Name: Sareena Abdul Razak
UNI : sta2378
CSEE4119 – Programming Assignment 1 –Readme

1. **Files: -**

tcp_chat_server.py → Main server program file
tcp_chat_client.py → Main client program file
Server.py → class Server and methods
Client.py → class Client and methods
User.py → class User and methods
credentials.txt → File containing user credentials

2. **Data Structure and Program Design: -**

Data Structure

I have defined three classes for implementing a chat room.

a. Server – Server class maintains all the information needed to support the functionalities mentioned in the document.
   - socketConnList → List of all sockets connected to the server
   - serverSocket → Server socket for welcoming connections
   - connSocket →Socket for connecting to clients, to forward messages. Since the TCP connection is nonpermanent, this socket is created and closed as and when it is needed
   - welcomePort → Port number of serverSocket , fixed as 6001. Configurable
   - We can specify the portnumber while creating the server instance.
   - bufferSize → Receive buffer size. Default 4096. Configurable.
   - blockTime →Time interval to block user for 3 wrong password trials. Default 60 seconds. Configurable.
   - heartBeatInterval → Time interval to check heartbeat of clients. Default 35 seconds. Configurable. But this value needs to be slightly higher than the client's heartBeatInterval
   - onlineConnList →List of users that are online
   - users → Dictionary of usernames and passwords
   - userObj→Dictionary of usernames and user objects

Server class also has methods to support all the functionalities needed. They will be explained in the program design.

b. Client – Client class maintains the following variables.
   - clientSocket → Socket for connecting to server. Since the
   - TCP connection to server is non permanent this socket is created and deleted as needed
   - listenSocket→ Socket for listening for messages from server/ other users. This acts as a server socket, so that server and other users can contact the client.

- pvtHostSocket→Socket for chatting privately with other users in p2p chat
- listenPort →port number associated with listenSocket
- socketList = [sys.stdin] →List of sockets to check and userinput,i.e sys.stdin
- bufferSize →Receive buffer size
- name →username of the client
- online → is the client online
- host → server Ip
- port → server's welcome port number
- heartBeatInterval →Interval to send heartbeat messages to server. Default 30 seconds. Configurable
- privateMessageDB →Dictionary for storing username, IP and port for private chat in p2p

c. User – Class to store information related to each user. This class is a helper class for Server class. Sever uses User class to check the status and information about users in the database.
   - name → username of client. Server gets this information
   - from the credentials.txt file
   - sock→ socket. Server saves this socket, however the socket keeps changing everytime the client tears down and creates a new TCP connection. This is just a dummy socket variable
   - isAuthenticated →Flag to check if user is authenticated
   - trial→ Number of times user has entered wrong password.
   - This value is incremented everytime the user enters a wrong password while authentication. When this number reaches 3, user is blocked for a fixed amount of time
   - isOnline →Flag to check if user is online
   - time→Time interval for which a user is blocked if wrong password is entered.
   - isBlocked →Flag to check if user is blocked from logging in
   - because the user entered wrong password three times.
   - ipAddr→ Ip address user logged in from
   - listenPort→ portnumber at which the client listens at
   - heartBeatLast ->Time at which the last hearbeat message
   - was received from the user. This value is updated everytime the user sends a heartbeat.
   - blockedList →List of users this user has blocked
   - offlineMessages →Messages that were sent ot this user, while offline. When the user comes online, these messages will be delivered and list will be cleared.

User class also has methods to set each of these values.

Program Design and Features

Program flow when server is started: -

1. Create a server object instance
2. Create a listening socket and bind it to 0.0.0.0 (so that the server can accept connections from any interface) and port 6001.
3. Add the listening port to the socket connection list .
4. Read credentials.txt and save the username and password in users.
5. Create user object instance for each user and initlialize all the variables.
6. Start a while 1 loop to read the sockets in the connection list. This is done using python's **select.select** function. Refer https://docs.python.org/2/library/select.html to see the function specification. Note : it works only on Unix based systems. Since clic machines are linux based systems, it is fine. What it does is continuously monitor the list of sockets for input.
7. Once a socket is ready to be read, there are two possibilities
    1. A new connection from a user logging in –
        a. User is accepted and authenticated
        b. User is prompted username and password , if the user enters wrong username, client is given infinite number of chances.
        c. If the username is correct and password is wrong, user is given three chances to enter the correct password. If the user fails to enter correct password , user will be blocked from logging in for 60 seconds.
        d. While authentication, the server checks its lists of users that are online and verifies if this particular user is not logged in from any other terminal/IP. If so, the former user is notified and logged off.
        e. If the user enters correct username and password, user will be logged in and added to server's online user list. Server saves all the information related to this user in the corresponding object. User's name, ip and listening port etc are stored. The server also sets the heartBeatLast of the particular user to the current time. Server also sets the isOnline and isAuthenticated flags to True.
        f. Server closes down the TCP connection and removed from the list of sockets to listen to
        g. Server sends a broadcast message to all the online users. However server doesn't notify the users that are in this new client's blocked userlist.
        h. Server checks the offLineMessage buffer of this user. If it is nonempty, server initiates a new TCP connection on the user's listening port and delivers these messages.
    2. A new connection from a user already logged in and authenticated

a. The first word of the incoming message is always a request. The second word is the client's username.
b. Server checks if the user is authenticated and online, if so, server processes the requests. After processing the request the server tears down the TCP connection.
c. If either the authentication flag or the isOnline flag is not True, the user has been internally logged off by the Server – reason being expiration of heartbeat timer. Server simply ignores these messages.

Request processing by the Server

The server caters to lot of the client's requests. Here is a list of requests and how the server processes them.

1. forward:- **Command for this from the client side is message**. Server extracts sender and receiver name. Checks if they are both valid usernames. Checks if the sender is there in receiver's blocked list. If yes, server sends a note to the sender saying, recipient has blocked the client. If the client is not blocked, the server checks if the receiver is online, and then delivers the message through a new TCP connection to the receiver's listening port. If the user is offline, Server saves these messages in the user object's offline message buffer.
2. broadcast: - If the sender is a valid user, server sends the message to all but the sender and users who have blocked the sender.
3. online :- Server has a list of users that are online. If a client requests for a list of users that are online, server sends this lsit. However the server doesn't send the usernames that have blocked the client who is requesting. Server doesn't include the name of requester as well because it is not needed.
4. block :- If a client wants to block another user, server adds the name to the client's blocked list of users. However blocking is not two way, the client can still message the user that it has blocked. If the username is not valid, server notifies the client.
5. unblock :- If a client wants to unblock a user that it blocked, the server removes the user from client's blocked list. If the user is not present in the blocked list, server notifies the client
6. getaddress: - This request is for p2p chat. If a client wants to contact another user privately, the server will send the client ip address and port number of the user. However, if the user is online, server notifies the client. If the user has blocked the client, server wont provide the information. **Server also support consent and privacy**. If the user is online , the server notifies the user that the client wants to private chat with the user. Server asks if the user consents or not. If yes, the server sends the information to the client. If the user answers

no, the server notifies the client that information cannot be provided.

7. private_answer:- When the server notifies the user that the another user wants to privately chat with them, the user is given an option to say yes or no. This answer is sent from the client through private_answer command (this is done internally by the client application).
8. alive : The client program internally sends heartbeat messages to the server every 30 seconds. Client program send these as "alive" messages. When server gets these messages, it resets the lastHeartBeat variable to current time.
9. logout : -logs out the user and notifies other online and unblocked users

Server runs two signal.signal processes.
One is to handle control+C interrupts. The handler, sends notification to all online users and the client program exits.
The second one is to check the heartbeat of the user.

Program flow when client is started: -

1. Connects to the server listening socket and gets authenticated
2. Starts listening port
3. Deletes to socket to de init the tcp connection
4. Client has two members in the select.select read list. Sys.stdin to read user inputs. Listening socket to receive incoming messages from server and other users.
5. Client process various messages from the server
    a. Displays the list of online users
    b. Displays the consent or no consent question for p2p chat
    c. Displays that the server has shutdown and logs of the user
6. Client application also processes all the commands given by the user and sends it to the server
7. Client application directly connects to the user port to facilitate private chat
8. Runs a signal.signal process to check for control c interrupts
9. Starts a signal alarm to go off every 30 seconds, so that the client program can send heartbeat messages to the server

Sample Commands

1) To message another user :
   message columbia hi
2) broadcast hi
3) online
4) getaddress columbia
5) private columbia Hi

6) block columbia
7) unblock columbia
8) logout

All the commands are as mentioned in the specification document.

How to run the program

No compilation required

1. To start the server application
   python tcp_chat_server.py
2. To start the client application
   python tcp_chat_client.py <ipaddress> 6001