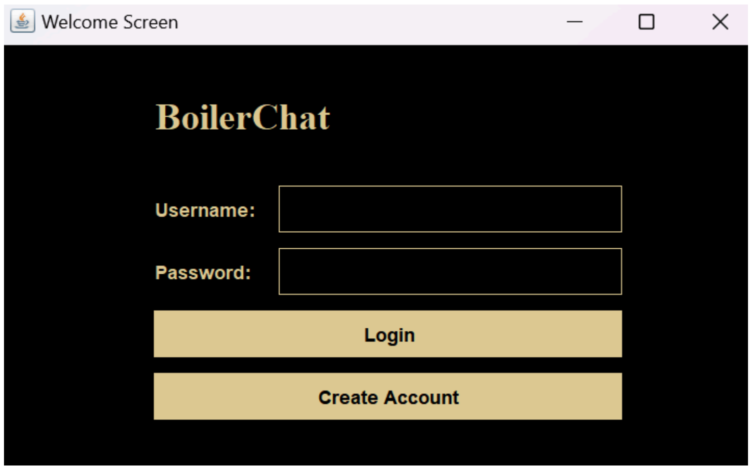
CS 18000 Team Project: Social Media Platform

Junyoung Bae, Mark Waldron, Zoe Amerman, Ray Chen, Ethan Lebon

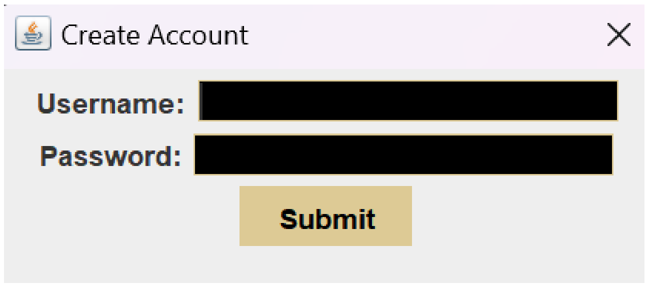
Part One

<u>Project and Functionality:</u>

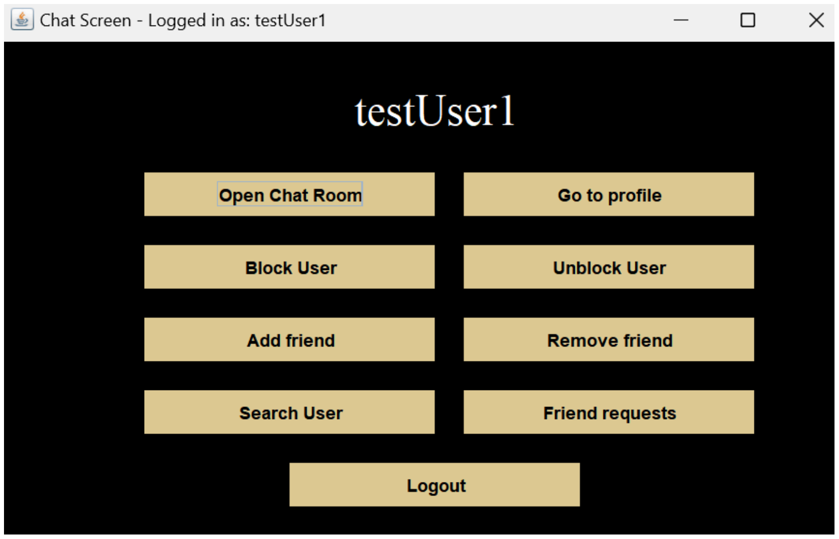
Upon first opening the program, the user is presented with the “Welcome Screen”.



On this screen there is the option to either log in (using the username and password fields) or to create a new account. If the user chooses to create an account, an additional pop-up window will appear:

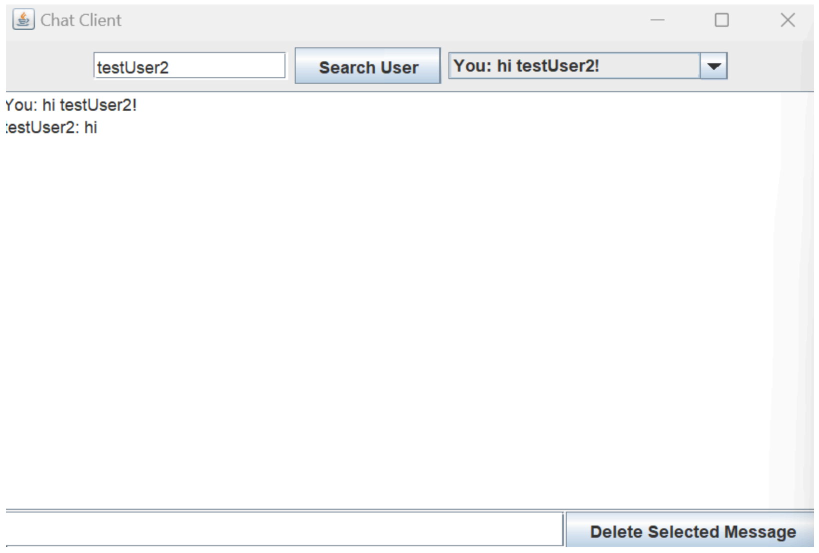


After logging in, the user will see the following screen:



The functionality of each of the buttons will now be explained:

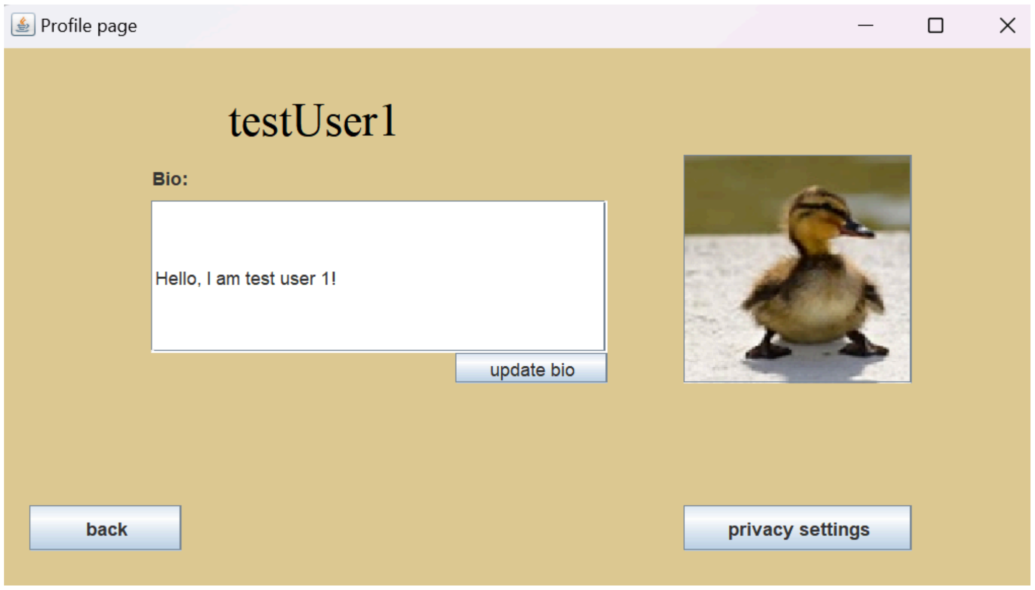
"Open Chat Room": This button opens the chat room, where users logged on at the same time as each other are able to chat with each other in real time.



You: hi testUser2!
:estUser2: hi

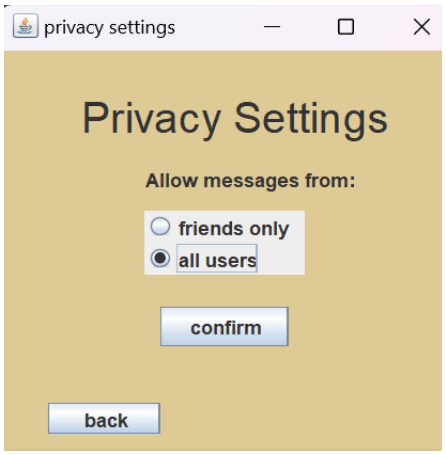
The “Search User” button at the top can be used to select which user one wants to chat with. A user can type a message into the message box and press ENTER in order to send a message. There is also a drop down in the top right which can be used to select a message, and a “Delete Selected Message” button which will remove any message that is selected.

“Go to profile”: This button takes the user to their profile page, where they have the ability to view and update their bio and profile picture. To update their bio, a user can edit the text in the text box and then click “update bio”. The user can also click on their profile picture in order to update it.



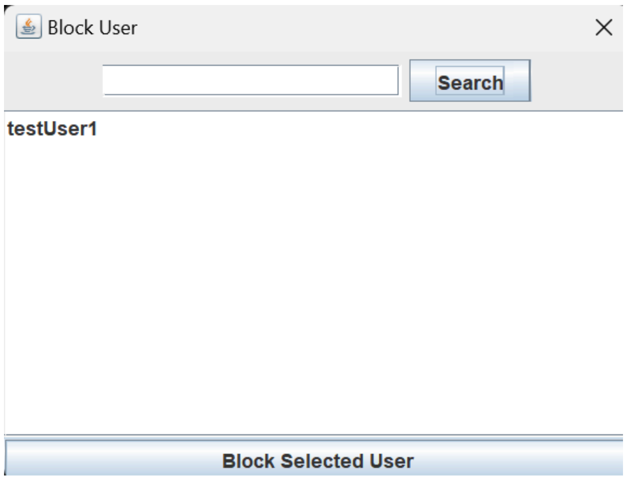
Hello, I am test user 1!

Also, from this page the user may go to their privacy settings using the “privacy settings” button. There they may select whether they would like to allow messages from all users or only from their friends, and press the “confirm” button to make their choice take effect.

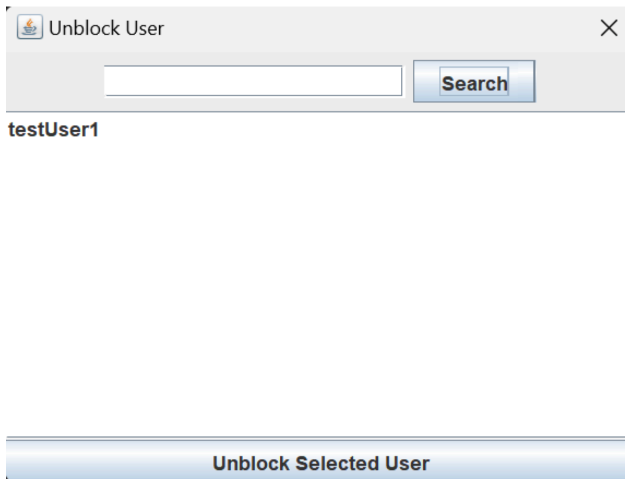


Allow messages from: friends only [✓ all users]

“Block User”: This button opens a dialog which can be used for blocking a user. Search the user's username or part of it, select it, and press “block selected user”.

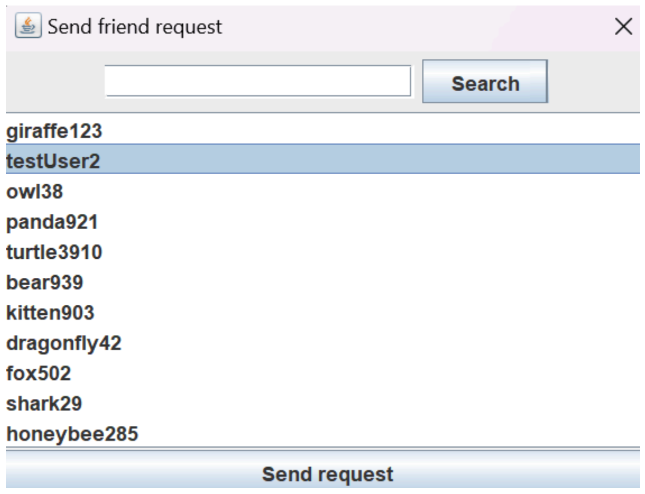


“Unblock User”: This button works similarly to the “Block User” button, except that it allows the user to unblock someone that they had previously blocked.



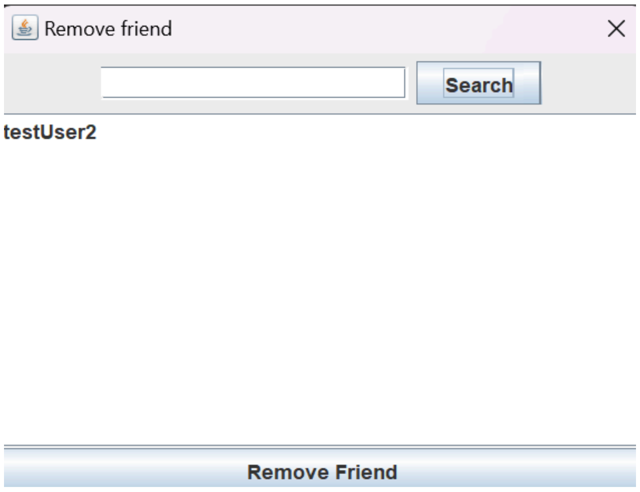
testUser1

“Add Friend”: This button will open a dialog which allows the user to send friend requests. Once a user is selected and the “send request” button is pressed, a request will be sent to that user.

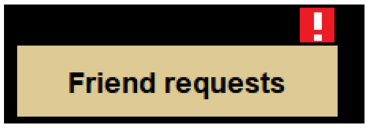


giraffe123
testUser2
owl38
panda921
turtle3910
bear939
kitten903
dragonfly42
fox502
shark29
honeybee285

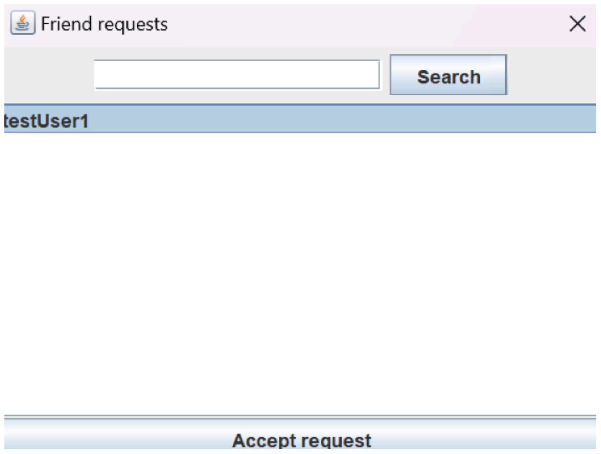
“Remove Friend”: This button will open a dialog which allows the user to remove a user from their friends list. Only the user’s current friends will be displayed here.



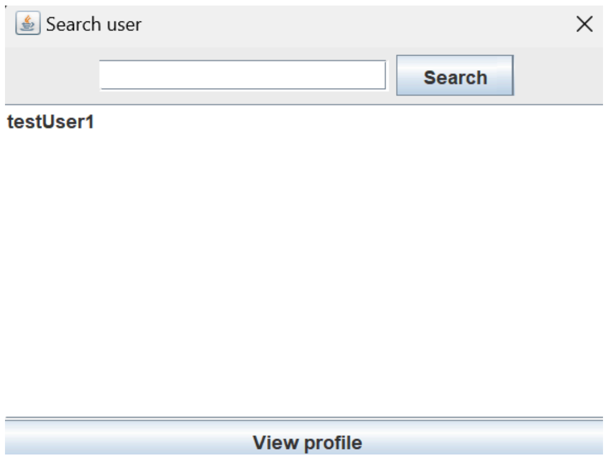
“Friend Requests”: A notification indicator will appear next to this button when the user has at least one incoming friend request.



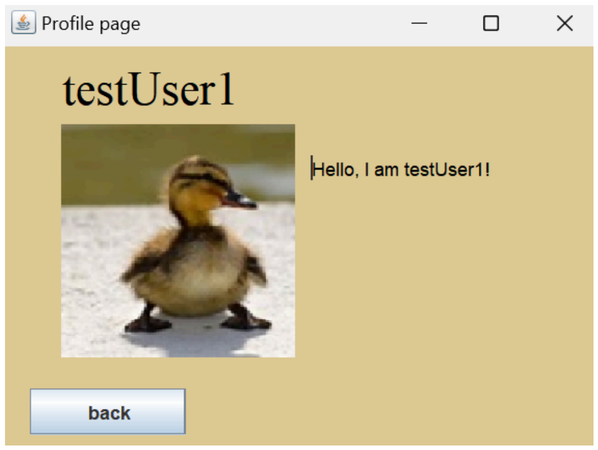
Clicking the button will open a dialog where the user can choose to accept any of the pending requests. Once they accept the request, both of the users will be added to each other’s friend lists.



“Search User”: This button brings up a user search dialog, which allows a user to search for another user and view that user’s profile. Select a user from the menu and click “View profile”



The user will then be presented with a page like this, where they can view that user’s profile picture and bio:



Hello, I am testUser1!

“Logout”: This button logs the user out of their account as well as saves changes that they might have made. It returns them to the welcome page where they can log in again.

<u>Design Choices:</u>

A list of some of our important design choices:

- One design choice we made which affected the appearance of the entire program was to make the color scheme black and gold, the colors of Purdue. We thought that this would be fitting for a project created in a Purdue class.

- Another design choice was to have it so that creating an account does not automatically log the user in, and instead they are presented with a login page where they can enter the information for the account that they just created. We found this to be the best way to do it because while it does not add very much additional time for a user to log in right after they create their account, if for any reason they wanted to create more than one account (to test the platform for instance) then it would be a lot faster to do so.

- On the home page, the buttons are grouped by functionality, that is, buttons with related functionality to each other are usually placed into the same row. This makes the button layout feel more intuitive and a user would not have to spend as much time searching for a given button as they otherwise might have had to.

- The user profile page is laid out in a way that is intended to be aesthetically pleasing and easy to navigate. The elements are kept relatively spaced from one another, and the privacy settings are moved to a separate page, for the reasons of not making the profile page too crowded and also because privacy settings would seem a bit out of place on the main profile page due to a different type of functionality from the rest of the items on the page.

- On the privacy settings page, a radio button group was used rather than regular buttons because this makes it more clear to the user that there is just one setting being controlled here with two options.
- The buttons for blocking, unblocking, friending, unfriending, or accepting friend requests from users all create pages that open as a separate dialog window rather than changing the contents of the main window. This is because any actions from these buttons should be relatively quick, and aren't worth shifting the user to a whole other page and making them adjust to that page.
- The dialog windows have a search bar if a user wants to narrow down the users displayed to them, but the full list of eligible users for whatever dialog it is will still appear without any search. This helps to prevent the user from being confused as otherwise, they might open the dialog and not be sure why no users are being displayed.
- A notification icon appears by the friend requests button when a user has at least one incoming friend request. This allows a user to know quickly whether they have any friend requests when they log in, without them having to manually check, and prevents them from accidentally ignoring another user's friend request if they forget to check.

Part Two

<u>Junyoung:</u>

Junyoung's Contributions: In our project, I was responsible for the foundational elements that facilitated the entire application's functionality. I developed the primary logic in Main.java, setting up the startup sequence from initializing the user interface to establishing server connections. This was crucial as it set the groundwork for the application's operation and user management, integrating smoothly with Server.java, which Mark handled, ensuring a seamless transition from local to server-based interactions. Furthermore, I wrote the basic structure of the User.java class, which includes essential attributes like user IDs, usernames, status information, and methods for user management. My implementation laid the foundation for various user-related operations, such as authentication and profile management, with Zoe adding further functionalities. I also played a pivotal role in the interface design, specifically in developing the WelcomeScreenGUI and ChatScreenGUI. The Welcome Screen was designed to have options for logging in or registering. I used the main.java file that I wrote for user authentication and creation. For the Chat Screen, I initiated the button designs, which were later visually enhanced by other team members to improve aesthetics. Additionally, I contributed to the presentation slides explaining the basic functionalities of our product and addressing FAQs that potential clients might have, ensuring our project was well-represented externally.

Reflecting on our project, I see significant opportunities for improvement, particularly in our approach to project management and design. One major challenge was adhering to our project timeline, where we often found ourselves rushing to meet deadlines. If given the chance to start over, I would advocate for a more structured action plan from the outset, with internal milestones set well before the actual deadlines. This would provide us with sufficient time for thorough testing and refinements, likely reducing the need for last-minute coding sprints and enhancing the overall quality of our work. Another area for improvement would be our initial underestimation of the complexities involved in Java GUI development. Our initial design plan was somewhat superficial, which did not fully account for the technical challenges encountered. Dedicating time to a comprehensive study of Java GUI features and best practices before finalizing our designs would have been immensely beneficial, allowing for a more robust and user-friendly interface.

Zoe:

Before the code was written at several stages of the project, I helped plan a lot of the structure that the program would take, and some of the classes that were needed. As for my coding contributions, I wrote a large portion of the methods in the User.java class, the class which represents a single user of the program. This includes the methods writeUsersToFile() and readUsersFromFile() which bring Users in and out of the database through Object I/O. I also wrote the entire TimeStamp.java class so that messages could have time stamps on them. GUI-wise, I wrote the entire user profile page through OwnProfilePageGUI.java (and also the privacy settings attached to it, PrivacySettingsGUI.java), which gives the user the ability to view and update their profile picture, and change whether the "allow messages from friends only" setting is turned on. I also added the entire friend request system, which includes a collection of fields and methods in User.java as well as the classes AddFriendDialog.java, RemoveFriendDialog.java, and FriendRequestsDialog.java (although, the basic structure for those three dialogs was copied from other dialogs made by Ray and Ethan intended for blocking users.) I also wrote a large portion of the README in each phase, especially the descriptions of the functionality of each class.

If given the opportunity to start over, I would go through a more detailed planning process beforehand than was already done, because disorganization seemed to be the cause of a lot of the problems that arose during the project. Team members would write code for instance that had a functionality that had already been implemented by a different team member previously. A lot of tedious-to-fix bugs seemed to arise because the classes written by different team members did not mesh very well with each other. Another thing that I would change is gathering more knowledge beforehand about how clients and servers interact, as well as better strategies for how to implement this interaction, because I felt that client-server interaction was the most difficult portion of the project to work on and the code that was written seemed messy in a way where I suspected there might have been much better ways to do it. A final thing that I would have changed is that I would have implemented Object I/O from the start. At first, when accessing data about users I had tried to do it using a simple BufferedReader and PrintWriter along with specifically formatted text files. However, the process for doing this seemed quite inefficient and any time a new parameter about a user was added the I/O-handling code would have to be updated as well. But then once I found out about Object I/O and implemented it, everything database-related became much easier.

Mark:

As a central contributor to the development of our social media platform, I focused primarily on integrating and enhancing the server-side functionalities that support real-time user interactions. I took charge of the `Server.java` class, which is critical for managing user sessions and routing messages between clients. My work ensured seamless communication across the platform, vital for maintaining a responsive user experience. In addition to the backend work, I also developed critical features that contribute to the user management system. This includes implementing the logic for user authentication, which involves verifying user credentials during login sessions and ensuring security standards are met to protect user data. My contribution was pivotal in establishing a secure environment. I further enhanced the system by refining the user blocking and unblocking functionalities. Initially, these features were basic, but I expanded them to include more sophisticated feedback mechanisms and user notifications. This not only improved the usability of the feature but also ensured users were well informed about the status of their social connections. Also, I was the main designer and creator of the messaging platform. I designed the ways to check for blocking, privacy settings, and deleting messages. I also created the way for users to talk back and forth to each other in real time, similar to how you would on other messaging apps. Lastly, I collaborated closely with Ethan in integrating some of the frontend features related to these functionalities. This collaboration was essential in ensuring that the backend logic was well-aligned with the frontend interface, providing a cohesive user experience across the platform.

Reflecting on the project, there are several aspects I would approach differently if given another opportunity. Primarily, I would advocate for a more rigorous initial planning phase, specifically in the areas of feature integration and system testing. Our project encountered some delays due to last-minute integrations and unforeseen complications with server-client interactions. A more structured testing phase would be introduced earlier in the development cycle. This change would allow us to identify and address integration issues much earlier, thereby reducing the rush during the final stages of the project. Implementing continuous integration tools could automate testing and help maintain a stable build despite frequent updates from multiple team members. Another area for improvement would be enhancing our communication protocols between frontend and backend systems. Although we managed to establish a functional interface, the process revealed the need for a more robust and flexible communication framework to handle increased loads and more complex data interactions. Overall, while I am pleased with our final product, these adjustments would likely lead to a more streamlined development process and a more robust final application.

Ethan:

In our team project, I was primarily tasked with developing and enhancing the Graphical User Interface (GUI) components of our social media platform. I focused on making sure the user interface was not only functional but also easy to use and aesthetically pleasing. I contributed to the design and implementation of the `OwnProfilePageGUI.java` and played a significant role in refining the `ChatScreenGUI.java` working alongside Junyoung. One of my contributions was helping with the development of the profile management system through `OwnProfilePageGUI.java`. This interface allows users to update their personal information, such as their profile picture and biography. I helped implement functionalities that made these interactions smooth and visually appealing, improving user engagement and satisfaction. Additionally, I worked on enhancing the chat functionalities in `ChatScreenGUI.java`. This involved improving the chat interface to make communication more interactive and user-friendly. In addition to my work on the GUI, I also collaborated closely with Mark and Ray on developing comprehensive test cases for the application. This was crucial for ensuring the functionality and reliability of our application across different user scenarios. Our testing process was thorough, involving both automated and manual testing phases, which helped us ensure that user interactions were smooth and error-free.

Looking back at the project, there are a couple of things I would consider doing differently. Firstly, the GUI design process could have benefited from a more iterative approach. Initially, we had a few user interface designs that did not entirely meet user expectations, which led to multiple revisions. Implementing user feedback loops early in the development process would have allowed us to gather insights and make adjustments more dynamically, leading to a better-aligned user interface with the users' needs. Additionally, while the integration of backend and frontend components was ultimately successful, the process highlighted the need for better initial integration planning. More detailed upfront coordination between the frontend and backend teams would have streamlined the integration process, reducing the complexity and time required for troubleshooting integration issues. Despite these reflections, I am proud of what we accomplished as a team and the skills I've developed through this project. If given the chance to start over, these changes would likely enhance the efficiency of our development process and result in an even more polished final product.

Ray:

In our project, I was primarily involved in the creation of test cases for many classes, including client, server, message, and others. I was also responsible for developing the

"Block/Unblock User" functionality and the "Search User" feature, all of which are integral to our platform. For the "Block/Unblock User" functionality, I designed and implemented the logic that allows users to block any messages from specific users. If they wish to see messages from that specific blocked user again, they are able to select the user to be unblocked. This involved creating methods that update the user's status in the database and reflect these changes immediately across the client interfaces. My implementation ensured that users can control which users they do not wish to interact with. In addition to the block and unblock features, I developed the "Search User" functionality, which allows users to search for any username that is recorded in the database and open their profiles to view their username, profile picture, and bio. I also ensured that the search results are presented in a user-friendly manner, making it easier for users to interact with the feature. Alongside these functionalities, I worked closely with Ethan on creating and executing test cases for our application. This collaboration was crucial for verifying the functionality and stability of our user interaction features. Our testing processes were comprehensive, covering multiple user scenarios to ensure that each feature performed as expected without any issues.

Reflecting on our project, I recognize several areas for improvement, especially in planning and testing. If starting over, I would place a greater emphasis on integration testing for new features. While our unit testing was thorough, integrating new functionalities sometimes caused unexpected issues with existing features. An earlier and more rigorous integration testing phase would help identify and resolve these challenges more quickly, enhancing platform stability. Additionally, I would focus on improving the scalability of the "Search User" functionality. As user numbers grow, the system's search capabilities must be able to handle increased demands efficiently. Researching and implementing advanced search algorithms would greatly boost performance. Moreover, I would advocate for enhanced collaboration during the development process. More frequent cross-functional team interactions could spur innovative solutions and create a more cohesive user experience. This collaborative approach would foster a better integration of various project components, leading to a more unified and high-quality final product.