z5147986
Comp3331 assignment report

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

Application Layer Protocol:
Main referenced used are server and client which is a select multi client connection tutorial and the provide udp multi threaded client and server examples

Before authentication and p2p Message format used is {header + data}
After authentication, message format used is {user(header + data) + message(header + data)}
Since there are quite a bit of complex features needed for the server to implement, it is easily to track users and print(client side) with the message format used user(header + data) + message(header + data)

Transport Layer Protocol: TCP

Features implemented:

Authentication	Works
Message	Works
Broadcast	Works
Whoelse	Works
Whoelsesince	Works
Block	Works
Unblock	Works
Logout	Works
Startprivate	Works
Private	Works
Stopprivate	Works
Timeout	Works, buggy during or after p2p

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
    b. The timeout message from server will get distorted after having p2p connection
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. Not very sure but if not response it receive, will probably need to press enter but 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

z5147986
Comp3331 assignment report

How my program works:

1. Must run server.py <port> <block duration> <timeout> then client.py <host> <port>. *note that it is first come serve
2. E.g 2 clients A and B,
    a. If A logged in first then B, 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, A will broadcast logout message then B will not receive logout message if A blocked B or B blocked A
    c. If A timeout, no message will be broadcast (not mentioned in the specs)
    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
3. Whoelse shows the number of clients(including you) and the clients currently online regardless of block
4. Whoelsesince shows the clients logged in since the past seconds
    a. Is online is display if currently online
    b. Was is display if currently offline
5. Startprivate creates a p2p client socket to connect another p2p socket
    a. Server will send private connection details to both clients to establish the p2p connection
6. Private send p2p messages
    a. Similar to client to server interaction where client also acts as a server and communicate to another client
7. Stopprivate ends the p2p connection

*I store my users as a list of dict, since I find it more comfortable using dict.

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. Use signals to not have to press enter to get messages
4. OOP and refactor
5. The bonus features looks interesting but also very time consuming
    a. P2p file transfer