CUSTOM RAG AND CONTINUE EXTENSION DOCUMENTATION

**Project Manager** – Nguyễn Trường Sơn

**Mentor** – Trương Minh Hiệp

**Intern** –Trần Quang An Quốc

Table of Contents

[1. Revision History 3](#_Toc180400701)

[2. Architecture Overview 3](#_Toc180400702)

[3. Flow of Operation 3](#_Toc180400703)

[3.1 Authentication: 3](#_Toc180400704)

[3.2 Indexing Code: 3](#_Toc180400705)

[3.3 Retrieving Code: 3](#_Toc180400706)

[3.4 Re-indexing Code: 3](#_Toc180400707)

[4. Related packages 4](#_Toc180400708)

[5. Running with Docker 4](#_Toc180400709)

[6. Related Documentations 4](#_Toc180400710)

## 1. Revision History

| Revision Number | Date | Comment |
| --- | --- | --- |
| 1.0 | September 30, 2024 | Explore RAG concepts, research Continue extension |
| 2.0 | October 3, 2024 | Explore Custom RAG system |
| 3.0 | October 7, 2024 | Utilize Gemini for embedding task and integrate into Continue extension |
| 4.0 | October 15, 2024 | Explore automatically re-indexing |
| 5.0 | October 21, 2024 | Add Docker deployment instructions |

## 2. Architecture Overview

This project utilizes several components:

* **Google Generative AI API**: Integrates AI-powered text embedding capabilities for content analysis.
* **LanceDB**: A vector database used for storing and querying embedded code chunks.
* **Tornado Web Framework**: Provides the web server infrastructure to handle HTTP requests and responses asynchronously.
* **PyJWT**: Used for generating and decoding JSON Web Tokens (JWTs) to authenticate and authorize users.
* **ThreadPoolExecutor**: Used for executing CPU-bound tasks asynchronously and concurrently.

## 3. Flow of Operation

### 3.1 Authentication:

* Clients authenticate using a unique token generated by the /auth endpoint.
* JWT tokens are used to authenticate subsequent requests to protected endpoints (/retrieve, /reindex, /index).

### 3.2 Indexing Code:

* **IndexHandler**: Accepts a GET request to /index with a folder\_path parameter.
* Retrieves all code files (\*.py) within the specified folder.
* Embeds chunks of code using Google Generative AI API and stores them in LanceDB.

### 3.3 Retrieving Code:

* **RetrieveHandler**: Accepts a POST request to /retrieve.
* Receives a query or input from the client, processes it, and searches for similar code chunks in LanceDB.
* Returns context items containing code file names, descriptions, and content based on search results.

### 3.4 Re-indexing Code:

* **ReindexHandler**: Accepts a POST request to /reindex with folder\_paths parameter.
* Deletes existing embeddings associated with specified folders.
* Indexes all code files within the specified folders again, updating LanceDB with new embeddings.

**3.5 Demonstration**

## 4. Related packages

*pip install –r requirements.txt*

## 5. Running with Docker

To run the application using Docker, follow these steps:

1. Ensure you have Docker installed on your system.
2. Build the Docker image:

* *docker build -t rag-system .*

1. Run the Docker container:

* *docker run -p 8000:8000 rag-system*
* This command will start the container and map port 8000 from the container to port 8000 on your host machine.

1. Access the application by opening a web browser and navigating to http://localhost:8000.

**Note: Make sure your Gemini API is added in file utils.py**

## 6. Related Documentations

* **Google Generative AI API Documentation**: [Google API Docs](https://developers.google.com/docs/api/reference/rest)
* **LanceDB Documentation**: [LanceDB Docs](https://lancedb.github.io/lancedb/)
* **Continue Documentation**: [Continue Docs](https://docs.continue.dev/customize/tutorials/custom-code-rag)
* **Tornado Documentation**: [Tornado Docs](https://www.tornadoweb.org/en/stable/index.html)
* **PyJWT Documentation**: [PyJWT Docs](https://pyjwt.readthedocs.io/en/stable/)
* **Docker Documentation**: [Docker Docs](https://docs.docker.com/)