

# 1. Introduction

The **Real-Time Multi-User Chat Application** is a full-stack project developed using Spring Boot, WebSockets (STOMP & SockJS), and client-side JavaScript, HTML, and CSS. The application allows users to:

- **Create chat rooms** (both public and private—with password protection for private rooms).
- **Join existing rooms** and engage in real-time conversations.
- **View a live list of active rooms**, enabling users to join ongoing discussions.

This project demonstrates real-time communication, dynamic room management, and a responsive user interface, making it an excellent example of modern web application development for software developer intern positions.

## 2. System Architecture

### Overview

- **Back-End:**
  - Built with **Spring Boot**.
  - Utilizes **WebSocket** technology with **STOMP** and **SockJS** for real-time messaging.
  - Rooms and messages are managed through dedicated controllers (RoomController and ChatController).
  - An in-memory data structure (ConcurrentHashMap) stores room details for rapid prototyping, with potential for future persistence.
- **Front-End:**
  - Developed with HTML, CSS, and JavaScript.
  - Provides separate views for room creation, user login, and chat.
  - Establishes a WebSocket connection to exchange messages and room data in real time.

### 3. Technical Details & Code Overview

#### Key Components

- **WebSocketConfig:**  
Configures the WebSocket endpoint (/ws), sets the application destination prefix (/app), and enables a simple broker on /topic.
- **ChatController:**
  - **/chat.sendMessage:** Receives chat messages and broadcasts them to /topic/public.
  - **/chat.addUser:** Handles join events by storing the username (and room, if applicable) in session attributes and broadcasting a join notification.
- **RoomController:**
  - **/room.create:** Allows users to create a room (public or private). The room is stored in an in-memory ConcurrentHashMap.
  - **/room.list:** Returns all currently created rooms to clients when requested.
- **WebSocketEventListener:**  
Listens for disconnect events and broadcasts a leave message when a user disconnects.
- **Models:**
  - **ChatMessage:** Contains fields such as type, sender, content, and an added room field for room-based messaging.
  - **Room:** Contains room details including name, password, and room type (an enum with values Public and Private).

#### Algorithms and Workflow

- **Room Creation Workflow:**
  1. **User Action:** A user (User1) fills out the room creation form selecting a room type (Public/Private) and enters a room name (and a password if Private).
  2. **Client-Side:** The JavaScript builds a room object (including a room property) and sends it to the endpoint /app/room.create.
  3. **Server-Side:** The RoomController validates the room, stores it in an in-memory map, and returns the room object on /topic/rooms.

4. **Client-Side UI Update:** The response triggers a subscription callback that updates the room list and transitions the UI (e.g., shows the username entry view).

- **Chat Messaging Workflow:**

1. **User Join:** When a user joins (via `/chat.addUser`), their username and the current room are saved in the session.
2. **Message Sending:** Chat messages include the current room property. The client sends messages to `/app/chat.sendMessage`.
3. **Message Broadcasting:** The server broadcasts the message to `/topic/public`. On the client side, messages are filtered by comparing the message's room with the current room, ensuring that only messages for the active room are displayed.

- **Room Listing and Joining:**

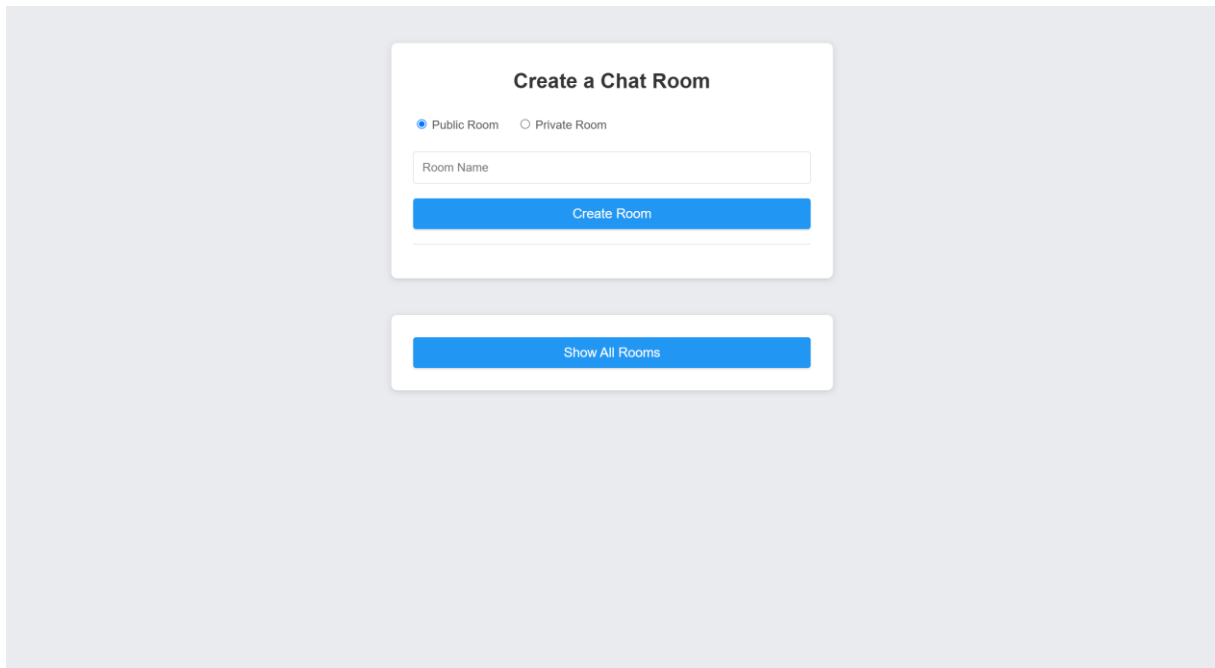
1. **Room Listing:** When a new user (User2) clicks "Show All Rooms", a message is sent to `/app/room.list` to retrieve all active rooms.
2. **UI Display:** The room list is displayed with Join buttons.
3. **Joining a Room:** When User2 clicks "Join Room", if the room is private, a password prompt appears. Upon successful password entry (and optionally verified by the server), User2 joins the room, and the client updates the `currentRoom` variable accordingly.

## 4. User Instructions

### For New Users (e.g., User1)

#### 1. Access the Application:

Navigate to <http://localhost:8080/>.

A screenshot of a web application interface for creating a chat room. The background is a light gray. In the center, there is a white rectangular box with a thin gray border. At the top of this box, the text "Create a Chat Room" is displayed in a bold, black font. Below this text, there are two radio button options: "Public Room" (which is selected, indicated by a blue dot) and "Private Room" (which is unselected, indicated by a gray dot). Below the radio buttons is a text input field with the placeholder text "Room Name". At the bottom of the white box is a blue rectangular button with the text "Create Room" in white. Below the white box, there is another white rectangular box, also with a thin gray border, containing a single blue rectangular button with the text "Show All Rooms" in white.

#### 2. Create a Room:

- On the room creation page, select a room type (Public or Private).
- Enter a room name (if Private, also enter a room password).
- Click "**Create Room**".
- The room is created and stored on the server, and you will be automatically taken to the username entry page.

Show All Rooms

room1 (Public)

Join Room

Join Chat Room

user1

Start Chatting

### 3. Join the Chat:

- Enter your username in the provided field and click **"Start Chatting"**.
- You will be taken to the chat page where you can send and receive messages in real time.

### For Returning Users (e.g., User2)

#### 1. Access the Application:

Navigate to <http://localhost:8080/>.

Create a Chat Room

☒ Public Room ☐ Private Room

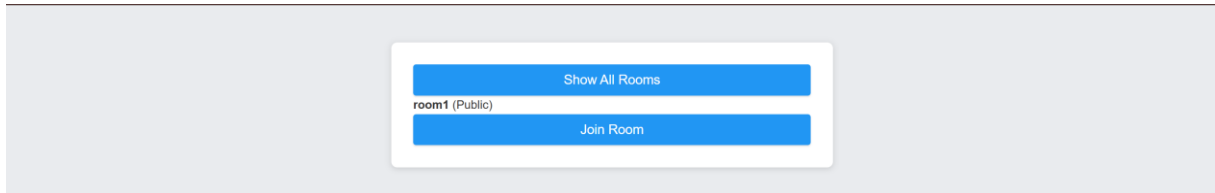
Room Name

Create Room

Show All Rooms

## 2. View Existing Rooms:

- On the room creation page, click the "**Show All Rooms**" button.
- The list of active rooms (including those created by other users) will be displayed.



## 3. Join an Existing Room:

- Click the "**Join Room**" button next to the room you wish to join.
- If the room is **Private**, you will be prompted to enter the room password.
- After entering your username on the subsequent page, you will join the chat for that room and see all messages specific to it.

