**Codebase Structure**

The codebase follows a clean, modular layout for ease of development and future expansion. Here is a deeper dive into the components:

project-root/

├── venv/ # Isolated Python environment containing dependencies

├── main.py # Main application file; integrates UI, file handling, and AI calls

├── requirements.txt # Required libraries (streamlit, pandas, google-generativeai, etc.)

├── .env # Environment file for sensitive keys like GOOGLE\_API\_KEY

**Key Design Principles:**

* **Modularity:** Functions are separated (e.g., load\_file(), generate\_summary(), create\_visuals()) for reusability.
* **Security:** Sensitive API keys are stored in a .env file and accessed using environment variables, avoiding hardcoded secrets.
* **Reproducibility:** The requirements.txt file ensures every contributor runs the app with consistent dependencies.
* **Scalability:** With a clear separation between UI, logic, and configuration, the app can be extended with additional financial documents, dashboard features, or alternate LLM backends.

**Contribution Scenario:**

* A new developer can clone the repo, create a .env file with their own API key, run pip install -r requirements.txt, and launch main.py via streamlit run main.py.
* Additions like PDF export or multi-language summarization can be built without modifying core logic.

**Real-World Context:**

* *Sarah* can easily plug in newer quarterly reports without needing tech support.
* *Rajesh* can request new charts (e.g., ratio analysis) by asking a contributor who can add it modularly.
* *Mark* can export insights once a future version includes a generate\_pdf() utility.

This codebase is not just functional—it’s future-ready.