GUI BASED CHAT SERVER

A PROJECT REPORT

Submitted by

Ronish Raja [RA2111003011037]
Somesh Kumar [RA2111003011072]
Ahnaf Ahmad [RA2111003011054]
Malik Aurangzaib [RA2111003011070]
Sheshank Gandivalasa [RA2111003011095]

Under the Guidance of
Dr. Jeyashekar A
Associate Professor, Computing Technologies

in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING

INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

DEPARTMENT OF COMPUTING TECHNOLOGIES
COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR- 603 203

NOV 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that 18CSC302J project report titled "**GUI BASED CHAT SERVER**" is the bonafide work of **"Ronish Raja [RA2111003011037], Somesh Kumar [RA2111003011072], Ahnaf Ahmad [RA2111003011054], Malik Aurangzaib [RA2111003011070] and Sheshank Gandivalasa [RA211100301095]"** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein doesnot form any other project report or dissertation on the basis of which a degree oraward was conferred on an earlier occasion on this or any other candidate.

DR. A. Jeyasekar
FACULTY IN CHARGE
Associate Professor
Department of Computing Technologies

Dr . Pushpalatha M
HEAD Of DEPARTMENT
Department of Computing Technologies

SIGNATURE OF INTERNAL EXAMINER

SIGNATURE OF EXTERNAL EXAMINER

Department of Computing Technologies

SRM Institute of Science and Technology

Own Work Declaration Form

Degree/ Course: B.Tech in Computer Science and Engineering

Student Names: Ronish Raja, Somesh Kumar, Ahnaf Ahmad, Malik Aurangazib, Sheshank Gandivasala.

Registration Number: RA2111003011037, RA2111003011072,RA2111003011054, RA2111003011070, RA2111003011095

Title of Work **:** GUI BASED CHAT SERVER

We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines.

We confirm that all the work contained in this assessment is our own except where indicated, and that we have met the following conditions:

- Clearly references / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc.)
- Given the sources of all pictures, data etc. that are not my own

- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:
I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.
If you are working in a group, please write your registration numbers and sign with the date for every student in your group.

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable Vice **Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our Registrar **Dr. S. Ponnusamy**, for his encouragement.

We express our profound gratitude to our Dean (College of Engineering and Technology) **Dr. T. V.Gopal**, for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman**, for imparting confidence to complete my course project.

We wish to express my sincere thanks to Course Audit Professor **Dr.Annapurani Panaiyappan, Professor and Head, Department of Networking and Communications and Course Coordinators** for their constant encouragement and support.

We are highly thankful to our my Course project Faculty Saminathan S Computer & Science & Engineering , Department of Computing Technologies , for his/her assistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our DR.Pushpalatha, Head of the Department and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

TABLE OF CONTENT

ABBREVIATION

1. SMS	Short Message Service
2. IM	Instant Messaging
3. CD	Compact Disc
4. JWT	JavaScript Object Notation Web Token
5.OAuth	Open Authorization
6. UX	User Experience

ABSTRACT

In today's digital age, effective communication is vital, and chat servers play a pivotal role in facilitating real-time interactions. This abstract introduces a User-Friendly Graphical User Interface (GUI) Chat Server, a solution designed to simplify and enhance the chat experience. This chat server is intended for both personal and professional use, offering a wide array of features to make communication convenient and secure. The GUI Chat Server provides a user-friendly interface that simplifies the registration and login processes, allowing users to quickly get started. Once logged in, users can effortlessly manage their contacts, create or join chat rooms, and engage in both private and group conversations. The platform supports text, multimedia, and file sharing, making it versatile for a variety of communication needs. Security is a top priority. The server employs message encryption to ensure the privacy and integrity of messages. It utilizes a robust client-server architecture for stability and efficient message routing. Administrators have access to comprehensive controls for managing user accounts and monitoring chat rooms, ensuring the system's integrity. This project combines elements of user interface design, network architecture, and security to create a chat server that is not only easy to use but also dependable. Whether it's for business collaborations, educational purposes, or social networking, the User-Friendly GUI Chat Server aims to be a valuable tool that promotes effective communication in a digital world.

CHAPTER - 1
INTRODUCTION

Our aim is to make a **GUI BASED CHAT SERVER** application where users can make their own connection functionality whereusers share the text with their friends to have text-based communication.

Teleconferencing or chatting, is a method of using technology to bring people and ideas "together" despite of the geographical barriers. While the technology for real-time chat and teleconferencing has existed for a number of years, its widespread acceptance and integration into our daily lives have been relatively recent developments. Our project, which serves as an example of a chat server, focuses on creating a real-time chat application with room functionality. In this application, users have the capability to create their own chat rooms and share the room names with friends, facilitating text-based communication.

The significance of teleconferencing and chat technology lies in its ability to harness technological innovation, effectively bridging people and ideas across vast geographical divides. This transformative means of communication has only recently gained the recognition it deserves, as it continues to play an increasingly central role in how we connect, collaborate, and interact with one another in today's digital age. At the core of our project lies a compelling illustration of this advancement – a robust chat server. This platform empowers users to engage in real-time text-based conversations through a chat application featuring room functionality. This tool facilitates seamless and instantaneous communication, fostering collaboration, and transcending geographic limitations. It embodies the ever-growing need for global connectivity and the power of technology to bridge physical distances in an era where real-time interaction and idea exchange have become integral to our personal and professional lives.

Relation to External Environment

This tool helps in too major aspects-

1. It lists the names of online users in the Chat server.
2. It is used for communication between multiple systems enlisted in the list.
3. It provides a platform for users to connect with like-minded individuals or professionals with similar interests, fostering networking and relationship-building.
4. It can be used for facilitating participant interaction and engagement.
5. It is a valuable tool for coordinating tasks, sharing updates, and ensuring everyone is on the same page, enhancing project management.
6. It can be used for discussions and group projects, enhancing remote learning experiences.
7. Users can receive real-time notifications and updates within the chat room, ensuring they stay informed about critical developments and events.
8. It wipes any existing data in the application when user exits, thus enhancing user privacy than many social networking sites.

CHAPTER - 2
LITERATURE SURVEY

In the contemporary era, the use of the internet has witnessed an unprecedented surge, solidifying its position as one of the most convenient and cost-effective means of communication. The digital landscape has been inundated with a plethora of messaging and chatting applications, further enriching our options for staying connected. While the stalwart of online communication remains electronic mail, commonly referred to as email, it continues to be the bedrock for both professional and personal correspondence. However, the digital age has ushered in a myriad of applications that cater to diverse communication needs. The ubiquitous presence of social media platforms such as Facebook, Snapchat, Twitter, and Instagram has not only transformed the way we share content but has also seamlessly integrated chat features, facilitating instant and engaging conversations with friends and contacts across the globe.

In addition to the ever-popular social media platforms, the Short Message Service (SMS) has maintained its prominence, cherished for its brevity and real-time communication capabilities.

Furthermore, applications like Snapchat prioritize user privacy, providing secure and ephemeral messaging options, enhancing the personal and intimate aspects of modern communication. Mobile chatting applications have surged in popularity, with WhatsApp, Telegram, and Signal leading the charge. These platforms have redefined the way we connect, offering instant messaging, voice calls, and even video calls on a global scale. This digital communication revolution has not only made interactions more accessible but has also broadened the spectrum of communication preferences, accommodating a wide range of needs and enhancing the internet's role as the central hub for human interaction in our interconnected world.

Instant Messaging

Instant Messaging (IM) constitutes private network communication between two users, while a chat session involves network communication among two or more users. These chat sessions can be either private, requiring invitations for participation, or public, allowing anyone to join. Approximately 100 million users engage in IM on the internet, defined as unique names on major public IM networks. Addressing the scalability challenge is a pivotal concern for IM service providers.

In an ideal scenario, each provider aims to have millions of customers logged into their systems simultaneously. Achieving this goal necessitates a scalable system architecture. Two approaches are available: symmetric and asymmetric. Symmetric architecture employs servers that perform identical functions, making it seamless for clients to interact with any server for their activities. Conversely, an asymmetric approach assigns specific server roles, such as login, user discovery, chat room maintenance, or message forwarding. Employing a client-server architecture empowers IM service providers with user control, helping surmount firewall-related technical obstacles. However, a downside is that both control and data paths pass through central servers, which poses challenges when scaling the service to accommodate millions of users.

Social Networking Sites

Chats on social networking platforms predominantly follow a peer-to-peer model, facilitating both one-on-one and group conversations. Unlike systems with queuing mechanisms, peer-to-peer chats do not involve waiting in line, ensuring immediate communication. These platforms employ algorithms to curate content, typically prioritizing the latest news updates and presenting online friends or individuals with whom the user frequently interacts. This user-centric approach enhances the overall experience by ensuring that pertinent and timely content is readily available.

However, the challenge of scalability emerges in this context. The absence of a queuing system can lead to variable response times for chat interactions, depending on the concurrent chat volumes. Maintaining consistent and responsive communication within these dynamic and evolving social networks presents a complex task, particularly when considering the need to address scalability concerns while adhering to real-time constraints.

CHAPTER - 3
REQUIREMENTS

Hardware Requirement

In hardware requirement we require all those components which will provide us the platformfor the development of the project. The minimum hardware required for the development of this project is as follows:

- Ram- minimum 128 MB
- Hard disk-minimum 5 GB
- Processor-Pentium 3
- Hard Disk with 7200 Rpm
- CD drive

These all are the minimum hardware requirement required for our project. We want to make our project to be used in any. Type of computer therefore we have largely taken minimum configuration.128 MB ram is used so that we can execute our project in the least possible RAM.5 GB hard disk is used because project takes less space to be executed orstored. Therefore, minimum hard disk is used. Other enhancements are according to the needs. However, this cannot be used in mobile phones because of the type of UI this project needs or supports therefore hence using it in phone is impossible

Software Requirements

Software can be defined as programs which run on our computer. It acts as petrol in the vehicle. It provides the relationship between the human and a computer. It is very important to run software to function the computer.Various software that are needed in this project for its development which are as follows:

- Python Tkinter
- Others-Visual Studio

We will be using visual basic as our front hand because it is easier to use and providesfeatures to the users which is used for the development of the project.

CHAPTER - 4
RESEARCH AND DESIGN

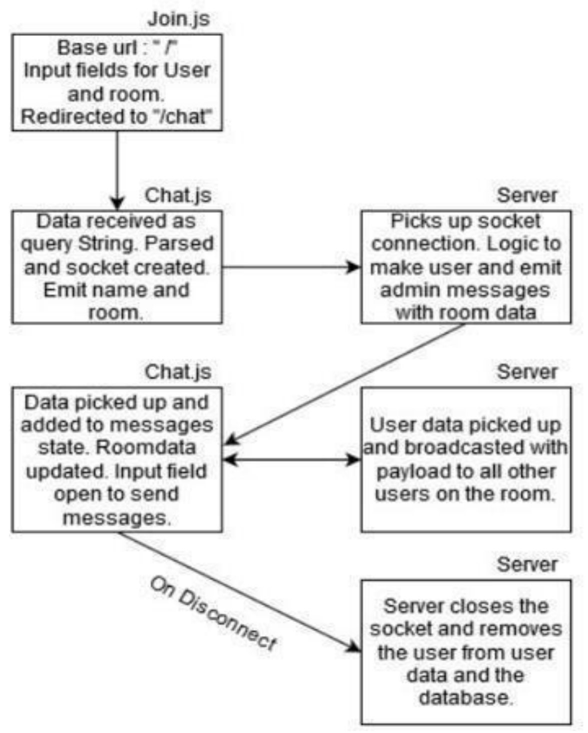
Design Consideration

We will be using Python Tkinter for the back end and Front end, which will be responsible for creating Connection and managing users in those rooms. Additionally, we will be using server client to enable real-time, bidirectional communication between the web server and the client (browser).

Our choice of Python for the backend of our application is a strategic one, given its efficiency and scalability in handling real-time interactions. Tkinter allows us to swiftly create and manage connections rooms, as well as seamlessly handle users' participation within those interactions. Complementing python, we have Socket and client, a crucial component that facilitates real-time, bidirectional communication between the web server and the client's application and server application, another component that shows you all you need to see from the eyes of the user, that helps in creating easy and interactive user interfaces.

With this combination of technologies, we're well-equipped to provide a dynamic, real-time chat experience, ensuring users can engage in text-based communication efficiently and without delay, ultimately enhancing the overall user experience.

Flow Chart:



Join.js
Base url : " /" Input fields for User and room. Redirected to "/chat"
Chat.js
Data received as query String. Parsed and socket created. Emit name and room.
Server
Picks up socket connection. Logic to make user and emit admin messages with room data
Chat.js
Data picked up and added to messages state. Roomdata updated. Input field open to send messages.
Server
User data picked up and broadcasted with payload to all other users on the room.
On Disconnect
Server
Server closes the socket and removes the user from user data and the database.

This represents the flow of logic and the main files. All other files are either helperfunctions, router or react components. The architecture of our application is designed around a clear and efficient flow of logic, with the primary focus on a few key files. At the core is our backend, implemented in Python an Tkinter package, which manages the creation and coordination of connection between server and client, as well as the handling of user interactions within this connection. The use of specific files for helper function, routers, and PIP components allows for a modular and organized approach, keeping our codebase clean and maintainable. Helper functions assist in various tasks, enhancing code readability and reusability, while routers define endpoints and routes for client-server communication. IP components, on the other hand, ensuring a seamless and engaging experience for clients. This separation of concerns streamlines development, debugging, and collaboration, enabling our team to efficiently create a feature-rich chat application with a structured codebase. And also connect the port using the port number which also improve the performance.

CHAPTER - 5
IMPLEMENTATION

Languages and Tools used.

1. Python
2. socket.io

Methodology

◈ The user will interact with the tool using a GUI

◈ The user enters a username and a room name

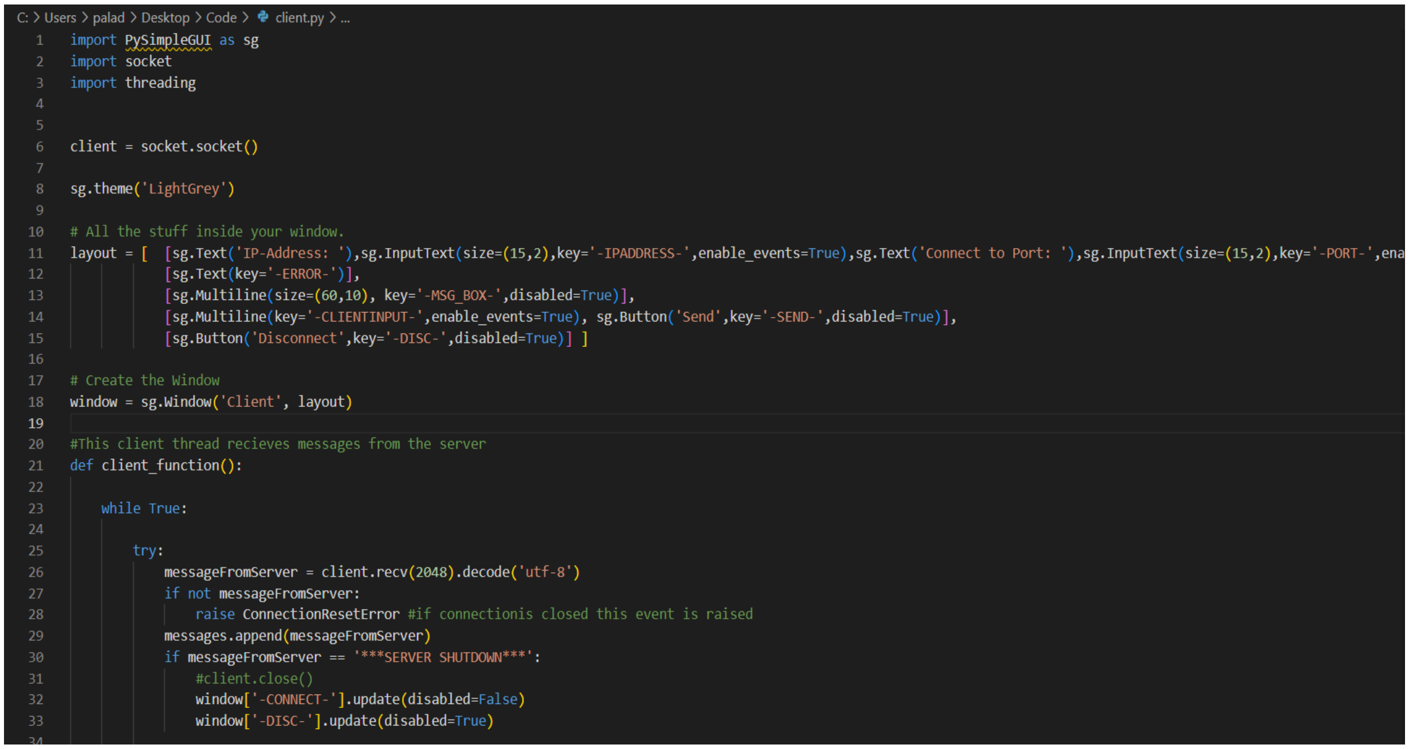
◈ A Connection is created, and it has a list form and a chat form Connection

◈ The list form contains the IP and address of the people in the Connection by port number

◈ The chat form makes the actual communication possible in the form of text and it will enable to communicate and secure.

Code Snippets of Functionalities

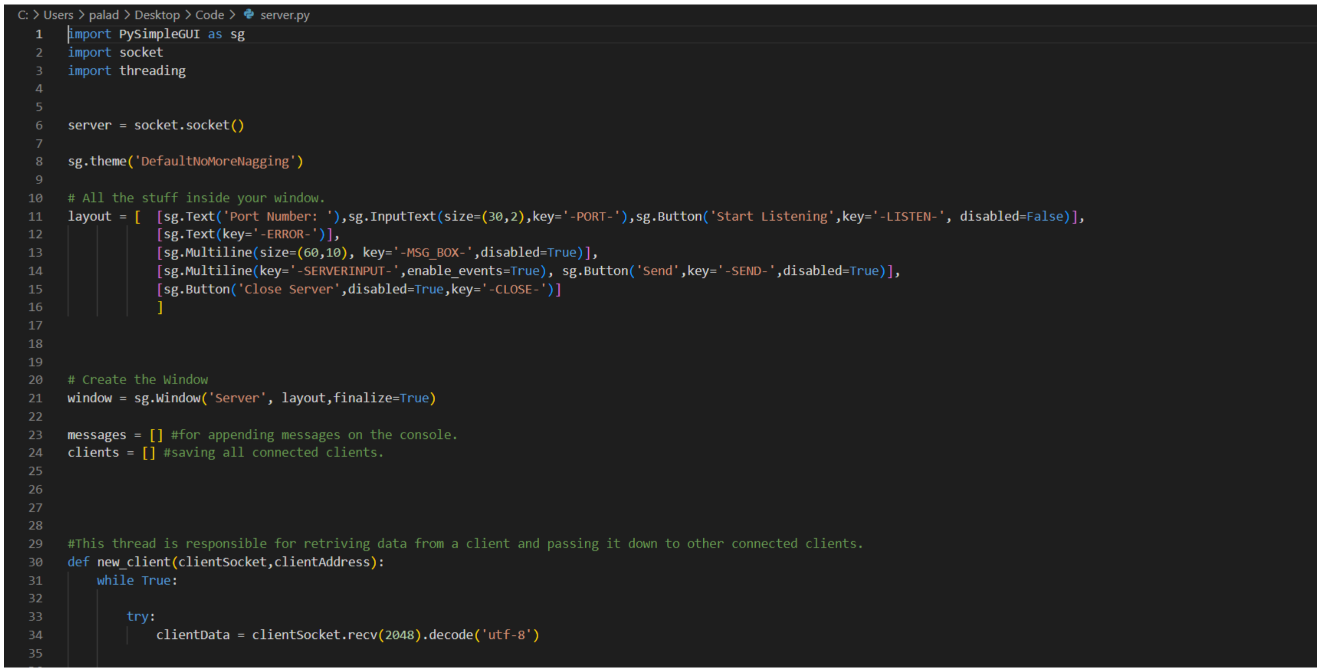
Server:



C: > Users > palad > Desktop > Code > client.py > ...

```python
import PySimpleGUI as sg
import socket
import threading


client = socket.socket()

sg.theme('LightGrey')

# All the stuff inside your window.
layout = [  [sg.Text('IP-Address: '),sg.InputText(size=(15,2),key='-IPADDRESS-',enable_events=True),sg.Text('Connect to Port: '),sg.InputText(size=(15,2),key='-PORT-',ena
            [sg.Text(key='-ERROR-')],
            [sg.Multiline(size=(60,10), key='-MSG_BOX-',disabled=True)],
            [sg.Multiline(key='-CLIENTINPUT-',enable_events=True), sg.Button('Send',key='-SEND-',disabled=True)],
            [sg.Button('Disconnect',key='-DISC-',disabled=True)] ]

# Create the Window
window = sg.Window('Client', layout)

#This client thread recieves messages from the server
def client_function():

    while True:

        try:
            messageFromServer = client.recv(2048).decode('utf-8')
            if not messageFromServer:
                raise ConnectionResetError #if connectionis closed this event is raised
            messages.append(messageFromServer)
            if messageFromServer == '***SERVER SHUTDOWN***':
                #client.close()
                window['-CONNECT-'].update(disabled=False)
                window['-DISC-'].update(disabled=True)
```

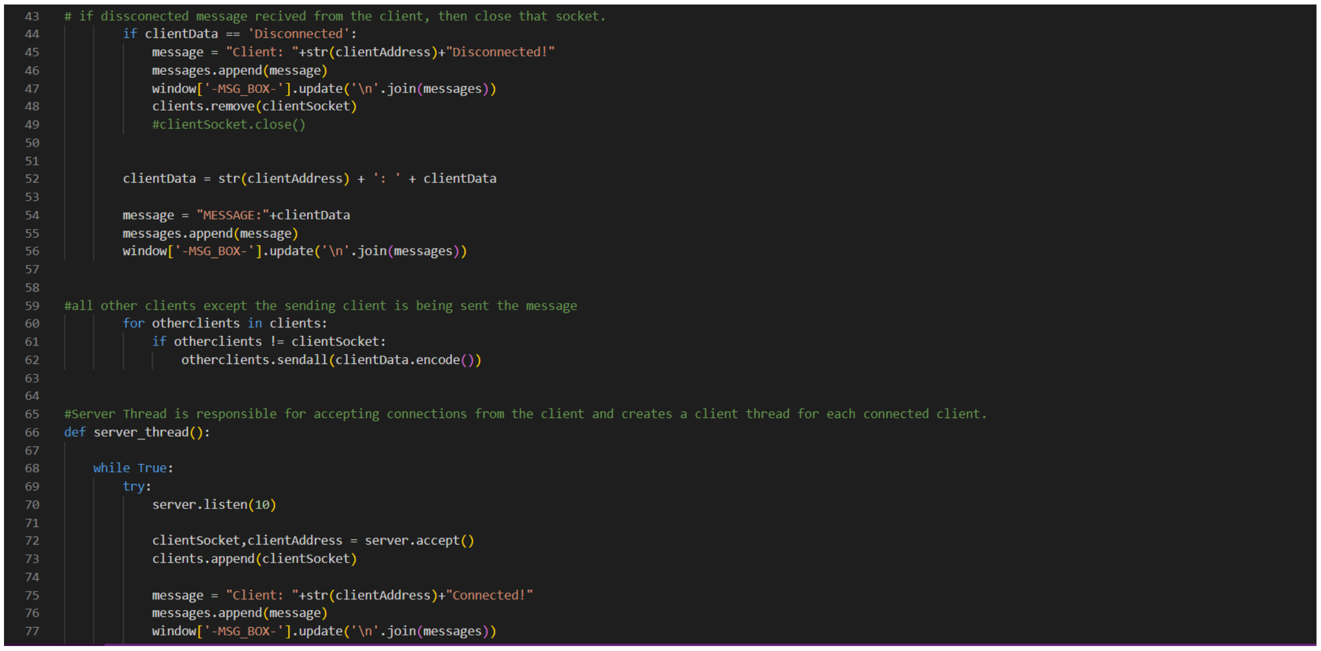
Client:



C: > Users > palad > Desktop > Code > server.py

```python
import PySimpleGUI as sg
import socket
import threading


server = socket.socket()

sg.theme('DefaultNoMoreNagging')

# All the stuff inside your window.
layout = [  [sg.Text('Port Number: '),sg.InputText(size=(30,2),key='-PORT-'),sg.Button('Start Listening',key='-LISTEN-', disabled=False)],
            [sg.Text(key='-ERROR-')],
            [sg.Multiline(size=(60,10), key='-MSG_BOX-',disabled=True)],
            [sg.Multiline(key='-SERVERINPUT-',enable_events=True), sg.Button('Send',key='-SEND-',disabled=True)],
            [sg.Button('Close Server',disabled=True,key='-CLOSE-')]
            ]



# Create the Window
window = sg.Window('Server', layout,finalize=True)

messages = [] #for appending messages on the console.
clients = [] #saving all connected clients.




#This thread is responsible for retriving data from a client and passing it down to other connected clients.
def new_client(clientSocket,clientAddress):
    while True:

        try:
            clientData = clientSocket.recv(2048).decode('utf-8')

```

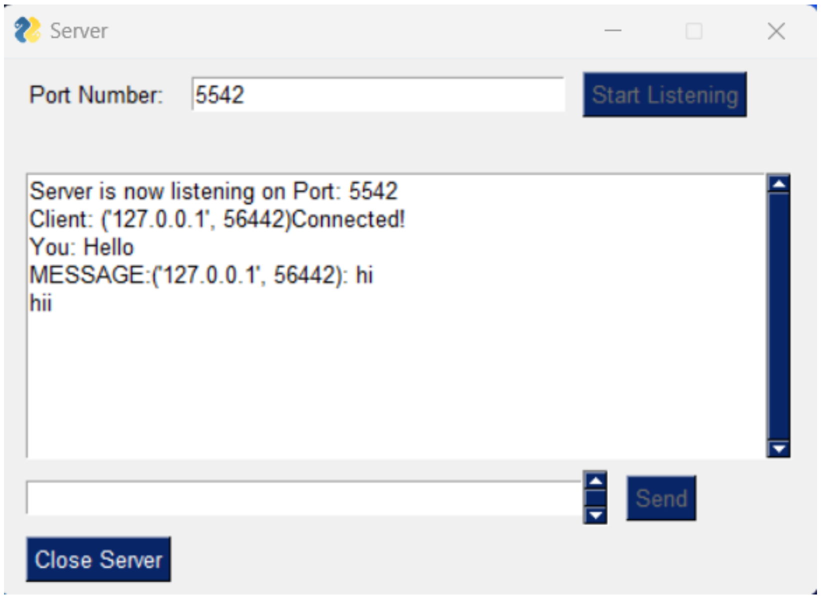
The chat Connection



```
# if dissconected message recived from the client, then close that socket.
        if clientData == 'Disconnected':
            message = "Client: "+str(clientAddress)+"Disconnected!"
            messages.append(message)
            window['-MSG_BOX-'].update('\n'.join(messages))
            clients.remove(clientSocket)
            #clientSocket.close()


        clientData = str(clientAddress) + ': ' + clientData

        message = "MESSAGE:"+clientData
        messages.append(message)
        window['-MSG_BOX-'].update('\n'.join(messages))


#all other clients except the sending client is being sent the message
        for otherclients in clients:
            if otherclients != clientSocket:
                otherclients.sendall(clientData.encode())


#Server Thread is responsible for accepting connections from the client and creates a client thread for each connected client.
def server_thread():

    while True:
        try:
            server.listen(10)

            clientSocket,clientAddress = server.accept()
            clients.append(clientSocket)

            message = "Client: "+str(clientAddress)+"Connected!"
            messages.append(message)
            window['-MSG_BOX-'].update('\n'.join(messages))
```

CHAPTER - 6
RESULTS AND DISCUSSION

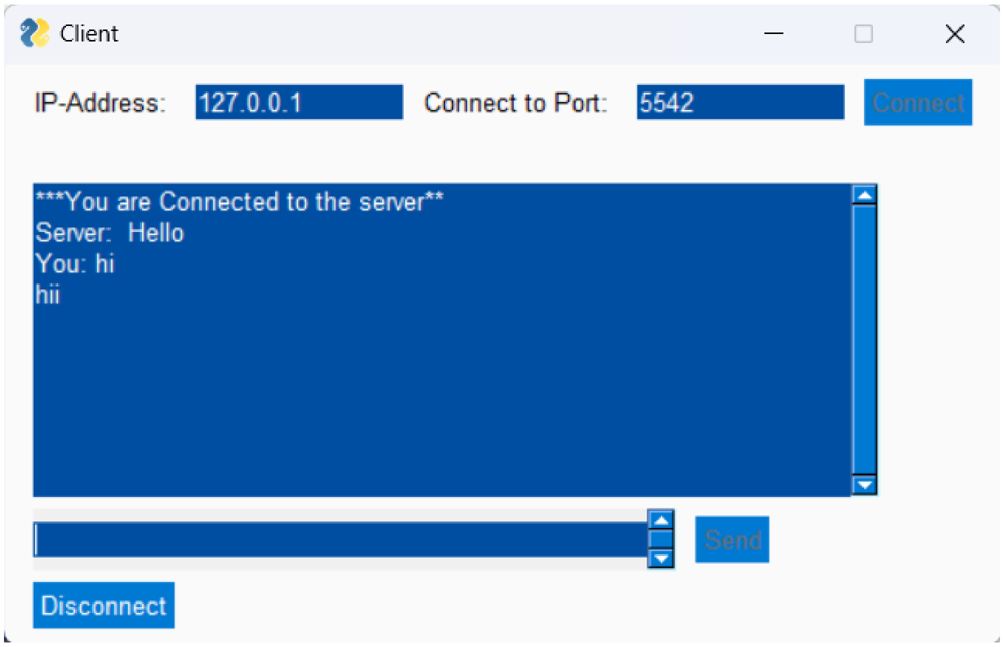
Server connection:



5542

```
Server is now listening on Port: 5542
Client: ('127.0.0.1', 56442)Connected!
You: Hello
MESSAGE:('127.0.0.1', 56442): hi
hii
```

Client Connection:



127.0.0.1

5542

```
***You are Connected to the server**
Server:  Hello
You: hi
hii
```

CHAPTER - 7
FUTURE SCOPE

To further improve your chat server application using Socket.io, you can consider. Following features enhancements:

1. User Authentication: Implement user authentication to secure your chat application. Use techniques like JWT or OAuth for authentication and authorization.

2. Message History: Store and retrieve message history so that users can access past conversations even after they log out or close the app.

3. Group Chats: Enable the creation of group chats and channels to facilitate team collaboration and discussions among multiple users.

4. File Sharing: Add the capability for users to share files and images within the chat. Implement secure storage and sharing mechanisms.

5. Emojis and Rich Text: Enhance the chat interface with support for emojis, stickers, and rich text formatting.

6. Continuous Testing and Improvement: Regularly test, monitor, and optimize the application for performance, scalability, and security. Keep up with the latest updates and security patches for Socket.io.

7. UX Improvements: Continuously work on improving the overall user experience, such as responsive design for various screen sizes and devices.

CONCLUSION

The development of a chat server application using Python is a valuable project, enabling real-time communication between clients and facilitating various use cases, such as chat applications, gaming, and collaborative tools. Socket.io offers a powerful framework for building these applications, with its simplicity and reliability making ita popular choice among developers.

In conclusion, here are some key points to consider:

1. Real-time Communication: Socket.io has proven to be a robust solution for real-time communication. It allows for bi-directional communication between clients and the server, enabling instant updates and messages without the need for constant polling.

2. Scalability: The application can be easily scaled to handle many usersby deploying it on multiple servers and using load balancers. Socket. Io's architecturesupports horizontal scaling.

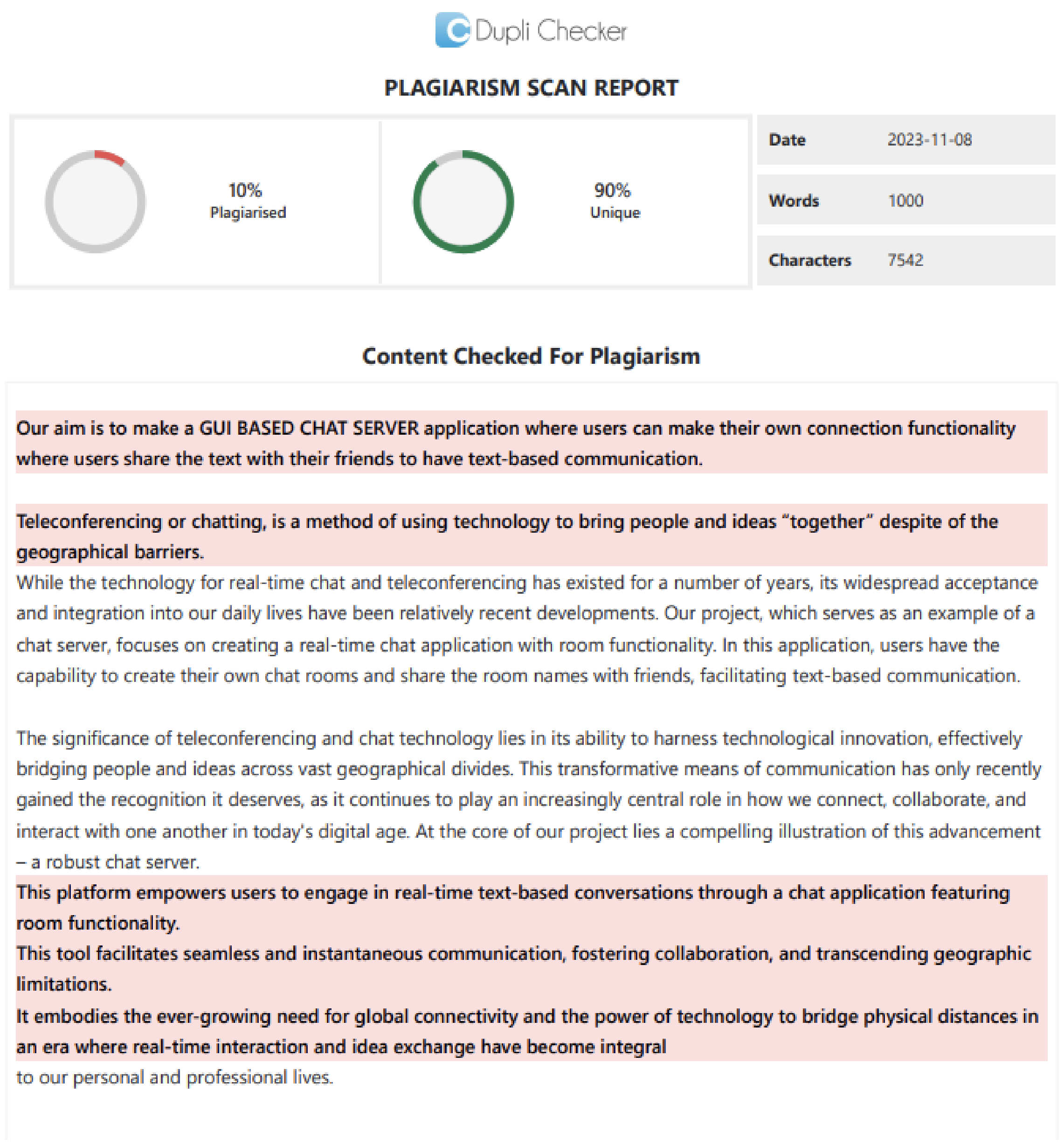
3. Features and Functionality: The application can be enhanced with features like user authentication, message history, group chats, file sharing, and push notifications to make it more versatile and competitive.

REFERENCES

- MDN Docs
- W3S Schools
- Npm docs
- Socket.io docs
- Stackoverflow
- Geeks for geeks
- Paul Goes, Noyak Ilk, J. Leon Zhao, Onadmission control policy for multi-taskingLive-Chat Service Agents.
- Arora A., Sinha M.(2012). Web ApplicationTesting: A Review on Techniques, Toolsand State of Art. In International Journal of Scientific & Engineering Research, Volume3, Issue

  2, February-2012 1 ISSN 2229-5518

PLAGIARISM REPORT



Dupli Checker

PLAGIARISM SCAN REPORT

10% Plagiarised	90% Unique	**Date**	2023-11-08
		Words	1000
		Characters	7542

Content Checked For Plagiarism

Our aim is to make a GUI BASED CHAT SERVER application where users can make their own connection functionality where users share the text with their friends to have text-based communication.

Teleconferencing or chatting, is a method of using technology to bring people and ideas “together” despite of the geographical barriers.
While the technology for real-time chat and teleconferencing has existed for a number of years, its widespread acceptance and integration into our daily lives have been relatively recent developments. Our project, which serves as an example of a chat server, focuses on creating a real-time chat application with room functionality. In this application, users have the capability to create their own chat rooms and share the room names with friends, facilitating text-based communication.

The significance of teleconferencing and chat technology lies in its ability to harness technological innovation, effectively bridging people and ideas across vast geographical divides. This transformative means of communication has only recently gained the recognition it deserves, as it continues to play an increasingly central role in how we connect, collaborate, and interact with one another in today's digital age. At the core of our project lies a compelling illustration of this advancement – a robust chat server.
This platform empowers users to engage in real-time text-based conversations through a chat application featuring room functionality.
This tool facilitates seamless and instantaneous communication, fostering collaboration, and transcending geographic limitations.
It embodies the ever-growing need for global connectivity and the power of technology to bridge physical distances in an era where real-time interaction and idea exchange have become integral
to our personal and professional lives.