COMP3331 Report

**The application layer message format and a brief description of how your system works. Also discuss any design tradeoffs considered and made. Describe possible improvements and extensions to your program and indicate how you could realise them. If your program does not work under any particular circumstances, please report this here. Also indicate any segments of code that you have borrowed from the Web or other books.**

**Design Choices**

**Closing Terminal on Client Side**

As in the requirements, the client terminal must be terminated when the client failed to login 3 times, logged out or timed out. I found this difficult to do as I had to stop my main as well as the threads that I spawned in the client program. Stopping the main was a simple *‘return’* however after some research I found out that closing threads was a much more difficult task.

After reading this article (<https://www.baeldung.com/java-thread-stop>), I decided to stop my threads through an interruption rather than deliberately setting a flag in my program. To do this, I had a *‘for’* loop in my client thread which checked for both my STDIN input and if the thread was interrupted. However because the *‘for’* loop is checking for both things at the same time, when an interruption is received by the thread, it does not immediately close and requires the user to press the enterkey an additional time before the program is completely closed.

This is something I wish to improve

**Message formatting**

>> did not want to add additional spaces in the car

**Dictionaries for both Socket and ObjectOutputStream**

As we had to send socket information and packets to different Clients, I kept a HashMap of each client username to their respective

**Having Functions send the Object rather than within my Main on Server side**

**Error Checking**

I did not do much input error checking on the client side and assumed . as I thought it would unnecessarily complicate the code and was also not within the scope of the project.

Server side however, I did some basic checks including ensuring n

**Application Layer Protocol**

For the application layer protocol, I wanted to design a protocol that was simplistic and easy to use. As such I replicated HTTP and created an object that I could easily transfer between my clients to server and peer to peer.

I named this object *‘TCPackage’.* It implements the ‘*Serializable’* interface and as such it can be transferred between clients or server through an ‘*ObjectOutputStream’.*

The *‘TCPackage’* class contains 5 different fields including:

**Content** – This field is the message that is sent between clients or between the client and server e.g. messages between clients or error messages sent by the server. This field is ALWAYS printed out to the terminal by the client program.  
  
**Header –** Below is the table of headers that the respective client or server would expect to receive and parse.  
  
**Client**

|  |  |  |
| --- | --- | --- |
| **Header** | **Description** | **Action** |
| Login/pass | User has successfully logged in | Spawns a thread that reads STDIN and sends data to the server |
| Login/fail/retry | Correct username but wrong password was entered | User is prompted to re-enter their password |
| Login/fail/user | Username does not exist in “credentials.txt” | User is prompted to re-enter their username and password |
| Logout/user | User has either logged out, timed out or been blocked by the server for failed password entries | All threads and socket connections to the user is closed. Program is terminated |
| Msg/user | Default heading for all standard messages between client & server | Do nothing (as package content is already printed out) |

**Server**

|  |  |  |
| --- | --- | --- |
| **Header** | **Description** | **Action** |
| User/authenticate | User wants to login | Checks username and password to ensure user exists and password is correct |
| User/broadcast | User wants to broadcast a message | Goes through list of all logged in users and sends them the user’s message |
| User/msg | User wants to send a message to another user | Finds output stream of other user and sends them the message if not blocked |
| User/whoelse | User wants to see who else is logged on | Sends list of all logged in users (excluding current user) back to the user |
| User/whoselsesince | User wants to see who else is logged on within the last ‘X’ seconds | Sends list of all logged in users within last ‘X’ seconds (excluding current user) back to user |
| User/block | User wants to block ‘X’ user | Adds user to ‘X’ users list of users that have blocked him/her |
| User/unblock | User wants to unblock ‘X’ user | Removes user from ‘X’ users list of users that have blocked him/her |
| User/logout | User wants to logout | Removes user from list of users loggedIn and updates other respective lists. Sends request to user to close connection |

**User** – field used to store the username of the person you want to message/interact with. Only prior to login, this field is used to store your own username.

**IPAddress & Port** – Used to store InetAddress and port number for private messaging / P2P connection