Technology Stack Documentation

This document outlines the complete technology stack used in the project. The architecture follows modern Python development practices, combining a high-performance backend, a lightweight frontend, a relational database, and locally hosted AI services.

Backend (API Layer)

- **FastAPI** – High-performance Python web framework used for API routing, dependency injection (authentication, database sessions), and automatic OpenAPI documentation.
- **Uvicorn** – ASGI server responsible for running the FastAPI application efficiently.
- **Pydantic** – Handles data validation and serialization using strongly-typed schemas for requests and responses.
- **Python-dotenv** – Loads environment variables such as API keys and database URLs from a .env file into the runtime environment.

Frontend (User Interface)

- **Streamlit** – Enables rapid development of an interactive web interface for authentication, repository selection, and project generation without requiring frontend frameworks like React.

Database Layer

- **SQLAlchemy** – Python ORM used to define database models and perform database operations using Python objects instead of raw SQL queries.
- **SQLite** – Lightweight, file-based relational database (app.db) used for local development and fast prototyping.

AI & External Services

- **Ollama** – Runs open-source large language models locally (e.g., Llama 3) to parse and summarize README files without relying on cloud APIs.
- **GitHub REST API** – Used for user authentication, repository listing, and fetching README content securely.
- **HTTPX** – Asynchronous HTTP client used for making non-blocking requests to the GitHub API and local Ollama service.