# Chat Application Design Document

Table of Contents

## 1. <u>Introduction</u>

### 1.1 Purpose

The purpose of this chat application is to provide users with a reliable platform for real-time communication. Users can log in and engage in conversations with other users through text messages.

### 1.2 Document Conventions

- <u>User</u>: Refers to an individual using the chat application.

- <u>Server</u>: Refers to the central server managing user connections and message broadcasting.

- <u>Client</u>: Refers to the application instance running on a user's device.

### 1.3 References

Java Network: [Java Socket Programming].

Search on Google.

YouTube Videos.

## 2. <u>Overview</u>

### 2.1 Key Features

- User Login.

- User Authentication.

- Message Broadcasting.

- Saving Client’s messages.

- Statistics of Client’s Message.

### 2.2 High-Level Architecture

The application follows a client-server architecture. Clients connect to the central server for user authentication and message broadcasting.

### 2.3 Technology

- <u>Backend:</u> Java for server-side logic.

- <u>Networking:</u> Java Socket for communication.

## 3. <u>User Authentication</u>

### 3.1 User Login

To log in, users enter their username and password. The server verifies the credentials by comparing the entered username and password with the stored username and password.

## 4. <u>Server-Side Implementation</u>

### 4.1 Server Initialization

The server initializes a Socket to listen for incoming connections on a specified port.

### 4.2 Load User Database

The server loads the stored usernames and passwords the verify the user login.

### 4.3 Handling Client Connections

Upon a client connection request, the server creates a new thread to handle communication with that client.

### 4.4 User Authentication

User authentication involves verifying the entered credentials against the stored usernames and passwords.

### 4.5 Message Broadcasting

The server broadcasts messages received from one client to all connected clients, excluding the sender.

### 4.6 Threading Model

A multi-threaded approach is used to handle multiple client connections concurrently, ensuring responsiveness.

## 5. <u>Client-Side Implementation</u>

### 5.1 Connecting to the Server

Clients connect to the server using a Socket. The server's IP address and port number are specified during connection.

### 5.2 User Authentication

Clients provide their username and password during login.

### 5.3 Sending and Receiving Messages

Clients can send messages to the server, which then broadcasts them to all connected clients. Clients receive and display messages in real-time.

### 5.4 Saving Client Messages

Before the client leaves the group chat. All his current messages are being stored.

### 5.5 Client Messages Statistics

Before the client leaves the group, simple statistics are made to all his messages containing all the words used by the client and the number of its occurrences.

## 6. Message Broadcasting

### 6.1 Broadcasting Logic

Messages sent by clients are broadcasted to all connected clients by the server.

### 6.2 Excluding the Sender

The server ensures that the sender of a message is excluded from receiving their own messages.

## 7. User Guide

### 7.1 Client Login

After running, the user will be asked to enter his username and password to enter the chat room.

### 7.2 Sending and Receiving Messages

#### 7.2.1 Sending Messages

1. Once logged in, you can type and send messages by entering text and pressing Enter.
2. Your message will be broadcasted to all other connected users, and you will see their messages in real-time.

#### 7.2.2 Receiving Messages

1. Messages sent by other users will appear in your client's console.
2. The sender's username and the message will be displayed.

### 7.3 Disconnecting from the Chat

1. To exit the chat, type "exit" and press Enter.

2. Your client will disconnect from the server, and a message will be broadcasted to notify other users that you have left.