z5147986
Comp3331 assignment report

Program Design:
Language: Python 3.7
Platform: MacOS, works on CSE too

Features implemented:
Basic: login, message, broadcast, whoelse, whoelsesince, block, unblock, logout, timeout
P2P: startprivate, private, stopprivate

Problems or Potential Problems:

1. Timeout works but
    a. if server is forced closed before clients read the timeout message, OSError: [Errno 57] Socket is not connected will be shown
2. Multiple clients work but the first client that runs the client.py will have to login or disconnect for the rest of the clients to run else it will wait. First come first serve (login)
3. If multiple clients, e.g. run 3 client.py enter credentials from last to first.
    a. Control C the middle one BrokenPipeError: [Errno 32] Broken pipe will be shown.
    b. Control C the last one the server will keep waiting
    c. Control C the first (before submitting) will work
4. Will need to press enter for sent messages or broadcast (including login and logout) the recipient will need to press enter to get the messages
5. To be considered a valid user, user have to exist in credentials.txt and login successful
6. If encountered any issues, it is best to rerun the server

Data Structure used:
Mainly dict, since I am more comfortable and find it easier to implement

Application Layer Protocol:
Message format used:

- Header (Encoded length of encoded content)
- Data (Encoded content)

The message data has to be decoded!

During authentication, client will encode the credentials as Data and send the header + data to the server.
After authentication, the client only needs to send the 1 message, the actual message to the server but the server will send 2 messages, the first message contains details of the sender and the second message contains the actual message content.

Transport Layer Protocol: TCP

Possible Improvements:

1. Better logic and design
    a. Many if – else statement, code gets really long hard to read. And debug
2. Use select the right way, utilising the writefds
    a. Seems pretty difficult to implement it the right way
3. OOP and refactor

z5147986
Comp3331 assignment report

How my program works:

1. Must run server.py <port> <block duration> <timeout>
2. then client.py <host> <port>. *note that it is first come serve for clients stated above
3. E.g 2 clients A and B,
    a. If A **logged in** first then B,
        i. B will broadcast login message to A but if A blocked B or B blocked A, A will not receive the login broadcast.
    b. If A **logged out**,
        i. A will broadcast logout message then B will not receive logout message if A blocked B or B blocked A
        ii. Console A will terminate
    c. If A **timeout**, no message will be broadcast (not mentioned in the specs)
        i. Console A will not terminate since possible p2p connections
    d. If A **message** B:-
        i. If B **block** A, B will not receive any message from A
        ii. If A **block** B, A will not be able to send message to B
        iii. If B **block** then unblock A, B can receive message from A
        iv. If A **block** then unblock B, A can send message to B
        v. If B **block** A then B logout or timeout, B will not receive any message from A
        vi. If B **block** A then unblock A, then B logout or timeout , B can receive any message from A
        vii. If A **block** B then B logout or timeout, then A unblock B , A can send B offline messages
        viii. If A **block** B then B logout or timeout, A will not be able to send B offline messages
4. **Block account** for the block_duration after 3 failed login attempt
5. **Whoelse** shows the number of clients(including you) and the clients currently online regardless of block
6. **Whoelsesince** shows the clients logged in since the past seconds regardless of block
    a. Is online is display if currently online
    b. Was is display if currently offline
7. **Startprivate** creates a p2p client socket to connect another p2p socket
    a. If blocked or block can't startprivate
    b. Server will send private connection details to both clients to establish the p2p connection
    c. Multi p2p is supported
8. **Private** send p2p messages
    a. Have to startprivate first
    b. Similar to client to server interaction where client also acts as a server and communicate to another client
9. **Stopprivate** ends the p2p connection
    a. Have to startprivate first

References:

1. Python socket chatroom tutorial which uses select (main reference)
    - server
    - client
2. The UDP multi-threaded code provided below the assignment specification