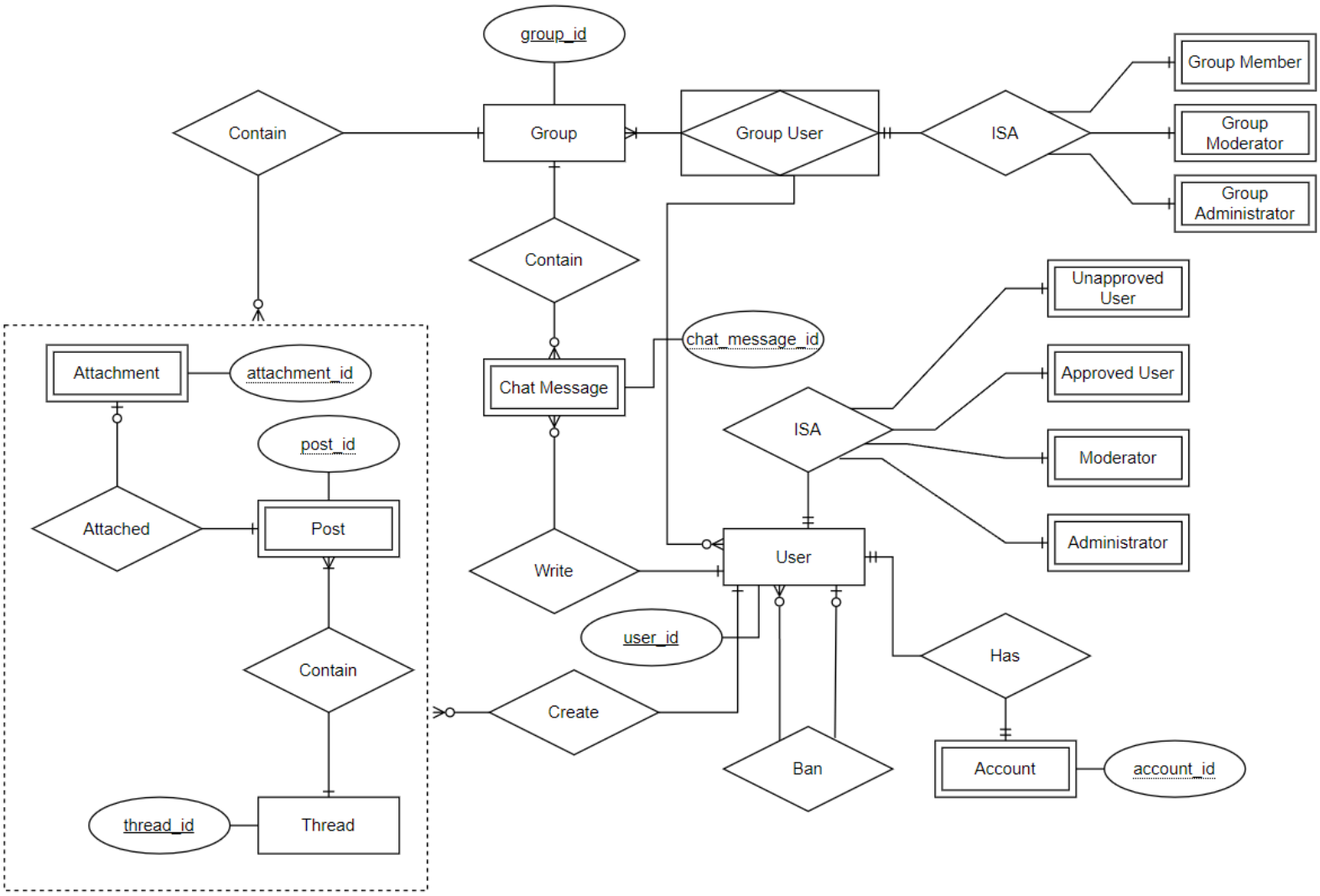
Link to Figma Wireframe:
https://www.figma.com/file/DIbMW3CeBzCwxkem2SSRU6/M3V2?node-id=0%3A1&t=zcUI59QqIIyu5MI1-1

After clicking on the link, press the play button on the top right of the screen to demo our wireframe.

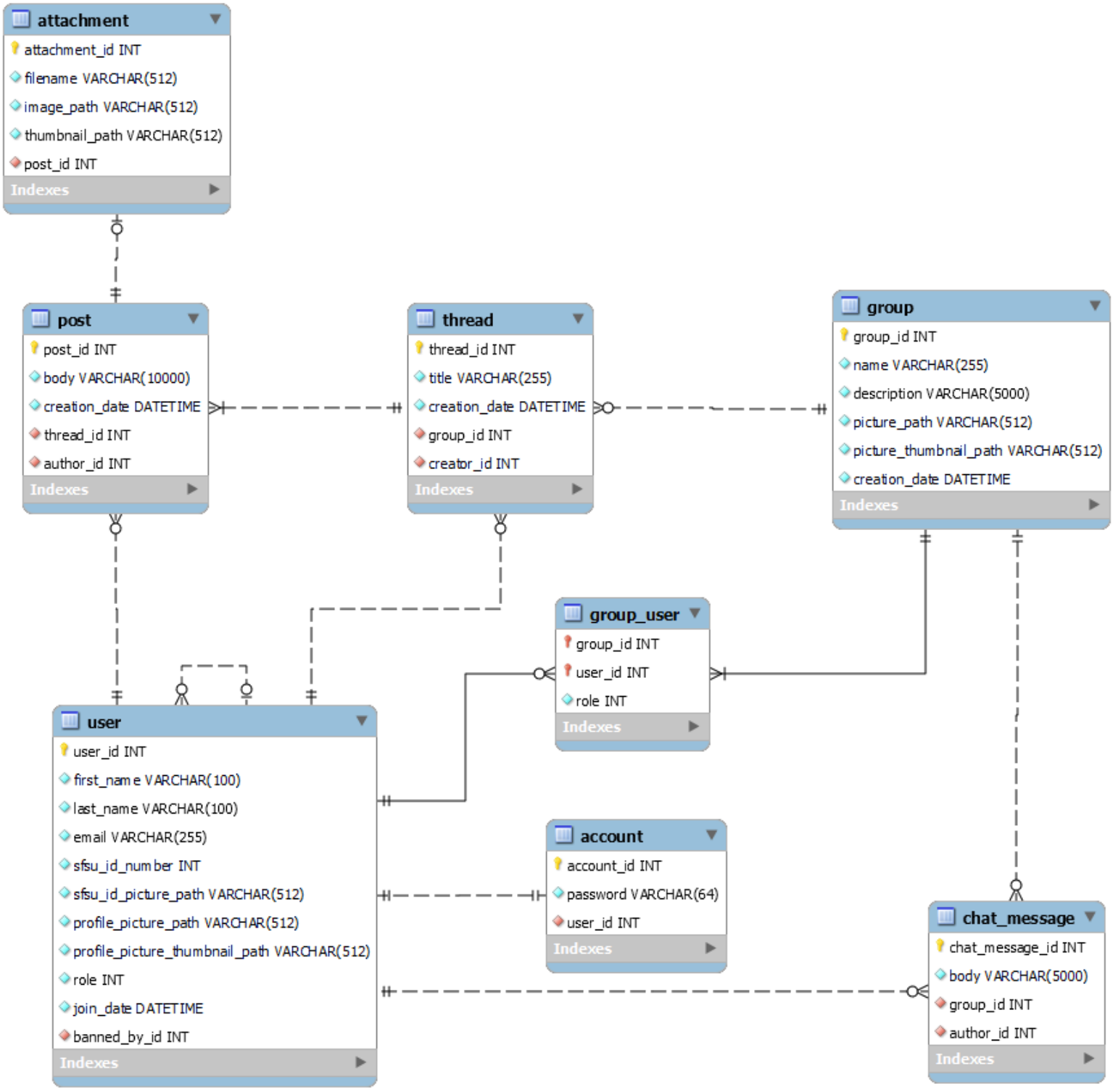
The wireframe page contains all of our pages and their variants (e.g. modals) in our application. Some components are stored in the wireframe page because prototyping does not appear to work when two frames are on different pages.
The static assets page is for components that do not use prototyping and therefore can be on their own page.

4. High Level Database Architecture and Organization V2

ERD from M2V2:

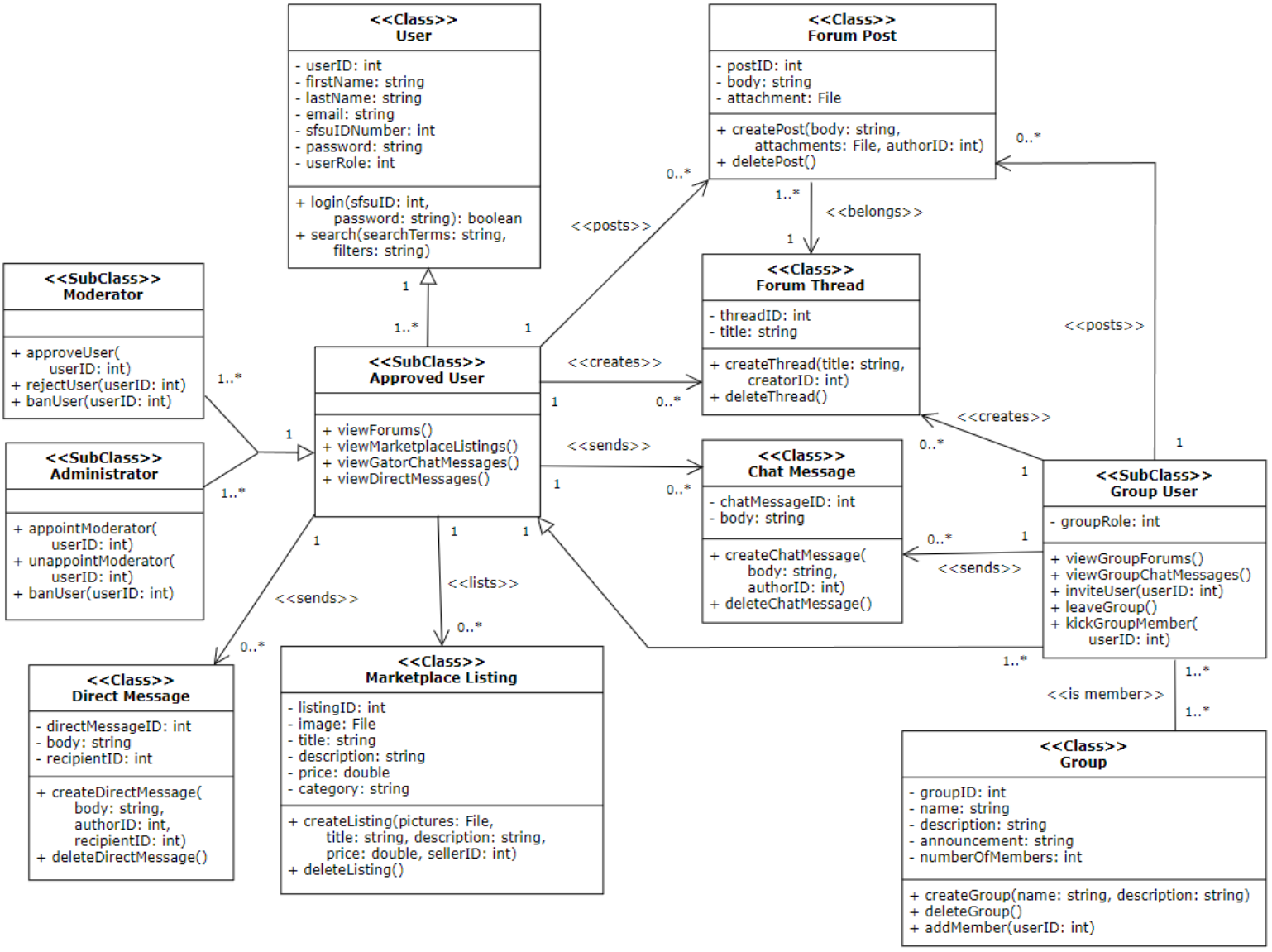


EER Model Based on the M2V2 ERD:

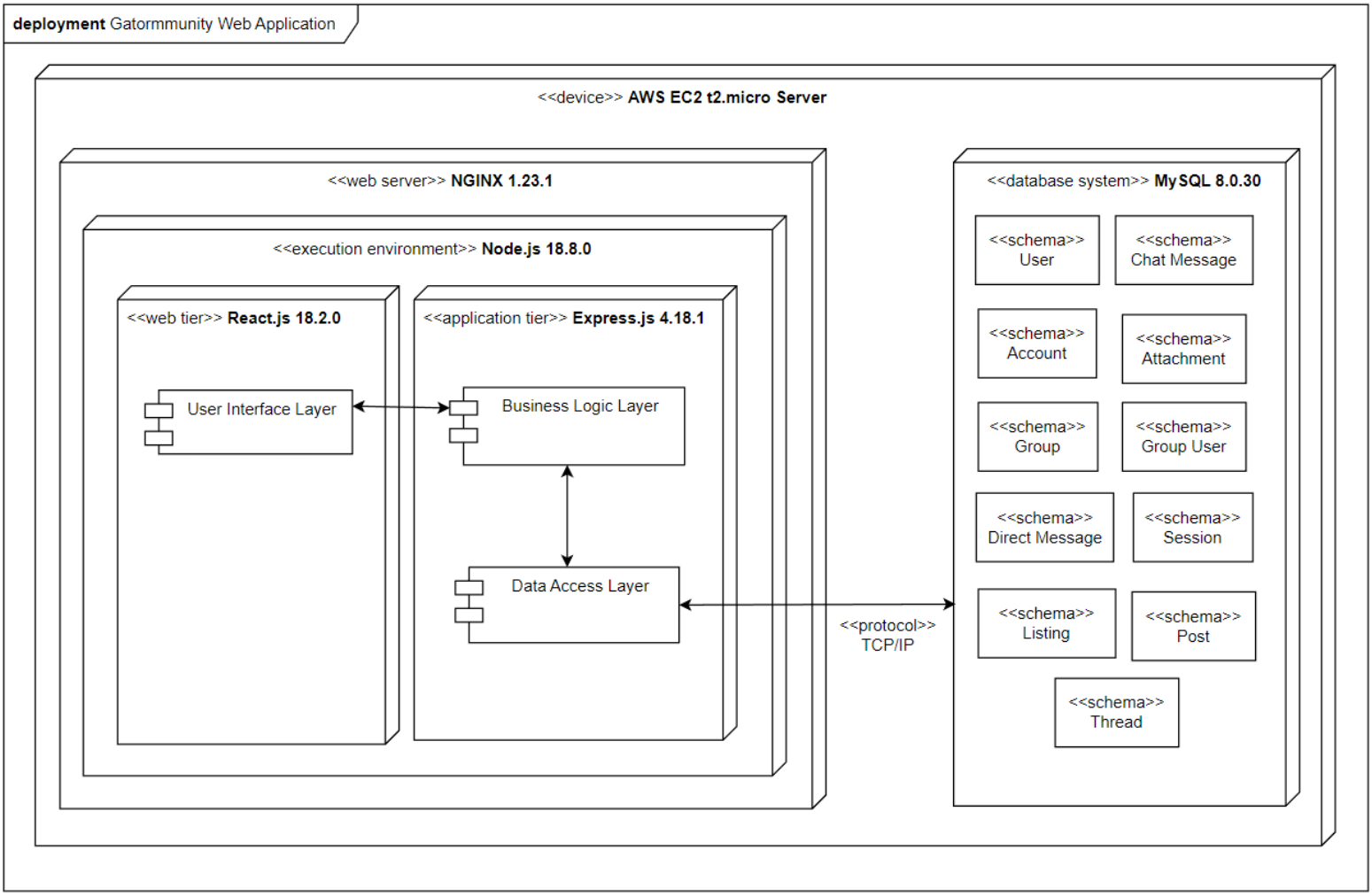


5. High Level Diagrams V2

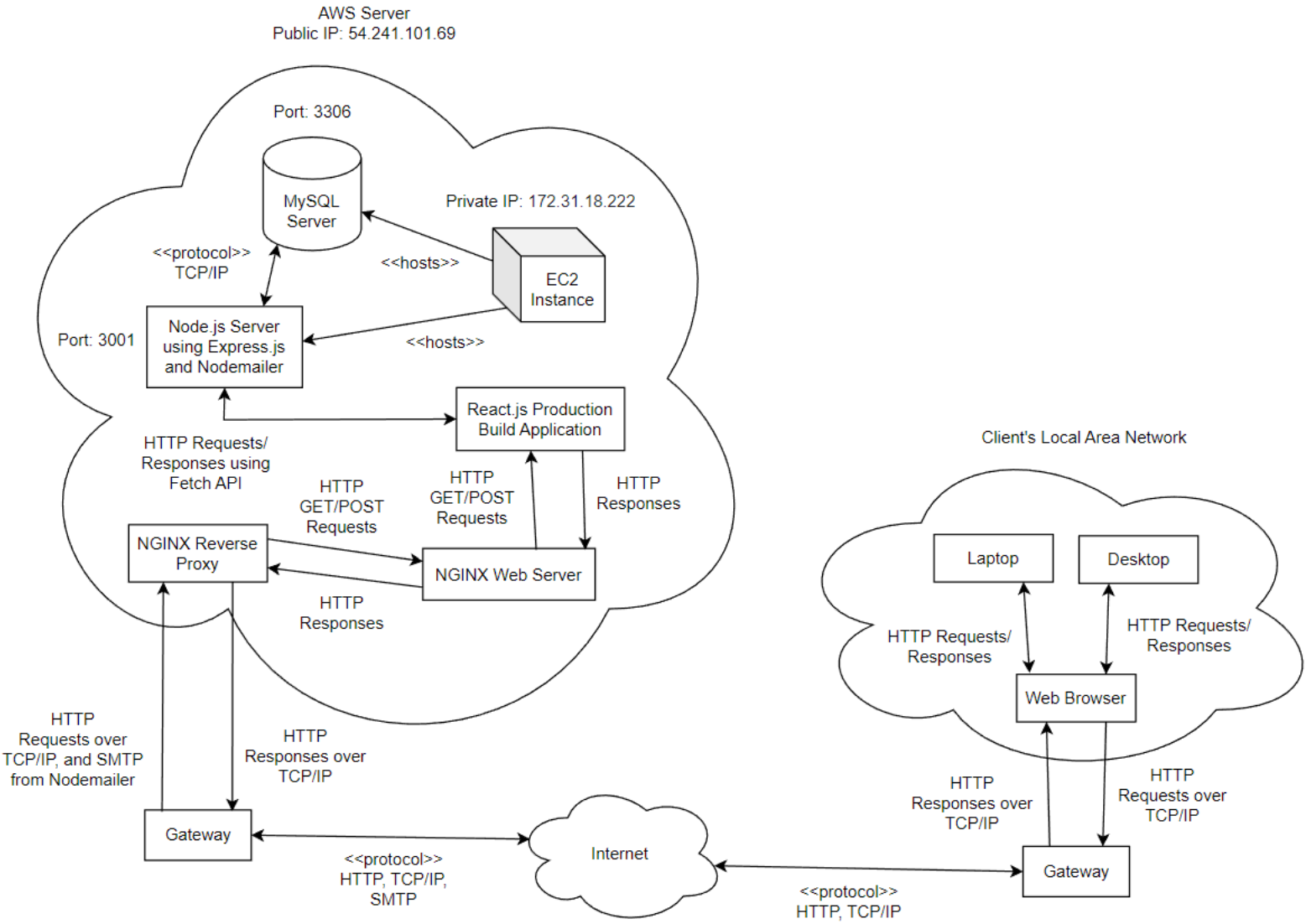
UML Diagram:



Deployment Diagram:



Network Diagram:



6. List of Contributions in This Milestone

Anthony Zhang:

- Active participant in the meetings and team Discord server
- Discussed how to apply the feedback from M2V1 and the Vertical Prototype
- Applied the feedback and edited M2V2, then brought the changes from M2V2 into M3V1 for the Data Definitions, ERD, UML diagram, deployment diagram, and network diagram
- Helped reprioritize our functional requirements
- Created the final Figma frames for the pages: Home, Registration, Login, Dashboard, User Profile, Chat, Inbox, Search, Marketplace, User's Groups, Group Home, Group Members, Create Group, Create Listing, View Listing, Group Forums, Gatormmunity Forums, Group Create Thread, Create Thread, View Thread, Contact Us, Help, Answer
- Created most of the Figma components and created the prototyping links between the Figma frames
- Created the EER Model
- Created the pages: Terms of Service, Privacy Policy, Help, Answer, Group Home, Group Forums, Group Create Thread, View Thread, Search, User Profile, Create Group, View Listing
- Helped create the pages: Contact Us, Create Listing, Create Thread
- Created the modals: Change Profile Picture, Group Invite, Group Change Announcement, Forgot Password
- Implemented the back end for the Change Profile Picture feature
- Helped create the mock/hardcoded data

Marwan Alnounou:

- Active participant in the meetings and team Discord server
- Helped reprioritize our functional requirements
- Set up the front end for SW development by copying over our vertical prototype's front end folder into the horizontal prototype folder
- Created the pages: Login, Registration
- Helped create the Search page
- Styled the pages: Terms of Service, Privacy Policy, Home, Contact Us
- Styled the Navbar and Footer

Mohamed Sharif:

- Active participant in the meetings and team Discord server
- Discussed how to apply the feedback from M2V1 and the Vertical Prototype
- Helped reprioritize our functional requirements
- Set up the back end for SW development by copying over our vertical prototype's back end folder into the horizontal prototype folder
- Refactored our back end code by splitting our routes file into a controller file and a route file. This was done to follow the MVC model more closely.

- Helped create the pages: Contact Us
- Tried to create the Gatormmunity Forums page
- Implemented the back end for the pages: Contact Us, Registration

Jose Lopez:

- Active participant in the meetings and team Discord server
- Discussed how to apply the feedback from M2V1 and the Vertical Prototype
- Helped reprioritize our functional requirements
- Created the initial Figma frames for the pages: Login, Registration, User Profile, Create Post, Gatormmunity Forums, Inbox. They were later simplified by Anthony to make the pages easier to implement.
- Helped create the pages: Create Listing, Create Thread
- Tried to create the Inbox page

Florian Cartozo:

- Helped reprioritize our functional requirements
- Set up the shared Figma file and taught the team how to use components and Figma's prototyping feature
- Created the Figma components: Navbar, Footer
- Created the final Figma frames for the pages: About Us, About Me, Privacy Policy, Terms of Service, Page Not Found
- Created the pages: About Us, About Me, Dashboard, Marketplace, Gatormmunity Forums, Inbox, Chat, Page Not Found, User's Groups, Group Members
- Helped create the mock/hardcoded data

Contribution Scores:

Anthony Zhang: 10
Marwan Alnounou: 5
Mohamed Sharif: 5
Jose Lopez: 5
Florian Cartozo: 10

SW Engineering CSC648/848 Fall 2022
Gatormmunity

Team 7

Anthony Zhang (Team Lead),
azhang12@mail.sfsu.edu

Marwan Alnounou

Mohamed Sharif

Jose Lopez

Florian Cartozo

Milestone 4
12/14/2022

History Table

Version	Submission Date
M4V1	12/01/2022
M4V2	12/14/2022

Table of Contents

1) Product Summary

Name of the Product: Gatormmunity

All Major Committed Functions:

1. All users shall be able to contact the developers via the Contact Us page.
2. All users shall be able to view the home page.
3. All users shall be able to view the about page to learn about Gatormmunity.
4. A guest user shall be able to view the registration page.
5. A guest user shall be able to view the login page.
6. A guest user shall be able to register for an account.
7. An approved user shall be able to log in to their account.
8. An approved user shall be able to reset their password by contacting the admins via the Contact Us page.
9. An approved user shall be able to report other users via the Contact Us page.
10. An approved user shall be able to sort forum threads.
11. An approved user shall be able to create a forum thread.
12. An approved user shall be able to assign their forum thread to a category.
13. An approved user shall be able to create a forum post.
14. An approved user shall be able to attach an image when creating a thread. The image will be displayed in the first post of the thread.
15. An approved user shall have a profile page.
16. An approved user's profile page shall show their details, e.g. their full name.
17. An approved user's profile page shall show their recent activities, e.g. their recently created listings or threads.
18. An approved user shall be able to view a user's profile page.
19. An approved user shall have a profile picture.
20. An approved user shall be able to change their profile picture.
21. An approved user shall be able to create a user group.
22. An approved user shall be able to join a user group after being invited to it, via the link provided in the invite message.
23. An approved user shall be able to send direct messages to other users.
24. An approved user shall be able to receive direct messages from other approved users.
25. An approved user shall be able to send messages in Gator Chat.
26. An approved user shall be able to see marketplace listings in the marketplace.
27. An approved user shall be able to select marketplace listings in the marketplace.
28. An approved user shall be able to report marketplace listings via the Contact Us page.
29. An approved user shall be able to buy items from the marketplace by contacting a listing's seller.
30. An approved user shall be able to sell items on the marketplace by creating a listing.
31. An approved user who is selling an item shall be able to assign their listing to a category.
32. An approved user shall be able to upload a picture of an item they are selling.

33. An approved user shall be able to delete their own listings from the marketplace.
34. An approved user shall be able to search for users.
35. An approved user shall be able to search for marketplace listings.
36. An approved user shall be able to search for forum threads.
37. An approved user shall be able to apply filters to their user search results.
38. An approved user shall be able to apply filters to their marketplace search results.
39. An approved user shall be able to apply filters to their forum thread search results.
40. A moderator shall be able to approve unapproved users via the Admin page.
41. A moderator shall be able to reject unapproved users via the Admin page.
42. A moderator shall be able to ban unapproved and approved users from Gatormmunity via the Admin page.
43. A moderator shall be able to delete forum threads.
44. A moderator shall be able to delete forum posts.
45. A moderator shall be able to delete listings in the marketplace.
46. An administrator shall be able to appoint moderators via the Admin page.
47. An administrator shall be able to unappoint moderators via the Admin page.
48. An administrator shall be able to ban moderators from Gatormmunity after first unappointing the moderator via the Admin page.
49. An administrator shall be able to change a user's password via the Admin page.
50. A group member shall be able to invite other users to their group.
51. A group member shall be able to leave their group.
52. A group member shall be able to create forum threads in their group's forum.
53. A group member shall be able to create forum posts in their group's forum.
54. A group member shall be able to send messages in their group's chat.
55. A group moderator shall be able to kick group members from their group via the Group Members page.
56. A group moderator shall be able to delete forum threads in their group's forum.
57. A group moderator shall be able to delete forum posts in their group's forum.
58. A group administrator shall be able to delete their group.
59. A group administrator shall be able to appoint group moderators via the Group Members page.
60. A group administrator shall be able to unappoint group moderators via the Group Members page.
61. A group administrator shall be able to kick group moderators from the group after first unappointing the moderator via the Group Members page.
62. A group administrator shall be able to edit the announcement on their group's home page.

What is Unique about our Project:

Gatormmunity distinguishes itself from its competitors like Craigslist and OfferUp by having mandatory identity verification by requiring a picture of one's SFSU ID card in order to register. This makes our platform safer because all registered users will know every other user is a member of SFSU and has been vetted by Gatormmunity's moderators. Also, we have a Forums page that allows members of Gatormmunity to talk to each other and plan events.
Furthermore, users can form groups with others and have their own private group chat and forum. This lets friends or people who share interests talk to each other in private. Gatormmunity also has a live chat (Gator Chat) for all of its members, where users can talk to one another in real time.

Product URL: http://54.241.101.69/

Since new accounts must be approved by a moderator or administrator in order to log in, we provided some credentials to some already approved accounts at the bottom of the Credentials README on GitHub. There is one account for each permission level: one for Approved Users, one for Moderators, and one for Administrators.

2) Usability Test Plan

Usability Test Plans for each of the 6 Test/Use Cases:

Test/Use Case #1: Create a Forum Thread

Test Objectives:
We are testing if the user can find the Gatormmunity Forums page, click on the “Create Thread” button, fill out the form on the Create Thread page, and optionally, if they are able to use the resources in the Helpful Links section of the Create Thread page. We are also testing if the user is able to recognize that their forum thread was successfully created, since we received feedback that sometimes it was not clear if an action was successful.

We want the user to be able to find the Gatormmunity Forums page easily because it is a key part of our website, and we need to see if the user can quickly locate the Create Thread button since that is how users will post events and questions to the Gatormmunity Forums. We note whether the user clicks on a link in the Helpful Links section because we feel that the helpful links are hard to notice. One of our users in Milestone 3 thought uploading an image for a thread was mandatory when it was optional, so we will observe whether or not a user tries to upload an image.

Test Description

System Setup:
The user will be performing this test on their computer or laptop using a supported operating system (Windows 10, Windows 11, or macOS Monterey 12.5/12.6) with a supported browser (Google Chrome 105, Microsoft Edge 105, or Safari 15.5). The user must have reliable internet access.

Starting Point:
The user should begin the test from the dashboard page after logging in because this is the first test, and presumably the user will not yet have logged into the test account provided for them. Ultimately, it does not make a difference where the user starts because the navigation bar which houses the Forums button is accessible on every page. However, the user must be logged in to see the Forums button in the navigation bar.

Intended Users:
This test is intended for every registered user because the Forums page is one of our core functionalities, and should be easy to access for users of almost all skill levels. The user should at least have some prior experience with using websites because our website requires that the user know that the navigation bar has clickable links. This may be a concern for some students and faculty because not every member of SFSU is familiar with websites.

URLs to be Tested:
Gatormmunity Forums Page (accessible from the navbar): http://54.241.101.69/forums
Create Thread Page (accessible from the Forums page): http://54.241.101.69/create-thread
View Thread Page (the user is redirected here after creating a thread): http://54.241.101.69/thread/N, where N is an integer representing the id of the thread.

What is Being Measured:
We are measuring if the user is able to rapidly find the Forums button in the navigation bar, if the user is able to find the Create Thread button in the Gatormmunity Forums page, if the user is able to fill out the form without getting confused, and if the user is able to recognize that their thread was successfully created after pressing the Submit button on the Create Thread page.

Test/Use Case #2: Post in a Forum Thread

Test Objectives:
We are testing if the user is able to recognize what a thread looks like on the Gatormmunity Forums page, if the user will know that a thread is clickable on the Gatormmunity Forums page, if the user will know where to find the Create Post form, if the user knows how to submit the form, and if the user will recognize that their post was successfully submitted. We are also testing if the user can find the "Back to Forums" button in the top left in case they want to find some other thread to post in.

We are testing the user's ability to recognize threads on the Forums page because being able to view other people's threads and the posts within them is one of the ways in which a user can interact with Gatormmunity's community. We added a darken on hover effect to the threads on the Forums page so that the user can recognize that the threads are clickable, which we are testing for. Since posting in a thread is how you can reply to other users in a thread, we want to ensure that the process is easy and intuitive for our users. Again, we want to test if the user knows their post was successfully posted because we received feedback saying it was not so obvious in Milestone 3.

Test Description

System Setup:
The user will be performing this test on their computer or laptop using a supported operating system (Windows 10, Windows 11, or macOS Monterey 12.5/12.6) with a supported browser (Google Chrome 105, Microsoft Edge 105, or Safari 15.5). The user must have reliable internet access.

Starting Point:
The user will begin the test from the View Thread page (i.e. they are already viewing a thread) because this test will take place immediately after the user created a thread from Test #1. Since it would be too simple for the user to post in their newly created thread, we ask that the user find some other interesting thread in the Forums page to post in. We want the user to experience the process of

posting to a thread from the very beginning, and we need to test if the user is capable of doing this without support.

Intended Users:
This test is intended for every registered user because being able to post in a thread is one way users can communicate with one another on Gatormmunity. The forums feature is one of our key features, so posting to a thread should be easy for users of almost all skill levels. The user should at least have some prior experience with using websites because posting to a thread requires that the user know that something darkening on hover and the cursor changing to a hand means that the item being hovered over is clickable.

URLs to be Tested:
View Thread Page (the user starts on this page due to the test starting immediately after Test #1): http://54.241.101.69/thread/N, where N is an integer representing the id of the thread.
Gatormmunity Forums Page (the user gets here via the “Back to Forums” button or from the Forums button in the navbar): http://54.241.101.69/forums
View Thread Page (this is a different thread than the one the user started on. The user gets here by clicking on a thread from the Gatormmunity Forums page): http://54.241.101.69/thread/M, where M is an integer representing the id of a different thread from the one the user started the test on.

What is Being Measured:
We are measuring if the user can find out how to go back to the Forums page via either the Back to Forums button or navigation bar, if the user can find a thread they want to post to in the Forums page, if they can find where to type text into to post to the thread, and if they are able to recognize that their post was successful.

Test/Use Case #3: Sell an Item

Test Objectives:
We are testing if the user knows that the “Market” button in the navigation bar leads to the marketplace page where users can sell things, if the user knows that creating a listing means that they are selling an item, if the user knows how to fill out the Create Listing form, and if the user is able to see their listing after submitting the create listing form.

We test the user’s ability to find the “Market” button in the navigation bar because the marketplace is another one of our key functionalities, so it should be easily findable by Gatormmunity’s users. We are checking if the term “Create Listing” is clear to users that it is how they can sell items and services on Gatormmunity. We ask that the user fill out the Create Listing form because we want to test if it is clear what the user should enter in each of the text fields. We also want to see how the user reacts to learning that a listing photo is required to create their listing. We test if the user can see their listing since we want to ensure that the user knows that their listing creation was successful.

Test Description

System Setup:
The user will be performing this test on their computer or laptop using a supported operating system (Windows 10, Windows 11, or macOS Monterey 12.5/12.6) with a supported browser (Google Chrome 105, Microsoft Edge 105, or Safari 15.5). The user must have reliable internet access.

Starting Point:
Since the test occurs after Test #2, the user will likely start from the View Thread page, though it does not matter where they start because the navigation bar is available on every page. Naturally, the user must be logged in to access the marketplace page.

Intended Users:
This test is intended for every registered user because we want everyone to know how to create a listing in case they would ever want to do so in the future. The ideal user for this feature should be aware of the risks of meeting up with another person to trade because even though the risk of an incident is less with Gatormmunity compared to its competitors, the risk is still there.

URLs to be Tested:
Marketplace Page (accessible from the navbar): http://54.241.101.69/marketplace
Create Listing Page (accessible from the marketplace page): http://54.241.101.69/create-listing
View Listing Page (the user is redirected here after creating a listing): http://54.241.101.69/listing/N, where N is an integer representing the id of the listing.

What is Being Measured:
We are measuring if the user can find the Marketplace page without incident, if the user recognizes that the Create Listing button is what they need to press to sell an item, if the user correctly fills out the required fields in the form, and if the user successfully creates the listing. We observe the reaction of the user after being redirected to the View Listing page showing their newly created listing, in case there is any confusion about what they are seeing.

Test/Use Case #4: Create a User Group

Test Objectives:
We are testing if the user is able to successfully find the Create Group button on the Users Groups page accessible from the navigation bar. We are testing if the user understands what “User Group” means because it might not be the best terminology for referring to groups that users can form with their own private forums and chat rooms. We test if the user can successfully fill out the Create Group form, and if the user is satisfied with the appearance of the Group Home page.

We test the user on their ability to find the Groups page because this is how users can view the list of groups they are in, so being able to find the Groups page without trouble is important. We test that

users can successfully create a group because at first, new users will not be in any groups, and will want to either create their own group or join one. Since any approved user can create a group, we believe many users will want to create one to see what being in a group is like, therefore we want to ensure that creating a group is a pain free process for the user. Finally, we ask that the user give their thoughts on the appearance of the group home page because we feel that the group home page could benefit from having more content on it, but we also do not wish to overwhelm the user.

Test Description

System Setup:
The user will be performing this test on their computer or laptop using a supported operating system (Windows 10, Windows 11, or macOS Monterey 12.5/12.6) with a supported browser (Google Chrome 105, Microsoft Edge 105, or Safari 15.5). The user must have reliable internet access.

Starting Point:
Since the test occurs after Test #3, the user will likely start from the View Listing page, though it does not matter where they start because the navigation bar is available on every page. The user must be logged in to access the Users Groups page.

Intended Users:
This test is intended for every registered user since we believe that being able to join groups just to see what groups are like is something many users will try at least once. Anybody who wants to create a group must go through the process the tester goes through, therefore it is important that this feature works as expected.

URLs to be Tested:
Users Groups Page (accessible from the navbar): http://54.241.101.69/groups
Create Group Page (accessible from the users groups page): http://54.241.101.69/create-group
Group Home Page (the user is redirected here after creating a group): http://54.241.101.69/group/N, where N is an integer representing the id of the group.

What is Being Measured:
We are measuring if the user can find the Users Groups page without problems, if the user recognizes that the Create Group button is what they need to press to “Create a User Group”, if the user correctly fills out the required fields in the form, if the user successfully creates the group, and the user’s reaction to the new group’s home page shown upon creating a group.

Test/Use Case #5: Send Messages in Group Chat

Test Objectives:
We are testing if the user is able to find the Group Chats page, if the user is able to find the group chat room they are looking for on the Group Chats page, if the user can see the chat messages from other group members, if the user is able to send a message in the group chat, if the user can see their sent chat message, and if the chat messages successfully update in real time.

We test if the user can find the Group Chats page since group chats are a significant feature of Gatormmunity, and we want the chat room to be easy to find and get to. We test if the user can see chat messages from other group members because we want to make sure that fetching the messages from the database works without problems. We test if the user can send messages because we want to see if the user has any difficulties or complaints about the message sending process. Lastly, we test that the user can see their message show up in the chat log because if a user cannot see their messages in the chat log, then they will not know if their message sending was successful.

Test Description

System Setup:
The user will be performing this test on their computer or laptop using a supported operating system (Windows 10, Windows 11, or macOS Monterey 12.5/12.6) with a supported browser (Google Chrome 105, Microsoft Edge 105, or Safari 15.5). The user must have reliable internet access.

Starting Point:
Since the test occurs after Test #4, the user will likely start from the Group Home page, though it does not matter where they start because the navigation bar is accessible on every page. The user must be logged in to access the Group Chats page.

Intended Users:
This test is intended for every registered user. If the group chat function works, then Gator Chat also works, and all approved users have access to Gator Chat whether they are in a group or not. As a result, we must ensure that the Group Chats page is easy and intuitive to find for all of our registered users.

URL to be Tested:
Group Chat Page (accessible from the navbar): http://54.241.101.69/chat
Different group chats are accessible from the same URL by clicking on the group chat rows on the left column of the Group Chat page.

What is Being Measured:
We are measuring how quickly the user can locate the Group Chats page, if they are able to find a particular group to send chat messages in, if they are able to see the chat messages for a particular group on the screen, how quickly the chat messages load, if the user is able to successfully send a message, and how long it takes for the sent message to appear in the chat message log.

Test/Use Case #6: Search

There are 6 tests rather than 5 since it was said in class that 5 is the minimum number of tests we must have rather than the exact amount we must have.

Test Objectives:
We are testing that the search bar in the navigation bar successfully takes user input if any was provided, if the search bar takes the user to the search results page, if the search page shows the expected results, if the search filters work, if the search results are able to be clicked on, and that the clicked on search result takes the user to the correct page.

We test the search bar because virtually every user will use it, hence it must be good enough such that users are satisfied with it. We want to ensure that the search message and search results do not confuse the user because we do not want the user to avoid using the search bar, we want them to use it as much as possible if it will help them find what they want faster. We test the search filters since they allow a user to refine their search results and change what they are searching for. We test the search results are laid out correctly because styling and formatting has been a recurring problem, and we test that the search results take the user to the correct page because we want the user to be able to see more about the item they are searching for, e.g. the profile page of the user, the listing details and contact details for a listing, or a thread's posts.

Test Description

System Setup:
The user will be performing this test on their computer or laptop using a supported operating system (Windows 10, Windows 11, or macOS Monterey 12.5/12.6) with a supported browser (Google Chrome 105, Microsoft Edge 105, or Safari 15.5). The user must have reliable internet access.

Starting Point:
Since the test occurs after Test #5, the user will likely start from the Group Chat page, though it does not matter where they start because the search bar and the navigation bar are accessible on every page. The user does not need to be logged in to access the search page, but to be able to search for listings and threads requires a logged in user.

Intended Users:
This test is intended for logged in users, but non-logged in users can use the search bar too, just with less functionality such as not being able to search listings and threads, and not being able to see profile pages. Every logged in user will be using the search bar, so every one of our testers must interact with the search bar so that we can determine its problems.

URL to be Tested:
Search Page (accessible from the search bar in the navbar): http://54.241.101.69/search OR http://54.241.101.69/search/X, where X is the user's search terms, if any were entered into the search bar. We allow no input into the search bar, which acts as a search for all items regardless of their name or title, although other filters such as category and price can still be applied.

What is Being Measured:
We are measuring how quickly the user is able to find what they are looking for, if they were confused on any part of the searching process, if the search was able to display recommendations on a search that matches nothing, how quickly the search results load, and how well the search results look on the tester's computer, since screen sizes have an effect on how many columns there are or how wide a search result is, for instance.

Usability Test Table for Measuring Effectiveness and Efficiency:

Test/Use Case	% Completed	Errors	Comments	% in Time Completed
Create a Forum Thread	100%	None	One tester was not aware they could click on the text in the navbar. Most testers did not click on the helpful links. Most testers typed only a few words or a sentence in their thread body, despite the massive text field.	100%
Post in a Forum Thread	100%	None	One tester did not see the "Back to Forums" button at the top of the View Thread page; we made the button text larger as a result. One tester suggested that we use a rich text editor instead of a simple plaintext input form. No testers made use of the category filter on the Forums page.	100%

			Asking the tester to post in a different forum thread than the one they created in Test #1 made the test more confusing for them.	
Sell An Item	100%	One tester tried to upload a PDF as the listing's image, which caused an alert message to show telling the user to upload an image with one of the accepted image formats. One tester did not upload an image, which caused the browser to tell the tester: "Please select a file."	One tester did not have any images on their computer to upload for the listing's image, so they had to find an image online, save it, then upload it. We should consider making images optional to increase the task's efficiency. One tester said we should add more categories for our listings. One tester tried to use AirDrop to upload a file. It did not work, since we do not support AirDrop.	75%
Create a User Group	100%	One tester did not upload an image, which caused the browser to tell the tester: "Please select a file."	One tester said that it does not look too good to have the group's name and "Announcement" be centered, while the group's description and announcement text are left-aligned. One tester said that we should round the profile picture thumbnail in the navbar because images look better rounded, such as in the group home page.	100%
Send Messages in Group Chat	100%	None	One tester asked how come they could not see any messages in their group chat. The chat log was empty because their group was newly created and thus did not have any messages sent in it yet. We should have sent some messages beforehand in the groups the test user was in. One tester said the message input box covered up some of the chat messages in the message log. This issue could not be reproduced on the observer's computer despite resizing the window, zooming in/out, etc.	100%

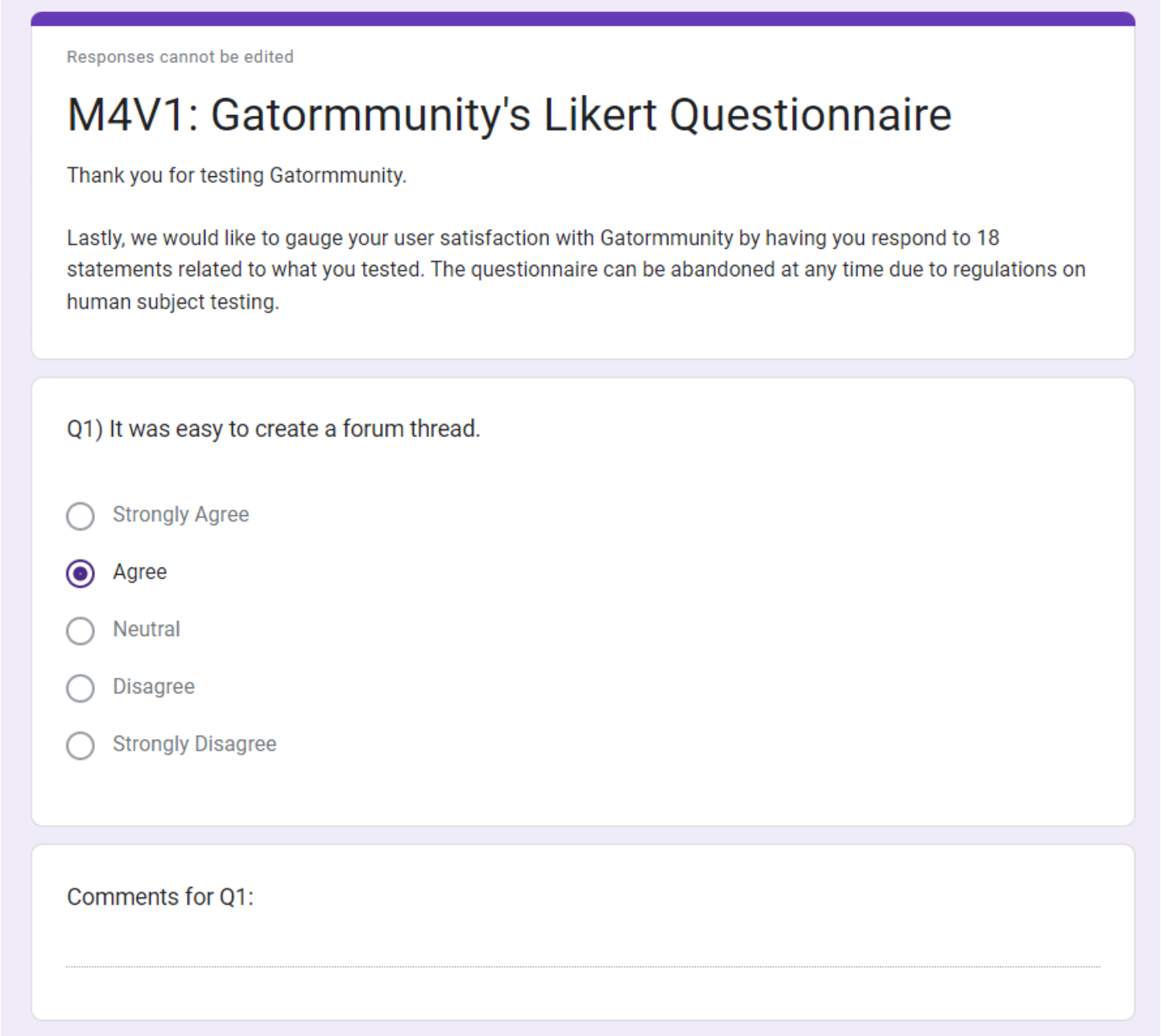
Search	66%	None	The search page works but the testers were not sure how to change the search category from users to listings and asked the observer how to change the search category. Multiple testers did not like how Users was the default search category because they were expecting to see listings but saw users instead, which confused them. We should make it more obvious that the user can change their search category in the left column where the search filters are. Alternatively, we could consider having the search page search for users, listings, and threads at the same time and have all of the matching items show in the same search results page. Then, the user can narrow down their search category with the search filters. Or, we can add a message to the top of the search results page telling the user they can switch their search category with the search filters on the left. We will need to revise the search page so that nobody struggles with finding what they are looking for (listings, more often than not).	0%

User Satisfaction using a Likert Questionnaire:

We used Google Forms to create a Likert Questionnaire, which is accessible here: https://forms.gle/GvjsXjuFmMoBQXjbA

There are 18 different statements, 3 for each function tested.

Sample Response:



Responses cannot be edited

M4V1: Gatormmunity's Likert Questionnaire

Thank you for testing Gatormmunity.

Lastly, we would like to gauge your user satisfaction with Gatormmunity by having you respond to 18 statements related to what you tested. The questionnaire can be abandoned at any time due to regulations on human subject testing.

Q1) It was easy to create a forum thread.

- ○ Strongly Agree
- ◉ Agree
- ○ Neutral
- ○ Disagree
- ○ Strongly Disagree

Comments for Q1:

Q2) I found the Helpful Links shown when creating a thread helpful.

- ○ Strongly Agree
- ◉ Agree
- ○ Neutral
- ○ Disagree
- ○ Strongly Disagree

Comments for Q2:

Q3) It was easy to find the thread I created.

- ◉ Strongly Agree
- ○ Agree
- ○ Neutral
- ○ Disagree
- ○ Strongly Disagree

Comments for Q3:

Q4) It was easy to create a post in a forum thread.

- ○ Strongly Agree
- ◉ Agree
- ○ Neutral
- ○ Disagree
- ○ Strongly Disagree

Comments for Q4:

Q5) It was clear that my forum post was successfully posted.

- (●) Strongly Agree
- ( ) Agree
- ( ) Neutral
- ( ) Disagree
- ( ) Strongly Disagree

Comments for Q5:

Q6) I would use the forums in the future to interact with others on Gatormmunity.

- ○ Strongly Agree
- ○ Agree
- ◉ Neutral
- ○ Disagree
- ○ Strongly Disagree

Comments for Q6:

Q7) It was easy and intuitive to sell an item.

- ○ Strongly Agree
- ◉ Agree
- ○ Neutral
- ○ Disagree
- ○ Strongly Disagree

Comments for Q7:

Q8) I would use Gatormmunity in the future for selling my belongings.

- ○ Strongly Agree
- ○ Agree
- ○ Neutral
- ◉ Disagree
- ○ Strongly Disagree

Comments for Q8:

Q9) I found the Helpful Links shown when creating a listing helpful.

- ○ Strongly Agree
- ◉ Agree
- ○ Neutral
- ○ Disagree
- ○ Strongly Disagree

Comments for Q9:

Q10) I found it easy to create a user group.

- ○ Strongly Agree
- ◉ Agree
- ○ Neutral
- ○ Disagree
- ○ Strongly Disagree

Comments for Q10:

Q11) The design and user interface of my group's home page was to my liking.

- ○ Strongly Agree
- ◉ Agree
- ○ Neutral
- ○ Disagree
- ○ Strongly Disagree

Comments for Q11:

Q12) When Gatormmunity is released, I would want to create my own group.

- (●) Strongly Agree
- ( ) Agree
- ( ) Neutral
- ( ) Disagree
- ( ) Strongly Disagree

Comments for Q12:

Q13) It was easy and intuitive to find the Group Chat page.

- (●) Strongly Agree
- ( ) Agree
- ( ) Neutral
- ( ) Disagree
- ( ) Strongly Disagree

Comments for Q13:

Q14) It was easy to send messages to my group.

- (●) Strongly Agree
- ( ) Agree
- ( ) Neutral
- ( ) Disagree
- ( ) Strongly Disagree

Comments for Q14:

Q15) I would use Gatormmunity's Group Chat feature to talk with my SFSU friends rather than through another social media service like Instagram.

- ○ Strongly Agree
- ○ Agree
- ○ Neutral
- ○ Disagree
- ◉ Strongly Disagree

Comments for Q15:

Q16) I found it easy and intuitive to use the Search feature.

- ○ Strongly Agree
- ◉ Agree
- ○ Neutral
- ○ Disagree
- ○ Strongly Disagree

Comments for Q16:

Q17) I would use the search function in the future for finding things on Gatormmunity.

- ○ Strongly Agree
- ○ Agree
- ○ Neutral
- ◉ Disagree
- ○ Strongly Disagree

Comments for Q17:

Q18) The design and user interface of the search page was to my liking.

- ○ Strongly Agree
- ◉ Agree
- ○ Neutral
- ○ Disagree
- ○ Strongly Disagree

Comments for Q18:

3) QA Test Plan

The five non-functional requirements to test:

3.3. Passwords shall be hashed before being stored in the database. In other words, there shall be no plaintext passwords stored in the database.
6.3. Approved users shall only upload files less than or equal to 5 MB in size.
7.2. The application shall use the personal data of its users for verifying their identity.
7.3. The application shall store and use its users' credentials for logging them in.
8.2. The application shall support the Google Chrome 105, Microsoft Edge 105, and Safari 15.5 web browsers.

Test #1: Password Hashing

3.3. Passwords shall be hashed before being stored in the database. In other words, there shall be no plaintext passwords stored in the database.

Test Objectives:
We are testing if the passwords are being hashed before being inserted into the database. The only instance where we insert a password into the database is when someone registers for an account, so we will be registering for accounts with passwords of various lengths and key combinations. In all successful registration cases, we are expecting our account table to have a new entry in it with the hashed password.

HW and SW Setup:
We will test on a computer or laptop using a supported operating system and a supported web browser. We will have MySQL Workbench open so that we can check what we just inserted into the database. The URL of the page to do the testing is that of the registration page:
http://54.241.101.69/register

Feature to be Tested:
We are testing the database hashing code on the back end. Since we are registering for an account, this test also tests our registration feature's front end and back end.

QA Test Plan Table:

Test #	Test Title	Test Description	Test Input	Expected Correct Output	Test Results
1	Alphabetical Password	Tests password hashing with a password with only alphabetical characters.	A new user with the password: testing	The account table should have a new row containing the new user's hashed password.	Pass
2	Alphanumeric Password	Tests password hashing with a password with alphabetical and numeric characters.	A new user with the password: 123testing123	The account table should have a new row containing the new user's hashed password.	Pass
3	Complex Password	Tests password hashing with a password with alphabetical, numeric, and special characters.	A new user with the password: ?123!testing&123#	The account table should have a new row containing the new user's hashed password.	Pass

Test #2: File Input Validation

6.3. Approved users shall only upload files less than or equal to 5 MB in size.

Test Objectives:
We will be testing if our file input validation works. Our file input validation is supposed to only allow users to upload files that are at most 5 MB in size and the file must be an image of type JPEG, PNG, WebP, GIF, or AVIF. We reuse our file input validation function since our application mandates the same file size and type requirements for all forms that allow file input, therefore we can choose any form that allows for file input to test.

HW and SW Setup:
We will test on a computer or laptop using a supported operating system and a supported web browser. We will have MySQL Workbench open so that we can get the filename of the file we just inserted into the database. We will have the marketplace page open so that we can see if the file and listing was successfully uploaded.
The URL of the page to do the testing is that of the create listing page:
http://54.241.101.69/create-listing

Feature to be Tested:
We are testing the file input validation on the Create Listing page. We will be uploading files of various sizes and types, and verifying that our file input validation meets the non-functional requirement.

QA Test Plan Table:

Test #	Test Title	Test Description	Test Input	Expected Correct Output	Test Results
1	Upload Valid Image File	Create a listing with an image that is of an accepted file type and file size.	A new listing with an image that shows a picture of tea. The file type is JPEG, and the file size is 148 KB.	The listing should be successfully created, and we should be able to see the file in the public/listing_photos folder.	Pass
2	Upload .txt File	Create a listing with a file that is not of an accepted file type (.txt) but with a valid file size.	A new listing with a text file with this text in it: “Hello”. The file type is TXT, and the file size is 5 bytes.	The listing should NOT be created, and the user should see an alert message saying the file is of an invalid type.	Pass
3	Upload Invalid Image File	Create a listing with an image that is of an accepted file type but a file size that is too large.	A new listing with an image that shows a picture of a fridge. The file type is PNG, and the file size is 9.30 MB.	The listing should NOT be created, and the user should see an alert message saying the file cannot exceed 5 MB.	Pass

Test #3: Checking an Unapproved User’s personal data to verify their identity

7.2. The application shall use the personal data of its users for verifying their identity.

Test Objectives:
We will be testing if the user data that is provided at registration is being stored in the database and that this data will be visible to moderators and administrators for the purpose of verifying the new user’s identity. In particular, the mods/admins should be able to see an unapproved user’s user ID, full name, email, SFSU ID number, SFSU ID picture, and registration date.

HW and SW Setup:
We will test on a computer or laptop using a supported operating system and a supported web browser. We will have the administration page open so that we can see the approval requests from unapproved users. In the approval requests tab is where mods/admins can verify users' identities and make unapproved users into approved users.
The URL of the page to do the testing is that of the administration page, which is only accessible to users with a moderator or administrator account: http://54.241.101.69/admin
To register for the accounts, the URL is that of the registration page: http://54.241.101.69/register

Feature to be Tested:
We are testing the approval requests section of the Administration page. We will be registering for accounts with different SFSU ID pictures, names, SFSU ID numbers, and so on, and we will be verifying that we are able to see this information in the approval requests section of the Administration page.

QA Test Plan Table:

Test #	Test Title	Test Description	Test Input	Expected Correct Output	Test Results
1	Unapproved User with Legitimate Credentials	Register for an account with credentials that look like a real SFSU student's account. The image should actually be an SFSU ID photo.	A new account with a real name, SFSU email, SFSU ID number, password, and genuine SFSU ID picture.	The new user's personal data (except the password) should be visible in the Approval Requests section of the Administration page.	Pass
2	Unapproved User with Fake Credentials	Register for an account with credentials that are clearly fake. The image should NOT be an SFSU ID photo.	A new account with a name composed of random characters, an illegitimate SFSU email, a random string of numbers for the SFSU ID number, password, and a picture that does not show an SFSU ID.	The new user's personal data (except the password) should be visible in the Approval Requests section of the Administration page.	Pass
3	Unapproved User with Legitimate Credentials but Wrong	Register for an account with credentials that look like a real SFSU student's account. The image should NOT be an SFSU ID photo.	A new account with a real name, SFSU email, SFSU ID number, password, but not an SFSU ID	The new user's personal data (except the password) should be visible in the	Pass

	Image		picture.	Approval Requests section of the Administration page.	

Test #4: Log In

7.3. The application shall store and use its users' credentials for logging them in.

Test Objectives:
We will be testing if a user's SFSU ID number and hashed password are correctly inserted into the database, and that a user is able to log in (after being approved). By extension, we are testing that the registration and approval requests work correctly. We will check the user and account table to verify that the user was successfully created.

HW and SW Setup:
We will test on a computer or laptop using a supported operating system and a supported web browser. We will have the administration page open so that we can see the approval requests from the newly created unapproved users. We will have the registration page open on another tab in order to create new users. In this same tab, we will log in after being approved.
We will have MySQL Workbench open to check the user and account table to verify successful insertion into the database.
The URL of the page we are testing is that of the login page, which is accessible only if you are not logged in: http://54.241.101.69/login
The URL of the page to approve unapproved users is that of the administration page, which is only accessible to users with a moderator or administrator account: http://54.241.101.69/admin
To register for the accounts, the URL is that of the registration page: http://54.241.101.69/register

Feature to be Tested:
We are testing the login feature. Since a new user has to be approved after registering before they can log in, we are testing the registration and approval features as well.

QA Test Plan Table:

Test #	Test Title	Test Description	Test Input	Expected Correct Output	Test Results
1	Register and Login After Being Approved	Register for an account and get approved by a moderator or administrator. We will check the database tables to see if the correct credentials were inserted. Then, we will	Register for a new account with the SFSU ID number: 913911011 and the password: CSC680. The new	The new user's SFSU ID number should be present in the `user` table, a hashed password should be present in	Pass

		log in after being approved.	user should be approved.	the `account` table with the new user's user_id, and the user should be able to login with the provided test input credentials.	
2	Attempt to Login with an Unapproved Account	Try to login with an account that has not yet been approved. The credentials should be correct.	Register for a new account with the SFSU ID number: 812310552 and the password: CSC317. The new user will not be approved.	The new user's SFSU ID number should be present in the `user` table, a hashed password should be present in the `account` table with the new user's user_id, and the user should NOT be able to login with the provided test input credentials.	Pass
3	Attempt to Login with Incorrect Credentials	Try to login with invalid credentials.	Login with the SFSU ID Number: 413413413 and the password: hello	The user should see an alert message telling them their credentials are incorrect.	Pass

Test #5: Web Browser Support

8.2. The application shall support the Google Chrome 105, Microsoft Edge 105, and Safari 15.5 web browsers.

Test Objectives:
We will be testing if our application runs fine and looks the same on the three web browsers in the non-functional requirement. We will check the Forums page, the Marketplace page, and the Chat page and see if there are any visual glitches or inconsistencies.

HW and SW Setup:
We will test on a computer or laptop using a supported operating system. We will test on each of the three supported browsers. For this test, we used two computers: a Windows 10 computer for Google Chrome 105 and Microsoft Edge 105 testing, and a Mac for Safari 15.5.
The first URL to test is the forums page: http://54.241.101.69/forums
The second URL is the marketplace page: http://54.241.101.69/marketplace
The third URL is the chat page: http://54.241.101.69/chat

All three URLs require the user to be logged in.

Feature to be Tested:
We will be testing the Forums page, the Marketplace page, and the Chat page, and perhaps some other random pages and buttons. The features we are testing are the appearance of the pages, their speed, and whether the pages perform as expected or not.

QA Test Plan Table:

Test #	Test Title	Test Description	Test Input	Expected Correct Output	Test Results
1	Test on Google Chrome 105	Use the application with Google Chrome 105 and see if everything performs as expected.	Use Google Chrome 105 and go to the Forums, Marketplace, and Chat page. Press buttons and click on links.	The pages should look the same as it does on every other supported browser, and the functions should be the same too. Nothing should be broken.	Pass
2	Test on Microsoft Edge 105	Use the application with Microsoft Edge 105 and see if everything performs as expected.	Use Microsoft Edge 105 and go to the Forums, Marketplace, and Chat page. Press buttons and click on links.	The pages should look the same as it does on every other supported browser, and the functions should be the same too. Nothing should be broken.	Pass
3	Test on Safari 15.5	Use the application with Safari 15.5 and see if everything performs as expected.	Use Safari 15.5 and go to the Forums, Marketplace, and Chat page. Press buttons and click on links.	The pages should look the same as it does on every other supported browser, and the functions should be the same too. Nothing should be broken.	Pass

b) We have performed the tests and verified that everything works as expected. All tests have passed, and the results of our application match up with the Expected Correct Output. We used various browsers and everything appears to function as expected.

4) Code Review

a)
The coding style we chose has: a maximum of 130 characters per line (with a few exceptions, such as with npm scripts), the use of semicolons in JavaScript, and our code is indented with tabs with a tab size of 4 spaces.

We try to reduce the number of lines in our code by using single-line if statements if it does not impact readability too much, and we use spaces in our code to make it more readable, e.g. we do `{ title, description, price }` rather than `{title,description,price}`.

Our JavaScript variables are written in camelCase, while the attributes in our database are in snake_case. Most of our functions should have a JSDoc comment that explains what the function does, written above the function's signature. They are formatted as follows: /** comment */
Most of our files should have a header comment that explains what the file is for.

b)
1. Anthony Zhang submitted code to the members in his team (Team 7) for peer review. It is a 2 page snippet of the Create Listing page's front end code. It includes the React states and the fetch() function used to send the request to the back end.

2. Anthony Zhang submitted code to Team 5's Team Lead, Zhenyu Lin, and to Team 6's Team Lead, Alex Sanchez, for peer review. It is a 2 page snippet of the Create Thread page's back end code. In particular, it is the controller function which handles the back end logic, and one of the model functions which handles the database querying.

We peer reviewed Team 5's code since they peer reviewed our code.
We received Team 6's peer review and their own code on 12/1, so we did not have too much time to discuss the feedback they left us and peer review their code.

3. **Code submitted to the team for review in Section 1:**
We chose this code because it shows how we use React states, how we document our code with JSDoc and in-line comments, how the front end uses fetch() to communicate with the back end, and how we handle the data that is returned by the back end. The Create X pages we have (Create Thread, Create Listing, Create Group) are all very similar code-wise, so the peer review we do for the Create Listing page will benefit all of our Create X pages.

```
/* This file handles the display of the Create Listing page for the Gatormmunity Marketplace.
 * The page takes user input for creating the listing, which is sent to the back end for validation
and listing creation. */

import { useContext, useEffect, useRef, useState } from "react";
import { Button, Col, Form, Container, Row } from "react-bootstrap";
import { UserContext } from '../App.js';
import { Navigate, useNavigate } from 'react-router-dom';
import HelpfulLinksList from "../components/help/HelpfulLinksList";
import * as questions from '../components/help/Questions';
import { ERROR_STATUS, LISTING_CATEGORIES, SUCCESS_STATUS } from "../components/Constants.js";

export default function CreateListing() {
    const userSession = useContext(UserContext); // the user's session data
    const navigate = useNavigate();
    const itemPhotoInput = useRef();

    /** The links that will be shown in the helpful links list. */
    const links = [questions.meetingBuyerSeller, questions.reportUser];

    // contains the form's data
    const [form, setForm] = useState({
        listingTitle: "",
        listingDescription: "",
        price: "",
        category: LISTING_CATEGORIES[0] // the default category
    });

    const [itemPhoto, setItemPhoto] = useState(null); // stores the photo of the item
    const [returnData, setReturnData] = useState(null); // stores the data sent back from the back
end

    /** Updates the create listing form's state. */
    function updateForm(value) {
        return setForm((prev) => {
            return { ...prev, ...value };
```

```
        });
    }

    /**
     * Sends the form to the back end to create the listing.
     *
     * The front end sends:
     * listingTitle: {string} The listing's title. Required; must be 1-255 characters.
     * listingDescription: {string} The listing's description. Required; must be 1-2'500 characters.
     * price: {string} The item's price. Required; must be in a valid currency format, e.g. 21.55 or
21; must be non-negative.
     * category: {string} The item's category. Required; must be one of the predefined category
options.
     * sellerId: {int} The seller's user id. Required; must be a positive integer.
     * itemPhoto: {File} The item's photo. Required; must be JPEG, PNG, WebP, GIF, or AVIF; cannot
exceed 5 MB.
     */
    async function createListing(e) {
        e.preventDefault();

        /** The form that we send to the backend containing the listing's data. */
        const formData = new FormData();

        // append the form's content to `formData`
        for (const index in form) formData.append(index, form[index]);
        formData.append("sellerId", userSession.user_id);
        formData.append("itemPhoto", itemPhoto);

        await fetch("/api/listings/create-listing", {
            method: "POST",
            body: formData
        })
            .then((res) => res.json())
            .then((data) => setReturnData(data))
            .catch(console.log());
    }

    /* Redirects to the newly created listing's page upon successful listing creation. Otherwise,
show an error. */
    useEffect(() => {
        if (returnData?.status === SUCCESS_STATUS) navigate(`/listing/${returnData.listingId}`);
        else if (returnData?.status === ERROR_STATUS) alert(returnData.message);
    }, [returnData]); // eslint-disable-line react-hooks/exhaustive-deps
```

Code submitted to Team 5 and 6 for review in Section 2:
We chose this code from our Create Thread page's back end because its structure is similar to the rest of our back end code. It shows the JSDoc comment describing what the back end sends to the front end, how we initialize our variables from what the front end sends with req.body, and the .then() promise chain which has us perform multiple database queries to insert the thread, post, and attachment into the database, if an image was provided. Also, we show our error-handling code.

```
/**
 * Creates a thread either for Gatormmunity or for a group, depending on what groupId's value is.
 * The thread's original post and optionally, its attachment, is created too.
 *
 * The back end sends:
 * On successful thread creation:
 *     status: {string} "success",
 *     threadId: {int} The id of the new thread.
 *
 * On failure:
 *     status: {string} "error",
 *     message: {string} The error message to display the user.
 */
exports.createThread = (req, res) => {
    let returnData = {}; // holds the data that will be sent to the front end
    const { threadTitle, threadBody, category, groupId, creatorId } = req.body;

    /* The thread image is optional, thus we check if `req.file` exists first before assigning these
variables a value. */
    const threadImagePath = req.file ? req.file.path : null;
    const threadImageThumbnailPath = req.file ? path.join(req.file.destination,
`tn-${req.file.filename}`) : null;
    const threadImageOriginalFilename = req.file ? req.file.originalname : null;

    createThumbnail(req.file, threadImagePath, threadImageThumbnailPath, 250, 250)
        .then(() => { // if the thumbnail was successfully created, create the thread
            return ThreadModel.createThread(threadTitle, category, groupId, creatorId);
        })
        .then((threadId) => {
            if (threadId < 0) throw new Error("Error with createThread().");

            returnData.threadId = threadId;
            return PostModel.createPost(threadBody, true, threadId, creatorId); // create the
thread's original post
        })
        .then((postId) => {
```

```
            if (postId < 0) throw new Error("Error with createPost().");

            // If a thread image was provided, attach it to the post, otherwise, we are done and can
send back `returnData`.
            if (threadImagePath) {
                return PostModel.createAttachment(threadImageOriginalFilename, threadImagePath,
                                                  threadImageThumbnailPath, postId);
            } else {
                returnData.status = SUCCESS_STATUS;
                res.json(returnData); // send the success to the front end

                return Promise.reject("SKIP"); // break out of the .then() promise chain early
            }
        })
        .then(attachmentId => {
            if (attachmentId < 0) throw new Error("Error with createAttachment().");

            returnData.status = SUCCESS_STATUS;
            res.json(returnData); // send the success to the front end
        })
        .catch((err) => {
            if (err === "SKIP") return; // if returnData has already been sent, do nothing

            returnData = { status: ERROR_STATUS };
            returnData.message = "An error occurred while creating your thread.";

            // delete the uploaded files on failed thread creation, if they exist
            if (threadImagePath) {
                fs.unlink(threadImagePath, () => { });
                fs.unlink(threadImageThumbnailPath, () => { });
            }

            console.log(err);
            res.json(returnData); // tell the front end of the failure
        })
};
```

Code showing how we query the MySQL database using JavaScript in PostModel.createPost():

```
/**
 * Inserts a new post into the database.
 *
 * @param {string} body The body of the thread.
```

```
 * @param {boolean} is_original_post Whether the post is the original post, i.e. the post created
when the thread is created.
 * @param {int} thread_id The id of the thread the post belongs to.
 * @param {int} author_id The id of the user who created the post.
 * @returns On success, the new post's post_id. On failure, -1.
 */
PostModel.createPost = (body, is_original_post, thread_id, author_id) => {
    const insertSQL = `INSERT INTO post (body, is_original_post, thread_id, author_id)
                       VALUES (?, ?, ?, ?);`;

    return database
        .query(insertSQL, [body, is_original_post, thread_id, author_id])
        .then(([results]) => {
            if (results.affectedRows) return results.insertId;
            else return -1;
        })
        .catch((err) => Promise.reject(err));
};
```

Discussion about the Review for Section 1:

The code we sent to our own team in Section 1 received this feedback:
We do not have comments indicating where a function starts and ends. A client who does not know much about coding will be confused by our code. We should use React Class components rather than React Functional components. Our code and comments should be written such that non-computer scientists will be able to understand it.

In our discussion about the review, we thought that our files are becoming rather long and we should try to split up the larger files into smaller files so that it is easier to navigate through the code. This is especially true for the code we have in our back end.
We could benefit from having more in-line comments and more header comments that describe the purpose of the file, and some of our functions are missing JSDoc comments that describe the function, its parameters, and its return values.

Discussion about the Review for Section 2:

The code we sent to Team 5 in Section 2 received this feedback:
The reviewer was not sure what an original post was; we should elaborate on what it is in a comment. It is not clear what an attachment is: is it any kind of file, a text file, an image file? We should state what it is more explicitly. Someone unfamiliar with our website might think ‘thread’ refers to process threads, rather than a forum thread.

In our discussion about the review, we felt that the feedback was fair. While our milestone document clarifies what an original post is, it might help to have it in a code comment too for easy reference. The same goes for attachments: the milestone documents say it is an image, but the code that we sent to Team 5 did not explicitly say as such.
We coined the term “attachment” back in the early stages of development when we would allow users to upload any type of file. Now, we only allow images for the sake of simplicity, but we kept the term “attachment” since it was what we were used to. We might have to change the name of the table and the code to make it more clear to any new developers who would work on the codebase.

Team 6 left us this feedback about our code in Section 2:
We should have a header comment for our file with threadController in it. We should not display the original filename of the thread’s image because it could be inappropriate and get us in trouble.

In our discussion about the review, we felt that the feedback was fair. We should spend more time writing comments in our code, and that is what we will be focusing on in the coming days. All P1s are implemented, so all that is left to do is to make small improvements. We have moderators and admins who can delete threads and ban any user who tries to post inappropriate things, but if we feel that the potential for abuse is too high, we will hide the original filename.

5) Self-Check on Best Practices for Security

Major Assets being Protected:

- Passwords are hashed before being stored in the database. Thus, nobody can know the password of another user, not even the developers who have access to the database.

- Only approved users (users who were manually approved by a moderator after registering) can login and use Gatormmunity. Unapproved users and people not logged in cannot view most of the pages on Gatormmunity, including the Forums, Marketplace, Groups, Chat, and Inbox page. However, everyone can access the Terms of Service and Contact Us page, for example.

- Only moderators and administrators can access the Administration page. On the Administration page, moderators can approve/reject unapproved users and ban approved users. Administrators can do that, as well as appoint and unappoint moderators.

- Only moderators and administrators can view the SFSU ID number and SFSU ID card picture of Gatormmunity's users. This is necessary because moderators need to be able to see these credentials in order to approve/reject unapproved users.

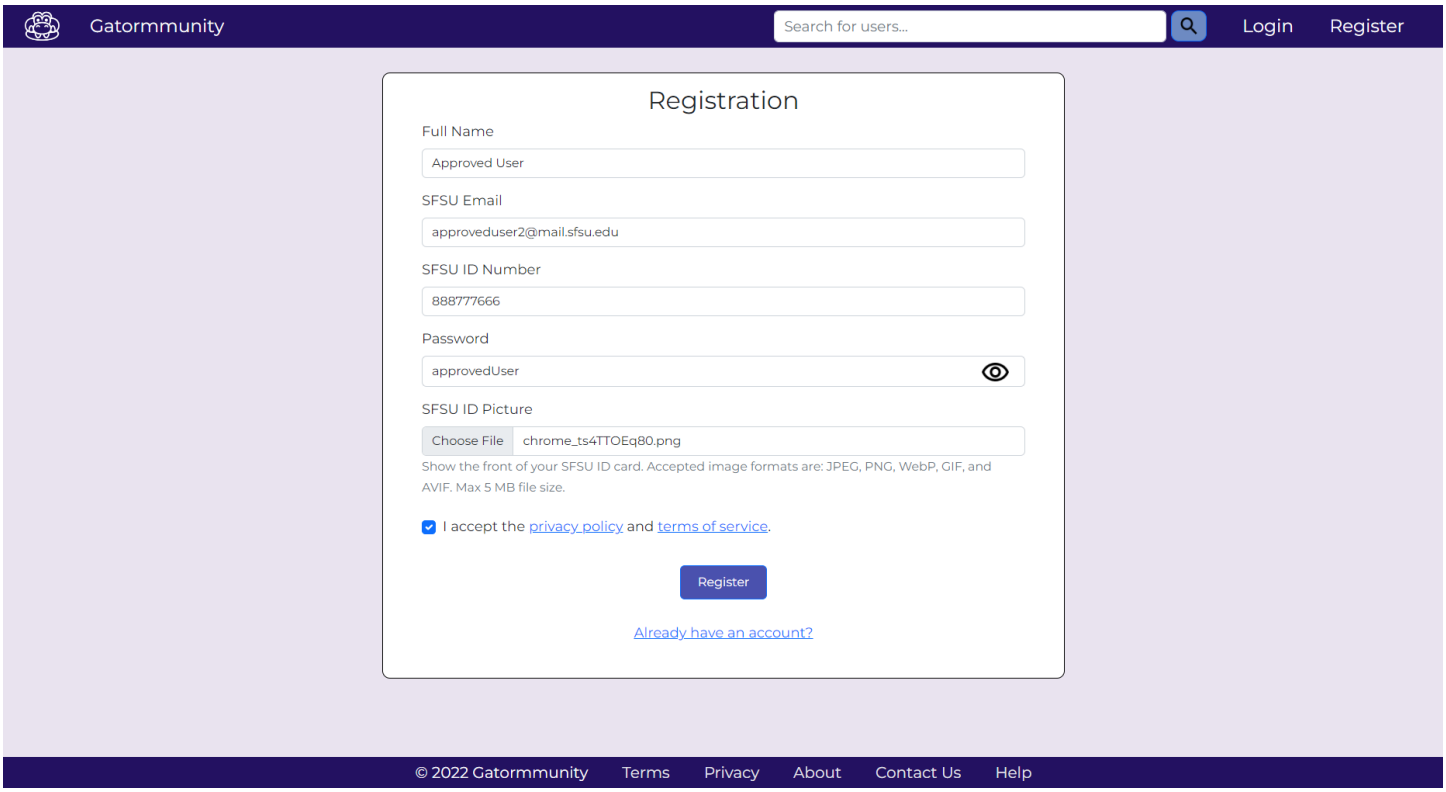
- Only a group's members can see the threads and posts in that group's forums as well as send and receive messages in that group's chat.

Password Encryption Process:

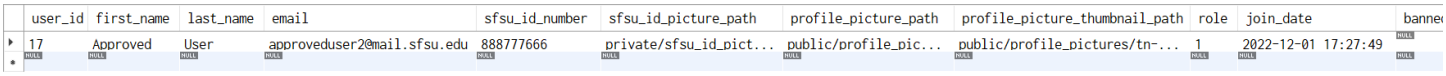
Passwords are hashed before being stored in the database. When a user registers for an account, we take their plaintext password and use the bcrypt npm library's hash function to hash that plaintext password with 10 rounds. That hashed password is then inserted into the account table in our database. We store the user's password in a separate table from the rest of the user's data.

We show real examples (screenshots) below.

First, we register for a new account with these credentials. We can show/hide the password by pressing the eye icon in the password field.

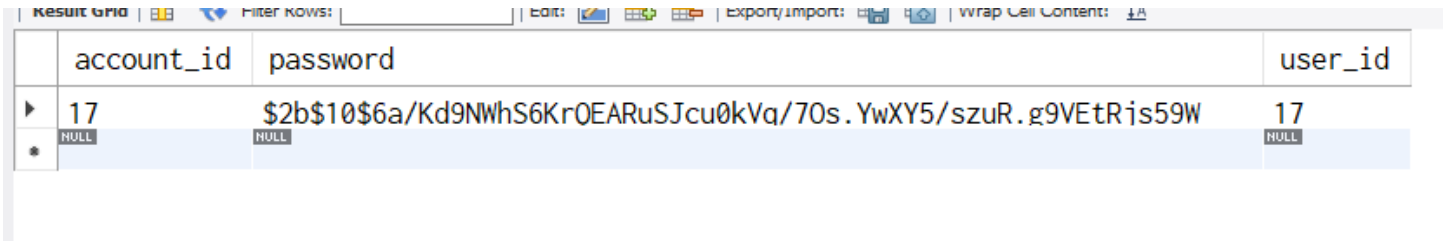


Here is the newly created user's entry in the 'user' table of our database. We do not store the password in the 'user' table. As you can see, they have a user_id of 17.



user_id	first_name	last_name	email	sfsu_id_number	sfsu_id_picture_path	profile_picture_path	profile_picture_thumbnail_path	role	join_date	banne
17	Approved	User	approveduser2@mail.sfsu.edu	888777666	private/sfsu_id_pict...	public/profile_pic...	public/profile_pictures/tn-...	1	2022-12-01 17:27:49	

Here is the newly created user's entry in the 'account' table of our database, which stores the user's password. The user_id foreign key is 17, the same as the user_id from the 'user' table. As you can see, the password is hashed and is NOT stored in plaintext.



account_id	password	user_id
17	$2b$10$6a/Kd9NWhS6KrQEARuSJcu0kVq/7Os.YwXY5/szuR.g9VEtRjs59W	17

Input Data Validation:

We validate input on both the client and server-side. The client-side performs basic validation such as requiring required fields, only allowing certain types of input (e.g. numbers), and checking the email format. The server-side performs thorough validation.

We validate:

- **Images:** All images must be either JPEG, PNG, WebP, GIF, or AVIF file types, and they must be within 5 MB.

- **Direct Messages:** The message body must be between 1-5000 characters, and the provided conversation_id and sender_id must exist, be positive, and be integers.

- **Groups:** An image must be uploaded. The group's name must be between 1-255 characters, the group's description must be at most 5000 characters, the provided admin_id must exist, be positive, and be an integer.

- **Chat Messages:** The message body must be between 1-5000 characters, and the provided group_id and sender_id must exist and be integers. sender_id must be positive.

- **Group Announcements:** The announcement must be at most 5000 characters, and the provided group_id and user_id must exist, be positive, and and be integers.

- **Listings:** An image must be uploaded. The listing title must be 1-255 characters, the description 1-2500 characters, the price must be a decimal with at most 2 decimal places, the price must be from 0.00 to 99'999.99, the category must be one of those in the dropdown, and the seller_id must exist, be an integer, and be positive.

- **Forum Threads:** The thread title must be between 1-255 characters, the body must be between 1-10'000 characters, the category must be one of those in the dropdown, and the creator_id must exist, be an integer, and be positive.

- **Forum Posts:** The post body must be between 1-10'000 characters, the thread_id and author_id must exist, be an integer, and be positive.

- **Registration:** An SFSU ID picture must be uploaded. The full name must be between 1-100 characters, and a first and last name must be provided. The email must have an @ symbol, must end with "sfsu.edu", and must be between 1-255 characters. The SFSU ID number must be 9 digits long, be an integer, and be positive. The password must be 6-64 characters long.

- **Listing Search Input:** The category filter must be one of those in the dropdown, and the max price filter must be a decimal with at most 2 decimal places.

- **Thread Search Input:** The category filter must be one of those in the dropdown.

- **User Search Input:** The role filter must be one of those in the dropdown.

The code that we use to validate user input is stored in the directory: ‘server/middleware’. We use the yup npm library to validate the forms, while we use regular JavaScript to validate the image file type and file size.

6) Self-Check: Adherence to Original Non-Functional Specs

1. System Requirements
   1.1. The AWS server's region shall be North California, United States.
        DONE
   1.2. The database shall be stored on the AWS server.
        DONE

2. Performance Requirements
   2.1. The application should be available for at least 23 hours a day.
        DONE
   2.2. The application's homepage should load in at least 7 seconds.
        DONE
   2.3. The application should be accessible from anywhere in the world.
        DONE

3. Storage, Security, and Environmental Requirements
   3.1. The server's storage shall be a 30 GiB volume.
        DONE
   3.2. The database shall be secured with a password.
        DONE
   3.3. Passwords shall be hashed before being stored in the database. In other words, there shall be no plaintext passwords stored in the database.
        DONE

4. Usability Requirements
   4.1. Guest users shall accept the privacy policy and terms of service before they can register for an account.
        DONE
   4.2. Approved users should abide by the terms of service.
        DONE

5. Marketing and Legal Requirements
   5.1. The application shall be known as Gatormmunity in its marketing.
        DONE
   5.2. The application shall be known as Gatormmunity in its licensing.
        DONE
   5.3. The application shall have a logo.
        DONE
   5.4. The application shall have a privacy policy.
        DONE
   5.5. The application shall have terms of service.
        DONE

6. Content Requirements
    6.1. The application shall support the English language.
        DONE
    6.2. The application shall support US Dollars as currency.
        DONE
    6.3. Approved users shall only upload files less than or equal to 5 MB in size.
        DONE

7. Privacy Requirements
    7.1. The application shall collect and store personal data from its users.
        DONE
    7.2. The application shall use the personal data of its users for verifying their identity.
        DONE
    7.3. The application shall store and use its users' credentials for logging them in.
        DONE
    7.4. The application shall not reveal any of a user's private data to any other user.
        DONE

8. Compatibility Requirements
    8.1. The application shall support the Windows 10, Windows 11, and macOS Monterey 12.5 and 12.6 operating systems.
        DONE
    8.2. The application shall support the Google Chrome 105, Microsoft Edge 105, and Safari 15.5 web browsers.
        DONE
    8.3. The application should not be designed for mobile devices.
        DONE
    8.4. The application should be designed for and work on laptops.
        DONE
    8.5. The application should be designed for and work on desktop computers.
        DONE

9. Organizational Requirements
    9.1. The AWS server should not incur any costs to the developers.
        DONE
    9.2. The developers should use semicolons in their JavaScript code.
        DONE
    9.3. The developers should use tabs in their code, and not spaces.
        DONE
    9.4. The developers should not have lines of code that exceed 130 characters in length.
        DONE

9.5. The developers should document their code with comments.
DONE

9.6. The application's code should not be messy.
DONE

9.7. The application shall use working and tested code. In other words, faulty and untested code shall not be served from the remote server.
DONE

9.8. The master branch of the application's GitHub repository shall contain only working and tested code.
DONE

9.9. The documentation and technical reports shall have a table of contents.
DONE

9.10. The documentation and technical reports shall have page numbers.
DONE

9.11. The documentation and technical reports shall start each section on a new page.
DONE

7) List of Contributions to the Document and Contribution Scores

Anthony Zhang:

- Was an active participant in the meetings and team Discord server
- Sent the email to Ortiz for M4V1 Section 7
- Updated M3V2
- Wrote Sections 1, 2, 3, 4, 5, 7 of M4V1: Product Summary, Usability Test Plan, QA Test Plan, Code Review, Self-Check on Best Practices for Security, List of Contributions
- Helped peer review Team 5's and Team 6's code
- Helped test our live website by creating forum threads, creating forum posts, creating listings, sending chat messages, creating groups, joining groups, approving users, and got a tester to test our app and fill out the questionnaire
- Added Leave Group button for group members, and a Delete Group button for group admins
- Added Delete Thread, Delete Post, and Delete Listing buttons for moderators; sellers can delete their listings too
- Added show password eye button to the Login page
- Wrote the front end fetch functions for and worked on these pages: Administration, Chat, Create Thread, Dashboard, Gatormmunity Forums, Group Forums, Group Home, Group Members, Inbox, Join Group, Search, User Profile, View Thread
- Worked on the back end for the components: Change Profile Picture
- Wrote the back end code for the pages: Administration, Create Listing, Create Thread, Group Create Thread, Group Forums, Group Members, Inbox, Join Group, Login, Search, User Profile, View Thread

Marwan Alnounou:

- Was an active participant in the meetings and team Discord server
- Wrote Section 6 of M4V1: Adherence to Original Non-Functional Specs
- Sent the email to Ortiz for M4V1 Section 7
- Helped test our live website by sending chat messages
- Wrote some rules for Gatormmunity in the Terms of Service
- Added a Group Forums button for each of the groups in the Users Groups page
- Worked on the front end for the components: Group Change Announcement, Group Invite, Navbar
- Wrote the front end fetch functions for and worked on these pages: Group Members, Marketplace, View Listing

Mohamed Sharif:

- Was an active participant in the meetings and team Discord server
- Helped peer review our own code and Team 5's code
- Worked on the back end for the components: Group Invite
- Wrote the back end code for the pages: Group Home, User Profile, Users Groups, View Listing

Jose Lopez:

- Was an active participant in the meetings
- Sent the email to Ortiz for M4V1 Section 7
- Helped peer review Team 5's code
- Helped test our live website by creating forum threads, creating listings, sending chat messages, and banning users
- Helped make the styling more consistent for the Create pages (Create Thread, Create Group, etc) by having the pages share one CSS file
- Moved the login page's error messages from inside the page to an alert message box
- Wrote the front end fetch functions for and worked on these pages: Create Group, Create Listing, Group Create Thread, Group Home, Users Groups

Florian Cartozo:

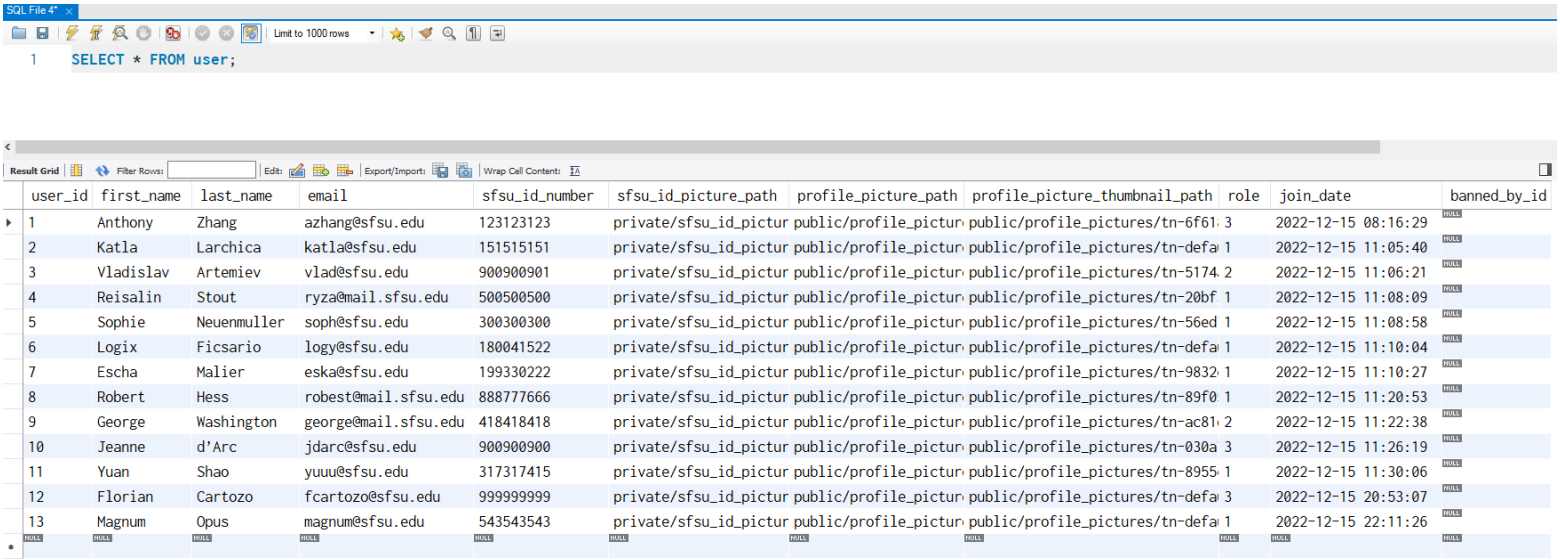
- Sent the email to Ortiz for M4V1 Section 7
- Helped peer review Team 5's code
- Helped test our live website by creating forum posts, joining groups, sending chat messages, sending emails via the contact us page, and got a tester to test our app and fill out the questionnaire
- Created M4's EER Model and Database
- Added a message is currently sending indicator for the Contact Us page
- Worked on the back end for the components: Group Change Announcement
- Wrote the back end code for the pages: Chat, Create Group, Dashboard, Gatormmunity Forums, Marketplace

Contribution Scores:

Anthony Zhang: 10
Marwan Alnounou: 10
Mohamed Sharif: 8
Jose Lopez: 9
Florian Cartozo: 10

4) Screenshots of Key DB Tables

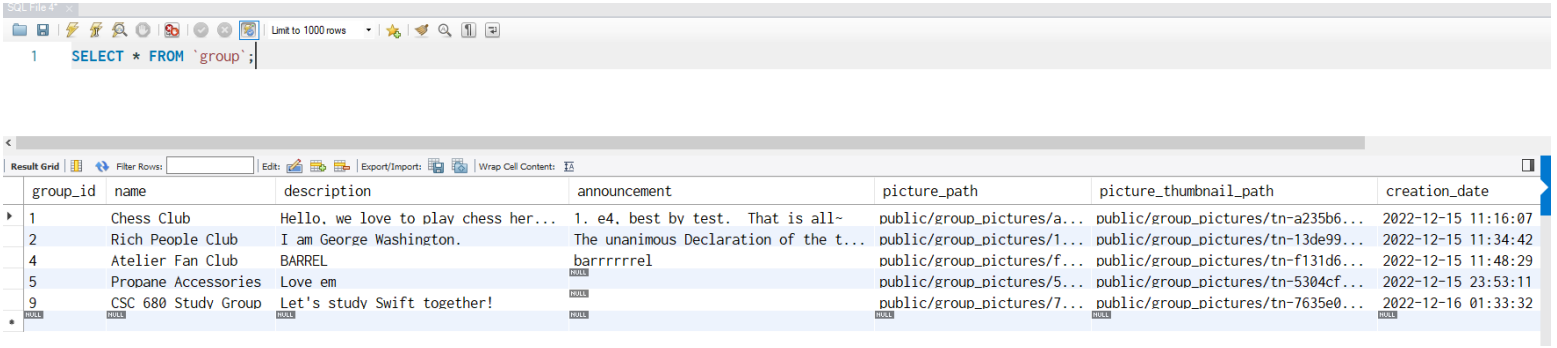
The user table which stores our users. The paths to the users' pictures could not be shown in full because they were too long:



```
SELECT * FROM user;
```

user_id	first_name	last_name	email	sfsu_id_number	sfsu_id_picture_path	profile_picture_path	profile_picture_thumbnail_path	role	join_date	banned_by_id
1	Anthony	Zhang	azhang@sfsu.edu	123123123	private/sfsu_id_pictur	public/profile_pictur	public/profile_pictures/tn-6f61	3	2022-12-15 08:16:29	NULL
2	Katla	Larchica	katla@sfsu.edu	151515151	private/sfsu_id_pictur	public/profile_pictur	public/profile_pictures/tn-defa	1	2022-12-15 11:05:40	NULL
3	Vladislav	Artemiev	vlad@sfsu.edu	900900901	private/sfsu_id_pictur	public/profile_pictur	public/profile_pictures/tn-5174	2	2022-12-15 11:06:21	NULL
4	Reisalin	Stout	ryza@mail.sfsu.edu	500500500	private/sfsu_id_pictur	public/profile_pictur	public/profile_pictures/tn-20bf	1	2022-12-15 11:08:09	NULL
5	Sophie	Neuenmuller	soph@sfsu.edu	300300300	private/sfsu_id_pictur	public/profile_pictur	public/profile_pictures/tn-56ed	1	2022-12-15 11:08:58	NULL
6	Logix	Ficsario	logy@sfsu.edu	180041522	private/sfsu_id_pictur	public/profile_pictur	public/profile_pictures/tn-defa	1	2022-12-15 11:10:04	NULL
7	Escha	Malier	eska@sfsu.edu	199330222	private/sfsu_id_pictur	public/profile_pictur	public/profile_pictures/tn-9832	1	2022-12-15 11:10:27	NULL
8	Robert	Hess	robest@mail.sfsu.edu	888777666	private/sfsu_id_pictur	public/profile_pictur	public/profile_pictures/tn-89f0	1	2022-12-15 11:20:53	NULL
9	George	Washington	george@mail.sfsu.edu	418418418	private/sfsu_id_pictur	public/profile_pictur	public/profile_pictures/tn-ac81	2	2022-12-15 11:22:38	NULL
10	Jeanne	d'Arc	jdarc@sfsu.edu	900900900	private/sfsu_id_pictur	public/profile_pictur	public/profile_pictures/tn-030a	3	2022-12-15 11:26:19	NULL
11	Yuan	Shao	yuuu@sfsu.edu	317317415	private/sfsu_id_pictur	public/profile_pictur	public/profile_pictures/tn-8955	1	2022-12-15 11:30:06	NULL
12	Florian	Cartozo	fcartozo@sfsu.edu	999999999	private/sfsu_id_pictur	public/profile_pictur	public/profile_pictures/tn-defa	3	2022-12-15 20:53:07	NULL
13	Magnum	Opus	magnum@sfsu.edu	543543543	private/sfsu_id_pictur	public/profile_pictur	public/profile_pictures/tn-defa	1	2022-12-15 22:11:26	NULL

The group table which stores our user groups:

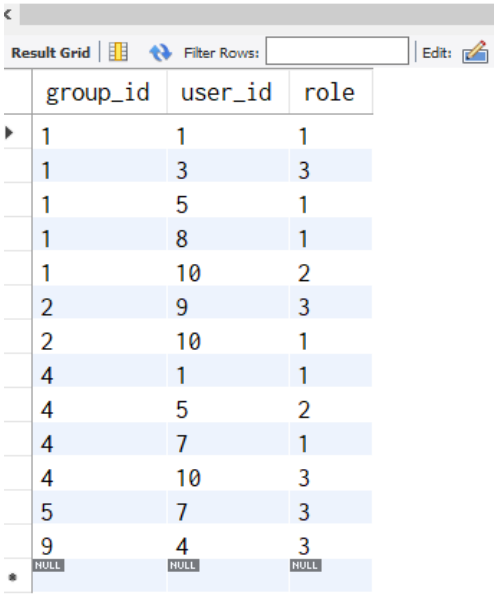


```
SELECT * FROM `group`;
```

group_id	name	description	announcement	picture_path	picture_thumbnail_path	creation_date
1	Chess Club	Hello, we love to play chess her...	1. e4, best by test. That is all~	public/group_pictures/a...	public/group_pictures/tn-a235b6...	2022-12-15 11:16:07
2	Rich People Club	I am George Washington.	The unanimous Declaration of the t...	public/group_pictures/1...	public/group_pictures/tn-13de99...	2022-12-15 11:34:42
4	Atelier Fan Club	BARREL	barrrrrrel	public/group_pictures/f...	public/group_pictures/tn-f131d6...	2022-12-15 11:48:29
5	Propane Accessories	Love em	NULL	public/group_pictures/5...	public/group_pictures/tn-5304cf...	2022-12-15 23:53:11
9	CSC 680 Study Group	Let's study Swift together!	NULL	public/group_pictures/7...	public/group_pictures/tn-7635e0...	2022-12-16 01:33:32

The group_user table, which stores a user's membership within a group and their role in that group:

```
1    SELECT * FROM group_user;
```



group_id	user_id	role
1	1	1
1	3	3
1	5	1
1	8	1
1	10	2
2	9	3
2	10	1
4	1	1
4	5	2
4	7	1
4	10	3
5	7	3
9	4	3
NULL	NULL	NULL

The listing table which stores the marketplace listings:

```
1    SELECT * FROM listing;
```

listing_id	title	description	price	category	image_path	image_thumbnail_path	creation_date	seller_id
1	Queen Chess Piece	Brand new. Ivory.	3.99	Miscellaneous	public/l...	public/listing_phot...	2022-12-15 11:36:34	9
2	Propane and Propane Accessories	I tell you hwhat	16.00	Miscellaneous	public/l...	public/listing_phot...	2022-12-15 11:41:40	9
3	12-pack of Diet Coke	Hi, I'm moving soon and I can't drink ...	6.00	Perishables	public/l...	public/listing_phot...	2022-12-15 11:43:40	9
4	iPhone 14 [NEW]	Colour: Green Model: iPhone 14 Stora...	800.00	Electronics	public/l...	public/listing_phot...	2022-12-15 11:44:58	9
5	Full Gothic Late 15th Century Plate...	this armour is in mint-condition, comp...	1800.00	Apparel	public/l...	public/listing_phot...	2022-12-15 11:58:40	10
6	Kettle Helmet	Good for archers!	175.00	Apparel	public/l...	public/listing_phot...	2022-12-15 12:00:54	10
7	Tenshi Eating a Corndog NFT	Tenshi eating a corndog. You can be it...	200.00	Perishables	public/l...	public/listing_phot...	2022-12-15 12:15:14	1
8	Antique XBOX 360	It's pretty well-worn but I'll throw i...	85.99	Electronics	public/l...	public/listing_phot...	2022-12-15 12:22:46	1
9	GFUEL, 1 can	Will keep you up for hours, guaranteed...	3.00	Perishables	public/l...	public/listing_phot...	2022-12-15 23:23:58	5
10	Chess Set with 2 Spare Queens, 1 bl...	I'm down to meet on campus to sell thi...	20.00	Entertainment	public/l...	public/listing_phot...	2022-12-15 23:43:53	5
11	puni	1 blue puni for sale, come and get it~	100.00	Entertainment	public/l...	public/listing_phot...	2022-12-15 23:49:35	1

The thread table which stores our forum threads for both the Gatormmunity and Group Forums. Threads with no group_id are Gatormmunity forum threads.

```
1  SELECT * FROM thread;
```

thread_id	title	category	creation_date	group_id	creator_id
1	Fellow Americans	Discussion	2022-12-15 11:37:59	NULL	9
2	How do I lower the gas ...	Help	2022-12-15 11:39:54	NULL	9
3	I am selling diet coke	Promotion	2022-12-15 11:45:57	NULL	9
4	How might we raise taxes?	Discussion	2022-12-15 11:46:44	2	9
5	Burgundians	Off-Topic	2022-12-15 11:52:24	NULL	10
6	barrel	Discussion	2022-12-15 11:56:20	4	10
7	Join Chess Club!	Promotion	2022-12-15 12:06:26	NULL	3
8	How did everyone do on ...	General	2022-12-15 12:07:09	NULL	3
9	Anyone down to party af...	Social	2022-12-15 12:11:06	NULL	3
10	Anyone play Atelier?	Off-Topic	2022-12-15 12:12:59	1	3
11	What's everyone up to?	General	2022-12-15 12:17:34	2	10
12	Algebra Homework help n...	Help	2022-12-15 12:19:33	NULL	10
13	Cirno Appreciation Post	Off-Topic	2022-12-15 23:31:05	NULL	5

The post table which stores the forum posts. Forum posts can be thought of as replies to forum threads.

```
1  SELECT * FROM post;
```

post_id	body	is_original_post	creation_date	thread_id	author_id
1	Four score and seven years ago our fathers brought forth on this continent a new na...	1	2022-12-15 11:37:59	1	9
2	I am president but it is out of my power... What shall I do?	1	2022-12-15 11:39:54	2	9
3	It's in the marketplace and nobody is buying it. Please buy it, I am leaving very ...	1	2022-12-15 11:45:57	3	9
4	please	0	2022-12-15 11:46:07	3	9
5	Any ideas?	1	2022-12-15 11:46:44	4	9
6	Dm'd	0	2022-12-15 11:48:56	3	10
7	No taxation without representation, if you ask me	0	2022-12-15 11:50:19	4	10
8	I am president	0	2022-12-15 11:50:32	2	10
9	awful	1	2022-12-15 11:52:24	5	10
10	discuss	1	2022-12-15 11:56:20	6	10
11	english	0	2022-12-15 11:59:09	5	10
12	skill issue	0	2022-12-15 12:05:29	5	3

The chat_message table which stores Gator Chat and Group Chat messages. Chat messages with no group_id are Gator Chat messages.

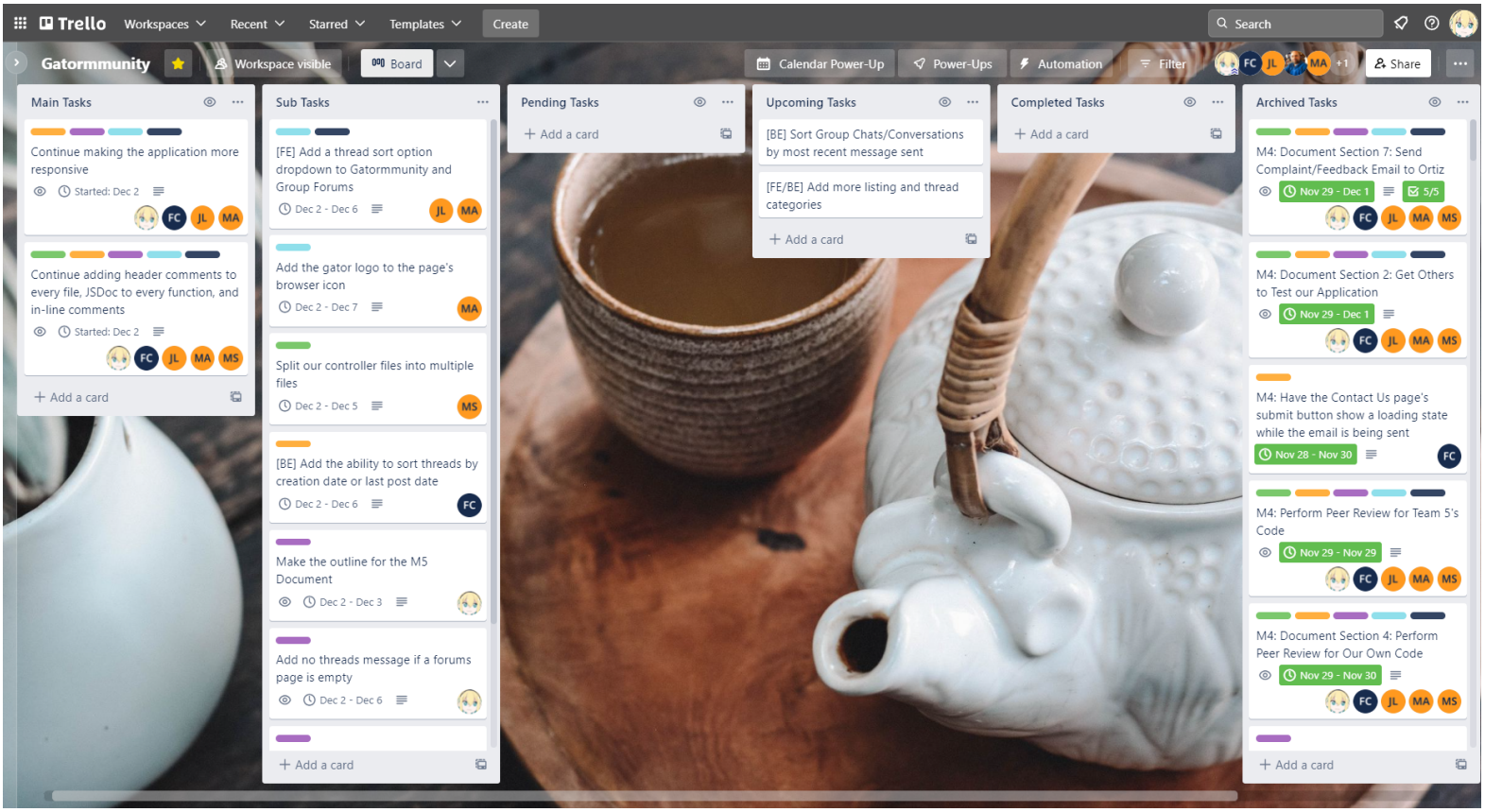
```
Limit to 1000 rows
1   SELECT * FROM chat_message;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

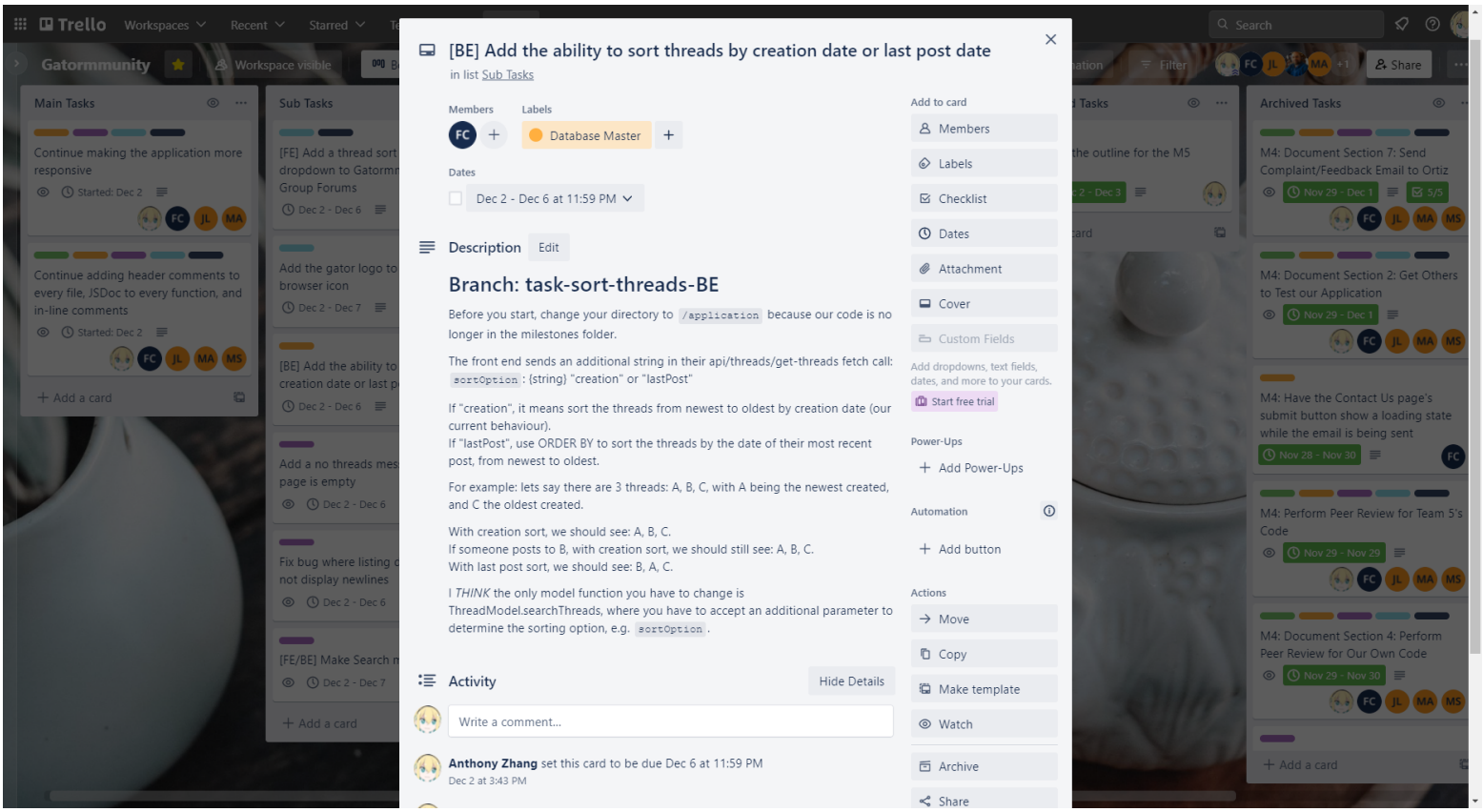
chat_message_id	body	creation_date	group_id	sender_id
2	Hi	2022-12-15 11:13:35	NULL	1
3	1. c4. discuss.	2022-12-15 11:16:55	1	3
4	Hey. thanks for the invite.	2022-12-15 11:17:32	1	1
5	I tell you hwhat	2022-12-15 11:41:59	NULL	9
6	I am George Washingmachine	2022-12-15 11:42:07	2	9
7	Hi everyone. I am selling diet coke	2022-12-15 11:45:21	NULL	9
8	Barel	2022-12-15 11:55:23	4	10
9	I am Joan of Arc	2022-12-15 11:55:30	2	10
10	No ploblem	2022-12-15 11:55:37	1	10
11	I dm'd already Please don't ignore me	2022-12-15 11:55:50	NULL	10
12	Hey everyone. please join my club if you like chess! h...	2022-12-15 12:02:54	NULL	3
13	discuss i said	2022-12-15 12:03:09	1	3
14	its ok	2022-12-15 12:03:21	1	10
15	👎	2022-12-15 12:03:37	NULL	10
16	http://54.241.101.69/join-group/4/f131d6ff-f91d-43e8-ab...	2022-12-15 12:04:54	NULL	10

5) Screenshots of Our Task Management System

Screenshot of our Trello board:



Screenshot of one of our tasks on Trello:



6) Team Member Contributions

Contributions to Milestone 5:

Anthony Zhang:

- Attended all of M5's meetings
- Completed the Milestone 5 Document
- Added and revised header comments, JSDoc comments, and inline comments for most of the front end and back end files
- Worked on making the application more responsive (i.e. made it look somewhat better on smaller screens)
- Made search easier to understand by making listings the default search category, improving the search results message, and adding button highlighting for the search options
- Added more Help/FAQ pages
- Helped with adding a "Sort By" dropdown for sorting the threads in the Gatormmunity Forums and Group Forums page
- Wrote the front end code that allows for the Enter key to send Chat and Direct Messages; wrote the front end code that lets admins change passwords
- Populated the database with users, threads, posts, listings, groups, chat messages, direct messages, etc.
- Wrote the back end code that sorted Group Chats and Conversations by most recent message sent; wrote the back end code that handles admins changing passwords

Marwan Alnounou:

- Added the Gator logo for Gatormmunity's favicon

Mohamed Sharif:

- Organized our back end code by splitting our overly large controller files into separate smaller files which were more manageable
- Tried to write the back end code that sorted Group Chats by most recent message sent

Jose Lopez:

- Helped with adding a "Sort By" dropdown for sorting the threads in the Gatormmunity Forums and Group Forums page

Florian Cartozo:

- Attended all of M5's meetings
- Wrote the back end code to sort the threads in the Gatormmunity Forums and Group Forums pages
- Wrote the back end model function for changing user passwords

<u>Contribution Scores for the Entire Semester:</u>

Anthony Zhang: 10
Marwan Alnounou: 6
Mohamed Sharif: 8
Jose Lopez: 7
Florian Cartozo: 9

8) Post Analysis

The experience of leading a team has been very educational and rewarding, and I am glad I applied for the position. When our team first assembled, I questioned whether I should be leading a team or not since I had no prior experience leading anybody and I was taking 5 classes this semester including Software Engineering, so there was a risk that I would not be able to handle the workload. Plus, I am not a particularly outgoing person so I was concerned if I would be able to lead the discussions within the team meetings. Now, I have no regrets, and I am thankful to the team for tolerating all of the mistakes I made.

We faced numerous challenges and setbacks, many of which could have been avoided or I could have handled better. In the first milestone, I set unrealistic expectations for everyone and evaluated the team too harshly as a result. In later milestones I adjusted my expectations so that instead of evaluating what a team member *could* have done, I evaluated what they actually did and if they were able to complete their assigned tasks on time and satisfactorily. In addition, we faced challenges related to communication in Milestone 2 because I did not communicate my expectations clearly enough, so there were disagreements about what I expected to see versus what the team thought I meant. I made an effort to remedy this by being more specific in the Trello task descriptions and encouraging members to ask questions if they had any. Also, in Milestone 2, I did not handle drama/conflict within the team well. I let conflicts get out of hand, which made the team uncomfortable and it hurt the team's morale. What I would do next time is address the conflict immediately and settle it privately so that the team would not have to be placed in uncomfortable situations.

Another challenge that we faced was that our project was a lot harder than we had anticipated back in Milestone 1. We had many ideas for features to add but it made our project too large and too difficult for our team of 5. Even though we cut many interesting features by moving them to Priority 2 or 3, there was still a lot we had to do if we were to implement everything in our use cases. Furthermore, the difficulty in our project ramped up significantly as we entered Milestone 3. If I were to do things over, I would not have chosen such an ambitious project, given everyone's schedules and skillsets. But we also could have done more to manage the difficulty of the project: during the relatively calm period of Milestones 1 and 2, I should have been more firm in asking everyone to study and practice their respective technologies (e.g. React) in preparation for Milestone 3. It was not fair to the back end developers that I asked them to work on the front end as well as the back end since that was not necessarily where their strengths lied. Next time, I will ensure that the team is better prepared for web development by requiring that each front end developer show proof that they are capable of making a nice-looking page with our chosen front end framework.

To summarize, I learned countless lessons in this class by leading a team and by making mistakes that I had better not make again. I learned a lot from both Professor Ortiz and the team, and the experience we gained with Figma, GitHub, and Trello has been invaluable for our futures as well. I am grateful that I was able to practice speaking to others and for being able to meet some very nice people in this class, including other team leads.