VIDHI ROHIRA

SY BTECH

COMPUTER ENGINEERING

PD LAB-3

231071052

PD LAB 3

AIM:- Develop messaging application using socket programming and tkinter.

THEORY:-

TKINKTER:- Tkinter is a Python library for creating graphical user interfaces (GUIs). It wraps the Tcl/Tk toolkit and offers a range of features:

- **Widgets**: Components like labels, buttons, and text boxes.
- **Layout Management**: Tools like pack, grid, and place for arranging widgets.
- **Event Handling**: Bind functions to user actions like clicks and key presses.
- **Geometry Management**: Control the size and placement of windows and widgets.
- **Customization**: Modify the appearance and behavior of widgets, such as colors and fonts.

It’s included with Python and is ideal for building desktop applications.

Q] WHAT IS GUI?

A GUI, or Graphical User Interface, is a type of user interface that allows users to interact with software through graphical elements rather than text-based commands. Instead of typing commands or navigating through text menus, users interact with visual elements such as:

- **Windows**: Contain and organize content and controls.
- **Icons**: Represent programs, files, or functions with visual symbols.
- **Buttons**: Trigger actions or commands when clicked.
- **Menus**: Offer a list of options or commands.
- **Text Boxes**: Allow users to input or edit text.
- **Sliders and Checkboxes**: Provide controls for adjusting settings or making selections.

GUIs make software more accessible and user-friendly by leveraging visual and interactive elements to simplify interactions and enhance usability. They are commonly used in operating systems, applications, and web interfaces.

Q] WHAT IS A MESSAGING APPLICATION?

A messaging application is software designed for exchanging messages and multimedia over the internet. It allows users to send text, images, videos, and voice messages, and often includes features like voice and video calls, group chats, and file sharing. Messaging apps enhance communication through real-time interactions and may offer additional features such as notifications, encryption for security, and integration with other services. Examples include WhatsApp, Telegram, and Slack.

Q] WHAT IS SOCKET PROGRAMMING?

Socket programming is a method used in network communication to enable data exchange between computers over a network. It involves creating and managing network connections through "sockets," which are endpoints for sending and receiving data. In socket programming, a server application listens for incoming connections on a specific port and IP address, while client applications initiate connections to the server.

Q] HOW TO DEVELOP MESSAGING APPLICATION USING SOCKET PROGRAMMING AND TKINKTER?

To develop a basic messaging application using socket programming and Tkinter, you need to create both a server and client application. The server, implemented in Python, listens for incoming connections on a specific port and handles multiple clients using threads. It receives messages from clients and broadcasts them to all connected clients. The client application connects to the server and uses Tkinter to provide a graphical user interface for sending and receiving messages. The GUI includes a text display area for incoming messages, an entry field for typing messages, and a button to send messages. The client handles incoming messages in a separate thread to ensure the GUI remains responsive. By running the server and client scripts, you can simulate a simple messaging environment where multiple clients can communicate through a central server.

CODE, OUTPUTS AND SCREENSHOTS :-

CODE FOR SERVER:-

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

class ServerApp:
    def __init__(self, root):
        self.root = root
        self.root.title("TCP Server")

        # Set up GUI components
        self.text_area = scrolledtext.ScrolledText(self.root, wrap=tk.WORD,
height=20, width=80)
        self.text_area.pack(padx=10, pady=10)

        self.message_label = tk.Label(self.root, text="Message to Client:")
        self.message_label.pack(padx=10, pady=5)

        self.message_entry = tk.Entry(self.root, width=80)
        self.message_entry.pack(padx=10, pady=5)

        self.send_button = tk.Button(self.root, text="Send",
command=self.send_to_client)
        self.send_button.pack(padx=10, pady=5)

        self.clients = {}
        self.start_server()

    def start_server(self):
        threading.Thread(target=self.run_server, daemon=True).start()

    def run_server(self):
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as
server_socket:
            server_socket.bind(('localhost', 12345))
            server_socket.listen(5)
            self.log("Server started on port 12345")

            while True:
                client_socket, addr = server_socket.accept()
                self.log(f"Connection from {addr}")
                self.clients[client_socket] = addr
                threading.Thread(target=self.handle_client,
args=(client_socket,), daemon=True).start()

    def handle_client(self, client_socket):
        while True:
            try:
                message = client_socket.recv(1024).decode()
                if not message:
                    break
                addr = self.clients.get(client_socket, "Unknown")
                self.log(f"Received from {addr}: {message}")
            except ConnectionResetError:
                break
            except Exception as e:
                self.log(f"An error occurred: {e}")
                break

        client_socket.close()
        self.log(f"Client {self.clients.get(client_socket, 'Unknown')}
disconnected")
        del self.clients[client_socket]

    def send_to_client(self):
        message = self.message_entry.get()
        for client_socket in self.clients.keys():
            try:
                client_socket.sendall(message.encode())
            except Exception as e:
                self.log(f"Error sending to client: {e}")

    def log(self, message):
        self.text_area.insert(tk.END, message + "\n")
        self.text_area.yview(tk.END)

if __name__ == "__main__":
    root = tk.Tk()
    app = ServerApp(root)
    root.mainloop()
```

EXPLANATION:-

The script creates a TCP server with a Tkinter GUI. The `ServerApp` class sets up the GUI with a text area for logging, an entry field for composing messages, and a button to send messages. The `start_server` method initializes the server to listen for incoming connections in a separate thread. The `handle_client` method processes messages from clients, while `send_to_client` broadcasts messages to all connected clients.

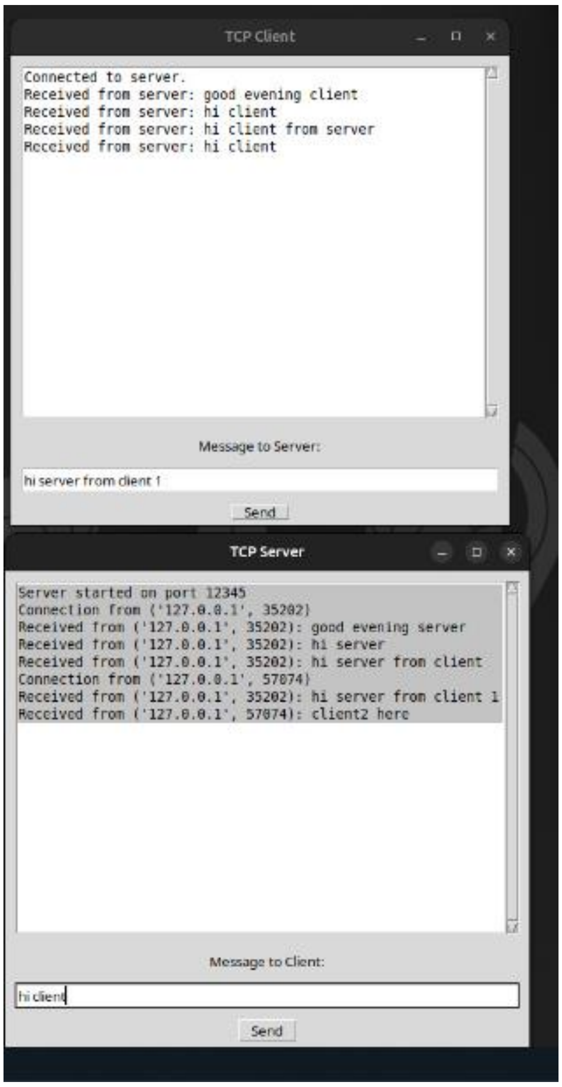
CODE FOR CLIENT:-

```python
import socket
import threading
import tkinter as tk
from tkinter import scrolledtext

class ClientApp:
    def __init__(self, root):
        self.root = root
        self.root.title("TCP Client")

        # Set up GUI components
        self.text_area = scrolledtext.ScrolledText(self.root, wrap=tk.WORD,
height=20, width=80)
        self.text_area.pack(padx=10, pady=10)

        self.message_label = tk.Label(self.root, text="Message to Server:")
        self.message_label.pack(padx=10, pady=5)

        self.message_entry = tk.Entry(self.root, width=80)
        self.message_entry.pack(padx=10, pady=5)

        self.send_button = tk.Button(self.root, text="Send",
command=self.send_message)
        self.send_button.pack(padx=10, pady=5)

        self.server_ip = 'localhost'
        self.server_port = 12345

        # Connect to the server and start receiving messages
        self.connect_to_server()

    def connect_to_server(self):
        self.client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.client_socket.connect((self.server_ip, self.server_port))
        self.log("Connected to server.")

        # Start a thread to listen for incoming messages from the server
        threading.Thread(target=self.receive_from_server, daemon=True).start()

    def receive_from_server(self):
        while True:
            try:
                response = self.client_socket.recv(1024).decode()
                if not response:
                    self.log("Server disconnected.")
                    break
                self.log(f"Received from server: {response}")
            except ConnectionResetError:
                self.log("Server disconnected abruptly.")
                break
            except Exception as e:
                self.log(f"An error occurred: {e}")
                break

        self.client_socket.close()

    def send_message(self):
        message = self.message_entry.get()
        try:
            self.client_socket.sendall(message.encode())
        except Exception as e:
            self.log(f"An error occurred: {e}")

    def log(self, message):
        self.text_area.insert(tk.END, message + "\n")
        self.text_area.yview(tk.END)

if __name__ == "__main__":
    root = tk.Tk()
    app = ClientApp(root)
    root.mainloop()
```

EXPLANATION:-

This script creates a TCP client with a Tkinter GUI. The `ClientApp` class sets up the GUI with a text area for displaying messages, an entry field for composing messages, and a button to send messages. It connects to a server, listens for incoming messages in a separate thread, and updates the GUI with received messages. The `send_message` method sends typed messages to the server, and the `log` method displays messages in the text area.

OUTPUT:-



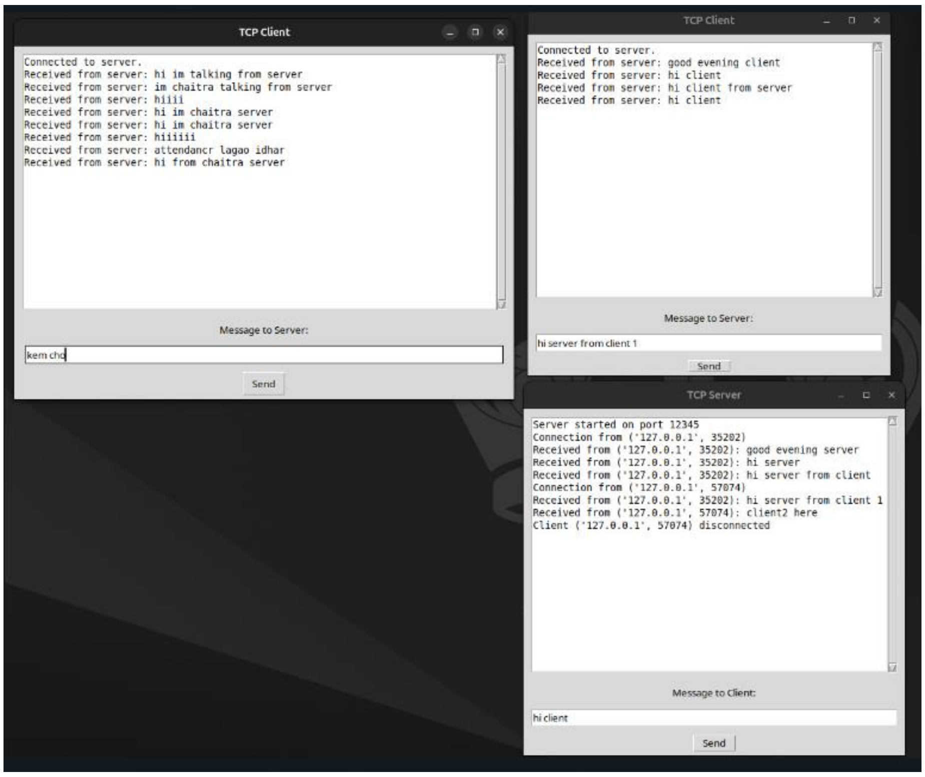
```
Connected to server.
Received from server: good evening client
Received from server: hi client
Received from server: hi client from server
Received from server: hi client
```

hi server from client 1

```
Server started on port 12345
Connection from ('127.0.0.1', 35202)
Received from ('127.0.0.1', 35202): good evening server
Received from ('127.0.0.1', 35202): hi server
Received from ('127.0.0.1', 35202): hi server from client
Connection from ('127.0.0.1', 57074)
Received from ('127.0.0.1', 35202): hi server from client 1
Received from ('127.0.0.1', 57074): client2 here
```

hi client

CLIENT AND SERVER BOTH ON SAME PC



TCP Client

```
Connected to server.
Received from server: hi im talking from server
Received from server: im chaitra talking from server
Received from server: hiiii
Received from server: hi im chaitra server
Received from server: hi im chaitra server
Received from server: hiiiiii
Received from server: attendancr lagao idhar
Received from server: hi from chaitra server
```

Message to Server: kem cho

TCP Client

```
Connected to server.
Received from server: good evening client
Received from server: hi client
Received from server: hi client from server
Received from server: hi client
```

Message to Server: hi server from client 1

TCP Server

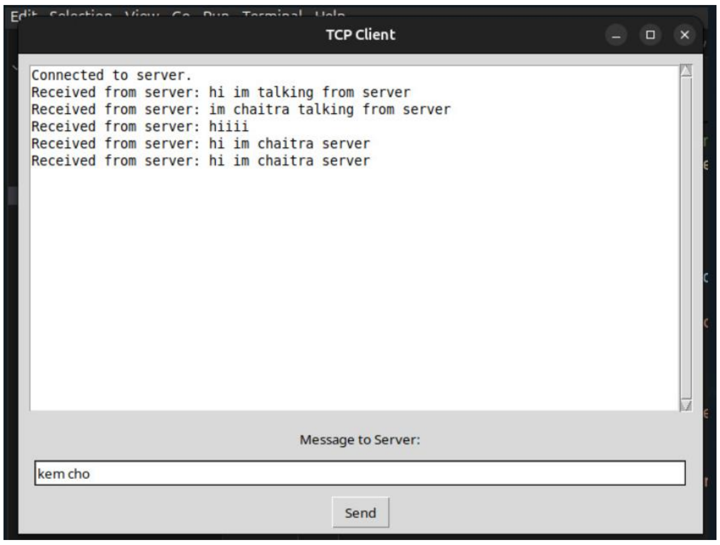
```
Server started on port 12345
Connection from ('127.0.0.1', 35202)
Received from ('127.0.0.1', 35202): good evening server
Received from ('127.0.0.1', 35202): hi server
Received from ('127.0.0.1', 35202): hi server from client
Connection from ('127.0.0.1', 57074)
Received from ('127.0.0.1', 35202): hi server from client 1
Received from ('127.0.0.1', 57074): client2 here
Client ('127.0.0.1', 57074) disconnected
```

Message to Client: hi client

CLIENT AND SERVER ON DIFFERENT PC

#NOTE:- FOR CLIENT AND SERVER TO BE CONNECTED THEY SHOULD HAVE SAME ID

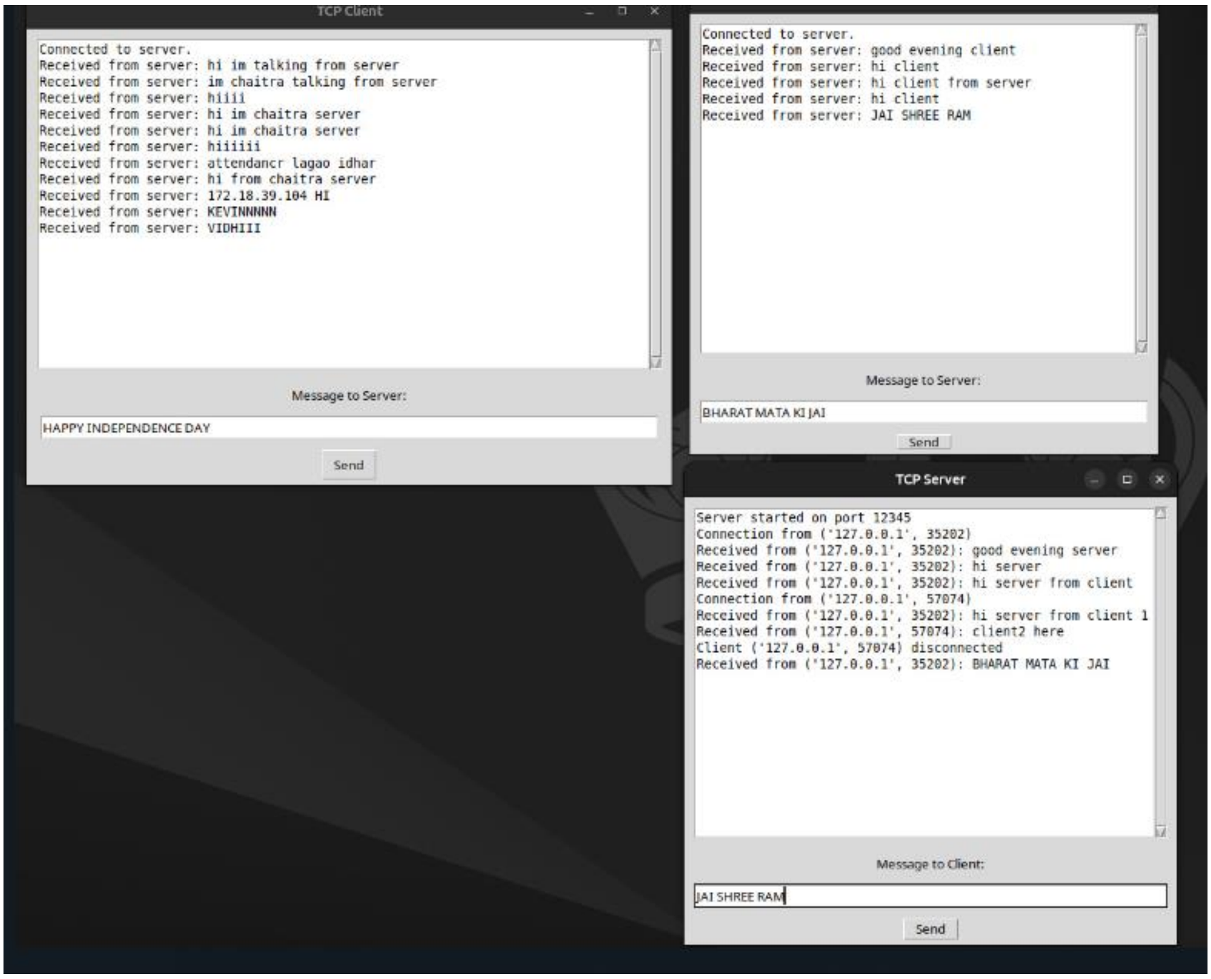
OTHER SCREENSHOTS:-



TCP Client

```
Connected to server.
Received from server: hi im talking from server
Received from server: im chaitra talking from server
Received from server: hiiii
Received from server: hi im chaitra server
Received from server: hi im chaitra server
```

Message to Server: kem cho



```
Connected to server.
Received from server: hi im talking from server
Received from server: im chaitra talking from server
Received from server: hiiii
Received from server: hi im chaitra server
Received from server: hi im chaitra server
Received from server: hiiiiii
Received from server: attendancr lagao idhar
Received from server: hi from chaitra server
Received from server: 172.18.39.104 HI
Received from server: KEVINNNNN
Received from server: VIDHIII
```

HAPPY INDEPENDENCE DAY

```
Connected to server.
Received from server: good evening client
Received from server: hi client
Received from server: hi client from server
Received from server: hi client
Received from server: JAI SHREE RAM
```

BHARAT MATA KI JAI

```
Server started on port 12345
Connection from ('127.0.0.1', 35202)
Received from ('127.0.0.1', 35202): good evening server
Received from ('127.0.0.1', 35202): hi server
Received from ('127.0.0.1', 35202): hi server from client
Connection from ('127.0.0.1', 57074)
Received from ('127.0.0.1', 35202): hi server from client 1
Received from ('127.0.0.1', 57074): client2 here
Client ('127.0.0.1', 57074) disconnected
Received from ('127.0.0.1', 35202): BHARAT MATA KI JAI
```

JAI SHREE RAM

CONCLUSION:-

Hence in this experiment we have learnt about tkinkter, GUI, socket programming, messaging application and how we can implement messaging application using socket programming. We have also seen the codes and outputs for the same.