CHAT APPLICATION

A PROJECT REPORT

Submitted by

SWATHI S (8115U23AM053)

in partial fulfillment of requirements for the award of the course

CGB1201 - JAVA PROGRAMMING

In

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING

(ARTIFICAL INTELLIGENECE AND MACHINE LEARNING)

K. RAMAKRISHNAN COLLEGE OF ENGINEERING

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

SAMAYAPURAM – 621 112

DECEMBER - 2024

K. RAMAKRISHNAN COLLEGE OF ENGINEERING (AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on **" CHAT APPLICATION"** is the bonafide work of **SWATHI S (8115U23AM053)** who carried out the project work during the academic year 2024 - 2025 under my supervision

SIGNATURE

MR.B.KIRAN BALA. B.Tech., M.E., M.B.A., Ph.D., M.I.S.T.E., U.A.C.E.E., IAENG

HEAD OF THE DEPARTMENT

Department Of Artificial Intelligence And Machine Learning,

K.Ramakrishnan College of Engineering (Autonomous)

Samayapuram–621112.

SIGNATURE

Mrs.P. GEETHA M.E.,

ASSISTANT PROFESSOR

Department of Artificial Intelligence

And Data science,

K.Ramakrishnan College of Engineering (Autonomous)

Samayapuram–621112.

Submitted for the end semester examination held on …………….

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

SWATHI S

Place: Samayapuram

Date:

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution "**K.Ramakrishnan College of Engineering (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. D. SRINIVASAN, B.E, M.E., Ph.D.,** Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. B. KIRAN BALA, B.Tech., M.E., M.B.A., Ph.D., M.I.S.T.E., U.A.C.E.E., IAENG** ,Head of the department, **Artificial Intelligence and Machine Learning** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **Mr.P.GEETHA M.E,** Department of **Artificial Intelligence and data Science,** 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.

INSTITUTE VISION AND MISSION

VISION OFTHE INSTITUTE:

To achieve aprominent position among the top technical institutions.

MISSION OF THE INSTIITUTE:

M1: To best owstandard technical education par excellence through state of the art infrastructure, competent faculty and high ethical standards.

M2: To nurture research and entrepreneurial skills among students in cutting edge technologies.

M3: To provide education for developing high-quality professionals to transform the society.

DEPARTMENT VISION AND MISSION

DEPARTMENT OF CSE(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

Vision of the Department

To become a renowned hub for Artificial Intelligence and Machine Learning

Technologies to produce highly talented globally recognizable technocrats to meet

Industrial needs and societal expectations.

Mission of the Department

M1: To impart advanced education in Artificial Intelligence and Machine Learning,

Built upon a foundation in Computer Science and Engineering.

M2: To foster Experiential learning equips students with engineering skills to

Tackle real-world problems.

M3: To promote collaborative innovation in Artificial Intelligence, machine

Learning, and related research and development with industries.

M4: To provide an enjoyable environment for pursuing excellence while upholding

Strong personal and professional values and ethics.

Programme Educational Objectives (PEOs):

Graduates will be able to:

PEO1: Excel in technical abilities to build intelligent systems in the fields of Artificial Intelligence and Machine Learning in order to find new opportunities.

PEO2: Embrace new technology to solve real-world problems, whether alone or

As a team, while prioritizing ethics and societal benefits.

PEO3: Accept lifelong learning to expand future opportunities in research and

Product development.

Programme Specific Outcomes (PSOs):

PSO1: Ability to create and use Artificial Intelligence and Machine Learning

Algorithms, including supervised and unsupervised learning, reinforcement

Learning, and deep learning models.

PSO2: Ability to collect, pre-process, and analyze large datasets, including data

Cleaning, feature engineering, and data visualization..

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 to 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 effectivelyon 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 preparationand ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

The **Chat Application** project demonstrates a simple client-server communication model using Java's networking API. It is designed to showcase the implementation of a multi-threaded server and a client capable of sending and receiving messages in real time. This system can serve as a foundational framework for more advanced chat systems, emphasizing core concepts such as socket programming, threading, and message handling.The server, implemented in the **ChatServer** class, listens for incoming connections on a specified port. It employs a multi-threaded architecture, enabling it to handle multiple clients concurrently. Each client is managed by a dedicated thread, which facilitates efficient and isolated communication. The server also echoes received messages back to the originating client, demonstrating basic message processing and response handling. This architecture ensures that the server remains responsive and scalable, even under multiple simultaneous connections.On the client side, the **ChatClient** class establishes a connection to the server and provides an interactive console-based interface for users. The client utilizes a dual-thread mechanism: one thread listens for messages from the server, while the main thread handles user input. This design ensures seamless, asynchronous communication, allowing the user to send messages without interruption from incoming data. User messages are sent to the server via sockets, and responses from the server are displayed in real time, simulating a basic chat experience.This project highlights key software engineering principles, including multi- threading, input/output stream management, and client-server interaction. It also introduces error handling for scenarios such as network failures or abrupt disconnections, ensuring robustness and reliability.

ABSTRACT WITH POs AND PSOs MAPPING

ABSTRACT	POs MAPPED	PSOs MAPPED
The **Chat Application** is a Java-based client-server system demonstrating core networking concepts like socket programming, multi-threading, and real-time communication. The **ChatServer** handles multiple clients concurrently using a thread-per-client model, efficiently managing isolated connections and echoing messages back. The **ChatClient** connects to the server with a console-based interface, supporting seamless bidirectional communication through a dual-thread design that handles both user input and incoming messages. This project highlights key practices such as input/output management and error handling, offering a robust foundation for extensions like GUIs, message broadcasting, and security features. It serves as a practical introduction to Java network programming and a scalable base for more advanced systems.	PO 1 PO 2 PO 3 PO 4	PSO 1 PSO 2 PSO 1 PSO 3

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

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1. Objective

The objective of this project is to develop a Java-based chat application that facilitates real-time communication between multiple clients through a centralized server. This project aims to demonstrate the core principles of socket programming, which enable data exchange over network connections, and multi- threading, which allows the server to handle multiple clients simultaneously. By implementing input and output stream management, the application ensures seamless bidirectional communication between users.Additionally, the project focuses on showcasing practical error handling, ensuring the system remains robust and functional even in cases of client disconnections or network interruptions.

2. Overview

The Chat Application is a Java-based client-server system designed to enable real- time communication between multiple users. Built using Java's socket programming and multithreading, the application facilitates seamless messaging through a central server that manages connections and message exchange.The server employs a multi-threaded design to handle multiple client connections simultaneously, while the client uses a dual-threaded approach for concurrent message sending and receiving. This project demonstrates essential concepts such as input/output stream management, error handling, and robust communication. Although console-based, it provides a strong foundation for ng

3. Java Programming Concepts

- **Socket Programming**:

  The project utilizes Java's Socket and ServerSocket classes for establishing client-server communication over a network. The server listens for incoming connections on a specified port, while clients use sockets to connect to the server, send, and receive data.

- **Multi-threading**:

  To handle multiple client connections concurrently, the server creates a new thread for each client. This allows the server to process each client's requests independently, ensuring efficient communication without blocking other connections. The client also uses a separate thread to listen for incoming messages while allowing the user to send messages.

- **Input/Output (I/O) Streams**:

  Java's InputStream and OutputStream (specifically BufferedReader and PrintWriter) are used for reading and writing messages between the client and server. These streams handle text-based communication, ensuring data is transmitted smoothly between both ends.

- **Exception Handling**:

  The project uses Java's exception handling mechanisms (try-catch) to manage errors, such as network failures, disconnections, or invalid input. This ensures that the application remains stable even when issues arise, such as when a client unexpectedly disconnects.

- **Object-Oriented Programming (OOP)**:

  The project follows the principles of object-oriented programming, encapsulating related functions into classes (ChatServer and ChatClient). Methods such as run() for handling client communication and constructors for initializing sockets follow OOP principles like modularity and reusability

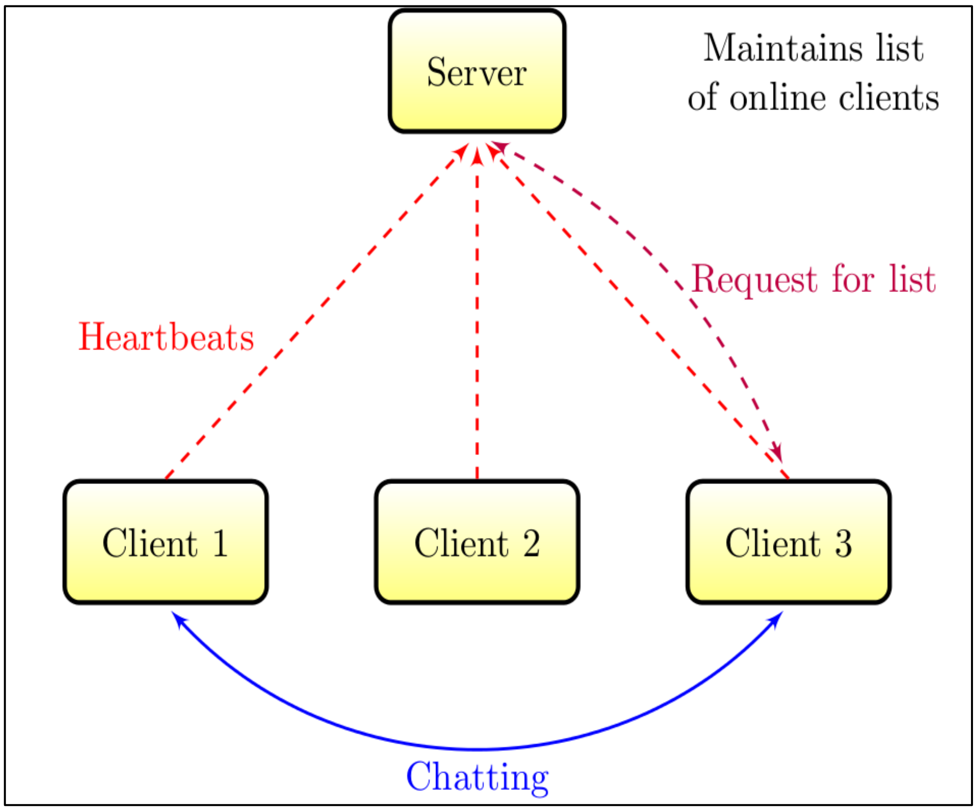
CHAPTER 2
PROJECT METHODOLOGY

1. Proposed Work

The proposed work for this project is to develop a multi-user chat application using Java, leveraging socket programming and multi-threading to enable real-time communication. The server will handle multiple client connections concurrently by assigning each client a dedicated thread, ensuring efficient and uninterrupted communication. Clients will be able to send messages to the server, receive responses, and display messages in real time.

The project will focus on error handling, ensuring the system remains stable during network interruptions or client disconnections. The core components will utilize Java's Socket, ServerSocket, BufferedReader, and PrintWriter for communication. Future enhancements could include adding graphical user interfaces (GUIs), message broadcasting, user authentication, and encryption for secure communication.

2.2 Block Diagram



CHAPTER 3
MODULE DESCRIPTION

3.1 Server Setup Module

This module handles the initialization and configuration of the server. It listens for incoming client connections using Java's ServerSocket class and assigns each new client to a separate thread. The server manages multiple client connections simultaneously, ensuring smooth communication without blocking other clients.

3.2 Client Communication Module

The client module is responsible for establishing a connection with the server via a Socket. It provides a console-based interface where the user can input messages. The module also listens for messages from the server and displays them in real time.

3.3 Multi-threading Module

This module implements the multi-threaded design of the server and client. The server creates a new thread for each client connection, allowing the server to handle multiple clients concurrently without delay. On the client side, a separate thread handles incoming messages while the user can continue typing and sending messages.

3.4 Error Handling Module

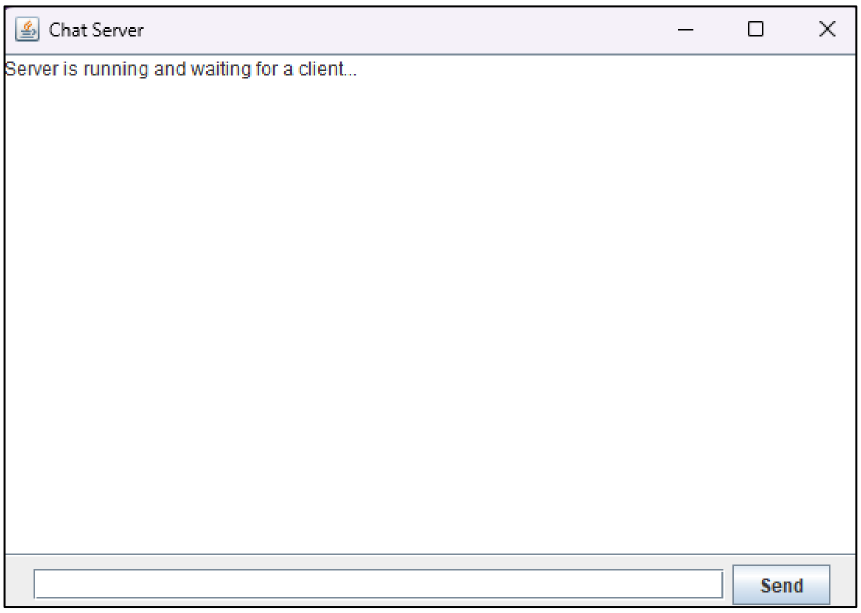
This module deals with exceptions and errors that may occur during communication. It handles issues such as client disconnections, network failures, or incorrect inputs. Proper exception handling ensures the system continues to function smoothly, even when unexpected issues arise.

3.5 GUI Module

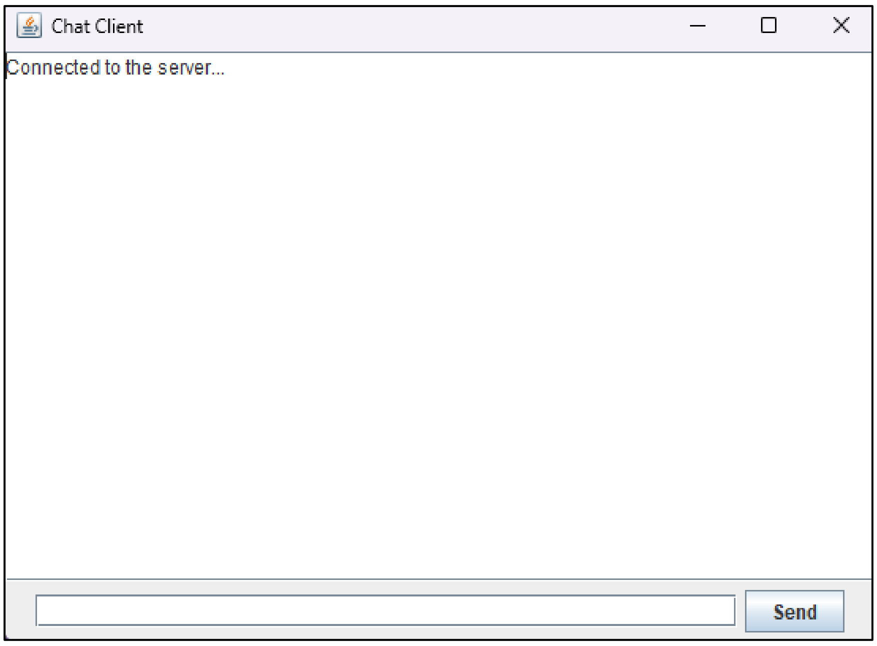
The current chat application uses a console-based interface, where users input messages through the terminal, and messages are displayed in the same console. It allows real-time communication with the server, where incoming messages are continuously displayed while the user can send new messages. This simple text-based approach focuses on core functionalities, such as message exchange and multi-threading, without the complexity of a graphical interface.

CHAPTER 4

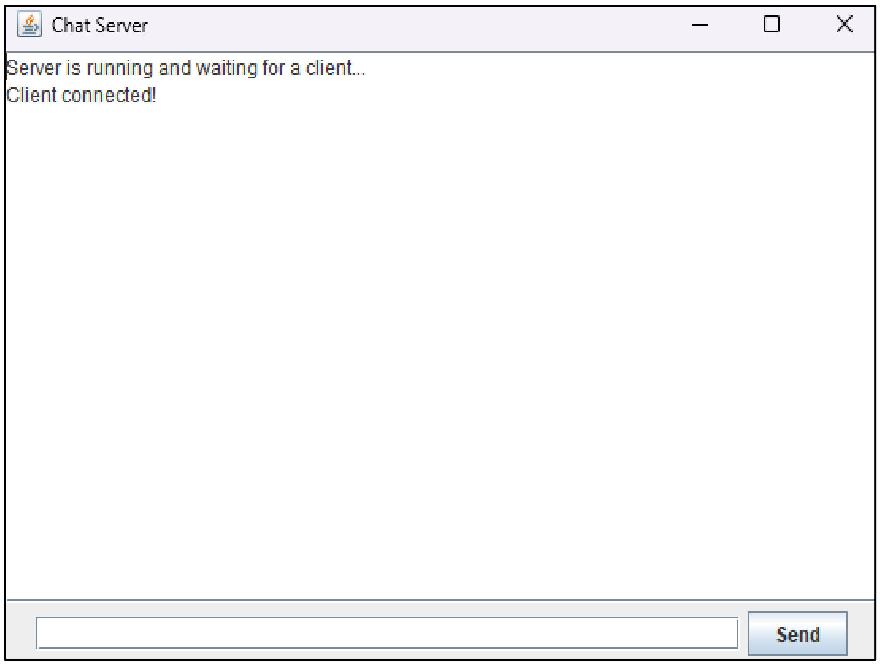
RESULTS AND DISCUSSION



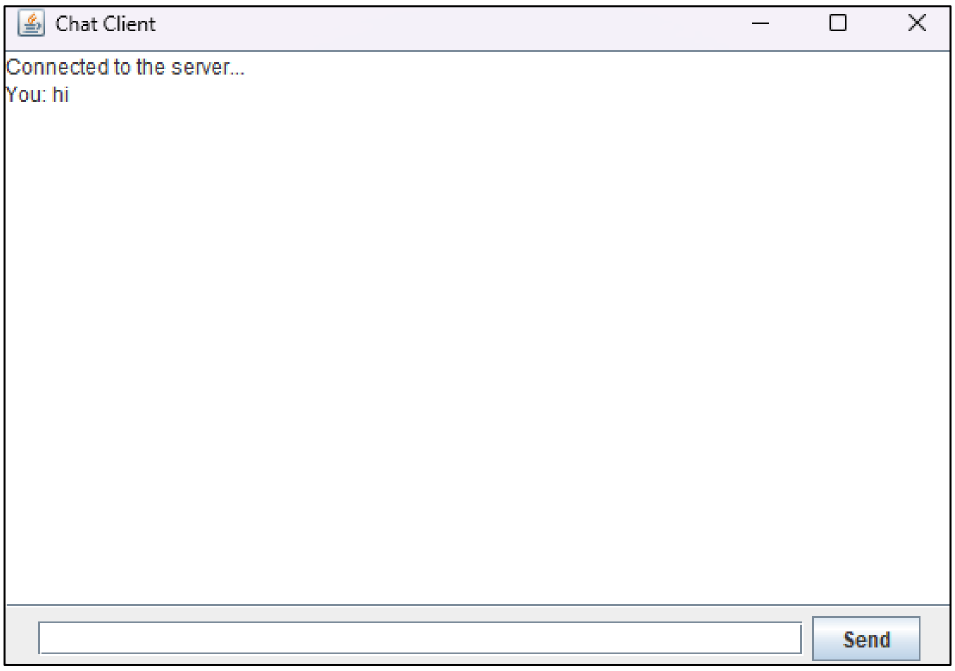
Server is running and waiting for a client...



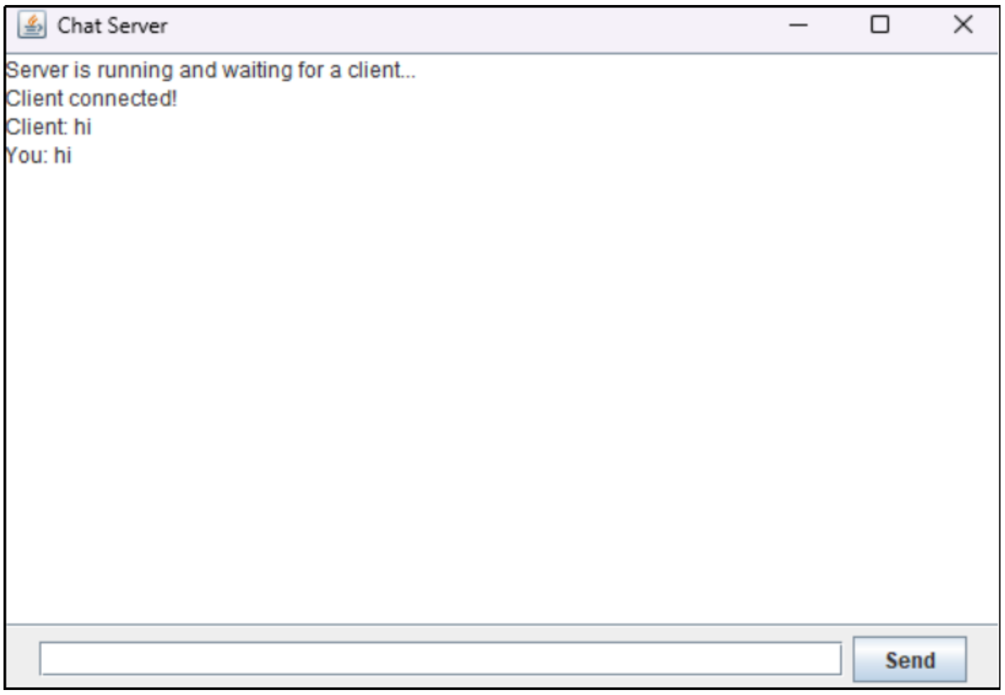
Connected to the server...



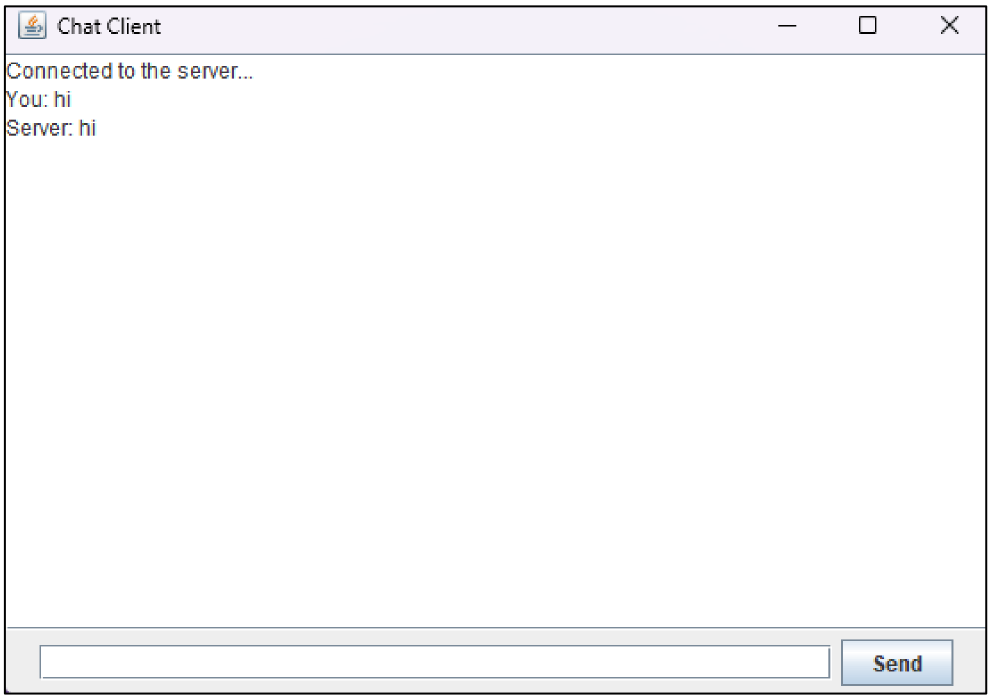
```
Server is running and waiting for a client...
Client connected!
```



```
Connected to the server...
You: hi
```



```
Server is running and waiting for a client...
Client connected!
Client: hi
You: hi
```



```
Connected to the server...
You: hi
Server: hi
```

DESCRIPTION:

User Interface (UI):

- A graphical interface designed using **Swing** or **JavaFX**, where users can interact with the application.

UI components like:

- ➢ **Text field** for entering messages.
- ➢ **Text area** or chat window displaying the conversation history.
- ➢ **Send button** to send messages.
- ➢ Optional: A **list of connected users** in a sidebar.

Message Exchange:

- Messages sent by one user are displayed in the chat window of both the sender and the receiver.
- The sender sees their message aligned (e.g., to the right), while the receiver sees it aligned (e.g., to the left).

Client-Server Communication:

- **Server Output**:
- ➢ Logs of connection establishment with clients.
- ➢ Display of incoming and outgoing messages between clients.
- **Client Output**:
- ➢ Connection status to the server (e.g., "Connected to Server").
- ➢ Received messages from other users in real-time.

Multi-user Support (if implemented):

- Multiple users can connect to the server, send messages, and participate in a group hat.
- Messages are broadcasted to all connected clients.

Error Handling:

- Appropriate messages for connectivity issues (e.g., "Server not reachable").
- Input validation (e.g., prevent sending blank messages).

Optional Enhancements:

- Message timestamps.
- Notification sound or popup for new messages.
- Typing indicator (e.g., "User is typing...").

CHAPTER 5
CONCLUSION

In conclusion, the Java-based Chat Application successfully demonstrates the principles of client-server communication, socket programming, and multi-threading, offering a real-time messaging platform for multiple users. Through the use of Java's Socket, ServerSocket, BufferedReader, and PrintWriter classes, the project showcases how data can be transmitted over a network using TCP/IP protocol, providing a foundational understanding of networking in Java.The multi-threaded server design ensures that multiple clients can be handled concurrently without any communication delays, enhancing the performance and scalability of the system. The application handles basic functionalities like sending and receiving messages, ensuring smooth interaction between users through the server. Additionally, error handling mechanisms are implemented to manage disconnections, network failures, and invalid inputs, ensuring that the system remains robust under various conditions.

APPENDIX
(Coding)

ChatServerUi.java import java.io.*;

```
import java.net.*;
import javax.swing.*;
 import java.awt.*;
import java.awt.event.*;

public class ChatServerUI {
private static JTextArea chatArea;
private static JTextField inputField;
private static PrintWriter output;


public static void main(String[] args) {
JFrame frame = new JFrame("Chat Server");
chatArea = new JTextArea(20, 50);
inputField = ew JTextField(40);


JButton sendButton = new JButton("Send");
JPanel panel = new JPanel();
panel.add(inputField); panel.add(sendButton);


chatArea.setEditable(false);
frame.add(new JScrollPane(chatArea), BorderLayout.CENTER);
```

```
}



frame.add(panel, BorderLayout.SOUTH);
frame.pack();
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setVisible(true);

try (ServerSocket serverSocket = new ServerSocket(12345)) {
chatArea.append("Server is running and waiting for a client...\n");
Socket clientSocket = serverSocket.accept();
 chatArea.append("Client connected!\n");


BufferedReader input = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
output = new PrintWriter(clientSocket.getOutputStream(), true);

sendButton.addActionListener(e -> sendMessage());
inputField.addActionListener(e -> sendMessage());

String clientMessage;
while ((clientMessage = input.readLine()) != null) {
chatArea.append("Client: " + clientMessage + "\n");
 if (clientMessage.equalsIgnoreCase("bye")) {
chatArea.append("Client disconnected.\n");
 break;
}

}
```

```
 } catch (IOException e) {
chatArea.append("Server error: " + e.getMessage() + "\n");
}
}
private static void sendMessage() { String
message = inputField.getText();
 if (!message.isEmpty()) {
chatArea.append("You: " + message + "\n");
output.println(message); inputField.setText("");
if (message.equalsIgnoreCase("bye")) {
chatArea.append("Ending chat...\n");
System.exit(0);
}
}
}
}
```

ChatClientUI.java

```
import java.io.*; import
java.net.*; import
javax.swing.*; import
java.awt.*;
import java.awt.event.*;
```

```
public class ChatClientUI {
private static JTextArea chatArea;
 private static JTextField inputField;
 private static PrintWriter output;

 public static void main(String[] args) {
JFrame frame = new JFrame("Chat Client");
chatArea = new JTextArea(20, 50);
 inputField = new JTextField(40);

JButton sendButton = new JButton("Send");
JPanel panel = new JPanel();
 panel.add(inputField);
panel.add(sendButton);
chatArea.setEditable(false);

frame.add(new JScrollPane(chatArea), BorderLayout.CENTER);
 frame.add(panel, BorderLayout.SOUTH);
frame.pack();
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setVisible(true);

try (Socket socket = new Socket("localhost", 12345)) {
chatArea.append("Connected to the server...\n");

 BufferedReader input = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
output = new PrintWriter(socket.getOutputStream(), true);
}



sendButton.addActionListener(e -> sendMessage());
inputField.addActionListener(e -> sendMessage());

String serverMessage;
while ((serverMessage = input.readLine()) != null) {
chatArea.append("Server: " + serverMessage + "\n");
 if (serverMessage.equalsIgnoreCase("bye")) {
chatArea.append("Server disconnected.\n");
 break;
}
}
} catch (IOException e) {
chatArea.append("Client error: " + e.getMessage() + "\n");
}
}
private static void sendMessage() {
String message = inputField.getText();
 if (!message.isEmpty()) {
chatArea.append("You: " + message + "\n");
output.println(message); inputField.setText("");
if (message.equalsIgnoreCase("bye")) {
chatArea.append("Ending chat...\n");
System.exit(0);
}
}
}
}
```

REFERENCES:

1. **Books**:

1. *Java: The Complete Reference* by Herbert Schildt
2. *Effective Java* by Joshua Bloch
3. *Java Programming for Beginners* by Nathan Clark

2. **Websites**:

1. Oracle Documentation for Java: https://docs.oracle.com/javase/
2. Java Tutorials on W3Schools: https://www.w3schools.com/java/
3. Stack Overflow: https://stackoverflow.com/

3. **Research Papers and Articles**:

1. A Guide to Java Exception Handling: https://www.geeksforgeeks.org/exception-handling-in-java/
2. Trigonometric Functions in Java: https://www.journaldev.com/1635/java- math-functions

4. **Java Libraries and Tools**:

1. **Java Math Library**: https://docs.oracle.com/javase/7/docs/api/java/lang/Math.html
2. **JavaFX Documentation**: https://openjfx.io/