Members: Eric Remington Davey, Hanna Rumpler
Date: 11/5/2019
CS - 436 Networking
Professor: Yanyan Li

Chatting Application Project

Introduction: The requirements for this lab were to implement a chatting system app in python that utilized the TCP socket library. The project specified that there were two programs, one was a client program that was for the user to talk to others, and one was a server to accept connections from multiple client programs so that they could talk to one another. The server also lists who is currently connected, and what their username and availability was.

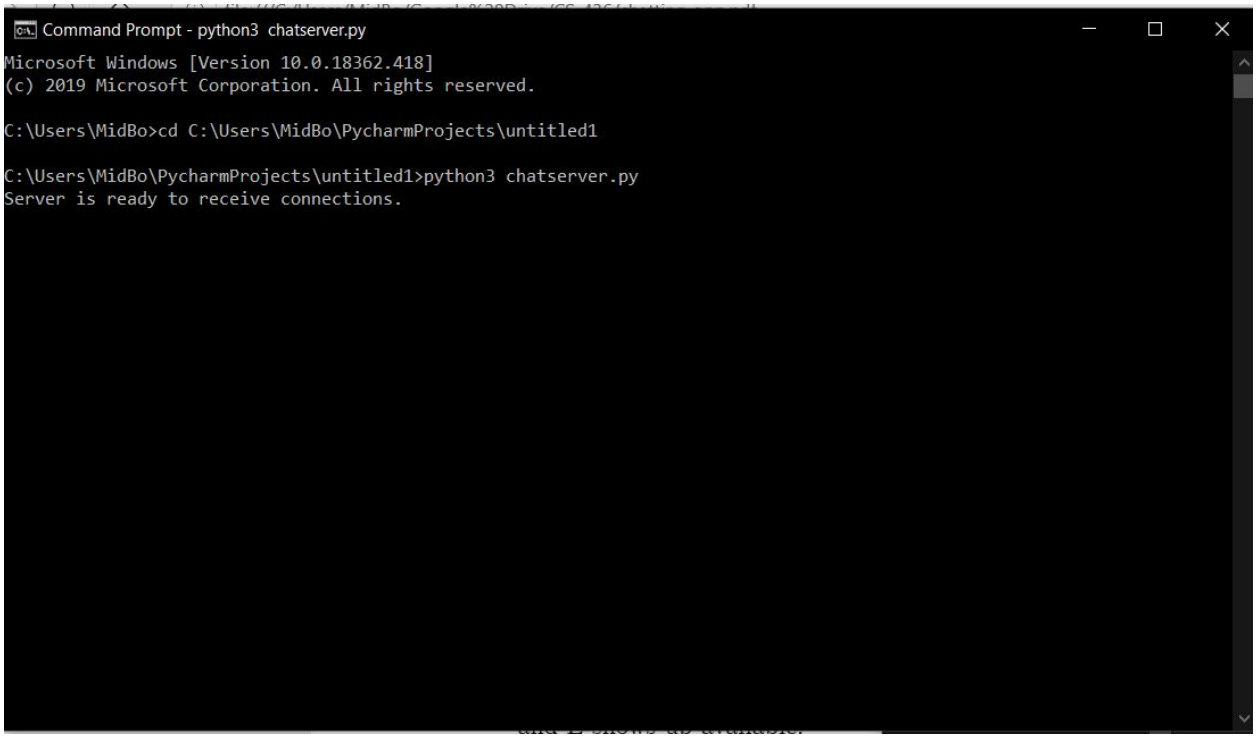
Implementation: We implemented the server side without using the _pThread library for python. The main way we went about this was to place each socket into a python dictionary object, with all pertaining information to that socket user from the client (i.e. 'username', 'available' (whether the person is chatting with someone or not), 'chatting_with' (the name OF the person chatting) and finally the socket information (I.P. address of connecting client, port, ect.)) and to iterate through the set of dictionaries inside of a while loop constantly iteratively checking for an incoming message (socket.recv()) from each client connection and then handling the subsequent request depending on the content of the message.

Results: The finished lab code resulted in a working chat application that could allow two client connections under two different usernames to connect and chat with one another and for other clients under other usernames to connect and see that A and B are chatting and even connect to other users D and E if they want to.

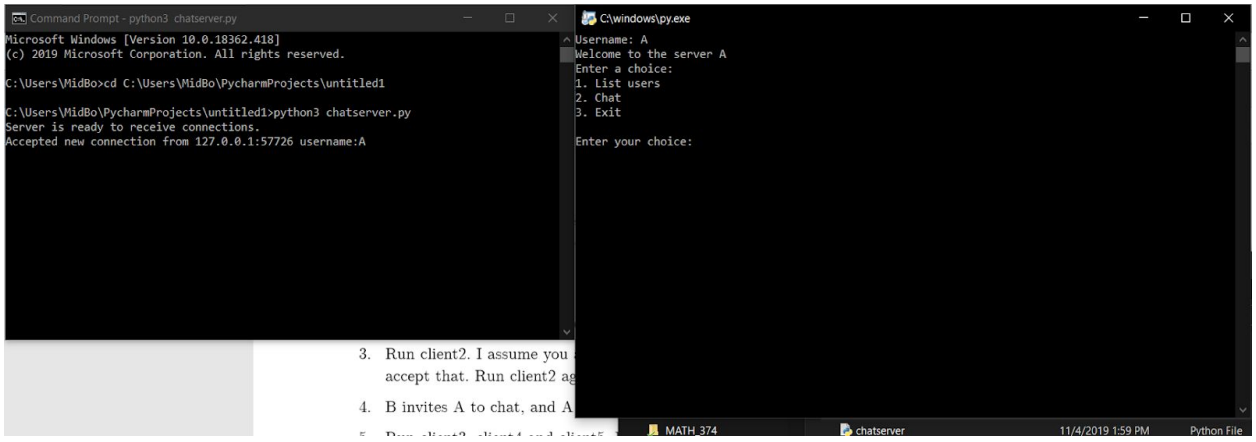
Conclusion: This project was a lot more work than we expected, and we had to rewrite it a few times. At one point, we completely scrapped what we had and decided to look through the example code provided to us one final time. Our final code managed to implement the code without using multithreading or _Pthread library. When we got stuck, we reached out to you in class and searched the internet for examples of similar projects. Some code we borrowed from a tutorial we found (https://www.youtube.com/watch?v=CV7_stUWvBQ). The only problem with the implementation of this chat app was that there isn't an easy way to interrupt the input() command from the keyboard without using _pthread (in the case of receiving a request to chat from another user while being prompted to enter input into the keyboard terminal). Overall, this project has taught us a lot about the python programming language and about sockets.
screenshots:

Below you will find screenshots of the chat app working.

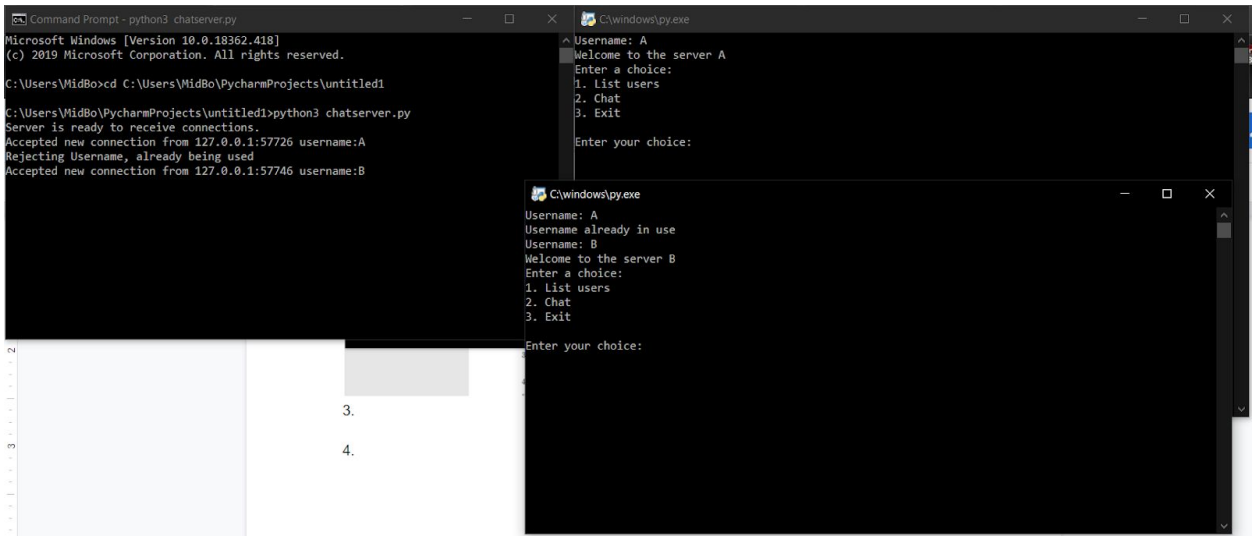
1. As shown below, server executes, waits for connections



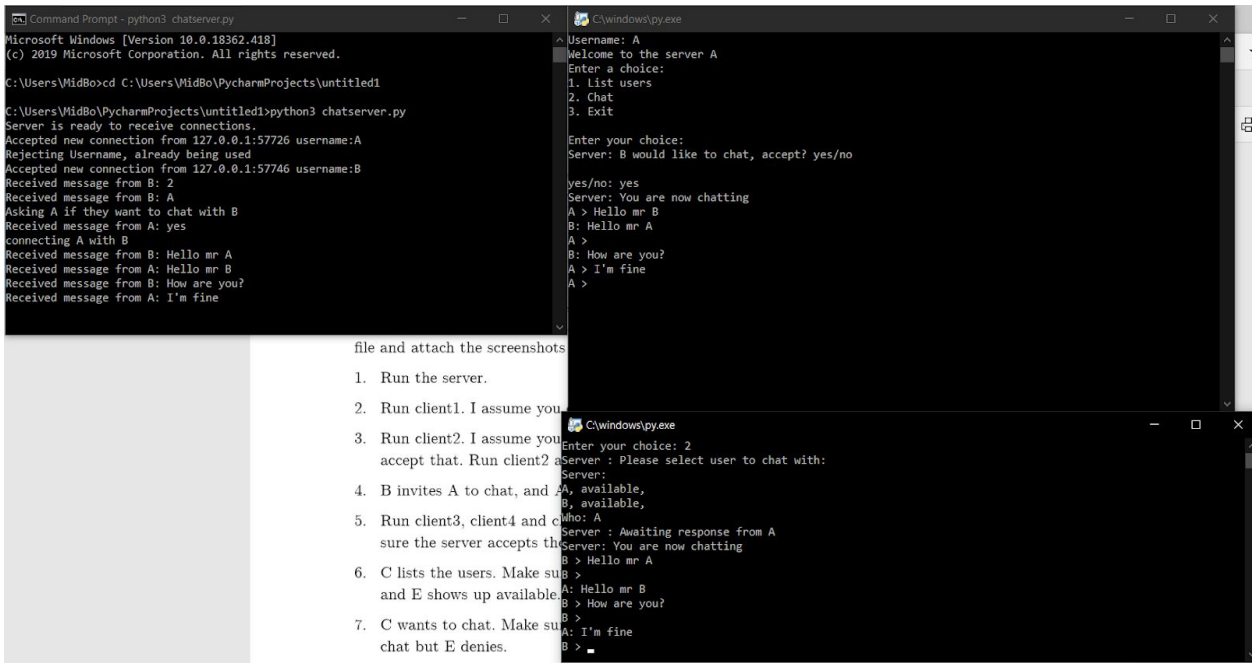
2. Second window is first client enters username A and server accepts.



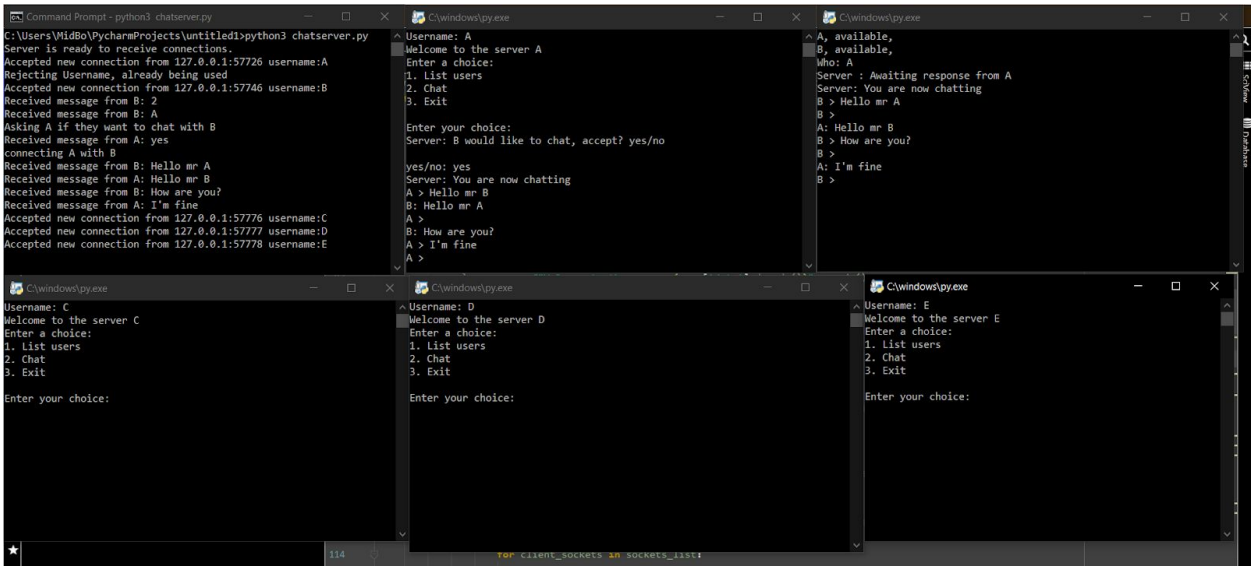
3. Third window is also client, enters A but A is rejected, enters B and is accepted.



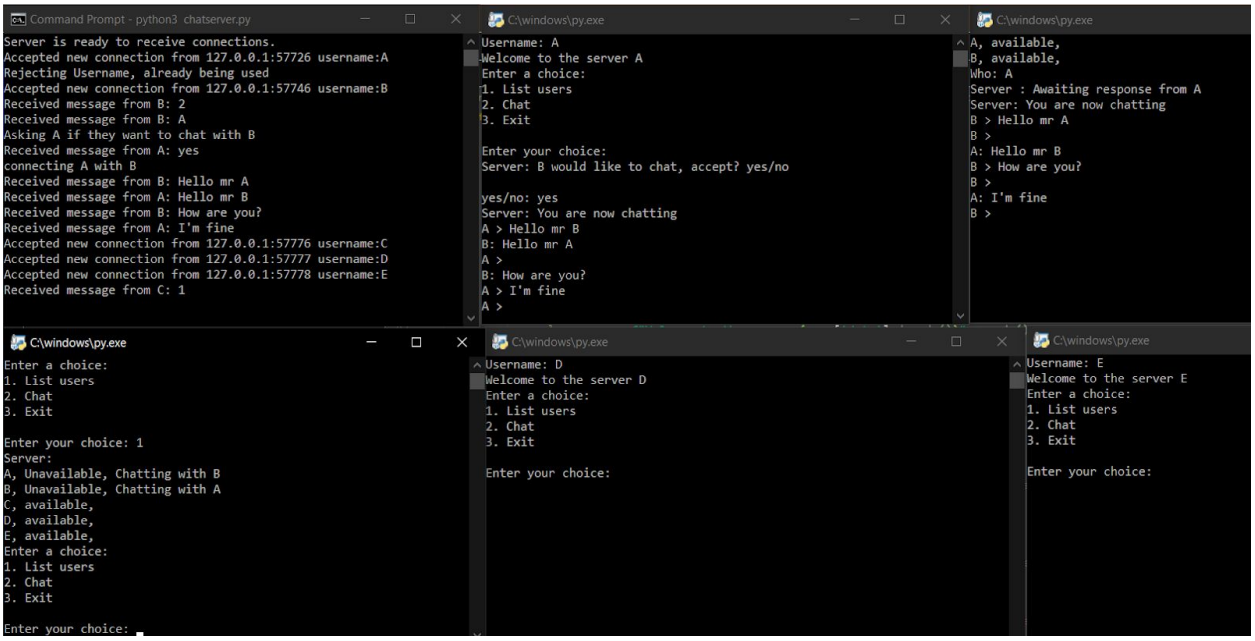
4. Third window client B enters 2, requests to chat with A, A enters yes and accepts request to chat, server connects both users and they send two messages (you have to press enter in order to see the next message from server socket)



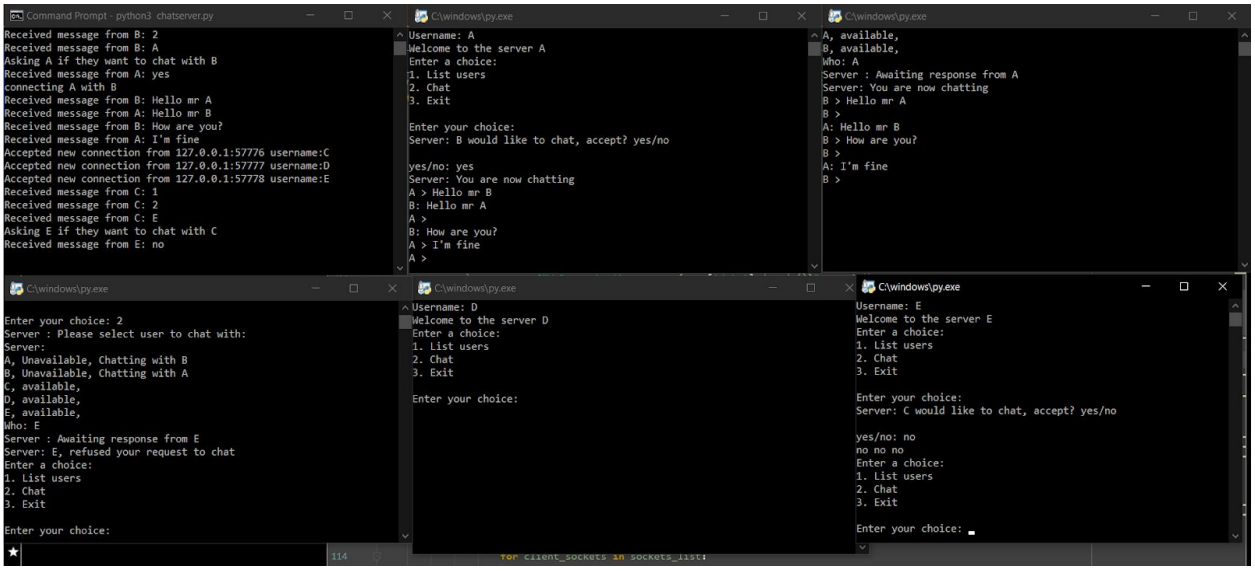
5. new clients (on the bottom) C, D, E all connect



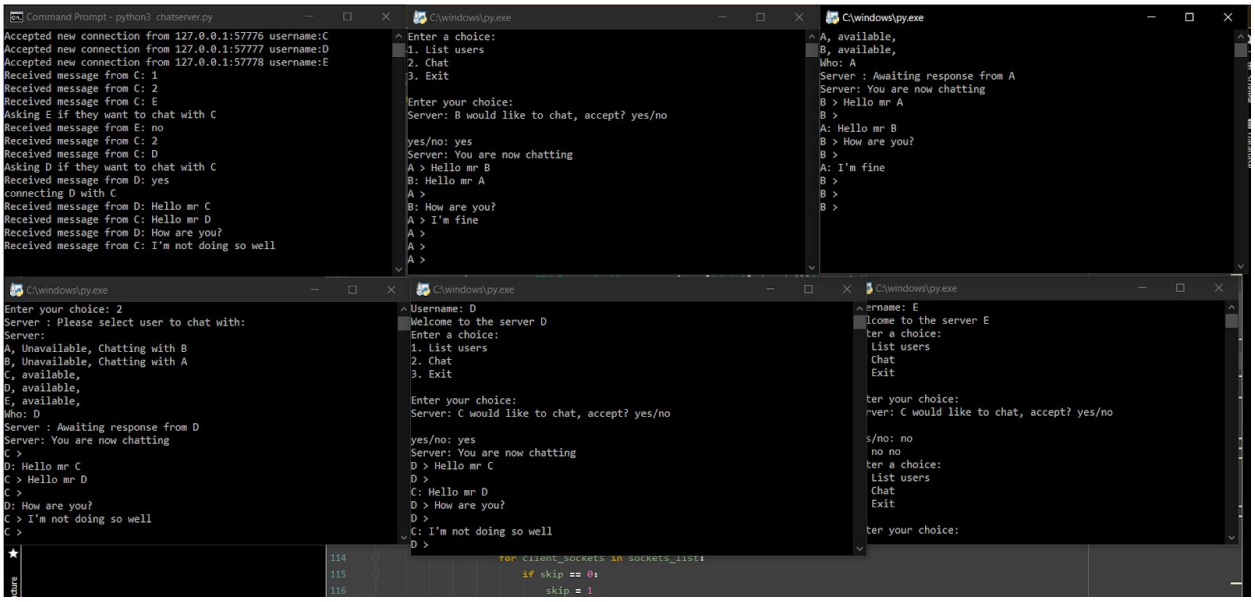
6. C requests a list of users, server shows that A and B and unavailable and chatting with one another.



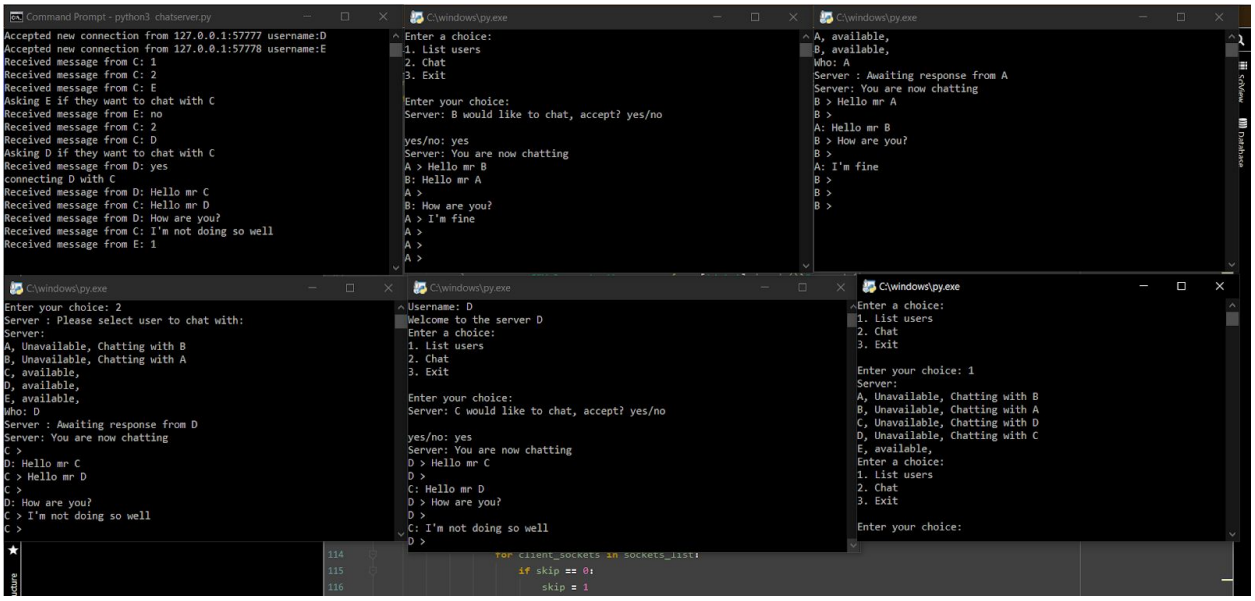
7.C requests to chat with E, E rejects request, both clients return to menu



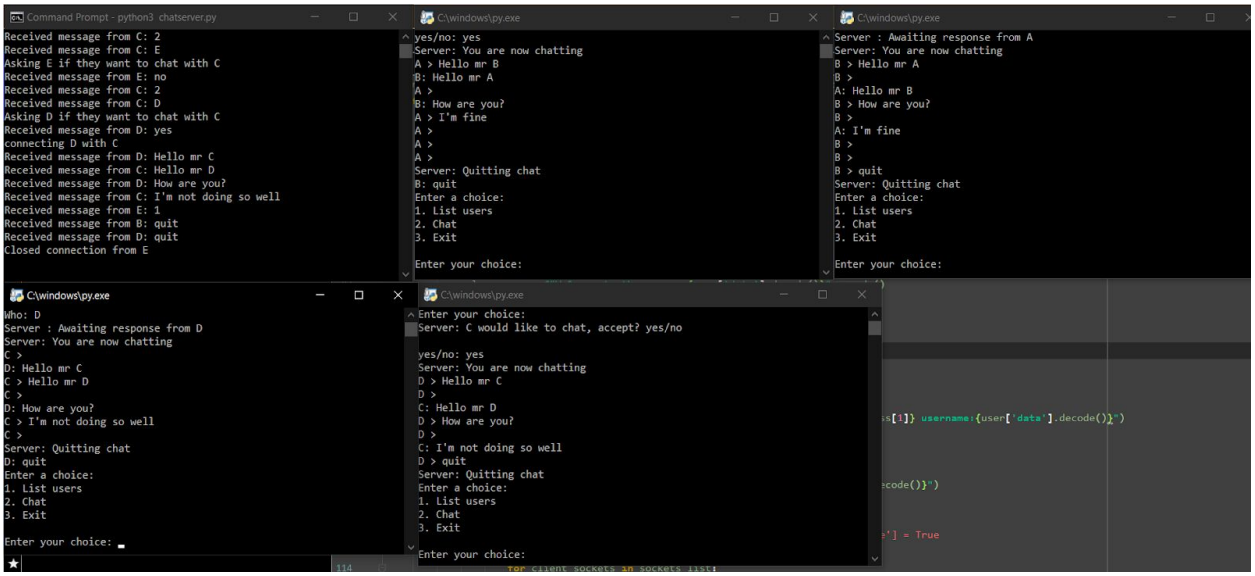
8. C requests to chat with D, they both exchange messages.



9. E requests a list of users who are chatting, sees that A and B are unavailable and chatting with each other, C and D are unavailable and chatting with each other.



10. A and B, C and D all enter quit into chat to quit chatting, return to menu, E enters 3 to exit program, server closes connection from E.



11.

No group chat function implemented

12. A, B, C, and D press 3 and exit out, server closes the connection from all clients.