

**Documentation for Server-Client Chat Application**

**Name: M.MUSADIQ**



July 14, 2024

**Table of Contents**

[**Overview** 2](#_Toc171877880)

[**Components** 3](#_Toc171877881)

[**1.** **Server** 3](#_Toc171877882)

[**Main Functionalities:** 3](#_Toc171877883)

[**Classes and Methods:** 3](#_Toc171877884)

[ **Main Method:** 3](#_Toc171877885)

[ **Constructor:** 3](#_Toc171877886)

[ **ClientAccept:** 3](#_Toc171877887)

[ **MsgRead:** 3](#_Toc171877888)

[ **PrepareClientList:** 3](#_Toc171877889)

[ **initialize():** 3](#_Toc171877890)

[**2.** **LoginPage** 3](#_Toc171877891)

[**Main Functionalities:** 4](#_Toc171877892)

[**Methods:** 4](#_Toc171877893)

[**3.** **Client** 4](#_Toc171877894)

[**Main Functionalities:** 4](#_Toc171877895)

[**Classes and Methods**: 4](#_Toc171877896)

[**Usage** 5](#_Toc171877897)

[ **Server Initialization:** 5](#_Toc171877898)

[ **Client Connection:** 5](#_Toc171877899)

[ **Chatting:** 5](#_Toc171877900)

[ **Disconnection:** 5](#_Toc171877901)

[**Instructions:** 5](#_Toc171877902)

[ **Server Initialization:** 5](#_Toc171877903)

[ **Client Login:** 5](#_Toc171877904)

[ **Username Entry:** 5](#_Toc171877905)

[ **Chatting:** 5](#_Toc171877906)

[ **Disconnection:** 5](#_Toc171877907)

[**Important Notes:** 5](#_Toc171877908)

[ **Unique Usernames:** 5](#_Toc171877909)

[ **Message Delivery:** 5](#_Toc171877910)

[ **User Disconnection:** 5](#_Toc171877911)

# **Overview**

This Java-based Server-Client Chat Application facilitates real-time communication between multiple clients. The server manages connections, maintains active user lists, and ensures message delivery to designated recipients. The clients can either broadcast messages to all active users or multicast messages to selected users.

# **Components**

1. **Server**

The Server class is the backbone of the application, handling client connections, maintaining user lists, and relaying messages.

### **Main Functionalities:**

* **Startup:** Initializes the server and starts listening for client connections on a specified port.
* **Client Management:** Manages active and all-time user lists, ensuring unique usernames.
* **Message Handling:** Handles broadcast and multicast messages, ensuring proper delivery and managing user disconnections.

## **Classes and Methods:**

* **Main Method:** Initializes the server UI and starts the server.
* **Constructor:** Sets up the server socket, message board, and starts the client acceptance thread.
* **ClientAccept:** A thread that continuously accepts client connections, checks for username uniqueness, and initiates message reading threads for each client.
* **MsgRead:** A thread responsible for reading messages from clients and taking appropriate actions based on the message type (multicast, broadcast, or exit).
* **PrepareClientList:** Updates the list of active users and notifies all clients of the current active user list.
* **initialize():** Initializes the UI components of the server application.

1. **LoginPage**

The LoginPage class provides the user interface for clients to enter their username and connect to the server.

### **Main Functionalities:**

* **Startup:** Displays the login page UI for clients.
* **Username Entry:** Allows clients to enter a username and attempts to connect to the server.
* **Validation:** Checks for username uniqueness and prompts the user to enter a new username if the chosen one is already taken.

### **Methods:**

* **Main Method:** Displays the login page UI.
* **Constructor:** Initializes the UI components.
* **initialize():** Sets up the login page UI components.
* **ActionListener for Login Button:** Attempts to connect to the server with the entered username and handles server responses.

1. **Client**

The Client class provides the main chat interface for clients, allowing them to send and receive messages.

### **Main Functionalities:**

* **Chat Interface:** Displays the chat interface where clients can send messages and view the chat history.
* **Message Sending:** Allows clients to send multicast or broadcast messages.
* **Message Receiving:** Continuously listens for incoming messages from the server and updates the chat interface.
* **User Disconnection:** Allows clients to disconnect from the server and notifies other users of the disconnection.

### **Classes and Methods**:

* **Constructor:** Sets up the client UI, initializes the input and output streams, and starts the message reading thread.
* **Read:** A thread that listens for incoming messages from the server and updates the chat interface.
* **initialize():** Initializes the UI components of the client application.
* **ActionListener for Send Button:** Sends the typed message to the server, either as a multicast or broadcast.
* **ActionListener for End Chat Button:** Disconnects the client from the server and closes the chat interface.

# **Usage**

* **Server Initialization:** Start the server application to begin listening for client connections.
* **Client Connection:** Open the client login page, enter a unique username, and connect to the server.
* **Chatting:** Once connected, use the chat interface to send messages to selected users (multicast) or all users (broadcast). Note that messages cannot be sent to users who have disconnected.
* **Disconnection:** Use the "End Chat" button to disconnect from the server. The server and other clients will be notified of the disconnection.

# **Instructions:**

* **Server Initialization:** Run the Server application first to start listening for client connections.
* **Client Login:** After the server is running, open the LoginPage application on each client machine.
* **Username Entry:** Enter a unique username on the login page and attempt to connect to the server.
* **Chatting:** Once connected, use the client interface to send messages to selected users (multicast) or all users (broadcast). Note that messages cannot be sent to users who have disconnected.
* **Disconnection:** Use the "End Chat" button to disconnect from the server. The server and other clients will be notified of the disconnection.

# **Important Notes:**

* **Unique Usernames:** Each client must use a unique username to connect to the server. If a username is already taken, the client will be prompted to choose a different one.
* **Message Delivery:** Messages can only be sent to active users. Disconnected users will not receive any messages, and the sender will be notified of any delivery issues.
* **User Disconnection:** When a user disconnects, the server and all active clients will be updated to reflect the change in the active user list.