

PROJECT-1

MULTI-USER CHAT APPLICATION

Members:

Sree Divya Keerthi Paravasthu Siddanthi (801149985)

Utkarsha Anil Gurjar (801149356)

Devika Unnikrishnan (801135267)

Socket programming:

Socket programming is used to communicate between multiple processes running on two different nodes. A client and a server on the network are connected this way.

Step 1: In order to establish this connection, a socket connection is opened by the server and then binds its address to that socket.

Step 2: Server should be open to accept incoming connections. A client can send a connection request to a server when the server is started and it is up and running.

Once the server accepts the client connection request, the client can send and receive data to and from the server.

The above described steps define the fundamental process for connection establishment. This project aims to establish a multi-user chat room. A user is free to join and exit the chat room at any time he wants to. We used Python to implement this functionality. We deployed two Python scripts that are the Client and the Server. By using multi-threading in our python script, we were able to account for multiple users joining a chat room.

Multi-user chat application:

A multi-user chat room is a system which enables various clients to join and communicate with one another on the same connection in a chat room. We have built a multi-user chat room in our project based on sockets. A socket address comprises two things: IP address and Port Number. Every client uses a socket address to establish a connection with the server. The client can communicate with another client/clients connected on the same server once the connection is established. In our project, a client can leave the chat room at any point during the active connection. We have created client and server programs using python. We have used TCP protocol instead of UDP because it is more reliable, and data read by the receiver is in order with the sender's write.

Multithreading:

Multithreading of a program is its ability to manage multiple requests by the same user without having many copies of the program in the same system. It is used to execute multiple parts of a program concurrently. In our project, we have implemented multi-threading in Python in order to implement multi-user chat room. We used two threads on the Server side to accept connections and simultaneously send and receive data to and from the server. Similarly, we used two threads on the Client side to send and receive data simultaneously.

Sockets:

Few python TCP socket API functions we have used for our project are:

bind(): It binds the socket to the correct IP address and the port number.

socket(): It returns a *socket object* whose methods implement many socket calls.

listen(): It listens for the connections made to the socket.

accept(): It accepts the connection.

connect(): It connects to a remote socket at the mentioned address.

send(): It sends data to the mentioned socket.

recv(): It receives the data from the socket.

close(): Closes the socket connection and will fail in case of all the future operations.

For the purpose of our project, we have kept port number 5000 and localhost i.e. 127.0.0.1.

Server Program:

Running the server program is the first step in executing this project i.e. socket programming. The server opens the connection and accepts all the incoming client requests. The multiple clients must connect to the same port number and IP address as that of the server in order to communicate with each other. This project deals with multiple user chat room, so multiple users can connect to this server. This connection is done using TCP protocol.

Client Program:

We have used tkinter package for the GUI chat client purpose. When we create a client program our first step would be to connect it to the desired host. To establish a connection to the host we open a TCP socket. In order to accept multiple clients, the server makes use of multithreading to accept multiple client requests as well as to send and receive messages to and from the server. In this project, if any client joins in or

leaves the conversation then it will be notified to the other clients. If one client types some message then it will be broadcasted to all the other clients.

Establishing a TCP connection:

Step1: The server instantiates a server socket object, denoting on which port number the communication is to occur. bind() function is used for the same.

Step2: The server starts accepting connections using the accept() function. We have used an infinite loop that waits forever for all incoming connections. Also, we have kept only five connections to be accepted.

Step3: After running the client program, the client connects to the server.

Step4: The GUI of the client's chat room opens. The client puts his name and receives a welcome message.

Step5: Then that client starts communicating with the other clients. If he wants to leave then, he clicks on the quit button.

Code for server:

Server code for chat system

```
from socket import AF_INET, socket, SOCK_STREAM
from threading import Thread
```

```
clients = {}
addresses = {}
```

```
HOST = "127.0.0.1"
```

```
PORT = 5000
```

```
BUFFERSIZE = 1024
```

```
ADDR = (HOST, PORT)
```

```
SOCK = socket(AF_INET, SOCK_STREAM)
```

```
SOCK.bind(ADDR)
```

```
def accept_incoming_client_conn():
```

```
    # Function handling the clients
```

```
    while True:
```

```
        client, client_address = SOCK.accept()
```

```
        print("%s:%s has connected." % client_address)
```

```
        client.send("Hello everyone in the CCN Course ChatRoom , please enter your name to start
```

```
! ".encode("utf8"))
```

```
        client.send("Enter your message and enter press enter!".encode("utf8"))
```

```
        addresses[client] = client_address
```

```
        Thread(target=handle_client, args=(client, client_address)).start()
```

```

def handle_client(conn, addr): # Takes client socket as argument.
    name = conn.recv(BUFSIZE).decode("utf8")
    welcome = 'Welcome %s! If you want to quit, press quit button' % name
    conn.send(bytes(welcome, "utf8"))
    msg = "%s from [%s] has joined the chat!" % (name, "{}:{}".format(addr[0], addr[1]))
    broadcast_message(bytes(msg, "utf8"))
    clients[conn] = name
    while True:
        msg = conn.recv(BUFSIZE)
        if msg != bytes("#quit", "utf8"):
            broadcast_message(msg, name + ": ")
        else:
            conn.send(bytes("#quit", "utf8"))
            conn.close()
            del clients[conn]
            broadcast_message(bytes("%s has left the chat." % name, "utf8"))
            break

def broadcast_message(msg, prefix=""): # prefix is for name identification.
    # this function will broadcast message to all clients connected
    for sock in clients:
        sock.send(bytes(prefix, "utf8") + msg)

if __name__ == "__main__":
    SOCK.listen(5) # Listens for 5 connections at max.
    print("Chat Server has Started !!")
    print("Waiting for connections !")
    ACCEPT_THREAD = Thread(target=accept_incoming_client_conn)
    ACCEPT_THREAD.start() # Starts the infinite loop.
    ACCEPT_THREAD.join()
    SOCK.close()

```

Code for client:

```

import tkinter
from socket import AF_INET, socket, SOCK_STREAM
from threading import Thread

```

```

def receive():
    # Function handles receiving the messages
    while True:
        try:
            msg = sock.recv(BUFSIZE).decode("utf8")
            msg_list.insert(tkinter.END, msg)
        except OSError: # Possibly client has left the chat.
            break

```

```

def send(event=None):
    # Function handles the sending message
    msg = my_msg.get()
    my_msg.set("") # Clears input field.
    sock.send(bytes(msg, "utf8"))

```

```

def on_closing(event=None):
    # This function will be called when the client wants to close the chat
    sock.close()
    top.quit()

```

```

top = tkinter.Tk()
top.title("Multi-User Chat ITCS 6166-8166")

```

```

messages_frame = tkinter.Frame(top)

```

```

my_msg = tkinter.StringVar() # For the messages to be sent.
my_msg.set("")
scrollbar = tkinter.Scrollbar(messages_frame) # To navigate through past messages.
msg_list = tkinter.Listbox(messages_frame, height=15, width=70,
                             yscrollcommand=scrollbar.set)
scrollbar.pack(side=tkinter.RIGHT, fill=tkinter.Y)
msg_list.pack(side=tkinter.LEFT, fill=tkinter.BOTH)
msg_list.pack()

```

```

messages_frame.pack()

```

```

button_label = tkinter.Label(top, text="Enter Message:")
button_label.pack()
entry_field = tkinter.Entry(top, textvariable=my_msg, foreground="Blue")

```

```
entry_field.bind("<Return>", send)
entry_field.pack()
send_button = tkinter.Button(top, text="Send", command=send)
send_button.pack()

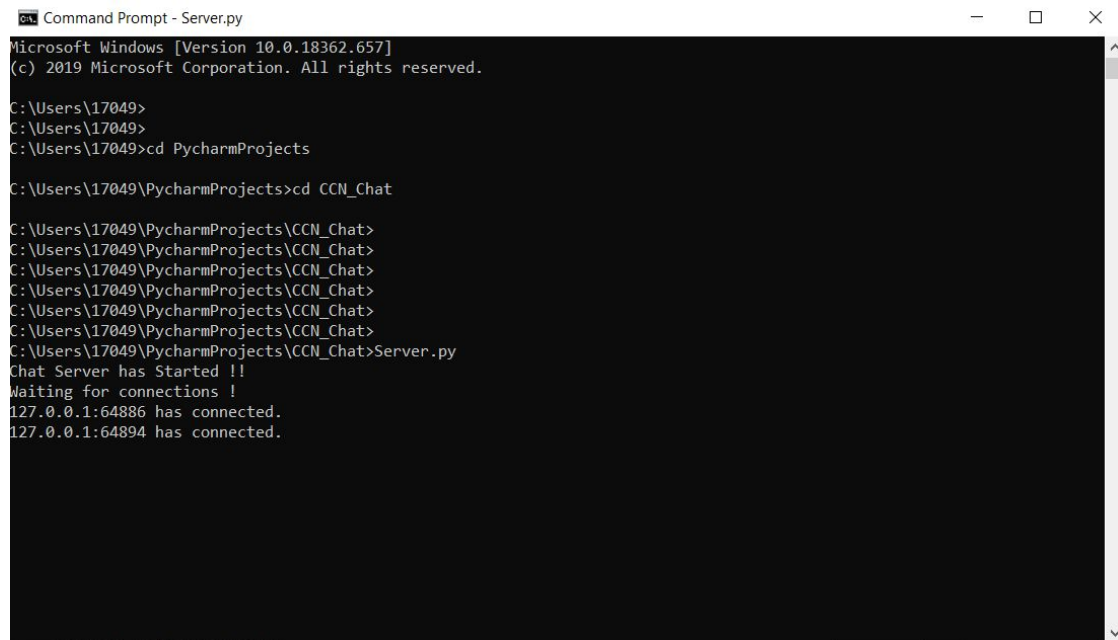
quit_button = tkinter.Button(top, text="Quit", command=on_closing)
quit_button.pack()

top.protocol("WM_DELETE_WINDOW", on_closing)

HOST = "127.0.0.1"
PORT = 5000
BUFFERSIZE = 1024
ADDR = (HOST, PORT)
sock = socket(AF_INET, SOCK_STREAM)
sock.connect(ADDR)

receive_thread = Thread(target=receive)
receive_thread.start()
tkinter.mainloop() # Starts Chat User Interface
```

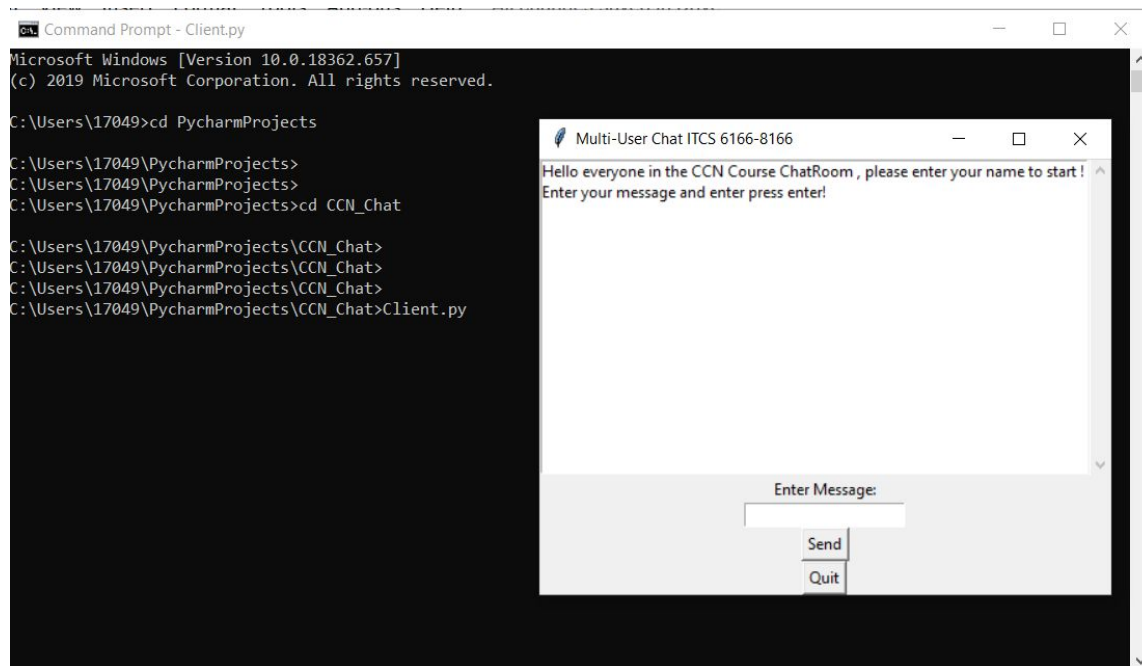
Connecting with the server:



```
Command Prompt - Server.py
Microsoft Windows [Version 10.0.18362.657]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\17049>
C:\Users\17049>
C:\Users\17049>cd PycharmProjects
C:\Users\17049\PycharmProjects>cd CCN_Chat
C:\Users\17049\PycharmProjects\CCN_Chat>
C:\Users\17049\PycharmProjects\CCN_Chat>
C:\Users\17049\PycharmProjects\CCN_Chat>
C:\Users\17049\PycharmProjects\CCN_Chat>
C:\Users\17049\PycharmProjects\CCN_Chat>
C:\Users\17049\PycharmProjects\CCN_Chat>
C:\Users\17049\PycharmProjects\CCN_Chat>Server.py
Chat Server has Started !!
Waiting for connections !
127.0.0.1:64886 has connected.
127.0.0.1:64894 has connected.
```

Connecting with the client:



Entering in the chat-room:

Multi-User Chat ITCS 6166-8166

Hello everyone in the CCN Course ChatRoom , please enter your name to start !
Enter your message and enter press enter!
Welcome Utkarsha! If you want to quit, press quit button
Devika from [127.0.0.1:64894] has joined the chat!
Divya from [127.0.0.1:64909] has joined the chat!
Divya: Hi guys
Utkarsha: Hey
Devika: Hello
Divya: I have some doubts in CCN HW2
Utkarsha: Oh no , please tell me
Devika: What is the doubt?

Enter Message:

Send

Quit

Multi-User Chat ITCS 6166-8166

Hello everyone in the CCN Course ChatRoom , please enter your name to start !
Enter your message and enter press enter!
Welcome Devika! If you want to quit, press quit button
Divya from [127.0.0.1:64909] has joined the chat!
Divya: Hi guys
Utkarsha: Hey
Devika: Hello
Divya: I have some doubts in CCN HW2
Utkarsha: Oh no , please tell me
Devika: What is the doubt?

Enter Message:

Send

Quit

