Anthony Comstock

Project Summary

First and foremost this project does not have a graphical user interface, due to time and complexity the feature was not added. There is however all of the structure to add on. What I mean by this is the server was designed to parse tags and take actions based on those tags. Tags are a word at the beginning of a message sent to the server that start with /, for example to login the client needs to pass /login to the server followed by username and password. Originally this was to be obscured to the user by making the graphical user interface generate these tags based on the button that was pressed, but since it was not implemented this application works more like irssi or weechat where it is the users responsibility to supply these tags.

Instructions for use:

- clone the git repository branches client and server.
- Build the server application with Server as the entry point.
- Build the client application with client as the entry point.
- As prompted by the client put in the address of the server if they are being run on the same computer this is local host “127.0.0.1”
- Next you will be prompted by the client to login or create account type **/login** followed by enter to login or **/create** followed by enter to create an account(note: the default behavior is login and if nothing is entered this will be assumed, this is by design because originally this was to be abstracted by the graphical user interface, but in its absence random input needs to be handled. Failure to match login credentials or using already in use credentials will result in getting a new login prompt.)
- Next enter your username or a new username depending on if you chose create or login followed by a space and a password ex.(uname 123123123). passwords can be any length. NO SPACES are allowed in the username or password just one to separate the two.
- After you login the general chat history of the room will be displayed to you followed by any personal messages you have received this is done automatically so that you will be up to date with the conversation.
- To send a general message to everyone type **/message** followed by the message you want to send ex.(/message this is my message).
- To view users that are currently logged in type **/loggedin** followed by enter
- To send a message to a user type **/username** where username is the name of the user ex.(/bob hi bob) would send a personal message to bob saying hi bob.
- When you are done with the application type **/exit** followed by enter to quit.

This project ended up having a lot fewer classes than I expected, needing to use threads and locking caused me to have to really re-evaluate how to program the application. I found that many of the ideas that I initially thought would simplify my program instead added unnecessary overhead and complicated it further. As I said before the GUI was going to abstract away the tags by presenting users with a limited set of options that resulted and a specific tag at the front of their message, but this did not come to fruition unfortunately. Still as a whole the project turned out decent it is not optimized in any way, but it runs and has all of the functionality apart from the GUI.