Real-Time Chat Application

1. Executive Summary

The Real-Time Chat Application is a fully functional, real-time messaging platform designed to enable instant communication between users. Built using Spring Boot for the backend and a responsive frontend with HTML, CSS, and JavaScript, this application leverages WebSockets for a seamless and responsive user experience. This report provides an in-depth look at the application's features, architecture, and implementation details, as well as setup instructions for local deployment.

2. Project Objectives

The primary objectives of the Real-Time Chat Application are as follows:

- Facilitate real-time communication between users.
- Provide a clean and intuitive interface that aligns messages by the sender (user) and receiver.
- Ensure scalability and maintainability by leveraging Spring Boot and WebSockets.

3. Features

- **Real-Time Messaging**: Users can communicate instantly with each other, with messages displayed as soon as they are sent.
- **User-Specific Message Alignment**: Messages sent by the user appear right-aligned, while incoming messages appear left-aligned, enhancing the chat experience.

4. Technology Stack

- **Backend**: Java, Spring Boot framework, WebSocket protocol for real-time message exchange.
- **Frontend**: HTML, CSS, JavaScript for a responsive and user-friendly chat interface.
- **Build Tool**: Maven for project management and dependency resolution.

5. System Architecture

The application follows a client-server architecture, with a Spring Boot backend serving as the WebSocket server and a frontend interface built using HTML, CSS, and JavaScript.

Backend (Server Side)

- **Spring Boot**: Acts as the core of the backend, providing a robust framework for handling WebSocket connections and messaging.
- **WebSocket Integration**: Allows two-way, real-time communication between the server and clients, facilitating instant message delivery.

Frontend (Client Side)

- **HTML & CSS**: Provides the structure and styling for a clean, responsive chat interface.
- **JavaScript**: Manages WebSocket connections and message handling on the client side, ensuring messages appear in real-time.

6. Functional Workflow

1. **User Connection**: Upon visiting the application URL, users automatically connect to the WebSocket server.
2. **Message Exchange**:
    - Messages are typed in the chat input and sent using WebSocket connections.
    - The server processes these messages and broadcasts them to all active clients.
3. **Message Display**:
    - Sent messages are aligned to the right for the sender.
    - Received messages are aligned to the left for a visually organized chat experience.

7. Setup and Installation

Running the Project Locally

Prerequisites

Before running the project, ensure the following are installed on your system:

- Java 17 or above
- Maven

Steps to Run the Backend

1. **Clean and Build the Project**
    - Open a terminal (or command prompt).
    - Navigate to the project directory and run the following command to clean and build the project using Maven: mvn clean install
2. **Run the Spring Boot Application**
    - After the build is successful, start the server by executing: mvn spring-boot: run

Accessing the Application

Once the backend is running, open your web browser and go to: http://localhost:8080

8. Testing and Validation

Manual Testing

1. **Functional Test**: Test message sending and receiving in real-time with multiple browser instances or users to ensure real-time capabilities.
2. **UI Testing**: Verify the alignment of sent and received messages to ensure user-specific message alignment.

Performance Testing

- Conduct load testing to evaluate the application's behavior under multiple simultaneous WebSocket connections.
- Measure response times and message delays during high traffic.

9. Conclusion

The Real-Time Chat Application successfully demonstrates the use of Spring Boot and WebSockets to create a reliable, real-time communication platform. While the current implementation is effective for instant messaging between users, there are opportunities for further enhancement, such as adding user authentication, group chat functionality, and message persistence. This application serves as a foundation for building scalable chat platforms and can be expanded with additional features as needed.