CN ASSIGNMENT-1 REPORT

TOPIC: Chat Application: Develop a. chat application where multiple clients can connect to a server and communicate with each other using sockets

TEAM:

YAMAN GUPTA (PES2UG21CS619)

VIDULA.L.S.(PES2UG21CS602)

ABSTRACT:

The development of a chat application where multiple clients can connect to a server and communicate with each other using sockets involves the use of socket programming concepts.

Socket programming enables communication between different devices over a network. The chat application uses a socket as a software interface that provides a connection between the clients and the server.

To create the chat application, the server must be first set up to receive connections from multiple clients. Once the server is set up, clients can connect to it using their unique IP address and port number. The server then creates a separate thread for each client to handle incoming and outgoing messages.

Once a client is connected to the server, it can send messages to all other clients connected to the server. The server acts as a mediator, receiving messages from one client and transmitting them to the other clients. The messages are

transmitted in real-time, allowing for seamless communication between the clients.

In conclusion, the development of a chat application using socket programming enables multiple clients to connect to a server and communicate with each other in real-time. The application can be further improved with the addition of advanced features such as user authentication, message encryption, and message history.

CLIENT CODE:

```
# import required modules
import socket
import threading
import tkinter as tk
from tkinter import scrolledtext
from tkinter import messagebox

HOST = '192.168.163.15'
PORT = 2001


DARK_GREY = '#121212'
MEDIUM_GREY = '#1F1B24'
OCEAN_BLUE = '#464EB8'
WHITE = "white"
FONT = ("Helvetica", 17)
BUTTON_FONT = ("Helvetica", 15)
SMALL_FONT = ("Helvetica", 13)

# Creating a socket object
# AF_INET: we are going to use IPv4 addresses
# SOCK_STREAM: we are using TCP packets for communication
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)


def add_message(message):
    message_box.config(state=tk.NORMAL)
    message_box.insert(tk.END, message + '\n')
    message_box.config(state=tk.DISABLED)


def connect():
    # try except block
    try:

        # Connect to the server
        client.connect((HOST, PORT))
        print("Successfully connected to server")
        add_message("[SERVER] Successfully connected to the server")
    except:
        messagebox.showerror("Unable to connect to server", f"Unable to
connect to server {HOST} {PORT}")

    username = username_textbox.get()
    if username != '':
        client.sendall(username.encode())
    else:
        messagebox.showerror("Invalid username", "Username cannot be
empty")

    threading.Thread(target=listen_for_messages_from_server,
args=(client,)).start()

    username_textbox.config(state=tk.DISABLED)
    username_button.config(state=tk.DISABLED)


def send_message():
    message = message_textbox.get()
    if message != '':
        client.sendall(message.encode())
        message_textbox.delete(0, len(message))
    else:
        messagebox.showerror("Empty message", "Message cannot be empty")


root = tk.Tk()
root.geometry("600x600")
root.title("Messenger Client")
root.resizable(False, False)

root.grid_rowconfigure(0, weight=1)
root.grid_rowconfigure(1, weight=4)
root.grid_rowconfigure(2, weight=1)

top_frame = tk.Frame(root, width=600, height=100, bg=DARK_GREY)
top_frame.grid(row=0, column=0, sticky=tk.NSEW)

middle_frame = tk.Frame(root, width=600, height=400, bg=MEDIUM_GREY)
middle_frame.grid(row=1, column=0, sticky=tk.NSEW)

bottom_frame = tk.Frame(root, width=600, height=100, bg=DARK_GREY)
bottom_frame.grid(row=2, column=0, sticky=tk.NSEW)

username_label = tk.Label(top_frame, text="Enter username:", font=FONT,
bg=DARK_GREY, fg=WHITE)
username_label.pack(side=tk.LEFT, padx=10)

username_textbox = tk.Entry(top_frame, font=FONT, bg=MEDIUM_GREY, fg=WHITE,
width=23)
username_textbox.pack(side=tk.LEFT)

username_button = tk.Button(top_frame, text="Join", font=BUTTON_FONT,
bg=OCEAN_BLUE, fg=WHITE, command=connect)
username_button.pack(side=tk.LEFT, padx=15)

message_textbox = tk.Entry(bottom_frame, font=FONT, bg=MEDIUM_GREY,
fg=WHITE, width=38)
message_textbox.pack(side=tk.LEFT, padx=10)

message_button = tk.Button(bottom_frame, text="Send", font=BUTTON_FONT,
bg=OCEAN_BLUE, fg=WHITE, command=send_message)
message_button.pack(side=tk.LEFT, padx=10)
```

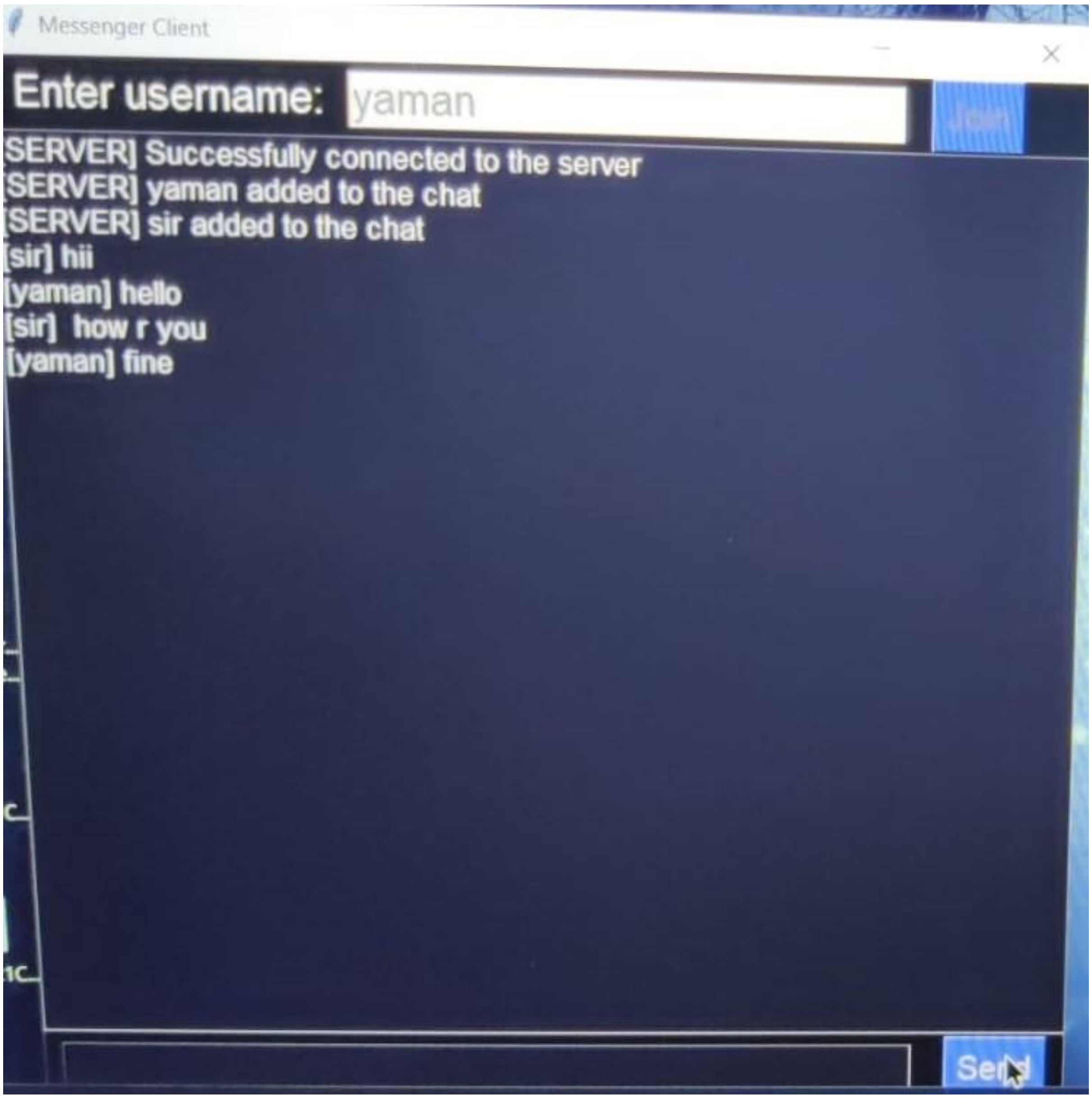
```
message_box = scrolledtext.ScrolledText(middle_frame, font=SMALL_FONT,
bg=MEDIUM_GREY, fg=WHITE, width=67, height=26.5)
message_box.config(state=tk.DISABLED)
message_box.pack(side=tk.TOP)


def listen_for_messages_from_server(client):
    while 1:

        message = client.recv(2048).decode('utf-8')
        if message != '':
            username = message.split("~")[0]
            content = message.split('~')[1]

            add_message(f"[{username}] {content}")

        else:
            messagebox.showerror("Error", "Message recevied from client is
empty")


# main function
def main():
    root.mainloop()


if __name__ == '__main__':
    main()
```

SERVER CODE:

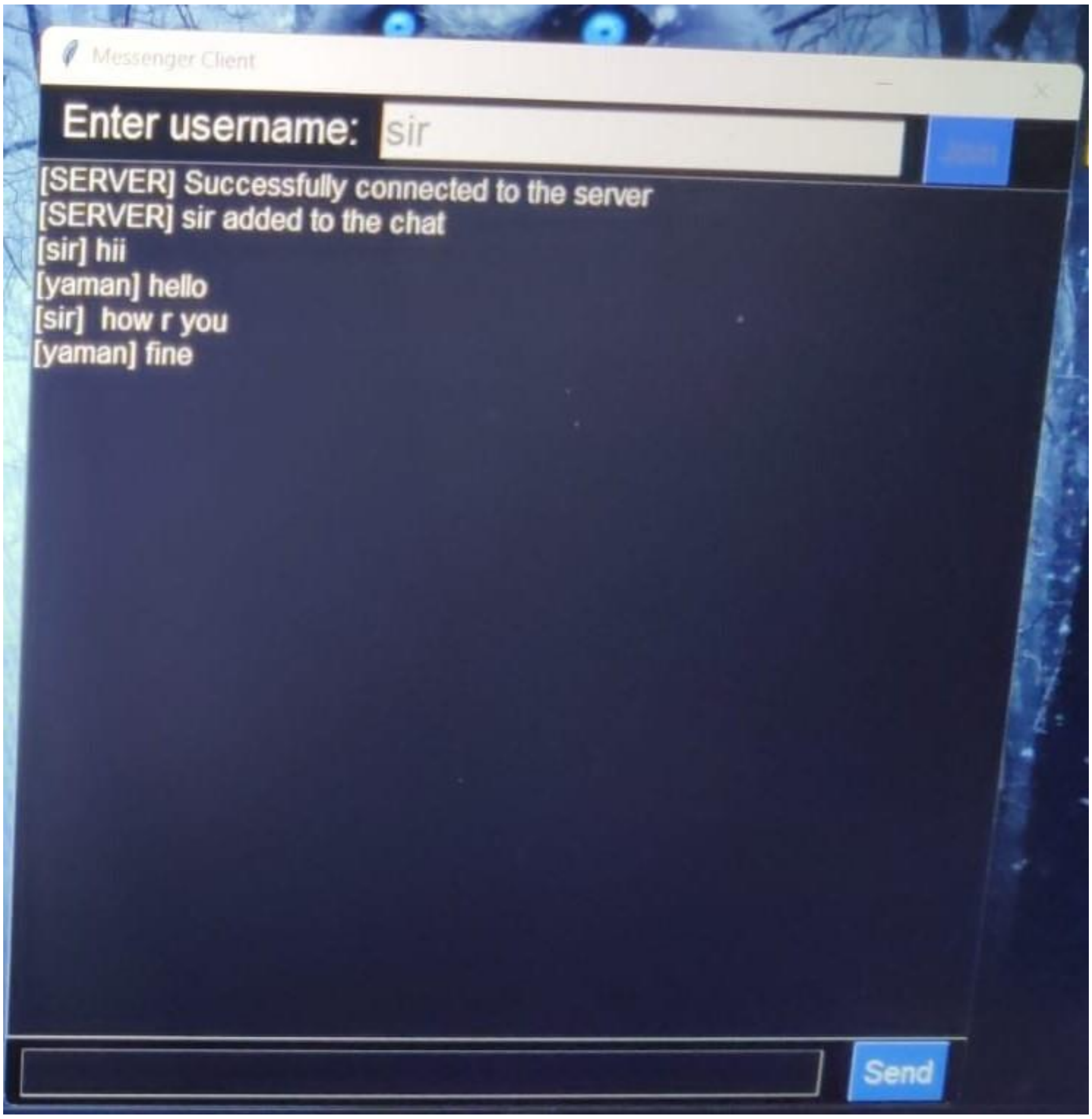
```
# Import required modules
import socket
import threading

HOST = '106.216.236.78'
PORT = 2002

LISTENER_LIMIT = 5
active_clients = []  # List of all currently connected users


# Function to listen for upcoming messages from a client
def listen_for_messages(client, username):
    while 1:

        message = client.recv(2048).decode('utf-8')
        if message != '':

            final_msg = username + '~' + message
            send_messages_to_all(final_msg)

        else:
            print(f"The message send from client {username} is empty")


# Function to send message to a single client
def send_message_to_client(client, message):
    client.sendall(message.encode())


# Function to send any new message to all the clients that
# are currently connected to this server
def send_messages_to_all(message):
    for user in active_clients:
        send_message_to_client(user[1], message)


# Function to handle client
def client_handler(client):
    # Server will listen for client message that will
    # Contain the username
    while 1:

        username = client.recv(2048).decode('utf-8')
        if username != '':
            active_clients.append((username, client))
            prompt_message = "SERVER~" + f"{username} added to the
chat"
            send_messages_to_all(prompt_message)
            break
        else:
            print("Client username is empty")

    threading.Thread(target=listen_for_messages, args=(client,
username,)).start()


# Main function
def main():
    # Creating the socket class object
    # AF_INET: we are going to use IPv4 addresses
    # SOCK_STREAM: we are using TCP packets for communication
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    # Creating a try catch block
    try:
        # Provide the server with an address in the form of
        # host IP and port
        server.bind((HOST, PORT))
        print(f"Running the server on {HOST} {PORT}")
    except:
        print(f"Unable to bind to host {HOST} and port {PORT}")

    # Set server limit
    server.listen(LISTENER_LIMIT)

    # This while loop will keep listening to client connections
    while 1:
        client, address = server.accept()
        print(f"Successfully connected to client {address[0]}
{address[1]}")

        threading.Thread(target=client_handler, args=(client,)).start()
```

```
if __name__ == '__main__':
    main()
```

OUTPUT:



```
[SERVER] Successfully connected to the server
[SERVER] yaman added to the chat
[SERVER] sir added to the chat
[sir] hii
[yaman] hello
[sir]  how r you
[yaman] fine
```



```
[SERVER] Successfully connected to the server
[SERVER] sir added to the chat
[sir] hii
[yaman] hello
[sir]  how r you
[yaman] fine
```