Architecture & Design Explanation

This project is designed with a modular client-server architecture, ensuring scalability, security, and maintainability. It is structured into three core layers — backend, frontend, and integration — each with clearly defined responsibilities.

1. **Backend (FastAPI):**

   The backend is responsible for handling core application logic including user authentication, project management, file handling, and AI-driven chat requests. It communicates securely with the OpenRouter API (Google Gemini 1.5 Pro) to process and return intelligent responses. Authentication is implemented using JWT tokens to ensure secure user sessions. Currently, data is stored in an in-memory structure for simplicity, but the design is adaptable for integration with production-ready databases such as PostgreSQL or MongoDB, ensuring scalability and persistence.

2. **Frontend (React):**

   The frontend delivers a responsive and user-friendly interface where users can register, log in, create projects, and interact with the AI chat system. It communicates with backend APIs using secure HTTP requests and manages user sessions through locally stored JWT tokens. The design emphasizes modular components, ensuring maintainability and scalability. React's state management allows smooth handling of chat histories and project data, while its flexible architecture supports future enhancements such as real-time updates and advanced UI features.

3. **Integration Layer:**

   The integration layer connects the backend services with the OpenRouter API, enabling AI-powered interactions through the Google Gemini 1.5 Pro model. It handles message formatting, API requests, and response parsing to ensure smooth communication between the user interface and the AI engine. This layer also applies safeguards such as token limits to optimize cost and prevent errors. Its modular design allows easy replacement or extension of AI models and third-party APIs in the future, ensuring long-term adaptability.

4. **System Design Overview:**

   The system is built on the principle of separation of concerns, ensuring that the **frontend**, **backend**, and **AI integration layer** remain loosely coupled. This modular approach improves maintainability, as updates or changes in one component do not directly impact the others. It also enhances scalability, allowing the system to grow by integrating new models, APIs, or

user interface features without major rework. The design prioritizes flexibility and long-term adaptability, making it suitable for both experimental and production environments.

System Architecture Diagram