# WebSocket-test.html

This document demonstrates how to use a WebSocket client to test backend communication via SockJS and STOMP. The goal is to showcase connection, message transmission, and debugging in a simple and clear manner. – This serves as the foundation for implementing the chat functionality in the project.

## Technology and Setup

The file is based on HTML and JavaScript. The code uses SockJS to establish the connection and STOMP.js to implement the messaging protocol. The user specifies a server URL (for example, http://localhost:8080/ws) and then sends messages to the backend. The interface consists of a field for the URL, input fields for sender and receiver, and a text area for the message. Everything is logged in a separate section to make debugging and verifying the communication easier.

## Functionality

When the user clicks "Connect", a WebSocket connection is established. The connection status is displayed on screen with brief feedback such as "Connecting" and "Connected". If something goes wrong, a short error message is logged. Once the connection is established, the user can send a message by filling in the sender ID, receiver ID, and message text. The message is sent to the backend via the destination /app/chat.sendMessage. The client receives responses on the topic /topic/messages, as well as updates on /topic/messages.read and /topic/messages.update. There is also a function to send a simple "ping", which tests the stability of the connection.

## Testing Procedure

To test the solution, the backend server must be running and properly configured for WebSocket communication. The student opens websocket-test.html in a modern

browser, fills in the URL, and clicks "Connect". Then, test messages are sent to verify that the message exchange works. Finally, disconnection is tested using the "Disconnect" button, which also updates the visual feedback.