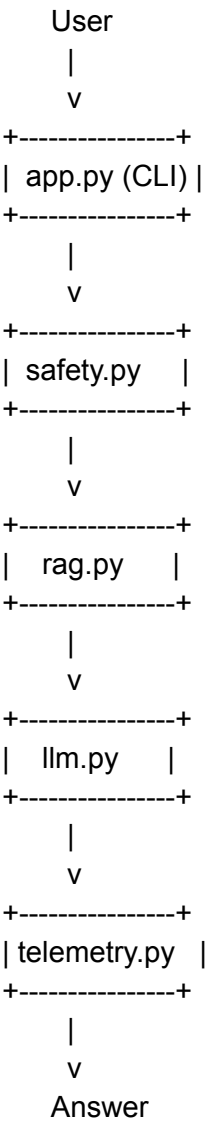


Technical Note

Architecture Diagram



## Overview of System Architecture

The Study Buddy application is a command-line tool that answers questions about a user's local study notes. The user interacts with the app through a simple CLI, where they can ask questions and receive answers. The system uses a Retrieval-Augmented Generation (RAG) pipeline to ground the LLM's responses in the user's own data. This is achieved by converting the notes and the user's question into vector embeddings and then using similarity to find the most relevant text chunks. These chunks are then passed to the LLM, which generates an answer based on the provided context.

### Guardrails Implemented

**System Prompt Rules:** The LLM is instructed to only use the provided notes, not to hallucinate, and to respond with "I don't know" if the answer is not in the context.

**Input-Length Validation:** The application rejects any input longer than 300 characters to prevent resource exhaustion and abuse.

**Prompt-Injection Detection:** The system checks for and blocks common prompt-injection phrases, such as "ignore previous instructions," to prevent malicious users from bypassing the system prompt.

**Error Fallback Response:** In the case of any unexpected errors, the application provides a generic and user-friendly error message, preventing technical details from being exposed to the user.

### Evaluation Method (Offline Eval)

The application includes an offline evaluation suite to measure the quality of the RAG pipeline. The evaluation is based on a `tests.json` file, which contains a list of test cases, each with an input question and an expected output pattern. The evaluation script (`run_tests.py`) iterates through each test case, runs the RAG pipeline, and checks if the generated answer contains the expected pattern. The script then reports a pass rate, which provides a quantitative measure of the system's performance.

### Telemetry

The application logs each request to a local file (`studybuddy_logs.txt`). Each log entry includes a timestamp, the latency of the LLM response, whether the RAG pathway was used, and the token count for the response. This data is useful for monitoring the performance of the application and for understanding how it is being used.

## Known Limitations

**Local-Only:** The RAG pipeline is limited to a small set of local note files and does not scale to large document repositories.

**No Persistent Memory:** The application does not have any memory of previous conversations.

**Variable LLM Output:** The quality of the LLM's output can vary depending on the model used and the hardware it is run on.

**Basic CLI UX:** The user experience is limited to a basic command-line interface.

When adding new files to the notes, you must delete `embeddings.json` so the app will be forced to scan the notes folder again.