Project Report Part 1

Functionality

Our project was to create a social media profile application that, in general terms, allows users to connect to a server using a client and create their profile and also add other users on the network as friends.

The first feature is obvious but important and it's the ability for our application to be used by multiple users at once. We achieved this by using a mix of multithreading and a temporary connection system where the clients only connect to the server to transmit necessary information. We believe this is similar to how modern web applications function, as a website doesn't set up a continuous connection to the server.

Next, all interactions with users had to be GUI based. This was achieved through use of the Java Swing and AWT libraries in the client.

Another important element to our application is that all data entered in is stored and persistent, regardless of whether the user is still connected or not, or if the server itself has been shutdown or crashed. All the data will still be there for users to access.

Descriptive errors using JOptionPanes have also been implemented in order to inform the user on any issue encountered.

Now for the main features of the program, users can create accounts when using the application and access to the application is only allowed through an account login consisting of a username and password. The accounts can then be edited by the user if he or she chooses to once they are logged into the application. Profile information can also be edited, including interests, the bio, phone number, email address, and more. Profiles can also be deleted at the request of the user who created it. An integral feature that makes this application a "social networking" application is the ability for users with profiles to add other users as friends. There is a friends list window that will pop up once users click on the number of friends they currently have (via JOptionPane). There, the user can see all the incoming and outgoing friend requests that the user has received and sent to other people on the application. The user can also open the profile pages and friend lists of those users in the friend window. From there, the user can then choose to reject or accept the requests and then the request turns into a friend. Accounts can also be unfriended once friended if the user desires. There is also a search feature integrated into the application. Users can search for other accounts on the application and from there, the user can then go and send friend requests to those people they searched. There is also an option to list all the users on the platform in the search menu by just keeping the search bar empty and then just pressing search. The basic menu is always open as clicking on buttons only open up new windows on top of the menu. This is to allow the user to continue using the program and providing ease of access even after the user closes the window of the task they are currently doing. To log off the service, the user has to exit out of the main menu.

Design

In regards to the design of the application, when the user first uses the program, the first thing that appears is a login screen. The user can then choose to either login with an existing account or create a new account to access the application. An account is needed to access the application to ensure security. If the user decides to create a new account, they can enter a username, a password, an email, a phone number, a bio and interests the user has.

After successfully logging in, the user is met with a general menu where the user can choose what they want to do next. This menu should always be open even after clicking on a button on the menu except when exiting the client. One option in the menu is being able to look at and edit your own profile, a common feature in most modern social media and allows the user the flexibility to customize their own profile to their liking whenever they want. From this menu, the user also has the option to delete their profile should they choose. If their profile is deleted, the user will then be met with a confirmation window that asks if the user's choice is final and if yes is pressed, the program is exited after a confirmation message.

The user can also view their friends list from this menu. In the friends menu, the user can see incoming and outgoing friend requests to other accounts on the platform. When opening the friends list, a new window pops up to provide ease of access as once you are done looking at the friends list, the user can just close the friends list and be back at the menu. The profiles and friends lists of friends can also be opened from the friends list screen. In order to become friends with another account, the user must first send a friend request and have their request accepted by the other user. This is to ensure the privacy of the other individual.

Another option for the user is being able to look for other users and being able to look at other users' profiles. From this search function, the user can then choose to send a friend request to another account. The user can also have the search window list everyone by merely keeping the search bar blank and just hitting enter. The last option the user can do is being able to log out of the client by just pressing x on the menu.

In regards to the connection our application uses, it only creates temporary connections with the server when information needs to be sent to and from the server. This allows multiple users to connect to the server at once. In case the server gets overloaded, it has the ability to multi-thread if necessary.

Regarding the server, it calls a management class that handles storing all the data sent from the client to the server and also handles account retrieval. There is also a separate account class that deals with distributing information stored in users' profiles. Since all information that is inputting into the program is meant to be stored

whether the server is currently running or not, all the data is written to a file via printwriter after it's inputted into the application. This text file, called allUsers, is then called upon and updated while the server is running by the management class.

Project Report Part 2

Christopher Chan

As a teammate, I think I did as much as I could coming in late. I didn't notice the group chat on Campuswire for quite a while as I was busy with other work but of course when I came into the group, I sincerely apologized to my group and got right to work. Most of my work involved helping out in the backend segment of the coding. I helped propose the architectural design for what solution we should use to allow for our program to have multiple users use it at once. After the proposal, I helped a little bit with the coding for the solution we agreed on but moved quickly to finish the part 1 of the project report right away after I noticed that nobody had started that yet. After the project report, I did all of the code documentation onto the README file for both the front end and the back end code before recording my part in the project video. I also did some video editing on our recording after we had finished it because we did several takes to get the information right and to clean up some mistakes that we said on camera so that the finished product would look more professional. In regards to what I would have done differently for this project, I definitely wished that I had noticed the group chat earlier than I did. Yes, the minus 20 points off the bat wasn't the best but in my opinion, that value paled in comparison to the extra work my teammates had to do because I wasn't there earlier. I always value my teammates more than myself when working as a group because that's just how I believe teams work. Everyone has to do their part and me not being there when they first started the project really made me feel guilty for potentially setting back my teammates' grades. In addition, if I had gotten started with the team, I would have done more coding work, which ultimately, is the point of this class: getting lots of coding exposure. Had I done more of the coding work, I feel like I would have gained more experience while working on the project.

Brandon Lamer-Connolly

For the first one to two weeks we worked on the project, I was working on making GUIs for the client class. Amy was also helping me with this too. However, Amy focused on the login or creating account GUIs, and I focused on making the GUIs to display the user's profile, friends list, and search menu. While the GUIs I originally designed underwent debugging and other improvements to become the ones submitted with the final version of the project, my original design goals were still present throughout the whole process. After discussing everything with the team to make sure we were all in accord, I went to work designing GUIs that were simple and intuitive. In particular, my teammates and I believed that whenever interacting with the app and opening up new menus/GUIs, everything should lead back to one main menu. We did not believe creating a single window app would work well because we'd have to implement something like a back button, for example, and we didn't think that we could do that with the restricted time we had. So then, it was up to me to build our multi-window-capable app user interface that, with input from teammates and manual testing, was refined into what we submitted. After coding the initial GUIs, our team did a little bit of a shakeup in terms of dividing tasks amongst each other, and as a result, I worked and consulted with Vincent to refine and debug the code for the GUIs.

If I could do this project all over again, I would've sought more opportunities to video call teammates and work on the project at the same time (collaborating through GitHub). While I think our team did a great job, I definitely think we could've built something even better if we had the opportunity to meet and work in person. However, traveling home for break and other obstacles presented itself where we couldn't do that. While I also understand that each member of the team has other things going on in their lives in which more frequent zoom calls may not have actually worked out, I wish I had at least tried regardless of the complications that discouraged me from trying to schedule something. Don't get me wrong, our team had routine zoom meetings each week, and I think we did great work. To summarize, I feel like a lot of the fun of the project is working together simultaneously and feeding off of each other's creative energy; the circumstances of the past weeks made that difficult and, at the time, unfeasible.

Parker Laughner

My job for this project was to write all of the test cases for the backend of our project. In order to do this I started by looking at old RunLocalTest files from previous projects to figure out how they worked. Once I had a good idea of how to write the test cases and set up the class to run all of the test cases I started to hammer away at writing all of the test cases. I learned how to use the Reflect library to write methods that would test a number of things on every method, field, and class such as if they had the correct accessor type, whether a class was concrete or abstract, whether or not something was static or final, and how to check correct return types and parameters. I also made sure to write two test cases for each method to ensure that they would work properly with proper input and if they would handle improper input without failing. This required me to figure out how to writeJUnit test cases that would effectively test the functionality of a method. I knew from writing the previous test cases how to set test cases that would check to see a number of things like if two values are equal, if a value is true or false, and if a value was null so I used that knowledge to run a number of tests to see if every single one of our non-GUI methods would work as they were intended. If I were to do things again I would definitely start working on test cases much earlier; I underestimated just how many test cases I would have to write and how long it would take for me to write every test case. I would also likely try to write more thorough test cases if I had to do this again because some of the test cases I felt were a bit lacking but I ended up submitting them anyways because of just how many more test cases I still had to write in such a short time. Lastly, I would want to work more on the actual implementation of the project, especially on the frontend working with GUIs. I don't really have much experience with creating GUIs so I think the experience gained from making them on this project would be of great use to me in future projects. That said, I am still glad I got some experience writing test cases and testing code so I can write more sturdy and hard to break code in the future.

Vincent Vu

Right when the project was released, I set out to try and bring our teammates together so that we can start planning, distributing tasks, and get this project done as efficiently as possible. Throughout the project's duration, I was responsible for starting up the meetings,

initializing the group discussions, distributing the tasks, and setting out reminders and deadlines.

Our initial subteam setup was that Chris would take care of documentation, Brandon and Amy would do frontend, Parker would develop the JUnit tests, and I would be responsible for backend. I created the server classes and the file/database manager code. While I was developing the backend, I realized we needed a common protocol between the client and server. We needed the server to respond accordingly to whatever request comes from the client. Thus, I was responsible for also creating this common protocol and the request schema.

As the person who distributes the tasks, I also made the decision to redistribute some of the subteams we had initially agreed upon. I decided to take Amy and have her help with developing the JUnit tests, as the JUnit tests appeared to be a larger tasks than everyone had anticipated.

Afterwards, I was responsible for integrating the frontend and backend together. I consulted with Brandon on fixing any errors, and we ended up with a full rewrite as our final solution.

As the project came to a close, I was responsible for proofreading the documentation that Chris wrote, as well as bring everyone back together to record our presentation.

I feel like our team was productive throughout the duration of the project. However, like any other team, there could be some stuff that we could've done differently. For starters, I wished I laid a more solid ground and offered some advice while developing. This could have potentially solved the issue with integrating frontend and backend, as much of frontend was untested and did not correlate well with backend, leading to some complications. This was also one of the reasons why a full rewrite of the frontend was necessary. Another ground rule I wished I had communicated with my team is properly using GitHub, and when we were switching from one branch to another, as well as how often we should be committing and pushing to the repository. Additionally, I hoped that I had enforced slightly better communication, or at least used some other communication means that everyone was familiar with, because it seemed as though GroupMe wasn't the best option for us. Otherwise, I believe our team was a strong one, and I have no bad things to say about my teammates.

Amy Curtland

The first part of the project I worked on was the framework of the client. I created all of the calls to and from the server with a method to handle all of the information that would be needed within each request, along with the basic startup and connection to the server. This was done to provide a base of code for the GUI of the client to be built off of. Afterwards, I transitioned to writing out JUnit tests for the Client to allow for all of the testing to be written quickly to allow for manual testing to begin. This process involved testing that every field that was needed by the Client existed with the right modifiers and type, along with the methods. Afterwards, I began performing manual tests on the Client to search for possible edge cases that would cause issues. One possible issue I uncovered in this process was that the IO between server and client, which was initially a space delineated string, would fail if the user included a space inside of any of the fields being sent. Due to this, the IO system was rebuilt to

send objects as an array of fields, which prevents the different items being sent from being improperly parsed.

If I were to be in a project like this again, I would spend more time to build out the process the end delivery of the project would be built to at the start. In this process, I would define the desired process for each segment of the project, what types of fields will be used in different segments, what Classes and methods will be made throughout the process, and as much as possible to define what is being built. This way there will be less need to wait for one part of the project to be developed in a functional state before the parts that rely on it can be started, and the code to test the end product can be built at the same time as the end product and adjusted to any changes in development, rather than created after the first version of the product exists, and modified to fit any new modifications. This would increase the overall efficiency of the coding process.