**Project Groups 8**

**Muhammed Hamza Duman**

**Ömer Akcan**

**Alaattin Öztürk**

**Kerem Keskin**

Our chat Application is including 2 Python files these are; Client.py, Server.py

Server-Side:

For executing the server side you should use



The first value is HOST which is 127.0.0.1, The other one is our PORT number which is 5009. It can get a value between 0 and 65535.

Server-side implementation for the chat application

The chat application's server side contains the following capabilities and features:

1. Initialization of the server: The server generates a socket and binds it to a given host and port.

- It keeps an eye out for client connections coming in.

2. Client management: - The server receives the client's username upon connection and adds the client to the list of active clients.

- All linked clients are promptly notified of the inclusion of the new client through a message.

3. Message Handling: Each connected client's message is continuously monitored by the server.

- Received messages are broadcast to all clients along with the sender's username attached.

4. Client Disconnection: The server manages client disconnection if a client's connection is closed or reset.

- The client gets taken off the list of active clients.

In order to support real-time communication in the chat application, the server-side solution controls client connections, promotes message exchange, and manages client disconnections.

Our Server GUI seems like this:

A screenshot of a computer

Description automatically generated with medium confidence

Client-Side

For executing the client side you should use:



A screenshot of a computer

Description automatically generated with medium confidence

Client-side implementation for the chat application

1. Client Connection: Using the supplied host and port, the client builds a socket and connects to the server using them.

- The "Connect" button disappears after a successful connection, allowing the customer to type a login.

2. Message Reception and Display: - In a separate thread, the client monitors the server for new messages.

- A message is encoded from UTF-8 and parsed into its username and content when it is received.

- The datetime module is used to get the current date and time.

- After formatting the message with colors and timestamps, tkinter is used to display it in the message text field.

- To specify the formatting styles, several tags (such as "color," "name," and "date") are inserted into the message text.

3. Sending a message: The client retrieves the message that was typed into the message entry form.

- Using the socket's sendall() method, the message is encoded and delivered to the server if it is not empty.

- Following that, the message entry box is cleared.

4. User Interface: Tkinter is used to build the client GUI.

- It has components including labels, text areas, entry fields, and buttons.

- The GUI offers features for starting the client application, communicating with the server, and sending messages.

- The proper event handlers for button clicks and other actions are defined.