WAC CHAT Client

CSI 470

In this assignment you'll be writing a client application to connect to a remote chat server. The application layer protocol for this chat server consists of the following. Use python to write a client application that automatically logs into a chat server with the following protocol description. The user of the application should just be able to run the application, happily type messages to whoever is connected to the chat server for awhile and then type quit to close their application. Under the hood, the client is talking to the server using an application layer protocol and the server is sending messages to the client all along as well. This document first lists the application layer protocol that describes how the client talks to the server. Then it discusses this additional feature of the client application. Lastly, this document describes, again, requirements to receive perfect credit on this assignment. Some sample python code is provided to help with new commands.

1 Application Layer Protocol

This describes how the client and server talk to one another at the application layer. Often, the actual client application "hides" much of this and that will be our goal. While there are three commands listed here that the client may send to the server, two of those application layer protocols do not need to be known by the client application in order to be used. The login message should happen automatically. The user of the application doesn't need to know whether its LIN or LOGIN or even that there is a protocol for this. It should just happen automatically. This is similar to HTTP. When you were browsing the web, at first, you had no knowledge that a GET message was being sent to fetch your web pages.

1.1 Messages from client to server

Messages that may be sent to server include:

- LIN "name"

  This is the login message. After this message runs, the user will forever be known by the "name" given after the LIN. This should happen automatically by our client application. Technically though, in a different client application (or a rogue one), it woudl be possible for a user to change their login name at any moment by sending this message.

- MSG “msg”

  This is the send my message with my name to everyone on the server message. When this message runs, every user connected to the server will receive whatever appears in “msg”. In the chat client terminology, this tells the server to send “msg” to everyone else and not yourself. Some chat servers send an echo of your message back to yourself so you can see how it is ordered among the other messages. This particular chat server is not implemented in this way.

- SEC

  This is a quirky little message in which the server sends a secret to the client. Make sure to provide an option for the client application to find out what the server's secret might be. Allow the client application to do this many times. Perhaps the SEC changes after awhile.

1.2 Messages from server to client

Messages that may be sent to the client include:

- “msg”

  Any information sent to the client is a message from the server. The client requires no state at the application layer and should be displaying any message it receives from the server.

2 Additional Client Features

The client for this chat server has one additional command:

- quit

  When the user of the client types quit, the client should - gracefully shutdown and close (both the connection and the - application itself)

3 Requirements for Full Marks

There are a few requirements for full marks on this assignment.

- Shortly after connection to the server, the client should send a login message to the server.
- The login message sent to the server should not require interaction with the client's user. In other words, your particular chat client should always use the same login name.
- The client's user should be able to type any message without the need to write MSG

- The client should gracefully close the connection when the user types 'quit' by itself, and the client should then close.
- The client should display, in full, every message received by the server