Chatroom Protocol 👍- Jesse, Makenna, Scott, Shreeya

Requirements

A user should be able to:

- Join the chatroom.
- Leave the chatroom.
- Send a broadcast message.
- Send a private message.
    - While sending messages, the user sends it to the server and the server distributes it to other users or specified users.

Port number 8000

Commands	**Functionality**
JOIN	The JOIN command is used to connect to the chatroom server. - The client will request a socket connection to the server. This request must include a username. - The username must have a minimum length of 3 alphanumeric characters, and the maximum length of 30 alphanumeric characters long with no spaces allowed. - Username cannot be “all” or anyone else's username - Parameter -> JOIN <username>\n - The user is then assigned that username for their session. - The server will check the existing client usernames connected to it and attempt to make a connection to the client. - If the connection is successful, the client will receive a response code of `200 OK\n`. - If the username is invalid, the client will receive a response code of `400 INVALID USERNAME\n` - If the connection fails for any reason, the server will send a response code of `500 SERVER ERROR\n`.
LEAVE	The LEAVE command is used for clients to leave the chatroom, and for the server to disconnect the socket with the client.

	- The client will send a request to the server to close the connection to the server. The request must include the client's username.
	
	- The server will then check for the username connection and close that connection down.
	
	- Parameter -> `LEAVE\n`
	  - Before the connection is successfully closed, the client will receive a response code of `200 BYE\n`.
	  - If the server fails to connect in time, the client will get a `500 SERVER ERROR\n`.

SEND	The SEND command is used to send a message to the server. The command's parameter is a JSON object like the following:
	`{`
	`"sender": "<username>",`
	`"header": "<recipient_username(s)>",`
	`"timestamp": "<timestamp>",`
	`"message: <message>",`
	`}`
	
	`"timestamp" should be in format HH:MM`
	
	The server will send the message to users on the network (only the original json object will be sent to recipients not the sender). The server determines how to distribute messages based on headers included in the client message.
	
	Messages have a length limit of 500 characters, anything longer will result in a `400 MESSAGE FAILED\n` error. (Try to limit characters in the JTextField in client side). Messages must consist of at least one non-space character to send.
	
	The sender will receive the response code of 200 SENT
	
	- `@all` will signal the server to send the message to all other users. This overrides any other headers in the command.
	- `@<username>` will signal the server to send the message to the specific user.
	- To send a message to multiple users, use multiple headers space separated like so: `@<username1> @<username2>...`
	
	- If the message is received by the server, the client will receive a response code of `200 SENT`.

	- If the message fails to be sent to the server, the response code will be `400 MESSAGE FAILED\n`. - Any other issues will send a `500 SERVER ERROR\n.`

USERBOARD	The USERBOARD command shows a list of all users in the current chatroom.
	
	- The user requests the chatroom that they want to see the usernames
	- The server will send a JSON response object in the following format:
	
	`{`
	`“<username0>”: “<status0>”,`
	`“<username1>”: “<status1>”,`
	`}`
	
	- If the server can send all usernames, the client will receive a response code of `200 BOARD {userboard object}\n`.
	- If the server is unable to get all the usernames, such as if there is an invalid username, the client will receive `500 SERVER ERROR\n`.

USERSTATUS	The USERSTATUS command allows users to SET their status/presence (e.g., “online”, “offline”, “do not disturb”). This will appear next to their username. - Client will be able to add, remove and edit their status - The status defaults to `ONLINE` - The different status you can set to are `ONLINE`, `OFFLINE`, `DO_NOT_DISTURB`. - If the status has been successfully set, the client will receive a response code of `200 USERSTATUS UPDATED\n` - If the request isn’t formatted correctly, `400 INVALID REQUEST\n` - Otherwise, a `500 SERVER ERROR\n`.

RECEIVED	The RECEIVED command is used to verify that a message has been received by the addressed client(s). The command’s parameter is a JSON object like the following:
	`{`
	`“sender”: “<username>”,`
	`“timestamp”: “<timestamp>”,`
	`“message”: “<message>”,`
	`“message_id”: “<message_id>”,`
	`}`

	- Once the message has been RECEIVED, update the `sent at <timestamp>` to `received at <timestamp>` for all clients in the conversation. - If the recipient client(s) have received the message, the server will return a response code `200 OK\n`. - If the recipient client(s) have not received the message, the server will return a response code `500 SERVER ERROR\n`.

EDIT	The EDIT command allows users to edit a recently sent message, which is indicated by "edited". This is only available for the original sender. The command's parameter is a JSON object like the following:
	
	`{`
	`"message_id": "<message_id>",`
	`"message: <message>",`
	`}`
	
	- The server will receive the edited message from the original sender
	- The server will then change the original text to the edited message for all users by utilizing the `message_id`.
	
	- If the edit was successful, the client will receive a response code of `200 MESSAGE UPDATED\n`.
	- If the edit window has passed, and the client could not successfully edit, they will receive `400 EDIT FAILED\n`.
	- Any other issues will send a `500 SERVER ERROR\n`.

Notes:

<u>To join and leave the chatroom:</u>

- Server has to be started
- Make a TCP socket when a user enters an empty room
- When client joins the server, keep the socket connection open
    - Send a command that sends the message *client_name has entered the chat.* JOIN command
- Once client leaves, close that user's socket connection

    - Send command that sends the message *client_name has left the chat.* LEFT command
- If no one is in the chatroom, close the TCP socket for that chatroom

~~To send a broadcast message:~~
- ~~Parse the message into the packets to be sent~~
- ~~Time out time~~
- ~~No response or response (confirming if messages made it safely)~~
- ~~Recompiling the packets into the message~~

TCP does the above.

-> Make it not too complex
-> it is the syntax/ format of the command that is exchanged from the client to the server and the server to the client
-> 2 sets of command

Requirements:
- User notification for sent, delivered, read, and replying (also did not send message)
    - Sent -> you have sent it
    - Delivered -> they have received your message (recipients see this message automatically)
- User has entered/left chat
- Indication for sending message privately or publicly
    - Utilize @ to send it to *everyone* or a specific person who is currently available in the chat room
- Show all users in chat so users know who they can private message
- You can block users
    - Print (client_name has blocked you.)
- If the user has sent the message, they get the delivered/sent at flavor text, else they just have the timestamp of delivered
- Timestamp does not have seconds, just the hours:minutes am/pm based on user's timezone

User:
- Usernames cannot have spaces
- Usernames must be unique
- If a user drops connection, they are kicked from the room