Report for assignment in COMP3331

My implementation requires python3 to function!

Program design:

My implementation revolves around server.py and client.py.

Server.py:

This python script sets up the server side of my application. It can be run in the terminal by writing:

```
python3 server.py server_port block_duration timeout
```

The file takes three arguments, server_port (select port for application to run on), block_duration (for how long is the user blocked after too many failed login attempts) and timeout (how many seconds need to pass before an inactive user is logged out). The script begins with initializing a series of global lists and dictionaries, which are used to store info about the users. Before the server socket is created, the script reads the credentials of the users from “credentials.txt”, so for testing make sure that the filename matches.

After the server socket is created, it enters a loop where it accepts incoming connections. A new thread is started to handle the incoming client. The client is first directed to the authenticate routine, whereupon after a successful login the client reaches the state of the program where the user can issue his own commands. This state loops until an error occurs or the user logs out, when the connection is the closed.

Client.py:

This python script sets up the client side of the application. It can be run in the terminal by writing:

```
python3 client.py server_address server_port
```

The client runs two concurrent threads. One that handles messages sent from the server and one that allows the user to issue commands to the server.

Message format:

The message format is generally as in the assignment specification, where all commands in the base assignment is implemented (no p2p). Messages from the server is preempted with a [Server]: and messages from another user is preempted with a simple <username>:

The broadcast function is however used both for user broadcasting and presence notification, where the server calls the function with a “has logged in” or “has logged out” message.

Inner workings of the system:

In general, almost all logic is implemented on the server side of the application. The client side is only focused on sending commands which are interpreted on the server side or receive messages from the server side. Only bit of logic is a simple logout check that terminates the client socket connection.

The logic on the server side is mostly contained in the handle_client routine. After a simple authentication process, the function enters a while loop. Here, some simple pattern matching is used to match the strings sent from the client to appropriate server side commands. Here the commands are also checked to ensure the proper amount of arguments are sent to each subroutine.

Design tradeoffs and possible improvements:

I elected to keep the client side as barebones as possible, so almost no logic is to be found in the client.py script. This would however make the p2p extension of the assignment quite difficult, so one would most likely have to include yet another thread to handle the private messaging. This choice was primarily made to make development as easy as possible. Another decision also had to be made when it came to handle concurrent requests from several users. One option that was considered was to user the select module. However, the function of this module depends heavily on the underlying OS, and as I was developing on my windows computer, I wasn’t necessarily sure how things would differ on the CSE Linux machines.