**Cohere Generative AI Integration**

This project demonstrates how to integrate with Oracle's Cohere Generative AI Service using a Flask API, Python functions, and configuration management through environment variables and JSON files. The setup is modular, allowing you to run the service as a Flask API, use it as a callable function, or incorporate it into other Python applications.

**Project Structure**

* **Cohere\_GenAI\_API.py**: A Flask API that listens for incoming requests and processes them using the Cohere Generative AI service. This is a full API that can be hosted and accessed remotely by other applications.
* **Cohere\_GenAI\_Function.py**: Contains the core function that integrates with the Cohere Generative AI service. This function can be imported into other Python applications, allowing for reusable and modular functionality.
* **Cohere\_GenAI\_Caller.py**: Demonstrates how to call the generate\_response function from Cohere\_GenAI\_Function.py. This is useful for testing or using the function as a part of other Python applications, running it directly via the command line.
* **config.json**: A configuration file that stores values such as the maximum tokens, top variables, and other settings that control the behavior of the AI model. This file can be edited and customized according to your needs.
* **.env**: This file contains environment variables needed for the OCI (Oracle Cloud Infrastructure) configuration, such as compartment ID, tenancy ID, user ID, and paths to necessary files (e.g., private key). This file is **NOT** included in GitHub for security reasons. You must create your own .env file based on the provided env\_example.txt.
* **env\_example.txt**: A sample environment configuration file (.env) that shows the structure and example values for setting up environment variables in your system.

**How to Set Up and Use**

**1. Set Up Your OCI Environment**

Before you can run any of the scripts, you need to configure your Oracle Cloud Infrastructure (OCI) account:

* **Get your OCI credentials**: These include your OCI\_COMPARTMENT\_ID, OCI\_TENANCY\_ID, OCI\_USER\_ID, and OCI\_FINGERPRINT, which you will need to store in your .env file.
* **Generate a private key** for authentication and store it in the specified path.
* **Update your .env file**: Use the env\_example.txt file as a template to create your own .env file with the appropriate values for your environment. Ensure to configure paths correctly and avoid pushing sensitive credentials to GitHub.

**2. Install Required Libraries**

Before running any scripts, ensure that you have all the necessary dependencies installed.

* Python
* OCI CLI and SDK
* Flask
* python-dotenv

**3. Running the Flask API (Cohere\_GenAI\_API.py)**

To run the Flask API and expose it for external calls, execute the following command:

python Cohere\_GenAI\_API.py

Once the API is running, you can send POST requests to http://127.0.0.1:5000/generate with a JSON payload containing the message.

Example request:

**POST to** [**http://127.0.0.1:5000/generate**](http://127.0.0.1:5000/generate)

**Headers:**

**Content-Type = application/json**

**Temperature = 0 to 1**

{

"message": "Are you there?”

}

You will receive a response from the AI model processed by the Cohere Generative AI service.

**4. Using the generate\_response Function (Cohere\_GenAI\_Function.py)**

If you prefer not to use the Flask API and instead want to call the function directly from another Python script, you can import the generate\_response function from Cohere\_GenAI\_Function.py.

For example, if you want to call it from another Python script:

from Cohere\_GenAI\_Function import generate\_response

response = generate\_response("Are you there?")

print(response)

This will call the function directly and return the AI-generated response.

**5. Using the Cohere\_GenAI\_Caller.py Script for Command Line Testing**

If you just want to quickly test the functionality from the command line, you can use Cohere\_GenAI\_Caller.py.

This script allows you to pass a message as a command-line argument:

python Cohere\_GenAI\_Caller.py "Are you there?"

It will call the function and print the response from the AI model directly to the console.

**6. Configuration (config.json)**

The config.json file contains values that you can adjust for your use case. Example content:

{

"max\_tokens": 600,

"frequency\_penalty": 0,

"top\_p": 0.75,

"top\_k": 0

}

Note: temperature is an input parameter on the Flask API and hardcoded in the local function.

These values control the behavior of the AI model. For example, you can adjust the temperature for more random outputs or change the max\_tokens for longer responses.

You can modify this file to suit your needs, and the generate\_response function will read from it.

**Summary of Files**

**Cohere\_GenAI\_API.py:**

* A Flask API that listens for POST requests at /generate and returns the AI-generated response.

**Cohere\_GenAI\_Function.py:**

* Contains the function generate\_response that interacts with Oracle's Cohere Generative AI service. This function can be imported into other applications or used directly.

**Cohere\_GenAI\_Caller.py:**

* A script for calling the generate\_response function from the command line, passing a message and printing the AI response.

**config.json:**

* Stores configuration values like max\_tokens, top variables, and others that control the AI response.

**.env:**

* Stores OCI credentials and configuration for connecting to the Oracle Cloud service. This file is not included in GitHub for security reasons.

**env\_example.txt:**

* A template for creating your .env file with example values. This file is included in GitHub to guide you in setting up your environment variables.

**Final Notes**

* **Security**: Be cautious with sensitive information like your OCI credentials, especially when using version control systems like GitHub. Always use .env files for sensitive configuration and make sure it is added to .gitignore.
* **Testing**: Use Postman or curl to interact with the Flask API for quick testing. You can also import the core function (generate\_response) into your own applications for further usage.