CS 461

Lab Assignment 5

Name: Gandhi Dhruv Vipulkumar

Institute ID: 202151053

Date: 15-10-2024

Q. Implement Distributed Chat Application

Server.py:

```python
import socket
import threading
import sqlite3

# Database connection
conn = sqlite3.connect('chat.db', check_same_thread=False)
cursor = conn.cursor()

# Creating tables if they don't exist
cursor.execute(
    '''CREATE TABLE IF NOT EXISTS users (username TEXT PRIMARY KEY,
password TEXT)''')
cursor.execute(
    '''CREATE TABLE IF NOT EXISTS messages (sender TEXT, recipient
TEXT, message TEXT)''')

# Dictionary to hold online clients and groups
clients = {}
groups = {}  # Dictionary to store groups and their members

# Function to broadcast messages to all members of a group

def broadcast_group(message, room, sender=None):
    if room in groups:
        for client_socket in groups[room]:
            if client_socket != sender:
                client_socket.send(f"Group {room}:
{message}".encode())

# Function to handle each client
```

```
def handle_client(client_socket, client_address):
    username = None

    # User authentication (login/signup)
    while True:
        try:
            choice = client_socket.recv(1024).decode()
            if choice == "signup":
                username, password = client_socket.recv(
                    1024).decode().split(':')
                try:
                    cursor.execute(
                        "INSERT INTO users (username, password)
VALUES (?, ?)", (username, password))
                    conn.commit()
                    client_socket.send(
                        "Signup successful! You can now start
chatting.".encode())
                except sqlite3.IntegrityError:
                    client_socket.send(
                        "Username already exists. Try a different
one.".encode())
            elif choice == "login":
                username, password = client_socket.recv(
                    1024).decode().split(':')
                cursor.execute(
                    "SELECT password FROM users WHERE username=?",
(username,))
                stored_password = cursor.fetchone()
                if stored_password and stored_password[0] ==
password:
                    client_socket.send(
                        "Login successful! Welcome to the
chat.".encode())

                    # Add the user to the clients dictionary
                    clients[username] = client_socket
                    break
                else:
                    client_socket.send("Invalid
credentials.".encode())
        except:
            client_socket.close()
            return

    # Handling messaging after login/signup
    while True:
        try:
            message = client_socket.recv(1024).decode()

            if message.startswith("/private"):
                _, recipient, msg = message.split(' ', 2)
                if recipient in clients:
                    clients[recipient].send(
                        f"Private from {username}: {msg}".encode())
                    cursor.execute(
                        "INSERT INTO messages (sender, recipient,
message) VALUES (?, ?, ?)", (username, recipient, msg))
                    conn.commit()
                else:
                    client_socket.send("User not online.".encode())

            elif message.startswith("/group"):
                _, room, msg = message.split(' ', 2)

                if room not in groups:
                    groups[room] = []
                if client_socket not in groups[room]:
                    groups[room].append(client_socket)

                # Broadcast the message to all group members
                broadcast_group(f"{username}: {msg}",
                                room, sender=client_socket)

            elif message == "/logout":
                client_socket.send("You have logged out.".encode())
                client_socket.close()

                # Remove user from clients and groups when they
logout
                if username in clients:
                    del clients[username]
                for group in groups.values():
                    if client_socket in group:
                        group.remove(client_socket)
                break
        except:
            # Handle disconnection
            client_socket.close()
            if username in clients:
                del clients[username]
            for group in groups.values():
```

```python
                if client_socket in group:
                    group.remove(client_socket)
            break

# Main server function to accept incoming connections

def start_server():
    server_socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
    server_socket.bind(('127.0.0.1', 12345))
    server_socket.listen(5)
    print("Server is listening...")

    while True:
        client_socket, client_address = server_socket.accept()
        print(f"New connection from {client_address}")
        thread = threading.Thread(
            target=handle_client, args=(client_socket,
client_address))
        thread.start()

if __name__ == "__main__":
    start_server()
```

Client.py

```python
import socket
import threading

# Function to receive messages

def receive_messages(client_socket):
    while True:
        try:
            message = client_socket.recv(1024).decode()
            print(message)
        except Exception as e:
            print(f"An error occurred while receiving message: {e}")
            client_socket.close()
            break

# Function to handle sending messages
```

```
def send_messages(client_socket):
    while True:
        message = input()
        if message.startswith("/private"):
            recipient = input("Recipient: ")
            msg = input("Message: ")
            client_socket.send(f"/private {recipient}
{msg}".encode())
        elif message.startswith("/group"):
            room = input("Room name: ")
            msg = input("Message: ")
            client_socket.send(f"/group {room} {msg}".encode())
        elif message == "/logout":
            client_socket.send(message.encode())
            break

# Function to start the client

def start_client():
    client_socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
    client_socket.connect(('127.0.0.1', 12345))

    print("Welcome to the chat app!")
    auth_choice = input(
        "Do you want to login or signup? (login/signup): ").strip()
    client_socket.send(auth_choice.encode())

    if auth_choice == "signup":
        username = input("Choose a username: ")
        password = input("Choose a password: ")
        client_socket.send(f"{username}:{password}".encode())
    elif auth_choice == "login":
        username = input("Username: ")
        password = input("Password: ")
        client_socket.send(f"{username}:{password}".encode())

    # Receive confirmation message (Signup/Login success)
    response = client_socket.recv(1024).decode()
    print(response)

    if "successful" in response:
        # Once signup/login is successful, allow sending and
receiving messages
        print("Welcome to the chat room. You can now send
messages!")
```

```
        # Start a thread for receiving messages
        receive_thread = threading.Thread(
            target=receive_messages, args=(client_socket,))
        receive_thread.start()

        # Handle sending messages
        send_messages(client_socket)

    # Close the socket connection after logout
    client_socket.close()

if __name__ == "__main__":
    start_client()
```

Code Explanation:

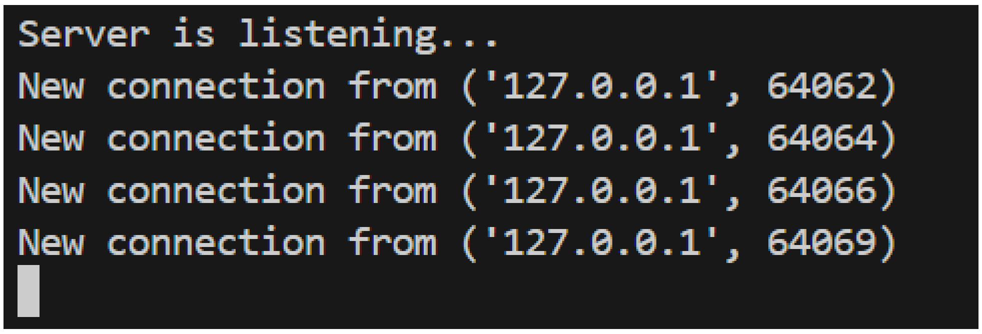
- **Server:**
  - The server listens for client connections.
  - When a client connects, it spawns a new thread to handle communication with that client.
  - It broadcasts messages to all connected clients except the sender.
  - If a client disconnects, it removes that client from the list.
- **Client:**
  - The client connects to the server and spawns a thread to listen for incoming messages.
  - The user can send messages to the server, which will be broadcasted to other clients.
  - If the user types exit, the client disconnects from the server.

Key Features:

1. User Authentication (Signup/Login)
2. Private Chat (Direct Messaging)
3. Group Chatrooms
4. Multi-server Support (Distributed Setup)

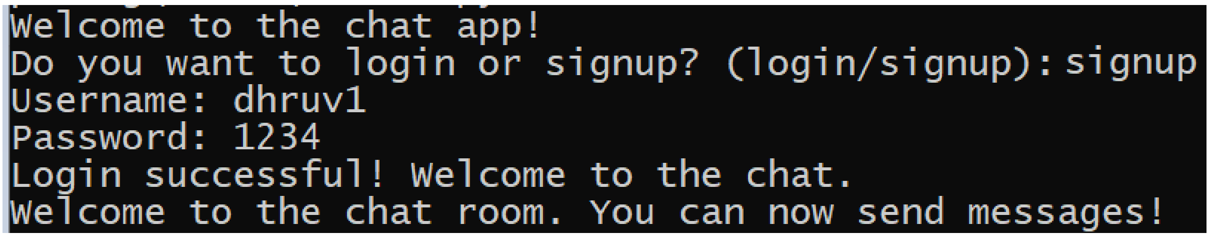
Testing Phase:

1) Start the server:

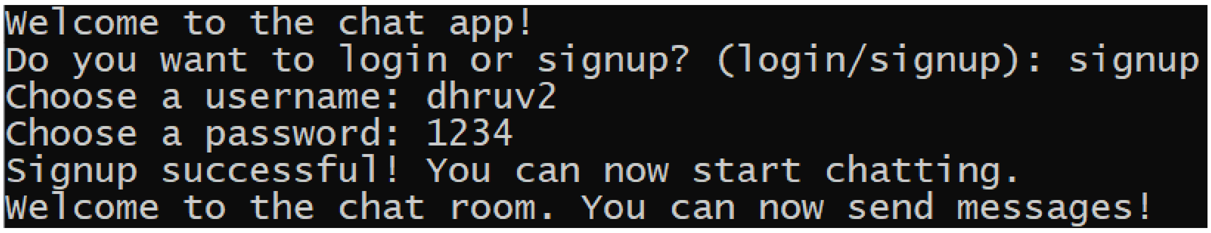


```
Server is listening...
New connection from ('127.0.0.1', 64062)
New connection from ('127.0.0.1', 64064)
New connection from ('127.0.0.1', 64066)
New connection from ('127.0.0.1', 64069)
```

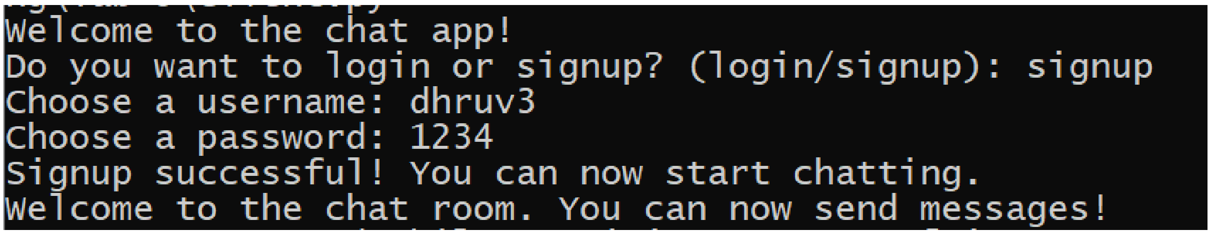
2) Signup and Login 4 different users dhruv1-4:



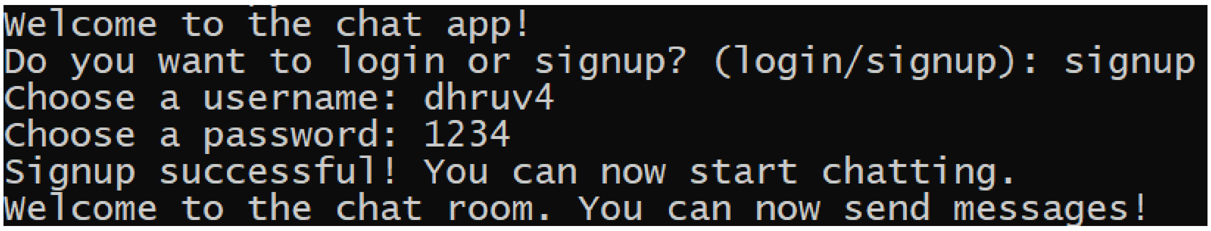
```
Welcome to the chat app!
Do you want to login or signup? (login/signup):signup
Username: dhruv1
Password: 1234
Login successful! Welcome to the chat.
Welcome to the chat room. You can now send messages!
```



```
Welcome to the chat app!
Do you want to login or signup? (login/signup): signup
Choose a username: dhruv2
Choose a password: 1234
Signup successful! You can now start chatting.
Welcome to the chat room. You can now send messages!
```



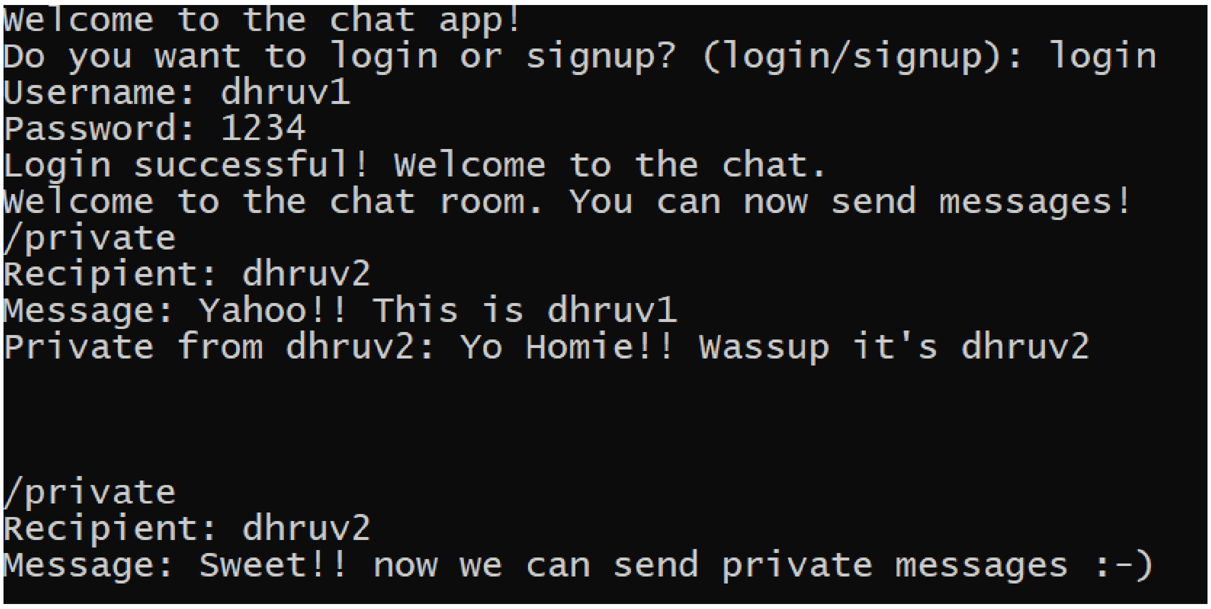
```
Welcome to the chat app!
Do you want to login or signup? (login/signup): signup
Choose a username: dhruv3
Choose a password: 1234
Signup successful! You can now start chatting.
Welcome to the chat room. You can now send messages!
```



```
Welcome to the chat app!
Do you want to login or signup? (login/signup): signup
Choose a username: dhruv4
Choose a password: 1234
Signup successful! You can now start chatting.
Welcome to the chat room. You can now send messages!
```

3) dhruv1 send private message to dhruv2

From dhruv1:

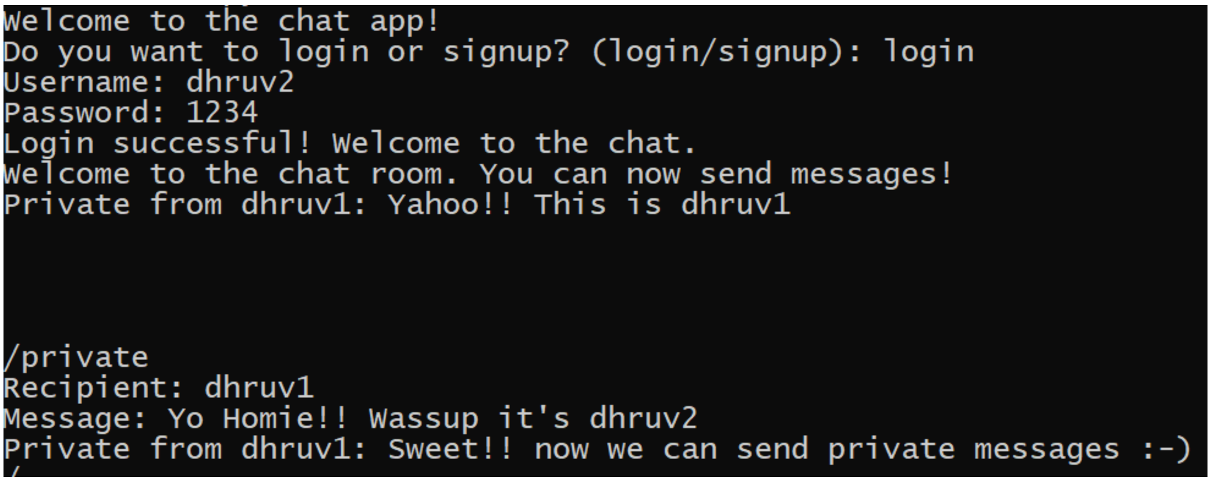


```
Welcome to the chat app!
Do you want to login or signup? (login/signup): login
Username: dhruv1
Password: 1234
Login successful! Welcome to the chat.
Welcome to the chat room. You can now send messages!
/private
Recipient: dhruv2
Message: Yahoo!! This is dhruv1
Private from dhruv2: Yo Homie!! Wassup it's dhruv2



/private
Recipient: dhruv2
Message: Sweet!! now we can send private messages :-)
```

From dhruv2:



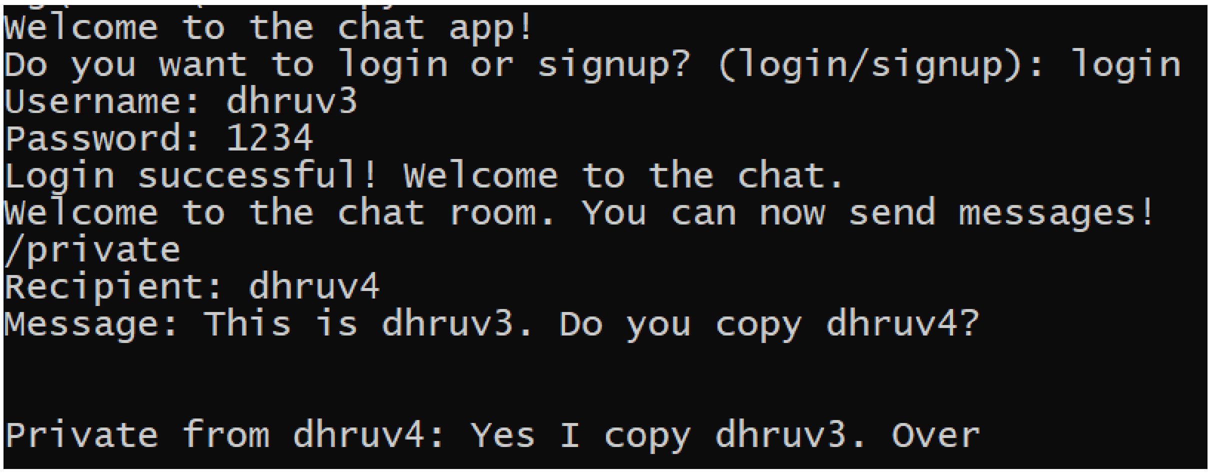
```
Welcome to the chat app!
Do you want to login or signup? (login/signup): login
Username: dhruv2
Password: 1234
Login successful! Welcome to the chat.
Welcome to the chat room. You can now send messages!
Private from dhruv1: Yahoo!! This is dhruv1



/private
Recipient: dhruv1
Message: Yo Homie!! Wassup it's dhruv2
Private from dhruv1: Sweet!! now we can send private messages :-)
```

4) sending message from dhruv3 to dhruv4:

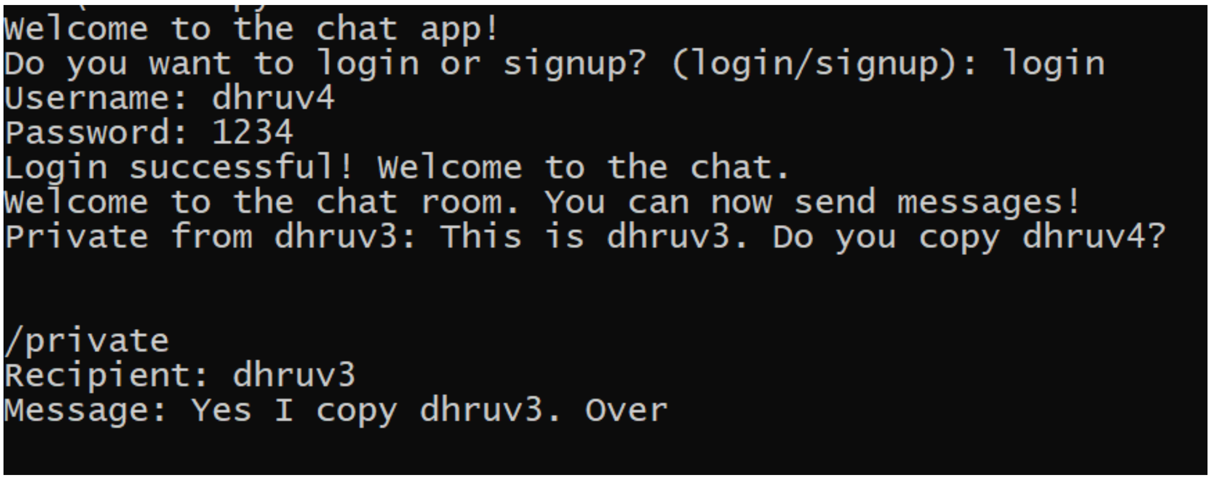
(From dhruv3)



```
Welcome to the chat app!
Do you want to login or signup? (login/signup): login
Username: dhruv3
Password: 1234
Login successful! Welcome to the chat.
Welcome to the chat room. You can now send messages!
/private
Recipient: dhruv4
Message: This is dhruv3. Do you copy dhruv4?


Private from dhruv4: Yes I copy dhruv3. Over
```

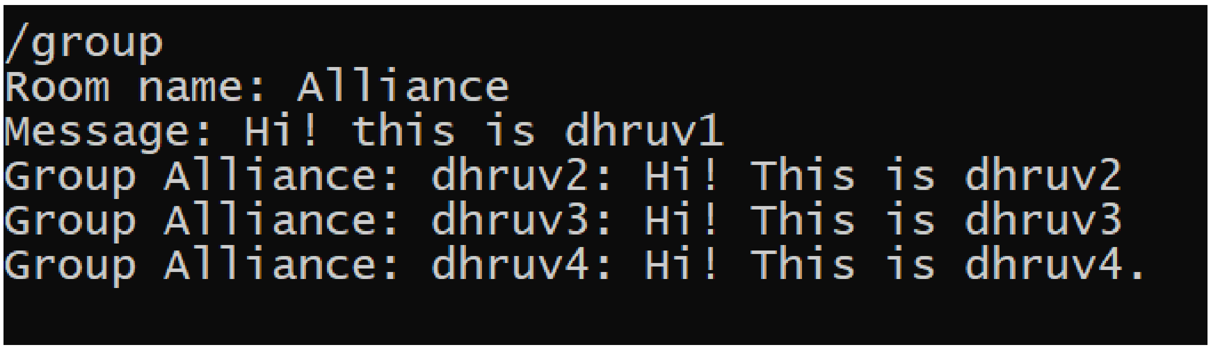
(From dhruv4)



```
Welcome to the chat app!
Do you want to login or signup? (login/signup): login
Username: dhruv4
Password: 1234
Login successful! Welcome to the chat.
Welcome to the chat room. You can now send messages!
Private from dhruv3: This is dhruv3. Do you copy dhruv4?


/private
Recipient: dhruv3
Message: Yes I copy dhruv3. Over
```

5) Create a group named “Alliance” and send messages:



```
/group
Room name: Alliance
Message: Hi! this is dhruv1
Group Alliance: dhruv2: Hi! This is dhruv2
Group Alliance: dhruv3: Hi! This is dhruv3
Group Alliance: dhruv4: Hi! This is dhruv4.
```

Conclusion: Successfully implemented chat application in python.