Vidyalankar Institute of Technology Accredited A+ by NAAC	**DEPARTMENT OF COMPUTER ENGINEERING**

Mini Project Report

Semester	S.E. Semester IV – Computer Engineering
Subject	Skill Base Lab Course: Python Programming (CSL405)
Subject Professor In-charge	Prof. Swapnil S. Sonawane
Assisting Teachers	Prof. Swapnil S. Sonawane

Roll Numbers	Name of Students
21102A0014	Deep Salunkhe
21102A0003	Omkar Patil
21102A0005	Pranav Redij
21102A0037	Sukant Thombare

Name of the Project: Chat Application

Project Description:We have built a chat application project that allows multiple clients to connect to a server using sockets and exchange messages in real-time. The project is divided into two parts, the front-end and the back-end.

The front-end is developed using the tkinter module in Python, which provides a graphical user interface for the chat application. It allows the user to enter their name and message and send it to the server using a send button. The chat box displays all the messages received from the server in real-time.

The back-end is developed using Flask and the threading module in Python. It creates a server that listens for incoming client connections using sockets. Once a client connects, the server creates a new thread to handle the client connection. The server then broadcasts any message received from a client to all the other connected clients.

Project Code:

Client:

```
import tkinter as tk
from tkinter import ttk
import threading
import socket

# Create a socket object
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Define host and port for server
HOST = 'localhost'
PORT = 5000

# Connect to the server
client_socket.connect((HOST, PORT))

# Define function to send message to server
def send_message():
    message = message_entry.get()
    sender = sender_entry.get()
    data = {'message': message, 'sender': sender}
    client_socket.sendall(str(data).encode())
    message_entry.delete(0, tk.END)
    if not sender_entry['state'] == 'disabled':
        sender_entry.config(state='disabled')

# Create the GUI
root = tk.Tk()
root.title('Chat App')
root.geometry('500x500')
root.configure(bg='black')

# Create widgets
sender_label = ttk.Label(root, text='Name:', foreground='white',
background='black')
sender_entry = ttk.Entry(root)
message_label = ttk.Label(root, text='Message:', foreground='white',
background='black')
message_entry = ttk.Entry(root)
send_button = ttk.Button(root, text='Send', command=send_message)
chat_box = tk.Text(root, height=20, width=60, foreground='white',
background='black')
scrollbar = ttk.Scrollbar(root, command=chat_box.yview)
chat_box.config(yscrollcommand=scrollbar.set, state='disabled')

# Add widgets to grid
sender_label.grid(row=0, column=0, pady=10, padx=10)
```

```
sender_entry.grid(row=0, column=1, pady=10, padx=10)
message_label.grid(row=1, column=0, pady=10, padx=10)
message_entry.grid(row=1, column=1, pady=10, padx=10)
send_button.grid(row=2, column=1, pady=10, padx=10, sticky='e')
chat_box.grid(row=3, column=0, columnspan=2, pady=10, padx=10)
scrollbar.grid(row=3, column=2, sticky='ns', pady=10)

# Define function to update chat box with received messages
def update_chat_box(message):
    chat_box.config(state='normal')
    chat_box.insert(tk.END, message + '\n')
    chat_box.config(state='disabled')

# Define function to receive messages from server
def receive_messages():
    while True:
        data = client_socket.recv(1024)
        message = data.decode()
        update_chat_box(message)

# Create a new thread to receive messages from server
receive_thread = threading.Thread(target=receive_messages)
receive_thread.start()

# Run the GUI
root.mainloop()
```

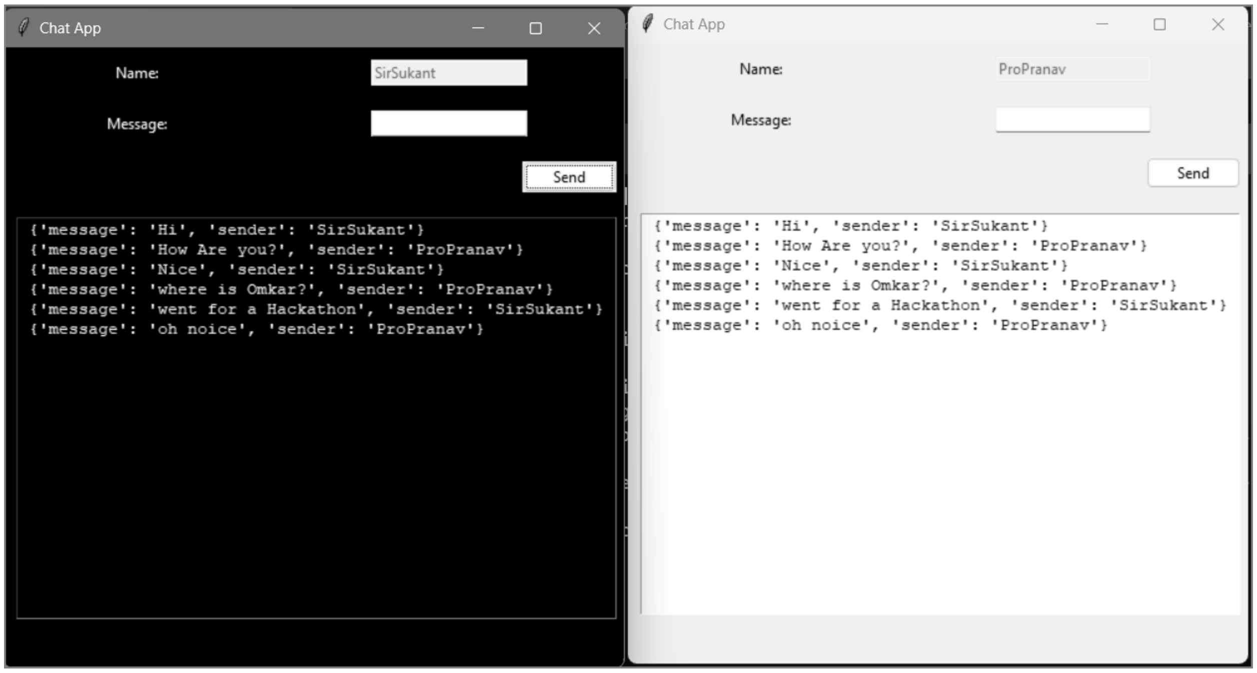
Server:

```
import socket
from threading import Thread
from flask import Flask, render_template

# Create a Flask app instance
app = Flask(__name__)
app.config['SECRET_KEY'] = 'secret!'

# Define host and port for server
HOST = 'localhost'
PORT = 5000

# Create a socket object
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Set socket option to allow re-use of the address and port
server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

# Bind the socket to the host and port
```

```
server_socket.bind((HOST, PORT))

# Listen for incoming client connections
server_socket.listen()

# Define list of connected clients
clients = []

# Define function to broadcast message to all connected clients
def broadcast_message(sender, message):
    for client in clients:
        if client != sender:
            try:
                # Send the message to the client
                client.sendall(f'{sender}: {message}'.encode())
            except socket.error:
                # If there's an error sending the message, remove the client from
the list of connected clients
                clients.remove(client)
                print(f'Client {client} disconnected')

# Define function to handle client connection
def handle_client(client_socket, addr):
    print(f'Client connected from {addr}')

    # Add the client socket to the list of connected clients
    clients.append(client_socket)

    while True:
        # Receive data from the client
        data = client_socket.recv(1024)

        # If there's no data, the client has disconnected
        if not data:
            # Remove the client from the list of connected clients and close the
connection
            clients.remove(client_socket)
            client_socket.close()
            break

        # Decode the received data into a string
        message = data.decode()

        # Broadcast the message to all connected clients
        broadcast_message(addr, message)

# Define a route for the index page
```

```python
@app.route('/')
def index():
    return render_template('index.html')

if __name__ == '__main__':
    print(f'Server running on {HOST}:{PORT}')

    # Listen for incoming client connections
    while True:
        # Accept an incoming client connection and create a new thread to handle 
the connection
        client_socket, addr = server_socket.accept()
        client_thread = Thread(target=handle_client, args=(client_socket, addr))
        client_thread.start()
```

Result/ Output:



```
{'message': 'Hi', 'sender': 'SirSukant'}
{'message': 'How Are you?', 'sender': 'ProPranav'}
{'message': 'Nice', 'sender': 'SirSukant'}
{'message': 'where is Omkar?', 'sender': 'ProPranav'}
{'message': 'went for a Hackathon', 'sender': 'SirSukant'}
{'message': 'oh noice', 'sender': 'ProPranav'}
```

```
{'message': 'Hi', 'sender': 'SirSukant'}
{'message': 'How Are you?', 'sender': 'ProPranav'}
{'message': 'Nice', 'sender': 'SirSukant'}
{'message': 'where is Omkar?', 'sender': 'ProPranav'}
{'message': 'went for a Hackathon', 'sender': 'SirSukant'}
{'message': 'oh noice', 'sender': 'ProPranav'}
```

References:

https://python.plainenglish.io/create-a-basic-lan-chat-room-with-python-f334776bf70c
https://www.geeksforgeeks.org/simple-chat-room-using-python/
