Hoosier Chat is a chat room application that allows users to send text messages to each other, broadcast messages and send files too! The application uses a client server architecture and is based on socket programming.

The Server (when active) sets up a server socket that listens at the IP address and port number it resides on. Clients can connect to this socket provided they know the IP and port that the server is active on. The server has one thread listening to new connections and creates a new socket connection for each client that requests it. This connections handles requests and push messages for that client only. The server has another thread which simply listens to the server’s console input and responds to the requests made.

The Client has one thread running to listen to the commands that the user types in to the client console and pass it to the server. So, unlike the server console thread, this one does a dual job instead of just one. Now the client also maintains another thread that is responsible to receive and process messages and data from the server. All interactions between clients are through the server so the clients need to listen to itself and the server only.

The application transmits all commands and messages in the form of strings from client to server and vice versa, except that for transferring a file I use byte transfers so I can transmit non-text based files as well.

**Features:**

(More on how to use them and the output in the Commands section)

1. Account registration:

A new user connects to the server and registers a new account using the Register command. The user will need to provide a username and password which will be stored on the server side. The user should not be registered already to the system. The lengths of both a valid username and password are between 4 and 8 characters including only letters (case-sensitive) and digits.

1. Login:

A user with an existing account can log onto the chat room with the Login command. The user will need to provide the username and password. The server admits the user if the credentials are correct. The server also logs the user out of all the other instances she’s logged in at. One user should not be logged into multiple instances (Inspired from WhatsApp). The server currently supports at most 16 online users.

1. Logout:

A user who is logged into the chat room can log out at any time. (We say a user is online if she is currently in the chat room). On one client, at most one user can be online at the same time.

1. Broadcasting messages publicly:

An online user can broadcast a message with the Broadcast command to all online users (except herself). By default, all recipients will see the sender's username and the message content. Optionally, the sender can choose to make the message anonymous using the BroadcastA command. The maximum length of a message is 4,096 bytes. The message is delivered and displayed in real time.

1. Sending messages privately:

An online user can send a one-to-one message with the Send command to an online user (except herself). By default, the recipient will see the sender's username and the message content. Optionally, the sender can choose to make the message anonymous using the SendA command. The maximum length of a message is 4,096 bytes. The message is delivered and displayed in real time.

1. List online users:

You can ask the server to list all the users who are currently logged into the system.

1. Server Events:

The Server displays all the command hits it gets from clients in real time and stores them all as well. These can be retrieved with the events command (in the order in which they were requested at the server).

1. Check status of a user:

You can check whether a user is logged into the system with the IsOnline command.

1. Who am I:

If the user forgets what her own username is after logging in, she can request the server to display her identity with the WhoAmI command.

1. Check whether a username exists in the system:

You can use the exists command to check whether there’s a user registered into the system with a username.

1. Sending files (Additional feature):

An online user can send a file to another online user (except herself) using the File command, provided the file is present at the sender’s machine. The maximum size of the file can be 1 MB.

**How to launch the application:**

1. ***Server***:
   1. Compile with the command
      * javac Server.java
   2. I’ve hosted the server on Amazon EC2:
      * IP address: 54.191.86.13
      * Port Number: 9001
   3. You can run it manually (in the same machine or a different one as the client) with the following command:
      * java Server

The console shall display the IP address and port number the server is running on (required at the client unless the server runs on the same machine as the client).

1. ***Client*** (Server needs to be in a running state before launching the client):
   1. Compile with the command
      * javac Client.java
   2. You can run the client manually using the command:
      * java Client <IP\_Address> <Port\_Number>

If you don’t pass the IP and port, the client by default tries to connect to the server running at the local machine with port number 9001

**Commands:**

Note: Commands are case-insensitive. Short hands are available for faster use.

Below are the set of commands that are currently supported with a brief way to use them:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Command** | **Shorthand** | **No. of Params** | **Syntax** | **Constraints** |
| Server | Events | e | 0 | Events |  |
|  |  |  |  |  |
|  |  |  |  |  |  |
| Client | Register | r | 2 | Register Username Password | User should not be registered already; Username and Password have to be alphanumeric and between 4 and 8 characters long |
| Login | l | 2 | Login Username Password | User must be registered, Password should match |
| Logout | lo | 0 | Logout | User must be logged in |
| Broadcast | b | 1 | Broadcast message | User must be logged in |
| BroadcastA | ba | 1 | BroadcastA message | User must be logged in |
| Send | s | 2 | Send TargetUsername message | User must be logged in, target user must be logged in, Username ≠ target Username |
| SendA | sa | 2 | SendA TargetUsername message | User must be logged in, target user must be logged in, Username ≠ target Username |
| List | - | 0 | List | - |
| WhoAmI | - | 0 | WhoAmI | User must be logged in |
| IsOnline | - | 1 | IsOnline TargetUsername | TargetUser must be registered |
| Exists | - | 1 | Exists Username | User must be logged in |
| File | f | 3 | File TargetUsername FileName | File must exist; TargetUser must be logged in; TargetUser ≠ User |
| man | - | 0 | man | - |

Below are the functions that each command performs on success and failure conditions:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Command** | **Success** | **Failure** |
| **Server** | Events | Lists all the requests that clients have hit the server with, sorted by time |  |
| <Other> |  | Invalid command. Please type man to see the list of commands |
|  |  |  |  |
| **Client** | Register | Registers the user the system and welcomes her to Hoosier Chat! | Prints an error message and keeps listening for commands that do not require user to be logged in to the system |
| Login | User is logged into the system and is monitored for activity at the server; Logged out if she's logged in at any other instance | Prints an error message and keeps listening for commands that do not require user to be logged in to the system |
| Logout | Logs the user out of the system | Error message that you can't log out if you're not already logged in |
| Broadcast | Sends the message prepended with the sender name to all the users who are online, except the sender herself | The only condition for failure would be a disconnection. In this case, the user is simply logged out automatically at the server and the client program gently shuts down |
| BroadcastA | Sends the message to all the users who are online, except the sender herself | The only condition for failure would be a disconnection. In this case, the user is simply logged out automatically at the server and the client program gently shuts down |
| Send | Sends the message prepended with the sender name to the target user | Target user doesn't exist; Target user is offline |
| SendA | Sends the message to the target user | Target user doesn't exist; Target user is offline |
| List | Lists names of all users who are currently logged in | The only condition for failure would be a disconnection. In this case, the user is simply logged out automatically at the server and the client program gently shuts down |
| WhoAmI | Displays the name of the user who is currently logged in | Prints an error message if the user is not logged in |
| IsOnline | Displays Yes if the user is online and No if the user's offline | Displays a message "User doesn't exist" and so can't tell whether she's online or offline |
| Exists | Displays Yes if the user is registered to Hoosier Chat and No if the user's not registered | The only condition for failure would be a disconnection. In this case, the user is simply logged out automatically at the server and the client program gently shuts down |
| File | File is successfully sent over to the client and a success message is displayed | Appropriate error message is displayed and the client and server keep listening. |
| man | Displays a list of all the commands that the system supports | The only condition for failure would be a disconnection. In this case, the user is simply logged out automatically at the server and the client program gently shuts down |
| <Other> | - | Invalid command. Please type man to see the list of commands |

**Future Work:**

1. Autocomplete commands.
2. Autocomplete messages from the English dictionary.
3. Implement learning models at server, so that the application can predict what the user might say (inspired from Google Allo!)
4. Set reminders, wherein user can request server to send a reminder at the requested time and date on the chat window itself or over email.
5. Block a user from sending messages.

**Acknowledgement:**

I wish to thank Professor Feng Qian for his support and advice throughout the duration of the project.