THE UNIVERSITY OF SYDNEY

The University of Sydney

School of Computer Science

INFO2222 Usability Report

Usability Report for our messaging and sharing platform 3chan

SID: 520627482 SID: 530325873

An Assignment submitted for the UoS:

INFO2222 Computing 2 Usability and Security

May 18, 2024

Table of Contents

Table of Contents

***Abstract*—In this report, we proudly present to you: the chat and forum website we designed – 3chan. 3chan is specifically designed for computer science students, incorporating learnability, efficiency, memorability, error recovery, and user satisfaction. With this web application, computer science students can freely chat with their friends remotely, create chat rooms, discuss learning outcomes, and share their learning experiences on the forum. We have conducted multiple iterations and core user surveys to ensure that our final product meets the needs and experiences of our core users, and is perceivable, operable, understandable, and robust. In addition to the development report, we have included the next steps in the development plan and a prioritized feature list to ensure that our program can be updated and further iterated in real-time.**

I. User Investigation

A. Investigation Setup

We divided our user investigation into two parts: First, a large-scale questionnaire survey to broadly collect user information and requirements, ensuring our application can best fit the market. Second, a small-scale core user consultation survey, where we identified a few typical students as our core users. We conducted multiple rounds of iterative surveys with these core users, and in the initial stage of the user investigation, we also consulted their detailed opinions.

In the large-scale questionnaire survey, we used Google Forms [1] to distribute the questionnaire to computer-related students, social media group chats, academic staff we know, and alumni. We received 26 questionnaire responses (2 from alumni and 3 from uni staff) and conducted PACT analysis and persona construction, extracting some typical user groups. (For some detailed results of the questionnaire survey, please refer to attachments.)

In the small-scale consultation survey, we consulted three typical users and gathered their detailed opinions. We integrated their information to build the persona of the student group.

B. PACT Analysis

To better understand web development and user needs, we first conducted a PACT analysis. The details are as follows:

- People: Student, Alumni, Uni staff, Admin user
- Activity: Chatting, Group Chatting, Sharing Knowledge, Commenting, Managing Posts, Modifying Profiles
- Context: Used in classroom, Used in library, Used in Dormitory, Used to discuss about the lecture content, Used to discuss about the homework, Used to share own knowledge, Used to seek for answers
- Technology: Desktop, Laptop, Mobile Phone

Through PACT, we can better align user understanding with developer understanding, making the website more easily accepted by users.

C. Persona

Student We conducted in-depth interviews with three core student users (outside the group), analyzing the difficulties they experienced and the core features they desired. From their key characteristics, we created a persona of a representative student user named Frank.

Fig. 1. Student Persona

Uni staff We have also consulted a tutor from the university and gathered his opinions. When designing the website, we intend to consider not only the ideas of the students but also the opinions of the staff who use the website. This is because they are also an important target group and are essential members in maintaining the website's ecosystem.

Through surveys and consultations, we have created a persona (anonymous) for this figure, covering the current challenges he faces and the desired features of the website.

D. Investigation Outcome

Based on the survey, PACT, persona, and individual consultations, we have compiled and analyzed the data to draw the following conclusions according to the users' needs. These conclusions will assist us in the subsequent card sorting design, wireframe, and the first round of program development.

1. The primary users are **students** (80.8%).
2. Most users prefer **Blue & Black** (41.7%).
3. Students from different disciplines desire their own dedicated forums.

Fig. 2. Uni Staff Persona

4. The top three features users want are: **Post** (69%), **Chat** (54%), and **Search** (42%).
5. 64% of users support the addition of a **Group Chat** feature.
6. Student users prefer to see if their friends are online
7. Staff users prefer to have a visually appealing and user-friendly platform

II. Card Sorting

A. Card Sorting Setup

After collecting sufficient investigation data, we organized an online card sorting session with some important target users. During the session, we used Zoom to conduct the meeting and Optimal Workshop [2] to gather user data and organize it.

During the session, we encouraged users to share their thoughts while sorting and held a discussion after the sorting was completed. In this discussion, we gained insights into users' ideas and understanding of our website layout and structure, which greatly assisted in the creation of the wireframe.

B. Card Sorting Process

We have attached some screenshots of our card sorting session in attachment.

And some of the sorting results are also shown in attachment.

C. Card Sorting Outcome

During this card sorting, we are primarily concerned with the following key points:

1. Whether users divide the cards into three or four categories.
2. Whether users can place certain functions, which we have assumed to be grouped together, into the same category.
3. With which cards users will group the more difficult-to-categorize cards such as Admin Support, Setting, Useful Link, etc.

The outcome is availiable in attachment. According to the result, we have produced the site map below.

D. Site Map

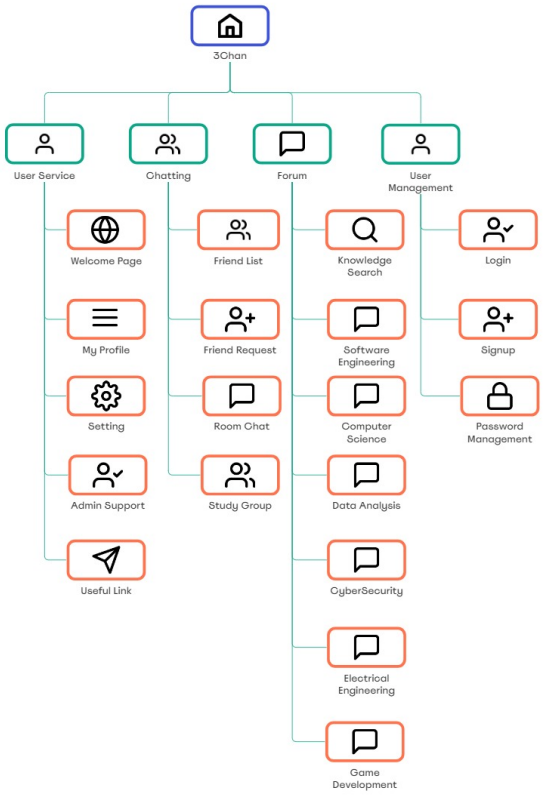

Fig. 3. Site Map

In the site map, we have some main focuses. These focuses are derived mainly from the card sorting session, but also partly from the user investigation.

1. We strive to make the interface structure simple, understandable, and aesthetically pleasing.
2. We divided the page into four main categories: User Service, Chatting, Forum, and User Management.
3. All knowledge repositories and knowledge research are grouped together.

4. All Chatting-related features and Friend-related features are grouped together.
5. Admin Support, Useful Links, and other similar items are categorized together with Settings and Profile.

III. Wireframe

A. Wireframe Design Process

The initial step in our web design process was to categorize the website functionalities based on the results of our Card Sorting Setup. This step was crucial in ensuring that the website structure would be intuitive and user-friendly. We focused on the following categories:

1) **Welcome Page (Login/Signup):** This section includes all features related to user authentication and welcoming new users.
2) **Chat Page:** This page groups together all chatting-related features and friend-related features, ensuring a seamless communication experience.
3) **Forum Page:** All knowledge repositories and research-related features are grouped here, providing a comprehensive knowledge-sharing platform.
4) **Settings:** This section includes admin support, useful links, and information management, making it easy for users to manage their accounts and access helpful resources.

B. Lofi Prototype

After categorizing the functionalities, we proceeded to create a low-fidelity (lofi) frame. The key social points in our design are as follows:

1) **Welcome Page Design:** We designed a technologically appealing and aesthetically pleasing background for the welcome page, which is also used for the login and signup pages. This design aims to make the website visually attractive, thereby increasing user engagement and the desire to use the site.
2) **Consistent Sidebar Navigation:** Both the chat page and the forum page utilize a manageable sidebar, facilitating easy navigation within the website. This consistent navigation structure enhances the user experience by providing a clear and straightforward way to move between different sections.
3) **Distinct Settings Section:** We designed the settings section to be relatively independent, distinguishing between user functionalities and administrative functions. This separation ensures that users can easily manage their settings without confusion.
4) **Chat Page Sidebar:** The chat page includes a sidebar that manages friend and group content, following a logical flow of "want to chat – find someone to chat with – enter chat room." This design aligns with the user's natural chatting process.
5) **Forum Page Organization:** In the forum page, subpages are managed according to different academic disciplines, facilitating easy information categorization and retrieval.

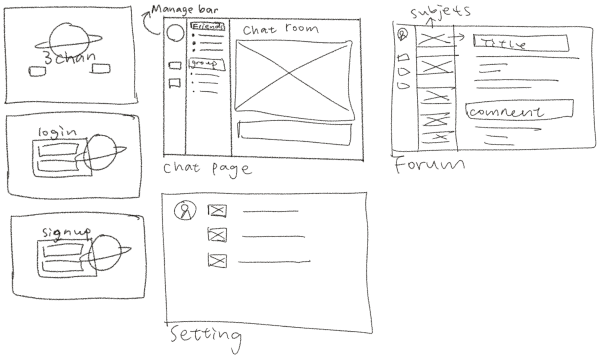

Fig. 4. lofi wireframe

Our lofi design approach aligns with the POUT (Perceivable, Operable, Understandable, Robust) principles as follows:

Perceivable

- **Design Consistency:** By using a visually appealing and consistent design across the welcome, login, and signup pages, we ensure that information and user interface components are perceivable. The sidebar navigation in chat and forum pages further enhances perceptibility.
- **Clear Information Presentation:** The clear categorization in the forum page by academic disciplines ensures that information is easily perceivable to users.

Operable

- **Ease of Navigation:** The use of a sidebar for navigation in the chat and forum pages makes the interface operable. Users can easily navigate the website without encountering barriers that impede their interaction.
- **Logical Flow:** The logical arrangement of the chat page sidebar, which follows the sequence of finding someone to chat with and then entering the chat room, ensures operability.

Understandable

- **Intuitive Design:** The separation of the settings section from other functionalities and the clear categorization in the forum page make the user interface understandable. Users can easily comprehend the structure and operation of the website.
- **User-Friendly Language:** The use of straightforward and user-friendly language throughout the design further enhances understandability.

Robust

- **Technology Compatibility:** Our design ensures that content can be reliably interpreted by a wide variety of user agents, including assistive technologies. This ensures that the website remains accessible as technologies evolve.
- **Future-Proof Design:** By adhering to web standards and best practices, we ensure that the website's content

remains robust and accessible to users with disabilities, meeting their needs now and in the future.

C. First Iteration of Guerrilla Test

The first iteration of our guerrilla testing provided valuable feedback on our low-fidelity design. Here are the key findings and suggestions:

Welcome Page

The welcome page design received positive feedback from users. They appreciated the aesthetically pleasing and technologically appealing background. Users suggested incorporating 3D effects to make it more attractive. Additionally, they recommended extending the technological design elements throughout the entire website, as computer science students generally prefer a cool and tech-savvy look.

Chat Page

Users want a more intuitive way to see if their friends are online. They also suggested adding a feature to manually set their online or offline status, which could serve as a privacy measure through an "invisible mode." The use of dual sidebars to manage different functionalities was well-received, as it made navigation straightforward and efficient.

Forum Page

Users wanted an overview of repository titles for the forum page before delving into detailed subpages. This request aims to provide a quick glance at available content, making it easier for users to find what they are looking for.

Conclusion

The suggestions and comments will guide the next steps in our design process, ensuring that our website meets user expectations and preferences. We will focus on integrating 3D effects and extending the tech-savvy design throughout the website, improving online status visibility and privacy features in the chat page, and providing a clear overview of repository titles in the forum page in our HIFI prototype.

D. Hifi Prototype

Building on the feedback from the guerrilla testing, our high-fidelity wireframe has been designed with enhanced aesthetics and functionality, while the overall structure remains consistent with the low-fidelity version.

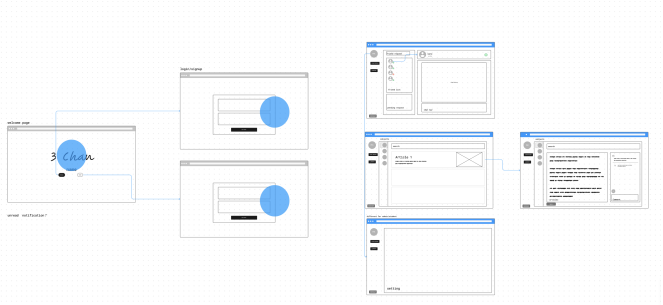

Fig. 5. HiFi wireframe

E. Prioritized List

Based on the results of the first round of guerrilla testing, we compiled and summarized the features that users most wanted to see. We also analyzed which features would form the foundation of the entire program. After extensive discussion and weighing the pros and cons, we developed the following Prioritized List. This list covers the content for both the first iteration and the second iteration, providing guidance for our future development.

1. Basic Chatting Functions
2. Basic Friend Functions
3. Group Chat
4. Basic Setting Functions
5. Forum/Knowledge Repo
6. Delete Friends
7. Save Message When Offline

IV. Incremental Development

A. First Iteration

The first iteration of development lasted a week. After listing the Prioritized List, our team members divided the work and began the first round of iterative development in full swing. The goal of the first round was to complete most of the basic functionalities, giving our webpage a usable prototype, which would facilitate our second round of Guerrilla Testing.

In the first developmental iteration, we primarily achieved the following:

1. An aesthetically pleasing welcome, login, signup, and home interface.
2. Basic chatting functionality and its extensions, as well as basic friend functionality and its extensions (inherited from the Security Part and slightly modified for smoother operation).
3. Users can delete their friends on the home interface.
4. Users can receive messages even when they are offline; these messages will be stored in the message history.
5. Implemented Group Chat, which can accommodate multiple users.
6. Implemented basic Setting functions: Update password, Update username, etc

For details of implementation, please refer to the links in "Basic Funcationalities" part or in "Specific User Functionalities" Part

After the first developmental iteration, we have gathered some of our primary target users and conducted the second Guerrilla Test session (The first Guerrilla Test session has been conducted for improving the wireframe). For the process and outcome of the second Guerrilla Test session, please refer to next part.

B. Second Iteration of Guerrilla Test

We gathered five core users from outside our group for an in-person discussion and Guerrilla Test. Some of these users

had participated in our card sorting, enabling them to provide valuable feedback, laying a solid foundation for the second iteration of development.

We all sat around a round table, using one person's computer connected to a screen to demonstrate our website, and encouraged the users to share their opinions.

Through this in-person Guerrilla Test, we received numerous evaluations and suggestions, many of which will be very helpful for future development. I will list all the suggestions we received and highlight the most important ones, selected based on their importance, workload, and feasibility. These will be incorporated into the second iteration of development to enhance user experience. Additionally, the remaining tasks from the prioritized list will also be addressed in the second iteration.

All suggestions are:

1. Add more prompts on the main interface to make it clearer and guide users through how to use the website.
2. Each input field on the website has a button that must be pressed to submit the input; pressing the Enter key doesn't work, which is inconvenient.
3. Some fonts on the interface are not very visible.
4. Users found a bug on the website: when the username is too long, it exceeds the text box.
5. Users want the ability to freely choose the color theme.
6. Users want the page to automatically refresh and display friend information after adding a friend.
7. There is a button at the bottom of the webpage to open the chat interface; users want to be able to click on a friend's avatar to start a chat.
8. Users want a Group Invitation feature to invite users who haven't joined an already created group.
9. A good navigation flow: sometimes after signing up, it redirects to the home page, and sometimes it redirects to the login page. Users want this to be consistent.

The suggestions we have chosen to add to the second iteration:

1. Add more prompts on the main interface to make it clearer and guide users through how to use the website.
2. Each input field on the website has a button that must be pressed to submit the input; pressing the Enter key doesn't work, which is inconvenient.
3. Some fonts on the interface are not very visible.
4. Users found a bug on the website: when the username is too long, it exceeds the text box.
5. A good navigation flow: sometimes after signing up, it redirects to the home page, and sometimes it redirects to the login page. Users want this to be consistent.

The other suggestions are not included in the second iteration considering the workload and time constraints. For example, "Users want the ability to freely choose the color theme" is a major feature that requires a significant amount of time to implement. Given the number of pre-existing tasks that need to be completed in the second iteration, we do not have the time to implement these additional features. Therefore, these tasks are placed at the bottom of the prioritized list. Due to time constraints, we cannot address these tasks now, but we will add them to the future prioritized list.

The suggestions we will put into the future prioritized list:

1. Users want the ability to freely choose the color theme.
2. Users want the page to automatically refresh and display friend information after adding a friend.
3. There is a button at the bottom of the webpage to open the chat interface; users want to be able to click on a friend's avatar to start a chat.
4. Users want a Group Invitation feature to invite users who haven't joined an already created group.

Regarding the improvements and revisions made to some features after the second guerrilla test, please refer to this part.

C. First Iteration Analysis

During the development of the first iteration, we achieved most of the basic functions and basically met the goals set before the development. However, there were still some flaws and shortcomings that need to be improved in the second iteration. These shortcomings include:

1. The development progress only met expectations but did not exceed them. While the basic functions of Chatting, Friend List, and Setting were mostly completed, the Knowledge Repo (forum) part was too complex and has not yet been started. Therefore, the development progress can only be described as meeting the requirements but not as satisfactory.
2. During the development process, we worked in isolation without continuous communication with users. Because we were detached from the user base and user experience, we developed and designed pages based solely on our understanding (with no further research after the first iteration guerrilla test). This led to many minor defects being discovered during the second iteration guerrilla test, such as not having a good navigation flow.

However, there are also many aspects worth carrying over to the second round of development from the first round, such as:

1. Just as we did before starting the first round, we held a meeting within our team before starting the second round to discuss the strengths and weaknesses of the first round. We then added user suggestions to the prioritized list, re-divided tasks, and set timelines.
2. During the first iteration, our development team members frequently communicated through various channels to discuss their development progress and next steps. Based on our experience, this significantly accelerated our development process and ensured consistency in the features developed. It also made the page framework clearer and allowed team members to collaborate more effectively.

D. Second Iteration

In the second iteration, we carefully analyzed and selected important tasks that could be completed within a week and began development. The ultimate goal is to complete the entire website, making it fully functional for users. Some improvement features, such as ”clicking on a friend’s avatar to open the chat interface,” were not included in this iteration because we deemed it unlikely to be completed within a week. Therefore, these tasks will be added to the future prioritized list for subsequent development plans. In summary, the tasks included in the second iteration development plan are:

1. Displaying the online status of friends
2. Categorizing users into three types: student, academic, and admin, with adjustments possible via invitation codes
3. Developing the Knowledge Repository page
4. Enabling users to post messages
5. Enabling users to comment on posts
6. Allowing admins to delete comments and posts
7. Allowing admins to ban other users
8. Enhancing the setting features
9. Improve existing functions based on the second iteration of Guerrilla Test

These features were perfectly completed before the end of the second iteration. However, some extended functionalities set to be implemented before the first iteration, such as the ”admin support” page for contacting admins online, were not completed in time. Therefore, these tasks have been added to the future Prioritized List.

E. Future Prioritized List

Due to limited task time, we were unable to complete all the initial plans. Therefore, the additional unfinished features have been added to the future prioritized list.

The contents of the future prioritized list include:

1. Useful Link Page
2. Admin Support
3. Knowledge Search
4. Allowing users to choose the website color scheme
5. Automatically refreshing
6. Clicking on a friend’s avatar to start a chat
7. Group invitation
8. Voting functionality
9. Sending emojis

V. Basic Functionalities

A. Overview of the website

Refer to All Screenshots

B. BASIC CHATTING FUNCTIONS

Most of the Chatting functionality inherits from the Security module. Here, I will re-describe all the details related to Chatting:

Joining a Room

When a user selects a friend and clicks Chat, a request to join the room is sent to the server. If the friend is already in a room, the server will have the user join the same room (which can accommodate up to two people). If the friend is not in a room, the server will create a new room and let the user join. Messages sent by the user will not reach the friend until the friend joins the room.

Sending Messages

When a user sends a message, it is encrypted with the recipient’s public key and sent to the server. The server does not process the message but directly forwards the encrypted message to the recipient. The recipient can then decrypt the message using their private key to read it.

Checking if the Friend is in the Room

The server uses a class to record the mapping of rooms to users. If a user sends a message to another user who is not in the room, the server will notify them that the friend is offline.

Message History

When a user sends or receives a message, it is re-encrypted with the user’s public key and stored on the server (making it unreadable by the server). If the user leaves the room and later rejoins, they can request the message history from the server.

C. BASIC FRIEND FUNCTIONS

The following functions in the `friend` module manage the basic operations for handling friendships, and their specific operations are described in detail below:

`request_friend(user1, user2)`

- This function handles the process of user `user1` sending a friend request to user `user2`.
- First, it searches the database for `user1` and `user2` to ensure they both exist.
- It checks if there is already a friendship between the two users. If not, it creates a new friend request record with the status ”requested”.
- If there is an existing friendship with the status ”rejected”, it updates the status to ”requested”. If the status is ”accepted”, it returns `False` indicating the request cannot be sent again.
- The database changes are committed, and it returns `True` indicating the friend request was successfully sent.

`accept_friend(user1, user2)`

- This function handles the process of user `user1` accepting a friend request from user `user2`.
- First, it searches the database for `user1` and `user2` to ensure they both exist.
- It checks if there is an existing friend request with the status ”requested” between the two users. If found, it updates the status to ”accepted”; otherwise, it creates a new friendship record with the status ”accepted”.
- It also checks for a reverse friend request (i.e., `user2` requesting `user1`) and handles it similarly.
- The database changes are committed, and it returns `True` indicating the friend request was successfully accepted.

rejected_friend(user1, user2)

- This function handles the process of user `user1` rejecting a friend request from user `user2`.
- First, it searches the database for `user1` and `user2` to ensure they both exist.
- It checks if there is an existing friend request with the status "requested" between the two users. If found, it updates the status to "rejected".
- It also checks for a reverse friend request (i.e., `user2` requesting `user1`) and handles it similarly.
- The database changes are committed, and it returns `True` indicating the friend request was successfully rejected.

get_all_friends_of_current_user(username)

- This function returns a list of all friends of the current user.
- It first calls the `remove_duplicate_friendships` function to ensure there are no duplicate friendship records.
- It queries all accepted friendship records where `user1` is the current user and returns the list of friends.

get_all_request(username)

- This function returns a list of all friend requests received by the current user.
- It first calls the `remove_duplicate_friendships` function to ensure there are no duplicate friendship records.
- It queries all friend requests with the status "requested" where `user2` is the current user and returns the list of requests.

get_all_request_to_sender(username)

- This function returns a list of all friend requests sent by the current user.
- It first calls the `remove_duplicate_friendships` function to ensure there are no duplicate friendship records.
- It queries all friend requests with the status "requested" where `user1` is the current user and returns the list of requests.

remove_duplicate_friendships()

- This function is used to remove duplicate friendship records from the database.
- It queries all friendship records and orders them by `user1`, `user2`, and `status`.
- It uses a set `seen` to keep track of processed records, finds duplicate records, and deletes them.
- The database changes are committed to ensure there are no duplicate friendship records.

Logical Flow and Usability Analysis

The logical flow of the friend management functions is structured to ensure a coherent user interaction process. Initially, users issue friend requests via the `request_friend` function. Subsequently, recipients of these requests can either accept them using the `accept_friend` function or reject them with the `reject_friend` function. Once a friendship is established, users have the ability to terminate it through the `delete_friend` function. For managing their social connections, users can retrieve a comprehensive list of their friends using the `get_all_friends_of_current_user` function, examine incoming friend requests via the `get_all_request` function, and inspect the requests they have sent using the `get_all_request_to_sender` function. Additionally, the `remove_duplicate_friendships` function plays a crucial role in maintaining database integrity by eliminating duplicate records in friendships.

D. DELETING FRIENDS

The `delete_friend` function handles the process of deleting an existing friendship between two users:

delete_friend(user1, user2)

- This function handles the process of user `user1` deleting the friendship with user `user2`.
- First, it searches the database for `user1` and `user2` to ensure they both exist.
- It searches for all friendship records between the two users (regardless of whether `user1` is the initiator or the recipient) and deletes these records.
- The database changes are committed, and it returns `True` indicating the friendship was successfully deleted.

E. SAVE MESSAGE EVEN OFFLINE

Even if a user was offline in a group chat, messages sent by others will be properly stored on the server and displayed to the user when they log back in.

In our program logic, each room number in a group chat is unique. This allows us to allocate a separate data storage space on the server for each room to store relevant information. We record the chat history in JSON format in this storage space. When a user re-enters the room, we verify their session. If they indeed belong to the room, we resend the historical messages to them so they can view the chat history.

F. GROUP CHAT

Data Models

- **`GroupList` Class**:
  - The `GroupList` class represents the groups in the system.
  - It contains an `id` field, which is the primary key, and a relationship with the `GroupMember` class through the `members` attribute.
- **`GroupMember` Class**:
  - The `GroupMember` class represents the members of a group.
  - It contains fields such as `id` (primary key), `group_id` (a foreign key referencing `GroupList`), `user_username` (a foreign key referencing `User`), and `status` (an enum indicating whether the member has joined or is invited).
  - Relationships are established with both `GroupList` and `User` classes.

G. IMPROVEMENT AFTER SECOND GUERRILLA TEST

First, we added introductions and descriptions of how to use various features. This allows users to easily understand how to navigate our website, providing them with a smooth user experience.

Secondly, we added an "ENTER" key event listener for important components. For example, when adding a friend, after entering the name, the user can simply press the ENTER key to send the add friend request to the server, without needing to click the add button manually. This design aligns better with global web functionality standards and makes the user experience more effortless.

Thirdly, we replaced the previously used decorative fonts with more readable ones, as the decorative fonts might have made it difficult for users with smaller screens to read.

Fourthly, we moved the display of usernames from the left border to the main central area. First, this ensures that regardless of the length of the username, it will automatically wrap and display completely. Second, placing the username in a more central location makes it more prominent, creating a clearer and more logical layout. Third, removing the username display from the left border makes the entire webpage look more aesthetically pleasing.

Finallly, we changed the page flow to make it more logical and in line with user expectations. Previously, after completing registration, the page would redirect to either the login interface or the logged-in user interface, depending on the session's existence. This caused inconsistent page transitions. Therefore, we adjusted it so that after completing the signup process, regardless of the information stored in the cookies, the page will always redirect to the login page.

H. DISPLAYING FRIENDS' ONLINE STATUS

The **user online** functionality is a crucial part of the user experience, providing visibility into which users are currently active on the platform. This feature is implemented through a combination of SQLAlchemy models, database interactions, and Flask routes. The following sections detail the various components and their interactions:

Data Models

- **`OnlineUser` Class:**
  - The `OnlineUser` class is used to store the online status of users.
  - It contains fields such as `username` (a foreign key referencing `User`) and `is_online` (a boolean indicating whether the user is online).
  - This class has a relationship with the `User` class, enabling easy access to user details.

Database Functions

- **`set_user_online_status`:**
  - This function sets the online status of a user.
  - It queries the `OnlineUser` table for the specified username. If found, it updates the `is_online` field; otherwise, it creates a new entry.
  - After making changes, it commits the transaction to the database.
- **`get_user_online_status`:**
  - This function retrieves the online status of a user.
  - It queries the `OnlineUser` table for the specified username and returns the `is_online` value. If no entry is found, it returns `False`.
- **`get_users_online_status`:**
  - This function retrieves the online status of multiple users.
  - It queries the `OnlineUser` table for a list of usernames and returns a list of their online statuses.
  - It constructs a dictionary mapping usernames to their online statuses and returns a list corresponding to the input list of usernames.

User Interface Integration

- **Friend List:**
  - The online status of friends is displayed in the friend list.
  - Each friend's online status is retrieved using the `get_users_online_status` function and displayed next to their name.
  - The status is shown as "online" or "offline" with appropriate icons and styles.
- **User Status Control:**
  - Users can control their visibility with the online/offline buttons.
  - Instructions and buttons for setting online and offline statuses are provided, allowing users to manage their presence.

Logical Flow

The logical flow for the online status functionality is as follows:

- When a user logs in, their online status is set to true using the `set_user_online_status` function.
- The user's friends can see their online status, which is retrieved using the `get_users_online_status` function and displayed in the friend list.
- Users can manually set their status to online or offline through the buttons provided in the UI.
- The `set_user_online_status` function updates the status in the database, ensuring it reflects the user's current choice.
- The system ensures the status is accurately updated and displayed, providing real-time feedback to users and their friends.

I. BASIC KNOWLEDGE REPO FUNCTIONS

The following functions in the `app.py` module handle the basic operations for managing forum posts and comments, and their specific operations are described in detail below:

`forum()`

- This function handles requests for accessing the forum.

- It first retrieves the current user's session information. If the user is not logged in, it prompts them to re-log in.
- It fetches the current user's type and gender from the database, and the category from the request parameters (defaulting to `default_category`).
- It retrieves all post titles for the category from the `knowledge_repo` along with the current post ID, content, sender, time, and comments.
- It renders the `forum.jinja` template, passing user information, post titles, content, etc.

`get_post_content()`

- This function handles requests for specific post content.
- It retrieves the category and post ID from the request parameters.
- It fetches the post's content, sender, time, and comments from the `knowledge_repo`, returning these as HTML.

`create_post()`

- This function handles requests for creating new posts.
- It checks if the user is logged in and not banned.
- It retrieves the post's category, content, and title from the request, adds these to the `knowledge_repo`, and returns the result.

`add_comment()`

- This function handles requests for adding comments to posts.
- It checks if the user is logged in and not banned.
- It retrieves the post's category, post ID, and comment content from the request, adds the comment to the specified post in the `knowledge_repo`, and returns the result.

`delete_comment(category, post_id, comment_id)`

- This function handles the deletion of specified comments from posts in the `knowledge_repo`.
- It checks if the specified category file exists; if not, it returns an error.
- It finds the specified post and removes the specified comment from its comment list, then saves the changes.
- It returns the result indicating success or failure.

`modify_post()`

- This function handles requests for modifying posts.
- It checks if the user is logged in and not banned.
- It ensures the user has the necessary permissions (either as an admin or the post owner).
- It modifies the specified post's content in the `knowledge_repo`, updates the post's time, and returns the result.

`delete_post(category, post_id)`

- This function handles the deletion of specified posts from the `knowledge_repo`.
- It checks if the specified category file exists; if not, it returns an error.
- It removes the specified post from the file and saves the changes.
- It returns the result indicating success or failure.

Logical Flow

The logical flow of these `forum` functions works as follows: Users access the forum page via the `forum()` function, which displays the forum interface and a list of posts. Users can create new posts through the `create_post()` function, adding posts to the `knowledge_repo`. Users can view detailed content of specific posts by calling the `get_post_content()` function to fetch and display the post content and comments. They can add comments to posts via the `add_comment()` function, with comments being added to the specified post in the `knowledge_repo`. Users with appropriate permissions can delete posts and comments using the `delete_post()` and `delete_comment()` functions, respectively. Additionally, authorized users can modify post content through the `modify_post()` function.

J. USER BAN FUNCTIONS

The **ban** functionality allows administrators to ban or unban users. The ban system is implemented through a combination of SQLAlchemy models, database interactions, and Flask routes. The following sections detail the various components and their interactions:

Data Models

- **Banuser Class**:
  - The `Banuser` class is used to store ban status information.
  - It contains fields like `username` (a foreign key referencing `User`) and `is_ban` (a boolean indicating the ban status).

Database Functions

`set_user_ban()`

- This function sets the ban status of a user.
- It queries the `Banuser` table for the specified username. If found, it updates the `is_ban` field; otherwise, it creates a new entry.
- After making changes, it commits the transaction to the database.

`get_user_ban_status()`

- This function retrieves the ban status of a user.
- It queries the `Banuser` table for the specified username and returns the `is_ban` value. If no entry is found, it returns `False`.

Logical Flow

The logical flow for the ban functionality is as follows:

1. An admin accesses the settings page through the `setting()` route, which displays the current user settings and provides options for banning/unbanning users. 2. To ban or unban a user, the admin inputs the username and selects the desired action (ban or unban), triggering the `set_ban()` route. 3. The `set_ban()` route calls `set_user_ban()` to update the user's ban status in the database. 4. The system verifies the current session and admin token before making changes, ensuring only authorized actions are performed. 5. The result of the ban/unban action is returned to the admin, confirming the operation's success or failure.

VI. Specific User Functionalities

A. VISUALLY PLEASING PAGES

Creating a visually pleasing and engaging user experience was a key goal in the design of our website. To achieve this, we incorporated several design elements and tools that enhance the overall aesthetic and functionality of the site.

Spline Integration

On the welcome, login, and signup pages, we utilized the Spline website's plugin to incorporate 3D modeling into our design. Through Spline, we crafted a transparent, futuristic planet as a central visual element. This planet, rendered in shades of blue, serves as a captivating focal point that aligns with our theme of technology and innovation.

Consistent Color Scheme

To maintain visual consistency across the site, we adopted a color palette dominated by blue hues. This choice was informed by user research, indicating a preference for blue in technology-themed designs. The consistent use of this color scheme across all pages helps create a cohesive and professional appearance.

Typography and Font Selection

We selected fonts with a playful yet modern feel to complement the technological theme of the site. The chosen fonts, with their game-like design, contribute to a relaxed and enjoyable browsing experience. This careful selection of typography enhances the visual appeal while maintaining readability and clarity.

Neon Effects and Interactivity

To further enrich the user experience, we added neon blue effects to buttons and text boxes throughout the site. Buttons feature additional hover effects, where their background color changes to light blue, and a glowing shadow effect is applied. These interactive elements not only improve the aesthetic quality but also provide users with visual feedback, making the interface more engaging and dynamic.

- **Buttons:** Buttons on the site have a transparent background with blue neon borders. When hovered over, they change color to light blue and exhibit a glowing effect. This interactive feedback enhances user engagement and reinforces the technological theme.
- **Text Boxes:** Text boxes are styled with a transparent background and blue neon borders, ensuring they are visually consistent with the rest of the design. Placeholder text is also styled in blue, maintaining the color scheme even in the absence of user input.

Futuristic and Tech-Savvy Design

The overall design of the site is intended to evoke a sense of futuristic technology. This is achieved through the use of modern, sleek visual elements, interactive neon effects, and a coherent color scheme. The design aims to create a lighthearted yet sophisticated atmosphere, inviting users to explore and interact with the site comfortably.

- **Backgrounds and Layouts:** The use of dark backgrounds with glowing blue elements helps to create a visually appealing contrast, enhancing the readability of text and the prominence of interactive elements. The layout is designed to be intuitive and user-friendly, ensuring a smooth navigation experience.
- **Visual Consistency:** Consistency in design elements across all pages helps to reinforce the site's brand identity. Users can easily recognize and familiarize themselves with the interface, improving their overall experience.

B. BASIC SETTING FUNCTIONS

In the Settings page, users can perform the following actions:

1. Change gender
2. Change user type (requires an access token)
3. Change major
4. Ban other users (if they are an admin)

Changing gender is managed by update_user_gender() in app.py, We recognize that users have the right to define their own gender, so they can freely enter a new gender in the input field in the settings. The entered gender will be updated and stored in the database.

Changing major is managed by update_user_major() in app.py. It has similar processing logic to "Changing gender," but with a difference: this time, we provide users with only six options (five computer science majors and one "secret" option). Users can choose from these options, and the newly submitted major will be updated and stored in the database.

For **Changing user type**, please refer to the section User Type Management

C. USER TYPE MANAGEMENT

Our program has three user types: student, staff, and admin. The difference between student and staff is only in the user type displayed to others. Admins, in addition to using the basic features, can mute other users and delete posts.

When managing user types, we need to restrict general users from changing their own user type. Therefore, we have implemented a unique access token mechanism.

In the settings interface, if a user wants to update their user type, they will be required to enter a token. This token can only be obtained by contacting an admin. During actual operation, the token will be set to 128 bits, regularly changed, and randomly generated, ensuring high security. It can only be obtained by contacting the website admin. Currently, we have simply set it to "admin" for testing purposes.

D. COMPILATION OF SPECIFIC USER FUNCTIONALITIES

In this section, I have compiled all the user-specific functions we implemented. Most of these features were derived from the opinions of core users collected through research, consultations, surveys, and guerrilla testing. Below is the list of these user-specific functions:

1. Users can define their own gender.
2. Users can change their major.

3. Users can see friends' gender, type, and major on the homepage.
4. Forum posts display the time of publication.
5. Attractive interface design.
6. Users can choose to appear offline.
7. Key input fields allow users to press Enter to execute functions.
8. A unique access token mechanism for changing user types.

E. All Screenshots

welcome page
signup page
login page
home page
chatroom
group chat
forum
forum
setting

VII. Final Usability Analysis

A. Perceivable

- **Design Consistency:** We have kept the design consistent with look-and-feel across the welcome page, login page, and signup page, so that anything on these information pages or elements of the user interface is perceivable. The stable sidebar navigation in chat and forum pages gives an added enhancement to the above feature so that the layout is also predictable.
- **Ease in Information Presentation:** The categorization that has been made in the forum page, by academic disciplines, makes easier for users to perceive the information. Different fonts, font colors, and layout structures have aided users who need full access to all information for perusal.
- **Contrast and Text Size:** Sufficient color contrasts of text and background, with provision for varying text size, allow those with visual impairments to be able to see content more clearly.

B. Operable

- **Ease of Navigating:** Potential for side-bar navigating mechanism for chat and forum pages enables operability in the interface. This means users can easily navigate websites without being physically blocked by any barriers that would otherwise stop them from navigating. It also facilitates the navigation of pages by keyboard so that even users who cannot use a mouse are also able to interact with the site.
- **Logical Flow:** The arrangement of elements from the searching of someone to chat with right up to entering the room in the chat page sidebar provides an operable process. In design, the structure of the forum page easily goes along with browsing categories and posts.
- **Interactive elements:** Buttons, selection, and other interactive elements are given designs that with such operability that they do not have accidental hits. Spacing around common touch targets should be large enough to minimize accidental interaction.
- **Error Prevention and Recovery:** The site should provide clear instruction and feedback for form input. It has mechanisms for error prevention and recovery so as to facilitate more effective usage of the website. For example, form validation and error messages allow users to correct mistakes before submission.

C. Understandable

- **Intuitive Design:** In the forum page, the settings functionality is separated from the rest of the functionalities, allowing users to clearly understand where to look for settings. The way the site is used and its form of operation is very lucid and understandable to users. Consistent use of icons and labels throughout the interface enhances this understandable feature.
- **User-Friendly Language:** The use of simple and user-friendly language in the design makes it more understandable to users. The instructions and feedback are in plain language and, therefore, very simple to understand and easy to act on.
- **Predictable Behavior** Making elements like navigation or any other interactivity predictable helps the users develop a mental model of how the site is functioning little by little. This would reduce the cognitive load on them and, for sure, enhance user experience.
- **Unambiguous:** Commands for complex interactions, such as how to join a group chat or how to do something with your settings, are clear and not confusing.

D. Robust

- **Future-proof:** Meeting web standards and best practice will mean the content on the website is going to be robust and available now and into the future. This is set to be backed up by the fact that regular updates and maintenance should be done to ensure that any newly arising issues of accessibility are properly addressed.
- **Responsive Design:** This is responsive about working in all the devices and screen sizes, due to which the website becomes robust. These websites can be accessed over desktop, tablet, and mobile devices without even losing the functionality and accessibility.
- **Error manage:** It should be designed in respect to points of failure in such a way that even if input is taken incorrectly or navigation mistakes are made, it does the website in the right way. It assures users of not wrecking the overall UX from use with some temporary errors through clear error messages and possible suggestions for correction.

VIII. Self Evaluation

Our website development has been a great success, with our team working smoothly and collaboratively throughout

the process. We actively considered user feedback at every stage, ensuring that the features we implemented were aligned with the needs and preferences of our core user base. From research and consultations to surveys and guerrilla testing, we meticulously gathered and analyzed user input to guide our development decisions. This approach not only improved the usability of our website but also ensured that the user experience was intuitive and satisfying. Our collaborative efforts and open communication channels within the team significantly accelerated our development progress, resulting in a consistent and coherent product.

In addition to prioritizing usability, we placed a strong emphasis on security. We incorporated robust security measures, such as encryption for messaging and a unique access token mechanism for changing user types, to protect user data and maintain privacy. Our dual focus on usability and security means that our website is both user-friendly and secure, providing a reliable platform for our users. Overall, the success of our development project is a testament to the effective teamwork, rigorous user-centric approach, and diligent attention to security that characterized our efforts.

Task allocation among members:

520627482:

1. User investigation
2. Card sorting
3. Site-Map
4. Prioritized List
5. Basic Chatting
6. Study Group
7. Save message even offline
8. User type management
9. Setting
10. Improvement based on second Guerrilla Test
11. Report formatting

530325873:

1. User investigation
2. Card sorting
3. Wireframe
4. Welcome Page design
5. Home Page design
6. Navigation design
7. Basic Friend Functions
8. Delete Friends
9. Study Group
10. Knowledge Repo/Forum
11. Ban user
12. Setting
13. Display Online Status

IX. Attachments

Here are the attached images

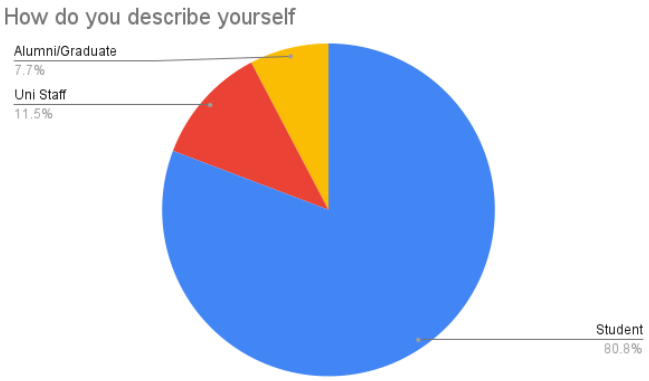

Fig. 6. Questionnaire Result

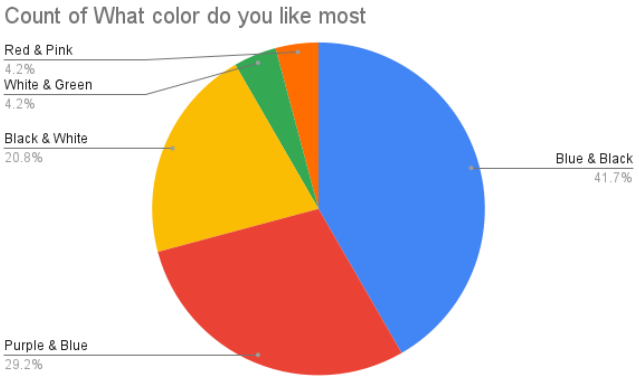

Fig. 7. Questionnaire Result

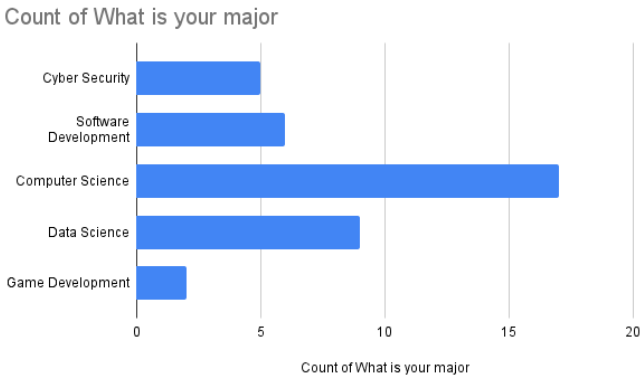

Fig. 8. Questionnaire Result

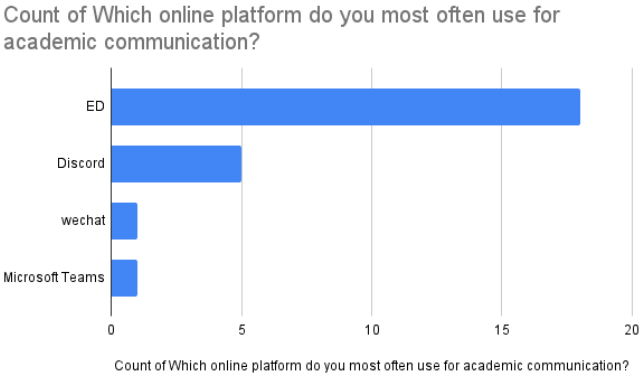

Fig. 9. Questionnaire Result

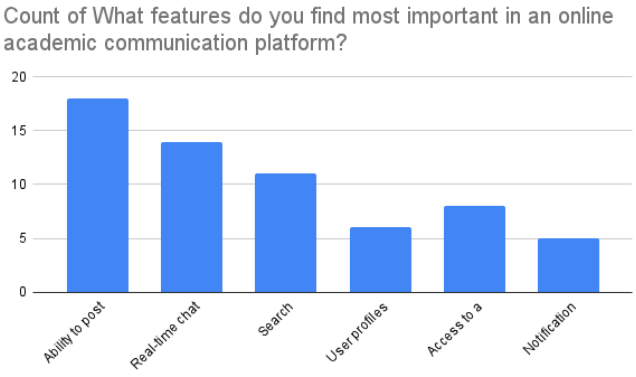

Fig. 10. Questionnaire Result

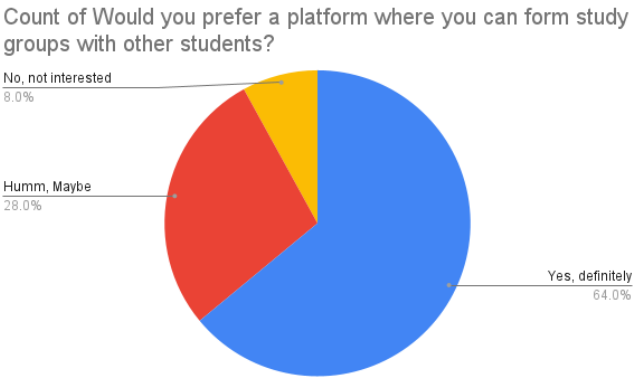

Fig. 11. Questionnaire Result

References

[1] Google form. https://www.google.com/forms/about/. User Investigation Form Creation.
[2] Optimal workshop. https://app.optimalworkshop.com/.

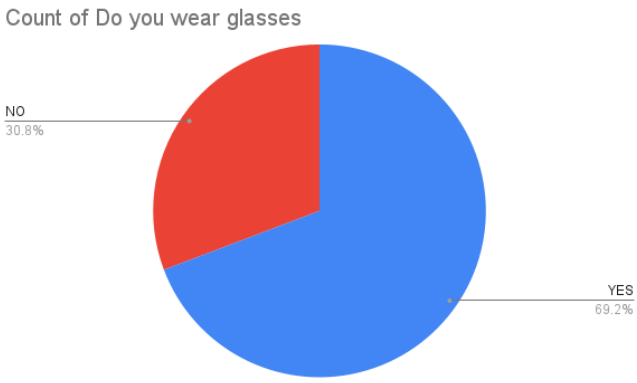

Fig. 12. Questionnaire Result

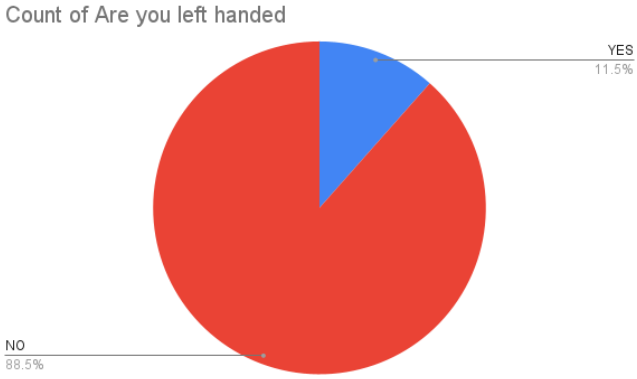

Fig. 13. Questionnaire Result

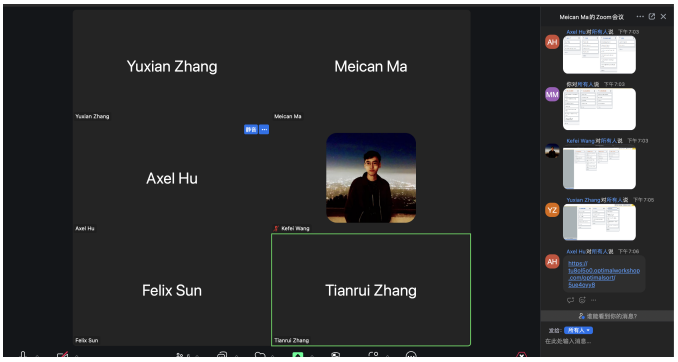

Fig. 14. Card Sorting Session

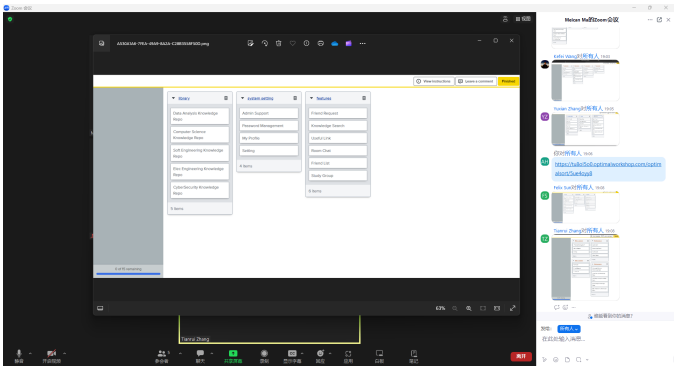

Fig. 15. Card Sorting Session

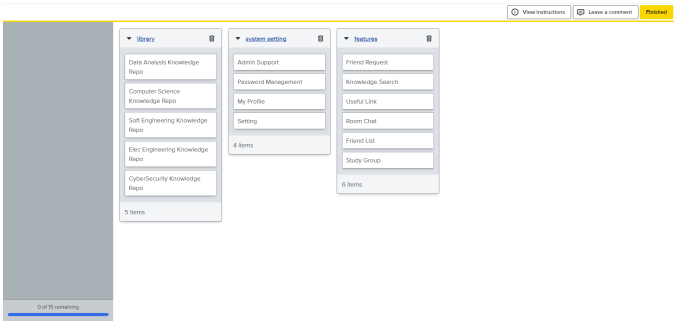

Fig. 16. Card Sorting Result

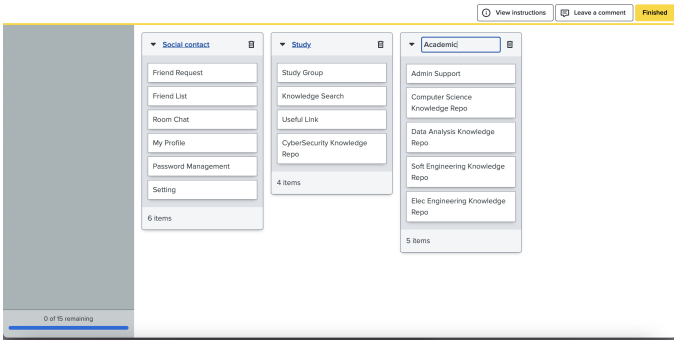

Fig. 17. Card Sorting Result

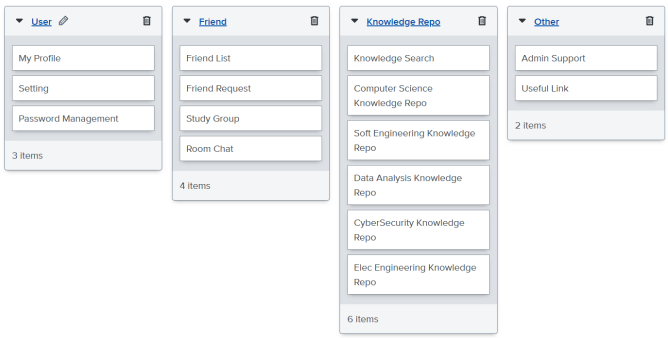

Fig. 18. Card Sorting Result

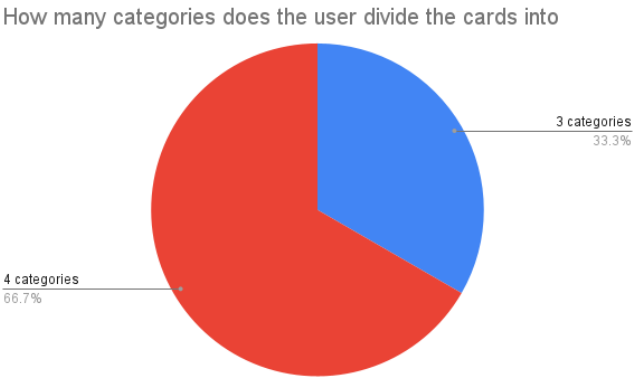

Fig. 19. Card Sorting Analysis

Fig. 20. Card Sorting Analysis

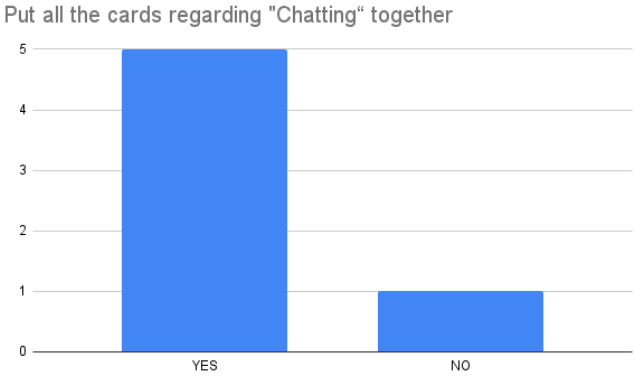

Fig. 21. Card Sorting Analysis

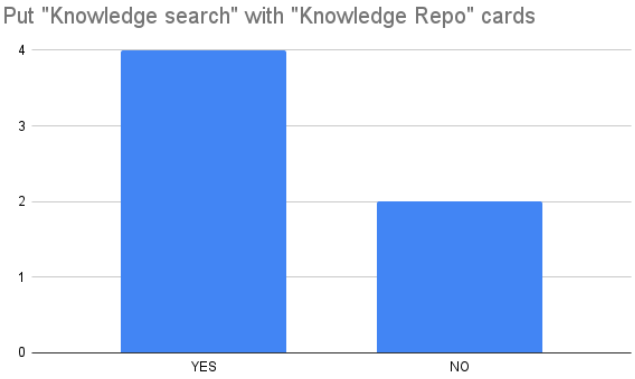

Fig. 22. Card Sorting Analysis

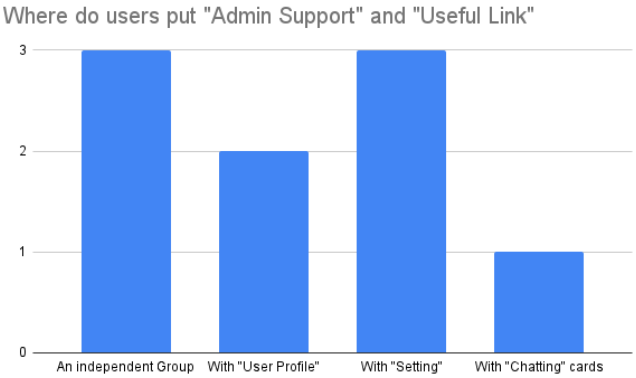

Fig. 23. Card Sorting Analysis

Fig. 24. Welcome Page

Fig. 25. Signup Page

Fig. 26. Login Page

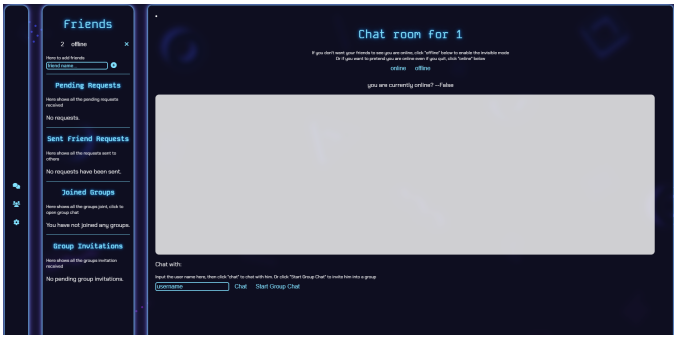

Fig. 27. Home Page

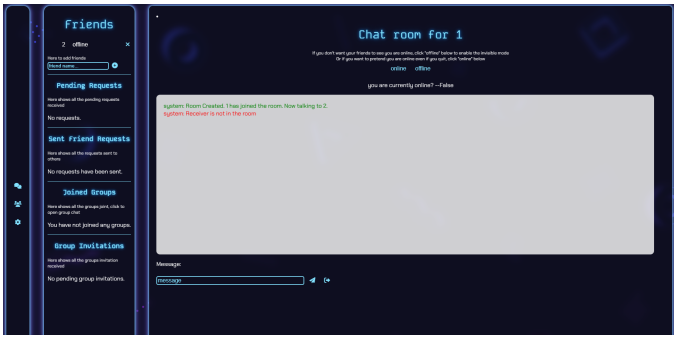

Fig. 28. Chat Room

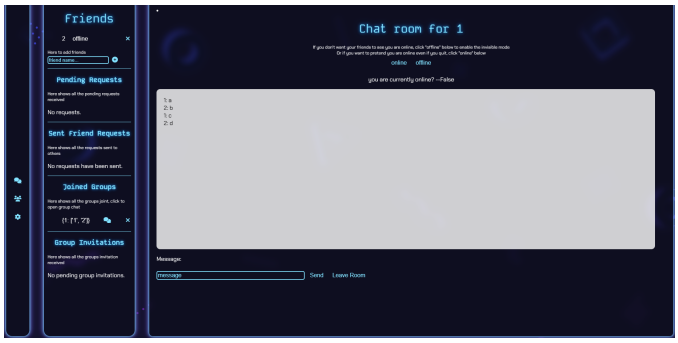

Fig. 29. Group Chat

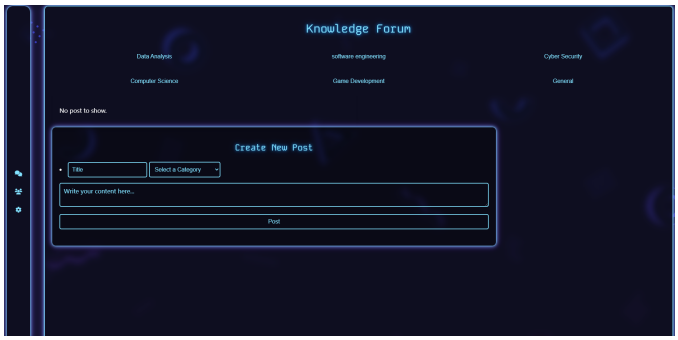

Fig. 30. Forum

Fig. 31. Forum

Fig. 32. Setting