PRACTICAL 9

Building real-time web application using WebSocket and a server-side framework (e.g. Socket.io, Django channels)

server.js

```
import express from 'express';
import { createServer } from 'http';
import { Server } from 'socket.io';
import cors from 'cors';
import path, { join } from 'path';
import url from 'url';

const app = express();

//middlewares
app.use(cors());

//create HTTP server
const httpServer = createServer(app);

//get __dirname in ES modules
const __dirname = path.dirname(url.fileURLToPath(import.meta.url));

//initialize Socket.io
const io = new Server(httpServer, {
  cors: {
    origin: '*', // adjust as needed
  },
});

io.on('connection', (socket) => {
  console.log("A user connected");

  //emit event for active clients update on connection
  let activeClients = io.engine.clientsCount;
  io.emit('active-clients', activeClients);
  console.log("Active clients:", activeClients);

  socket.on('disconnect', () => {
    console.log("User disconnected");
    activeClients = io.engine.clientsCount;
    io.emit('active-clients', activeClients);
    console.log("Active clients:", activeClients);
  });

  socket.on('message', (msg) => {
    io.emit('message-r', msg);
  });
});

//serve HTML file
app.get("/", (req, res) => {
  res.sendFile(join(__dirname, 'index.html'));
});
```

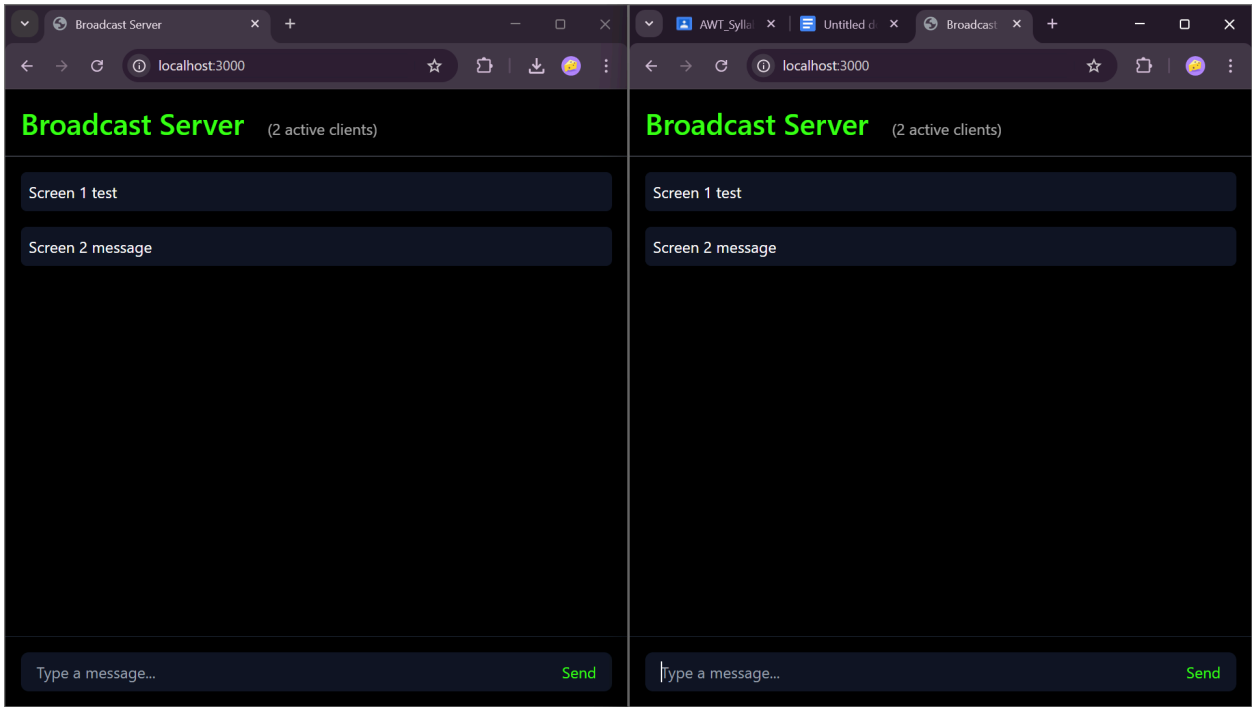
```
//start server
httpServer.listen(3000, () => {
  console.log("Server is running on http://localhost:3000");
});
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <script src="https://cdn.tailwindcss.com"></script>
  <script>
    tailwind.config = {
      theme: {
        extend: {
          colors: {
            "neon-green": "#39FF14",
          },
        },
      },
    };
  </script>
  <title>Broadcast Server</title>
</head>

<body class="bg-black text-white flex flex-col h-screen">
  <!-- Header -->
  <h1 class="text-3xl font-semibold pl-4 py-4 text-neon-green border-b
border-gray-500 flex items-center">
    Broadcast Server
    <span id="active-clients" class="ml-4 text-white opacity-65 font-normal
text-base"></span>
  </h1>

  <!-- Chat Messages -->
  <div class="flex-grow overflow-y-auto p-4 space-y-4" id="chat-messages">
    <!-- Messages appear here -->
  </div>

  <!-- Message Form -->
  <form id="chat-form" class="p-4 border-t border-gray-800 flex">
    <input
      type="text"
      id="message-input"
      class="flex-grow bg-gray-900 text-white rounded-l-lg px-4 py-2
focus:outline-none"
      placeholder="Type a message..."
      autocomplete="off"
    />
    <button
      id="send-button"
      class="bg-gray-900 text-neon-green rounded-r-lg px-4 py-2
hover:bg-gray-800"
    >
      Send
```

```
    </button>
  </form>

  <!-- Socket.IO -->
  <script src="/socket.io/socket.io.js"></script>
  <script>
    const socket = io();
    const msgContainer = document.getElementById("chat-messages");

    // Receive and display messages
    socket.on("message-r", (msg) => {
      const messageElement = document.createElement("div");
      messageElement.className = "bg-gray-900 rounded-md p-2";
      messageElement.textContent = msg;
      msgContainer.appendChild(messageElement);
      msgContainer.scrollTop = msgContainer.scrollHeight; // auto-scroll
    });

    // Send message
    const form = document.getElementById("chat-form");
    const input = document.getElementById("message-input");
    form.addEventListener("submit", (e) => {
      e.preventDefault();
      if (input.value.trim() === "") return;
      socket.emit("message", input.value);
      input.value = "";
    });

    // Active clients display
    const activeClientsCount = document.getElementById("active-clients");
    socket.on("active-clients", (clients) => {
      activeClientsCount.textContent = `(${clients} active clients)`;
    });

    // Media device access
    const openMediaDevices = async (constraints) => {
      return await navigator.mediaDevices.getUserMedia(constraints);
    };

    try {
      const stream = await openMediaDevices({ audio: true });
      console.log("Got MediaStream:", stream);
    } catch (error) {
      console.error("Error accessing media devices.", error);
    }

    // Get list of microphones
    const getConnectedDevices = async (type) => {
      const devices = await navigator.mediaDevices.enumerateDevices();
      return devices.filter((device) => device.kind === type);
    };

    getConnectedDevices("audioinput").then((mics) => {
      console.log("Mics found:", mics);
    });
  </script>
</body>
```

```
</html>
```


Broadcast Server
localhost:3000
Broadcast Server
(2 active clients)
Screen 1 test
Screen 2 message
Type a message...
Send
AWT_Syllal
Untitled d
Broadcast
localhost:3000
Broadcast Server
(2 active clients)
Screen 1 test
Screen 2 message
Type a message...
Send