**CERTIFICATE**

It is hereby to certify that the original and genuine investigation work has been carried out to investigate about the subject matter and the related data collection has been completed by **Team** of course **BCA** in **DSEU Dwarka Campus , a ChatApp** Regarding with the topic of **Socket Programming & GUI** using **PYTHON**

**TEACHER’S SIGNATURE EXAMINER SIGNATURE**

# ACKNOWLEDGEMENT

### The successful completion of any task would be incomplete without mentioning the names of those persons who helped to make it possible. I take this opportunity to express my gratitude in few words and respect to all those who helped in the completion of this project.

* I would like to thank my CN teacher MS. BUSHRA JAMAL as his constant and guidance directed me to make this project successful.

### I would like to thank my parents as their support during the completion of project really meant a lot.

* Last but not the least I would like to thank my friends as I would have not been able to complete my project without their help and support.

# PROJECT OVERVIEW

* + INTRODUCTION
  + OBJECTIVE OF THE PROJECT
  + PYTHON MODULES USED
  + HARDWARE AND SOFTWARE REQUIREMENTS
  + CODING OF THE PROGRAM
  + OUTPUT
  + BIBLIOGRAPHY

# Introduction

|  |  |
| --- | --- |
| Chat Application project is written in Python. The project file contains a python script This is a simple GUI based project which is very easy to understand and use. Talking about the system, it contains all the required functions like SENDING and RECEIVING of MESSAGES ***(using Socket Programmimg and Tkinter ).***While starting the client window he has to choose NICKNAME .Then click on a Message Box , after filling it user clicks on send Button. | |
|  |  |

**Objectives of the Project**

Here are the prime objectives of this project

* Provide service facilities to user.
* Implement client server model which is small yet powerful.
* Use Tkinter, Socket, threading for working with user data
* Send Messages
* Receive Messages
* See all messages and client’s name

In addition to above primary objectives, we aim to keep the project’s complexity to the lowest level possible, so that the operator of the software can carry out all the operators with ease.

# PYTHON MODULES USED

* + Socket

Sockets are **commonly used for client and server interaction**. Typical system configuration places the server on one machine, with the clients on other machines. The clients connect to the server, exchange information, and then disconnect. A socket has a typical flow of events.

 The socket module **provides various objects, constants, functions and related exceptions for building full-fledged network applications including client and server programs**.

* + Tkinter

Tkinter is the de facto way in Python **to create Graphical User interfaces (GUIs)** and is included in all standard Python Distributions. In fact, it's the only framework built into the Python standard library.

**Python GUI Programming With Tkinter**

* Displaying Text and Images With Label Widgets.
* Displaying Clickable Buttons With Button Widgets.
* Getting User Input With Entry Widgets.
* Getting Multiline User Input With Text Widgets.
* Assigning Widgets to Frames With Frame Widgets.
* Adjusting Frame Appearance With Reliefs.
  + Threading

The thread Module. This method **starts a new thread and return its identifier**. The thread executes the function "function" (function is a reference to a function) with the argument list args (which must be a list or a tuple). The optional kwargs argument specifies a dictionary of keyword arguments

**Software and Hardware Requirements**

The software being developed in Python programming language scores high portability, i.e. runs on almost all different types of computers with any operating system.

#### Software Requirements

* + - Operating Systems: Windows, Linux, Macintosh
    - Latest Python Interpreter along with following modules: tkinter, socket, threading.

#### Hardware requirements

Any computer capable of running above mentioned software can run our program. We tested our program in following hardware configurations:

* + - Processor: Intel, AMD
    - Ram: 2GB, 4GB, 8GB
    - Hard Disk: 80 GB, 160 GB, 500 GB, 1 TB
    - Keyboard, Mouse: Wired or Wireless
    - Monitor: 14 inch or above VGA or HDMI



SOURCE CODE OF PROGRAM



Server Side Script

import socket

import threading

HOST = '127.0.0.1' # private ip adddress here (cmd ipconfig or myip.is)

PORT = 9090

server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) # internet & tcp sockets

server.bind((HOST, PORT))

server.listen()   # acts as host

clients = []      # used for communication

nicknames = []

# broadcast the message

def broadcast(message):

    for client in clients:

        client.send(message)

# handle (individual connections)

def handle(client):

    while True:

        try:

            message = client.recv(1024)

            print(f"{nicknames[clients.index(client)]}") # lock the message It TO BE PRINTED ON SERVER CONSOLE

            broadcast(message)                           # f means formatted string

        except:   # if disconnects remove from clients or nicknames list

            index = clients.index(client)

            clients.remove(client)

            client.close()

            nickname = nicknames[index]

            nicknames.remove(nickname)

            break #thread stops

# recieve the message

def receive():

    while True:

        client, address = server.accept() #add new clients

        print(f"Connected with {str(address)}!")

        client.send("NICK".encode('utf-8')) # using nickname

        nickname = client.recv(1024)

        nicknames.append(nickname)

        clients.append(client)

        print(f"Nickname of the client is {nickname}")

        broadcast(f"{nickname} connected to the server!\n".encode('utf-8'))

        client.send("Connected to the server".encode('utf-8'))

        #connection with each client

        thread = threading.Thread(target=handle, args=(client,))  # , means tuple

        thread.start()

print("!Server Running..")

receive()

Client Side Script

import socket

import threading

import tkinter

import tkinter.scrolledtext

from tkinter import simpledialog

HOST = '127.0.0.1'  #public address here

PORT = 9090

class Client:

    def \_\_init\_\_(self, host, port):

        self.sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

        self.sock.connect((host, port))

        msg = tkinter.Tk()

        msg.withdraw()

        self.nickname = simpledialog.askstring("Nickname", "Please choose a nickname", parent=msg)

        self.gui\_done = False # to custom GUI

        self.running = True

        gui\_thread = threading.Thread(target=self.gui\_loop)

        receive\_thread = threading.Thread(target=self.receive)

        gui\_thread.start()

        receive\_thread.start()

    def gui\_loop(self):

        self.win = tkinter.Tk()

        self.win.title(self.nickname)

        self.win.configure(bg="lightgray")

        self.chat\_label = tkinter.Label(self.win, text="Chat:", bg="lightgray")

        self.chat\_label.config(font=("Arial", 12))

        self.chat\_label.pack(padx=20, pady=5)

        self.text\_area = tkinter.scrolledtext.ScrolledText(self.win)

        self.text\_area.pack(padx=20,pady=5)

        self.text\_area.config(state='disabled') #to disable content change of this area

        self.msg\_label = tkinter.Label(self.win, text="Message:", bg="lightgray")

        self.msg\_label.config(font=("Arial", 12))

        self.msg\_label.pack(padx=20, pady=5)

        self.input\_area = tkinter.Text(self.win, height=3)

        self.input\_area.pack(padx=20, pady=5)

        self.send\_button = tkinter.Button(self.win, text="Send", command=self.write)

        self.send\_button.config(font=("Arial",12))

        self.send\_button.pack(padx=20, pady=5)

        self.gui\_done = True

        self.win.protocol("WM\_DELETE\_WINDOW", self.stop) #to terminate the connection

        self.win.mainloop()

    def write(self):

        message = f"{self.nickname}: {self.input\_area.get('1.0', 'end')}"

        self.sock.send(message.encode('utf-8'))

        self.input\_area.delete('1.0', 'end')

    def stop(self):

        self.running = False

        self.win.destroy()

        self.sock.close()

        exit(0)

    def receive(self):

        while self.running:

            try:

                message = self.sock.recv(1024).decode('utf-8')

                if message == 'NICK':

                    self.sock.send(self.nickname.encode('utf-8'))

                else:

                    if self.gui\_done:

                        self.text\_area.config(state='normal')

                        self.text\_area.insert('end', message)

                        self.text\_area.yview('end')

                        self.text\_area.config(state='disabled')

            except ConnectionAbortedError:

                break

            except:

                print("Error(socket already closed)")

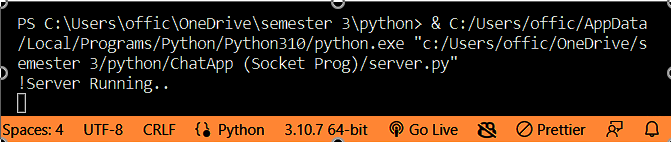
                self.sock.close()

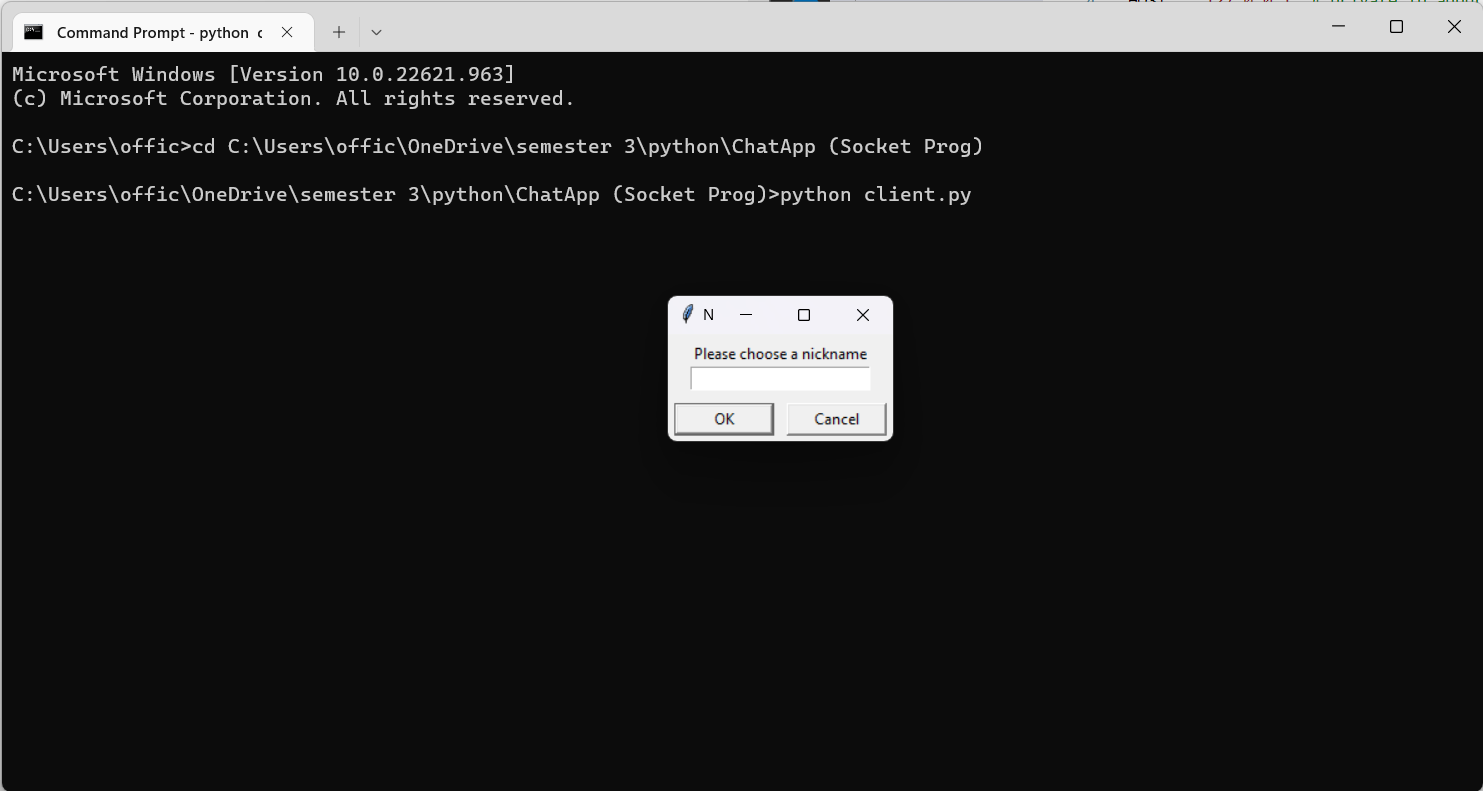
                break

client = Client(HOST, PORT)

OUTPUT:

* 1. Start Server

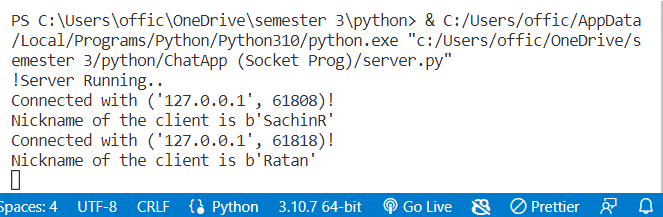


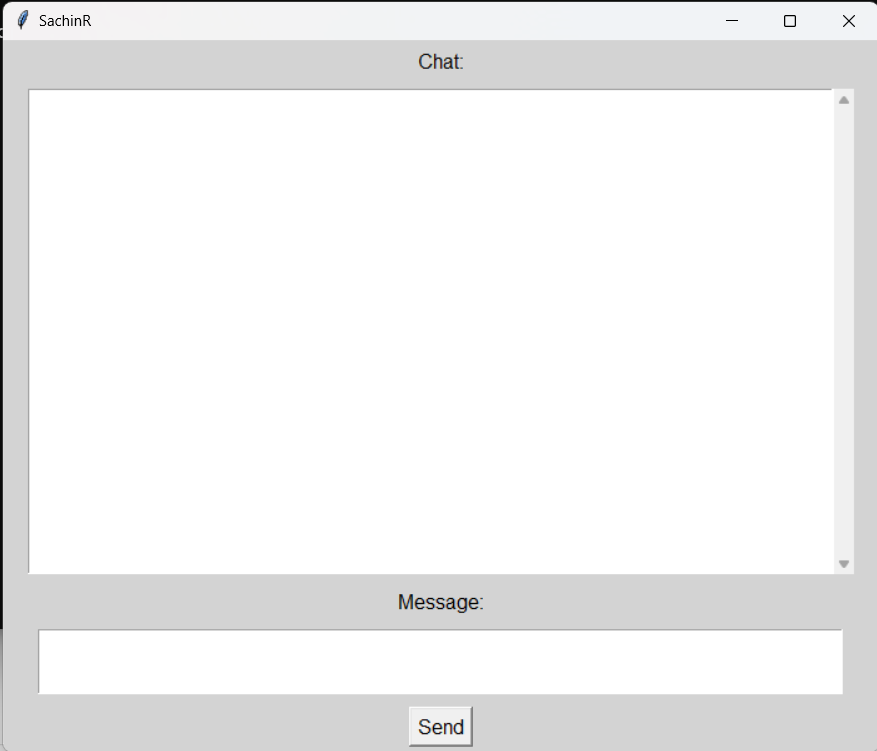
* 1.  Start Client Window

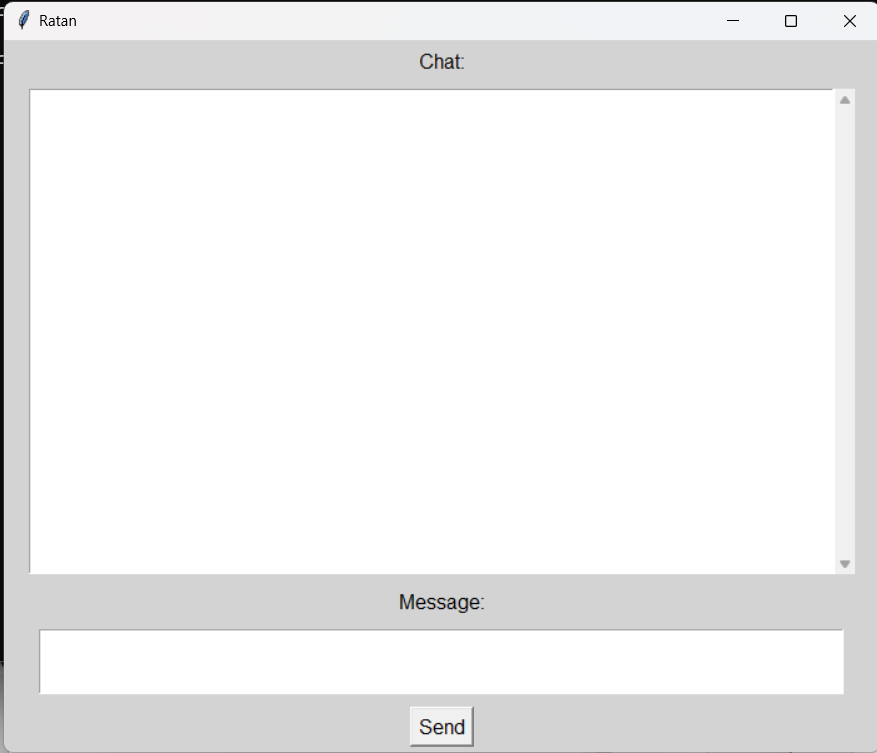
Enter Your Nickname

* 1. Connection is Established

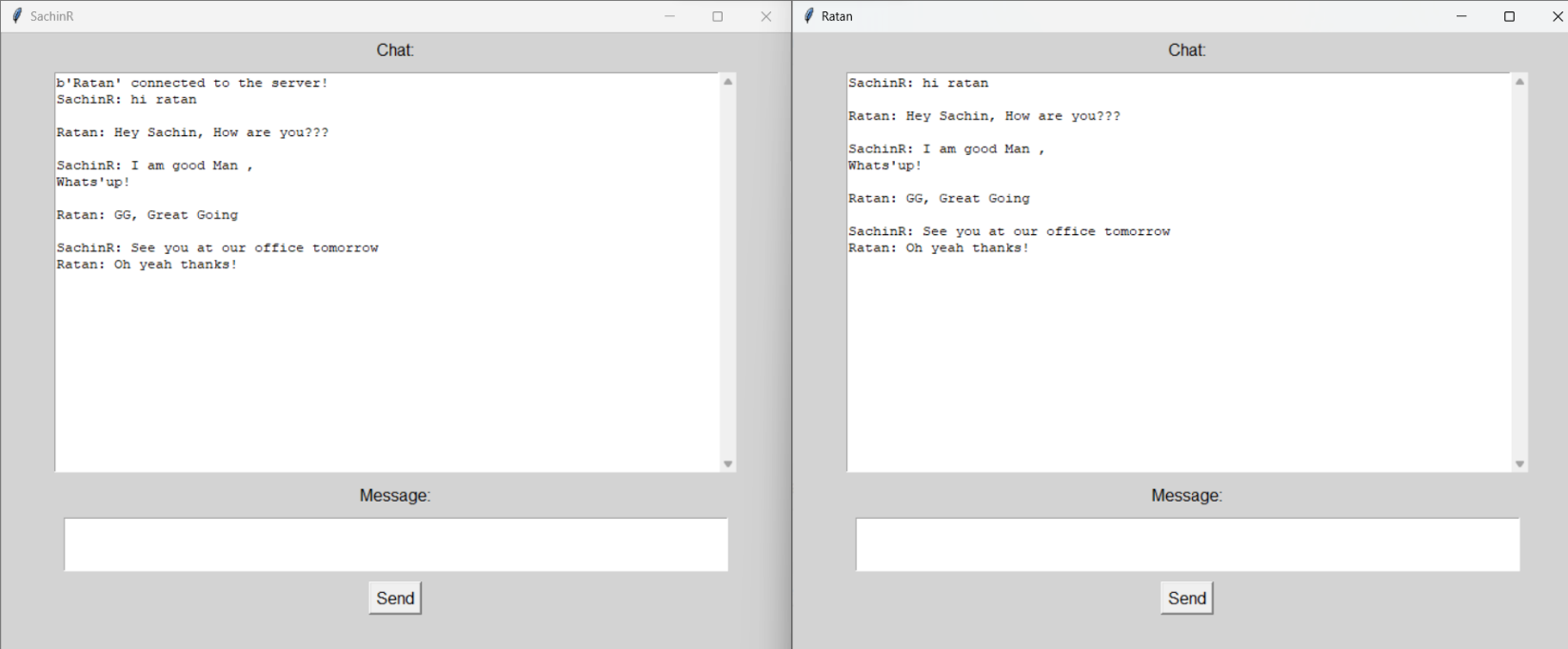








* 1. Chat Messages Transfer



**Bibliography**

**FOR THIS PROJECT, THE FOLLOWING BOOKS WERE CONSULTED:**

## Socket Prog. With Python ,

## Ajit Singh, 2019

AND FOLLOWING SITES:

* [www.google.com](http://www.google.com/)
* [www.wikipedia.com](http://www.wikipedia.com/)
* [WWW.YOUTUBE.COM](http://WWW.YOUTUBE.COM/)

##### 