AHxAI: A Retrieval-Augmented Code Assistant

Ali Abdallah Hassan Mohsen

Faculty of Engineering III, Lebanese University
{alikhaledabdallah454,hassanmohseen22}@gmail.com

18 July 2025

Abstract

AHxAI is a full-stack coding assistant that combines Retrieval-Augmented Generation (RAG) with different LLMs to deliver context-aware code analysis, refactoring and explanation. A dual-LLM workflow first extracts *useful libraries* from user requests, then invokes tool-enabled prompts to fetch code snippets and documentation, bridging gaps in both public and private knowledge bases. The system is deployed as a FastAPI + React application backed by PostgreSQL and Pinecone, wrapped in Docker for reproducible development. We detail the motivation, architecture and implementation choices, and we discuss initial qualitative results and future extensions.

1 Introduction

Large Language Models (LLMs) have revolutionised developer tooling, yet *generic* models suffer from outdated public corpora, no access to proprietary code and privacy constraints. AHxAI addresses these limitations via RAG (Figure 1) to ground LLMs in project-specific context.

2 Problem Statement

Limitations of generic LLMs include (i) stale or incomplete knowledge of public libraries, (ii) zero visibility into internal code bases and (iii) elevated privacy / data-leak risks. Consequently, naïve prompting yields hallucinations and unsafe suggestions—motivating an augmented approach.

3 Methodology

3.1 Retrieval-Augmented Generation

Our workflow (RAG based) embeds code/doc snippets into Pinecone; semantic, lexical and hybrid searches retrieve relevant context, which is stitched into the final LLM prompt.

3.2 Knowledge Bases

- **Public libs** (e.g. `numpy`, `matplotlib`) → queried live via Context7 API.
- **Private libs** → user-supplied docs/snippets embedded as vectors for secure in-house retrieval.

4 System Architecture

4.1 Backend

FastAPI orchestrates REST endpoints (`/process-code`, `/chat`, `/health`) and handles LLM calls through `langchain invoke method`. Data persist in PostgreSQL tables `code_sessions`, `chat_sessions` and `chat_messages`.

4.2 Frontend

A React SPA offers a code editor with syntax highlighting, real-time chat, collapsible sidebar and tabbed panes (Code, Explanation, Visualisation).

4.3 Dev-Ops

Docker Compose spins up *frontend*, *backend* and *db*. *Github* for code collaboration.

5 Implementation Details

5.1 LLM integration

Listing 1 shows the core invocation pattern using the `gemini-1.5-flash` model.

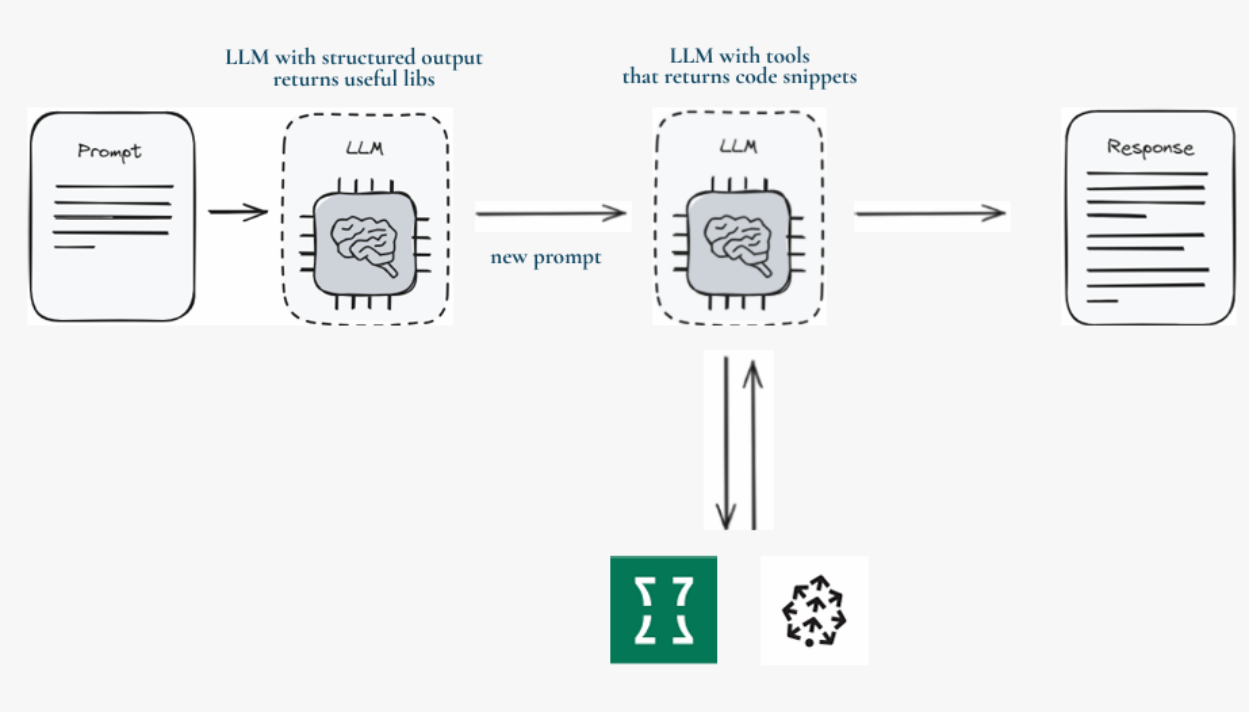

Figure 1: High-level pipeline of AHxAI.

```python
import google.generativeai as genai
from google.generativeai.types import
    GenerationConfig

genai.configure(api_key=os.getenv("
    GEMINI_API_KEY"))
model = genai.GenerativeModel("gemini-1.5-flash
    ")

resp = model.generate_content(
   prompt,
   generation_config=GenerationConfig(
      temperature=0.7, top_p=1,
          max_output_tokens=512))
```

Listing 1: Minimal Gemini call

Table	Key Columns
`code_sessions`	id, original_code, processed_code, created_at
`chat_sessions`	id, title, created_at
`chat_messages`	id, chat_id, message, response, created_at

Table 1: Core relational schema.

6 Conclusion

AHxAI demonstrates that bridging LLMs with targeted retrieval substantially improves coding assistance in privacy-sensitive settings. Future work will add unit-test generation, WebSocket streams for real-time collaboration and GPU-accelerated embedding pipelines.

Acknowledgements

We thank the open-source community and the Context7 team for API access.