

# PROGRAMMING PROJECT 1

## MULTI-USER CHAT

### Project Team:

Name: Anagha Sarmalkar

800#: 801077504

## 1. INTRODUCTION

The topic for this project is creating a Chat-room where multiple clients can communicate with each other. This topic was inspired from Slack and Discord which are popular chat framework for collaborations. This prototype has been hosted locally.

The Chat-room works on the principles of socket programming which is an effective means of connecting nodes on a network to communicate with each other. This chat-room has basic features of clients joining and leaving the chatroom whenever they want. The server is kept running until closed manually.

Sockets help in establishing a bidirectional connection between server and one or more clients. Multi-threading enabled handling connections from more than one client simultaneously.

## 2. PROGRAM STRUCTURE

The programming language used for this project is Python 3. The server program will run forever and the client program can be run from multiple terminals to establish new connections with the server. The program execution is as follows.

### ChatClient.py:

Upon receiving a connection from the client, the server will send a welcome message.

- **user\_action():** After executing the code on the commandline, the program will ask the client for a **JOIN** command. JOIN will connect the user with the server. User inputting JOIN command signifies user's consent to join the chatroom.
- **server\_connection():** This function takes the client\_socket object and connects with the server. Upon connection, a thread is started to receive input from the user (get\_client\_response()). The client actively maintains connection with the server and receives data from other clients (if any). The server response is printed on the client only if there are other clients on the network. A flag is maintained to check active client-socket connection with the server.

- **get\_client\_response():** This function runs simultaneously while receiving the data from the server. This function takes the user input and sends it to the server. If the user input is **LEAVE**, the flag value will be set differently. This flag value will be checked for in the server\_connection() where the client receives data from the server. The client socket will be closed and the loop will be broken accordingly.

### ChatServer.py:

The server is hosted locally on 127.0.0.1:5000. A server\_socket connection is bound at this location where the server is listening for client connections. Throughout the course, the server will remain in the active state to receive requests from the clients. Upon accepting a client connection, a dictionary of client sockets and their addresses is maintained (client\_list). As new clients connect to the server, new threads are assigned to these clients so that they can talk simultaneously.

- **new\_client\_connection():** This function takes the received client-socket and the address. It names the connecting client as USER+port number so that the other clients can identify it. Upon successful connection, the server sends a welcome message("Welcome to the chatroom") to the clients. As new clients join the server, the other clients are notified of this in the form of a message "USERxxxx has joined the chat". An event loop is run to keep the to and fro communication between the clients going on. If the client wishes to leave the chatroom, it must send the server a message "**leave**". The client connection is then closed, the client socket is deleted from the client\_list and the event loop is broken. The other users will be notified about this as the server broadcasts this information to the other connected clients. If only one client is connected to the server, the messages are not broadcasted back to the same client.

## **3. EXECUTION STEPS**

- Run the ChatServer.py on terminal
- Run the ChatClient.py on as many terminals as you want.
- Type JOIN in the client terminal to get connected to the server.
- To leave the chatroom type LEAVE.

## 4. SCREENSHOTS

### Server

```
C:\Windows\system32\cmd.exe - python ChatServer.py

C:\Users\anagh\Desktop\ChatSERVERCLIENT>python ChatServer.py
Listening on 127.0.0.1:5000
USER61014 has joined the chat
USER61015 has joined the chat
USER61014: Hi Guys!!!
USER61015: Hello mates!!!
USER61014: How are you?
USER61015: I am good! How are you!
USER61014: Very good. Nice talking to you!
USER61014: Bye
USER61014 has left the chat
USER61015: Hi Server. Looks like Im talking to just you
USER61015: Bye
USER61015 has left the chat
```

### Client 1


```
C:\Windows\system32\cmd.exe

C:\Users\anagh\Desktop\ChatSERVERCLIENT>python ChatClient.py
Please type JOIN to join the Chatroom : JOIN
Welcome to the chatroom!
USER61015 has joined the chat
Hi Guys!!!
USER61015: Hello mates!!!
How are you?
USER61015: I am good! How are you!
Very good. Nice talking to you!
Bye
leave

C:\Users\anagh\Desktop\ChatSERVERCLIENT>
```

Client 2:

---

 C:\Windows\system32\cmd.exe

```
C:\Users\anagh\Desktop\ChatSERVERCLIENT>python ChatClient.py
Please type JOIN to join the Chatroom : JOIN
Welcome to the chatroom!
USER61014: Hi Guys!!!
Hello mates!!!
USER61014: How are you?
I am good! How are you!
USER61014: Very good. Nice talking to you!
USER61014: Bye
USER61014 has left the chat
Hi Server. Looks like Im talking to just you
Bye
leave

C:\Users\anagh\Desktop\ChatSERVERCLIENT>
```

## 5. REFERENCES

- <https://realpython.com/python-sockets/>
- <https://www.geeksforgeeks.org/socket-programming-python/>
- <https://stackoverflow.com/>