**Assignment 9: Real-Time Communication with WebSockets**

**Learn the basics of WebSockets for real-time communication.**

**Implement a simple real-time feature in your MERN app.**

**WebSockets provide a full-duplex communication channel over a single, long-lived connection, enabling real-time communication between a client and a server. In this assignment, I'll guide you through the process of implementing a simple real-time feature in a MERN (MongoDB, Express.js, React, Node.js) app using WebSockets. We'll use the popular socket.io library for this purpose.**

**Step 1: Install socket.io in Express.js**

**Install the socket.io library in your Express.js server:**

**npm install socket.io**

**Step 2: Set Up WebSocket Server in Express.js**

**Modify your Express.js server (express-api/app.js) to include WebSocket support using socket.io:**

| **const express = require('express');**  **const http = require('http');**  **const socketIo = require('socket.io');**  **const app = express();**  **const server = http.createServer(app);**  **const io = socketIo(server);**  **const port = 3001;**  **io.on('connection', (socket) => {**  **console.log('A user connected');**  **// Handle events when a user sends a message**  **socket.on('sendMessage', (message) => {**  **// Broadcast the message to all connected clients**  **io.emit('receiveMessage', message);**  **});**  **// Handle events when a user disconnects**  **socket.on('disconnect', () => {**  **console.log('A user disconnected');**  **});**  **});**  **// Other routes and middleware**  **server.listen(port, () => {**  **console.log(`Express API with WebSocket is listening at http://localhost:${port}`);**  **});** |
| --- |

**Step 3: Integrate WebSocket in React App**

**Install socket.io-client in your React app**

| **npm install socket.io-client** |
| --- |

**Update your React component (src/DataFetching.js) to include WebSocket functionality:**

| **// src/DataFetching.js**  **import React, { useState, useEffect } from 'react';**  **import axios from 'axios';**  **import io from 'socket.io-client';**  **const socket = io('http://localhost:3001'); // Adjust the URL based on your server configuration**  **const DataFetching = ({ token }) => {**  **const [data, setData] = useState([]);**  **const [loading, setLoading] = useState(true);**  **const [error, setError] = useState(null);**  **const [message, setMessage] = useState('');**  **const [messages, setMessages] = useState([]);**  **useEffect(() => {**  **const fetchData = async () => {**  **try {**  **const response = await axios.get('http://localhost:3001/api/data/authenticated', {**  **headers: {**  **Authorization: `Bearer ${token}`,**  **},**  **});**  **setData(response.data.data);**  **setLoading(false);**  **} catch (error) {**  **setError(error.message);**  **setLoading(false);**  **}**  **};**  **fetchData();**  **}, [token]);**  **useEffect(() => {**  **// Listen for incoming messages from the server**  **socket.on('receiveMessage', (receivedMessage) => {**  **setMessages((prevMessages) => [...prevMessages, receivedMessage]);**  **});**  **// Cleanup on component unmount**  **return () => {**  **socket.off('receiveMessage');**  **};**  **}, []);**  **const handleSendMessage = () => {**  **// Emit a 'sendMessage' event to the server**  **socket.emit('sendMessage', message);**  **setMessage(''); // Clear the input field after sending**  **};**  **return (**  **<div>**  **<h2>Data Fetching with Authentication</h2>**  **{loading ? (**  **<p>Loading...</p>**  **) : error ? (**  **<p>Error: {error}</p>**  **) : (**  **<div>**  **<ul>**  **{data.map((item) => (**  **<li key={item.id}>{item.title}</li>**  **))}**  **</ul>**  **<div>**  **<h3>Real-Time Chat</h3>**  **<ul>**  **{messages.map((msg, index) => (**  **<li key={index}>{msg}</li>**  **))}**  **</ul>**  **<input**  **type="text"**  **value={message}**  **onChange={(e) => setMessage(e.target.value)}**  **placeholder="Type a message..."**  **/>**  **<button onClick={handleSendMessage}>Send</button>**  **</div>**  **</div>**  **)}**  **</div>**  **);**  **};**  **export default DataFetching;** |
| --- |

**Step 4: Run Your MERN App**

**Start both your Express.js server and React development server:**

| **# Start Express.js server**  **node express-api/app.js**  **# Start React development server**  **npm start** |
| --- |

**Visit http://localhost:3000 in your browser. You should see your React app fetching data from the protected API route, and a real-time chat feature where messages are instantly updated for all connected clients.**

**This is a basic example of integrating WebSockets into a MERN app for real-time communication. Depending on your application's requirements, you may want to add more features, security measures, and error handling.**