Nicholas Ruiz

ID: 968625442

Final Report

CS 300

7/8/17

Final Report

For the implementation of Chit Chat A server class was created for the server. This class implements a Multithreading for each of the connections that are made. This server class first loads loads the user database from an external file and load them to a list upon start up. The Server class accepts socket connections and creates a new thread for each of the individual connections. These connections are saved on a list for the public chat function. The user is then saved on a list that accounts for the online users. The thread that is created for all of these connections work to take care of the function of the user chatting. The thread is also in charge of registering a user or validating a user. Will return false to the client if the user is registered, online already, or user doesn't exist in the user database. It can distinguish between a private chat and a public chat, sending the appropriate message to the correct user or users. The thread updates whenever a new user logs into the chat application and when the application logs out, along with provide a message to all users of such action. No Gui is provided for the server class.

The Client class has a GUI implemented for a Login window that doubles as Registration and Login functions. This class has a Child class Login that is in charge of the Login processes, and includes its own GUI. They only close when either one of return true, otherwise they will remain open and pop an error window. Upon the correct input the window closes and the Main chat window pops open. This chat window allows the user to send a message to all the current users online. To send a private message to a user they click on the user on the side of the with the user list. To display an online user the user name will be highlighted. Click on it and a private message window will pop up. Type the message in the text box and press send and the message is only displayed to the user that is intended to be sent to, and display on their main chat window signalling that it is a private message. The Client class is responsible for updating all the gui functions, whether that includes opening, closing or updating chat on the window. For a user to logout of the application the user only needs to close their main chat window and that will signal to the user that the user has logged out. It will update the rest of the users that the user has logged out.

One Deliverable that was not delivered was delivered within the time constraint was the user being able to pull up their message history. Debugging presented bugs to get the other functions took priority before beginning to implement this deliverable. Some issues that were encountered had to do with first implementing a GUI and its action performers. At first was trying to implement a separate registration GUi for registration but combined this into the Login GUI. The biggest issue I ran into was trying to implement multithreading in both the client and server classes. Was running into some concurrency issues, and loops not allowing other threads to run its information. This was solved by periodically putting threads to sleep for short amount of times to let the other code run correctly. I implemented the thread class for the function while chatting in the client and used the thread in swing to update the GUI whenever a message was received to update the gui. For the server application I originally tried to implement too many threads for updating online status, messages, and Server connection. I ended up only using one separate thread for the server class and that is serverConnection. This was due to some performance issues with have many threads run on the computer, by using too much CPU.

To run the chat application one must first import the src package from Github at <https://github.com/Tookerton21/CS300> , which Bin Lin has access to. The user must first run **ServerTest,** and this will get the server up and running. Then run **ClientTest**, for each chat window that one wishes to use, one for each user. These two will load the appropriate files that are required to run.