Ever had the feeling how a certain code works when browsing others’ projects or when you are developing your own, you can’t seem to understand the code snippet you have managed to gather that might help you? Even when working with teams, sometimes it gets troublesome understanding the members’ codes and situations might arise where one might not be able to ask that person about it.

So, how to solve this?

Some would say copy the code and ask in community forums or use search engines. But customized codes would be very difficult to find and it would also take long time to get forum replies. Now, with the emergence of artificial intelligence, others would suggest using Large Language Models (LLMs) but even with this, one would need proper prompting ability.

That’s where my CodeDocs comes!

**What CodeDocs does?**

CodeDocs takes in a code snippet, analyzes it and returns a JSON-formatted documentation of the code with function summaries, missing docstrings, and improvement tips. And the best part? It even writes a story about the code, more like a real-life example on how the code works!

It uses Google’s Gemini-2.0-Flash model under the hood, along with a FAISS vector database to enable context-aware code retrieval, making explanations more accurate and contextually rich.

**Wondering how it works?**

Worry not! I will guide you through behind the scenes.

First of all, the user inputs the code snippet, selects an output format i.e. JSON or Story, and choose between basic or RAG-enhanced analysis. Secondly, the code snippet is embedded using Gemini’s embedding model, and stored in FAISS index allowing the system to retrieve similar code examples later. Next, it has the feature where the output can be enhanced using Retrieval-Augmented Generation (RAG). So, if it is selected, the app fetches the similar code snippets from the vector database to enrich the LLM prompt. After that, a custom prompt is built depending on the output type and Gemini then generates the explanation. Finally, the response is formatted using markdown, validated, and evaluated with basic checks like summary length or story richness.

**Features:**

* **Structured JSON**: Clean, parseable explanations perfect for APIs or developer tools.
* **Creative Story Mode**: A developer-friendly walkthrough of the code.
* **RAG for Smart Context**: Uses similar code to boost quality.
* **Quality Evaluation**: Basic checks to verify and rate the explanation.

All in all, I had a great time developing this fun CodeDocs project. Whether you are a solo dev, building tools for teams, or want to have a fun time understanding code, or just tired of writing docstrings — CodeDocs has your back.

*Interested in trying it or contributing? Let’s connect!*