**Explanation:**

* This script starts a WebSocket server on port 8080.
* The WebSocket server wraps the ChatGPTWebSocket class, which handles WebSocket communication and forwarding requests to ChatGPT.

**5. Running the WebSocket Server**

To start the WebSocket server, simply run the server.php file.

bash

Copy code

php server.php

Output:

arduino

Copy code

WebSocket server running on port 8080...

**6. Client-Side WebSocket Connection**

To interact with this WebSocket server from a client, such as a web browser or an external service, you can use JavaScript’s native WebSocket API.

Here’s an example of how a client can communicate with your WebSocket API:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>ChatGPT WebSocket Client</title>

</head>

<body>

<h1>ChatGPT WebSocket Client</h1>

<button id="sendMessage">Send Message</button>

<div id="response"></div>

<script>

const ws = new WebSocket('ws://localhost:8080'); // Connect to the WebSocket server

ws.onopen = function() {

console.log("Connected to WebSocket");

};

ws.onmessage = function(event) {

console.log("Message from server: ", event.data);

document.getElementById('response').innerText = "Response: " + event.data;

};

ws.onclose = function() {

console.log("Disconnected from WebSocket");

};

document.getElementById('sendMessage').onclick = function() {

const message = {

input: "Tell me more about buying a house.",

industry: "real\_estate"

};

ws.send(JSON.stringify(message)); // Send the user query to the WebSocket server

};

</script>

</body>

</html>

**Explanation:**

1. The JavaScript code creates a WebSocket connection to ws://localhost:8080 (assuming your PHP WebSocket server is running locally).
2. When a button is clicked, it sends a message (a user query) to the WebSocket server.
3. The server processes the message, interacts with ChatGPT, and sends the response back. The response is then displayed on the webpage.

**7. How to Deploy**

* **Hosting**: If you plan to host the WebSocket server on a remote server, ensure the server is configured to allow connections to the chosen WebSocket port (e.g., 8080).
* **Load Balancing**: If you expect a lot of traffic, you might need to use a load balancer to distribute WebSocket connections across multiple servers.
* **Security**: Consider using **wss://** for secure WebSocket connections, especially in production environments.