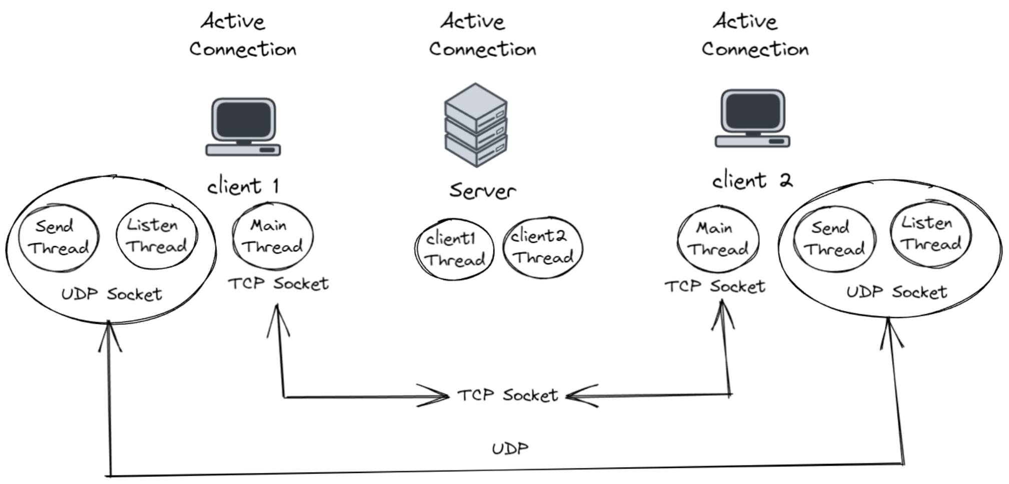
COMP9331 Assignment Report

P.S. Sorry for the late submission, I have carefully tested the code both locally and on the CSE machine, but bugs and unexpected problems may still occur, please restart the program and re-test if you encounter them.

i. Language and Platform

Python 3.9

Already tested on CSE Machine



ii. Program Design

a) Client Design

The client program adopts a process-oriented programming philosophy, which is simple but fulfills the task requirements. Upon initiation, the client s establishes a TCP connection with the server and execute the user authentication process. **Note that I did not authenticate the legitimacy of the username only the username-password match.** After authentication, the client starts a UDP sub-thread (**with no prompt characters**) to accept the file that may be transferred. Meanwhile, the main thread prompts for an available command and briefly checks the legitimacy of the command. Then, depending on the command entered by the user, it interacts with the server and gets the data from the server side and displays.

b) Server Design

The server program adopts object-oriented programming ideas. I constructed a MyThread Class that uses multiple threads to ensure that the server can interact with multiple clients simultaneously. The interaction with the client uses TCP connections and the server creates some log files to simulate the database to meet the functional requirements and to provide correct feedback to the client's queries.

iii. Program Flow

Server:

1) Create a TCP socket
2) Listen for socket connections requests from clients
3) Compare the data stored in credentials.txt with the data transmitted by the client. If successful, run the menu bar function and create a new thread. Else, prompts to re-enter the username and password. Even if the correct password is entered, the user will be locked out for 10 seconds after multiple incorrect entries.
4) Listen for socket connections requests from clients
5) Interact with the client based on commands

Client:

1) Create a TCP socket.
2) Enter username and password. If certification is successful, access to menu functions. Else, repeat the previous operation.
3) Start a UDP thread, listen for possible connections and accept files.
4) The main thread enters commands and interacts with the server

iv. The application layer message format

Command	**Server**	**Client**
0. Authentication		
Case 0.1 Login Success	[login] 1	Welcome to Toom!
Case 0.2 Login with an incorrect password	[login] 2	Invalid Password. Please try again
Case 0.3 Login attempt when account is blocked	[login] 3	Your count is blocked due to multiple login failures. Please try again later
1. BCM		
Case 1.1 BCM message	{username} broadcast BCM # {message_num} {content}	Broadcast message, #{message_num} broadcast at {timestamp}
2. ATU		
Case 2.1 ATU	{username} issued ATU command. Return message: {user}, active since {timestamp}	{username} active since {timestamp}
Case 2.2 ATU	No other active user	No other active user
3. SRB		
Case 3.1 SRB username1 username2 ……	[False]	Your provided usernames are offline
Case 3.2 SRB username1	[True] {username} issued	{username} issued SRB

username2 …	SRB command. Separate chat room has been created, room ID{ID}, users in this room{users}	command. Separate chat room has been created, room ID{ID}, users in this room{users}
Case 3.3 SRB username1 username2 …	[False1]	A separate room{ID} already created for these users
4. SRM		
Case 4.1 SRM roomID message	{username} issued a message in separate room{num} {content}	{username} issued a message in separate room{num} {content}
Case 4.2 SRM roomID message	[TrueR]	The separate room does not exit
Case 4.3 SRM roomID message	[True]	You are not in this separate chat room
5. RDM		
Case 5.1 RDM b timestamp	RDM command issued from {username} Return message {No message}	RDM command issued from {username} Return message {No message}
Case 5.2 RDM b timestamp	RDM command issued from {username} Return message {content}	RDM command issued from {username} Return message {content}
Case 5.3 RDM s timestamp	RDM command issued from {username} Return message {No message}	RDM command issued from {username} Return message {No message}
Case 5.4 RDM b timestamp	RDM command issued from {username} Return message {content}	RDM command issued from {username} Return message {content}
6. OUT		
Case 6.1	Good Bye {username}	Good Bye {username}
7. UPD		
Case 7.1 UPD username filename	[offline]	{username} is offline
Case 7.2 UPD username filename	Sending {filename} Done Press any key to return the menu!	Done Press any key to return the menu