Hoosier Chatroom Documentation

1. summary

For this project, I have implemented the basic features for a chatroom. The project has a server side (runs on the remote), and a client side (runs on local), the server can support multiple connections from clients and let them send messages or files to other clients except for themselves. Clients are asked to type commands from the following: register, login, logout, send, senda, sendf and list. They are not allowed to send or receive messages unless they log into their accounts. “senda” can let them send messages anonymously, and “sendf” allows them to transfer a file to all other online clients. Also, clients are allowed to send private messages to a particular online client by entering the username after command “send” or “senda”. They can also list all the users who are currently online.

The main problem I had when I was working on this project is on the client side, the user input blocked the listener to receive messages from server since the program was single thread. Client cannot receive messages from others unless they typed a command. I fixed it by creating two threads: sendThread and listenThread to let client input and receive messages simultaneously.

1. On the server side,

* First, Launch your EC2 server instance.
* Once you are on the remote server, copy over the ServerChat.java to the server, compile it by:

**javac ServerChat.java**

and execute:

**java ServerChat**

* Once you see “server started…” after executing, it means the server is running!

On the client side,

* Save ClientChat.java into your local machine
* Compile it by: **javac ClientChat.java**
* Execute: **java ClientChat**
* Now you are able to type commands after - -“.

(Note: case does not matter for command key words.)

1. User manual

Command Chart

|  |  |
| --- | --- |
| register | [username] [password] |
| Login | [username] [password] |
| Logout |  |
| send | [message] |
| send | [username] [message] |
| senda | [message] |
| Senda | [username] [message] |
| list |  |
| sendf | [file\_name] |

1. Protocol design

Once clients entered a string command, the command will be first transferred to bytes and then sent to the server. The server will receive the bytes and convert it back to a string. Then the server will read the command by splitting the arguments by spaces, and then check if the first arguments is in the command chat, if yes, the server will process the command and the correct output to clients.