Architecture & Design Explanation: AI Chatbot Platform

1. High-Level Architecture

The application follows a **Decoupled Client-Server Architecture**. This design ensures that the user interface (React) and the business logic (Spring Boot) remain independent, communicating securely through a **RESTful API**.

2. Tech Stack Rationale

- **Frontend (React.js):** Chosen for its component-based architecture, which allows for a highly responsive chat interface and efficient state management.
- **Backend (Spring Boot):** Provides a robust framework for managing security, database interactions, and integration with high-performance LLM APIs.
- **Database (MySQL):** Used for persistent storage of user credentials and custom AI Agent configurations.
- **AI Engine (Groq Cloud + Llama 3.1):** Leveraged for near-instant inference speed, ensuring the chatbot feels real-time.

3. Key Design Patterns & Security

- **Stateless Session Design:** To ensure maximum accuracy and prevent AI "hallucinations," we implemented a memory-less approach where the AI focuses strictly on the current project's **System Prompt** for every interaction.
- **JWT Security Layer:** We implemented a custom **Spring Security Filter** to intercept and validate JSON Web Tokens (JWT). This ensures that only authenticated users can access their private agents and chat sessions.
- **System Prompt Injection:** The design allows for dynamic "personality" switching. When a user selects an agent, the backend injects specific instructions into the API payload sent to the LLM.

4. Reliability Features

- **Smart Mock Fallback:** To handle external API rate limits or outages, the system includes a reliability layer that triggers a "Mock" response engine, ensuring the user always receives feedback instead of an error.
- **Markdown Rendering:** The frontend is designed to parse and render complex AI outputs, including **dark-themed code blocks** and bold text, providing a professional developer-centric experience.

5. Data Flow

1. **User Input:** The user sends a message via the React interface.
2. **Auth Check:** The Spring Boot backend verifies the **JWT token**.
3. **Prompt Construction:** The backend retrieves the Agent's system prompt from MySQL and combines it with the user message.
4. **AI Inference:** The request is sent to the **Groq API**.
5. **Response:** The structured response is returned and rendered in Markdown on the UI.