A Project Report on

MESSANGO (Chat Application)

Submitted by

Disha Rane(19IT5002)

Surabhi Gawade(19IT5004)

Priya More(19IT5009)

Under the guidance of

Mrs. Anitha Senathi

Department of Information Technology

Ramrao Adik Institute of Technology Nerul,

Navi Mumbai

Certificate

This is to certify that the project entitled **MESSANGO** is being submitted to the Department of Information Technology, Ramrao Adik Institute of Technology, Navi Mumbai.

Project Guide External
Examiner(Mrs. Anitha Senathi) ( )

Acknowledgement

We owe our gratitude to many people who have supported us throughout this journey.

We would,firstly like to express our heartfelt gratitude towards our respected Principal Dr. Ramesh Vasappanavara and our Head of Department Dr. Ashish Jadhav for providing us immense facilities, guidance and never ending support.

The completion of any inter-disciplinary project depends upon cooperation and combined efforts of several sources of knowledge. We take this opportunity to express our profound gratitude and deep regards to our guide Mrs.AnithaSenathi for his exemplary guidance, monitoring and constant encouragement throughout the course of this project.

Lastly, we thank our parents, family, friends and well wishers who always looked for the chance to help us in whatever means came forth and for their constant encouragement without which the project would not be a distant reality.

Introduction

Chat refers to the process of communicating, interacting and/or exchanging messages over the Internet. It involves two or more individuals that communicate through a chat-enabled service or software. Chat may be delivered through text, audio or video communication via the Internet.

A chat application has basic two components: server and client .A server is a computer program or a device that provides functionality for other programs or device. Clients who want to chat with each other connect to the server.

The chat application we are making is more likely a peer to peer chat.so multiple users can connect to server and send that message. Every message is broadcast to every connect user.

Proposed System

The main objective of our project is to perform Multi-Client communication.In this project we will see how to make a server and client chat room system using Socket Programming with Python.

The sockets are the endpoints of any communication channel. These are used to connect the server and client. Sockets are Bi-Directional. In this area, we will setup sockets for each end and setup the chatroom system among different clients through the server. The server side has some ports to connect with client sockets. When a client tries to connect with the same port, then the connection will be established for the chat room.

There are basically two parts. The server side and the client side. When the server side script is running, it waits for any active connection request. When one connection is established, it can communicate with it.

In this case we are using localhost. If machines are connected via LAN, then we can use IP addresses to communicate. The server will display its IP, and ask for a name for the server. From the client side, we have to mention a name, and also the IP address of the server to connect.

System Components

1.Frontend: Tkinter

2.Backend : Python 3, PyCharm, Socket

1.1 TKINTER :

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

1. Import the Tkinter module.
2. Create the GUI application main window.
3. Add one or more of the above-mentioned widgets to the GUI application.
4. Enter the main event loop to take action against each event triggered by the user.

Tkinter Widgets:
Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets

2.1 Python:

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. The fast edit-test-debug cycle makes this simple approach very effective.

2.2 PyCharm:

PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django as well as Data Science with Anaconda.

PyCharm is cross-platform, with Windows, macOS and Linux versions. The Community Edition is released under the Apache License, and there is also Professional Edition with extra features – released under a proprietary license.

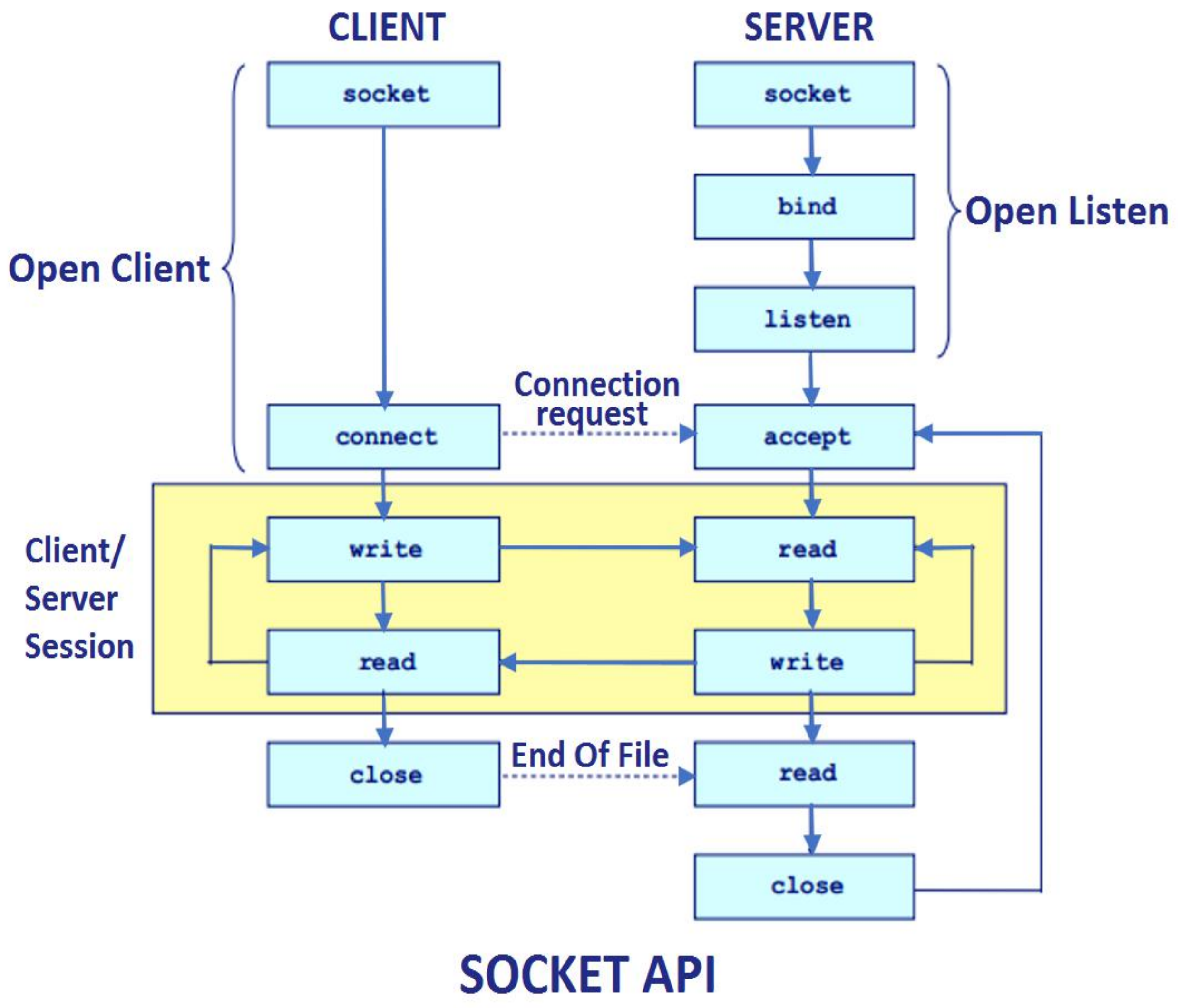
2.3 Sockets:

Sockets can be thought of as endpoints in a communication channel that is bi-directional, and establishes communication between a server and one or more clients. Here, we set up a socket on each end and allow a client to interact with other clients via the server. The socket on the server side associates itself with some hardware port on the server side. Any client that has a socket associated with the same port can communicate with the server socket.

To a programmer, a socket looks and behaves much like a low-level file descriptor. This is because commands such as send() and recv() work with sockets in the same way they do with files and pipes A socket is one endpoint of a two-way communication link between two programs running on the network.A socket is bound to a port number so that the tcp layer can identify the application that the data is destined to be sent to.Normally, a server runs on a specific computer and has a socket that is bound to a specific port number.The server just waits, listening to the socket for a client to make a connection request.On the client side : The client knows the host name of the machine of which the server is running and the port number in which the server is listening.To make a connection request .

The client also needs to identify itself to the server so it binds to local port number that it will use during connection. This is usually assigned by system.If everything goes well, the server accepts the connection. Upon acceptance, theserver gets a new socket bound to the same local port and also has its remote and endpoint set to the address and port of client. It needs a new socket for connection requests while trending to the needs of connected client.On the client side, if the connection is accepted a socket is successfully created and the client can use thesocket to communicate with the server.The client and server can now communicate by writing to or reading from their sockets.

Architecture:



Messango (flowchart)

WORKING OF SYSTEM

The MESSANGO basically consist of two modules:-

1.Server Side Script

The server side script will attempt to establish a socket and bind it to an IP address and port specified by the user (windows users might have to make an exception for the specified port number in their firewall settings, or can rather use a port that is already open). The script will then stay open and receive connection requests, and will append respective socket objects to a list to keep track of active connections.

Every time a user connects, a separate thread will be created for that user. In each thread, the server awaits a message, and sends that message to other users currently on the chat. If the server encounters an error while trying to receive a message from a particular thread, it will exit that thread.

2.Client Side Script

The client side script will simply attempt to access the server socket created at the specified IP address and port. Once it connects, it will continuously check as to whether the input comes from the server or from the client, and accordingly redirects output. If the input is from the server, it displays the message on the terminal. If the input is from the user, it sends the message that the users enters to the server for it to be broadcasted to other users.

Advantages of Messango:

1 . Instant communication

2. No waiting queues

3. Low barrier.

4. Can connect to multiple clients(5 clients) at same time.

5. Easy to use with GUI made with Tkinter

LIMITATION OF MESSANGO:

1. The clients can only chat by connecting to a local server.
2. Two clients on different networks cannot chat together.

Source Code

1. Server.py

```
import socket
import threading


class ChatServer:
    clients_list = []

    last_received_message = ""

    def __init__(self):
        self.server_socket = None
        self.create_listening_server()

    def create_listening_server(self):
        self.server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        local_ip = '192.168.1.206'
        local_port = 8000
        # this will allow you to immediately restart a TCP server
        self.server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR,
1)
        # this makes the server listen to requests coming from other computers
on the network
        self.server_socket.bind((local_ip, local_port))
        print("Listening for incoming messages..")
        self.server_socket.listen(5)
        self.receive_messages_in_a_new_thread()

    def receive_messages(self, so):
        while True:
            incoming_buffer = so.recv(256)
            if not incoming_buffer:
                break
            self.last_received_message = incoming_buffer.decode('utf-8')
            self.broadcast_to_all_clients(so)  # send to all clients
        so.close()

    def broadcast_to_all_clients(self, senders_socket):
        for client in self.clients_list:
            socket, (ip, port) = client
            if socket is not senders_socket:
                socket.sendall(self.last_received_message.encode('utf-8'))

    def receive_messages_in_a_new_thread(self):
        while True:
            client = so, (ip, port) = self.server_socket.accept()
            self.add_to_clients_list(client)
            print('Connected to ', ip, ':', str(port))
            t = threading.Thread(target=self.receive_messages, args=(so,))
            t.start()
    def add_to_clients_list(self, client):
        if client not in self.clients_list:
            self.clients_list.append(client)


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

2. GUI.py

```
from tkinter import Tk, Frame, Scrollbar, Label, END, Entry, Text, VERTICAL,
Button
import socket
import threading
from tkinter import messagebox


class GUI:
    client_socket = None
    last_received_message = None

    def __init__(self, master):
        self.root = master
        self.chat_transcript_area = None
        self.name_widget = None
        self.enter_text_widget = None
        self.join_button = None
        self.initialize_socket()
        self.initialize_gui()
        self.listen_for_incoming_messages_in_a_thread()

    def initialize_socket(self):
        self.client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        remote_ip = '192.168.1.206'
        remote_port = 8000
        self.client_socket.connect((remote_ip, remote_port))

    def initialize_gui(self):
        self.root.title("Messango")
        self.root.resizable(0, 0)
        self.display_chat_box()
        self.display_name_section()
        self.display_chat_entry_box()

    def listen_for_incoming_messages_in_a_thread(self):
        thread = threading.Thread(target=self.receive_message_from_server,
args=(self.client_socket,))
        thread.start()

    def receive_message_from_server(self, so):
        while True:
            buffer = so.recv(256)
            if not buffer:
                break
            message = buffer.decode('utf-8')
            if "joined" in message:
                user = message.split(":")[1]
                message = user + " has joined"
                self.chat_transcript_area.insert('end', message + '\n')
                self.chat_transcript_area.yview(END)
            else:
                self.chat_transcript_area.insert('end', message + '\n')
                self.chat_transcript_area.yview(END)

        so.close()

    def display_name_section(self):
        frame = Frame()
        Label(frame, text='Enter your name:', font=("Helvetica",
16)).pack(side='left', padx=10)
        self.name_widget = Entry(frame, width=50, borderwidth=2)
        self.name_widget.pack(side='left', anchor='e')
        self.join_button = Button(frame, text="Join", width=10,
command=self.on_join).pack(side='left')
        frame.pack(side='top', anchor='nw')

    def display_chat_box(self):
        frame = Frame()
        Label(frame, text='Chat Room:', font=("Serif", 12)).pack(side='top',
anchor='w')
        self.chat_transcript_area = Text(frame, width=60, height=10,
font=("Serif", 12))
        scrollbar = Scrollbar(frame, command=self.chat_transcript_area.yview,
orient=VERTICAL)
        self.chat_transcript_area.config(yscrollcommand=scrollbar.set)
        self.chat_transcript_area.bind('<KeyPress>', lambda e: 'break')
        self.chat_transcript_area.pack(side='left', padx=10)
        scrollbar.pack(side='right', fill='y')
        frame.pack(side='top')

    def display_chat_entry_box(self):
        frame = Frame()
        Label(frame, text='Enter message:', font=("Serif",
12)).pack(side='top', anchor='w')
        self.enter_text_widget = Text(frame, width=60, height=3, font=("Serif",
12))
        self.enter_text_widget.pack(side='left', pady=15)
        self.enter_text_widget.bind('<Return>', self.on_enter_key_pressed)
        frame.pack(side='top')

    def on_join(self):
        if len(self.name_widget.get()) == 0:
            messagebox.showerror(
                "Enter your name", "Enter your name to send a message")
            return
        self.name_widget.config(state='disabled')
        self.client_socket.send(("joined:" +
self.name_widget.get()).encode('utf-8'))

    def on_enter_key_pressed(self, event):
        if len(self.name_widget.get()) == 0:
            messagebox.showerror(
                "Enter your name", "Enter your name to send a message")
            return
        self.send_chat()
        self.clear_text()

    def clear_text(self):
        self.enter_text_widget.delete(1.0, 'end')

    def send_chat(self):
        senders_name = self.name_widget.get().strip() + ": "
        data = self.enter_text_widget.get(1.0, 'end').strip()
        message = (senders_name + data).encode('utf-8')
        self.chat_transcript_area.insert('end', message.decode('utf-8') + '\n')
        self.chat_transcript_area.yview(END)
        self.client_socket.send(message)
        self.enter_text_widget.delete(1.0, 'end')
        return 'break'

    def on_close_window(self):
        if messagebox.askokcancel("Quit", "Do you want to quit?"):
            self.root.destroy()
            self.client_socket.close()
            exit(0)


if __name__ == '__main__':
    root = Tk()
    gui = GUI(root)
    root.protocol("WM_DELETE_WINDOW", gui.on_close_window)
    root.mainloop()
```

Steps to run the project:

1. First we have to run the server.py on terminal by using command

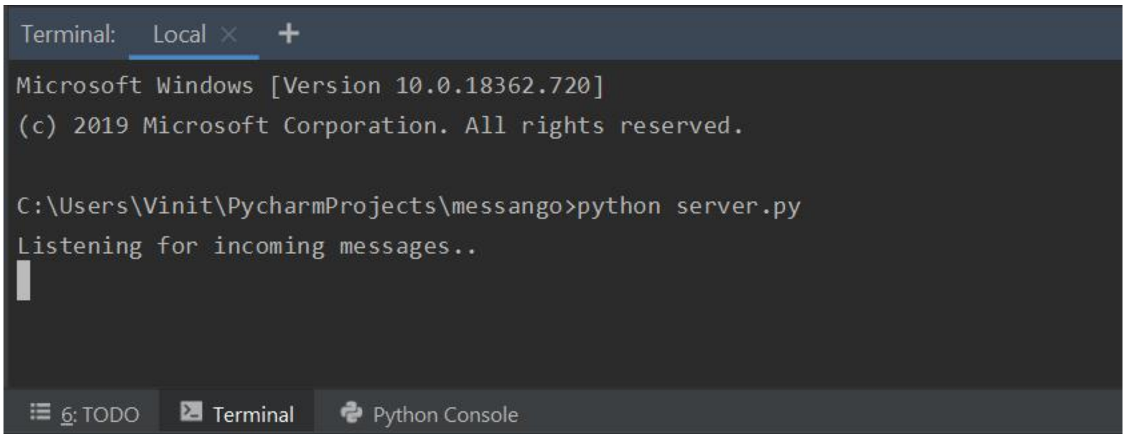
“python server.py” then we have to wait for receiving connections from client

2. Then we have to run GUI.py on other terminal by using command “python GUI.py” repeat this step for creating clients (upto 5 clients are allowed at same time)

3. Now, GUI i.e Messango chat window will appear on the screen of the clients and they will be able to interact with each other

OUTPUT:

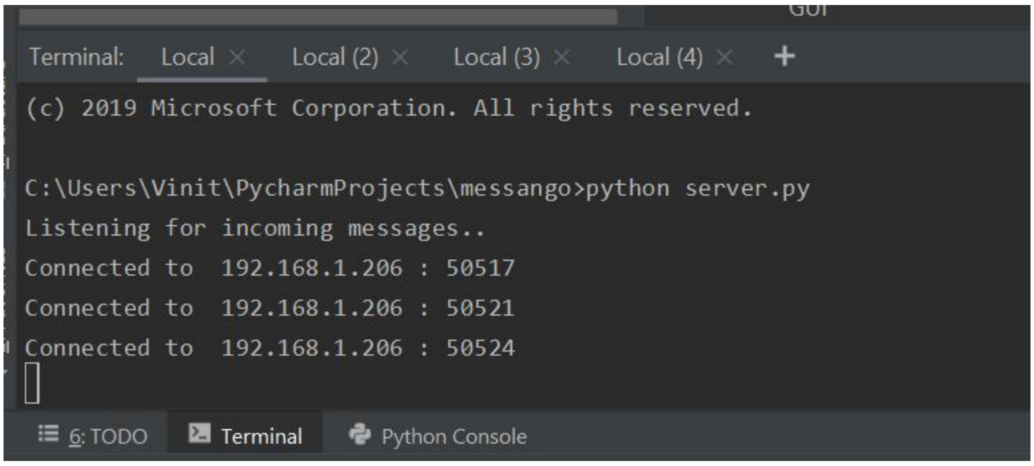
Server is waiting for receiving connections from client



```
Microsoft Windows [Version 10.0.18362.720]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Vinit\PycharmProjects\messango>python server.py
Listening for incoming messages..
```

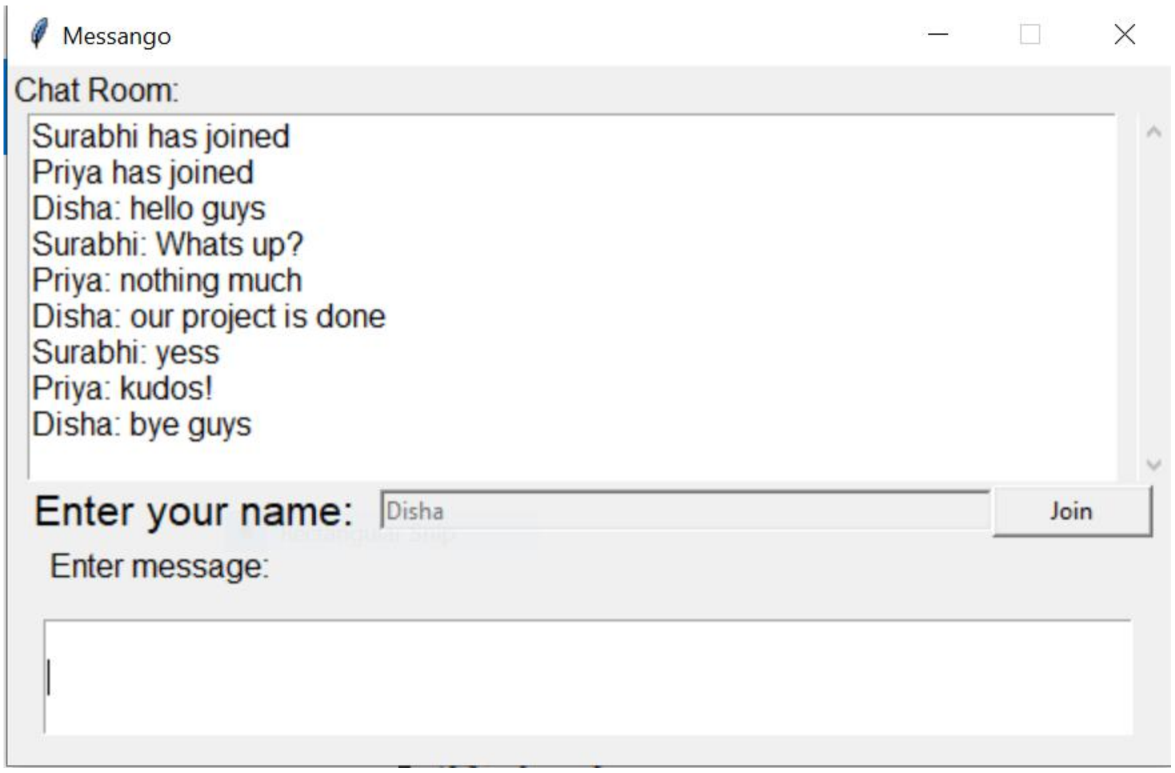
After receiving connection from clients server terminal show the IP and port number of its connected clients:



```
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Vinit\PycharmProjects\messango>python server.py
Listening for incoming messages..
Connected to  192.168.1.206 : 50517
Connected to  192.168.1.206 : 50521
Connected to  192.168.1.206 : 50524
```

Now, Clients can chat through each other using the Tkinter chat window created.

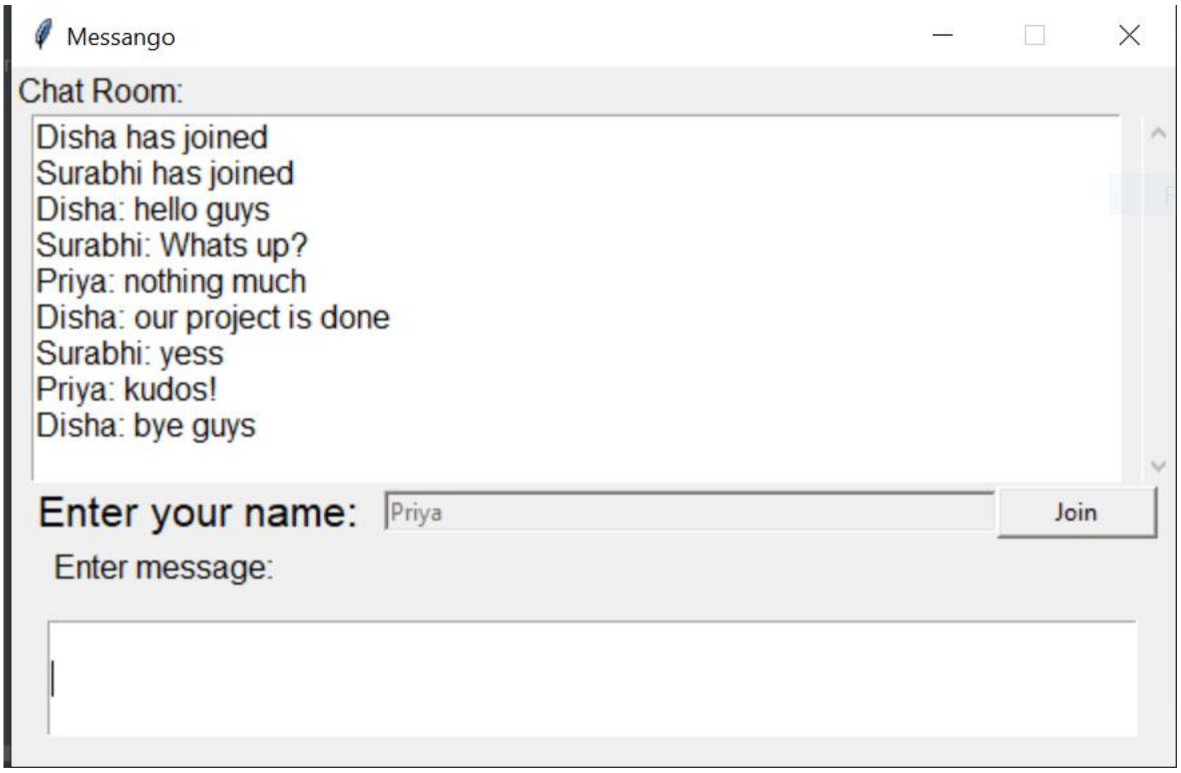


Chat Room:

```
Surabhi has joined
Priya has joined
Disha: hello guys
Surabhi: Whats up?
Priya: nothing much
Disha: our project is done
Surabhi: yess
Priya: kudos!
Disha: bye guys
```

Enter your name: Disha

Enter message:

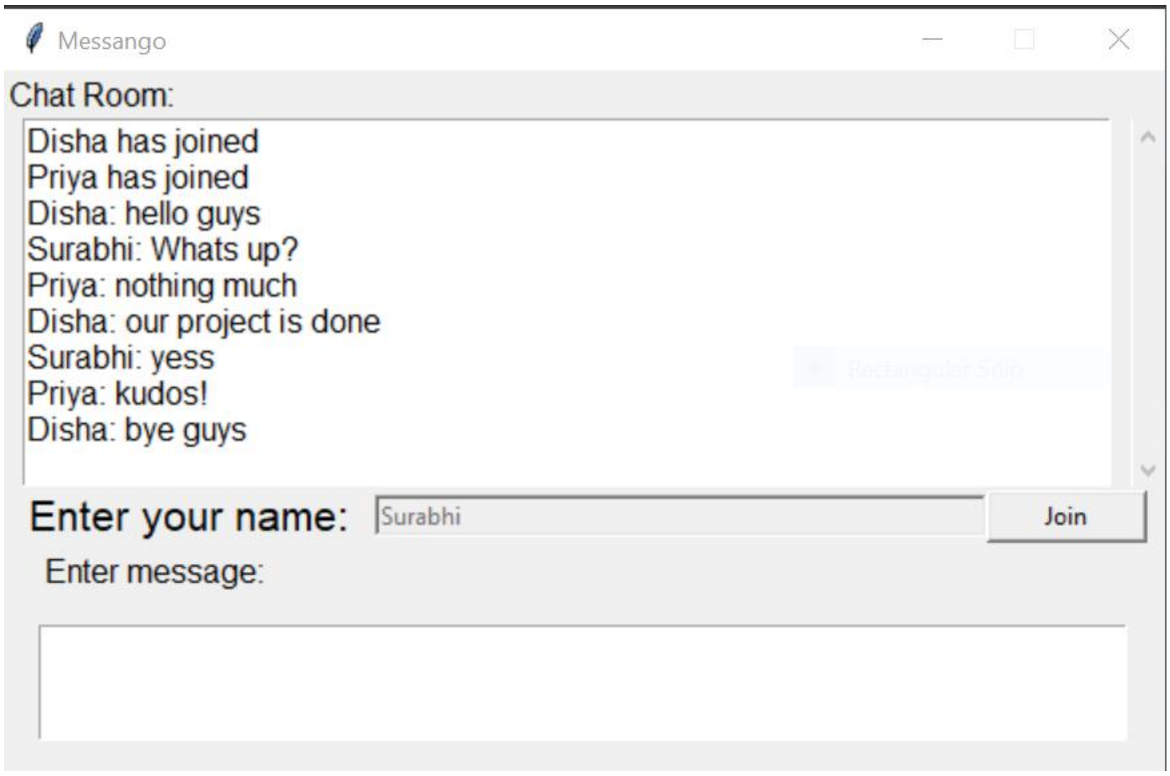


Chat Room:

```
Disha has joined
Surabhi has joined
Disha: hello guys
Surabhi: Whats up?
Priya: nothing much
Disha: our project is done
Surabhi: yess
Priya: kudos!
Disha: bye guys
```

Enter your name: Priya

Enter message:



```
Disha has joined
Priya has joined
Disha: hello guys
Surabhi: Whats up?
Priya: nothing much
Disha: our project is done
Surabhi: yess
Priya: kudos!
Disha: bye guys
```

Surabhi

Conclusion

In this project we have used the basics of networking in python. We also learned how to make a GUI for our application. This project teaches us how to create a basic chat application by creating a local network and using TCP sockets.

References

- https://www.geeksforgeeks.org/simple-chat-room-using-python/
- Python GUI programming CookBook
- https://docs.python.org/3/library/tk.html
- https://docs.python.org/3/howto/sockets.html