Client-Server Chat Application Report

Computer Networks

Semester 2nd

Submitted to: **Mam Poma Panezai**

Group Members:

- Muzammil Mehdi – 67459
- Murtaza Hassan – 67712
- Ali Gohar – 68757
- Irfan Ali – 66529

Submission Date: **07-06-2025**

BALUCHISTAN UNIVERSITY OF IT, ENGINEERING & MANAGEMENT SCIENCES, QUETTA

Table of Contents

1. Introduction

The project is a Java based Client-Server Chat Application that demonstrates fundamental concepts of networking communication using client-server architecture. The application uses JavaFX to provide a graphical user interface (GUI) for both the server and clients, allowing for real-time messaging.

Key aspects of the project include:

- A Main Launcher which helps in running the server and clients easily.
- Multiple clients can connect to the server simultaneously.
- Each client must register with a unique username.
- Clients can send messages to the server, which server can reply privately or broadcast messages to all connected clients.
- The server manages client connections, message routing, and ensures data integrity over the network.
- The application showcases multithreading for handling concurrent client connections and asynchronous communication.

The project serves as a practical implementation for learning java networking, concurrency, and GUI development.

2. Objectives

- To implement a basic LAN-based chat system using Java.
- To understand and apply client-server communication principles.
- To design a user-friendly GUI using JavaFX.
- To ensure real-time message exchange between users.

3. Tools & Technologies Used

- **Programming Language**: Java
- **GUI Framework**: JavaFX
- **Network communication**: Java Sockets
- **Concurrency**: Java Multithreading

- **IDE**: NetBeans or any other
- **Build & Run**: Standard Java Development Kit (JDK)
- **Version Control**: Git & GitHub

4. System Requirements/Prerequisites

- Operating System: Windows/Linux/Mac
- Java Development Kit (JDK) 8 or higher
- JavaFX latest version
- NetBeans IDE, or any other
- Git to clone repository

5. System Architecture

The application uses a client-server model. The server listens for incoming client connections. Each connected client can send messages to the server, which handles distribution (individual or broadcast). JavaFX is used to design interactive user interfaces for both the server and clients.

7. Working Flow

Starting the Main Launcher

After launching the Main Launcher, you can select what to run — either the **Server** or the **Client**.

- You can choose either Server or Client first (the order doesn't matter).
- However, note that the Client will not be able to communicate until the Server is started.
- You can launch multiple clients by clicking the **"Run Client"** button multiple times.

Starting the Server

- Launch the server application by running the `ServerGUI` class.

- The server window will display a message indicating it is waiting for clients to connect.
- The server supports multiple clients connecting simultaneously.

Connecting Clients

- Run the `ClientGUI` application on one or more machines.
- When the client window opens, you will be prompted to enter a unique username.
- After entering a valid username, the client connects to the server and can start sending messages.

Sending Messages

From Client to Server:

- Type your message in the input box.
- Press the Enter key or click the Send button to send the message.
- The message is sent exclusively to the server, not to other clients.

From Server to Client(s):

- The server can send messages to:
  - A specific client by entering the client's username.
  - All connected clients by selecting the broadcast option.
- Type the message in the server's input box.
- Choose the recipient (either single client or broadcast).
- Click Send to deliver the message.

Managing Clients

- Each client has a unique username for identification.
- The server maintains a list of connected clients.

- The server GUI displays client connection and disconnection events.
- If a client disconnects, the server updates its client list accordingly.

Error Handling

- If the client fails to connect to the server, an error message appears in the client's message area.
- If the server unexpectedly disconnects, clients will receive a disconnection notice.
- The server GUI shows error messages related to network or I/O issues.

11. Conclusion

This project showcases a working LAN-based chat system built with Java, combining core concepts like socket communication, multithreading, and GUI development. It allows multiple clients to connect and exchange messages through a central server in real-time, all within a simple and intuitive JavaFX interface.

Throughout this project, we learned how to manage multiple users, handle real-time messaging, and deal with typical networking challenges. It gave us hands-on experience in building a functional client-server model and understanding how network applications work under the hood.

12. Links

GitHub Repository: https://github.com/Meta-Captain819/JavaChatApp

YouTube link: https://youtu.be/pE9P3BoJF3k?si=FN5i0nYEe_tM_wsj