System Architecture – GitHub Project Generator

This document presents a clean and high-level system architecture of the GitHub Project Generator. It explains how data flows across the frontend, backend, external services, AI layer, and database. The architecture emphasizes modularity, security, and maintainability.

Architecture Layers Overview

- Presentation Layer – Streamlit-based frontend for user interaction and visualization.
- Application Layer – FastAPI backend responsible for orchestration, authentication, and business logic.
- Integration Layer – GitHub REST API and Ollama AI service.
- Data Layer – SQLite database for persistent storage.

Logical Architecture Diagram

User
Streamlit Frontend
FastAPI Backend
GitHub API | Ollama AI
SQLite Database

End-to-End Data Flow

- User authenticates via Streamlit using a GitHub Personal Access Token.
- Streamlit sends authentication and action requests to FastAPI backend.
- FastAPI validates identity using GitHub REST API.
- Repository metadata and README files are fetched from GitHub.
- README content is processed by Ollama-hosted local LLM.
- Structured project summary is generated and stored in SQLite.
- Final output is returned to the frontend for user visualization.

Component Responsibilities

- **Streamlit Frontend**: Handles user authentication, repository selection, and result display.
- **FastAPI Backend**: Manages routing, security, orchestration, and AI invocation.
- **GitHub API**: Acts as the source of repository data and user identity.
- **Ollama AI**: Processes unstructured README content into structured project data.
- **SQLite Database**: Persists users, repositories, and generated projects.

Key Architectural Characteristics

- Modular design with clear separation of concerns.
- Secure token-based authentication.
- Offline-capable AI inference using local models.
- Scalable backend architecture.
- Lightweight and developer-friendly deployment.