|  |  |
| --- | --- |
| Version number and Status | 1.0 |
| Document Date | 02/27/2024 |
| Security Classification | Insight |
| Author | Insight Team |
| Reviewer | CHD Team |

VULCAN AI CHD

# 1. Introduction

Church & Dwight is an American consumer goods company focusing on personal care, household products, and specialty products.

Business wants a solution that can create a bot using LLM, which will identify the questions asked by business regarding the sales and business related issues.

# 2. Prerequisites

Azure Subscription to access the Cognitive Search, Open AI , Bot Service, Storage Account and Key Vault.

# 3.1 How to run the Vulcan AI Bot?

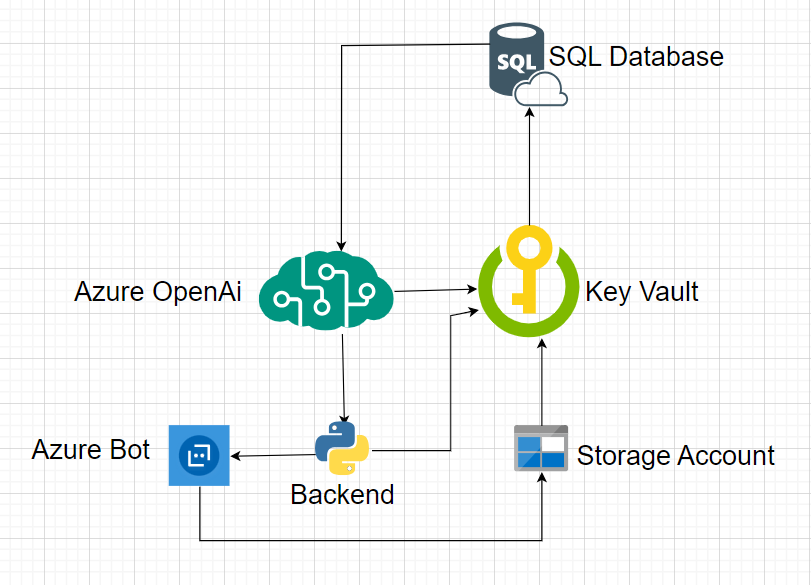
**Repository location:**

: Azure Devops à ChurchDwight à ACE à Repos à Files à az-vulcan-frontend-wapp à staging

* Clone the code base from staging and create the virtual environment using **python -m venv venv** and then activate it using **./venv/Scripts/activate.**
* Navigate to common folder and install requirements.txt using **pip install requirements.txt.** This will help us to install streamlit on which our UI is designed.
* Move out from the common folder using **cd ..** and again navigate to app folder to run the application.
* After navigating to app folder run the command **streamlit run home.py** to run the application on local host URL.

# 3.2 Flow Diagram

* RAG model for DSR Tool



- RAG model for DocSearch Tool

A screenshot of a computer

Description automatically generated

# 3.3 DSR Tool

- **DSR Historical data location:**

: Data à SSMS à az-idgwdev-01 àIGDW à VulcanAI.USDSR\_VulcanAI\_2023

A computer screen shot of a blue screen

Description automatically generated

* **Script location:**

: Azure Devops à ChurchDwight à ACE à Repos à Files à az-vulcan-backend-wapp à staging à app à utils.py ( Class SQLDbTool )

A screenshot of a computer

Description automatically generated

Run **utils script to create the DSR Tool** which is interacting with SQL db using ODBC driver based on user query.

# 3.4 FewShots Implementation

* **Script location:**

: Azure Devops à ChurchDwight à ACE à Repos à Files à az-vulcan-backend-wapp à staging à app à utils.py ( Class SQLDbTool )

A screenshot of a computer

Description automatically generated

Run **utils script to implement fewshots** whichimport the examples from prompt.py file and embedded that using OpenAi embeddings to find the most appropriate match with user query and pass on those examples alongwith prompt.

# 3.5 Tracking the Application/Langchain Logs

We have added logger.info/logging .info to track the application/langchain logs with timestamp and storing them directly to the blob container on azure.

# 3.6 DocSearch Tool (Load Data Into Index)

- **DocSearch Historical data location:**

: Data à Azure Portal à Home àeus2-openaiq-rg à [steus2openai891456313491](https://portal.azure.com/#@churchdwight.onmicrosoft.com/resource/subscriptions/7c7ec81c-5f71-4c06-bb32-981841492099/resourceGroups/eus2-openaiq-rg/providers/Microsoft.Storage/storageAccounts/steus2openai891456313491) à Containers àlucydata à RawFiles

* Clone the code base from staging and open **load\_data\_index.ipyb** from the test folder**.**

To load data into another index by creating a new index, follow these steps:

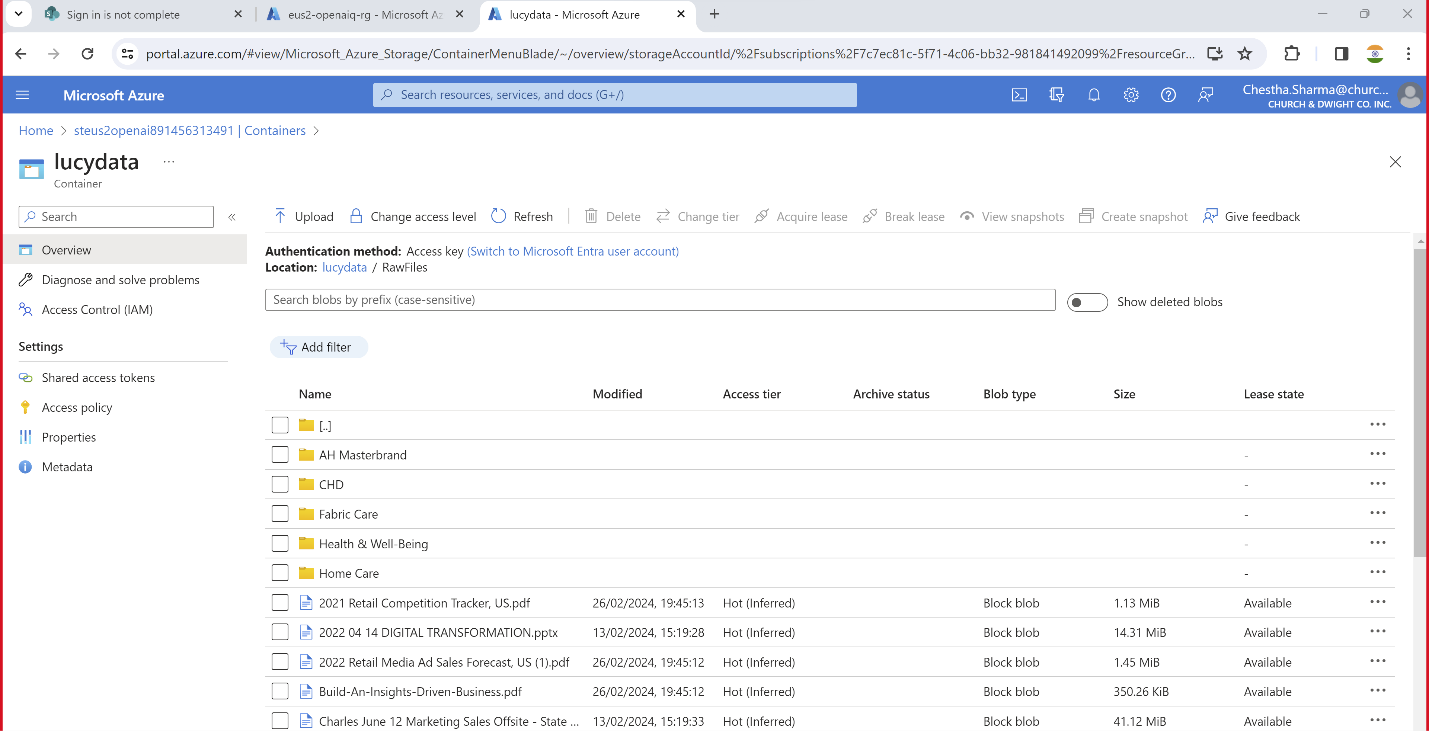
**1. Change Output Folder:** Modify the output folder where we are storing the text files. Update the new\_folder\_name variable to point to the desired folder location.

**2. Define New Index Name:** Specify the name of the new index where we want to load the data. This involves setting up a new index client.

**3. Update Index Loading Process:** Adjust the script to use the new index client and load documents into the newly created index. Ensure that the documents are formatted and processed correctly according to the schema of the new index.

**4. Test and Validate**: Before deploying the changes, thoroughly test the script to ensure that data is loaded into the new index accurately and efficiently.

By following these steps, we can seamlessly load data into a new index with minimal modifications to the existing script.



* **Script location:**

: Azure Devops à ChurchDwight à ACE à Repos à Files à az-vulcan-backend-wapp à staging à test à load\_data\_index.ipynb

A screenshot of a computer

Description automatically generated

Run Script **load\_data\_index to insert** the lucy data of storage account (all PPTs and PDF)into the azure cognitive search index that eventually retrieves the data.

# 3.7 DocSearch Tool ( Implementation on Frontend)

* **Script location:**

: Azure Devops à ChurchDwight à ACE à Repos à Files à az-vulcan-backend-wapp à staging à app à utils.py ( Class DocSearchTool )

**A screenshot of a computer

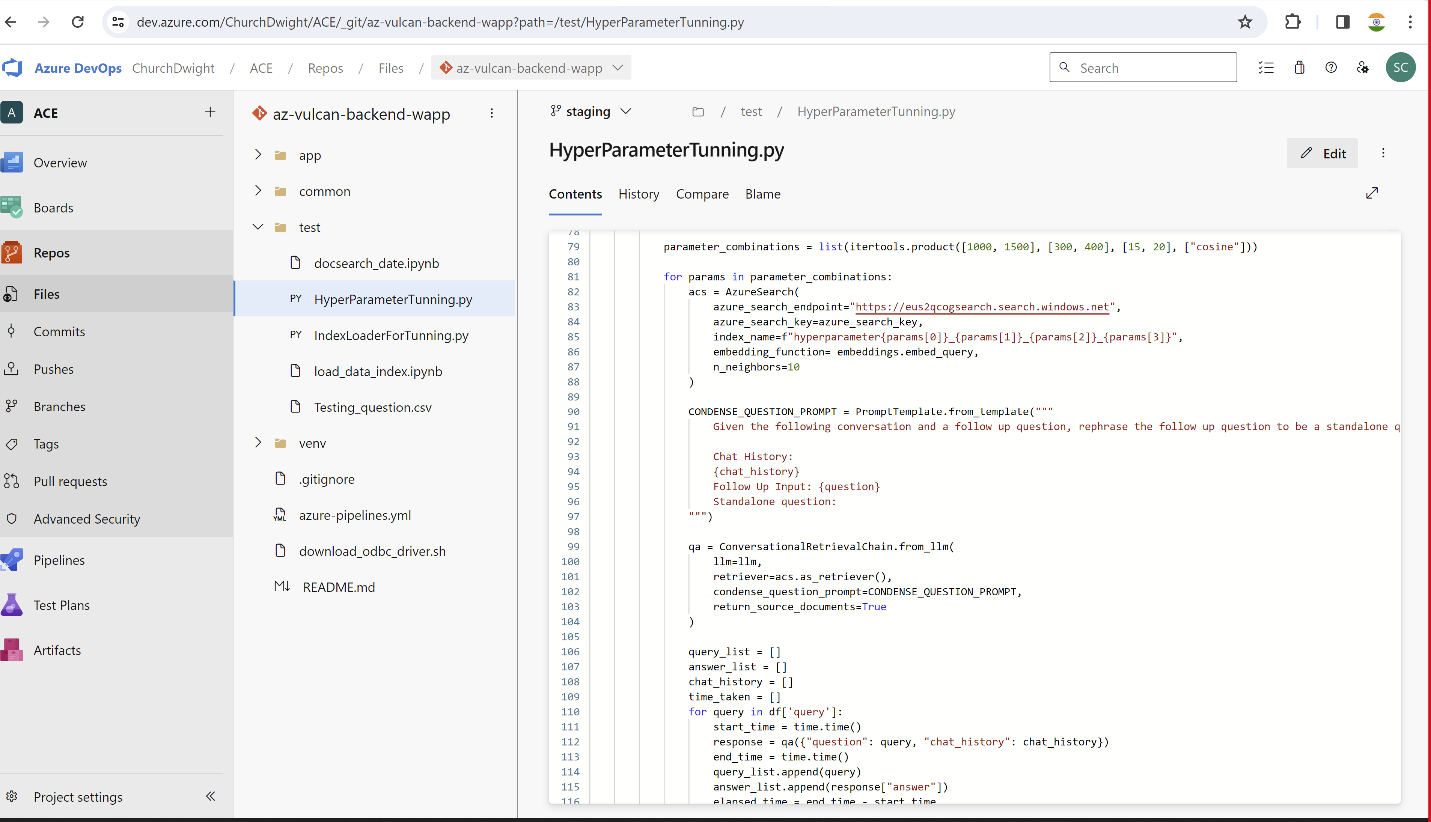
Description automatically generated**

Run **utils script to create the DocSearch Tool** which is hitting the index, queries for the particular document on the basis of keywords and give us response for the same generated by GenAI.

# Hyperparameter Tuning

* **Script location:**

: Azure Devops à ChurchDwight à ACE à Repos à Files à az-vulcan-backend-wapp à staging à test à HyperParameterTunning.py



Run Script **HyperParameterTunning.py** to implement hyperparameter tuning for different parameter combinations such as chunk size, overlap, number of neighbors etc. and created the index corresponding to each combination and generate the response respectively for a sample size of queries and compare it with the benchmark response using embedding and similarity search.

# 3.9 Recommendations / Next Steps

* Data cleaning and data preprocessing such as removing stopwords, normalization, stemming and lemmatization.
* Consider exploring the use of the Llama index as an alternative to the Langchain for analyzing natural language and generating more accurate results. This could potentially improve the accuracy of the results and provide better insights for businesses.
* Implement a data visualization tool to present the results in an interactive and user-friendly format.
* Connection should be made in such a way that we don’t need to write @dsr or @docsearch specifically by making certain changes in the codebase of frontend, backend and their connectivity.
* Conduct user testing to gather feedback and improve the user experience of the application.
* Can be discussed.