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
10/20/2022

History Table

Version	Submission Date
M2V1	10/20/2022
M2V2	N/A

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 home page, login page, and register page.
- Guest Users can register for an account by providing a first name, last name, email address, SFSU ID number, SFSU ID picture, and password. Optionally, they may upload a profile picture.
    - The pictures must be of an image format. Supported formats are: JPEG, PNG, WebP, GIF, and AVIF.
    - The pictures 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 home page, login page, and register page.
- Unapproved Users cannot login until their account has been approved by a moderator or admin.

Approved User: A person that registered for an account and has had their account approved by a moderator or admin.

- 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.

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.
- Moderators are appointed and unappointed by Server Administrators.

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.

Group Member: An approved user who is a member of a group.

- 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 private forum or sending messages in their group's private chat.
- Approved Users become Group Members when they join a group via an invite from another Group Member.

Group Moderator: An approved user who is a moderator of a group.

- 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.
- Group Moderators are appointed and unappointed by Group Administrators.

Group Administrator / Group Admin: An approved user who is an administrator of a group.

- Group Administrators have more privileges than Group Moderators, and have access to group administrative tools, including the ability to delete their group and appoint Group Moderators.

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 approved users can see these buttons in the navigation bar. Guest users can only see the logo, buttons that link to the login or registration page, and the search bar.

Dashboard: The page that approved users will see after logging in, similar to a home page for only logged in users.

- The dashboard will show the user's profile picture, name, the newest forum threads in the Gatormmunity forum, recent marketplace listings, and more.

Gatormmunity Forum:

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

Forum Category: The forum will have categories so that approved users can find threads that they are interested in.

- 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. 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. Forum posts are comparable to the replies in a discussion topic on iLearn, including the post that started the discussion topic.

- The first post that starts a thread is also considered a post, and is referred to as the “original post”.

Post Attachment: Approved users can attach multiple images to their forum posts.

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

Pin: A pinned forum thread will always appear first when displaying the 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:

Gatormmunity Marketplace / 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, accepted payment methods, possible delivery methods, preferred contact methods, and a category.

- 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 photos must be of an image format. Supported formats are: JPEG, PNG, WebP, GIF, and AVIF.
- The images 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.

- An image can be attached to a message, which 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.

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

- An image can be attached to direct messages, which 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.

Inbox: Direct messages from other approved users are stored in an approved user's inbox.

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.

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.

- The messages can have images which 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.

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 delete their account.
    - 3.4. An approved user shall be able to make a forum thread.
    - 3.5. An approved user shall be able to make a forum post.
    - 3.6. An approved user shall be able to attach images to their forum posts.
    - 3.7. An approved user shall have a profile page.
    - 3.8. An approved user shall be able to edit their own profile page.
    - 3.9. An approved user shall be able to view the profile page of another approved user.
    - 3.10. An approved user shall have a profile picture.
    - 3.11. An approved user shall be able to change their profile picture.
    - 3.12. An approved user shall be able to create a user group.
    - 3.13. An approved user shall be able to join a user group.
    - 3.14. An approved user shall be able to send direct messages to other approved users.
    - 3.15. An approved user shall be able to receive direct messages from other approved users.
    - 3.16. An approved user shall be able to send messages in Gator Chat.
    - 3.17. An approved user shall be able to see marketplace listings in the marketplace.
    - 3.18. An approved user shall be able to select marketplace listings in the marketplace.
    - 3.19. An approved user shall be able to report marketplace listings to a moderator.

3.20. An approved user shall be able to buy items from the marketplace.
3.21. An approved user shall be able to sell items on the marketplace.
3.22. An approved user shall be able to edit the details of an item they are selling.
3.23. An approved user shall be able to upload pictures of an item they are selling.
3.24. An approved user who is selling an item shall be able to list their accepted payment methods.
3.25. An approved user who is selling an item shall be able to list their possible delivery methods.
3.26. An approved user who is selling an item shall be able to list their preferred contact methods.
3.27. An approved user who is selling an item shall be able to assign their listing to a category.
3.28. An approved user shall be able to search for users.
3.29. An approved user shall be able to search for marketplace listings.
3.30. An approved user shall be able to search for forum posts.
3.31. An approved user shall be able to apply filters to their user search results.
3.32. An approved user shall be able to apply filters to their marketplace search results.
3.33. 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 reset their password.
    - 10.2. An approved user shall be able to attach images to their chat messages.
    - 10.3. An approved user shall be able to sort forum threads.
    - 10.4. An approved user shall be able to bookmark forum threads.
    - 10.5. An approved user shall be able to like forum posts.
    - 10.6. An approved user shall be able to report forum posts to a moderator.
    - 10.7. An approved user shall be able to report other users to a moderator.
    - 10.8. An approved user shall be able to block other users.
    - 10.9. An approved user shall be able to add items to a wishlist.
    - 10.10. 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.

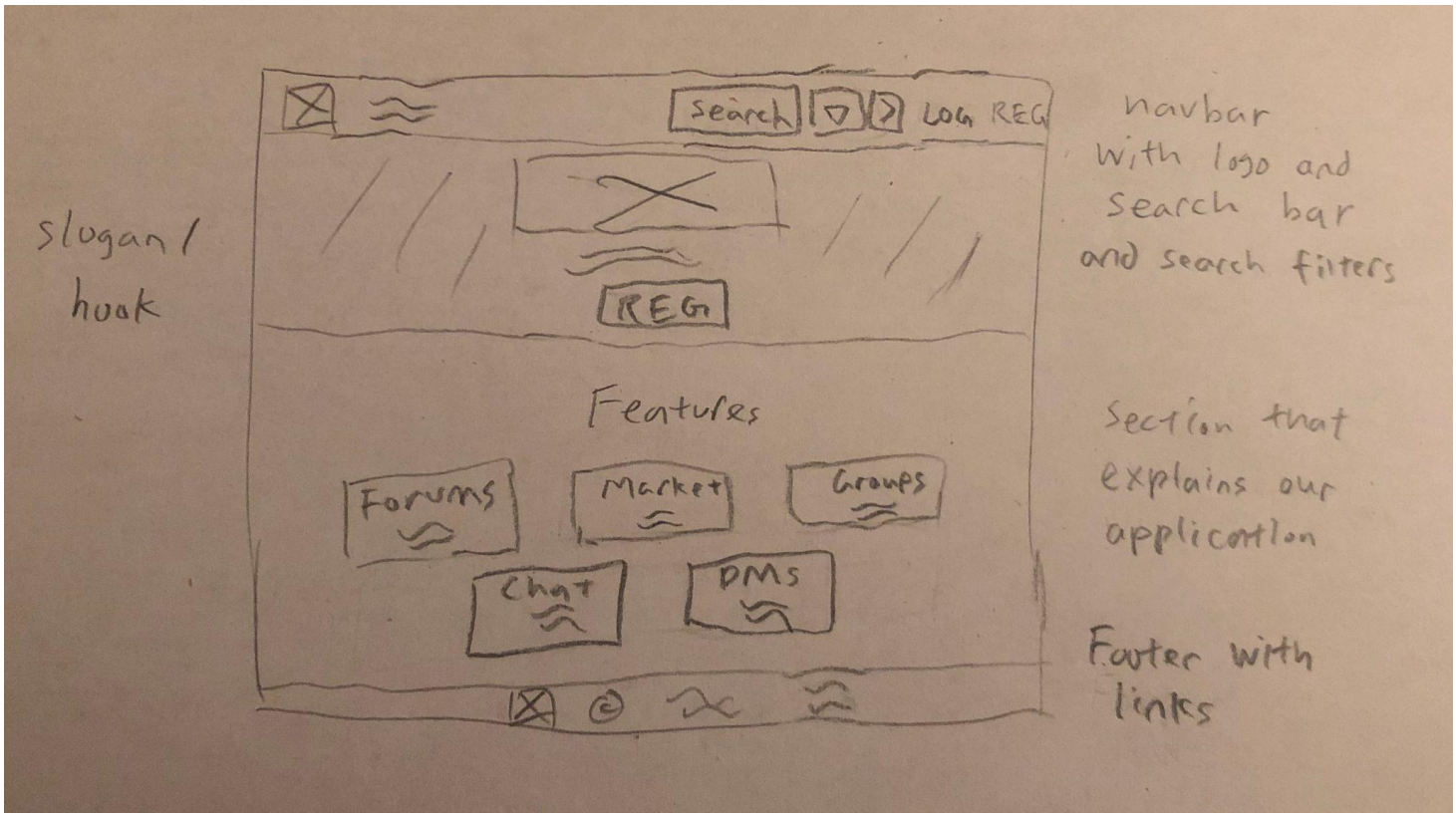
14. Group Moderator
    - 14.1. A group moderator shall be able to pin forum threads in their group's forum.

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.

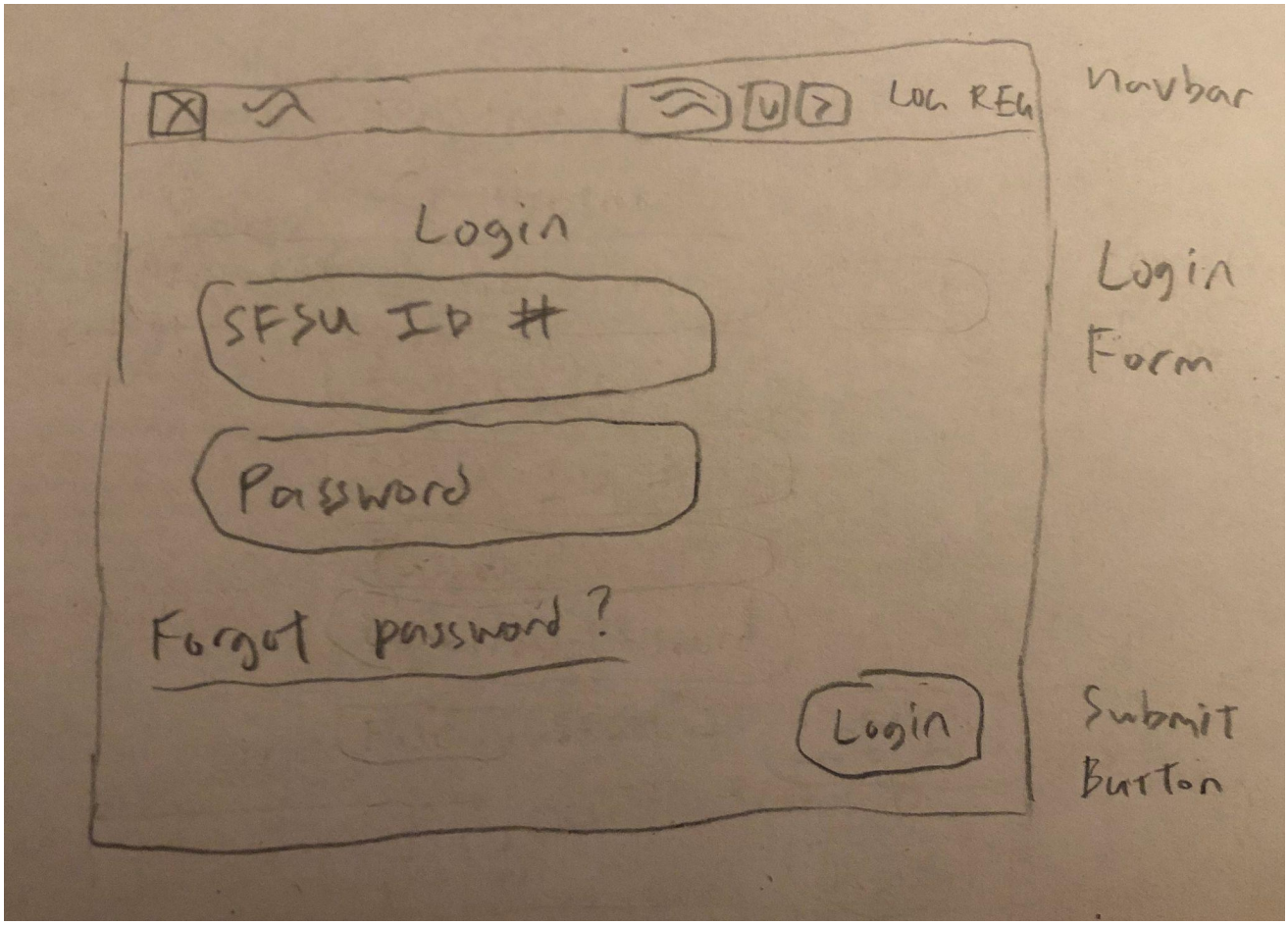
3. UI Mockups and Storyboards

UI Mockups of the Main Areas of the GUI:

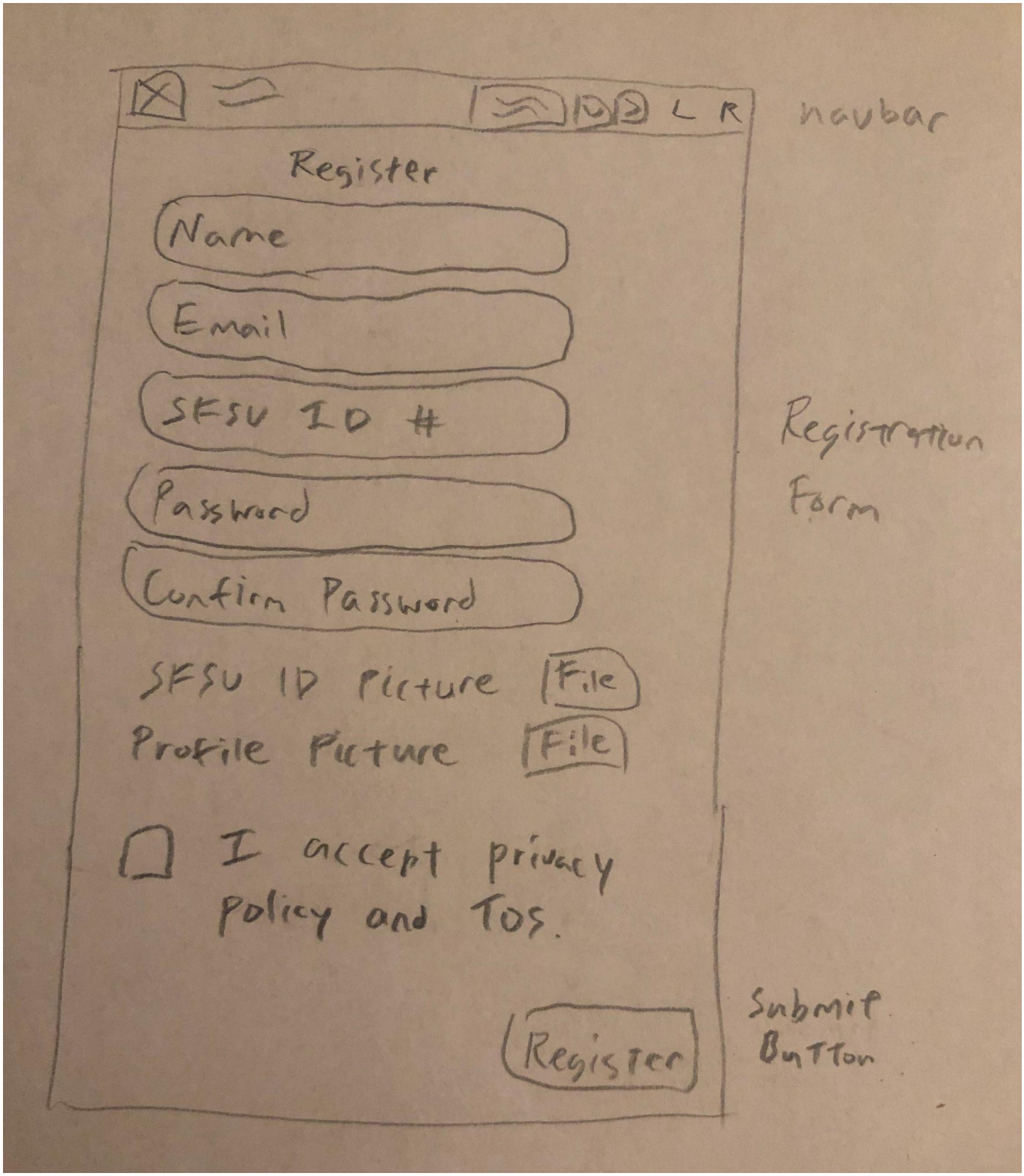
Home Page:



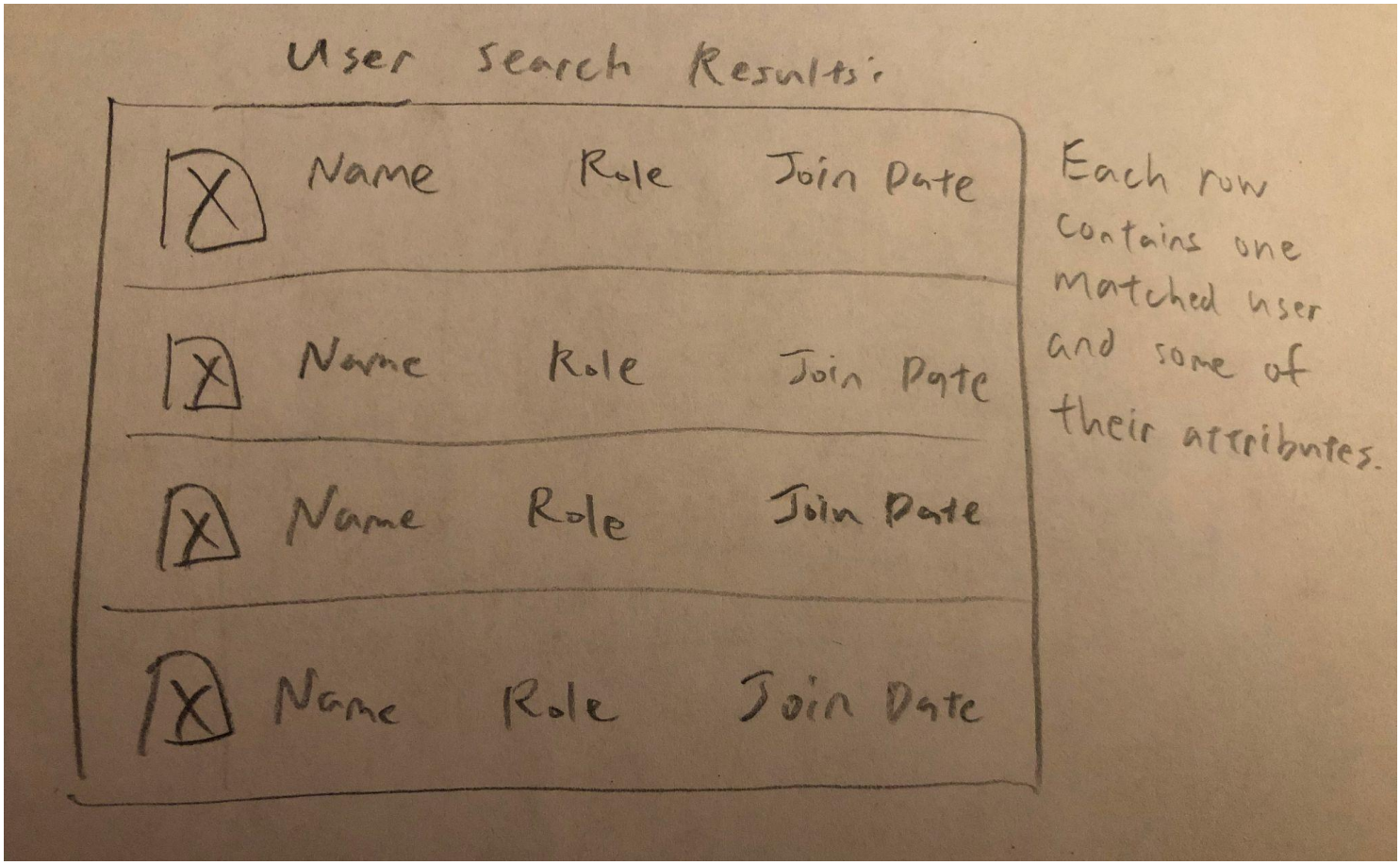
Login Page:


Log REG
navbar
Login
SFSU ID #
Password
Login Form
Forgot password?
Login
Submit Button

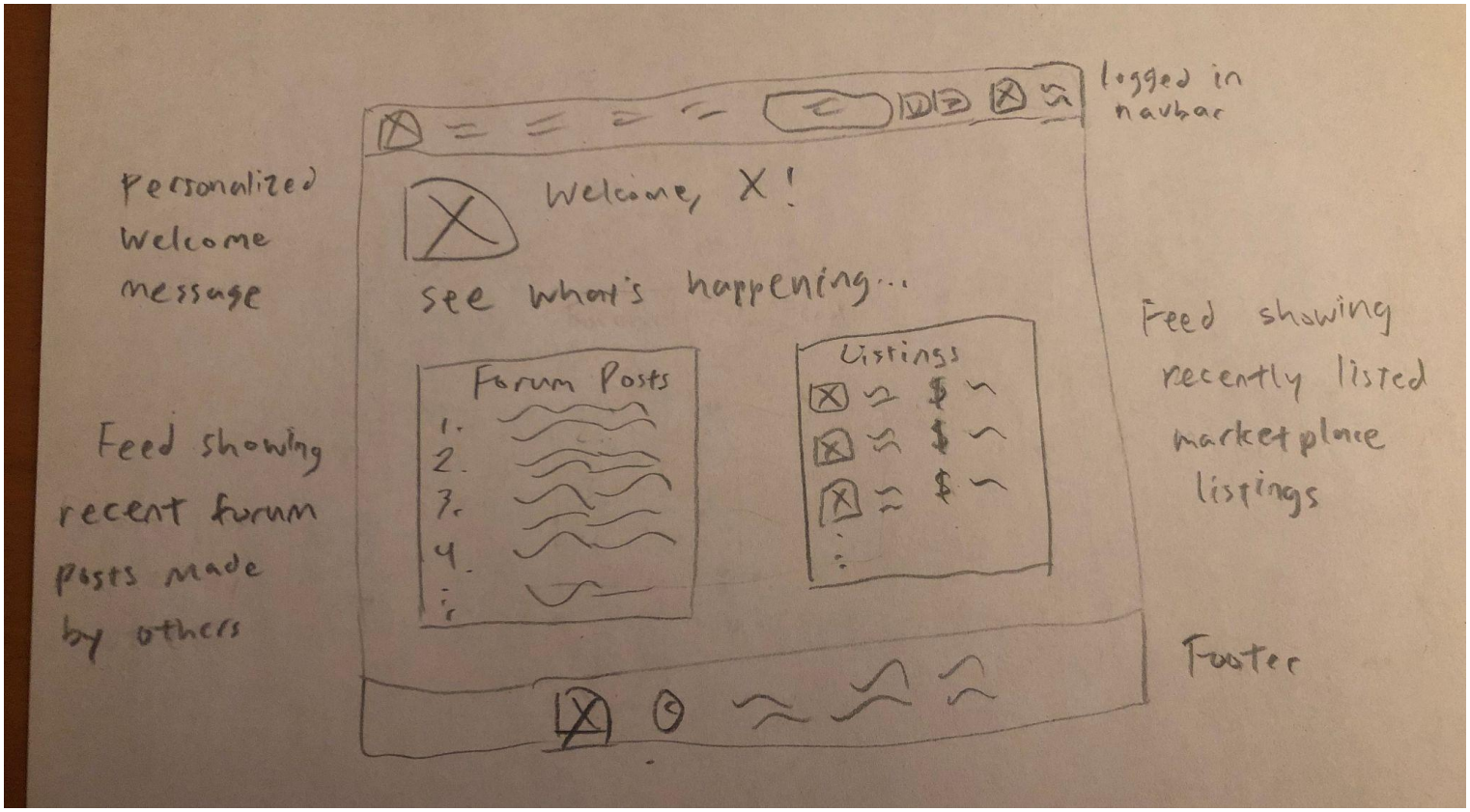
Registration Page:



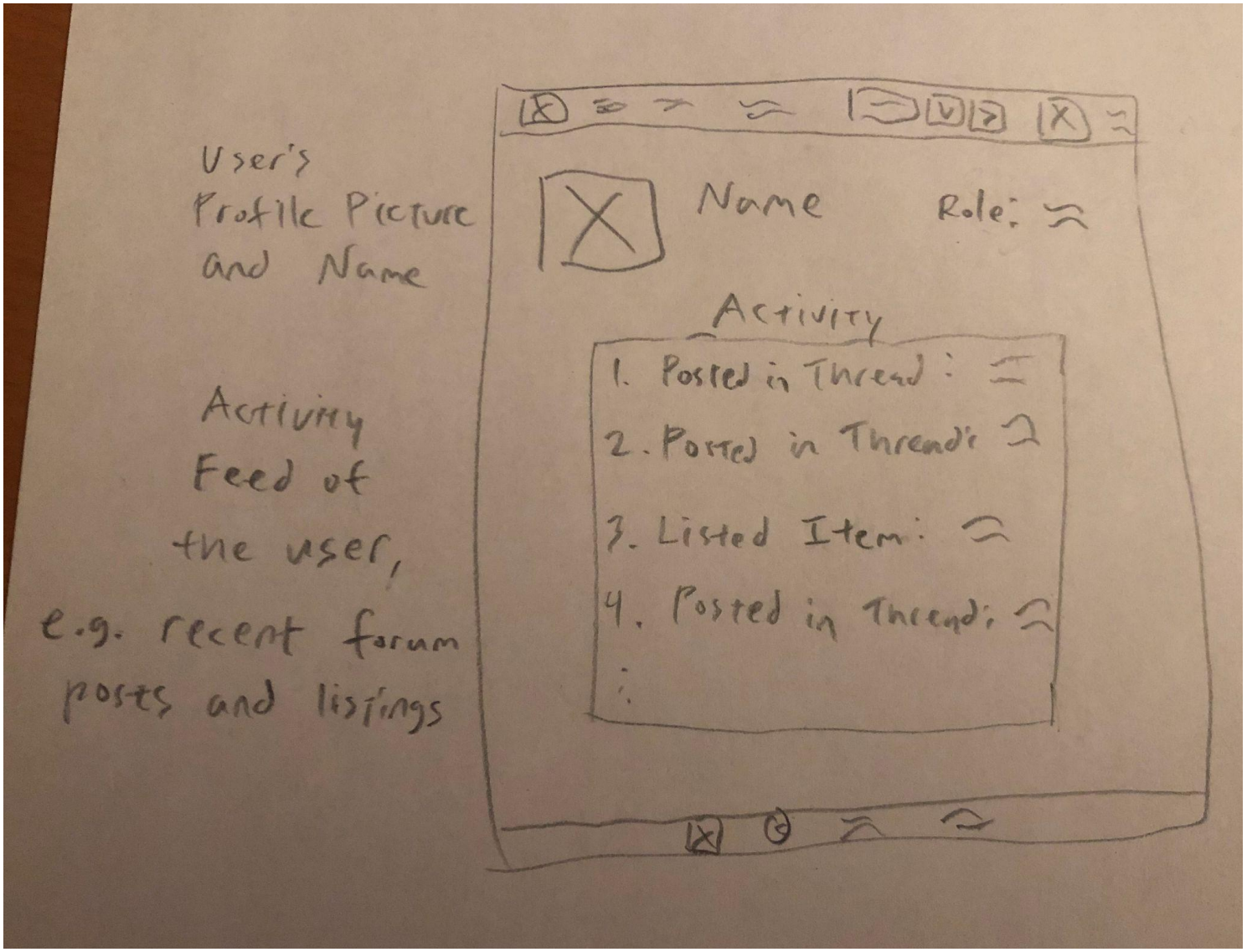
User Search Results:



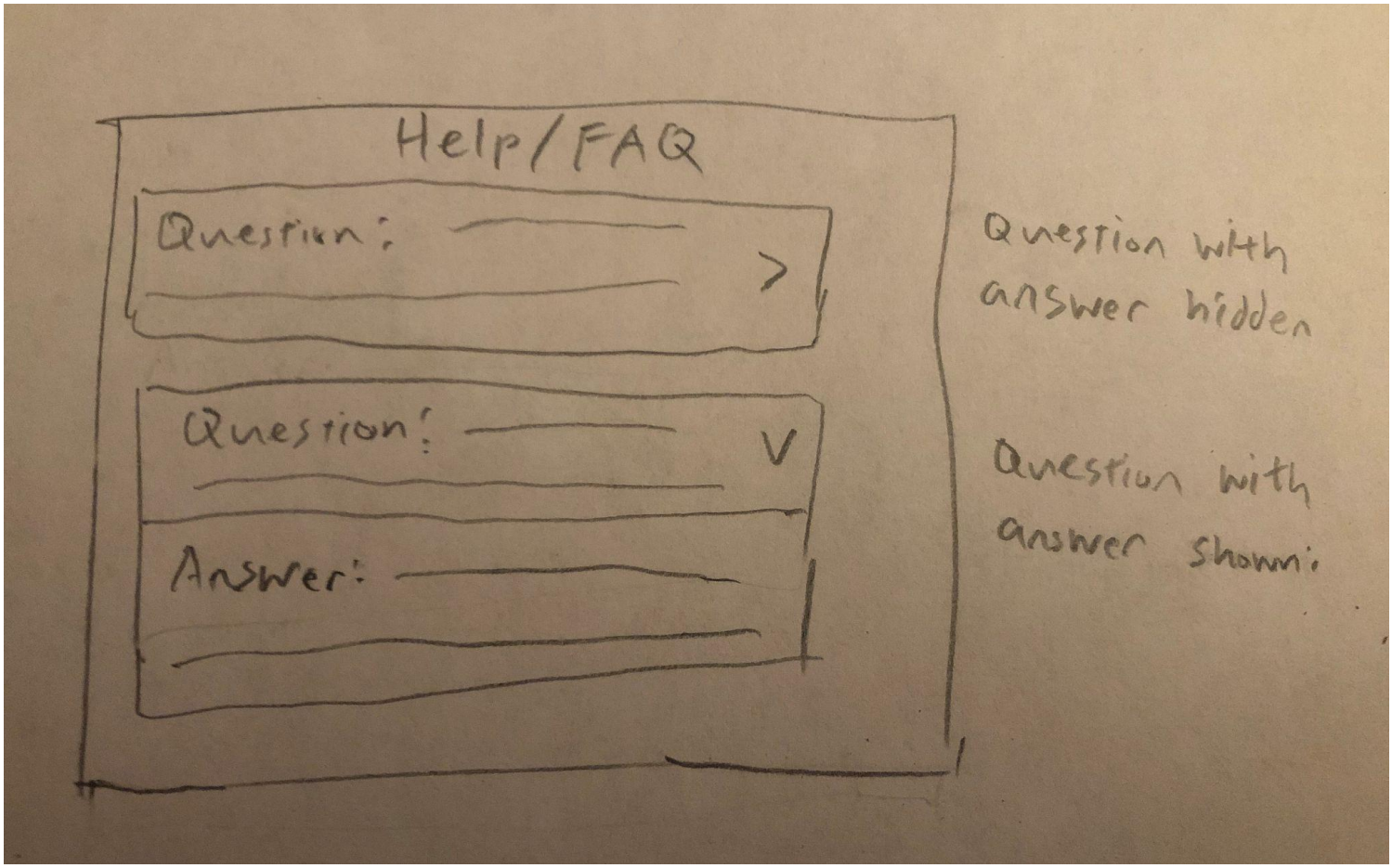
Dashboard Page:



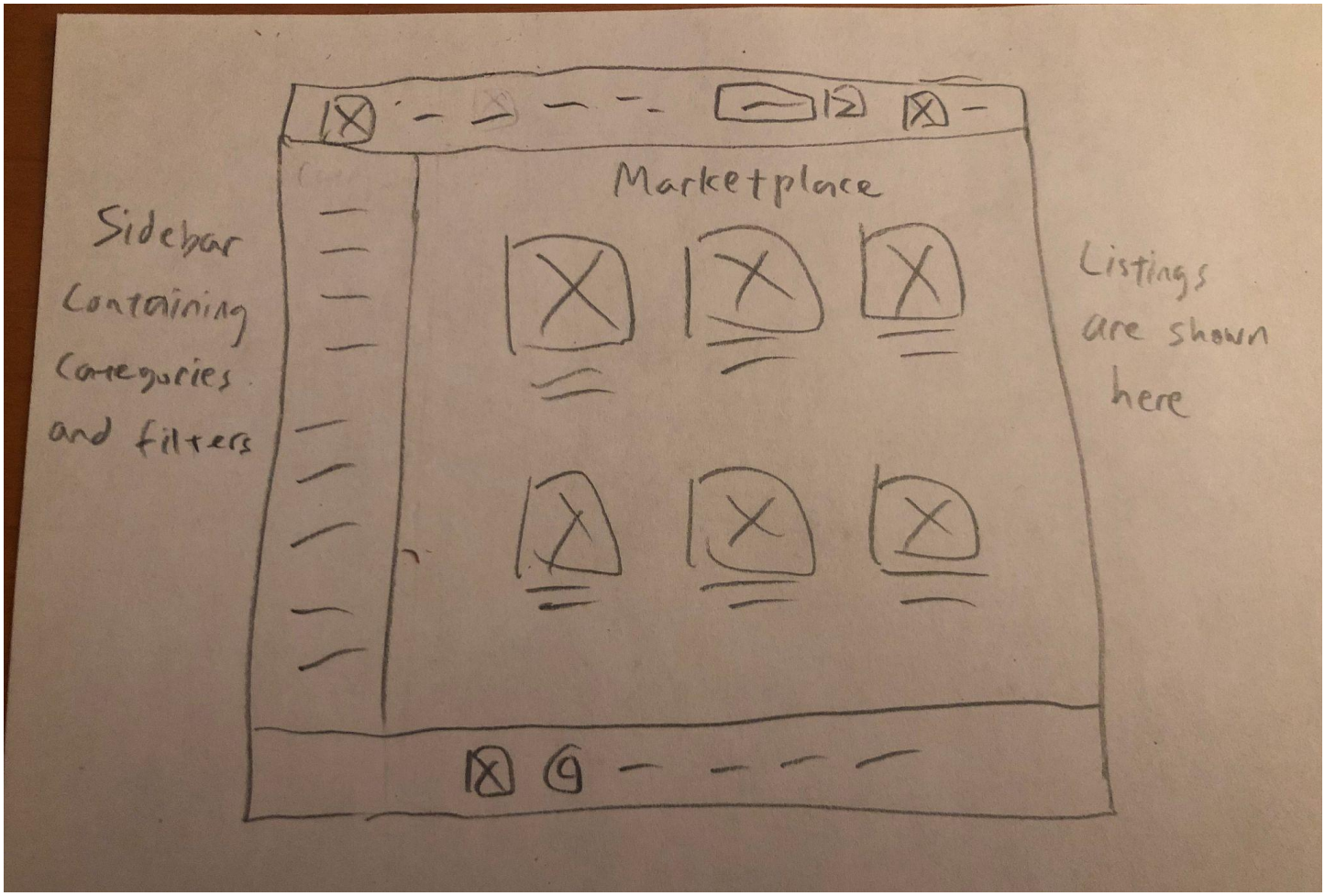
User Profile Page:



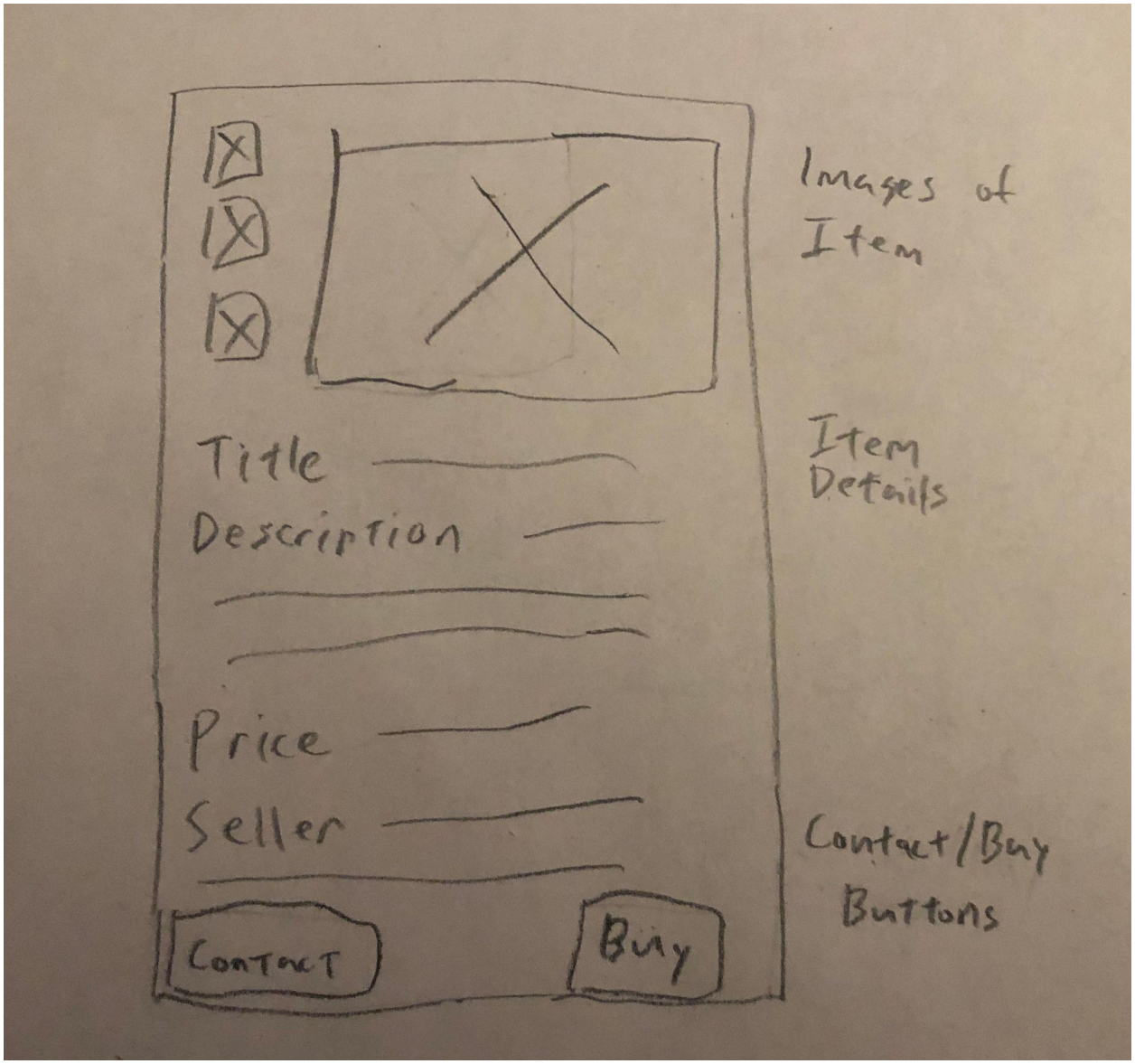
Help/FAQ Page:



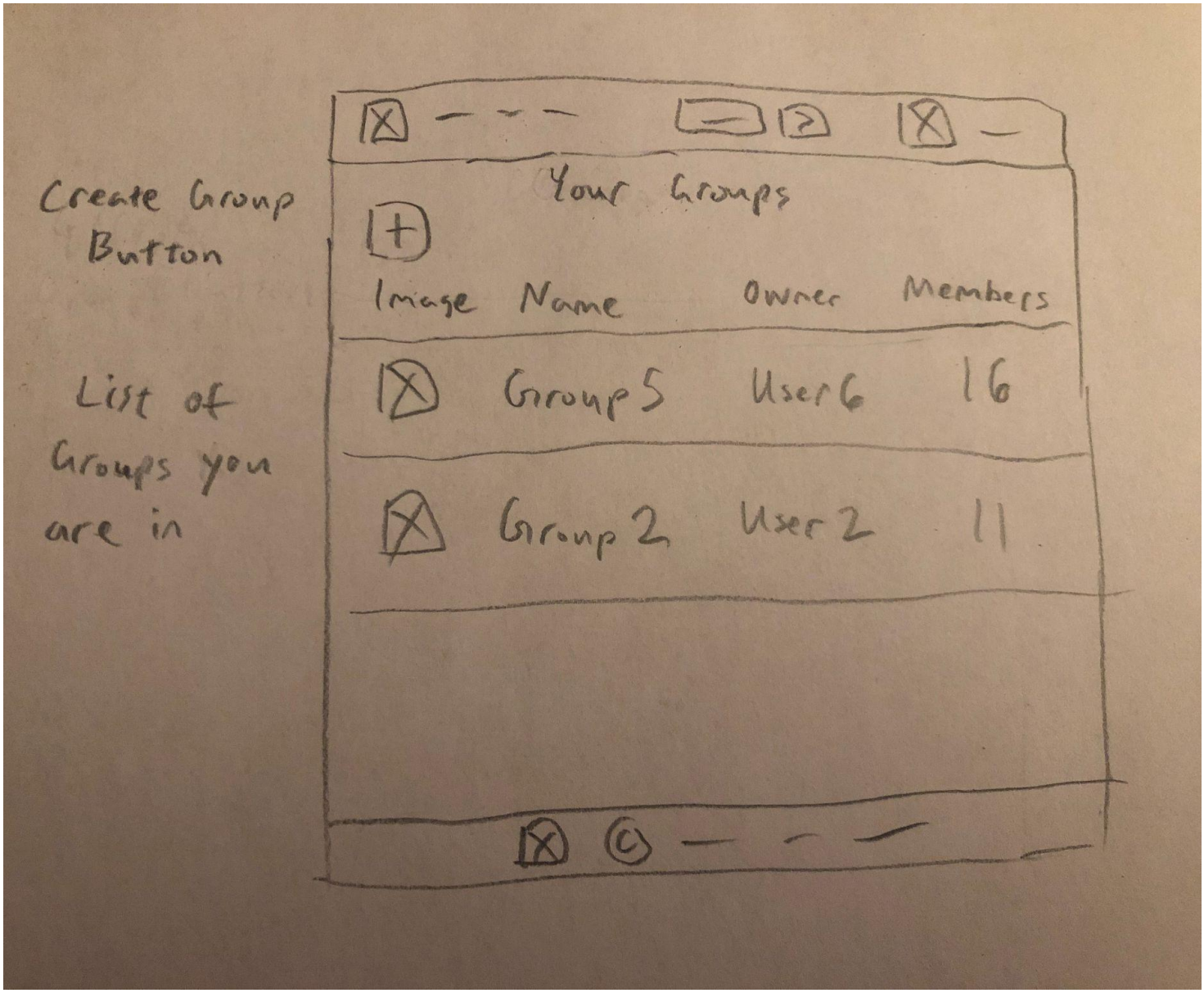
Marketplace Page:



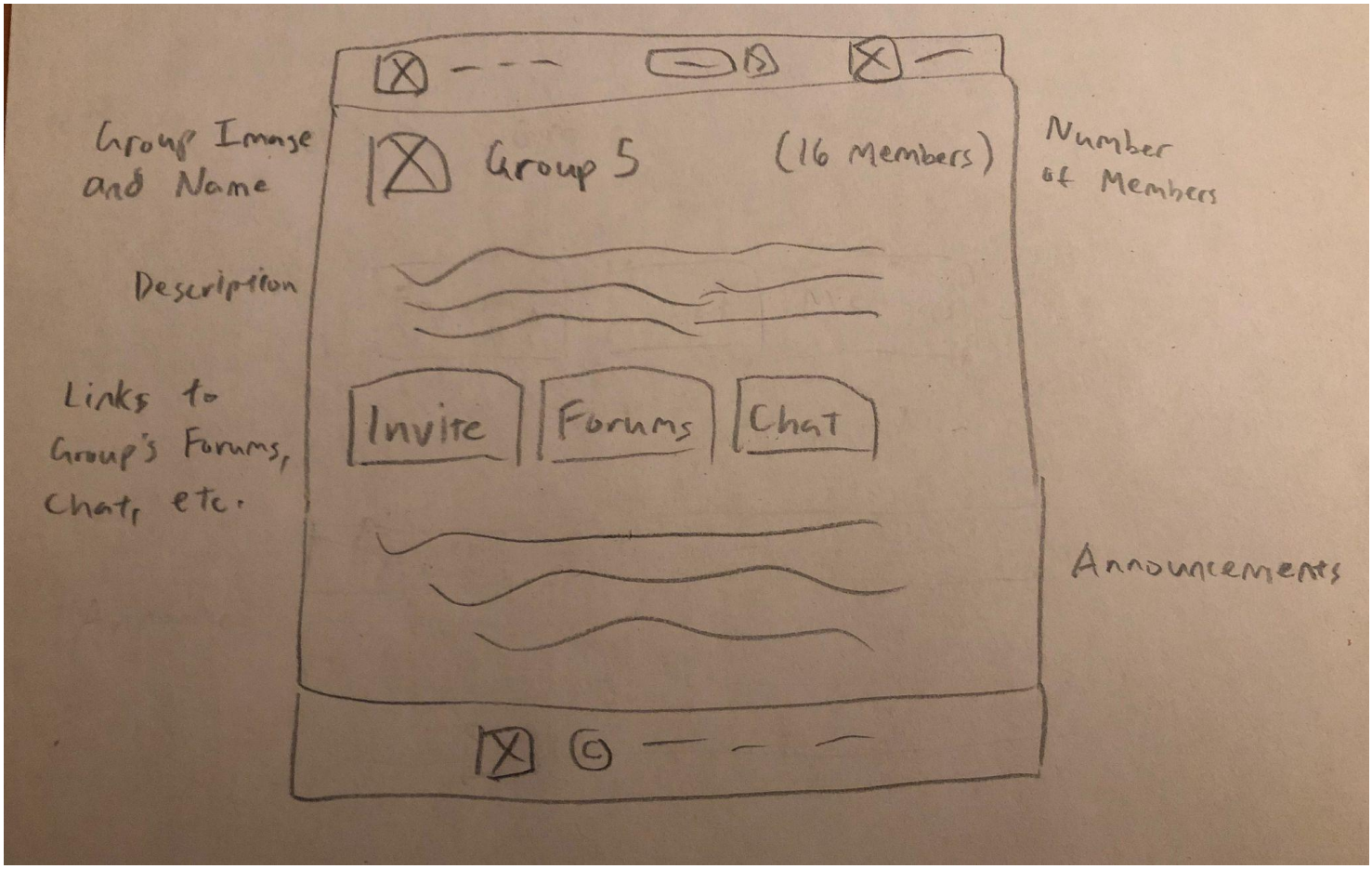
Marketplace Listing:



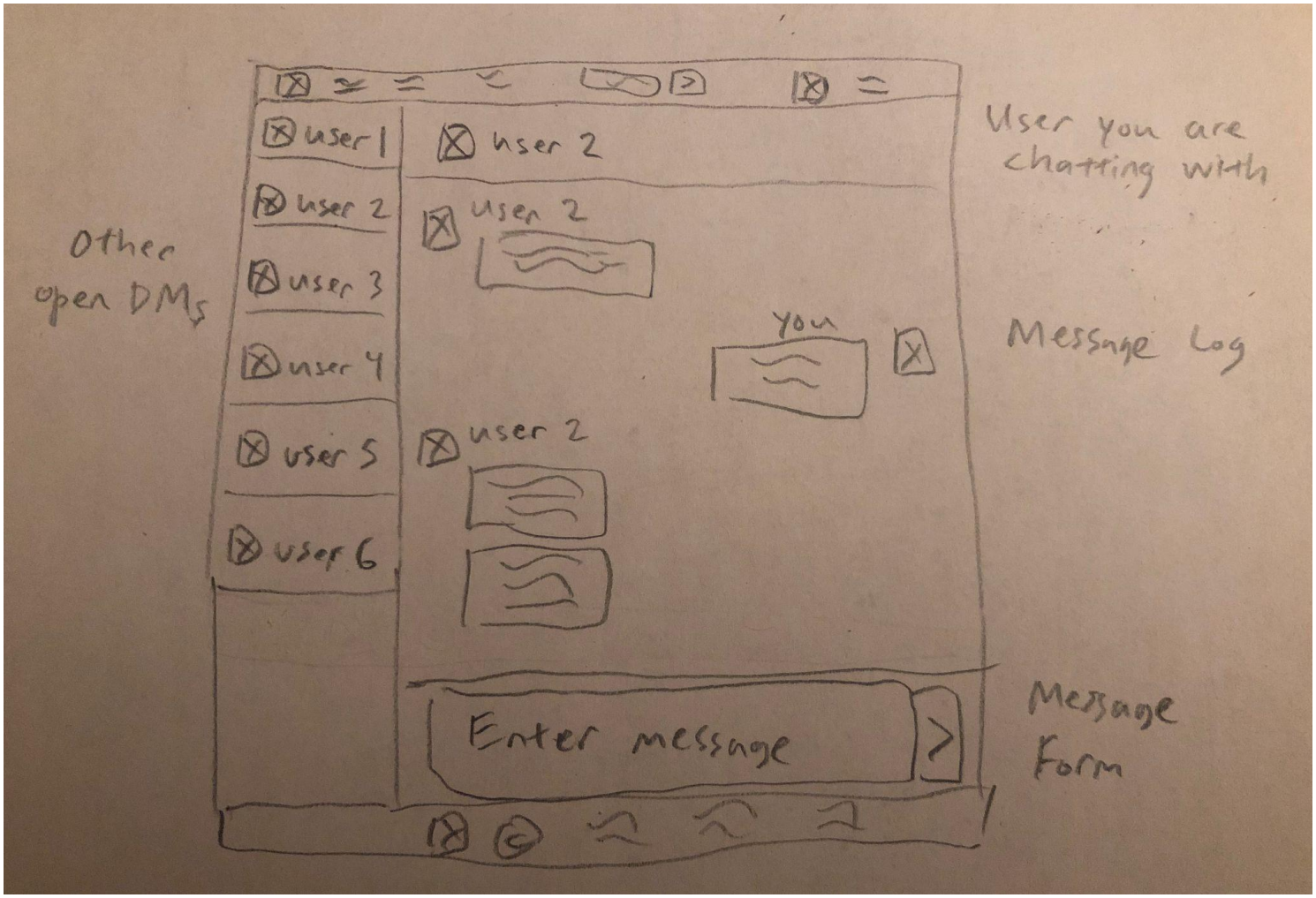
User's Groups:



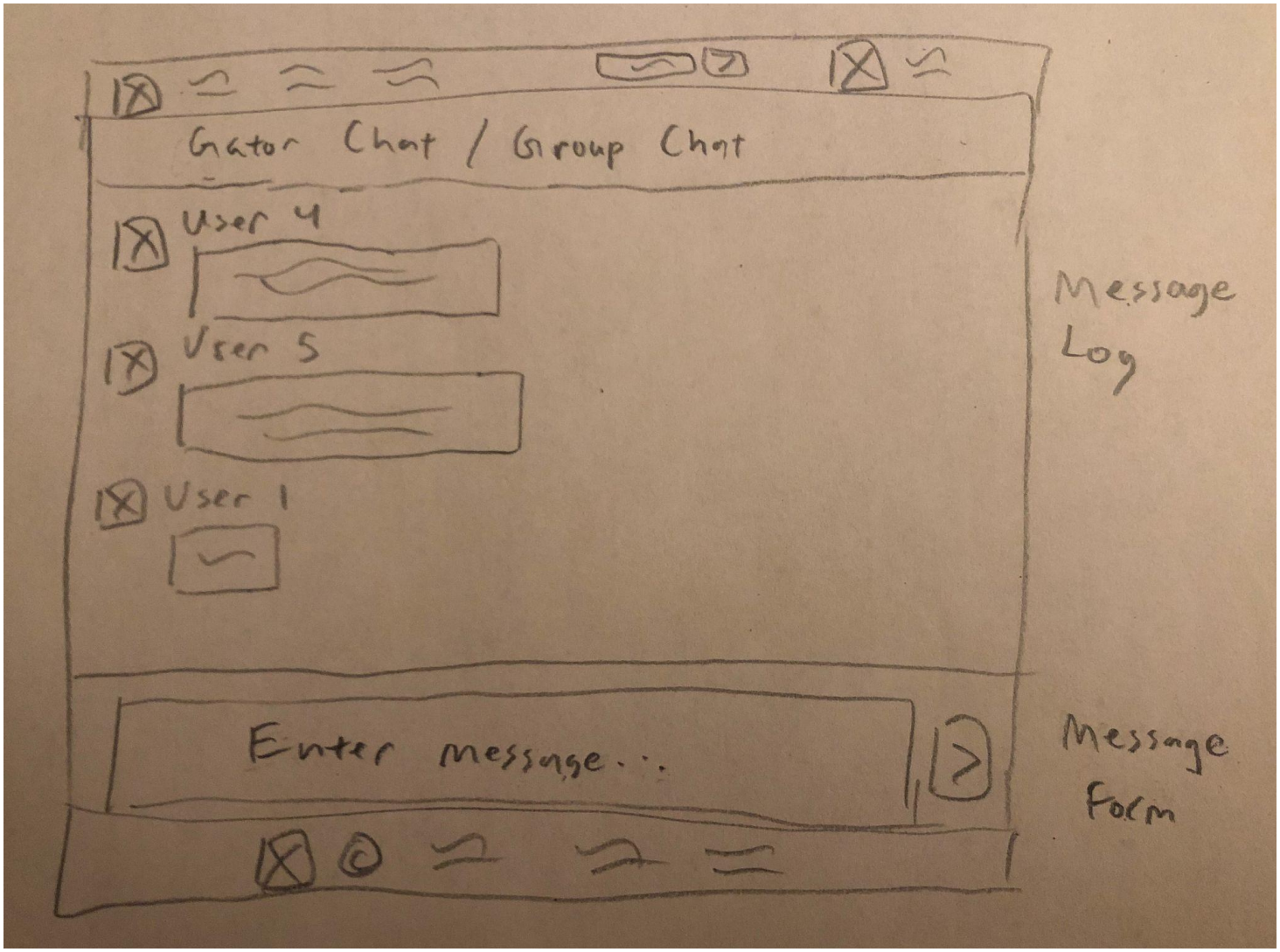
Group Home Page:



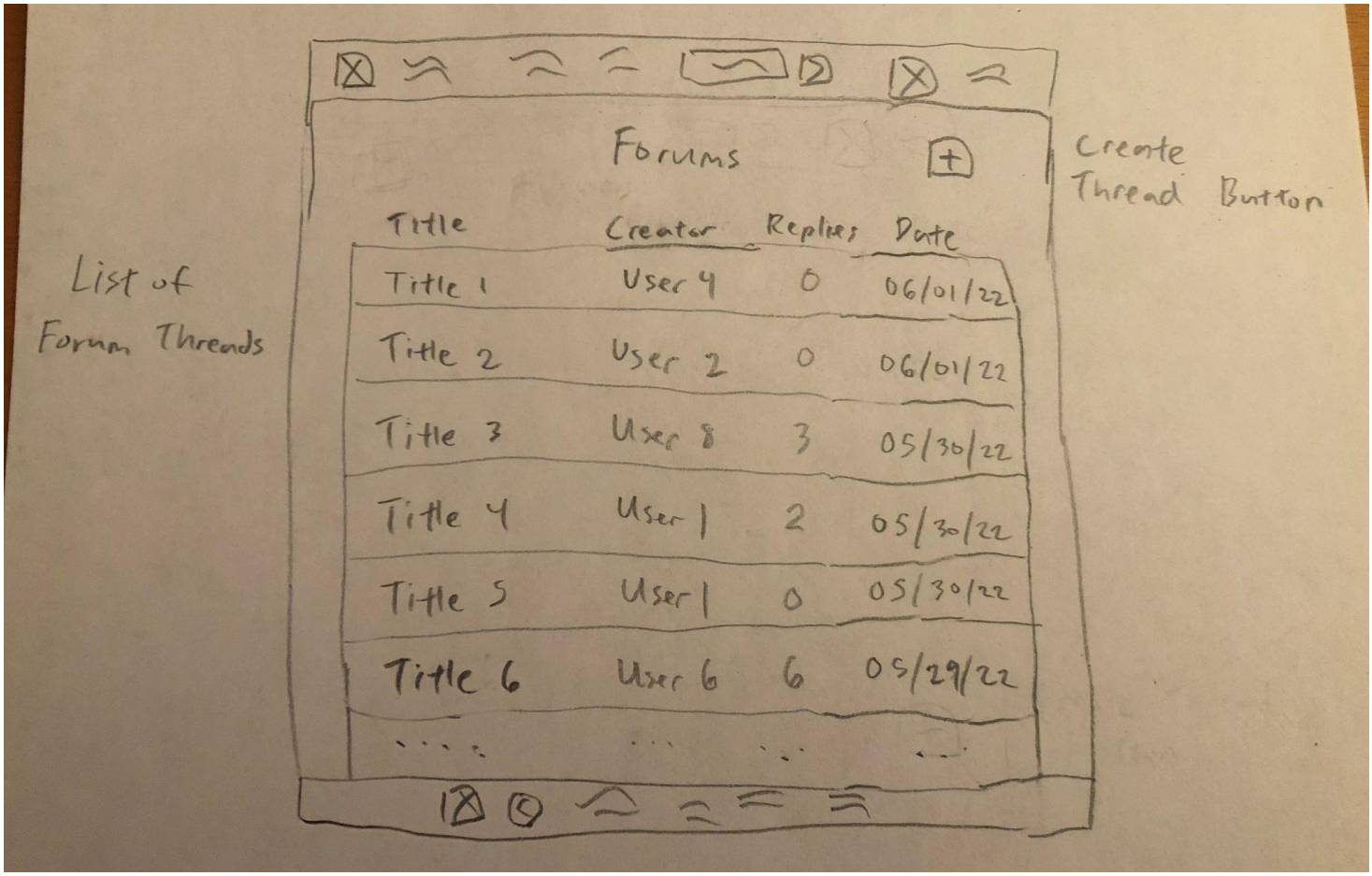
Direct Message (DM):



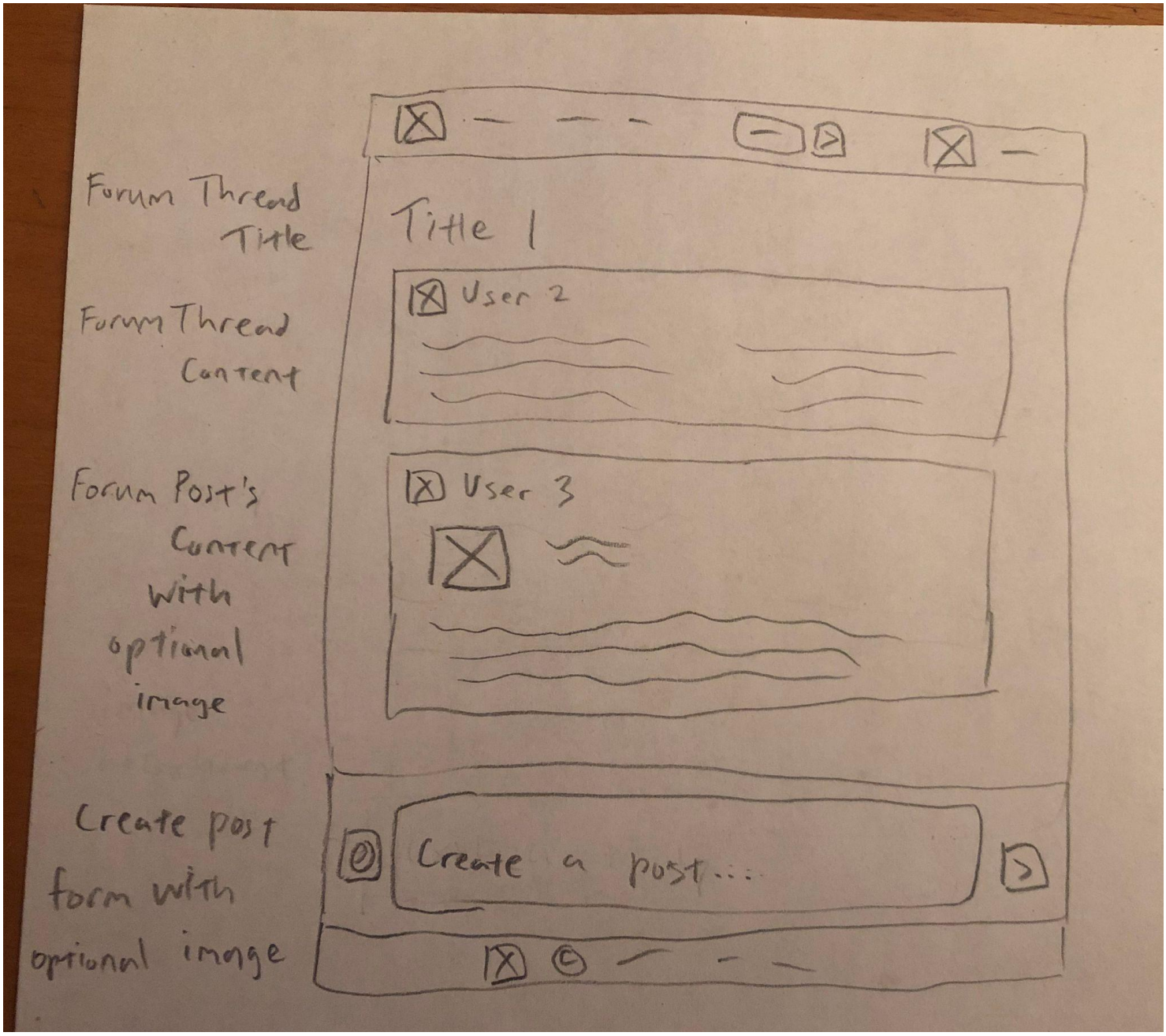
Gator/Group Chat:



Gatormmunity/Group Forum:

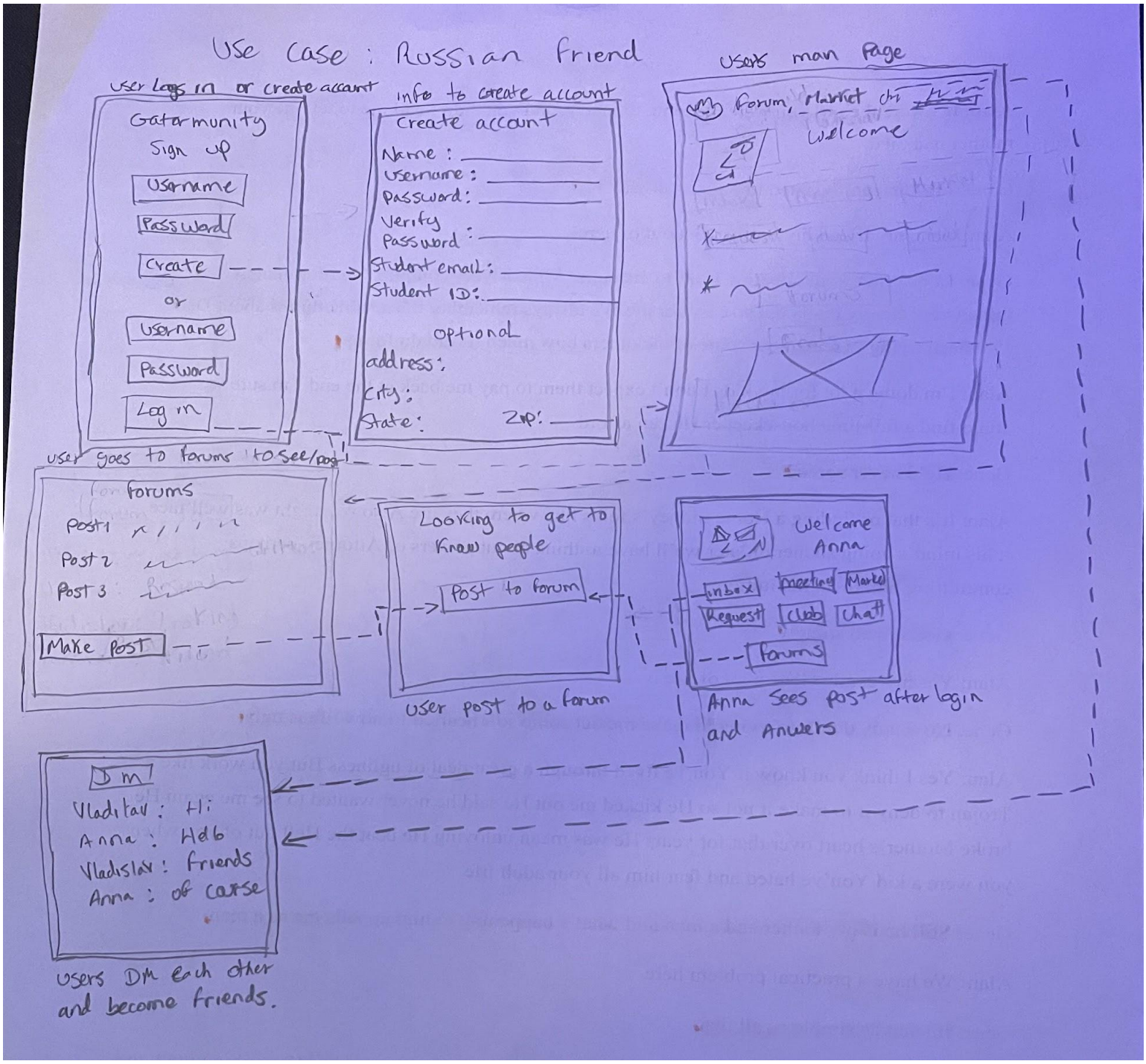


Viewing a Gatormmunity/Group Forum Thread:



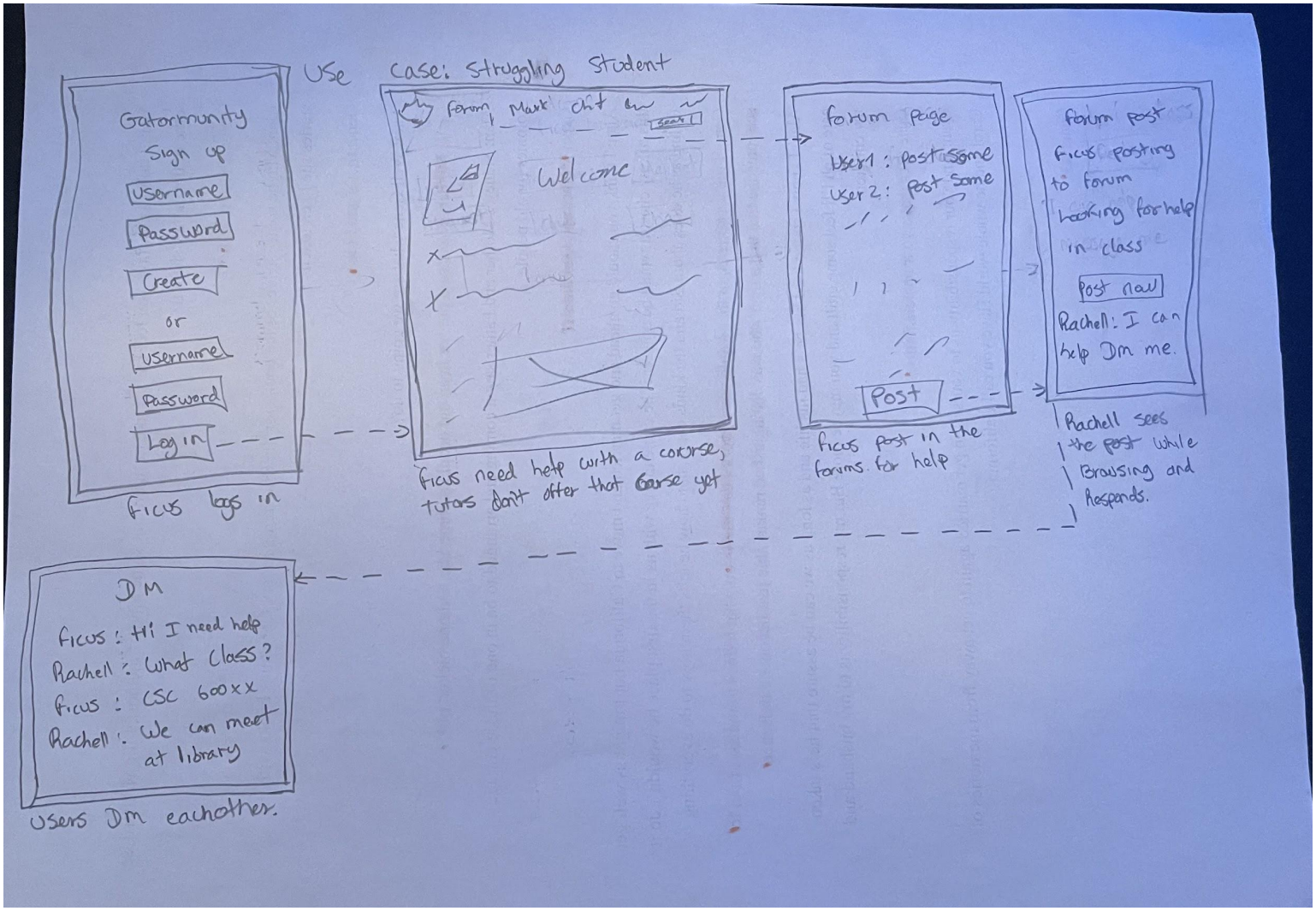
Storyboards for Use Cases:

Use Case #1: Russian Friends
Vladislav is trying to make friends in SFSU so he proceeds to make an account on Gatormmunity. After registering, he finds the forums and makes a thread stating that he is looking for friends. Anna sees his thread and initiates contact in order to befriend him.

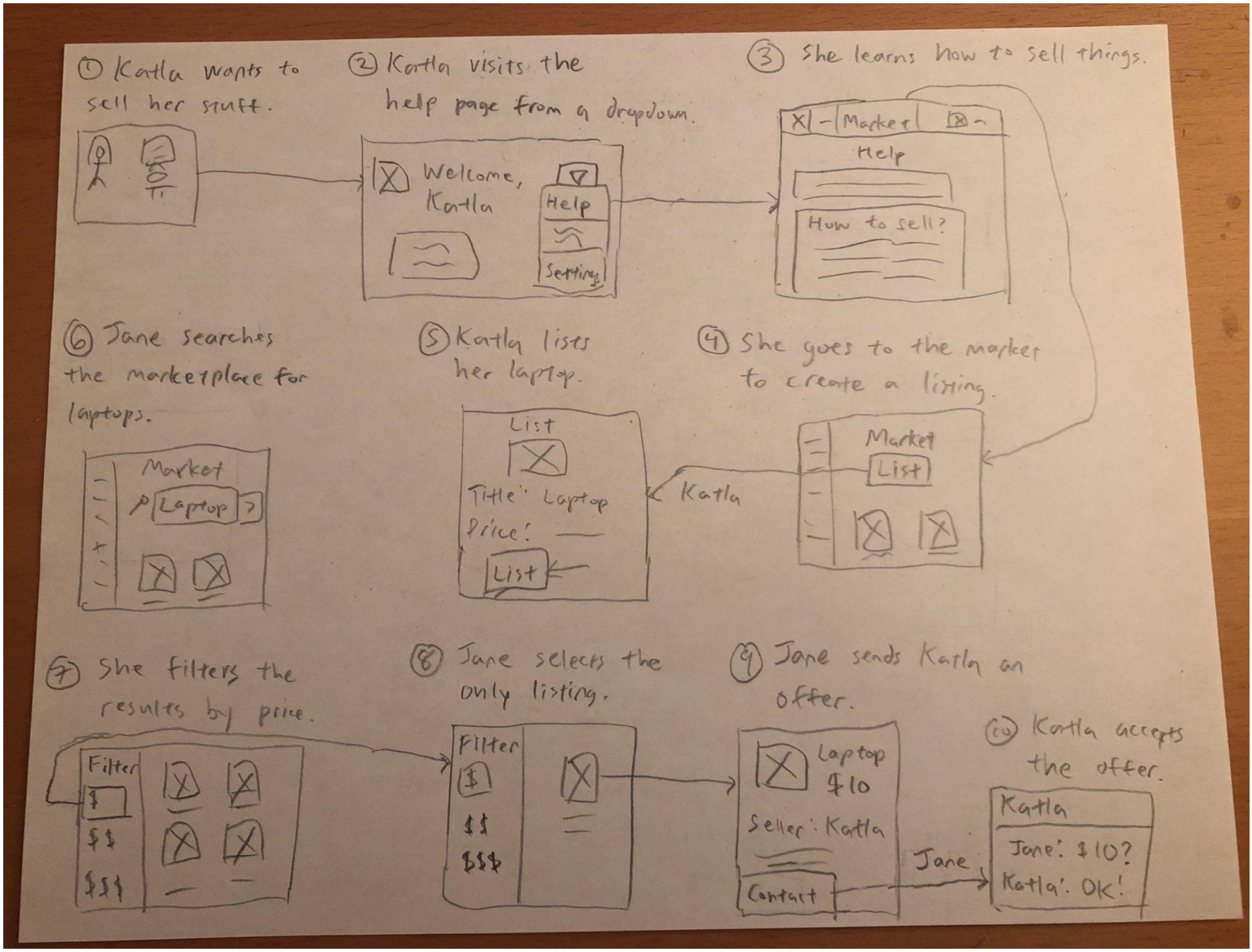


Use Case #2: Struggling Student

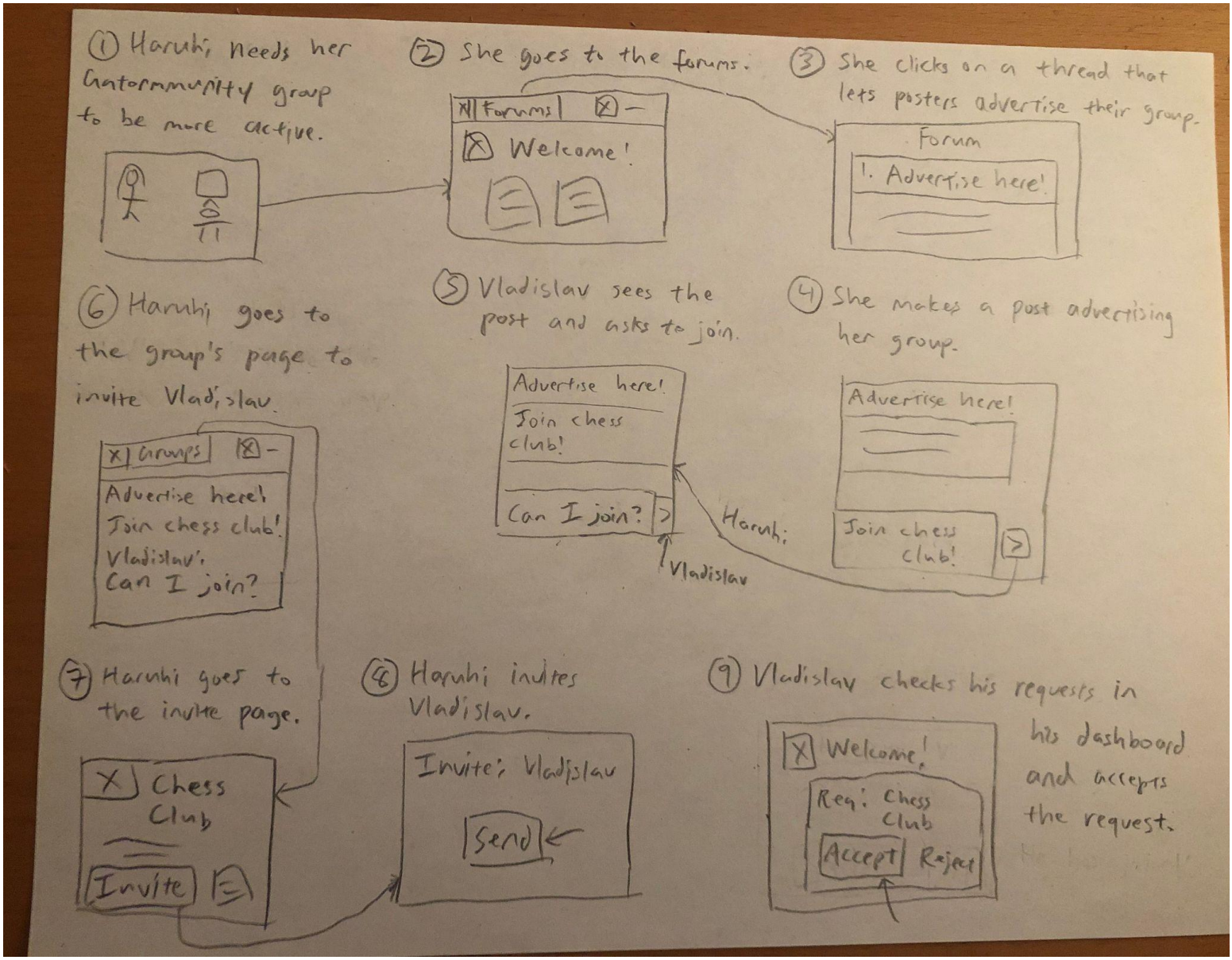
Ficus is looking for someone to help him with his coursework, therefore he asks in the Gatormmunity forums for help. Rachell sees his thread and provides him with help after contacting him through a direct message.

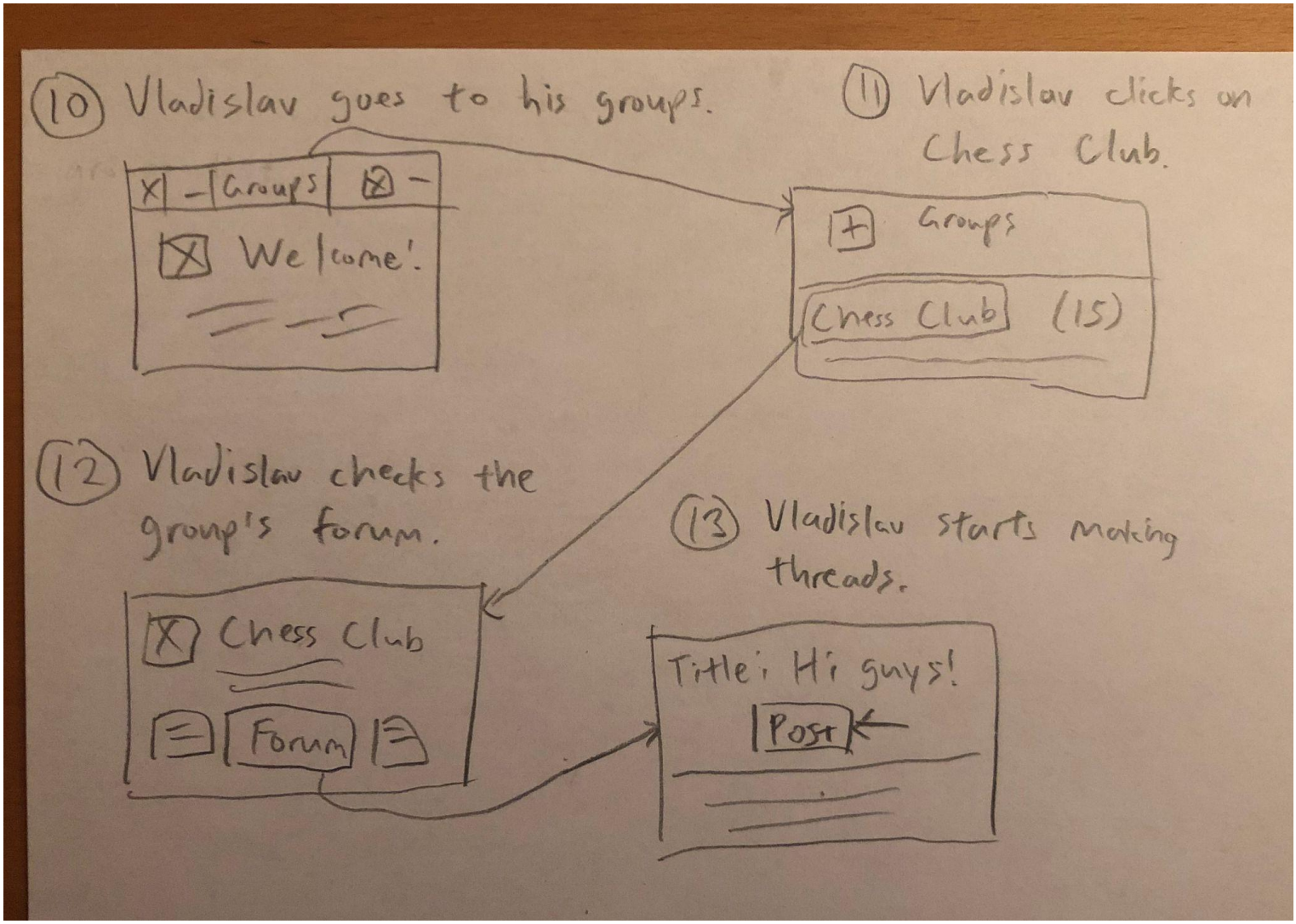


Use Case #3: Wary Traders
Katla is looking to sell her stuff and so she uses Gatormmunity. She was not sure how to sell things at first, until she looked at the help page. She was then able to list her laptop for sale. Coincidentally, Jane is looking for a cheap laptop on Gatormmunity and finds Katla's.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

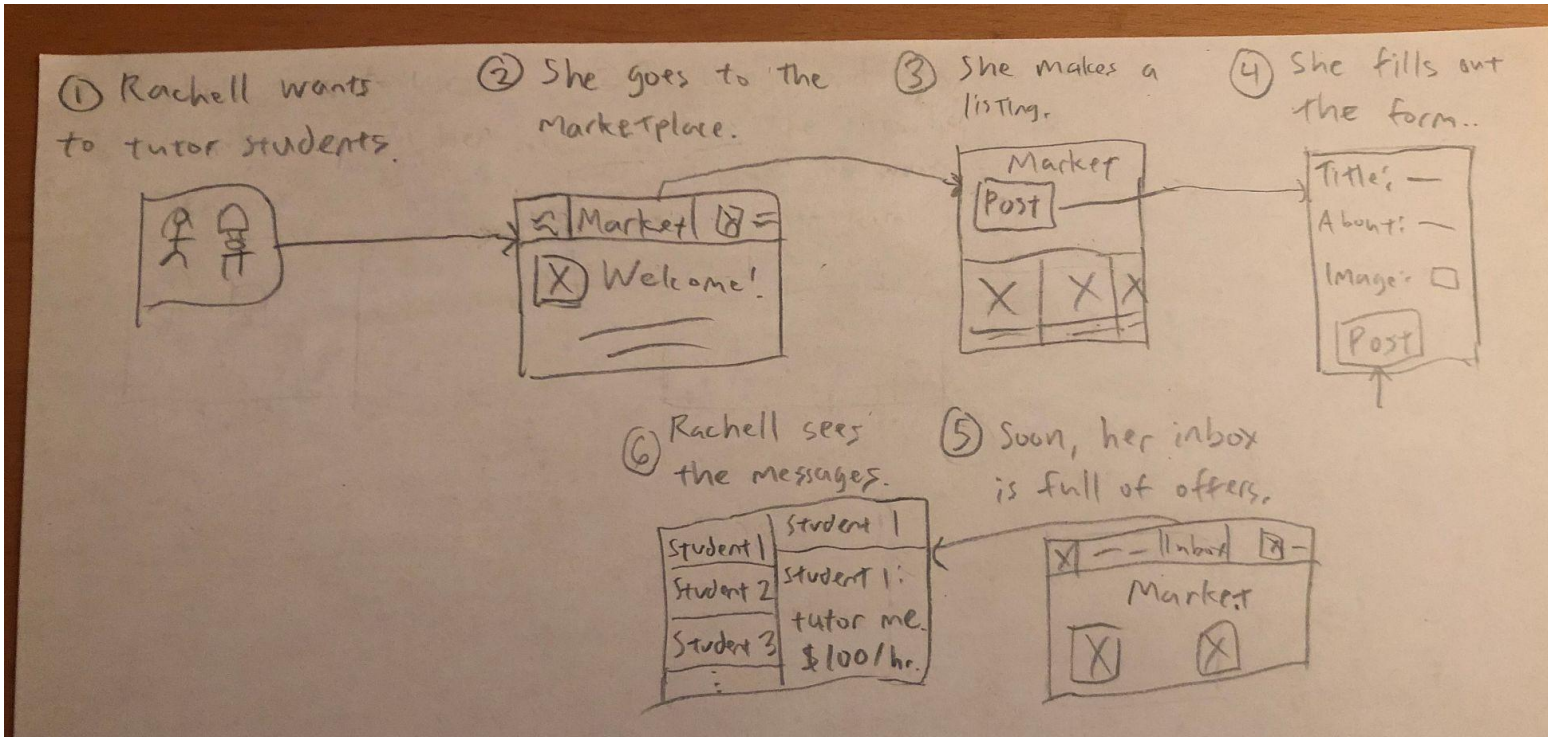


Use Case #4: Inactive Chess Club (has 2 storyboard pages)
Haruhi wants to make her Gatormmunity group more active, thus she posts in the Gatormmunity forums hoping to attract some new members. Vladislav sees the advertisement and asks to join. Haruhi invites him, and after he accepts, he starts to post in the group's forum.

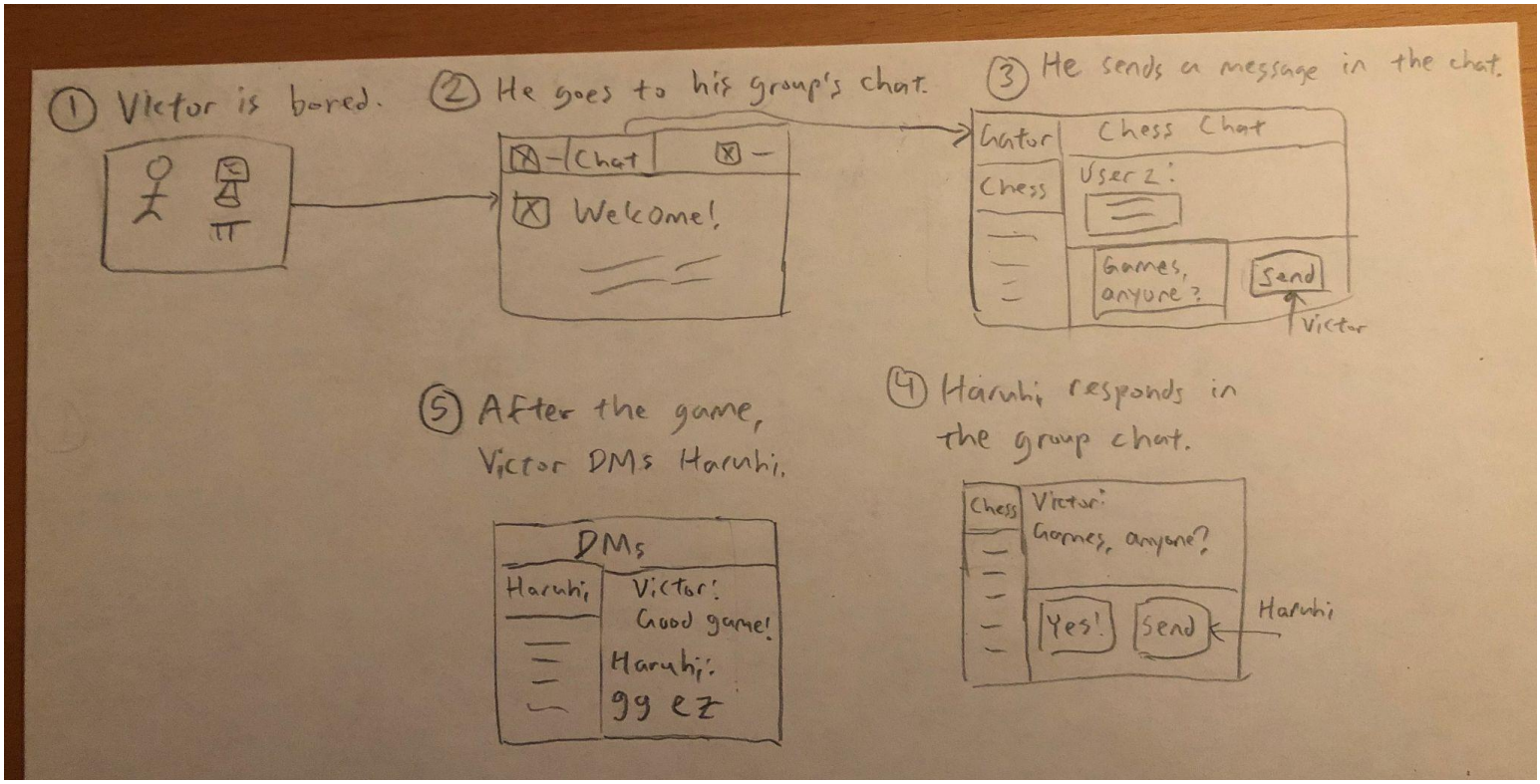



(10) Vladislav goes to his groups.
(11) Vladislav clicks on Chess Club.
Groups
Welcome!
Groups
Chess Club (15)
(12) Vladislav checks the group's forum.
(13) Vladislav starts making threads.
Chess Club
Forum
Title: Hi guys!
Post

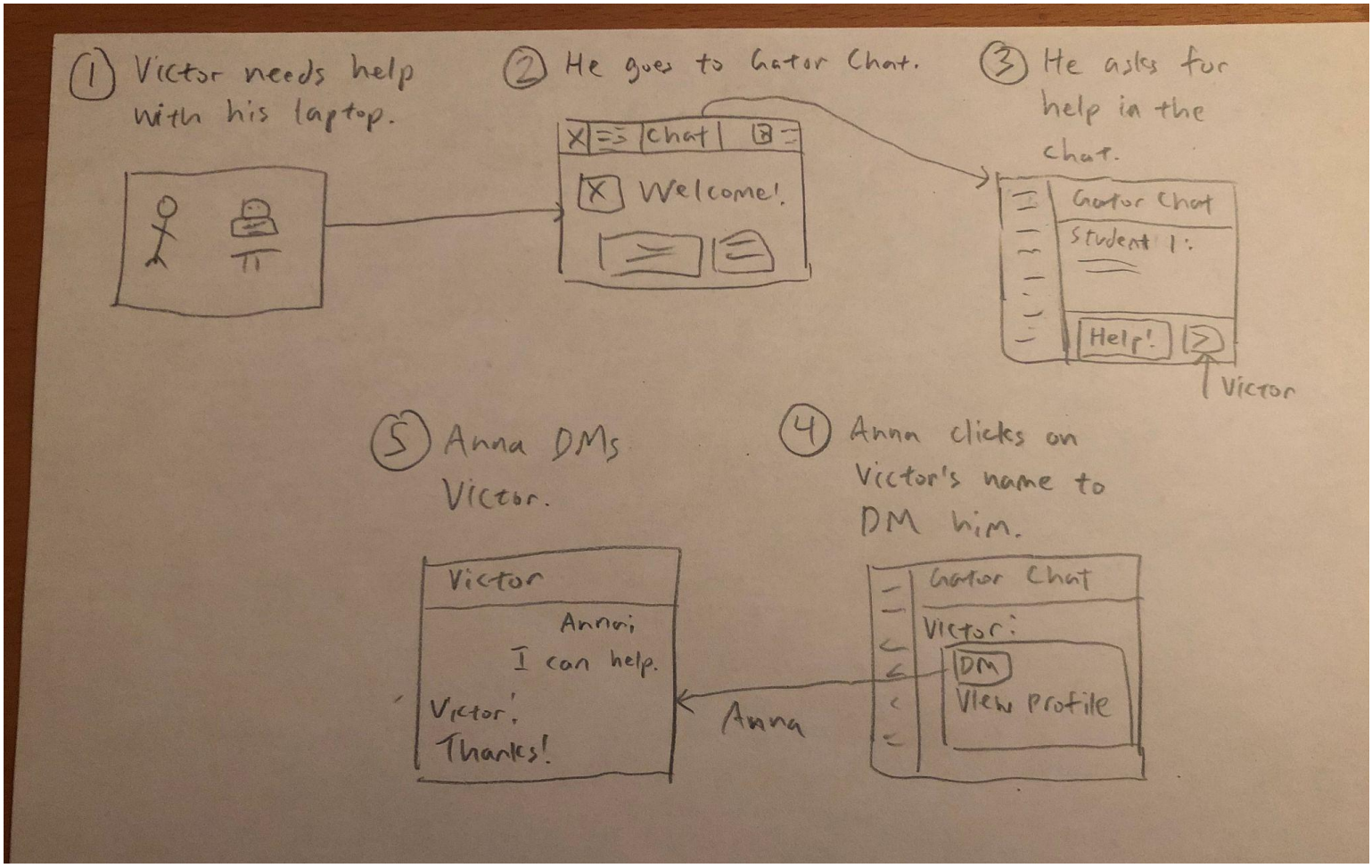
Use Case #5: Tired of Spam
Rachell is trying to find students to tutor on Gatormmunity. She makes a listing in the marketplace offering her services and she is quickly inundated with messages from students who wish to pay for her services.



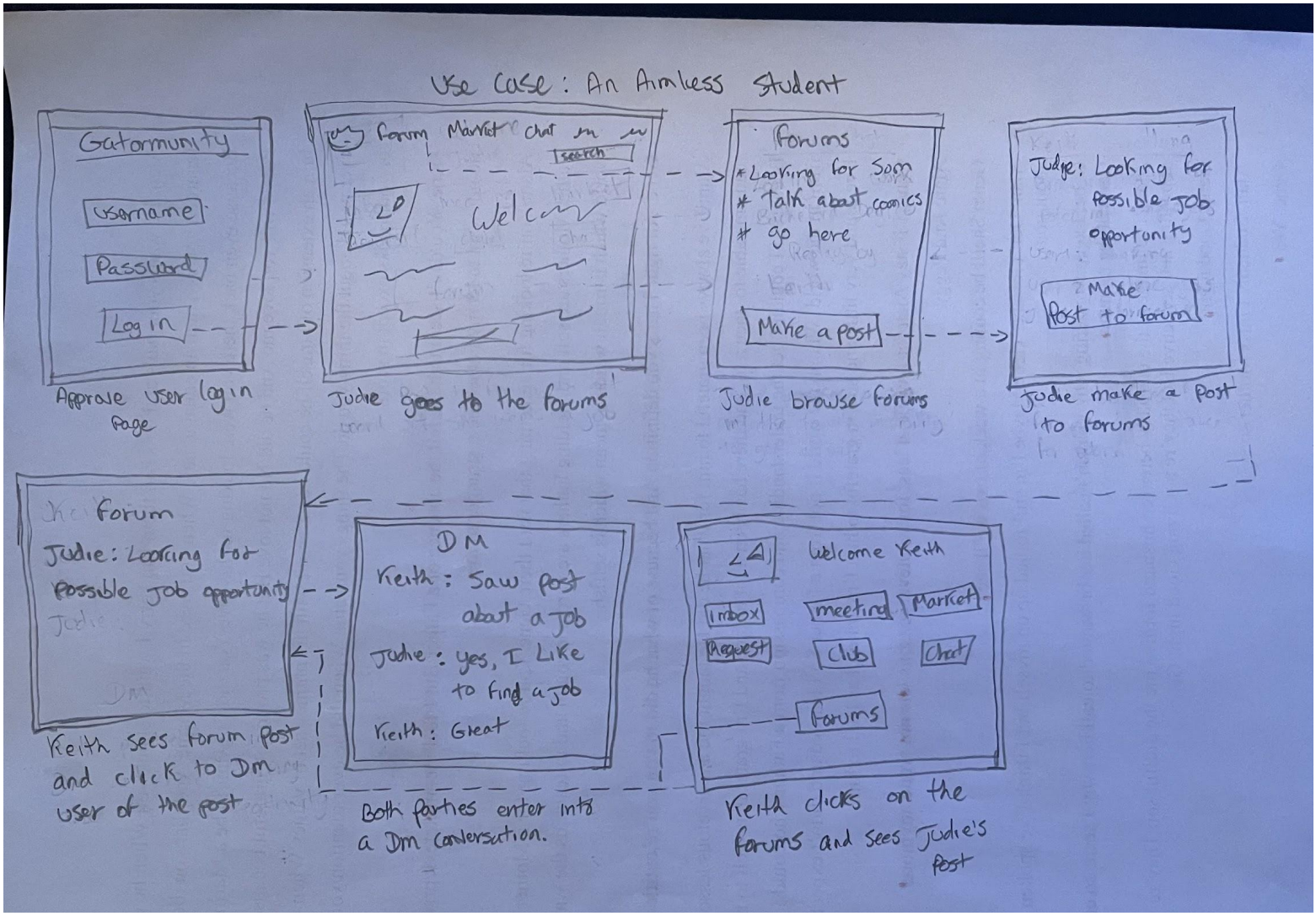
Use Case #6: A Game of Chess
Victor is bored and decides to ask in the group chat if anyone wants to play some chess. Haruhi accepts the challenge. After the game, the two talk about the game over direct messages.



Use Case #7: Powerpoint Problems
Victor is having problems with his laptop, so he tries to ask in Gator Chat if anyone can help. Anna notices the request and direct messages Victor to let him know she can help.

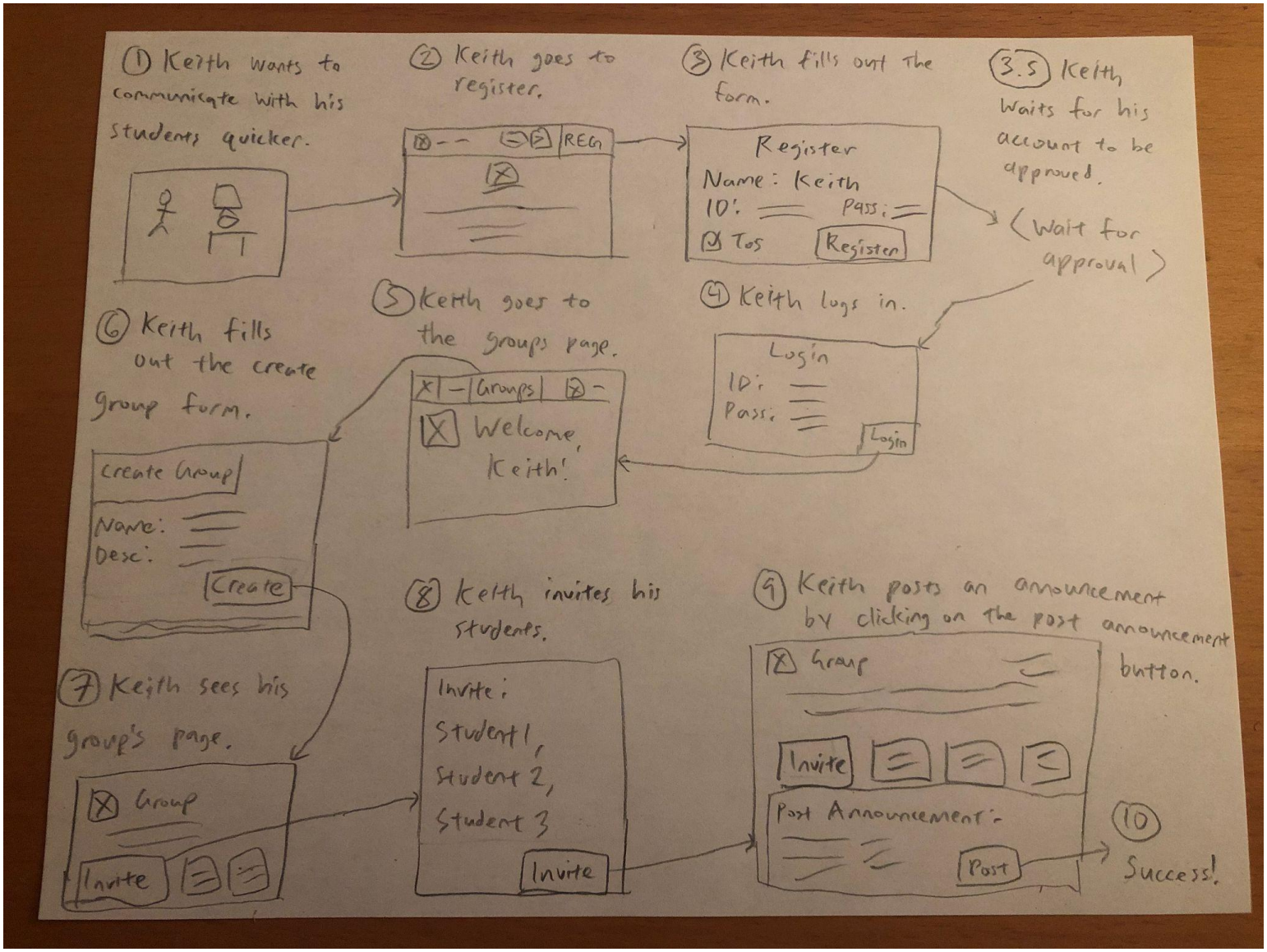


Use Case #8: An Aimless Student
Judie wants to find a job, therefore she creates a post in the forums looking for work. Keith sees the post and direct messages Judie about her post.

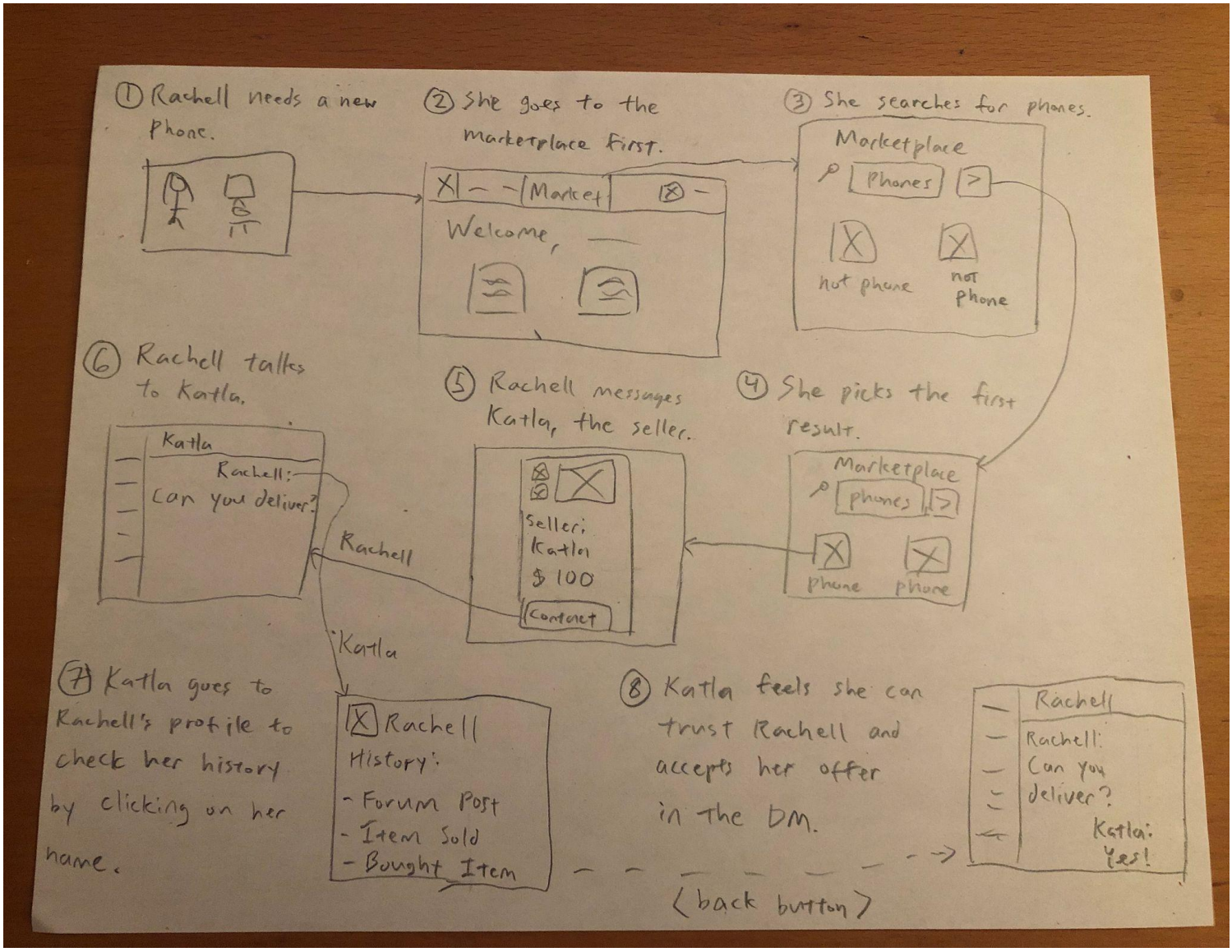


Use Case #9: A Professor's Pains

Keith heard Gatormmunity can meet his desire to message all of his students at once, thus he registers for an account and logs in after being approved. He creates a group and invites his students to the group. Finally, Keith can post announcements to everyone at once now.



Use Case #10: Supporting the Students
Rachell looks for a new phone in the Gatormmunity Marketplace and finds Katla's listing. Rachell messages Katla asking if Katla can deliver. Katla first checks Rachell's history before deciding to trust her. After the background check, Katla agrees to the delivery.



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 many 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. creation_date: alphanumeric
    3.4. group_id: foreign key, numeric
    3.5. creator_id: foreign key, numeric

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

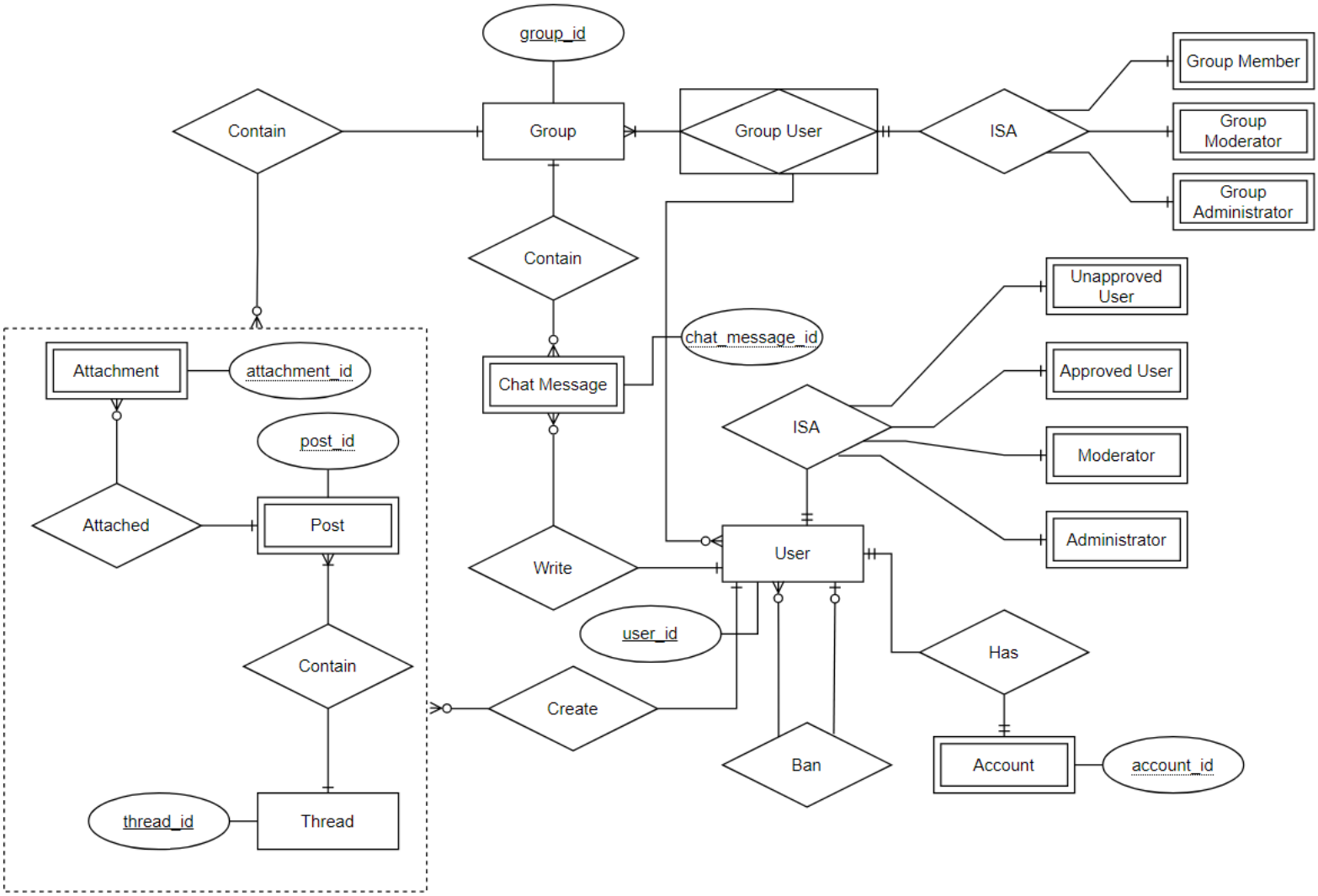
5. Attachment (Weak)
    5.1. attachment_id: primary key, numeric
    5.2. filename: alphanumeric
    5.3. image_path: alphanumeric
    5.4. thumbnail_path: alphanumeric
    5.5. post_id: foreign key, numeric

6. Group (Strong)
    6.1. group_id: primary key, numeric
    6.2. name: alphanumeric
    6.3. description: alphanumeric
    6.4. picture_path: alphanumeric
    6.5. picture_thumbnail_path: alphanumeric
    6.6. 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. group_id: foreign key, numeric
    8.4. 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:

Registration: Guest users can register for an account by entering their first name, last name, email address, SFSU ID number, password, and uploading their SFSU ID picture into the registration form. Optionally, a profile picture can be uploaded. Upon submitting the form, the user's browser and the server will validate the input, then the server will store the user's data in the database. The user will be notified with a message saying whether the registration was successful or not.

Login: Approved users can log in to their account by entering their SFSU ID number and password into the login form. Upon submitting the form, the server will check the database to determine if the entered credentials were valid. If the credentials were valid, the user will be logged into their account and stay logged in until they log out. If the credentials were invalid, the user will be notified with a message.

Search: All users can search for users, though only approved users can search for marketplace listings and forum posts. The user can enter search terms and filters into the search bar to refine their search results. Upon submitting the search request, the server will search the database for items matching the user's search terms and filters. If there are matching results, the browser will display these results. If the search terms do not match anything, the browser will still display some recommendations.

Forum Thread/Post Creation: Approved users can create forum threads/posts by entering a message and optionally attaching images. Upon submitting the forum thread/post, the browser and server will validate the data, then store it into the database. The user will be notified with a message saying whether the thread/post creation was successful or not.

Marketplace Listing: Approved users can post listings in the marketplace by uploading images, a title, description, price, category, payment methods, delivery methods, and contact methods. Upon posting the listing, the browser and server will validate the data then store it into the database. The listing will be visible to other users as soon as it is posted. The user will be notified with a message saying whether the marketplace listing was able to be created or not.

Group Creation: Approved users can create a group by entering the group's name, description, and uploading its picture into the group creation form. Upon submitting the form, the browser and server will validate the data, then store the new group in the database. The user will be notified if the group was able to be created or not.

Gator Chat / Group Chat / Direct Messaging: Approved users can send messages in Gator Chat, a group's chat, or by direct messaging by entering a message and optionally attaching an image. Upon sending the message, the server will validate the data, then store it into the database. The message will be visible to its recipient(s) as soon as it is sent.

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 sort the search results by price from lowest to highest or highest to lowest. The user may also search for a specific category of items using a category filter.

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

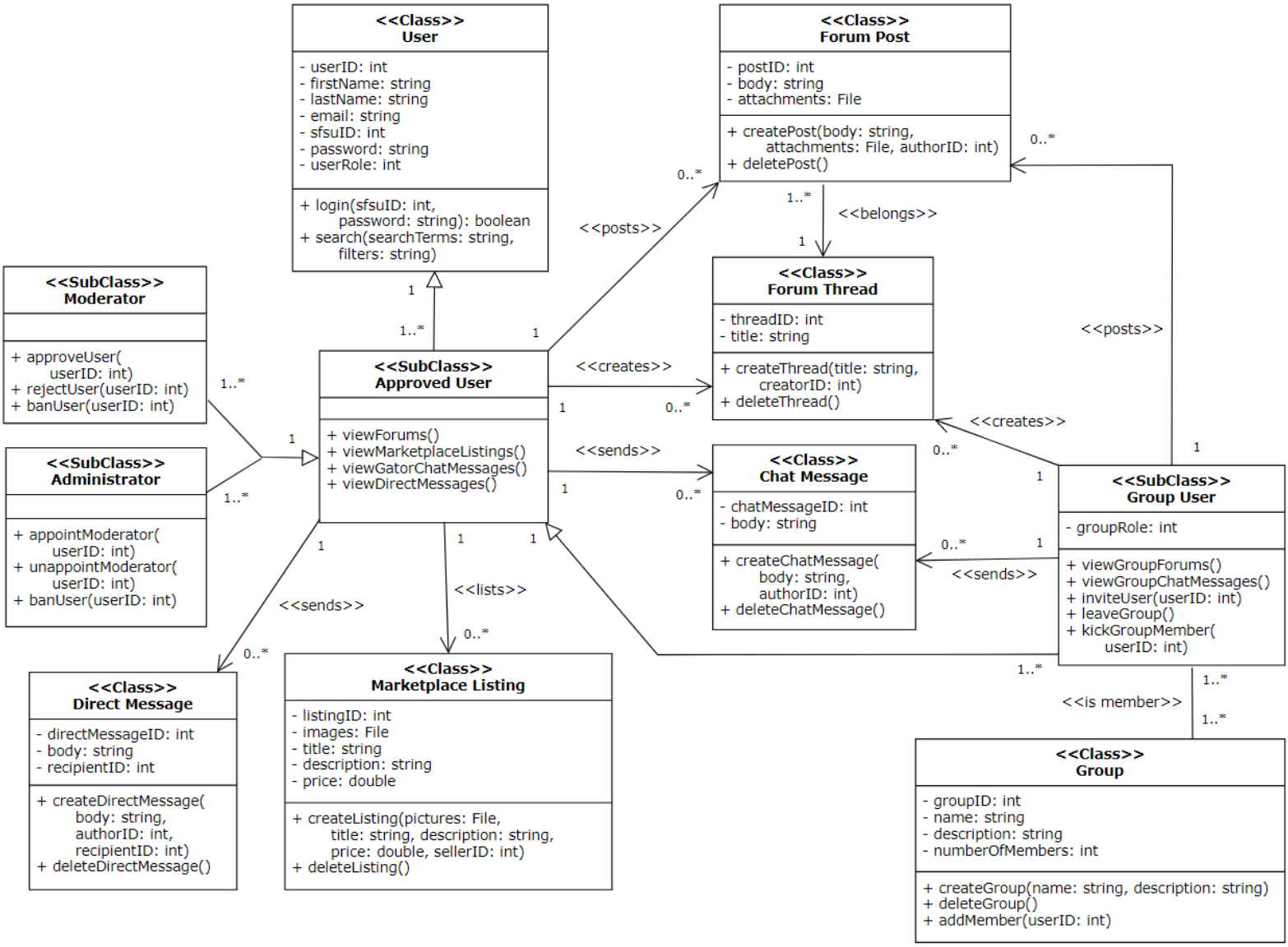
Forum Thread Sort: We will create a sorting algorithm for forum threads so that the user can see forum threads sorted by creation date, last reply date, or by thread title. The creation date and last reply date sort will be from newest to oldest or oldest to newest, while the thread title sort will be based on the alphabetical order of the title: from A to Z or from Z to A.

Forum Thread Pins: We will 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.

Seller Feedback/Rating: We will 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.

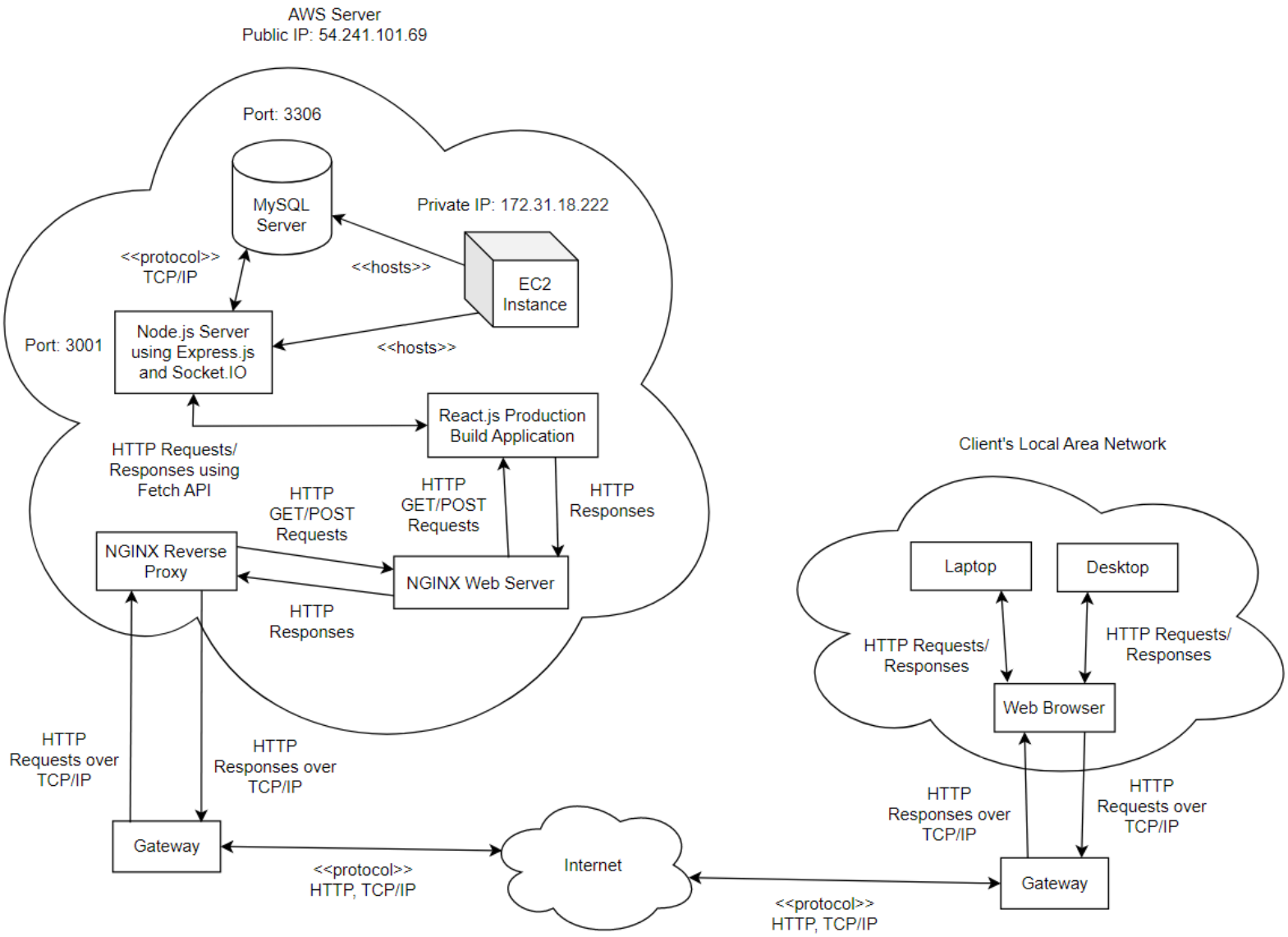
6. High Level UML Diagram

UML Class Diagram:

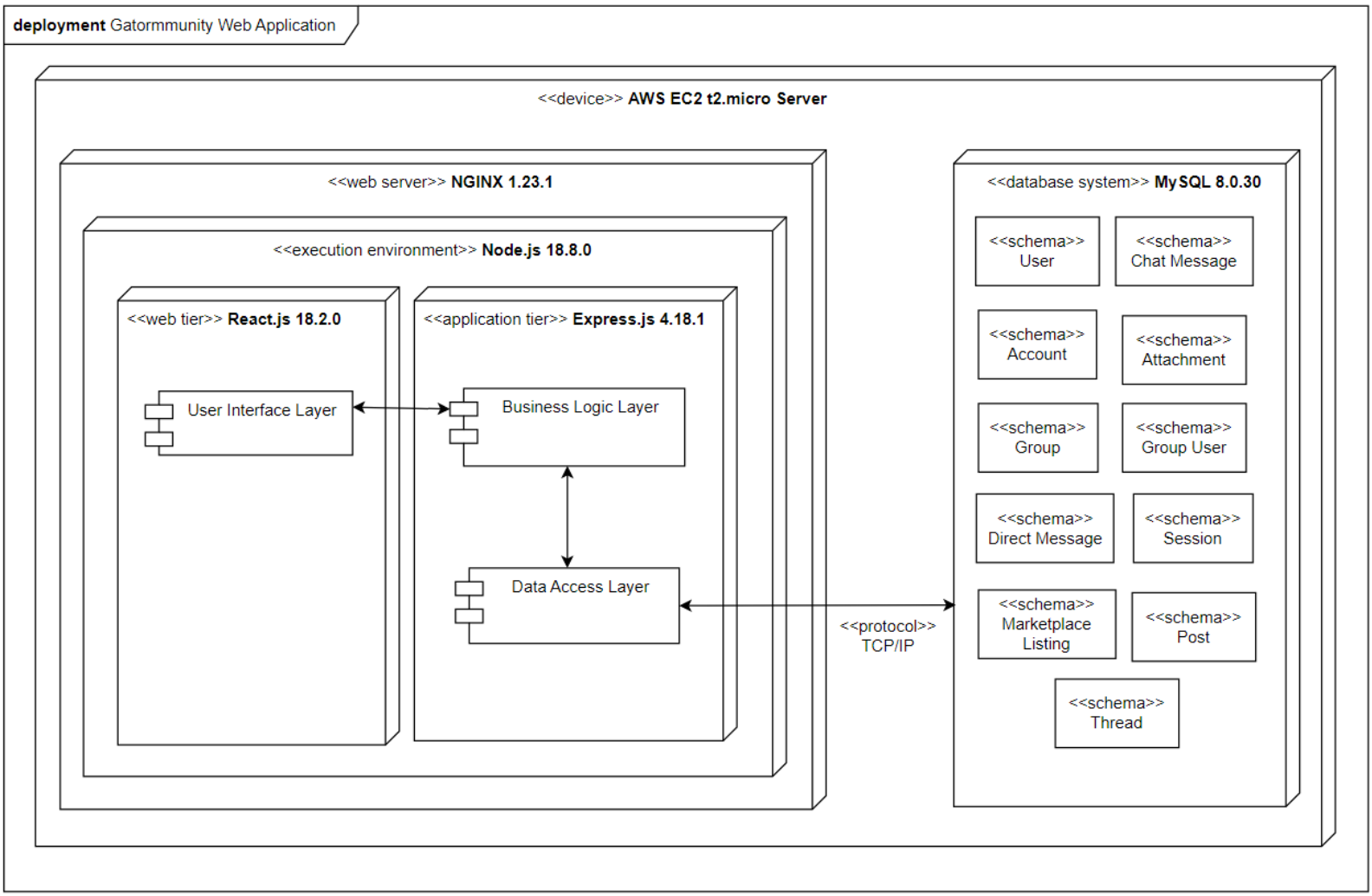


7. High Level Application Network and Deployment Diagrams

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