**Web Sockets in Express js**

Certainly, here's an example using Socket.IO for real-time communication between a server and multiple clients:

**Server-side (server.js):**

JavaScript

const express = require('express');

const app = express();

const http = require('http');

const server = http.createServer(app);

const { Server } = require("socket.io");

const io = new Server(server);

app.use(express.static('public')); // Serve static files from the 'public' directory

io.on('connection', (socket) => {

  console.log('Client connected');

  // Handle 'chat' message from client

  socket.on('chat', (msg) => {

    console.log(`Message from client: ${msg}`);

    // Broadcast the message to all connected clients (including the sender)

    io.emit('chat', msg);

  });

  // Handle client disconnection

  socket.on('disconnect', () => {

    console.log('Client disconnected');

  });

});

server.listen(3000, () => {

  console.log('Server listening on port 3000');

});

**Client-side (public/index.html):**

HTML

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Socket.IO Chat</title>

  <script src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/4.5.0/socket.io.js"></script>

</head>

<body>

  <h1>Socket.IO Chat</h1>

  <input type="text" id="message" placeholder="Enter your message">

  <button id="send-button">Send</button>

  <ul id="chat-messages"></ul>

  <script>

    const socket = io();

    const messageInput = document.getElementById('message');

    const sendButton = document.getElementById('send-button');

    const chatMessages = document.getElementById('chat-messages');

    sendButton.addEventListener('click', () => {

      const message = messageInput.value;

      if (message) {

        socket.emit('chat', message);

        messageInput.value = ''; // Clear the input field after sending

      }

    });

    socket.on('chat', (msg) => {

      const newMessage = document.createElement('li');

      newMessage.textContent = msg;

      chatMessages.appendChild(newMessage);

    });

  </script>

</body>

</html>

**Explanation:**

1. **Server-side:**
   * The code sets up an Express server and integrates Socket.IO.
   * It serves static files from the public directory, which includes the HTML file for the client.
   * The io.on('connection', ...) event listener handles incoming client connections.
   * Inside the listener:
     + socket.on('chat', ...) listens for the 'chat' message sent by the client.
     + The received message is logged and then broadcasted to all connected clients using io.emit('chat', msg).
     + socket.on('disconnect', ...) handles client disconnection.
2. **Client-side:**
   * The HTML file includes the Socket.IO library and sets up the chat interface.
   * The JavaScript code:
     + Establishes a connection to the server using const socket = io().
     + Attaches event listeners to the message input field and the send button.
     + When the send button is clicked, the message is emitted to the server using socket.emit('chat', message).
     + The socket.on('chat', ...) listener receives broadcasted messages from the server and adds them to the chat list.

**Running the example:**

1. Save the server-side code as server.js and the client-side code as public/index.html in the same directory.
2. Open a terminal and navigate to the directory containing these files.
3. Run the server: node server.js
4. Open http://localhost:3000 in multiple browser windows to simulate multiple clients.
5. You can now type messages in one browser window and see them displayed in all connected windows, demonstrating real-time chat functionality.

This is a basic example of Socket.IO. You can extend it further by:

* Implementing user authentication and