Project Report: RAG-Based Python Code Understanding Assistant with Groq

1. Project Overview

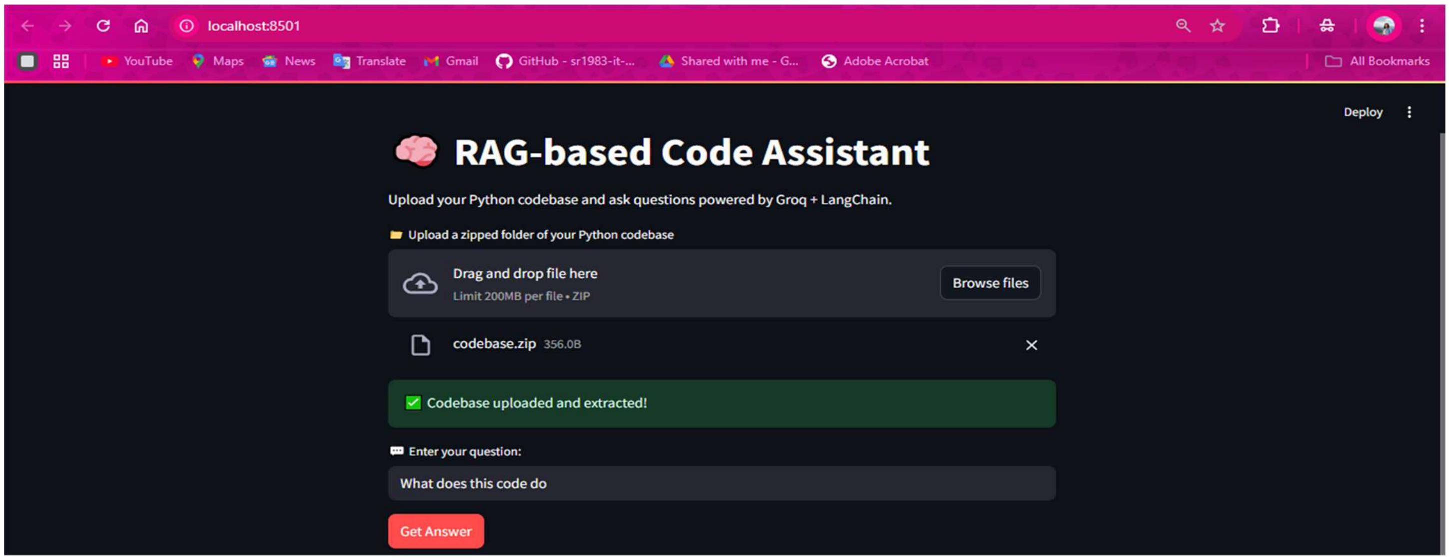
The goal of this project is to build a Retrieval-Augmented Generation (RAG) system that can understand and answer questions about a large Python codebase using LLMs (Large Language Models). Due to memory context limitations of LLMs, the RAG approach retrieves the most relevant code snippets and feeds them into the model to generate context-aware answers.

2. Tools & Technologies Used

Tool/Library	Purpose
LangChain	For vector database management and document loading
Groq	Ultra-fast inference API for LLMs (used with llama3)
FAISS	Vector similarity search to retrieve relevant code chunks
dotenv	To securely manage API keys
OpenAI Embeddings	To embed code chunks for vector search
PythonLoader (LangChain)	To parse .py files as documents
Streamlit (optional)	UI for user query input and output display

3. What We Built

A pipeline that:
1. Loads and splits `.py` files from a given folder.
2. Embeds those chunks using OpenAI-compatible embeddings.
3. Indexes them with FAISS for fast retrieval.
4. Accepts a user query, searches for relevant code snippets.
5. Feeds both query + retrieved code into a Groq-hosted LLM (llama3-70b-8192).
6. Returns a detailed natural language explanation.



RAG-based Code Assistant

Upload your Python codebase and ask questions powered by Groq + LangChain.

Upload a zipped folder of your Python codebase

Drag and drop file here
Limit 200MB per file • ZIP

codebase.zip 356.0B

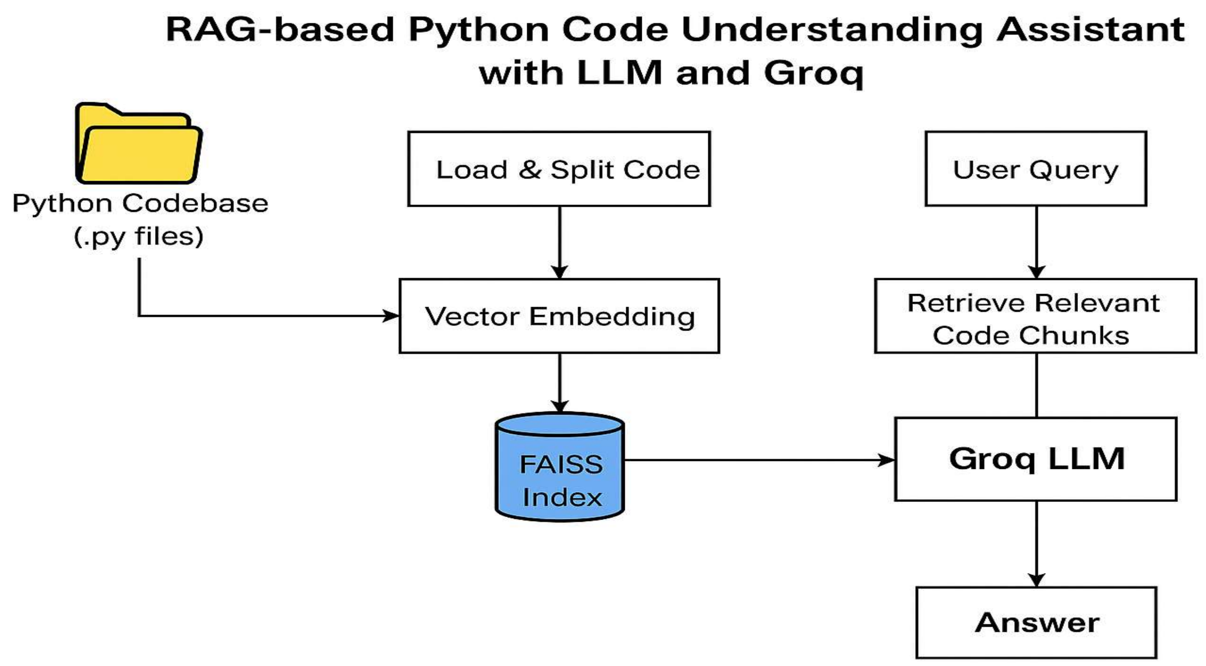
Codebase uploaded and extracted!

Enter your question:

What does this code do

4. How It Works

1. Directory Scanning:
   - Used PythonLoader to extract content from .py files.
2. Text Splitting:
   - Split into 1000-character chunks using RecursiveCharacterTextSplitter.
3. Vector Embedding:
   - Chunks embedded using OpenAIEmbeddings.
4. FAISS Indexing:
   - Stored vectors in FAISS index for similarity search.
5. Query Handling:
   - Retrieved top 5 similar chunks.
6. Groq LLM Inference:
   - Used llama3-70b-8192 with Groq SDK to generate an answer.



RAG-based Python Code Understanding Assistant with LLM and Groq
Python Codebase (.py files)
Load & Split Code
User Query
Vector Embedding
Retrieve Relevant Code Chunks
FAISS Index
Groq LLM
Answer

5. Errors & Troubleshooting Faced

Issue	Cause	Solution
Input must be string	LangChain formatting	Used Groq SDK directly
Model mixtral-8x7b-32768 decommissioned	Model no longer supported	Switched to llama3-70b-8192
Deprecation warnings	LangChain sub-package changes	Updated imports and packages
Directory not found	`codebase/` missing	Created folder and added `.py` files
Missing API key	Misconfigured .env	Used dotenv and correct variables

6. Conclusion

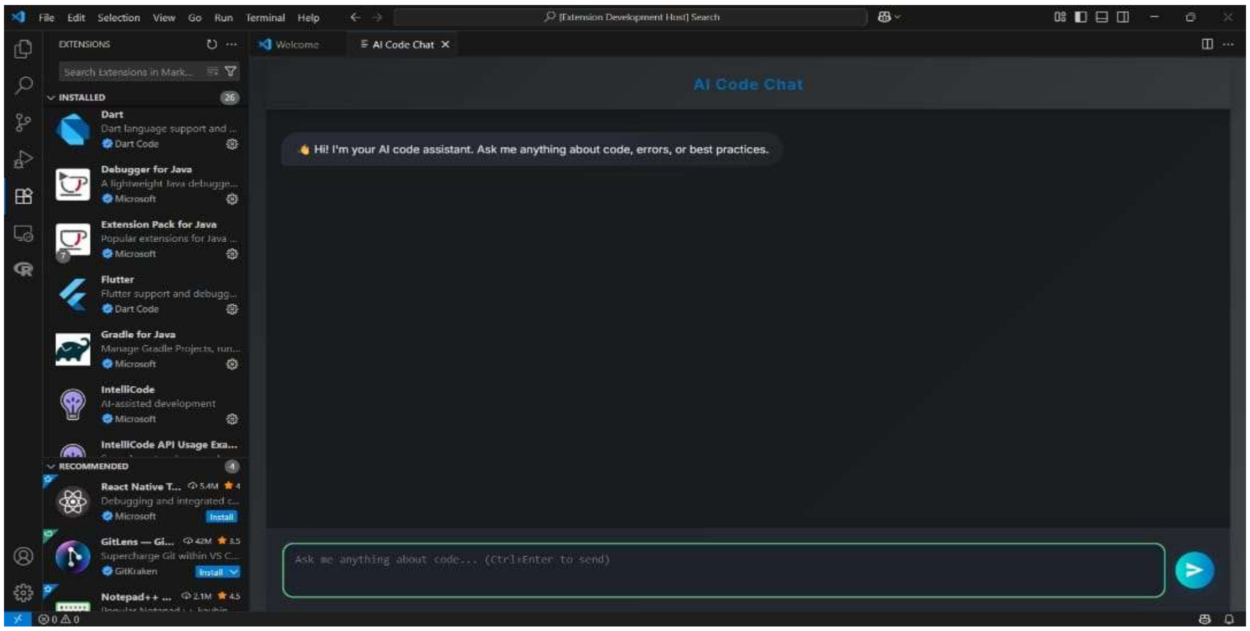
This project successfully demonstrates a real-world implementation of a code-aware assistant using a RAG (Retrieval-Augmented Generation) architecture. Despite challenges with model compatibility and library deprecations, we were able to:
- Retrieve relevant code chunks based on semantic similarity
- Feed those into a Groq-hosted LLM (llama3-70b-8192)
- Generate accurate explanations of the code logic

By replacing deprecated models and bypassing LangChain's LLM wrappers in favor of the Groq native SDK, we built a scalable assistant that can handle large codebases efficiently.

7. Acknowledging Team

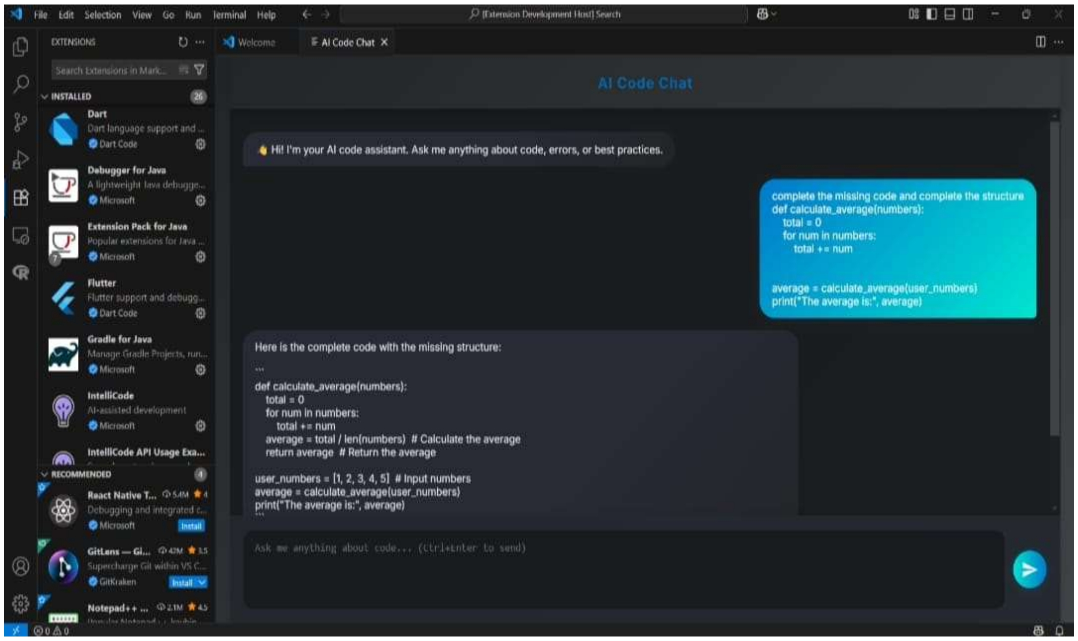
What my team offered in their models:

1. AI Code Chat: Real time VS Code extension made on Typescript and deployed over an extension for VS Code. (Himesh)
   Limitations: Working on RAG Implementation.



👋 Hi! I'm your AI code assistant. Ask me anything about code, errors, or best practices.

Ask me anything about code... (Ctrl+Enter to send)



AI Code Chat

👋 Hi! I'm your AI code assistant. Ask me anything about code, errors, or best practices.

```
complete the missing code and complete the structure
def calculate_average(numbers):
    total = 0
    for num in numbers:
        total += num


average = calculate_average(user_numbers)
print("The average is:", average)
```

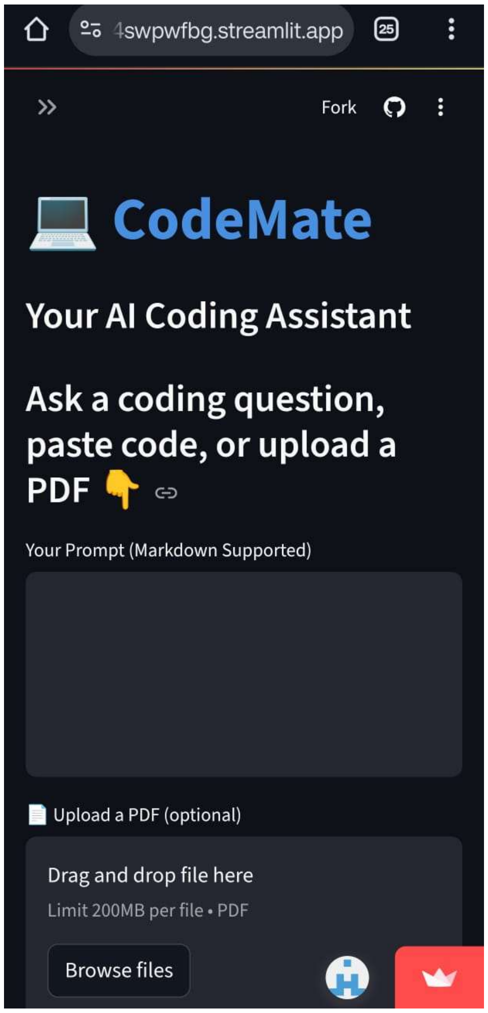
Here is the complete code with the missing structure:

```
def calculate_average(numbers):
    total = 0
    for num in numbers:
        total += num
    average = total / len(numbers)  # Calculate the average
    return average  # Return the average

user_numbers = [1, 2, 3, 4, 5]  # Input numbers
average = calculate_average(user_numbers)
print("The average is:", average)
```

Ask me anything about code... (Ctrl+Enter to send)

2. Code Mate: Streamlit based Coding assist interface. Working correctly over real time code files uploaded over the interface and answers the coding question as an AI Assistant. (Viraj)



💻 CodeMate

Your AI Coding Assistant

Ask a coding question, paste code, or upload a PDF 👇

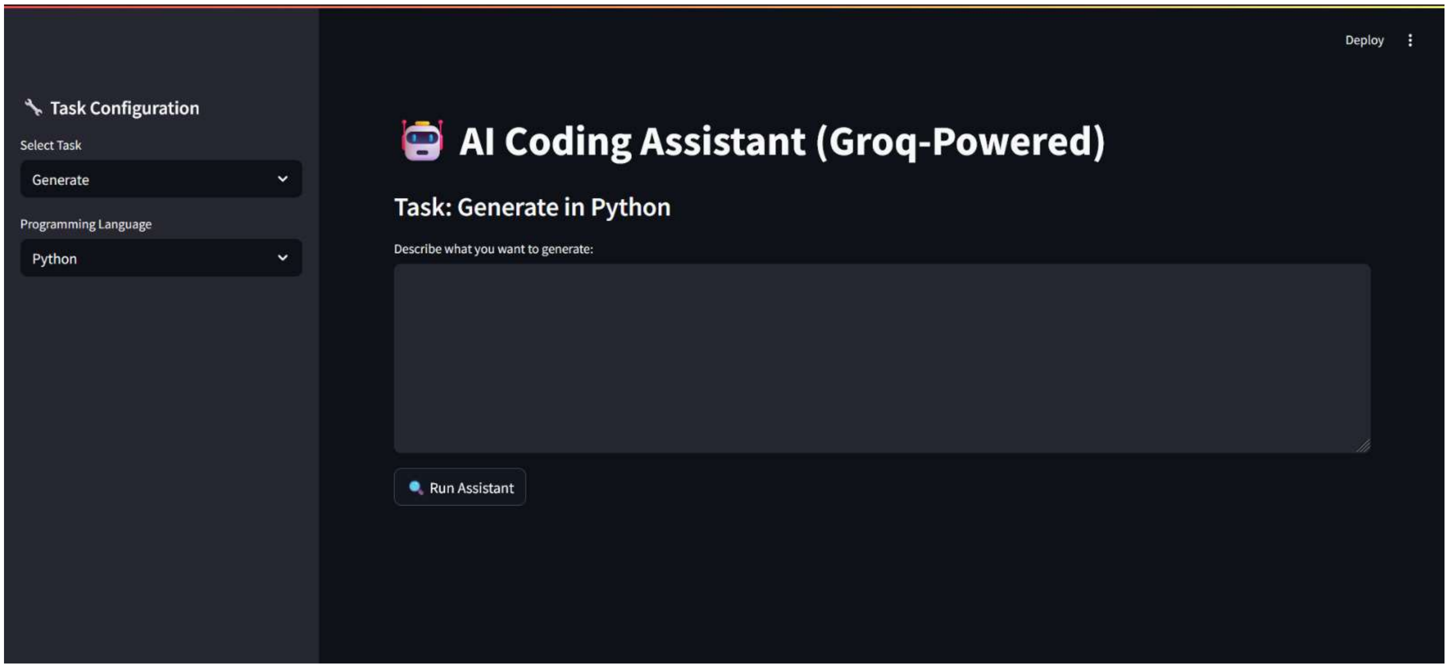
Your Prompt (Markdown Supported)

📄 Upload a PDF (optional)

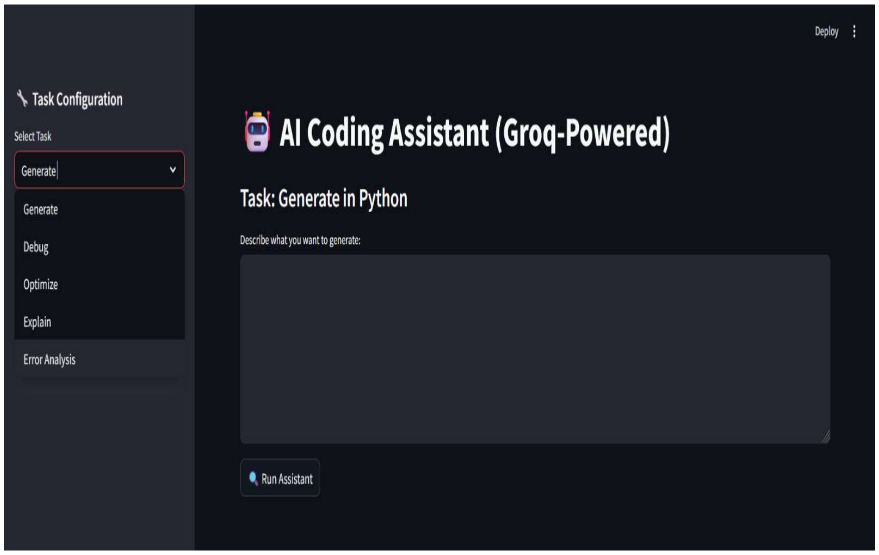
Drag and drop file here

Limit 200MB per file • PDF

3. AI Coding Assistant powered by Groq: Generates fast codes in selected coding language(Akriti).



It generates a code, debugs, optimizes and explains code ad analyzes error.



Thankyou!