Homework 4: bvChat

Due: 11:59 PM, March 24, 2023

For this homework assignment, you will create a chat application that supports a set of requirements posed in this document. The application will support a client-server model in which you will implement both the server and the client for the application.

Clients must ultimately be able to connect to the server, chat with others that are logged in, and leave messages for users that are not logged in. The server will maintain a core set of state that supports varied operations for the users as well as a set of protections for the server.

Groups Student will work in groups of 2-3 to complete this project. Groups of 3 will be required to complete an extra set of project requirements identified later in this document.

Programming Language You may use any of: Python, Java, C, or C++ to complete this project.

Requirements

The server must support the following set of requirements as they relate to **user authentication:**

1. **Clients must be able to authenticated with a username and password.** Clients will connect to the server by first providing a username (plain-text UTF8 terminated by a new-line character) followed by a password (plain-text UTF8 terminated by a new-line character). If this is the first time the server has received a connection from the specified username, they must save that name and password to file and support additional connection attempts from that user in the future when using the same password. If a future connection attempt occurs with a non-matching password, the connection is denied. That is, the first person to connect to a server using a particular username gets to set the password for that username. When the server is terminated, the usernames and passwords should persist in the file so that they can be automatically loaded and used the next time the server starts.
2. **Simultaneous connections from the same user must not be allowed.** Therefore the server must keep track of exactly who is logged on at any minute to not allow such simultaneous connections.
3. **Temporarily block users with repeated failed authentications.** That is, if a client at a specific IP address fails authentication 3 times in 30 seconds, they should be denied from attempting to log in again for 2 minutes.

The server must support the following set of requirements as they relate to **messaging** between clients and the server.

1. **Broadcasting:** If a user sends a message, it should be communicated to all other users that are currently logged in.
2. **Direct Messaging (Unicast):** If a user types "`/tell backman Hello world!`" Then that message should be sent specifically and only to the user "backman". As you might imagine, any username could be specified. If the user does not exist, report an error to the user attempting to perform the direct message.

3. **Offline Messaging:** When using the "`/tell`" command to send a direct message to a user that is not currently online, the server should save that message and send it to the destination user upon their next login.
4. **Message of the Day:** When a user logs into a server, that server should send a simple greeting back to specifically that user which welcomes them to the server. This can be invoked at any later time via "`/motd`".
5. **Login Message:** When any user connects to the server, a message should be sent to all other users stating that the user has just connected to the server.
6. **Disconnect Message:** When any user disconnects from the server, a message should be sent to all other users stating that the user has just disconnected from the server.

The client (and by extension, the server) must support the ability for the user to employ the following set of commands.

1. **/who** - Displays a list of all users that are currently connected to the server
2. **/exit** - Informs the server that the client intends to disconnect from the server and quit the client application.
3. **/tell username message with many words** - As previously stated, sends a direct message to a specific client. The received message also should denote the sender.
4. **/motd** - Displays the current Message of the Day back to the client.
5. **/me** - Displays an emote message. For example, typing "`/me says hi`" would results in all others seeing "`*backman says hi`" if the user's name is "`backman`".
6. **/help** - Displays a list of commands that the server is capable of supporting.

Requirements for Groups-of-3

Groups of three must implement the following set of additional functionality:

1. **/block username** - A client-side command that prevents the user from receiving any future message (broadcast or unicast) from the specified username.
2. **/unblock username** - A client-side command that allows the user to again begin receiving messages (broadcast or unicast) from the specified username.
3. **Administrator support** - The user that has been connected to the server the longest is deemed an administrator and has access to the following 3 commands.
4. **/kick username** - A command issued to the server that immediately kicks a user off of the server. This command is only enabled for the current admin.
5. **/ban username** - A command issued to the server that immediately kicks a user off of the server and dissalows them from reconnecting to the server. This command is only enabled for the current admin.

6. **/unban username** - A command issued to the server that removes a ban on the specified user so that the can connect to the server. This command is only enabled for the current admin.

7. **A spam blocker** - If a user sends 5 messages within 5 seconds, then they should be muted from sending any messages for 20 seconds. If a user is muted the server should report back to them that their messages will not be delivered every time that user tries to send before the timeout period has elapsed.

Submission: Turning in the assignment

After completing this homework assignment, you should have the following files/folders saved in your CS-Data folder:

CS-Data\Documents\Students\your_username\cmsc431\hw_4\bvChat-server.(py, c, cpp, java)
CS-Data\Documents\Students\your_username\cmsc431\hw_4\bvChat-client.(py, c, cpp, java)
CS-Data\Documents\Students\your_username\cmsc431\hw_4\partners.txt

The partners file should list all of those that you worked with on this project. **ALL** member of a project should save their work to their own folder in CS-Data.

This is what I will be looking for to grade. If you do not have these files/folders, something went wrong – come and contact me.