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.

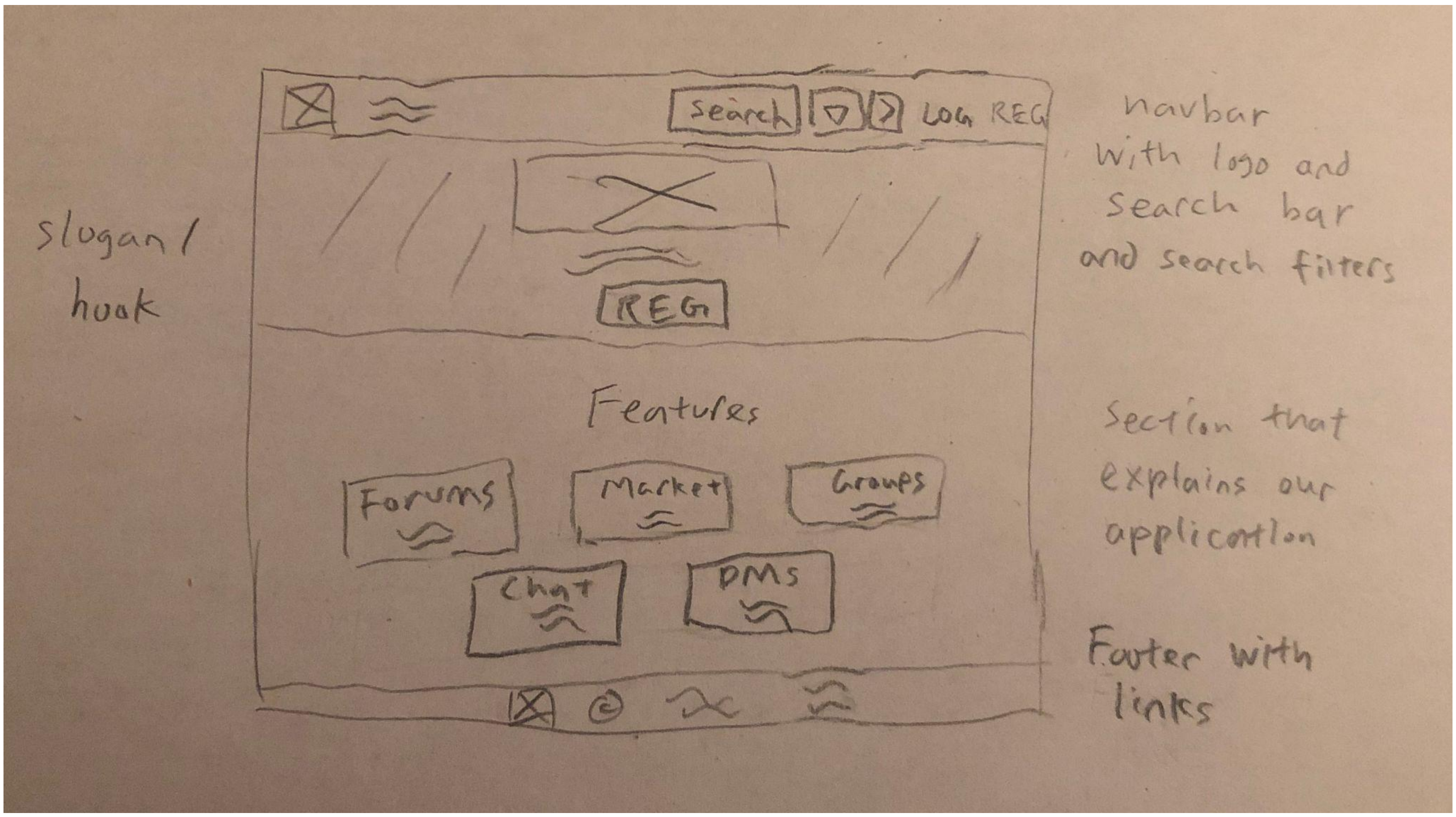
15. Group Administrator
    15.1. A group administrator shall be able to create new categories in their group's forum.
    15.2. A group administrator shall be able to delete categories in their group's forum.

3. UI Mockups and Storyboards

UI Mockups of the Main Areas of the GUI:

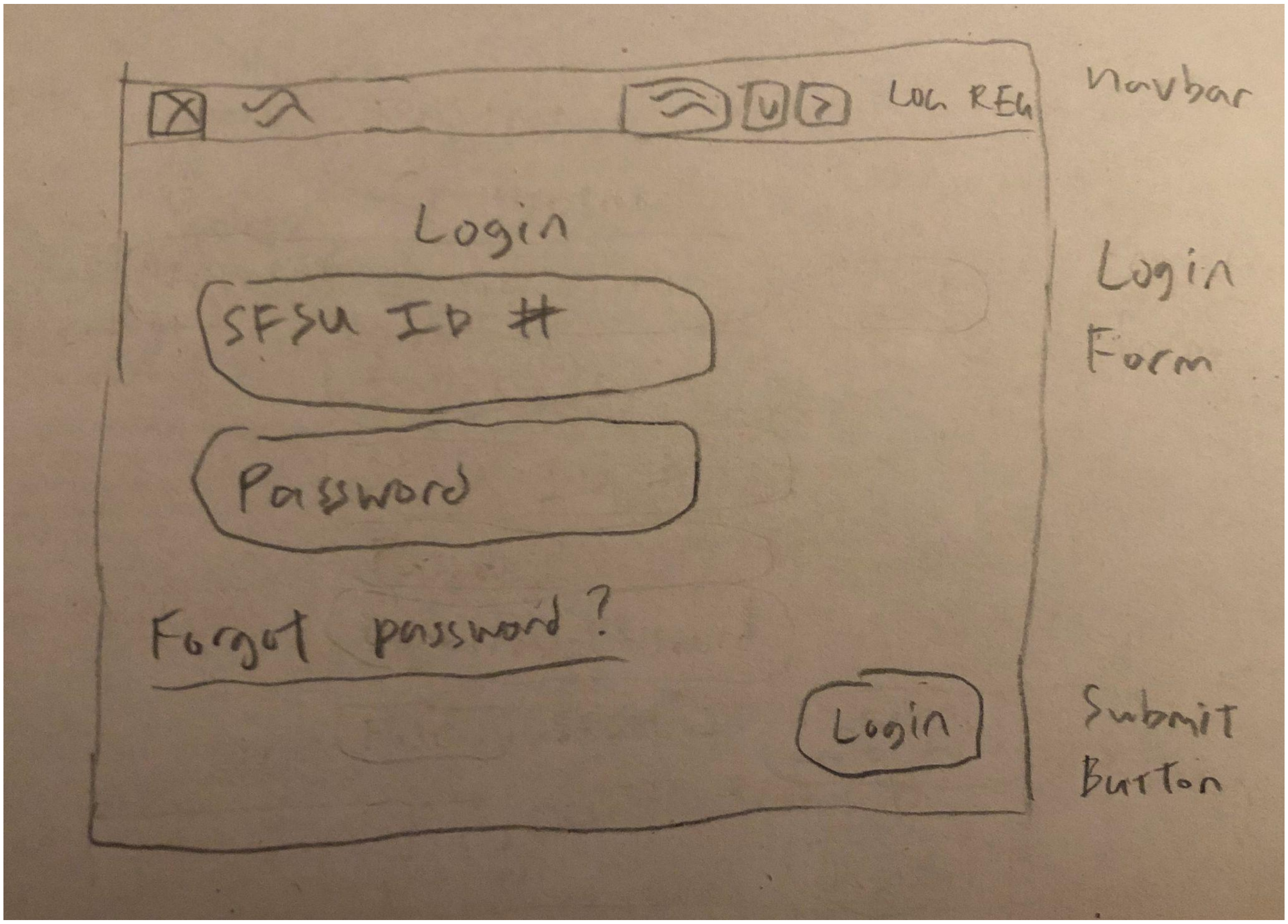
Home Page:

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.



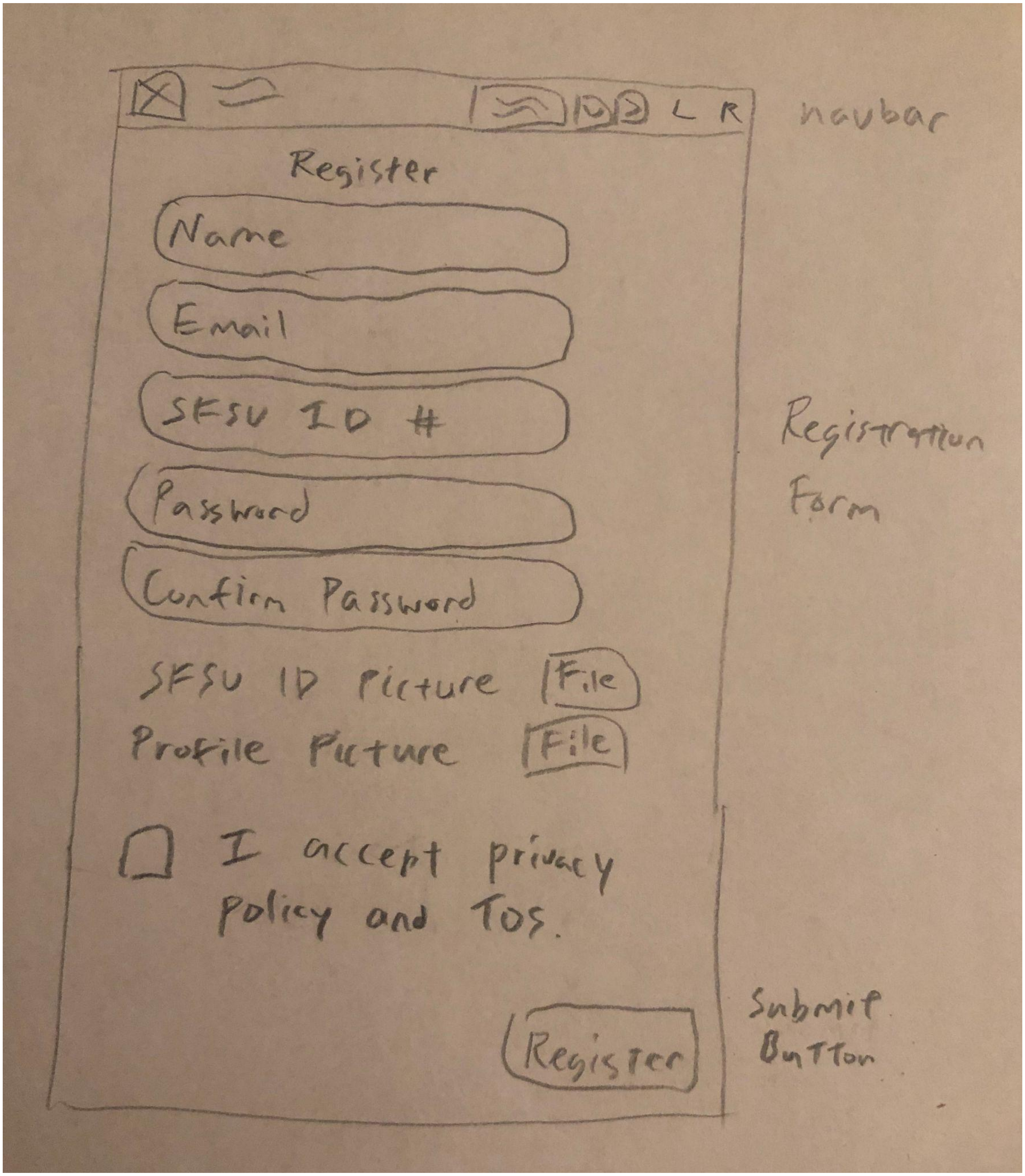
Login Page:

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



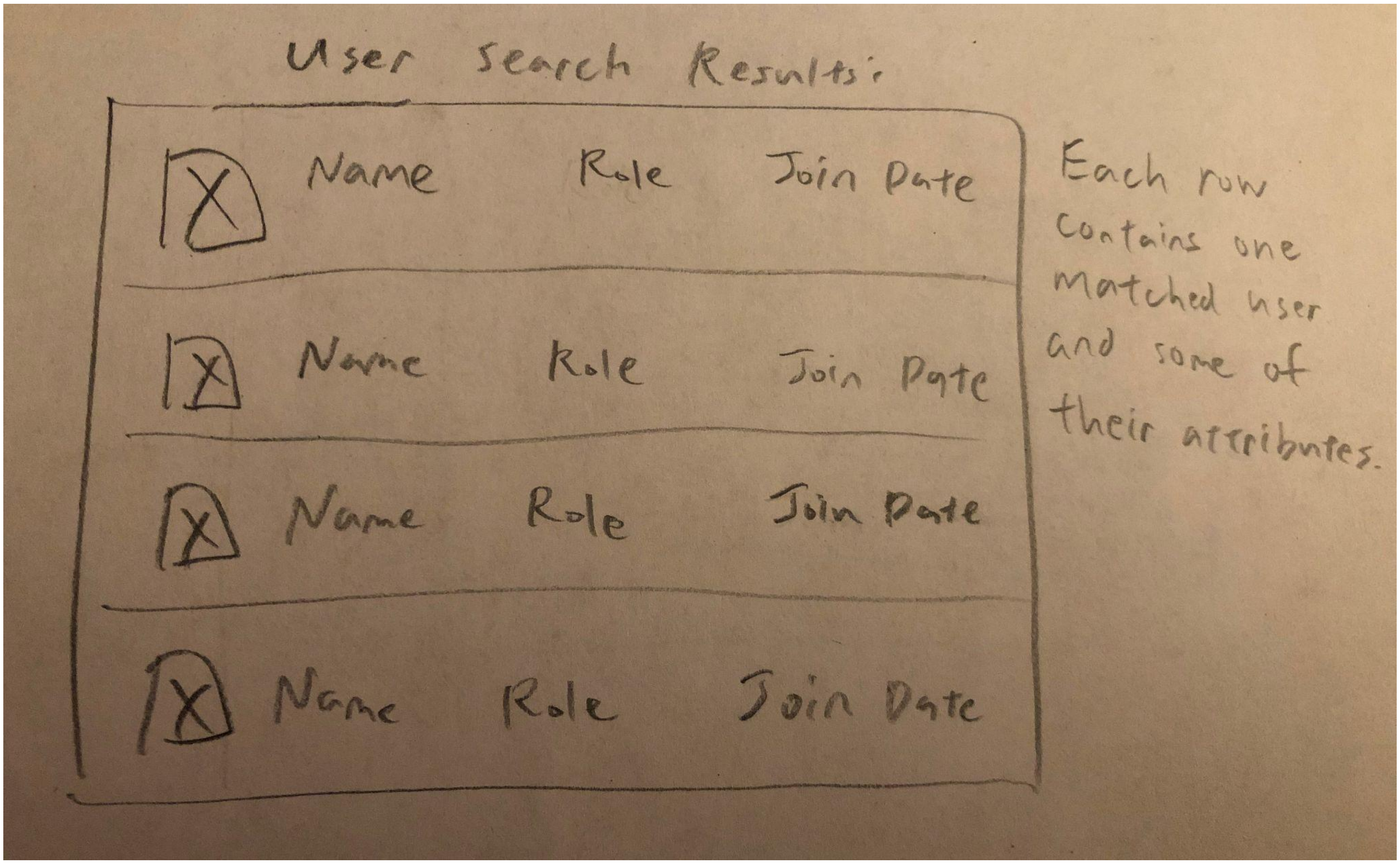
Registration Page:

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



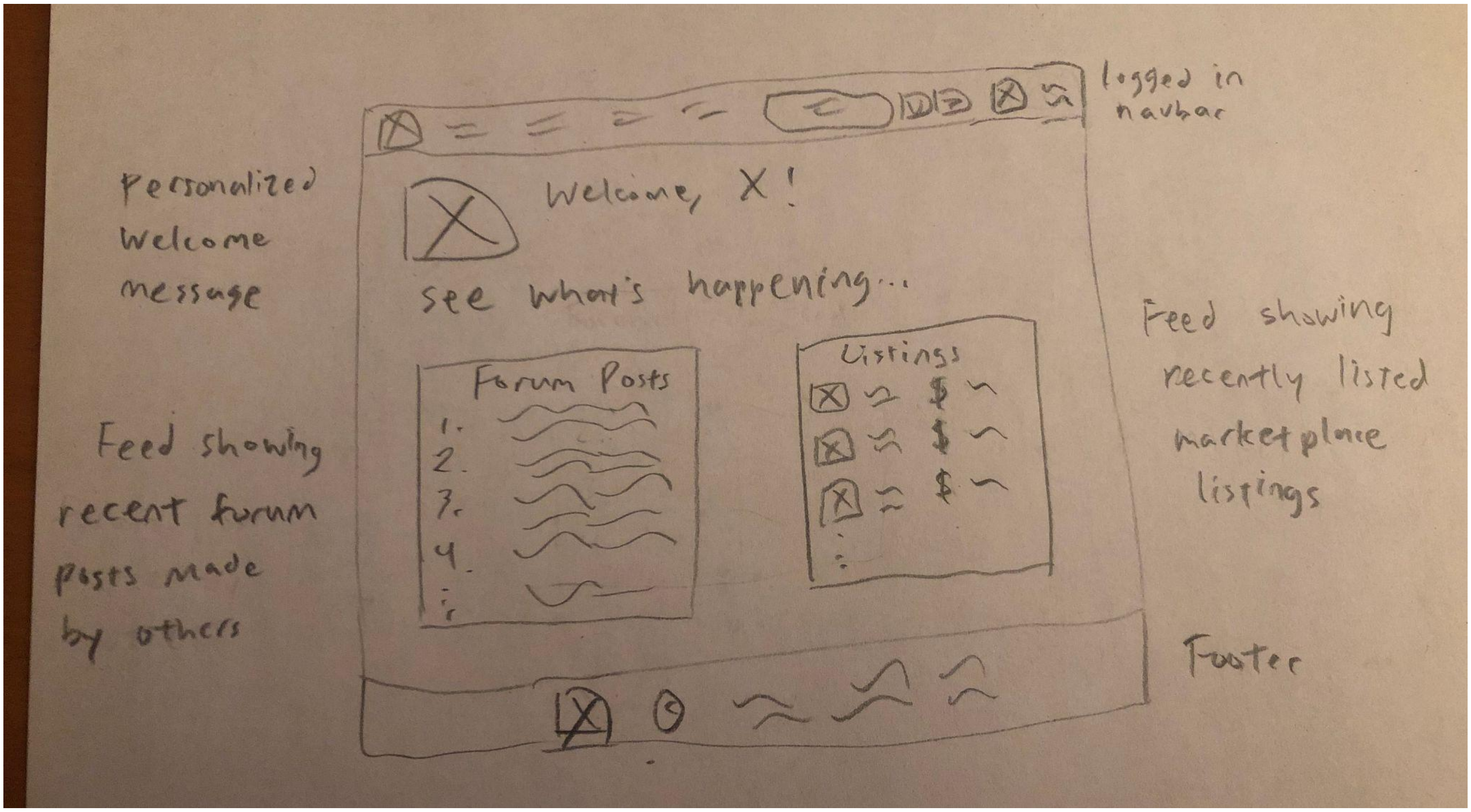
User Search Results:

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.



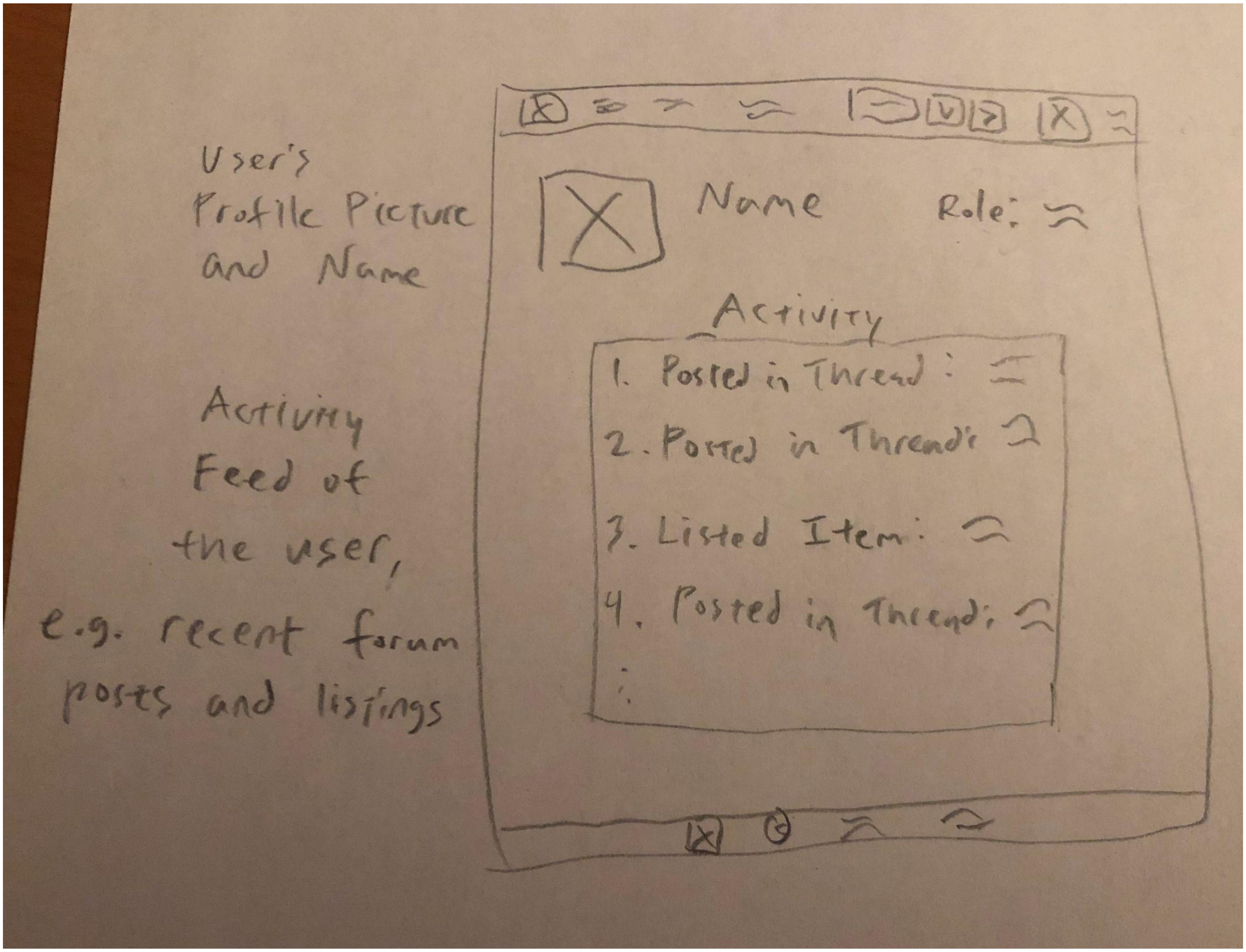
Dashboard Page:

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



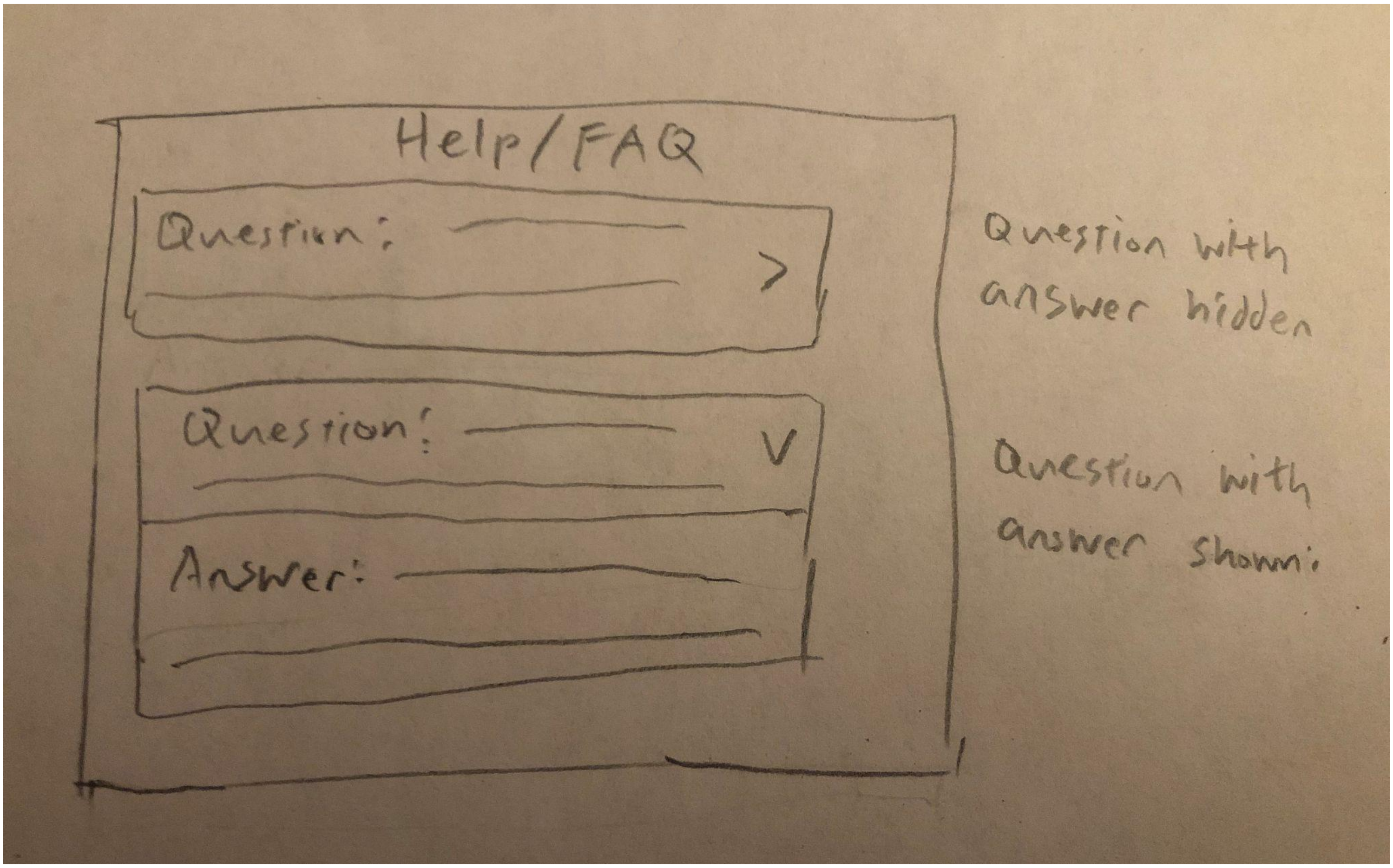
User Profile Page:

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.



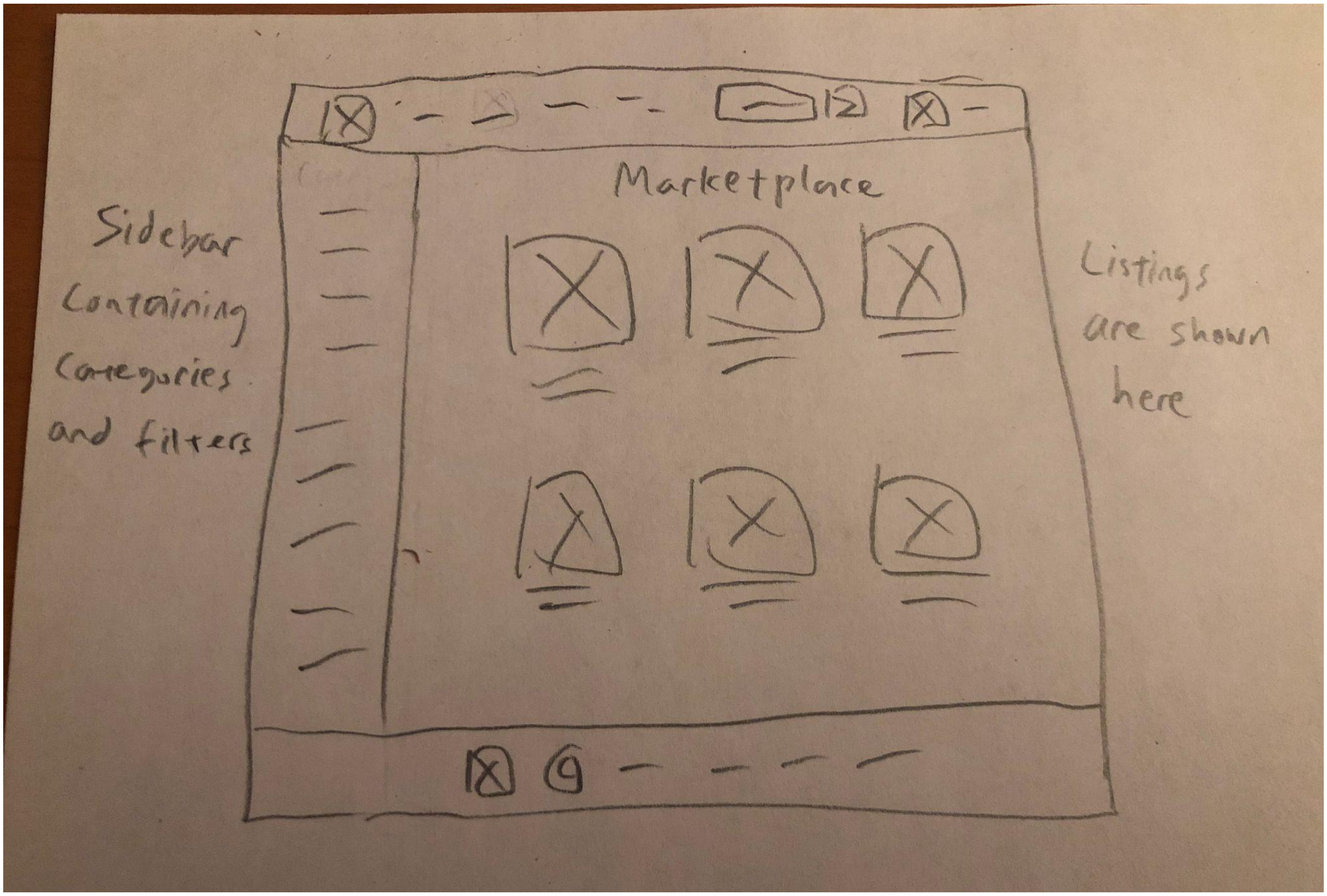
Help/FAQ Page:

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



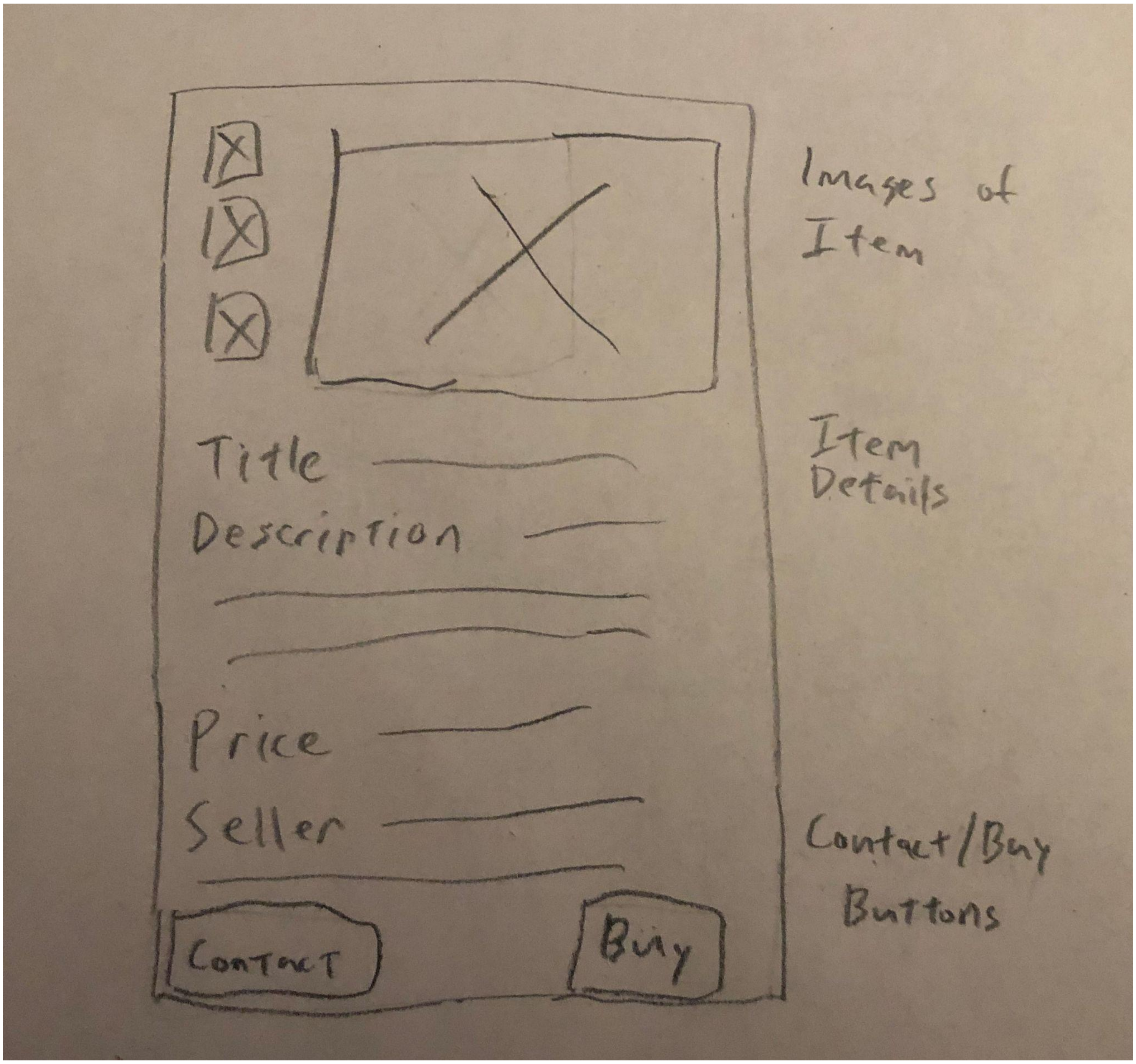
Marketplace Page:

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



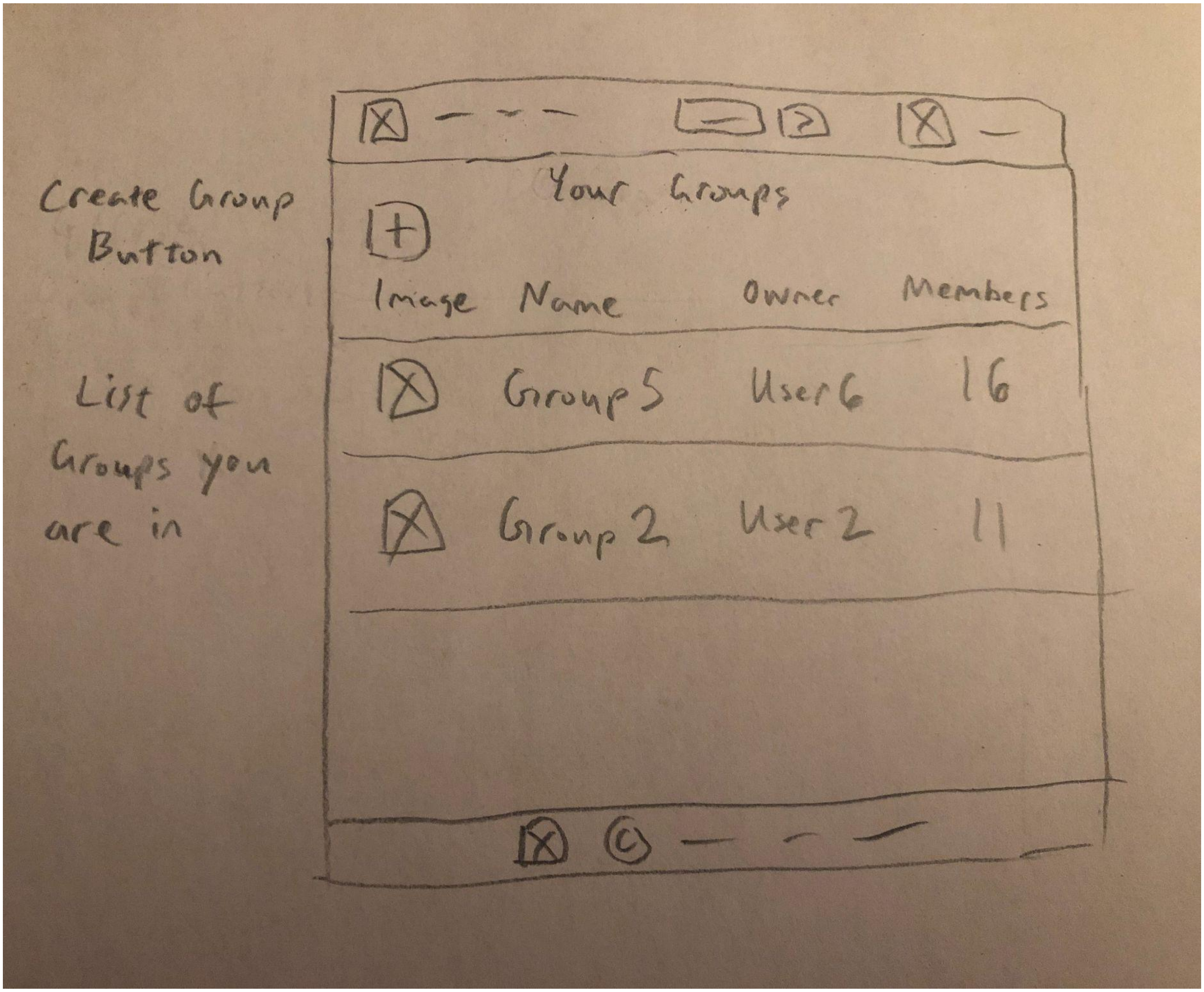
Marketplace Listing:

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.



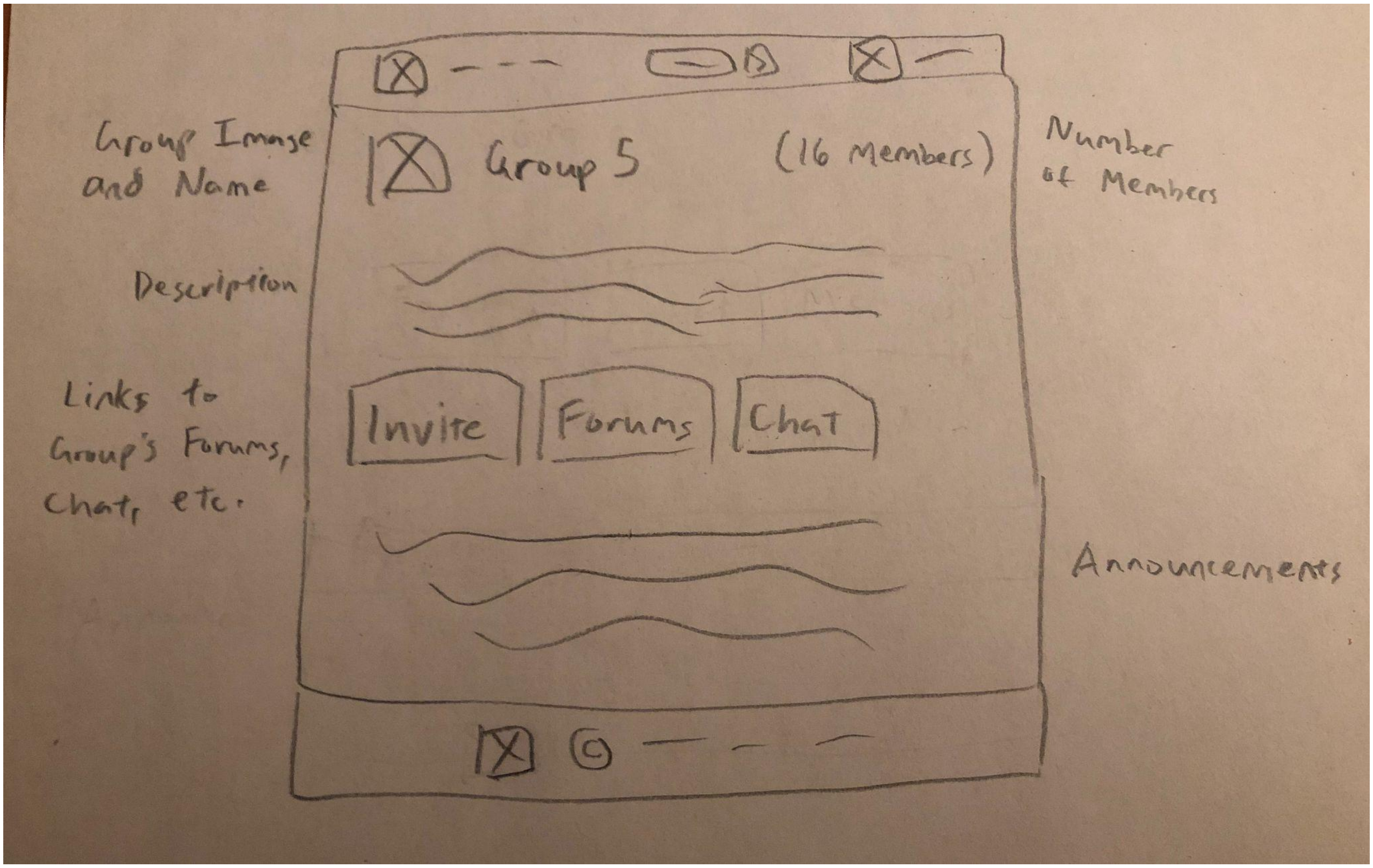
User's Groups:

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



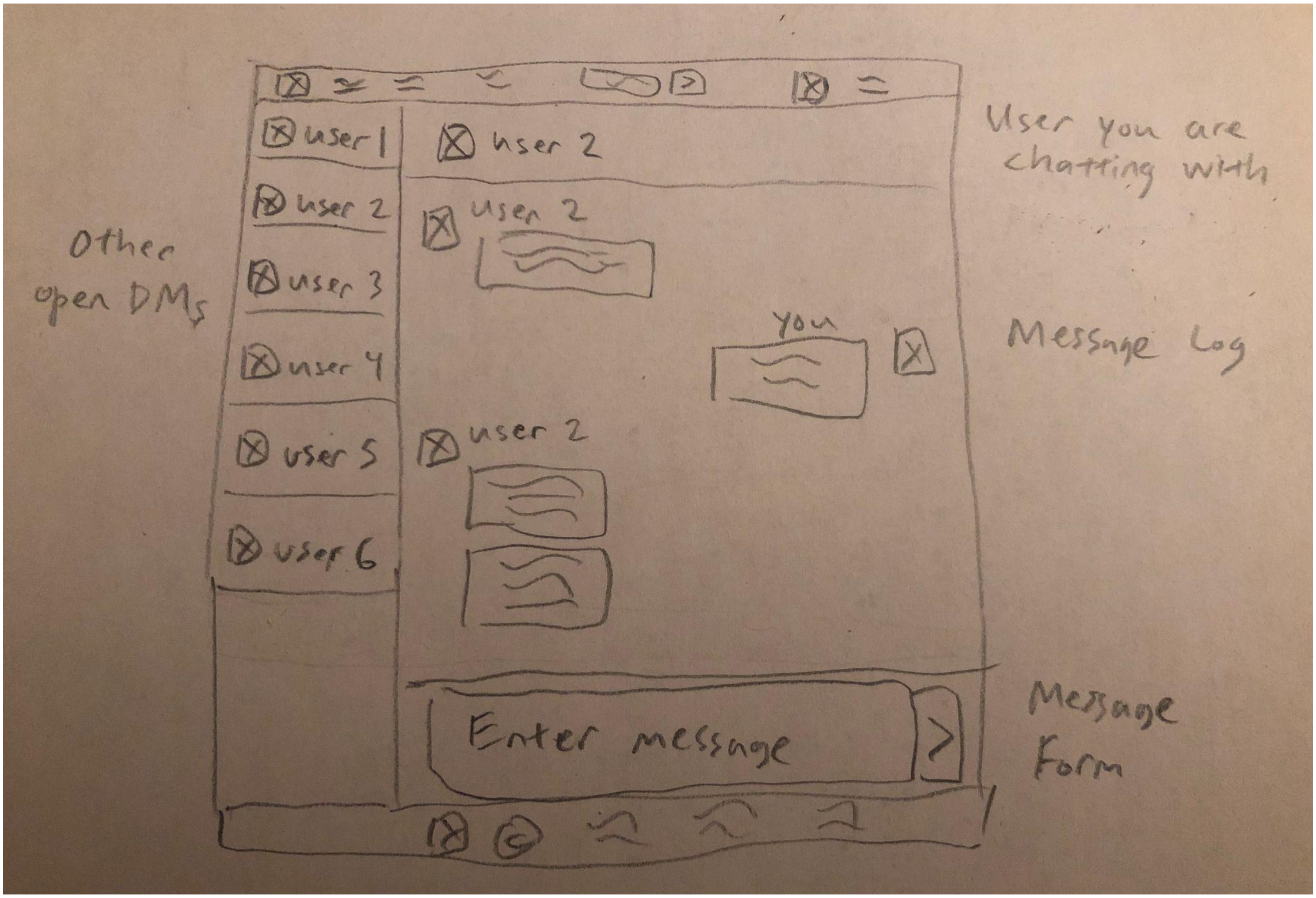
Group Home Page:

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



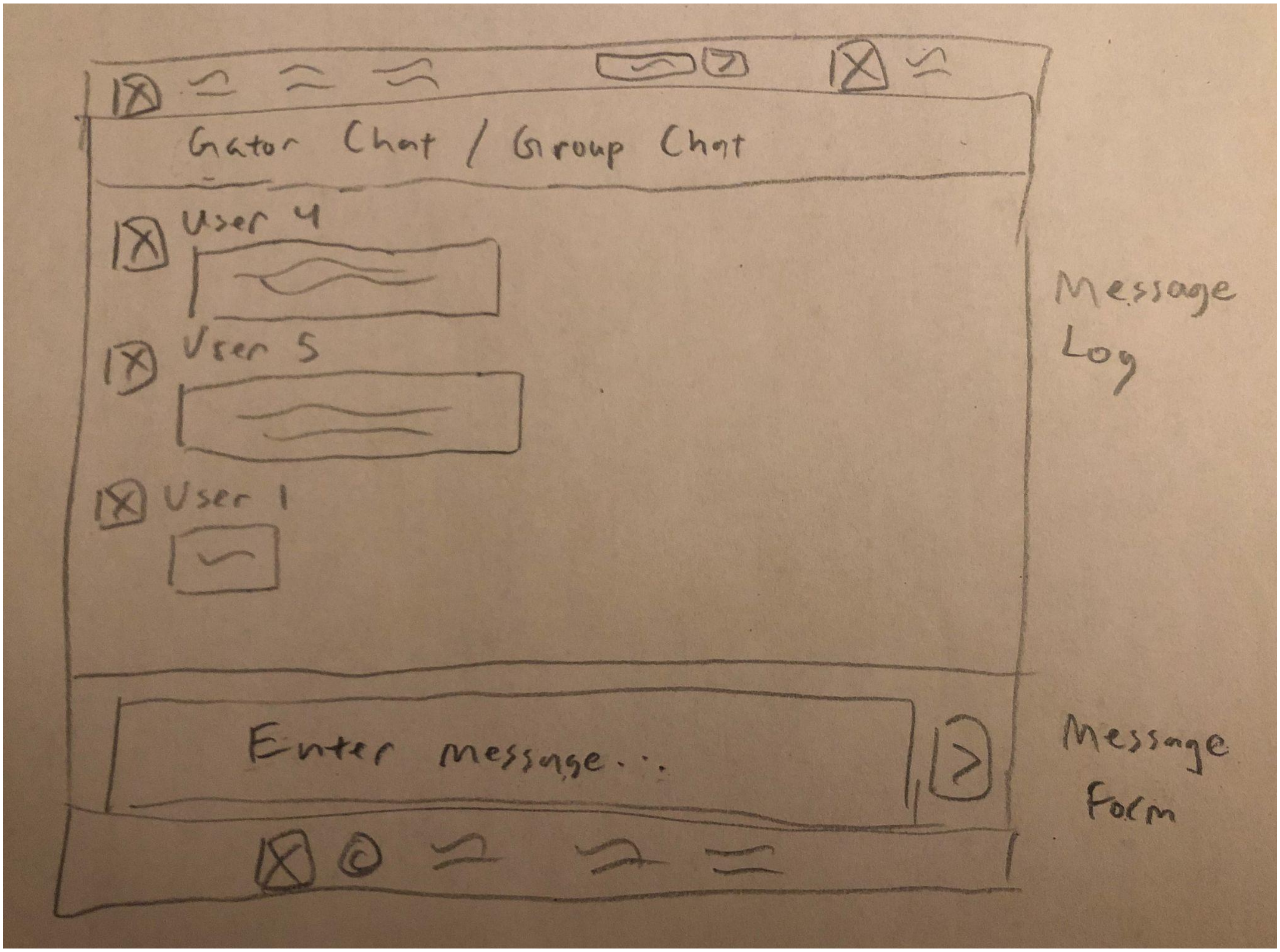
Direct Message (DM):

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



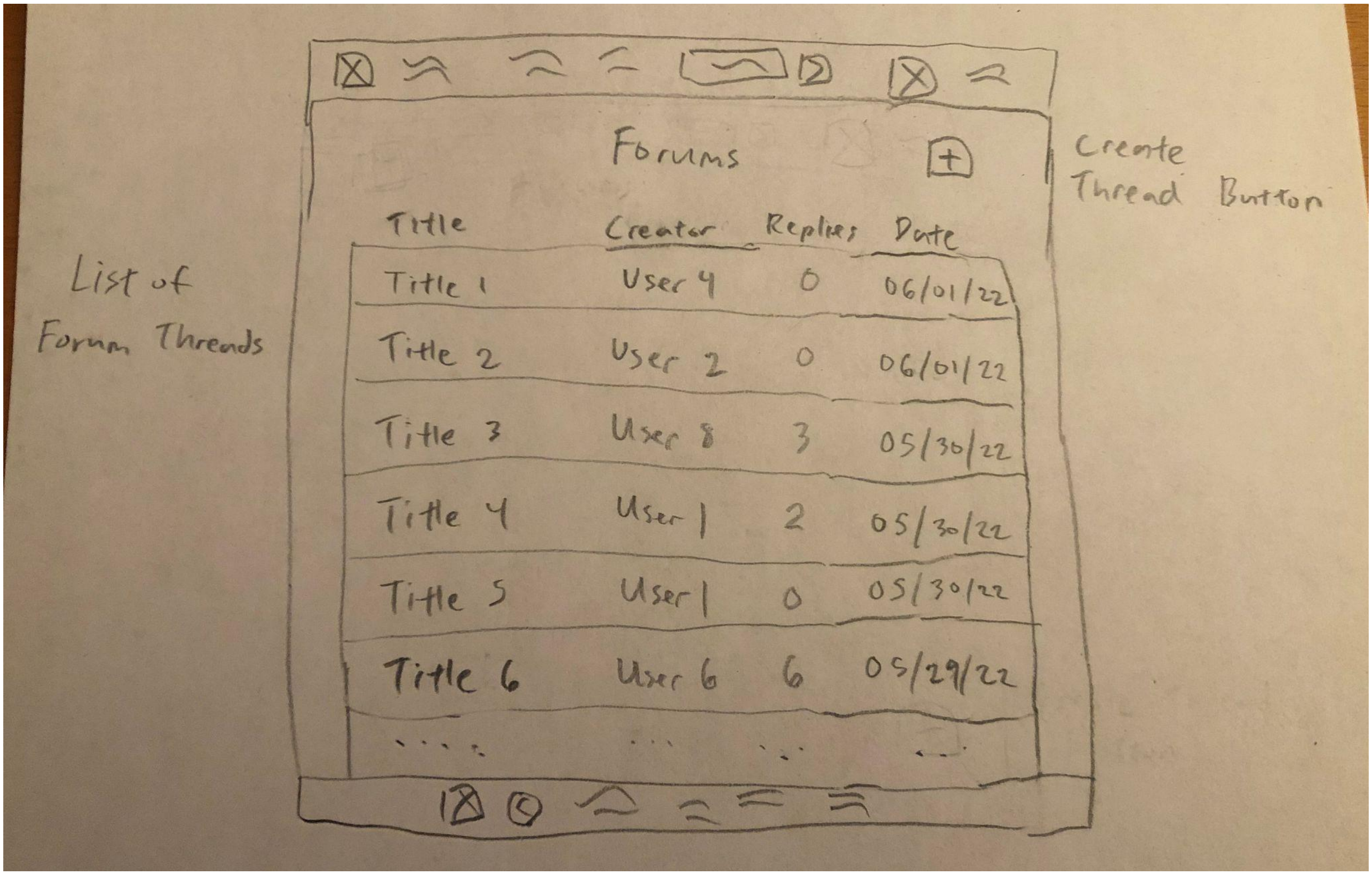
Gator/Group Chat:

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



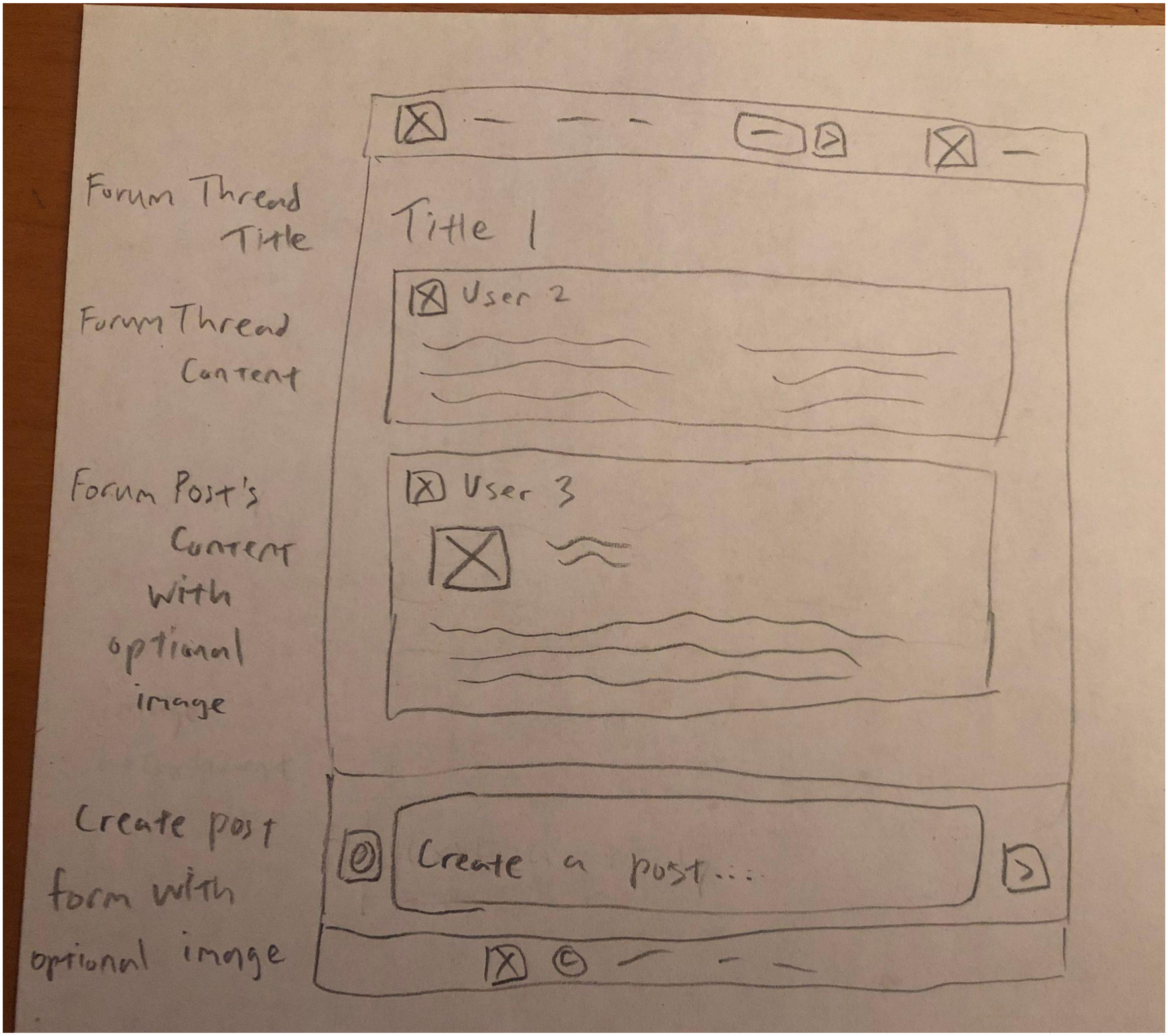
Gatormmunity/Group Forum:

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



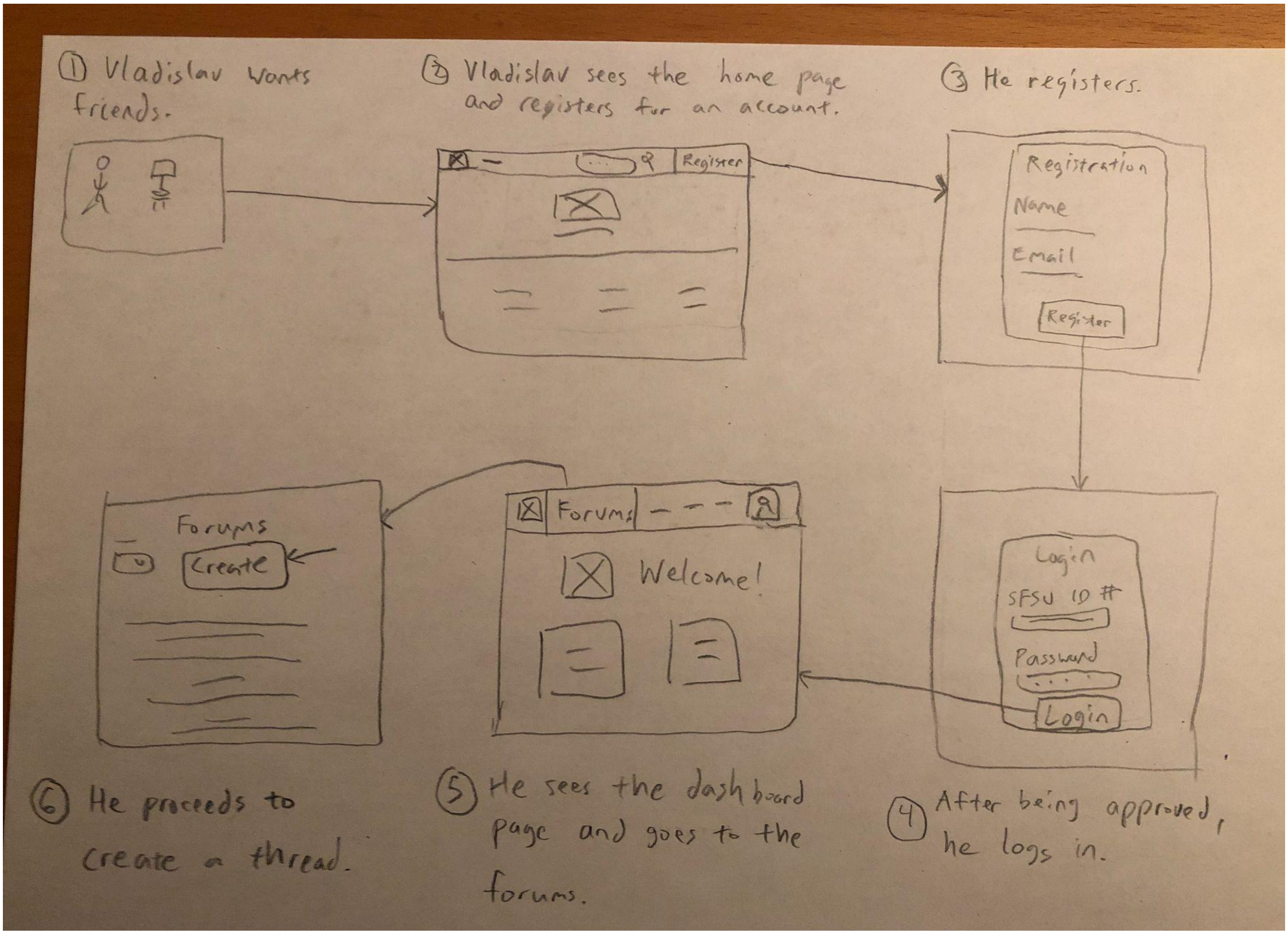
Viewing a Gatormmunity/Group Forum Thread:

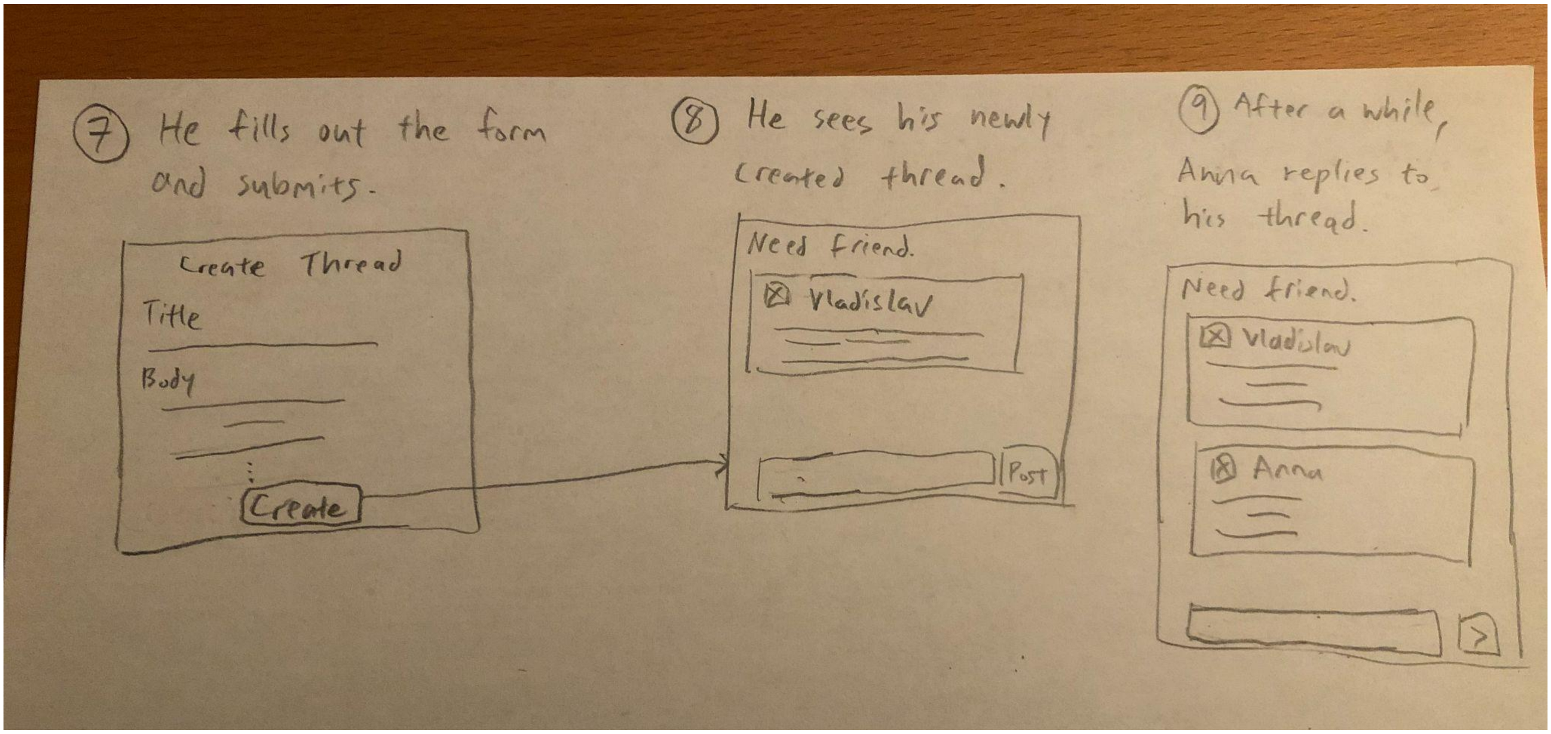
This page is accessible by clicking on a thread in the forums, for instance.



Storyboards for Use Cases:

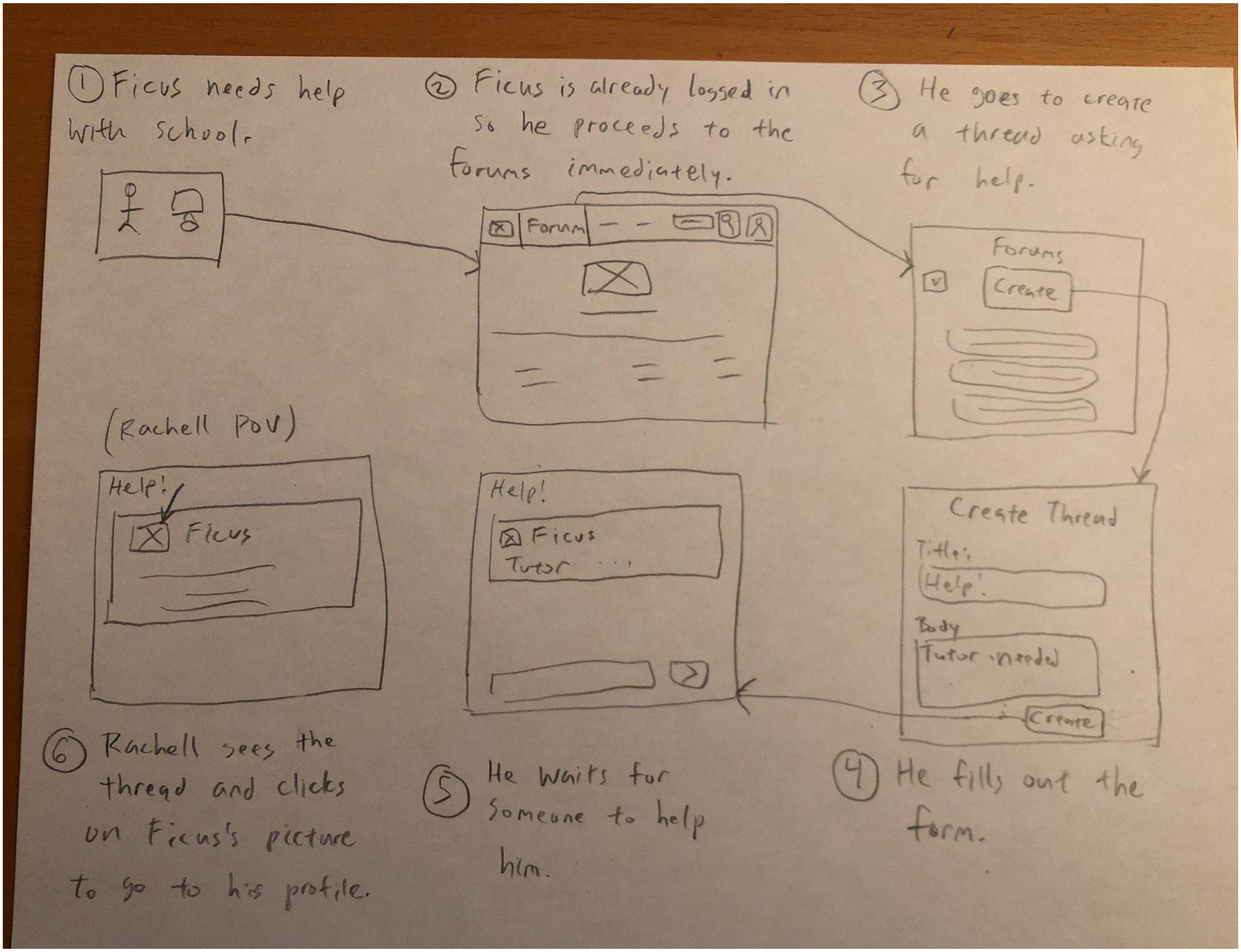
Use Case #1: Russian Friends
Vladislav is trying to make friends in SFSU so he proceeds to make an account on Gatormmunity. After registering and logging in, he goes to the forums and makes a thread stating that he is looking for friends. Anna sees his thread and replies saying she would like to be his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

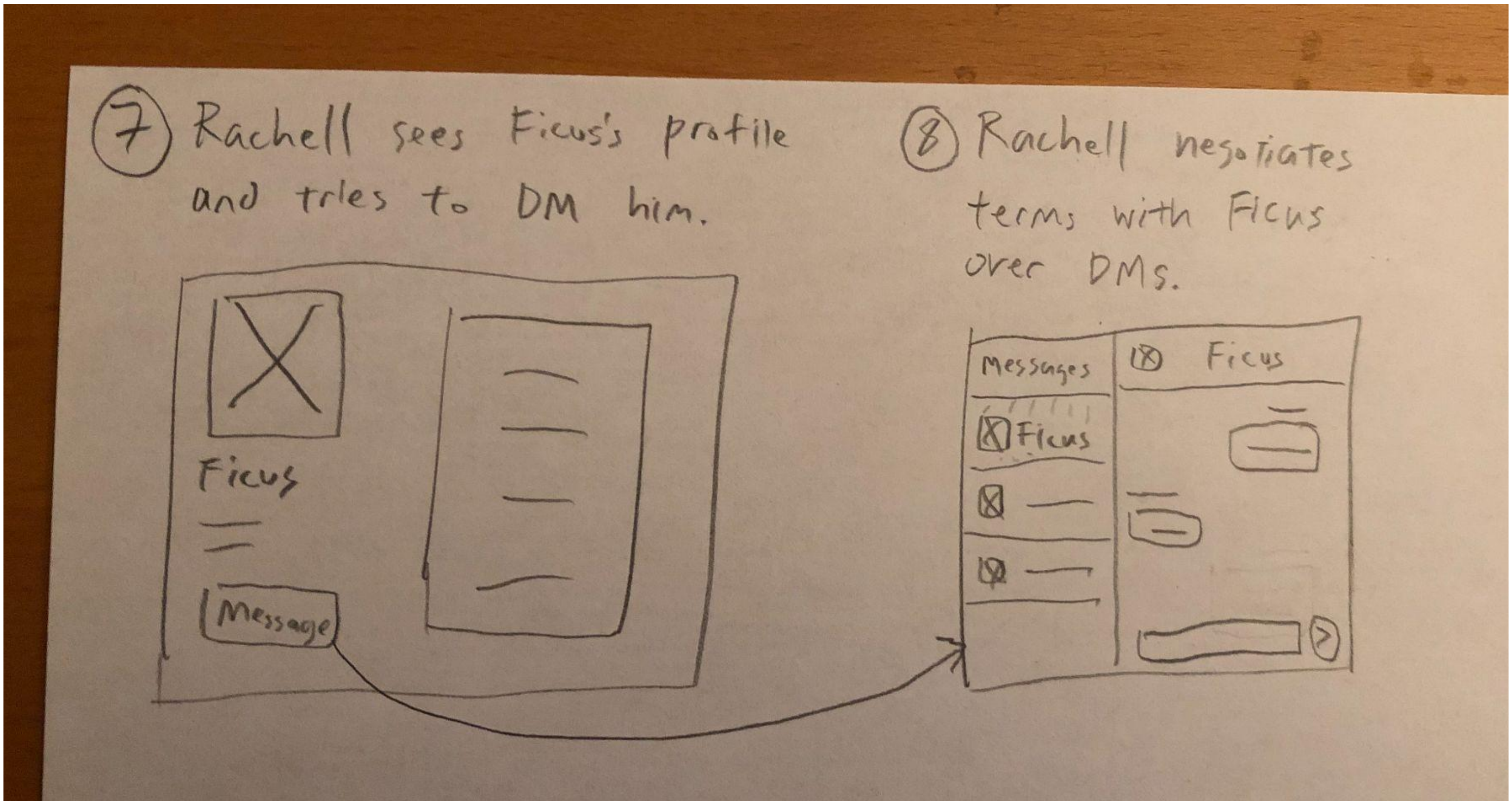




7 He fills out the form and submits.
Create Thread
Title
Body
Create
8 He sees his newly created thread.
Need friend.
Vladislav
Post
9 After a while, Anna replies to his thread.
Need friend.
Vladislav
Anna
>

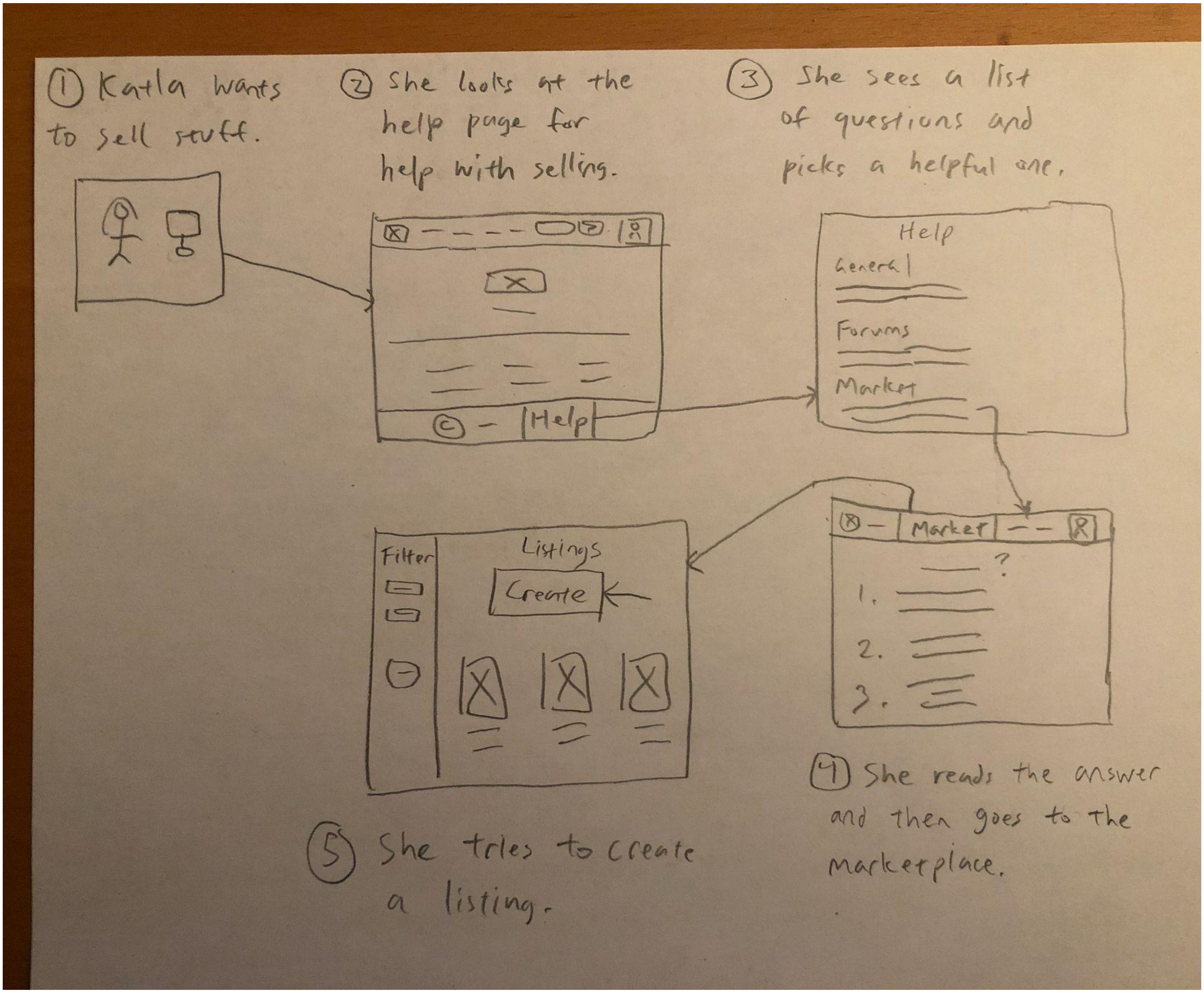
Use Case #2: Struggling Student
Ficus is looking for someone to help him with his coursework, thus he creates a thread in the forums asking for help. Rachell sees his thread and provides him with help after contacting him through a direct message.

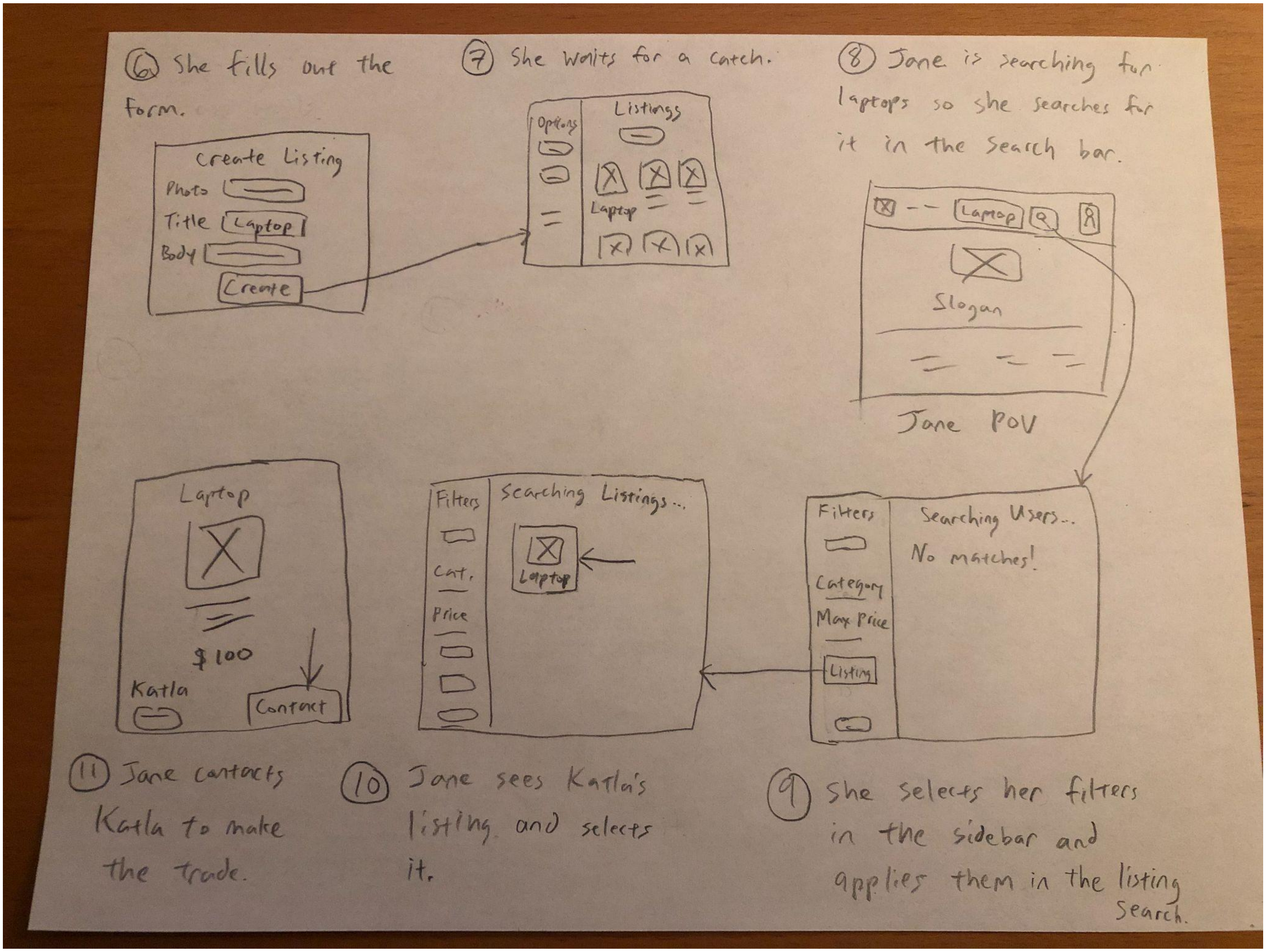


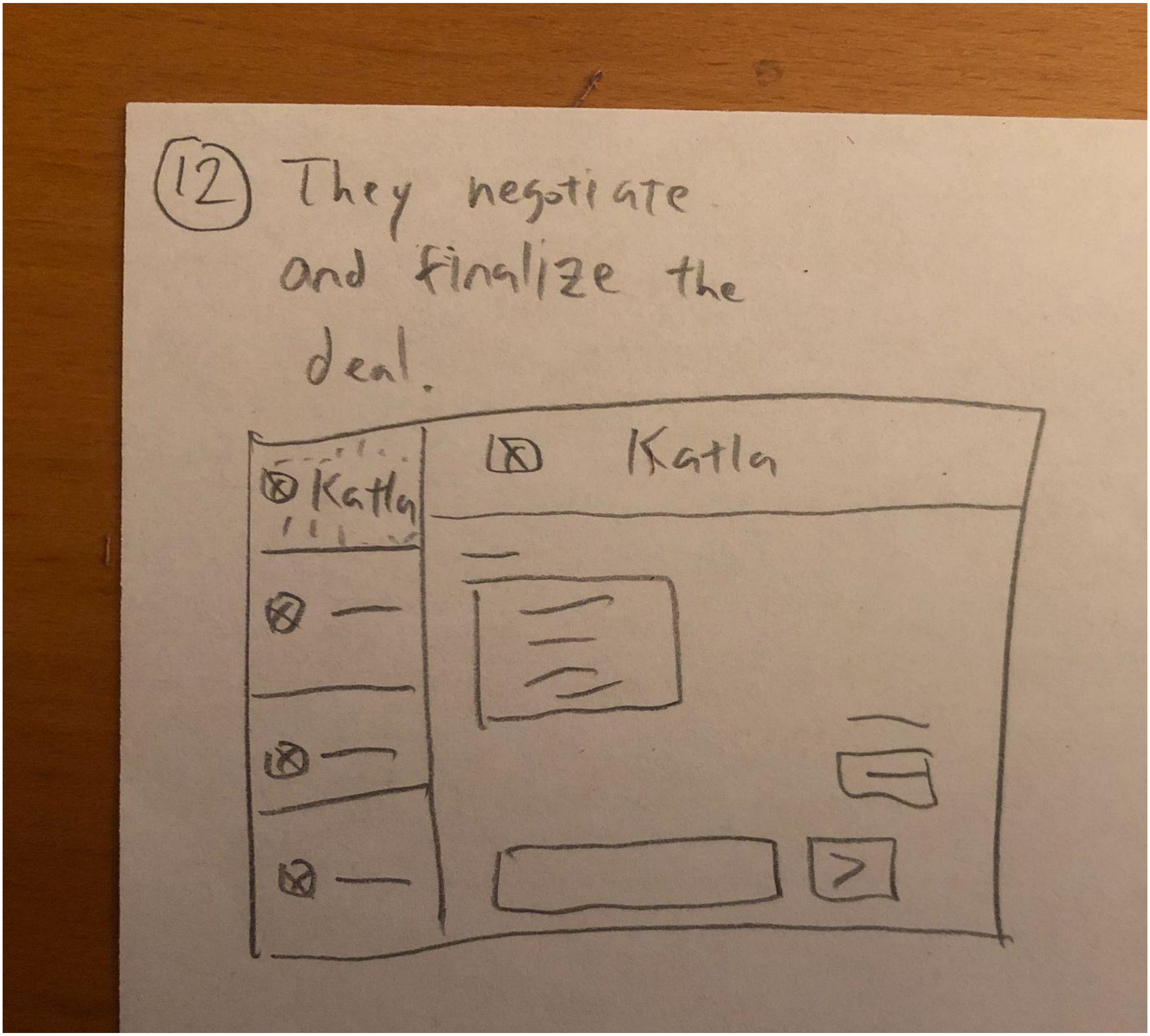

7 Rachell sees Ficus's profile and tries to DM him.
Ficus
Message
8 Rachell negotiates terms with Ficus over DMs.
Messages
Ficus
Ficus

Use Case #3: Wary Traders

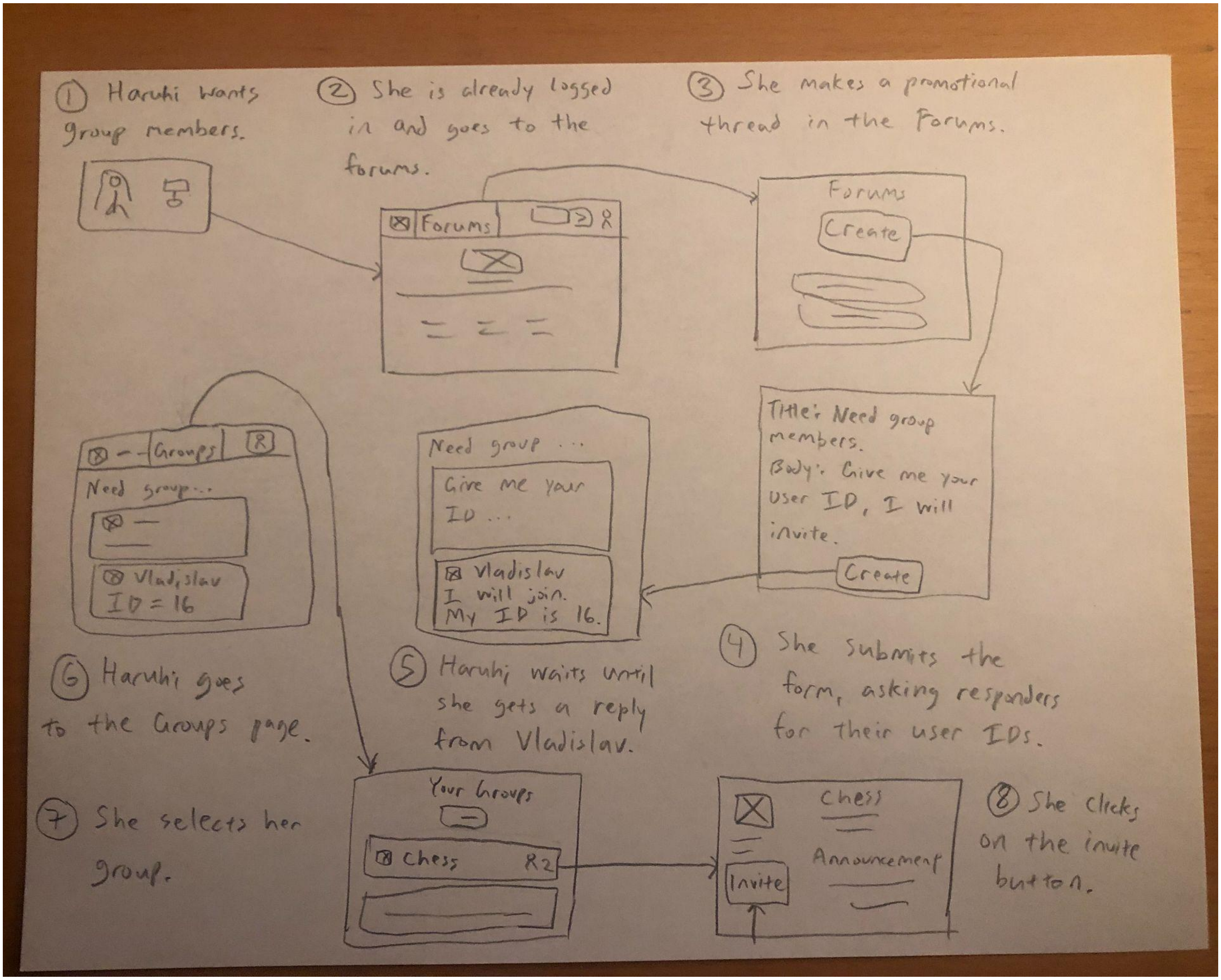
Katla wants to sell her stuff and so she uses Gatormmunity. She was not sure how to sell things at first, until she looked at the help page. She was then able to list her laptop for sale. Coincidentally, Jane was looking for a cheap laptop and saw Katla's listing. They agree to make the sale.

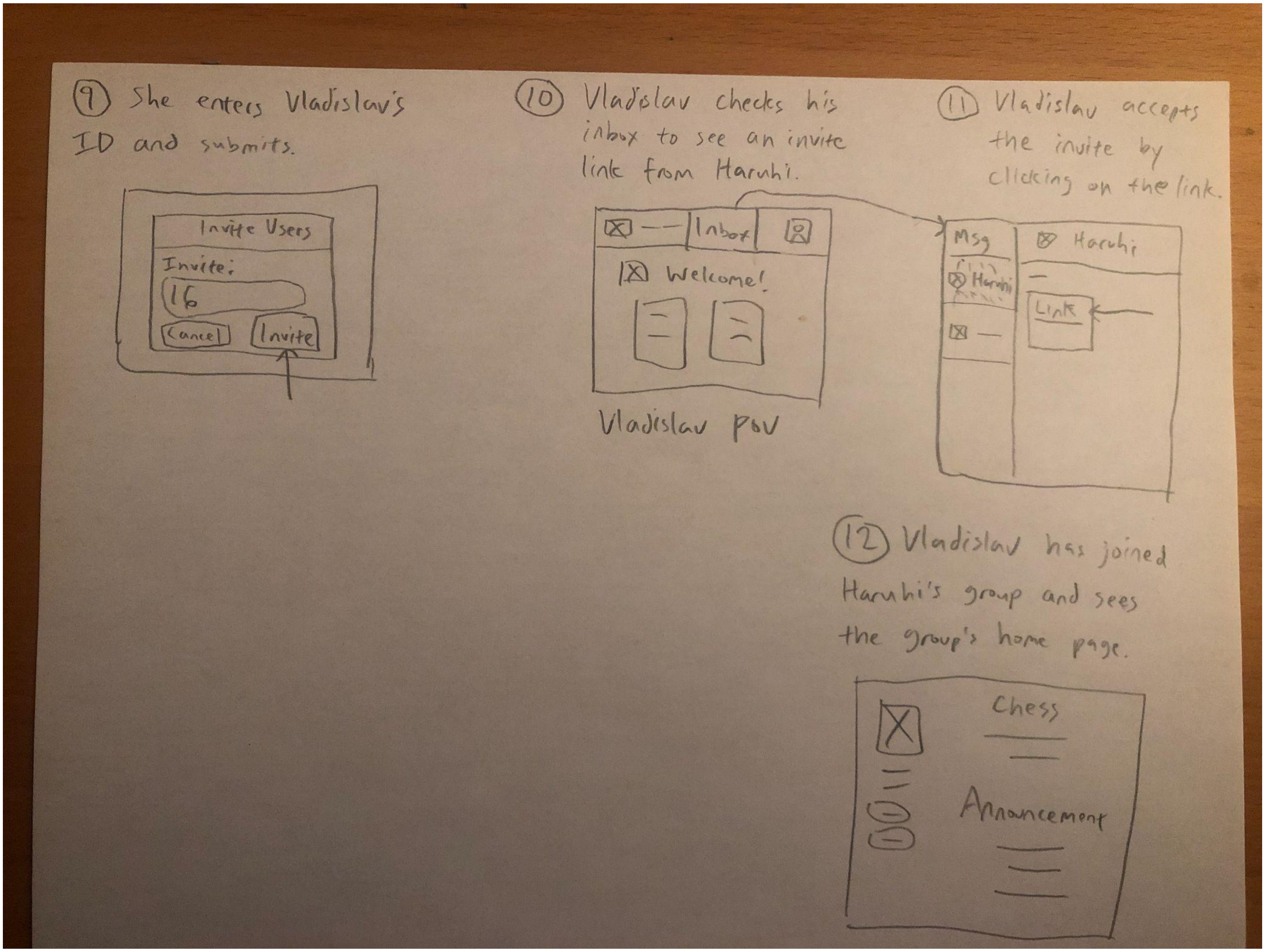



6 She fills out the form.
Create Listing
Photo
Title
Laptop
Body
Create
7 She waits for a catch.
Options
Listings
Laptop
8 Jane is searching for laptops so she searches for it in the search bar.
Laptop
Slogan
Jane POV
Laptop
$100
Katla
Contact
Filters
Searching Listings...
Cat.
Price
Laptop
Filters
Searching 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.
Katla
Katla

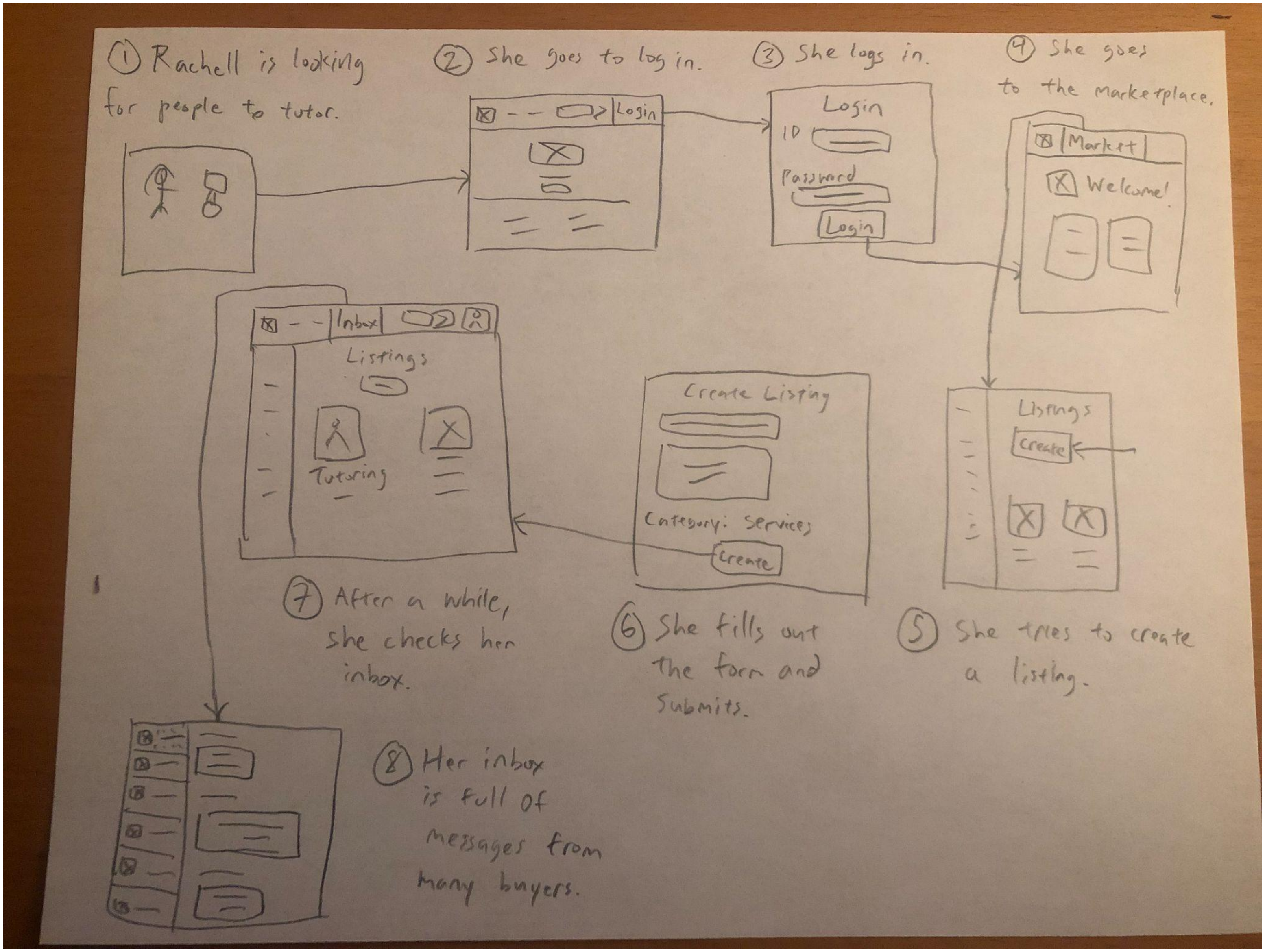
Use Case #4: Inactive Chess Club
Haruhi wants to grow her Gatormmunity group, thus she makes a thread in the Gatormmunity forums hoping to attract some new members. Vladislav sees the thread and gives Haruhi his user ID, which lets Haruhi invite him. He accepts the invitation.



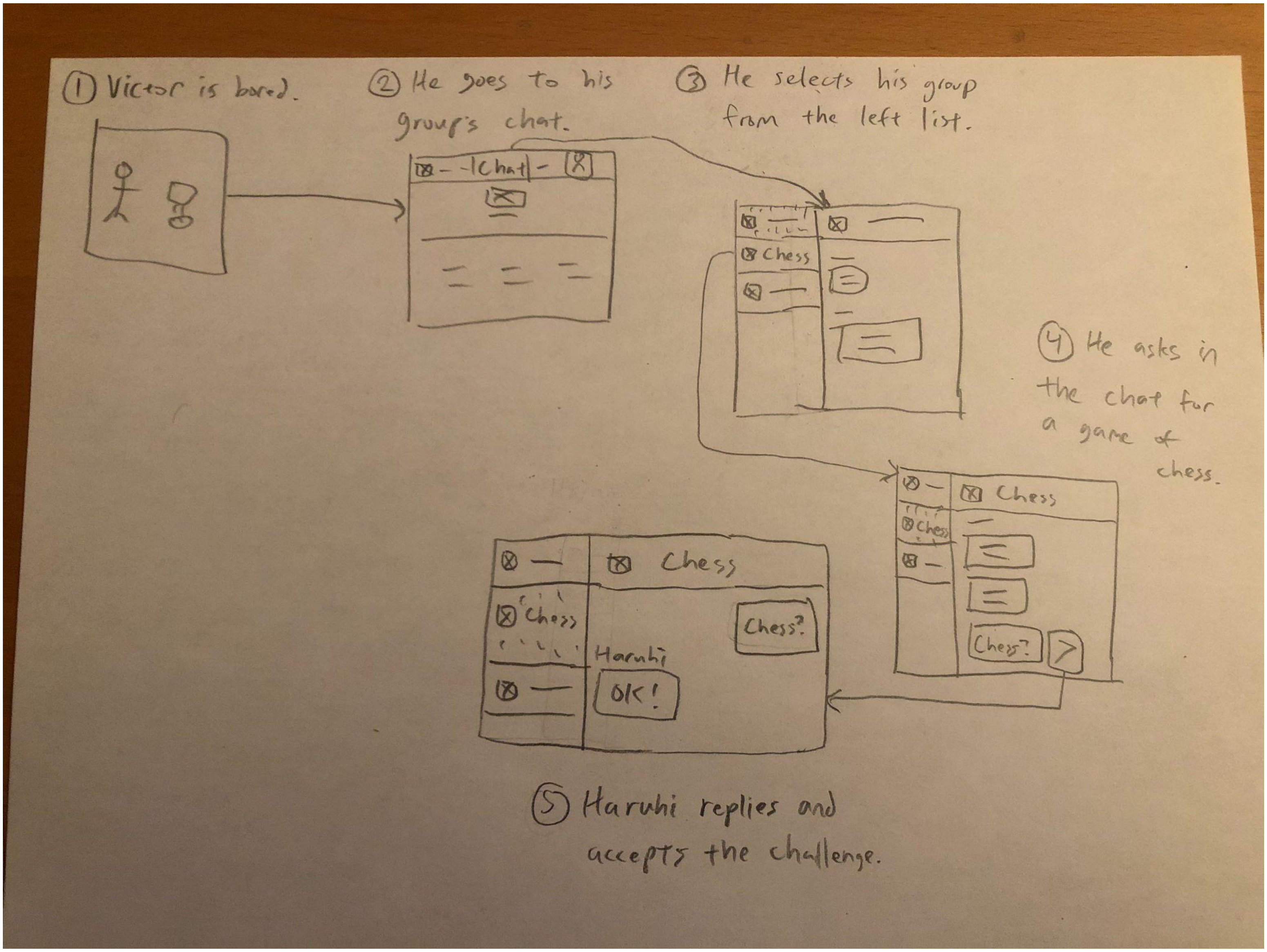

9 She enters Vladislav's ID and submits.
Invite Users
Invite:
16
Cancel
Invite
10 Vladislav checks his inbox to see an invite link from Haruhi.
Inbox
Welcome!
Vladislav POV
11 Vladislav accepts the invite by clicking on the link.
Msg
Haruhi
Haruhi
Link
12 Vladislav has joined Haruhi's group and sees the group's home page.
Chess
Announcement

Use Case #5: Tired of Spam

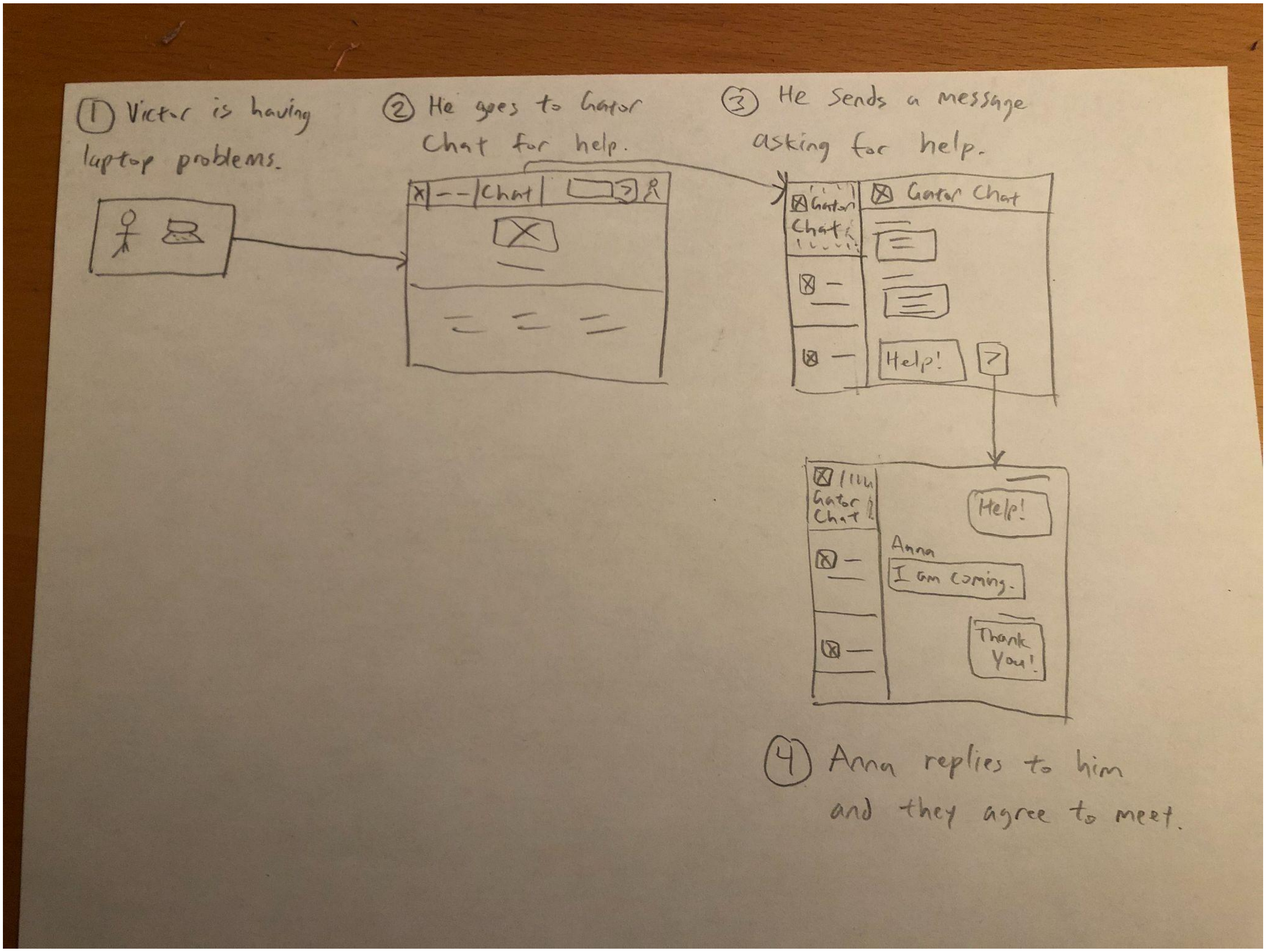
Rachell is trying to find students to tutor on Gatormmunity. She makes a listing in the marketplace offering her services and she is quickly inundated with messages from buyers who wish to pay for her services.



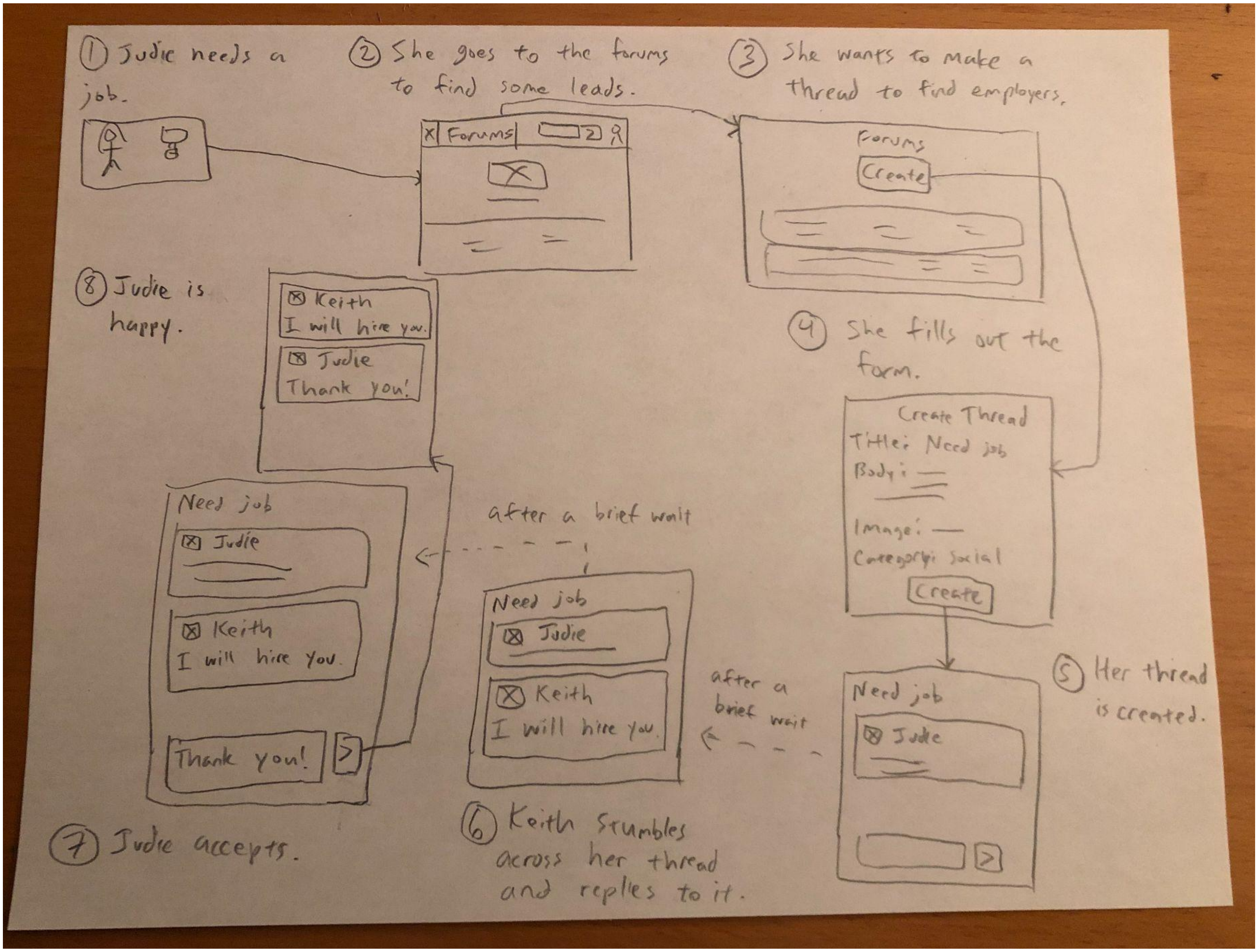
Use Case #6: A Game of Chess
Victor is bored and decides to ask in the Chess group's chat if anyone wants to play some chess. Haruhi accepts the challenge in the group chat.



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.

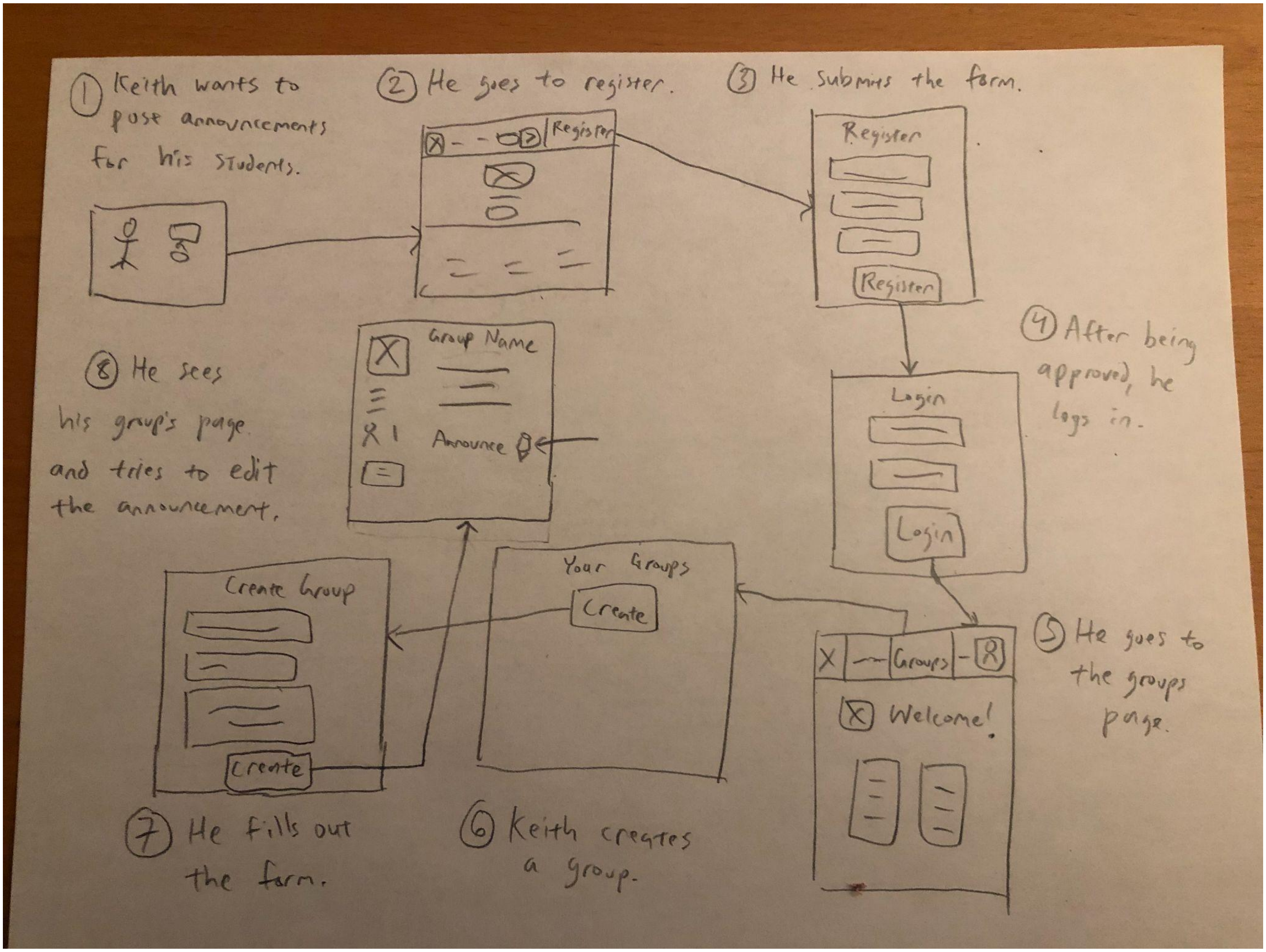


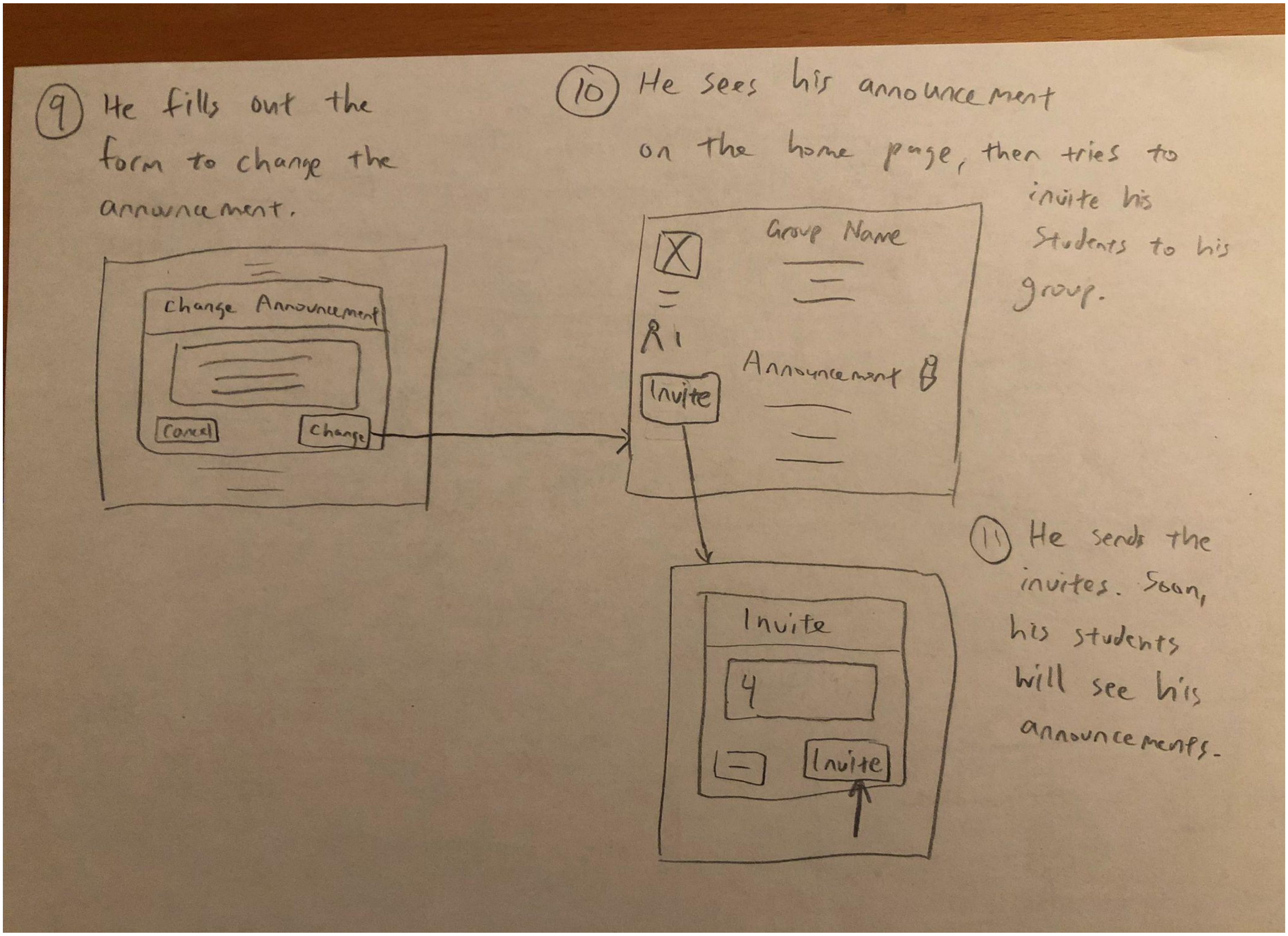
Use Case #8: An Aimless Student
Judie needs a job, therefore she creates a thread in the forums looking for work. Keith sees the thread and replies in the thread saying he can help Judie. Judie is elated.



Use Case #9: A Professor's Pains

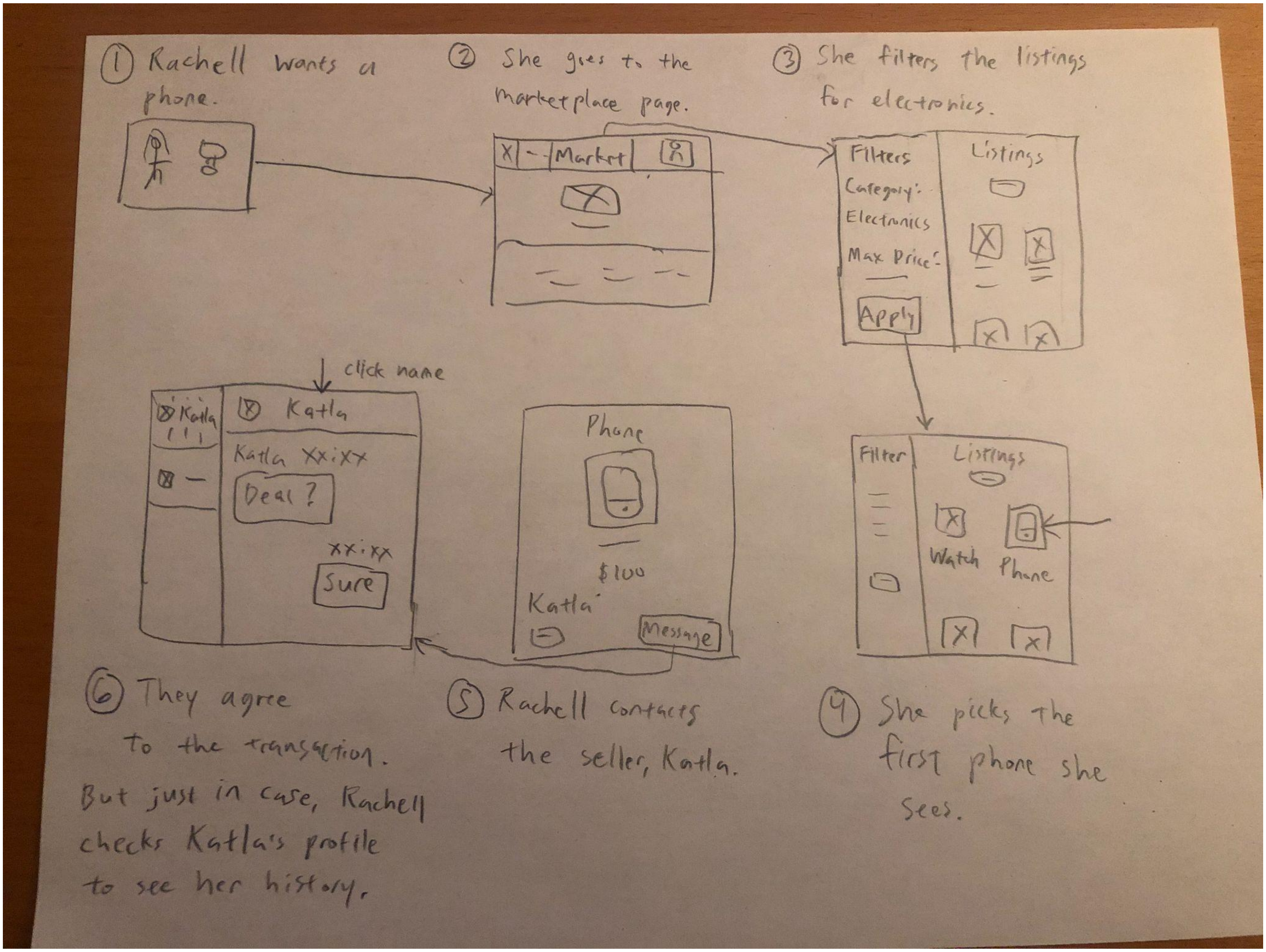
Keith heard Gatormmunity can meet his desire to post an announcement to all of his students at once, hence he registers for an account and logs in after being approved. He creates a group and posts an announcement, and then he invites his students to his group.




9 He fills out the form to change the announcement.
Change Announcement
Cancel
Change
10 He sees his announcement on the home page, then tries to invite his students to his group.
Group Name
Announcement
Invite
11 He sends the invites. Soon, his students will see his announcements.
Invite
4
Invite

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

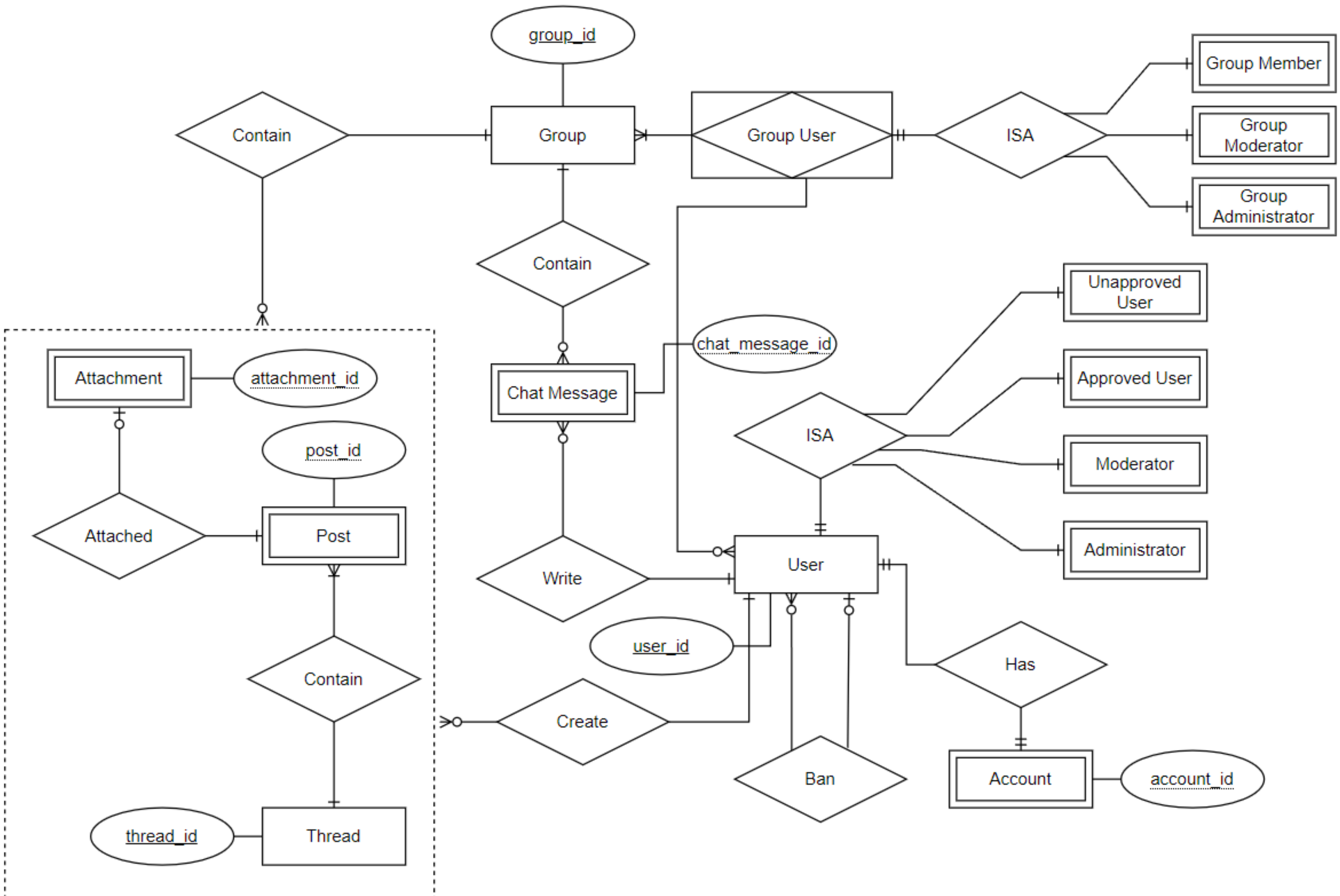
6. Group (Strong)
    6.1. group_id: primary key, numeric
    6.2. name: alphanumeric
    6.3. description: alphanumeric
    6.4. announcement: alphanumeric
    6.5. picture_path: alphanumeric
    6.6. picture_thumbnail_path: alphanumeric
    6.7. creation_date: alphanumeric

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.

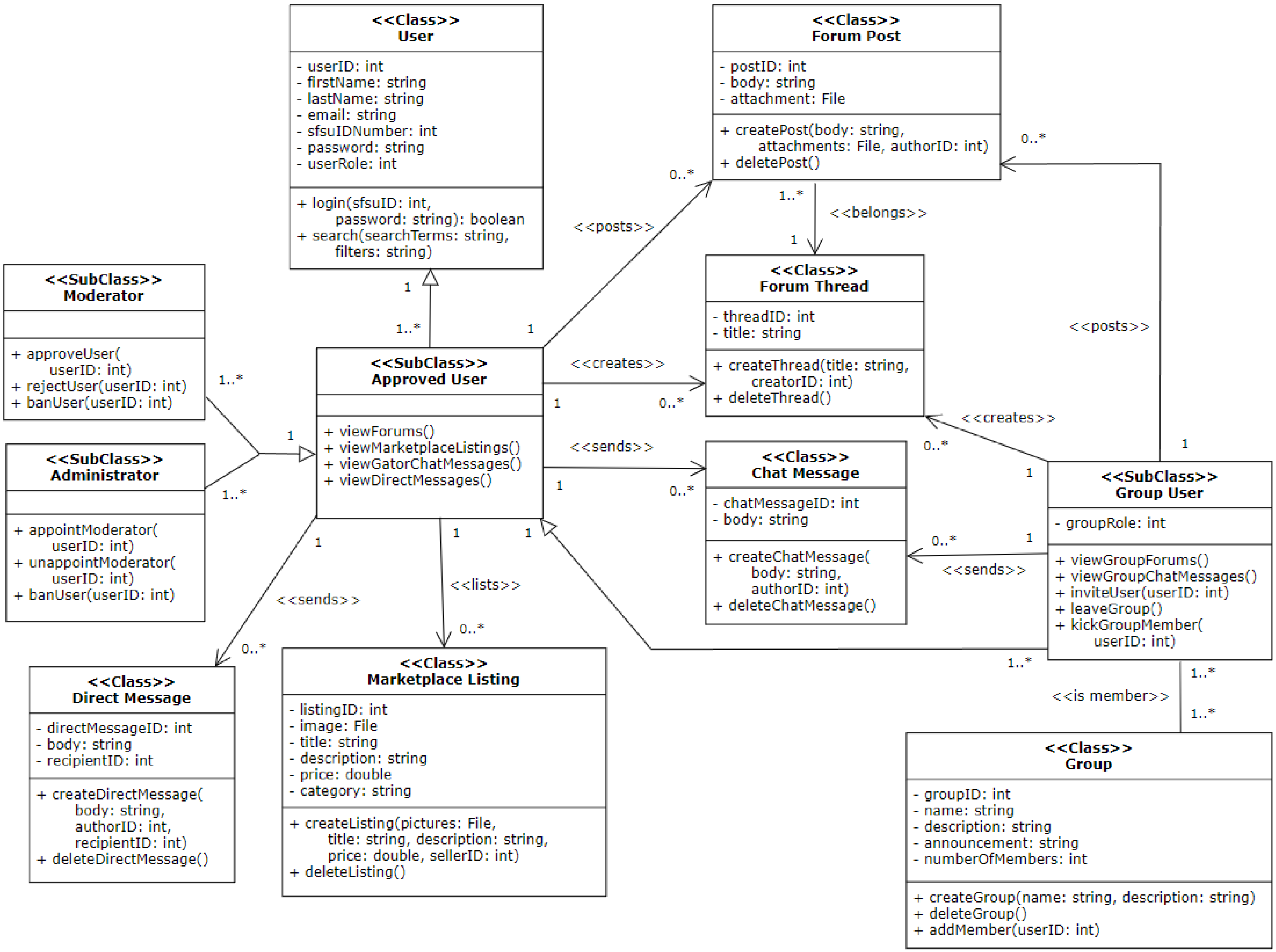
Forum Thread Sort: We may create a sorting algorithm for forum threads so that the user can see forum threads sorted by creation date, last reply date, or by thread title. The creation date and last reply date sort will be from newest to oldest or oldest to newest, while the thread title sort will be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

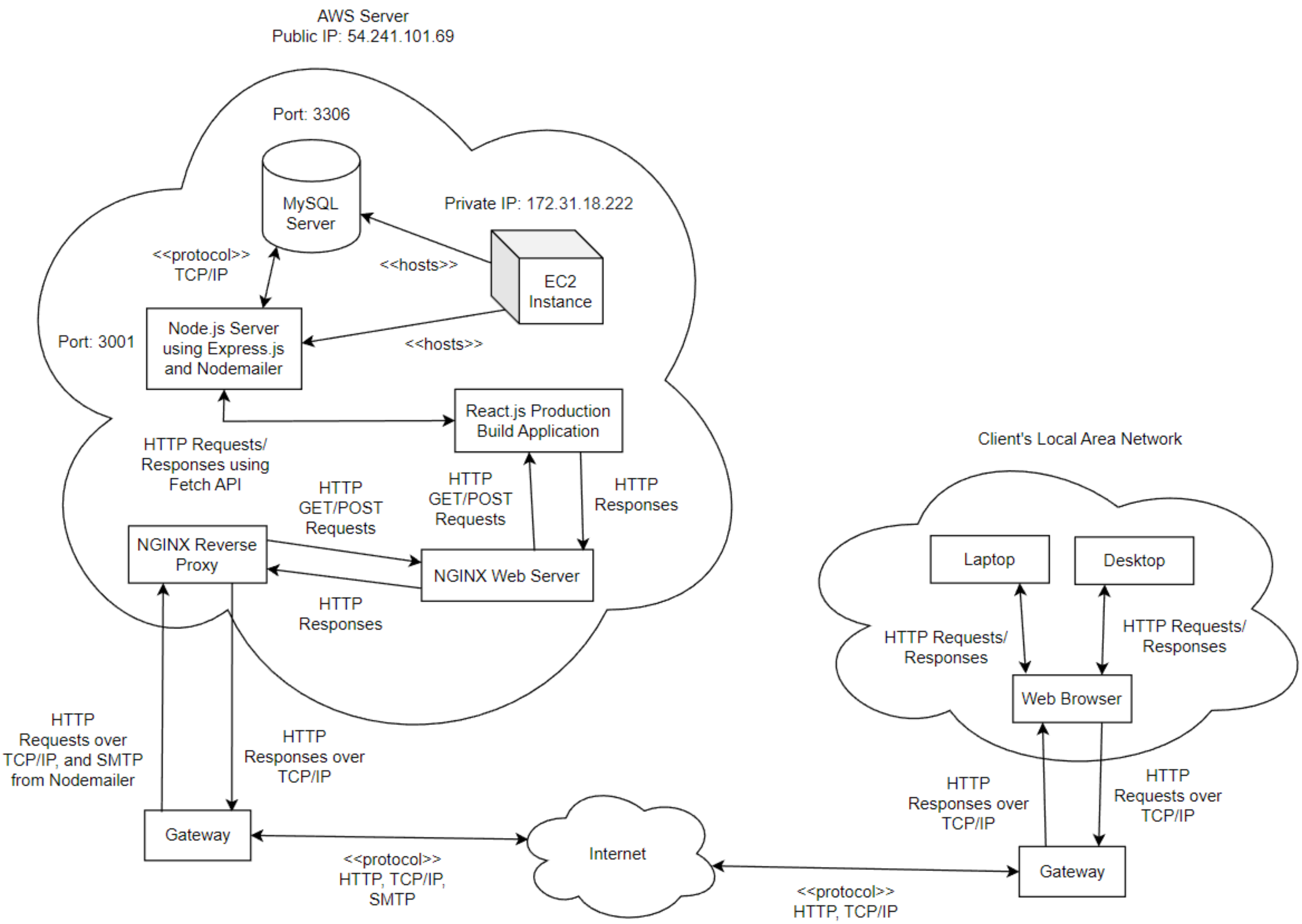
6. High Level UML Diagram

UML Class Diagram:

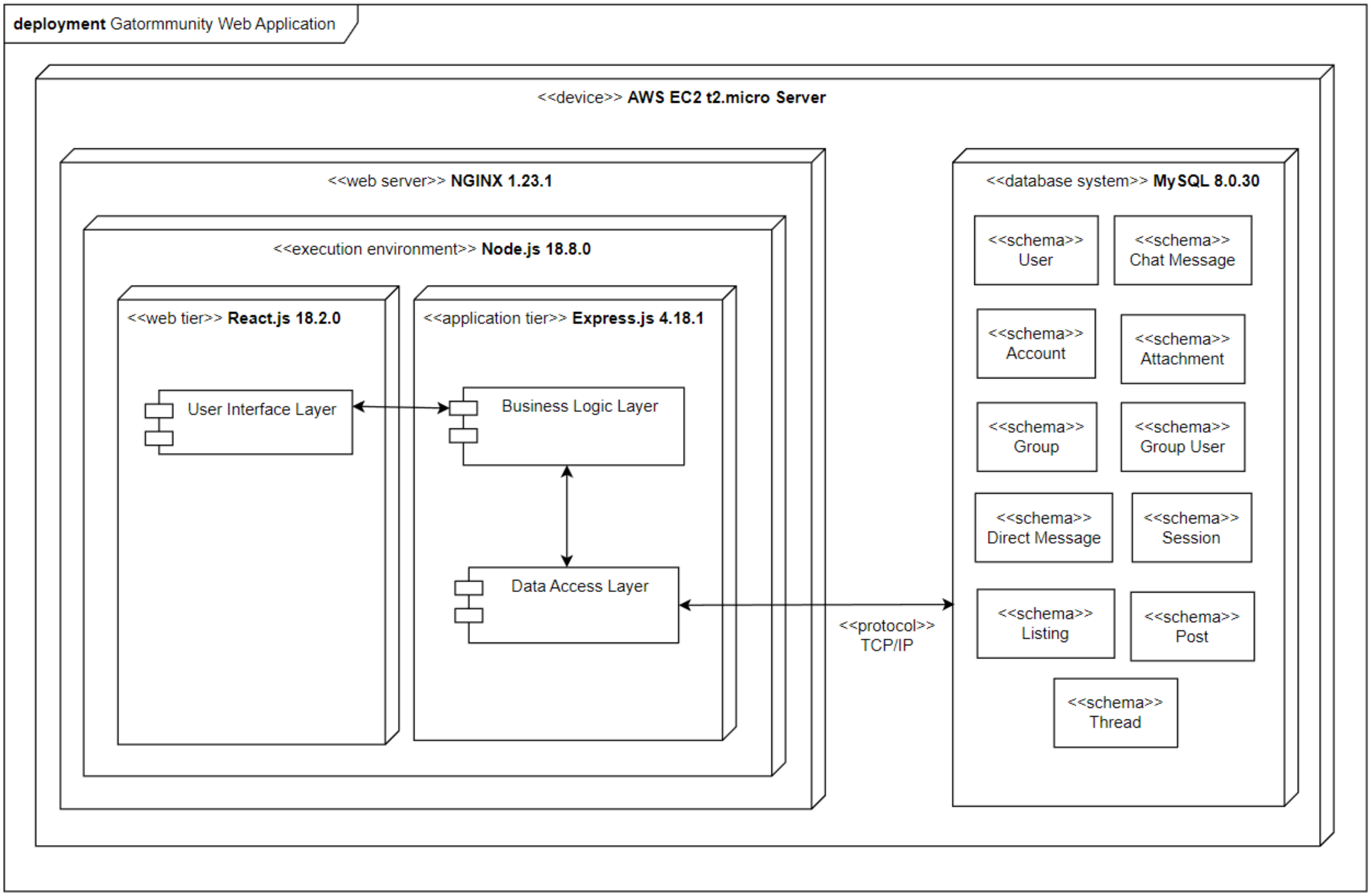


7. High Level Application Network and Deployment Diagrams

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