Chat Application

A PROJECT REPORT

Submitted by

SRI HARISH(2303811710421155)

in partial fulfillment of requirements for the award of the course

CGB1201 - JAVA PROGRAMMING

In

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

NOVEMBER- 2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY (AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on **"CHAT APPLICATION"** is the bonafide work of **SRI HARISH (2303811710421155)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

SIGNATURE	**SIGNATURE**
Mrs.A.Delphin Carolina Rani, M.E.,Ph.D.,	Mr. A. Malarmannan, M.E.,
HEAD OF THE DEPARTMENT	**SUPERVISOR**
PROFESSOR	ASSISTANT PROFESSOR
Department of CSE	Department of CSE
K.Ramakrishnan College of Technology (Autonomous)	K.Ramakrishnan College of Technology (Autonomous)
Samayapuram–621112.	Samayapuram–621112.

Submitted for the viva-voce examination held on 06/12/2024

INTERNAL EXAMINER EXTERNAL EXAMINER

DECLARATION

I declare that the project report on **"CHAT APPLICATION"** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1201 - JAVA PROGRAMMING.**

.

Signature

SRI HARISH

Place: Samayapuram

Date: 06/12/2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution "**K.Ramakrishnan College of Technology (Autonomous)**", for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding to our project and offering adequate duration in completing our project.

I would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.,** Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. A. DELPHIN CAROLINA RANI, M.E.,Ph.D.,** Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **MR. A. MALARMANNAN, M.E.,** Department of **COMPUTER SCIENCE AND ENGINEERING,** for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

VISION OF THE INSTITUTION

To serve the society by offering top-notch technical education on par with global standards

MISSION OF THE INSTITUTION

- Be a center of excellence for technical education in emerging technologies by exceeding the needs of the industry and society.
- Be an institute with world class research facilities
- Be an institute nurturing talent and enhancing the competency of students to transform them as all-round personality respecting moral and ethical values

VISION OF DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

MISSION OF DEPARTMENT

M1: Industry Specific: To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

M2: Research: To prepare students for research-oriented activities.

M3: Society: To empower students with the required skills to solve complex technological problems of society.

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

2. PEO2: Employability Skills and Research

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

3. PEO3: Ethics and Values

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Domain Knowledge

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

PSO 2: Quality Software

To apply software engineering principles and practices for developing quality software for scientific and business applications.

PSO 3: Innovation Ideas

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice

7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

This project focuses on developing a **real-time chat application** using Java. The application provides users with a platform to send and receive messages efficiently, offering features such as one-to-one messaging, group chats, and basic multimedia sharing. Built with a client-server architecture, the program ensures secure communication using techniques such as encryption for message confidentiality and authentication for user validation.

The front-end of the application is developed using Java's **Swing** library to deliver an intuitive and user-friendly interface, while the back-end leverages **Java Socket Programming** for handling communication between clients and the server. Additionally, the application supports multi-threading to allow simultaneous communication among multiple users.

The chat application is designed to be scalable, reliable, and easily extensible, catering to modern communication needs. This project serves as a practical demonstration of Java's networking capabilities and object-oriented programming principles, offering a robust foundation for further enhancements such as advanced encryption, voice/video integration, or cloud storage solutions.

ABSTRACT WITH POs AND PSOs MAPPING

CO 5 : BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POs MAPPED	PSOs MAPPED
This project focuses on developing a **real-time chat application** using Java. The application provides users with a platform to send and receive messages efficiently, offering features such as one-to-one messaging, group chats, and basic multimedia sharing. Built with a client-server architecture, the program ensures secure communication using techniques such as encryption for message confidentiality and authentication for user validation.	**PO1 -3** **PO2 -3** **PO3 -3** **PO4 -3** **PO5 -3** **PO6 -3** **PO7 -3** **PO8 -3** **PO9 -3** **PO10 -3** **PO11-3** **PO12 -3**	**PSO1 -3** **PSO2 -3** **PSO3 -3**

Note: 1- Low, 2-Medium, 3- High

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Objective

The objective of the provided code is to create a simple chat application with a graphical user interface (GUI) that facilitates real-time communication between a client and a server using Java sockets. The application demonstrates basic networking concepts and GUI design, allowing message exchange via a text area, input field, and send button.The application employs a multi-threaded approach, enabling simultaneous message handling and GUI responsiveness. The server waits for incoming connections and manages multiple clients, ensuring seamless message broadcasting. Additionally, the client's user interface updates dynamically to display received messages and supports user input through an intuitive GUI design.

1.2 Overview

The project is a basic chat application implemented in Java that enables real-time communication between a client and a server using sockets. It features a simple graphical user interface (GUI) created with AWT, which includes a text area for displaying messages, a text field for input, and a send button for transmitting messages. The server listens for client connections on a specified port, while the client connects to the server to exchange messages. Multithreading is used to handle real-time message updates while keeping the GUI responsive. This project demonstrates core concepts of networking, threading, and GUI design in Java. Furthermore, the application ensures that messages are delivered reliably and promptly, maintaining the integrity of the communication. The server handles multiple client connections simultaneously, thanks to its efficient multithreading approach. The client-side interface is designed to be user-friendly, allowing users to send and receive messages effortlessly.

1.3 Java Programming Concepts

Object-Oriented Programming Concepts (OOPs)

The project demonstrates key OOP principles:

Encapsulation:
The Main class encapsulates the data (GUI components like TextField, TextArea, and networking objects such as Socket) and methods (actionPerformed, run) required for the chat application's functionality.

Inheritance:
The Main class extends Frame, inheriting properties and methods from the AWT framework to create the graphical user interface.

Polymorphism:
Method overriding is used in the project, such as the actionPerformed method to handle button click events in a customized way.

Abstraction:
The project abstracts the complexities of socket communication and GUI interactions by encapsulating these details within the Main class, focusing only on essential functionalities like message exchange.

Project-Related Java Concepts

The project also employs several Java programming concepts directly related to its implementation:

Sockets and Networking:
It uses Socket (client) and ServerSocket (server) to establish a communication link between the client and server. Message transmission is handled using DataInputStream and DataOutputStream.

Multithreading:
A separate thread is used to continuously listen for incoming messages, ensuring the application handles real-time communication without freezing the GUI.

AWT (Abstract Window Toolkit):

The GUI is built using AWT components such as Frame, TextArea, TextField, and Button. Event handling is implemented via the ActionListener interface to manage user interactions.

I/O Streams:

Input and output streams (DataInputStream and DataOutputStream) are used to send and receive messages over the socket connection.

Exception Handling:

try-catch blocks are used throughout the program to handle exceptions like IOException during socket operations or GUI events, ensuring the application does not crash unexpectedly.

Event Handling:

The actionPerformed method is triggered when the user clicks the send button, allowing real-time communication.

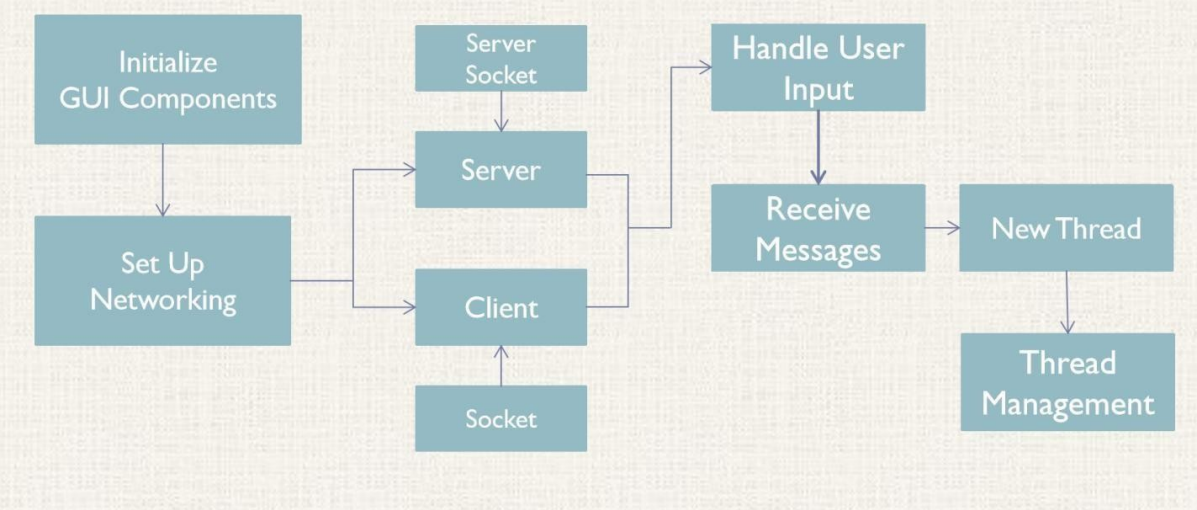
This project effectively combines fundamental OOP principles with practical Java concepts like networking, multithreading, and GUI design to create a functional and interactive chat application.

CHAPTER 2
PROJECT METHODOLOGY

2.1 Proposed Work

The proposed work aims to create a simple real-time chat application that enables communication between a client and a server. The application will use Java's networking features to establish a connection and exchange messages. The user interface will be built with Java AWT components, featuring a text area to display messages, a text field for input, and a send button. Multithreading will be implemented to ensure smooth real-time communication by handling both message reception and the GUI simultaneously. The application will also include basic error handling to maintain stability during use. Future enhancements may include multiple client support, secure messaging, or the ability to send files. The project will serve as a practical demonstration of network programming and GUI design in Java. It will provide a solid foundation for developing more complex communication applications in the future. Additionally, by using Java's object-oriented principles, the application will maintain clean and efficient code structure, making it easier to manage and expand. This project will also offer an opportunity to explore issues such as network latency, connection stability, and performance optimization. Ultimately, the application will aim to offer a user-friendly platform for communication, showcasing the integration of key programming concepts.

2.2 Block Diagram



CHAPTER 3
MODULE DESCRIPTION

3.1 Connection and Communication Module

This module is responsible for managing the communication between the client and the server. It establishes the connection using Java's Socket and ServerSocket classes and handles the exchange of messages using DataInputStream and DataOutputStream. The server continuously listens for incoming client connections, while the client sends and receives messages in real-time.

3.2 User Interface (UI) Module

The UI module is responsible for creating the graphical layout of the chat application using Java AWT components. It provides a TextArea to display messages, a TextField to type new messages, and a Button to send the message. The interface is designed to be intuitive and responsive, providing a smooth user experience during chat sessions.

3.3 Real-Time Messaging Module

This module ensures that the application can handle real-time message exchange between the client and server. It utilizes multithreading to run the message-reception task in a separate thread from the UI thread. This guarantees that the application remains responsive while listening for incoming messages without interrupting the user interface.

3.4 Message Processing and Exchange Module

The Message Processing module is responsible for handling the text-based communication between the client and server. It captures user input from the TextField, processes it, and sends it to the server. The server, in turn, processes the message and sends it back to the client, where it is displayed in the chat window.

3.5 Error Handling and Stability Module

The Message Processing module is responsible for handling the text-based communication between the client and server. It captures user input from the TextField, processes it, and sends it to the server. The server, in turn, processes the message and sends it back to the client, where it is displayed in the chat window.

These five modules together form the backbone of the chat application, ensuring efficient communication between the client and server, a responsive user interface, and a stable application. They address all the core aspects of the system, from networking and messaging to user interaction and error management.

CHAPTER 4
CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

This chat application successfully demonstrates key Java concepts like networking, multithreading, and GUI design. By integrating these elements, it enables real-time communication between a client and a server. The modular approach ensures each component functions independently while contributing to the application's overall stability. The use of object-oriented principles like encapsulation and inheritance makes the code maintainable and scalable. This project serves as a strong foundation for future enhancements and provides a valuable learning experience in Java programming and networked applications.

4.2 FUTURE SCOPE

The chat application has a lot of potential for growth and improvement. One key area for enhancement is adding support for multiple clients, which would allow users to engage in group chats with several participants simultaneously. Another important development would be incorporating message encryption to ensure secure communication, protecting users' privacy. Additionally, the application could be expanded to support file sharing, enabling users to send and receive various types of media like images and documents.

To further enhance the user experience, implementing user authentication and profile management would allow users to log in, register, and customize their chats. Finally, adapting the application for mobile or web platforms would increase its accessibility and expand its reach, making it available to a broader audience. These future improvements would make the chat application more robust, versatile, and user-friendly.

APPENDIX A
(SOURCE CODE)

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;

public class Main extends Frame implements Runnable, ActionListener
   {TextArea textArea;
   TextField textField;
   Button send;
   //ServerSocket serverSocket;
   Socket socket;
   DataInputStream dataInputStream;
   DataOutputStream dataOutputStream;
   Thread chat;
   Main(){
      textField = new TextField();
      textArea = new TextArea();
      send = new Button("send");
      send.addActionListener(this);
      try {
         //serverSocket = new ServerSocket( 12000);
```

```
            socket = new Socket("localhost",12000);
            dataInputStream= new DataInputStream(socket.getInputStream());
            dataOutputStream = new
DataOutputStream(socket.getOutputStream());
        }
        catch (Exception e){

        }
        add(textField);
        add(textArea);
        add(send);
        chat = new Thread(this);
        chat.setDaemon(true);
        chat.start();

        setSize(500,500);
        setLayout(new FlowLayout());
        //setShape(short);
        setTitle("ser");
        setVisible(true);

    }
    public void actionPerformed(ActionEvent
        e){String msg = textField.getText();
        textArea.append("ser:"+msg+"\n");
        textField.setText("");
        try {
            dataOutputStream.writeUTF(msg);
            dataOutputStream.flush();
```

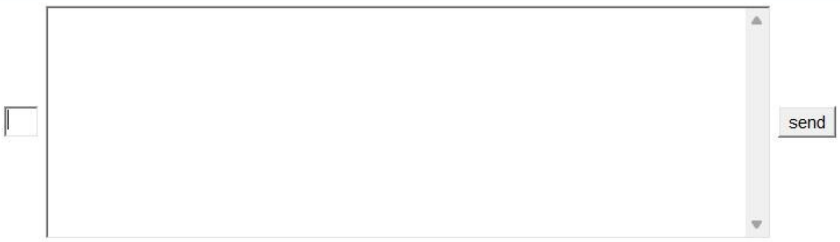
```
        } catch (IOException ex) {
            throw new RuntimeException(ex);
        }



    }

    public static void main(String[] args)
        {new Main();

    }
    public void
        run(){while(tru
        e){
            try{
                String msg = dataInputStream.readUTF();
                textArea.append("cli:"+msg+"\n");
            }
            catch(Exception e){

            }
        }
    }
}

import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.DataInputStream;
import java.io.DataOutputStream;
```

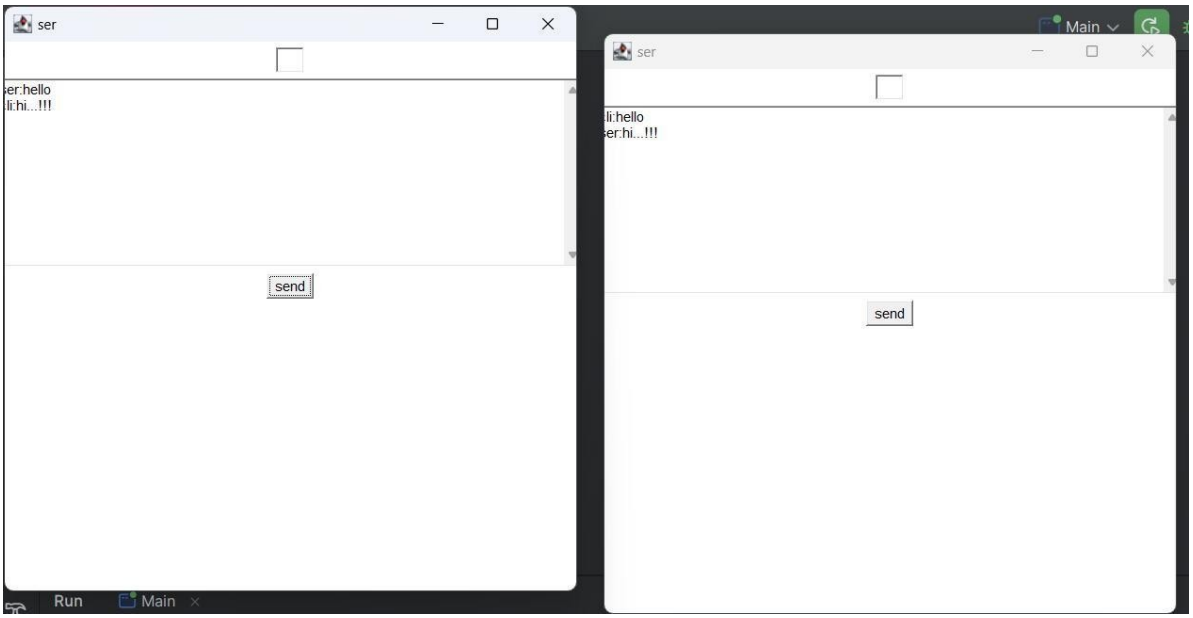
```
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;

public class Main extends Frame implements Runnable, ActionListener
   {TextArea textArea;
   TextField textField;
   Button send;
   ServerSocket serverSocket;
   Socket socket;
   DataInputStream dataInputStream;
   DataOutputStream dataOutputStream;
   Thread chat;
   Main(){
      textField = new TextField();
      textArea = new TextArea();
      send = new Button("send");
      send.addActionListener(this);
      try {
         serverSocket = new ServerSocket( 12000);
         socket = serverSocket.accept();
         dataInputStream= new DataInputStream(socket.getInputStream());
         dataOutputStream = new
DataOutputStream(socket.getOutputStream());
      }
      catch (Exception e){

      }
      add(textField);
```

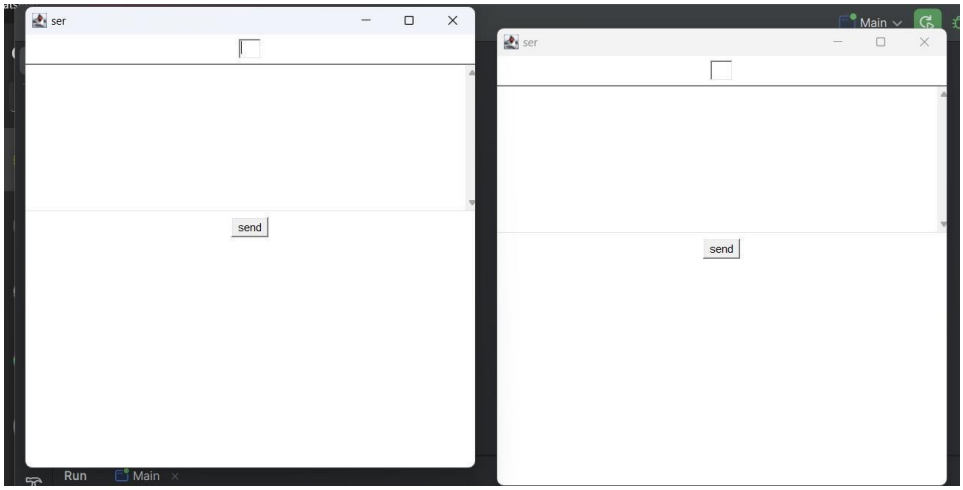
```
    add(textArea);
    add(send);
    chat = new Thread(this);
    chat.setDaemon(true);
    chat.start();

    setSize(500,500);
    setLayout(new FlowLayout());
    //setShape(short);
    setTitle("ser");
    setVisible(true);

}
public void actionPerformed(ActionEvent
    e){String msg = textField.getText();
    textArea.append("ser:"+msg+"\n");
    textField.setText("");
    try {
        dataOutputStream.writeUTF(msg);
        dataOutputStream.flush();
    } catch (IOException ex) {
        throw new RuntimeException(ex);
    }



}

public static void main(String[] args)
    {new Main();
```

```
    }
    public void
        run(){while(tru
        e){
            try{
                String msg = dataInputStream.readUTF();
                textArea.append("cli:"+msg+"\n");
            }
            catch(Exception e){

            }
        }
    }
}
```

APPENDIX B
(SCREENSHOTS)







REFERENCES

Books:

1. *Head First Java* by Kathy Sierra and Bert Bates – A beginner-friendly book covering core Java concepts, including networking and GUI development.
2. *Java: The Complete Reference* by Herbert Schildt – A comprehensive guide to Java, including networking, multithreading, and I/O operations.

Websites:

1. Oracle Java Documentation – Official Java documentation.
2. GeeksforGeeks - Java Tutorials – Tutorials on Java programming concepts.
3. Stack Overflow – A community Q&A site for Java-related problems.

YouTube:

1. Java Programming Tutorial – Various Java programming tutorials.
2. Telusko - Java Tutorials – A channel with clear Java tutorials.

Online Courses:

1. Udemy - Java Programming – A complete Java course from basics to advanced topics.
2. Coursera - Java Programming and Software Engineering Fundamentals – An online program on Java programming and software engineering.

These resources provide comprehensive learning materials on Java programming, networking, and GUI design.