CS 4850/7850 Computer Networks I

Project: ChatRoom Version1

Due Date: Friday, October 24, before 11:00am.

1. Overview

In this project, you will implement a simple chat room that includes a client and a server that utilizes sockets. Sockets are implemented in many programming languages. You are permitted to use your language of choice as long as it utilizes sockets.

The client program provides commands: **login** (allow users to join the chat room), **newuser** (create a new user account), **send** (send a message to other clients; actually send the message to the server and the server forwards the message to other clients), and **logout** (quit the chat room).

The server runs a chat room service and echoes messages back to the client.

2. Description

You will implement a server and a client. The server will use 1 plus the last four digits of your student ID as the server port number to avoid conflicting with other students' server program. For example, if the last four digits of your student ID is 3456, then as the server port number is 13456. When running both the client and the server on the same computer, please use 127.0.0.1 as the server IP address.

In this project, only one active client connects to the server. The commands login, newuser, send, and logout (see item 3 below) are input by the user on the client side. The client checks for correct usage of the commands, then relays the commands to the server. The server implements the corresponding functions required to support these commands. When the server starts, it should first read the user account information from the given file users.txt. For grading purpose, the initial user accounts (UserID, Password) are (Tom, Tom11), (David, David22) and (Beth, Beth33).

3. Client/Server Functions to be implemented

1. **login** UserID Password

The client first checks the correct usage of the command, and, if correct, sends the command to the server. If the server can verify the UserID and the Password, the server will send a confirmation message to the client; otherwise, the server will decline login and send an error message to the client.

2. **newuser** UserID Password

Creates a new user account. A new user can invoke the newuser command to create an account (we don't assume an administrator in this scenario). The length of the UserID should be between 3 and 32 characters, and the length of the Password should be between 4 and 8 characters. UserID and Password need to be case-sensitive. Also, assume that UserID and Password do not contain spaces (your program does not need to test for potential spaces).

The client first checks the correct usage of the command (including correct lengths of UserID and Password), and, if correct, sends the command to the server. The server will reject the request if the UserID is already there. The users' IDs and passwords should be kept in the given file users.txt on the server side.

3. **send** message

Send the "message" to the server. The server will precede the message with the UserID and send it back. Message size can be between 1 and 256 characters.

4. **logout**

Logout from the chat room. The connection between the server and client will be closed and the client should exit. The server should continue running and allow other clients to connect.

4. Program specifications

Client Side Specs

- While logged out, a user should only be able to either login or create a new user. All other commands should be invalid while logged out.
- While logged in, a user should only be able to send messages or log out. The user should not be able to login while already logged in or create a new user while logged in.
- Password length and username length restrictions should be implemented as outlined above.

Server Side Specs

- New user accounts should persist between sessions (i.e., the new user information needs to be stored in the users.txt file by the server). If the file does not exist, the server should create it when the first account is created.
- Usernames must be unique. A new user cannot be created with the same user name as an existing user

5. Programming Language

You can use any programming language you like (C, C++, Java, Python, Ruby,…etc). Server and client should be implemented as console applications using sockets, so please do not add a Graphic User Interface to your program. As most of you are familiar with C, client and server skeleton programs in C are posted on Canvas, including Visual Studio project files and compile instructions, as a starting point. You can download Visual Studio for free here: https://visualstudio.microsoft.com/ . Do NOT use *Visual Studio Code* as it does not support sockets.

6. Grading

For Undergraduate Students: Total 200 points

- 30 points for each of the four commands. You will lose points if the commands are not implemented as specified (120 points total)
- 80 points for neat source code and implementing appropriate error messages. Your source code must be well commented, including an overall header with student name, date, program description, etc.
- You will lose 160 points for any bug that causes the program to crash or makes the program exit abnormally even if all commands can be demonstrated.
- You will lose 200 points if you do not utilize sockets.

For Graduate Students: Total 100 points

- 15 points for each of the four commands. You will lose points if the commands are not implemented as specified (60 points total)
- 40 points for neat source code and implementing appropriate error messages. Your source code must be well commented, including an overall header with student name, date, program description, etc.
- You will lose 80 points for any bug that causes the program to crash or makes the program exit abnormally even if all commands can be demonstrated.
- You will lose 100 points if you do not utilize sockets.

7. Code submission

You have to submit your source code files through the course Canvas site. Late or email submissions, or submission of executables will not be accepted.

Please submit two Zip files, one for the client source code and one for the server source code (not executables). After the submission deadline, you are required to demonstrate your project to the TAs to receive credit. The TAs will send out a link to a site where you have to sign up for a demonstration time slot. During the demo, you will:

1. Download and run the code from your canvas submission
2. Demonstrate all required commands and project functionalities
3. Walk through your code to explain how it works

8. Outputs

The client/server functions need to be implemented exactly as shown in Section 3, including the function calls (e.g., newuser Mike Mike11). You are not allowed to change these calls; i.e., do not change it to something like:

Newuser
Please enter user name: Mike
Please enter user password: Mike11

The following shows an example chat room session. **Your client/server programs must reproduce this example exactly.**

Client output:

My chat room client. Version One.

newuser Mike Mike11
New user account created. Please login.
newuser Mike Mike11
Denied. User account already exists.
send
Denied. Please login first.
login Tom Tom12
Denied. User name or password incorrect.
login Tom Tom11
login confirmed
send Hello, is anybody out there?
Tom: Hello, is anybody out there?
send Bye for now
Tom: Bye for now
logout
Tom left.

Server output:

My chat room server. Version One.

New user account created
Tom login.
Tom: Hello, is anybody out there?
Tom: Bye for now
Tom logout