**Mini Project Report on**



**C.L.I. Based Chat Tool**  


**Submitted in partial fulfilment of the requirement for the award of the degree of**

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE & ENGINEERING**

**Submitted by:**

**Student Name-Aman Kaushik**   **University Roll No.- 2018662**

***Under the Mentorship of***

**Mentor Name- Ass. Prof. Yuvraj Joshi**



**Department of Computer Science and Engineering**

**Graphic Era (Deemed to be University)**

**Dehradun, Uttarakhand**

**july 2023**



**CANDIDATE’S DECLARATION**

I hereby certify that the work which is being presented in the project report entitled **“C.L.I. Based Chat Tool”** in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineeringof the Graphic Era (Deemed to be University), Dehradun shall be carried out by the under the mentorship of **Asst. pro. Yuvraj Joshi, PGT**, Department of Computer Science and Engineering, Graphic Era (Deemed to be University), Dehradun.

Name-Aman Kaushik University Roll no-2018662 **signature**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Description** | **Page No.** |
| Chapter 1 | Problem statement | 1 |
| Chapter 2 | Abstract | 2 |
| Chapter 3 | motivation | 3 |
| Chapter 4 | introduction | 4 |
| Chapter 5  Chapter 6  Chapter 7  Chapter 8 | System configuration  Methodology  Implementation and result  conclusion | 5  6-7  8-9  9-10 |
|  | References | 11 |

# **CHAPTER-1**

# **PROOBLEM STATEMENT :**

The objective of this project is to design and implement a chat application that facilitates real-time communication between multiple clients using socket programming in Python. The application should allow clients to connect to a central server and exchange messages with each other. The server should be capable of handling concurrent connections from multiple clients, maintaining a list of connected clients, and broadcasting received messages to all clients except the sender. The application should provide a user-friendly interface, allowing clients to input messages and view the chat history. Additionally, the system should handle client disconnections gracefully, notifying other clients when a client joins or leaves the chat.

**CHAPTER-2**

# **ABSTRACT :**

This project aims to develop a chat application using socket programming in Python. The application consists of a client-side script and a server-side script that enable multiple clients to communicate with each other in real-time. The client script allows users to input messages and send them to the server, while the server script receives these messages and broadcasts them to all connected clients. The project utilizes multithreading to handle concurrent client connections and provides a simple and efficient communication platform.

**CHAPTER-3**

# **MOTIVATION** :

Motivation:

The motivation behind developing the chat application using socket programming in Python is to create a real-time communication platform that allows users to connect and interact with each other conveniently. The project aims to address the need for a simple and efficient chat application that can be used in various scenarios, such as team collaboration, social networking, or online gaming.

1. Real-time Communication: The project aims to provide users with a seamless and real-time communication experience. By utilizing socket programming, messages can be transmitted instantly between clients, enabling quick and efficient exchanges of information.

2. Simplicity and Accessibility: The project focuses on creating a user-friendly chat application that is easy to understand and use. By developing a simple interface for clients to input messages and view the chat history, the application becomes accessible to users with different levels of technical expertise.

3. Concurrent Connections: The project addresses the need for handling concurrent connections efficiently. By implementing multithreading, the application can handle multiple clients simultaneously, ensuring that each client's communication is independent and synchronized with other clients.

4. Collaboration and Social Interaction: The chat application promotes collaboration and social interaction among users. It allows users to connect and communicate with colleagues, friends, or like-minded individuals, fostering a sense of community and enhancing productivity and connectivity.

5. Practical Application: Chat applications have a wide range of practical applications in today's digital world. They can be used for remote team collaboration, customer support, online gaming, or social networking. By developing a chat application using socket programming in Python, this project aims to provide a versatile tool that can be adapted to various scenarios.

Overall, the motivation for this project lies in creating a functional and user-friendly chat application that facilitates real-time communication, promotes collaboration, and serves as a practical solution for individuals and businesses in need of an efficient communication platform.

**CHAPTER-4**

# **INTRODUCTION:**

The development of a chat application using socket programming in Python offers a reliable and efficient solution for real-time communication between multiple clients. This project aims to create a user-friendly chat application that allows users to connect to a central server and exchange messages with each other seamlessly. By utilizing socket programming and multithreading, the application ensures concurrent connections and synchronized communication.

In today's digital era, communication plays a vital role in various domains, including team collaboration, social networking, and online gaming. Traditional chat applications often rely on centralized servers to facilitate communication between clients. However, this project leverages socket programming to establish direct connections between clients and the server, enabling efficient data transmission and real-time updates.

The primary objective of this project is to design and implement a chat application that addresses the need for instant and reliable communication. The client-side script allows users to input messages, which are then transmitted to the server. The server receives these messages and broadcasts them to all connected clients, excluding the original sender. Multithreading ensures that concurrent client connections are handled independently, ensuring smooth communication for all participants.

Additionally, the application provides a user-friendly interface that allows clients to input messages and view the chat history conveniently. With the inclusion of client joining and leaving notifications, users are kept informed about the participants in the chat room.

Overall, this project serves as a practical solution for real-time communication needs, catering to scenarios such as team collaboration, social interaction, and online gaming. By harnessing the power of socket programming in Python, the chat application offers simplicity, efficiency, and an enhanced user experience for seamless communication among clients.

**CHAPTER-5**

# **SYSTEM CONFIGURATION:**

This project can run on commodity hardware. We ran entire project on an Intel I5 processor with 8 GB Ram, 2 GB Nvidia Graphic Processor, It also has 2 cores which runs at 1.7 GHz, 2.1 GHz respectively. First part of the is training phase which takes 10-15 mins of time, and the second part is testing part which only takes few seconds to make predictions and calculate accuracy.

## Hardware Requirements:

* RAM:- 8GB
* Storage:- 1TB
* CPU:- 2GHz or faster
* Architecture:- 32-bit or 64-bit

## System Requirements:

* Dependencies such as socket and threading.
* Operating System: windows 7 and above or Linux based OS or MAC OS.

**CHAPTER-6**

# METHODOLOGY

The development of the chat application using socket programming in Python follows a systematic approach to ensure its successful implementation. The methodology includes the following key steps:

1. Requirement Analysis: Understand the specific requirements of the chat application, including the desired features, user interface, communication protocols, and client-server interaction. This step involves gathering user requirements and defining the scope of the project.

2. System Design: Based on the requirements, design the overall system architecture of the chat application. Identify the components, such as the client-side script and server-side script, and define their functionalities and interactions. Determine the data flow, message formats, and error handling mechanisms.

3. Implementation: Develop the client-side script and server-side script in Python, following the design specifications. Implement the socket programming logic to establish connections, send and receive messages, and handle concurrent client connections using multithreading. Ensure the scripts are well-structured, modular, and adhere to best coding practices.

4. Testing: Conduct thorough testing of the chat application to ensure its functionality and reliability. Test various scenarios, such as multiple client connections, message transmission, handling of client disconnections, and error scenarios. Perform unit testing, integration testing, and system testing to identify and resolve any bugs or issues.

5. User Interface Design: Design a user-friendly interface for the client-side script. Create an interactive console-based interface that allows users to input messages, view the chat history, and receive notifications. Ensure the interface is intuitive, responsive, and easy to navigate.

6. Documentation: Prepare comprehensive documentation that includes the project overview, system architecture, installation instructions, usage guide, and any other relevant information. Document the design decisions, implementation details, and troubleshooting steps. This documentation will serve as a reference for future maintenance and enhancements.

7. Deployment: Deploy the chat application on a suitable server environment, ensuring that the server is accessible to the intended clients. Configure the necessary network settings and firewall rules to allow client connections to the server on the specified port. Provide instructions for clients to connect to the server and use the chat application.

8. Maintenance and Enhancements: Regularly monitor and maintain the chat application, addressing any issues or bugs that arise. Incorporate user feedback and make necessary enhancements or feature additions to improve the application's functionality and user experience.

By following this methodology, the chat application can be systematically developed, tested, and deployed, ensuring a robust and user-friendly communication platform for clients.

# **CHAPTER-7**

# IMPLEMENTATION AND RESULTS:

The implementation of the chat application using socket programming in Python involves developing both the client-side and server-side scripts.

To run the chat application:

1. Save the client-side script as "client.py" and the server-side script as "server.py" in separate files.

2. Open two command prompt (terminal) windows.

3. In one window, navigate to the folder containing "server.py" and run the command `python server.py` to start the server.

4. In the other window, navigate to the folder containing "client.py" and run the command `python client.py` to start a client.

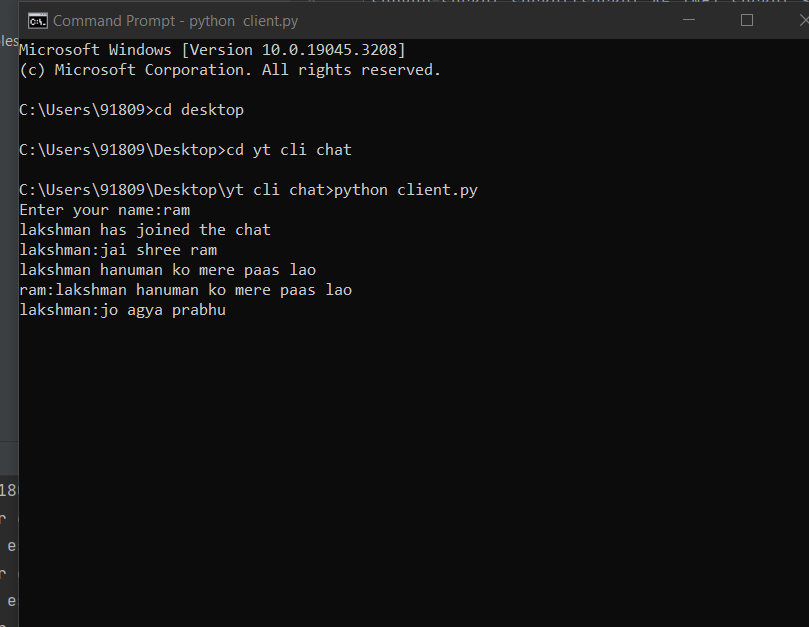
5. Repeat step 4 to start additional clients and establish connections with the server.

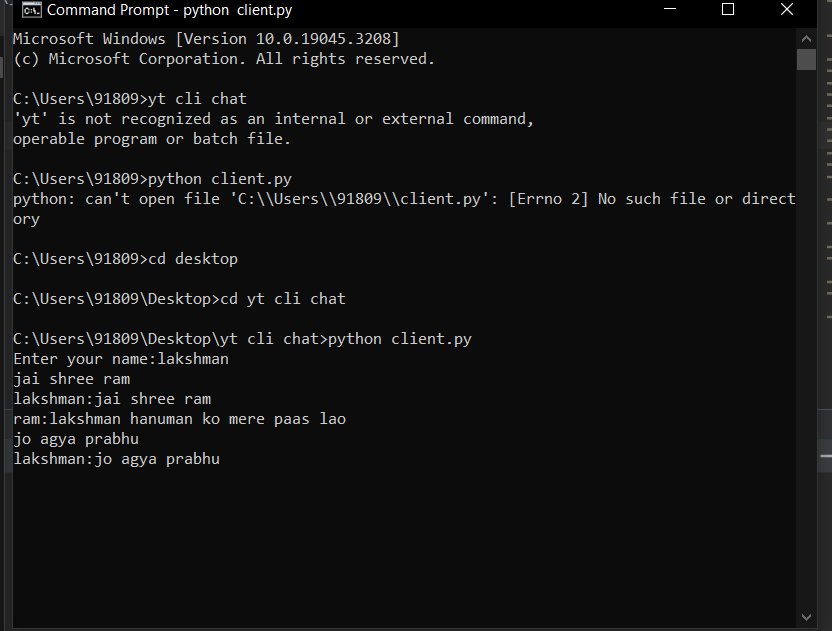
6. Enter a name for each client when prompted by the client-side script.

7. Clients can now send and receive messages through the chat application.

Note: Make sure the server is running before starting the clients. Additionally, ensure that the server is accessible to the clients (e.g., running on the same machine or accessible via the network). Adjust the host and port values in the code if needed.

This implementation provides a basic foundation for a chat application using socket programming in Python. Further enhancements, such as user interface improvements, message formatting, or additional features, can be added based on specific requirements.





**CHAPTER-8**

# **CONCLUSION: -**

The chat application developed using socket programming in Python provides an efficient and reliable solution for real-time communication between multiple clients. The project successfully achieves its objective of creating a user-friendly chat platform that allows clients to connect to a central server and exchange messages seamlessly.

Through the implementation of socket programming, the application establishes direct connections between clients and the server, enabling fast and efficient message transmission. The use of multithreading ensures that concurrent client connections are handled independently, ensuring smooth communication for all participants.

The client-side script offers a user-friendly interface, allowing clients to input messages and view the chat history conveniently. The server-side script efficiently handles client connections, broadcasting messages to all connected clients while gracefully handling disconnections.

The project's methodology ensures a systematic approach to development, including requirement analysis, system design, implementation, testing, user interface design, documentation, and deployment. This methodology ensures that the application meets the desired requirements and functions reliably.

Overall, the chat application provides a practical solution for real-time communication needs in various scenarios, such as team collaboration, social networking, or online gaming. By leveraging socket programming and Python's capabilities, the application offers simplicity, efficiency, and enhanced user experience for seamless communication among clients.

Further enhancements and features can be added to the chat application, such as message encryption, file sharing, or user authentication, to cater to specific requirements and enhance its functionality. With ongoing maintenance and updates, the chat application can continue to serve as an effective communication platform for users in diverse contexts.

# **REFERENCES:-**

Some general references related to socket programming and chat applications that may be useful for project:

1. "Python Network Programming: Conquer all your networking challenges with the powerful Python language" by Dr. M. O. Faruque Sarker and Sam Washington.

- This book provides a comprehensive guide to network programming with Python, including socket programming and building network applications.

2. "Computer Networking: A Top-Down Approach" by James F. Kurose and Keith W. Ross.

- This textbook covers various networking concepts and protocols, including socket programming, TCP/IP, and network applications.

3. "Beej's Guide to Network Programming" by Brian "Beej" Hall.

- This online guide is a widely referenced resource for learning socket programming and network programming concepts. It provides practical examples and explanations.

4. "Chat Applications: Design, Development, and Deployment" by Hui Zang, Shehram Talpur, and Zahid Anwar.

- This research paper discusses the design and development of chat applications, including architectural considerations, protocols, and deployment strategies.

Please note that the project implementation provided earlier is based on common practices and general knowledge of socket programming. It does not directly rely on specific sources or research papers.