**LET’S CHAT-A SIMPLE CHAT TOOL**

An Application Development-1 Report Submitted

In partial fulfillment of the requirement for the award of the degree of

## Bachelor of Technology

**In**

**Computer Science and Engineering - Artificial Intelligence and Machine Learning**

**By**

**USHA SREE VADDEBOYINA - 21N31A66J4**

**SATARSHETTY RISHIKESH - 21N31A66G0**

**PUTTAPAGA SUSHANTH KUMAR - 21N31A66F2**

Under the Guidance of

**Dr. Chandra Sekhar**

**Associate Professor**

**Department of Computational Intelligence**

**MRCET**



**DEPARTMENT OF COMPUTATIONAL INTELLIGENCE**

**MALLA REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY**

(Affiliated to JNTU, Hyderabad)

**ACCREDITED by AICTE-NBA**

**Maisammaguda, Dhulapally post, Secunderabad-500014.**

**(2021-2025)**

**DECLARATION**

## We hereby declare that the project entitled “LET’S CHAT-A SIMPLE CHAT TOOL” submitted to Malla Reddy College of Engineering and Technology, affiliated to Jawaharlal Nehru Technological University Hyderabad (JNTUH) for the award of the degree of Bachelor of Technology in Computer Science and Engineering - Artificial Intelligence and Machine Learning is a result of original research work done by us. It is further declared that the project report or any part thereof has not been previously submitted to any University or Institute for the award of degree or diploma.

**USHA SREE VADDEBOYINA (21N31A66J4)**

**SATARSHETTY RISHIKESH (21N31A66G0)**

**PUTTAPAGA SUSHANTH KUMAR (21N31A66F2)**



**CERTIFICATE**

This is to certify that this is the bonafide record of the project titled **“**LET’S CHAT-A SIMPLE CHAT TOOL**”** submitted by **Usha Sree Vaddeboyina (21N31A66J4), Sattarshetty Rishikesh (21N31A66G0) and Puttapaga Sushanth Kumar (21N31A66F2)** of B.Tech in the partial fulfillment of the requirements for the degree of **Bachelor of Technology** in **Computer Science and Engineering - Artificial Intelligence and Machine Learning**, Dept. of CI during the year 2023-2024. The results embodied in this project report have not been submitted to any other University or Institute for the award of any degree or diploma.

**D. Chandra Sekhar Reddy Dr. D. Sujatha**

(Assoc. Professor) (Professor)

**INTERNAL GUIDE HEAD OF THE DEPARTMENT**

**EXTERNAL EXAMINER**

# **ACKNOWLEDGEMENT**

We feel honored and privileged to place our warm salutation to our college Malla Reddy College of Engineering and technology (UGC-Autonomous), our Director ***Dr. VSK Reddy*** who gave us the opportunity to have experience in engineering and profound technical knowledge.

We are indebted to our Principal, ***Dr. S. Srinivasa Rao*** for providing us with facilities to do our project and his constant encouragement and moral support which motivated us to move forward with the project.

We would like to express our gratitude to our Head of the Department, ***Dr. D. Sujatha*** for encouraging us in every aspect of our system development and helping us realize our full potential.

We would like to thank our application development guide as well as our internal guide ***D. Chandra Sekhar*, *Assoc. Professor*,** for her structured guidance and never-ending encouragement. We are extremely grateful for valuable suggestions and unflinching co-operation throughout application development work.

We sincerely thank all the ***teaching and non-teaching staff*** of the Department of Computational Intelligence, for their timely suggestions, healthy criticism and motivation during the course of our work.

We would also like to thank our ***friends*** for providing help and moral support at the right timing. With great respect and obedience, we thank our ***parents*** who were the backbone behind our deeds.

Not but the least, we would like to thank the ***Almighty*** for constantly being with us in each and every crucial point of our life.

Finally, we express our immense gratitude with pleasure to all ***individuals*** who have either directly or indirectly contributed to our need at right time for the development and success of our project work.

**USHA SREE VADDEBOYNA (21N31A66J4)**

**SATTARSHETTY RISHIKESH (21N31A66G0)**

**PUTTAPAGA SUSHANTH KUMAR (21N31A66F2)**

**ABSTRACT**

LET’S CHAT is a tool that allows users to chat and send messages to everyone. A chat room is an interface where we can communicate with multiple users over the network. It can be interpersonal and group chat too. Chat tools enable users to start chatting with other users in real-time. This tool is highly scriptable allowing users to automate repetitive tasks. It is also highly portable as they rely on text-based input and output. This chat tool is designed to facilitate efficient and real-time text-based communication between users. It allows individuals to exchange messages, information and updates quickly and without the distractions of graphical interfaces. The main purpose of this chat tool is to provide users with a means of real-time text-based communication within a terminal or a command-line environment extended with an interface. The tool can be used to communicate with multiple users on a network. It also enables users to transmit text messages. It also enables instant messaging and can be used in a number of scenarios. These tools are often preferred over GUI applications by software developers. Since these chat tools run in terminal sessions, they can benefit from the security features of the terminal environment, as they often use a minimal amount of screen space, leaving more room for other terminal-based activities.

***Keywords:*** *Chat, scriptable, communication, graphical, interface, terminal*

**TABLE OF CONTENTS**

**S.No. Topic Page No.**

CHAPTER 1: INTRODUCTION ----------------------------------------------------------- 1-3

1.1: Purpose---------------------------------------------------------------------- 1

1.2: Background of project----------------------------------------------------- 1

1.3: Scope of project------------------------------------------------------------ 2

1.4: Project features------------------------------------------------------------- 2

CHAPTER 2: SYSTEM REQUIREMENTS----------------------------------------------- 4

2.1 H/W & S/W Requirements------------------------------------------------- 4

2.2 Functional Requirements---------------------------------------------------

CHAPTER 3: SYSTEM DESIGN ---------------------------------------------------------- 8

3.1: UML Diagrams

CHAPTER 4: IMPLEMENTATION -------------------------------------------------------10

4.1 Source Code

4.2 Output Screens

CHAPTER 5: CONCLUSION----------------------------------------------------------------12

BIBLIOGRAPHY------------------------------------------------------------------------------19

**Chapter-1**

1. **INTRODUCTION**

**1.1 Purpose:**

**1.1 Purpose:** Most often than not, we run heavy GUI-based applications for the simplest of tasks. One

such simple task is instant messaging or chatting. Chat tools enable users to start chatting

with other users in real-time. It also enables users to transmit text messages, images,

videos, and hyperlinks

Most often than not, we run heavy GUI-based applications for the simplest of tasks. One

such simple task is instant messaging or chatting. Chat tools enable users to start chatting

with other users in real-time. It also enables users to transmit text messages, images,

videos, and hyperlinks

Most often than not, we run heavy GUI-based applications for the simplest of tasks. One such simple task is instant messaging or chatting. Chat tools enable users to start chatting with other users in real-time. It also enables users to transmit text messages.

The main purpose of this document is to present the requirements of the project “Phishing detection using Machine Learning algorithms”. Most often than not, we run heavy GUI-based applications for the simplest of tasks. One

such simple task is instant messaging or chatting. Chat tools enable users to start chatting

with other users in real-time. It also enables users to transmit text messages, images,

videos, and hyperlinks

Most often than not, we run heavy GUI-based applications for the simplest of tasks. One

such simple task is instant messaging or chatting. Chat tools enable users to start chatting

with other users in real-time. It also enables users to transmit text messages, images,

videos, and hyperlin

**1.2 Background of project:**

A simple chat application is a software platform designed to facilitate real-time communication

between users. With an intuitive and user-friendly interface, this application allows individuals to exchange text messages in a seamless manner. Typically accessible across various platforms, including web browsers, desktop applications, and mobile devices, the simplicity of the design ensures easy navigation and use. Additional functionalities may include user profiles, notification systems, and message persistence for reviewing past conversations. The application's development emphasizes scalability to accommodate a growing user base and may involve integration with external services for enhanced features. Overall, a simple chat application aims to provide a straightforward yet effective means of communication, meeting the basic needs of users seeking instant messaging solutions.

**1.3 Scope of project:**

The scope of a simple chat application project encompasses the development of a user-friendly and real-time communication platform that facilitates instant messaging between users. The application is designed to meet the diverse needs of users seeking seamless and efficient interactions, whether for personal, professional, or social purposes. With cross-platform compatibility, users can engage in conversations across various devices, including web browsers, desktops, and mobile devices. The project involves the implementation of real-time features through technologies like WebSocket communication or long polling to ensure prompt message delivery. Security considerations are paramount, with encryption methods and robust user authentication mechanisms in place to safeguard user data and privacy. The application aims for scalability, accommodating a growing user base, and includes features such as notification systems, user profiles, and message persistence for a comprehensive user experience. Integration with external services, testing, and quality assurance are integral components, ensuring a reliable and bug-free chat application that enhances the efficiency and satisfaction of its users.

**1.4 Project Features­­­:**

* A simple chat tool typically includes fundamental features to facilitate communication between users. Here are some common features you might find in a basic chat application:
* User Registration and Authentication: Users should be able to create accounts and log in securely to access the chat application. Authentication ensures that only authorized users can participate in the chat.
* User Profiles: Users can create profiles with basic information and, optionally, profile pictures. This personalization helps users identify each other and adds a social aspect to the chat.

**Real-Time Messaging:**

The core feature is the ability to send and receive messages in real-time. Messages should appear promptly, providing an instant messaging experience.

**Group Chat:**

Users can create or join group chats to communicate with multiple people simultaneously. This is useful for team collaboration, social groups, or discussions on specific topics.

**Security Measures:**

Encryption of messages and data to ensure the privacy and security of user communications. Security features might also include secure login mechanisms and protection against common vulnerabilities.

**2. SYSTEM REQUIREMENTS**

# 

**2.1 Hardware Requirements:**

The hardware interfaces of this product consist of architecture, processing power, memory, secondary storage, display adapter, peripherals like CD-ROM drivers, keyboards, pointing devices, network devices, etc.

* Processor: i3 and above
* Ram: 8GB and above

**2.2 Software requirements:**

* Operating System: Windows
* IDLE (Python 3.11 64-Bit)

**2.3 Existing System:**

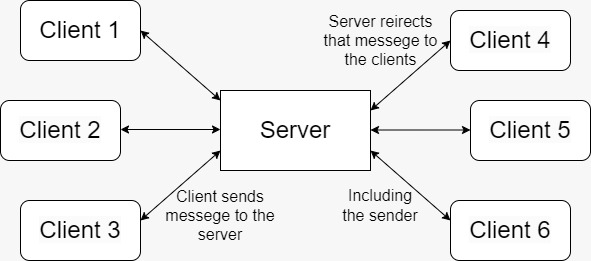
* **IRSSI**: IRSSI is a popular IRC (Internet Relay Chat) client with a text-based interface. It's widely used by developers and open-source communities for real-time chat and discussions.
* **Wee Chat**: Wee Chat is another IRC client that offers a highly customizable text-based interface. It supports scripting and has a plugin system for extending functionality.

**2.4 Proposed System:**

* This Chat tool can be used to communicate with multiple users on a network.
* To build this Chat tool, we use socket programming and multi-threading using python.
* We do not have to run heavy GUI-based applications for the simplest tasks such as chatting. Chat tools enable users to start chatting with other users in real-time.
* It also requires building a chat application that allows multiple clients to connect to server and communicate with each other.

**3. SYSTEM DESIGN**

**3.1 System Architecture**



**Figure 1 System Architecture**

**3.3 UML Diagrams**

#### **3.3.1 Use case diagram:**

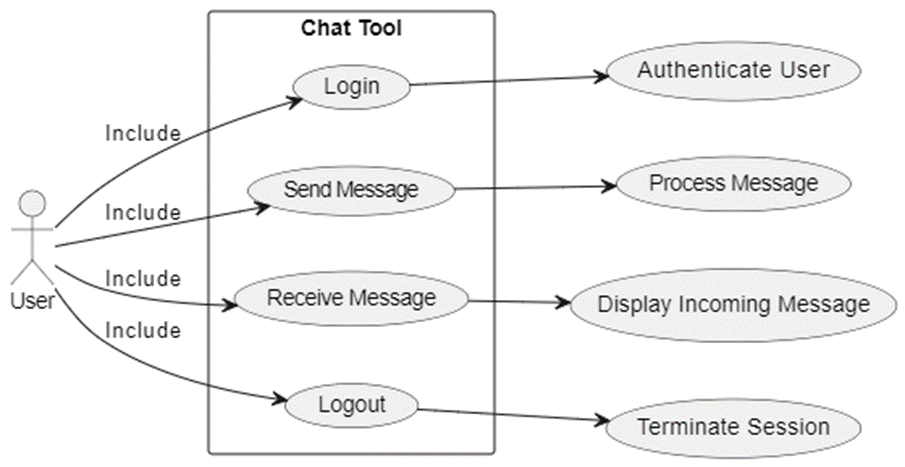
A use case is a description of how a user will interact with a system to achieve a specific goal. In the context of a chat application, a use case might be to allow users to send and receive messages, or to create and join groups.

USECase is significant in chat application using python because it helps to ensure that the application meets the needs of its users. By carefully considering the different ways that users will want to use the application, developers can design a system that is both easy to use and effective.

#### 

Use Case during requirement elicitation and analysis to represent the functionality of the system. Use case describes a function by the system that yields a visible result for an actor. The identification of actors and use cases result in the definitions of the boundary of the system i.e., differentiating the tasks accomplished by the system and the tasks accomplished by its environment.

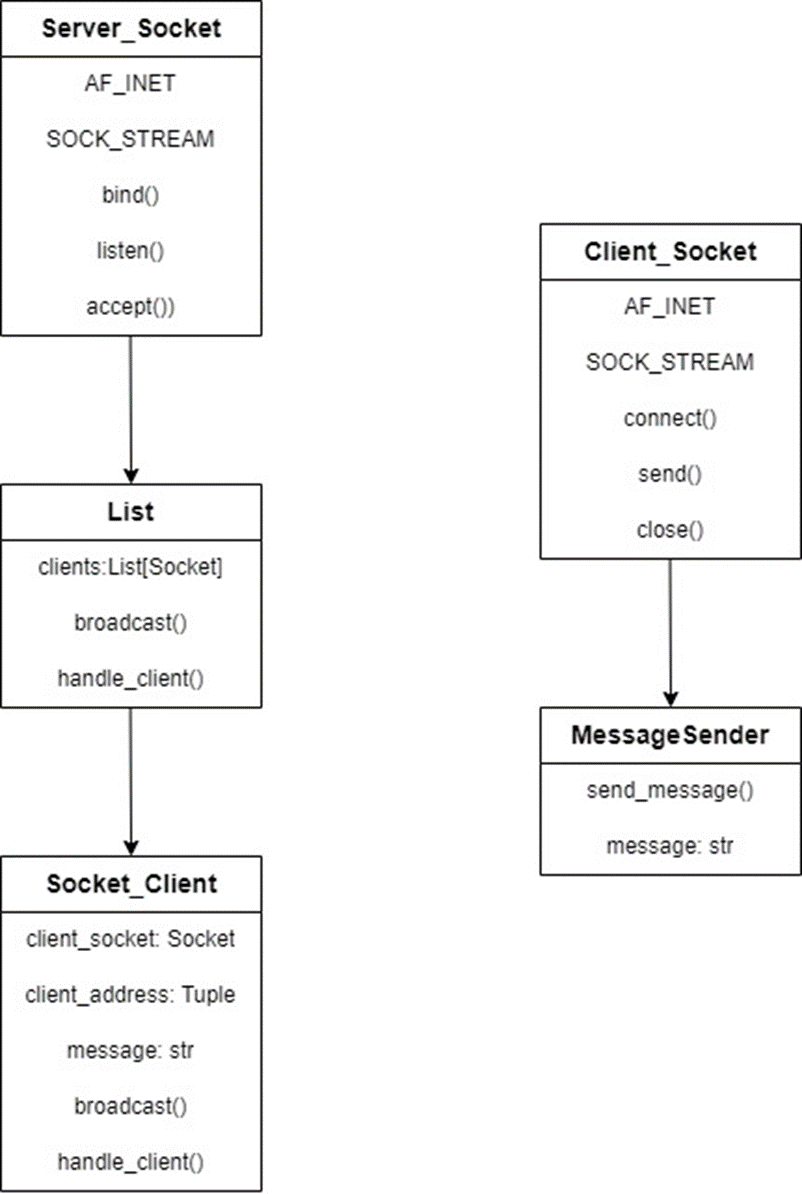
In the figure given below, it represents the interactions between the user and system. It represents the various functionalities that the system provides. Those include login of the user, sending and receiving the message and termination of the session.



#### **3.3.2 Class Diagram**

#### 

Class diagrams model class structure and contents using design elements such as classes, packages and objects. Class diagram describe the different perspective when designing a system-conceptual, specification and implementation. Classes are composed of three things: name, attributes, and operations. Class diagram also display relationships such as containment, inheritance, association etc. The association relationship is most common relationship in a class diagram. The association shows the relationship between instances of classes.

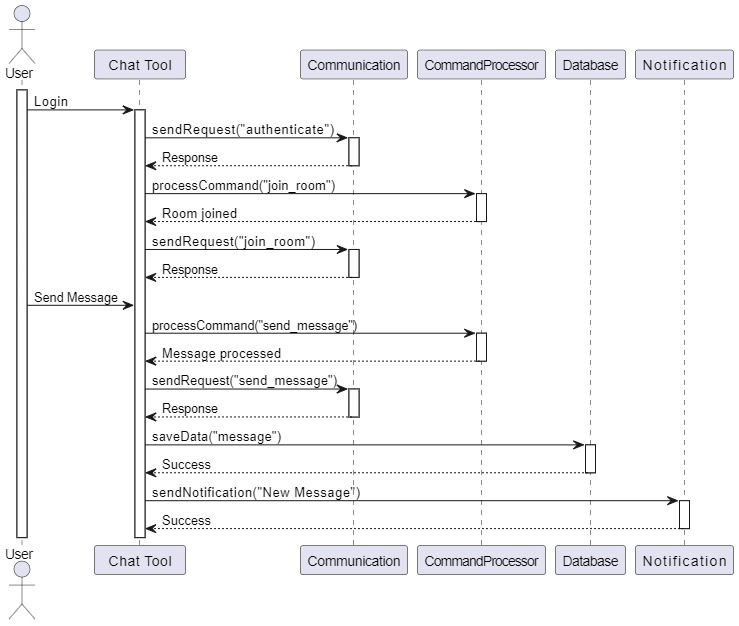


#### **3.3.3 Sequence Diagram**

#### 

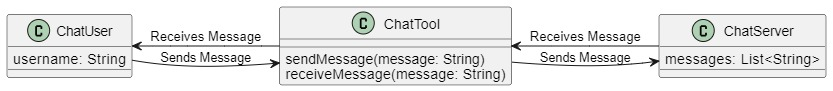
Sequence diagram displays the time sequence of the objects participating in the interaction. This consists of the vertical dimension (time) and horizontal dimension (different objects).

Objects: An object can be thought of as an entity that exists at a specified time and has a definite value, as well as a holder of identity. A sequence diagram depicts item interactions in chronological order. It illustrates the scenario's objects and classes, as well as the sequence of messages sent between them in order to carry out the scenario's functionality.



#### **3.3.4 Activity Diagram**

The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions.



**4. IMPLEMENTATION**

**4.1: Code**

SERVER CODE: chat\_server.py

import socket

import threading

HOST = '127.0.0.1'

PORT = 1234

LISTENER\_LIMIT = 5

active\_clients = []

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")

def send\_message\_to\_client(client, message):

client.sendall(message.encode())

def send\_messages\_to\_all(message):

for user in active\_clients:

send\_message\_to\_client(user[1], message)

def client\_handler(client):

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()

def main():

server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

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}")

server.listen(LISTENER\_LIMIT)

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()

CLIENT CODE: chat\_client.py

import socket

import threading

import tkinter as tk

from tkinter import scrolledtext

from tkinter import messagebox

HOST = '127.0.0.1'

PORT = 1234

DARK\_GREY = '#121212'

MEDIUM\_GREY = '#1F1B24'

OCEAN\_BLUE = '#464EB8'

WHITE = "white"

FONT = ("Helvetica", 17)

BUTTON\_FONT = ("Helvetica", 15)

SMALL\_FONT = ("Helvetica", 13)

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:

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")

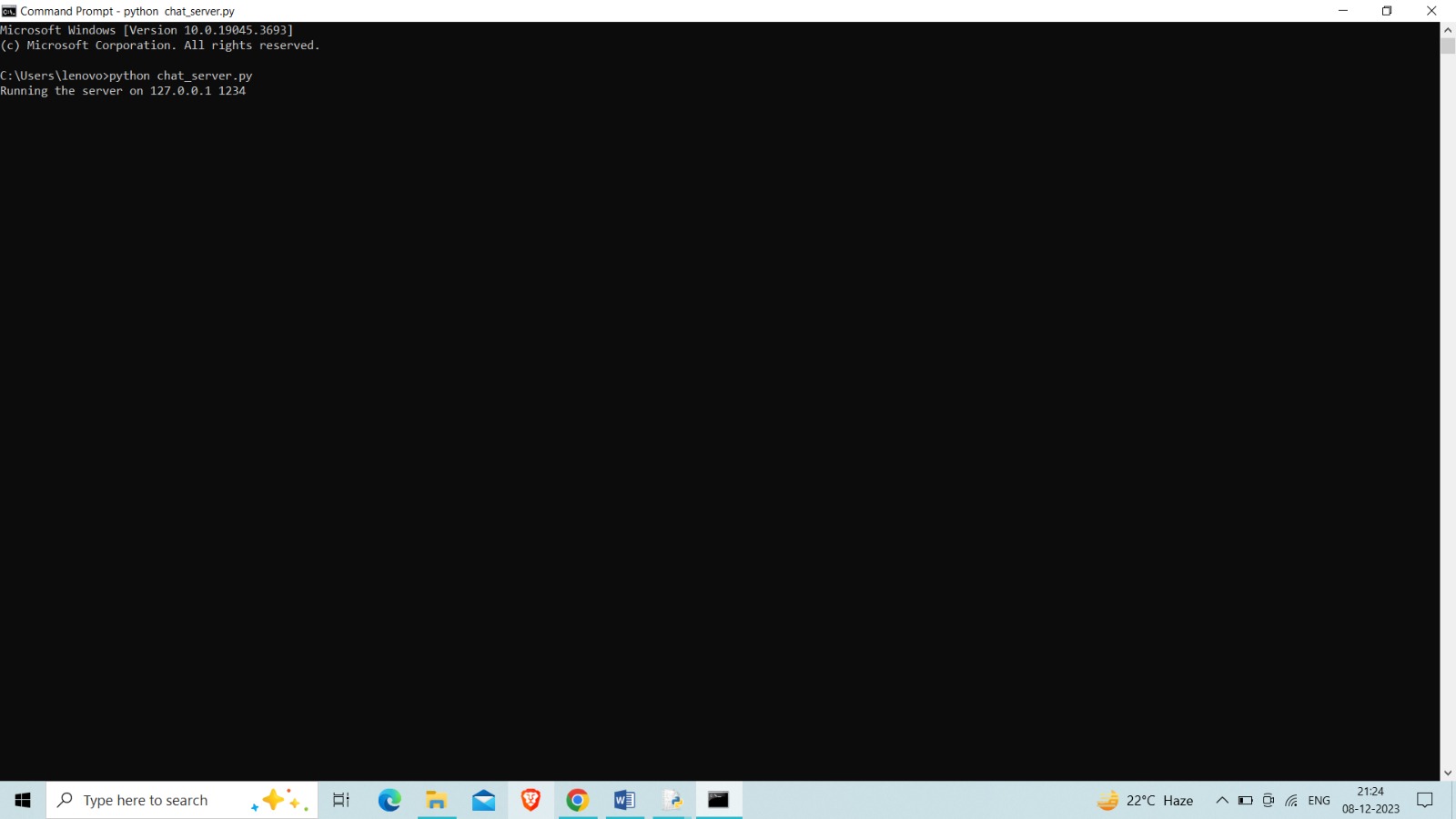
def main():

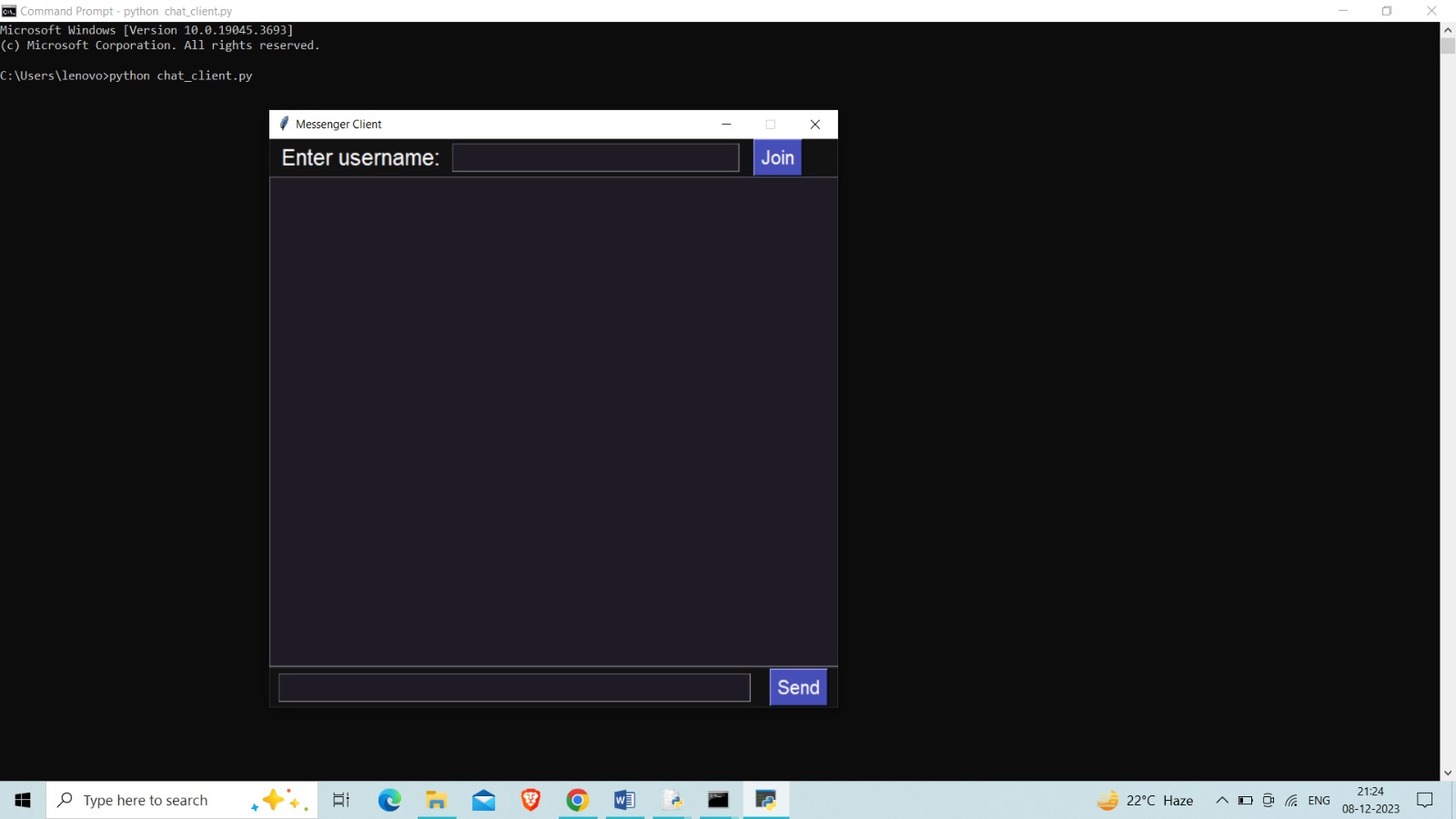
root.mainloop()

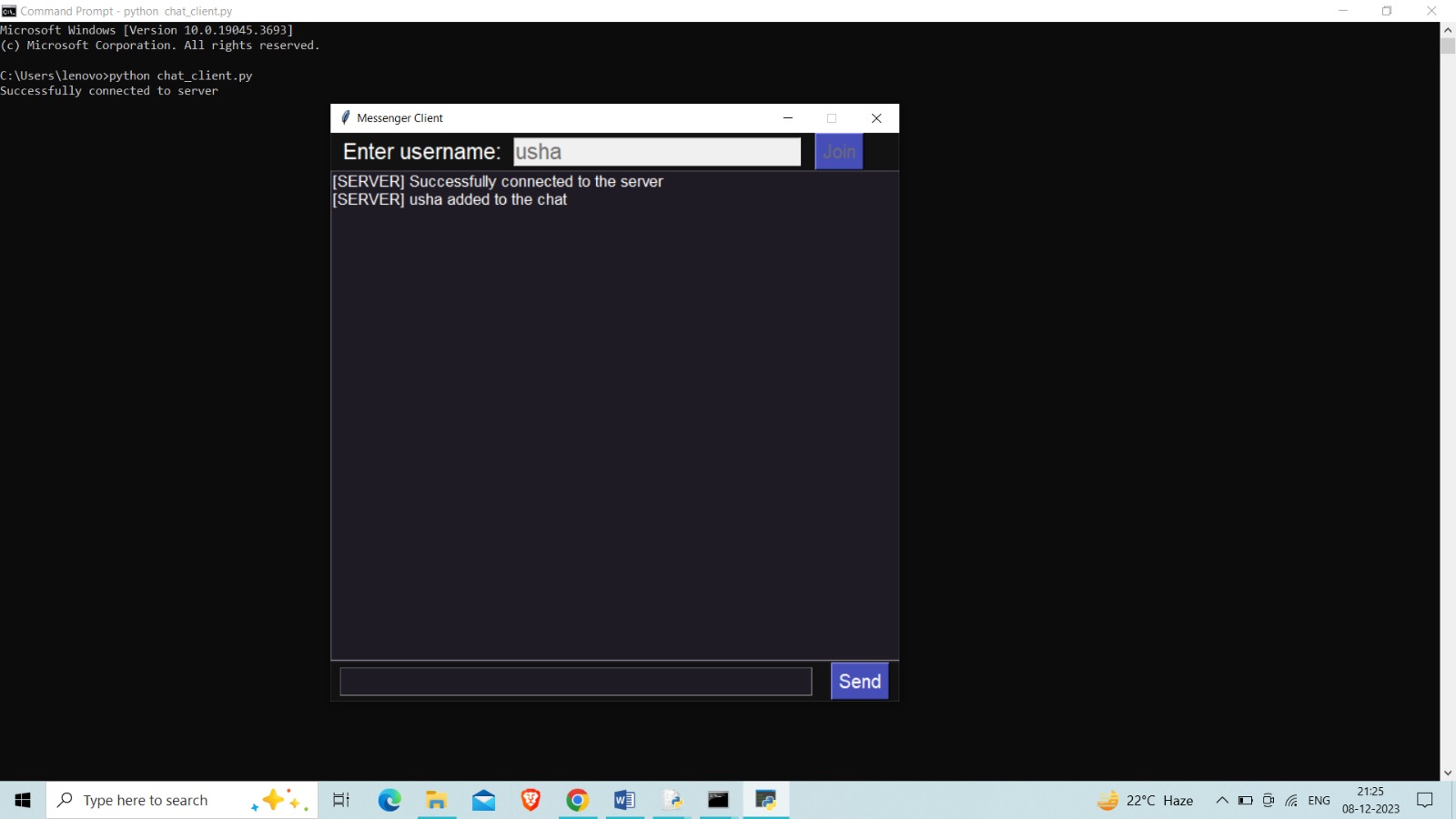
if \_\_name\_\_ == '\_\_main\_\_':

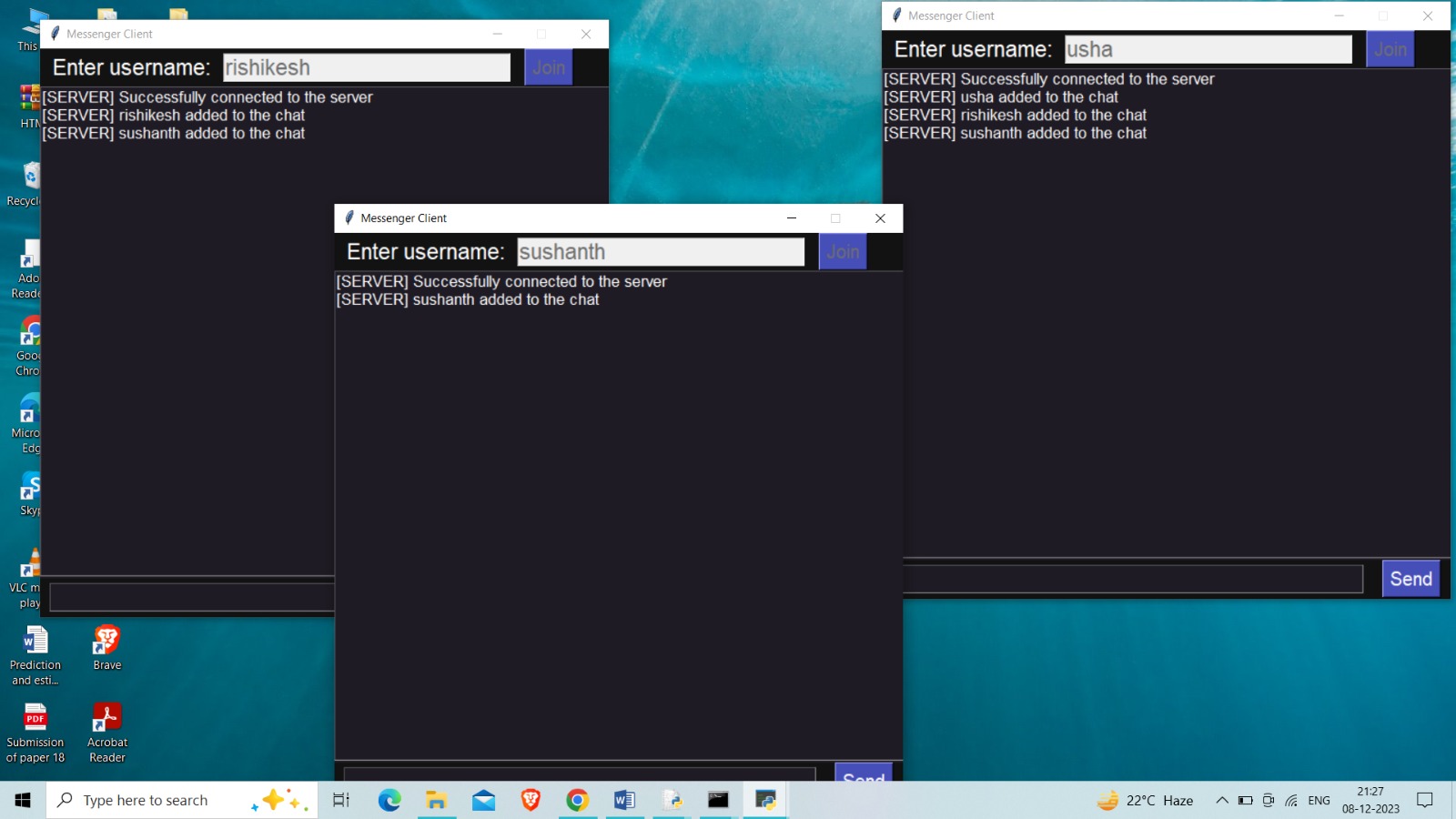
main()

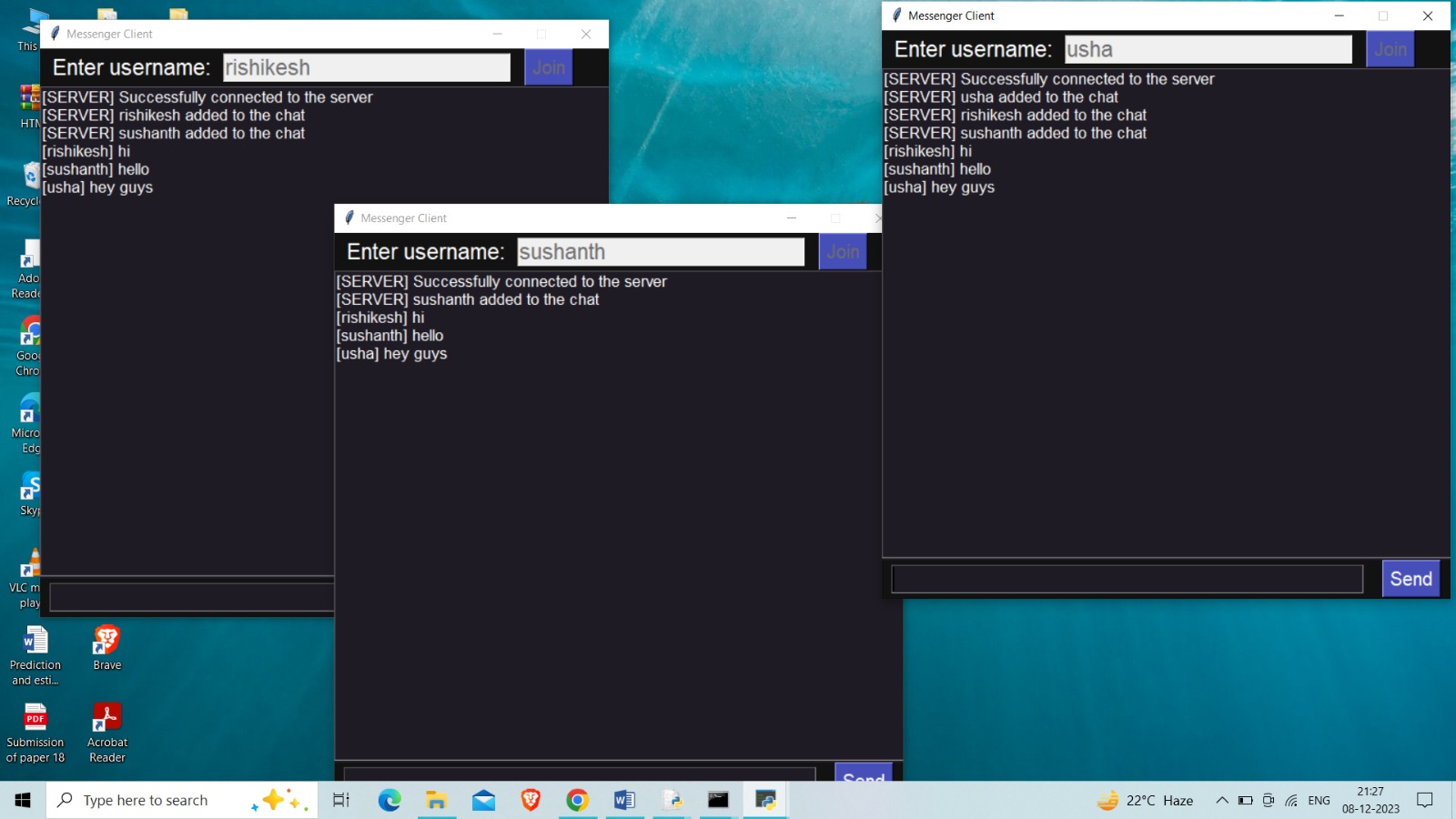
**4.2: Output Screens:**











**5. CONCLUSION:**

In conclusion, a simple chat tool based on the terminal or command prompt offers a streamlined and efficient means of communication. Stripped of graphical complexities, it caters to users who value a straightforward interface and appreciate the ease of navigating through commands. This type of application finds its niche in scenarios where resource efficiency, automation, or a preference for text-based interaction prevails. Its suitability for server administration, remote collaboration, and scripting environments underscores its versatility. With a focus on essential features such as real-time messaging, security, and cross-platform compatibility, this chat tool provides a reliable solution for users seeking a functional and uncomplicated means of instant communication.

## 6. BIBILOGRAPHY