Client Server Protocol

To create a client-server protocol for our chat app we intend to focus on setting up both the client-side and server-side components.

This is our proposed method;

Server-Side Setup

-Firstly we setup Django

-Then we create an App within our Django project, for the chat functionality.

-Then we define database models for users, messages, and chat rooms

- Then we use use Django Channels to handle real-time WebSocket connections for our WebSocket Integration by creating an ASGI Configuration to handle WebSocket connections, define WebSocket Routing by creating a routing.py file in the chat app and creating a consumer in consumers.py to handle WebSocket connections.

For Client-Side Setup

We will create a simple HTML page with JavaScript to handle WebSocket connections.

Thus our Chatapp Client-Server Protocol will use the WebSocket protocol for real-time, bidirectional communication between the client (browser) and server (Django). This is key to carrying out ;

Connection Establishment: The client establishes a WebSocket connection to the server using a URL

Message Transmission: Messages are sent as JSON objects over the WebSocket connection. The server receives the message, processes it, and broadcasts it to all clients connected to the same chat room.

Broadcasting: The server uses Django Channels to broadcast messages to all clients in the chat room in real-time.