# ServiceNow AI Bot

## TABLE OF CONTENTS



## Objectives

The main objective of this project is to answer user queries on ServiceNow's features.

The chatbot gets the query from the user and searches the ServiceNow docs semantically to fetch the relevant answer and return the same to the user.

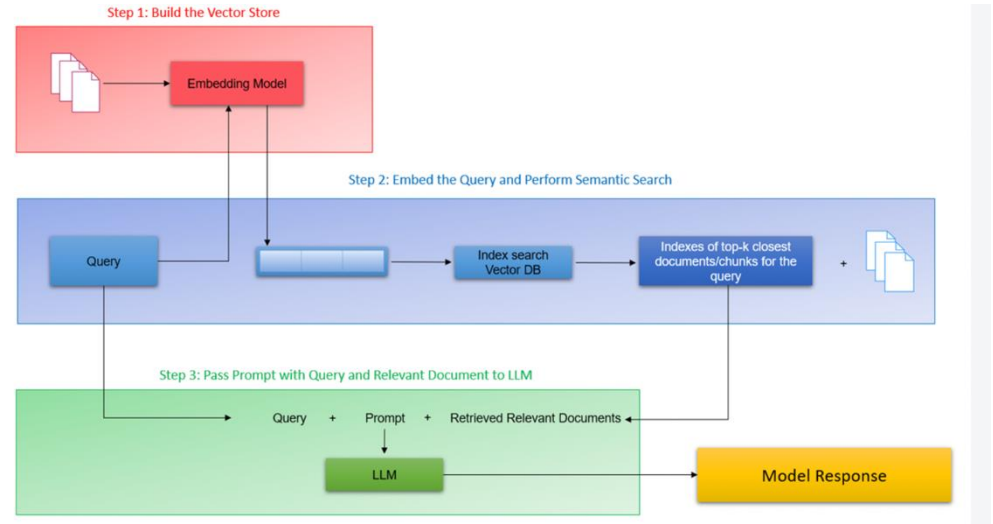
If the user's query doesn't match and not relevant to the bot proper responses will be shown to the user.

## Design

### Retrieve/build the Dataset

1. Downloaded the product docs from ServiceNow official site.
2. Picked aroung 6 docs from a specific product.
3. Used the pdfplumber to generate the dataframe based on the text, tables and other data that the document has
4. The dataframe has been built and it's kept aside for the semantic search
5. Used the RAG based approach to filter and showcase the appropriate results to the user based on the user query.

The design of this project is similar to the HelMate AI were the semantic search works in 3 layers.




RAG - System Design 1

Output dataframe generated from the perquisites layer where the PDFs are read and the chunks are created

```
service_mapping_pdfs_data
```

| | Page No. | Page_Text | Document Name | Text Length | metadata | Metadata |
|---|---|---|---|---|---|---|
| 0 | Page 1 | 20/06/2024, 16:29 Automated Service Suggestion... | Automated Service Suggestions.pdf | 220 | {'Document Name': 'Automated Service Suggestio... | {'Document Name': 'Automated Service Suggestio... |
| 1 | Page 2 | 20/06/2024, 16:29 Automated Service Suggestion... | Automated Service Suggestions.pdf | 25 | {'Document Name': 'Automated Service Suggestio... | {'Document Name': 'Automated Service Suggestio... |
| 2 | Page 1 | 20/06/2024, 16:27 Tag-based discovery in Servi... | Tag-based discovery in Service Mapping.pdf | 628 | {'Document Name': 'Tag-based discovery in Serv... | {'Document Name': 'Tag-based discovery in Serv... |
| 3 | Page 2 | 20/06/2024, 16:27 Tag-based discovery in Servi... | Tag-based discovery in Service Mapping.pdf | 27 | {'Document Name': 'Tag-based discovery in Serv... | {'Document Name': 'Tag-based discovery in Serv... |
| 4 | Page 1 | 20/06/2024, 16:28 Pattern-based discovery in S... | Pattern-based discovery in Service Mapping.pdf | 438 | {'Document Name': 'Pattern-based discovery in ... | {'Document Name': 'Pattern-based discovery in ... |
| 5 | Page 2 | 20/06/2024, 16:28 Pattern-based discovery in S... | Pattern-based discovery in Service Mapping.pdf | 394 | {'Document Name': 'Pattern-based discovery in ... | {'Document Name': 'Pattern-based discovery in ... |
| 6 | Page 1 | 20/06/2024, 16:29 Choose the right method for ... | Choose the right method for discovery and mapp... | 881 | {'Document Name': 'Choose the right method for... | {'Document Name': 'Choose the right method for... |
| 7 | Page 2 | 20/06/2024, 16:29 Choose the right method for ... | Choose the right method for discovery and mapp... | 32 | {'Document Name': 'Choose the right method for... | {'Document Name': 'Choose the right method for... |
| 8 | Page 1 | 20/06/2024, 16:29 Discovery based on Predictiv... | Discovery based on Predictive Intelligence.pdf | 754 | {'Document Name': 'Discovery based on Predicti... | {'Document Name': 'Discovery based on Predicti... |
| 9 | Page 2 | 20/06/2024, 16:29 Discovery based on Predictiv... | Discovery based on Predictive Intelligence.pdf | 27 | {'Document Name': 'Discovery based on Predicti... | {'Document Name': 'Discovery based on Predicti... |
| 10 | Page 1 | 20/06/2024, 16:28 Traffic-based discovery in S... | Traffic-based discovery in Service Mapping.pdf | 840 | {'Document Name': 'Traffic-based discovery in ... | {'Document Name': 'Traffic-based discovery in ... |
| 11 | Page 2 | 20/06/2024, 16:28 Traffic-based discovery in S... | Traffic-based discovery in Service Mapping.pdf | 35 | {'Document Name': 'Traffic-based discovery in ... | {'Document Name': 'Traffic-based discovery in ... |

This AI works across 3 layers and uses RAG

**Layer 1:**

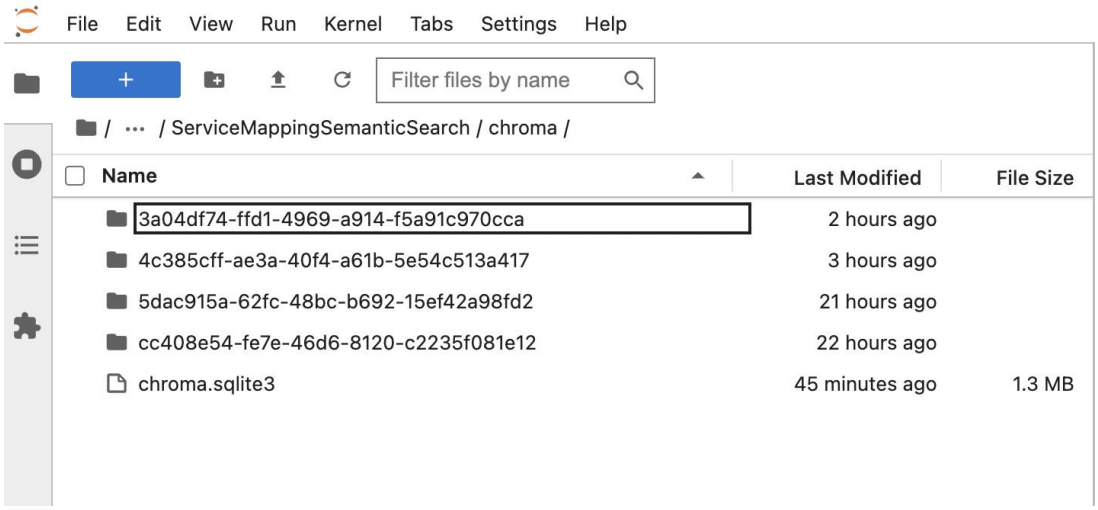
In layer 1, vector embeddings are generated for the documents in the dataframe and same is stored in the chroma database. Below is the sample code to do the same

```
def store_embeddings_in_chroma(self):
    self.generate_data_frame()

    #Layer 1
    document_collection = generate_store_embedding_chromadb(self.document_pdfs_data)
```

Two chroma collections were created to store the documents and to cache the collections based on the query for fast retrieval.

The chroma collections were created at the default location inside the project repo.



**Layer 2:**

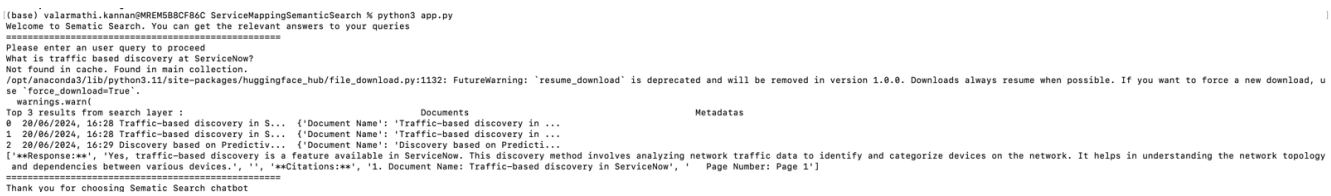
In layer2, the dataframe is filtered based on the user query and re-ranked to filter the top n results.
Below code does the same. The code tries cache to get the results and it it's not found, the main dataframe is searched.

```
def initialize_chatbot(self):
    #Layer 2 - Semantic search with cache

    # Read the user query
    print('Please enter an user query to proceed')
    query = input()

    # Search the Cache collection first
    # Query the collection against the user query and return the top 5 results
    top_3_RAG = do_semantic_search(query)
    print("Top 3 results from search layer :",top_3_RAG)
```

The output received from the layer 2 are 3 document chunks based on relevance




**Layer 3:**

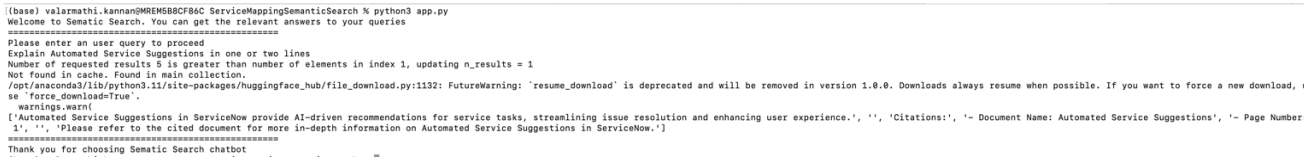
In layer 3, the top data filtered from original dataframe along with the user query has been passed through a chat completion model and appropriate results were retrieved.

Below are the sample outputs from this layer in different test cases

1. User querying chatbot with a irrelevant query on ServiceNow


Here is the response received from bot
I couldn't find the information related to your query. Sorry for the inconvenience

2. User querying chatbot with a relevant query on ServiceNow



Here for the above question a relevant 1-2 answers had been retrieved from the doc and citations were also provided to the user.

3. User querying chatbot with a relevant query on ServiceNow which is available in the cache

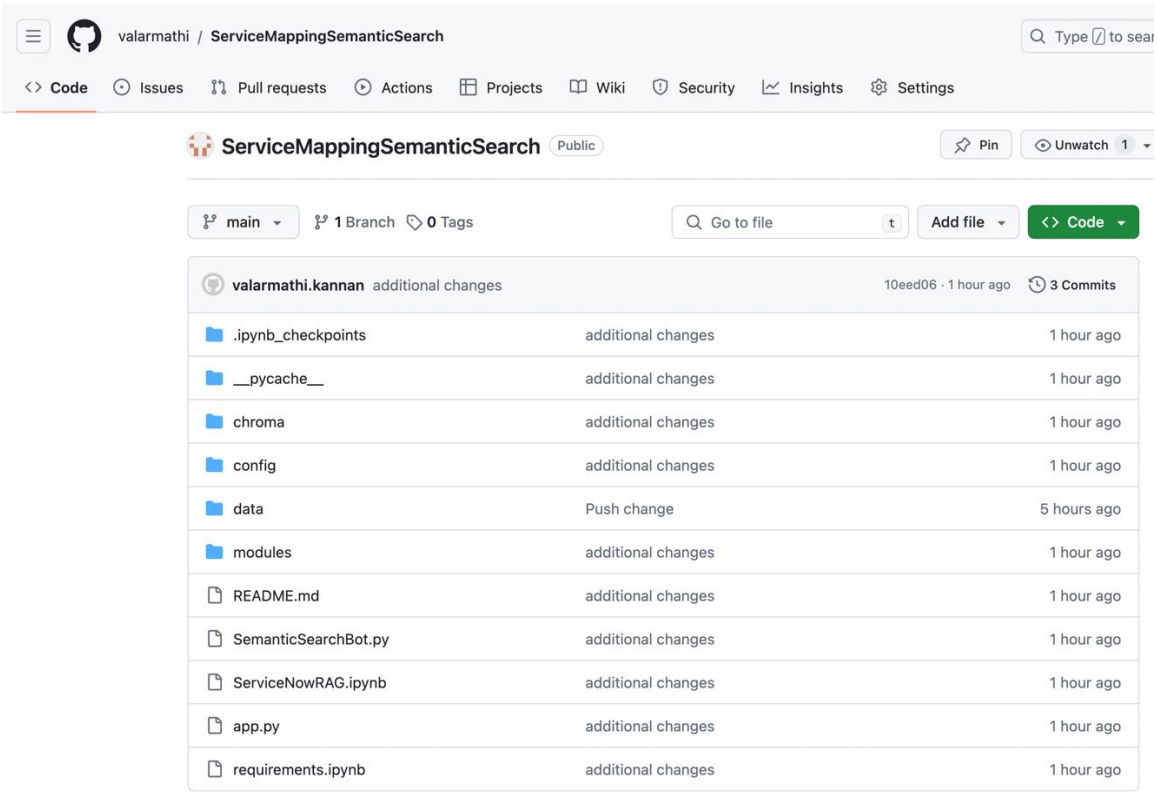
```
(base) valarmathi.kannan@MREM5B8CF86C ServiceMappingSemanticSearch % python3 app.py
Welcome to Sematic Search. You can get the relevant answers to your queries
==================================================
Please enter an user query to proceed
What is Automated Service Suggestions?
Number of requested results 5 is greater than number of elements in index 2, updating n_results = 2
Found in cache!
/opt/anaconda3/lib/python3.11/site-packages/huggingface_hub/file_download.py:1132: FutureWarning: `resume_download` is deprecated and will be removed in version 1.0.0. Downloads always resume when possible. If you want to force a new download, u
se `force_download=True`.
  warnings.warn(
['Automated Service Suggestions in ServiceNow refer to intelligent recommendations provided by the system to users based on patterns, historical data, and machine learning algorithms. These suggestions aim to streamline and enhance service deliv
ery by recommending relevant actions, solutions, or next steps to users automatically.', '', 'Unfortunately, the search results do not contain detailed information about Automated Service Suggestions. However, to learn more about this feature, I
 recommend reviewing the following documents from the ServiceNow product documentation:', '', '1. Document Name: Automated Service Suggestions', '2. Document Name: Automated Service Suggestions', '', 'These documents likely provide detailed insi
ghts into how Automated Service Suggestions work, their benefits, and how users can leverage them to improve service efficiency and effectiveness. Please refer to these documents for a comprehensive understanding of Automated Service Suggestions
 in ServiceNow.', '', "I couldn't find the information related to your query. Sorry for the inconvenience.", '', '**Citations:**', '1. Document Name: Automated Service Suggestions, Page Number: N/A', '2. Document Name: Automated Service Suggesti
ons, Page Number: N/A']
==================================================
Thank you for choosing Sematic Search chatbot
```

When user quiried the bot with the similar sematic meaning, the bot is able to retrieve the answer from the cache results and the same is showcased to the user.

## Implementation

The code repository for this bot can be seen here
https://github.com/valarmathi/ServiceMappingSemanticSearch

The project is modularized and the needed utilities are placed in the appropriate directory
Here is the code structure




All the prerequisites had been placed in Requirements.ipynb and on the 1st deployment one can run the cells to install the needed packages.

**Challenges**

1. Since I know the context of the documents that I retrieved for sematic searching, it has been bit easy for me.
2. Initially I was confused with the different layers involved in this RAG processing. But, I got a clear understanding by modularizing and going step-by-step.
3. The chat completion model is not able to understand the page number field and it was giving page number unavailable and not found in the metadata multiple times. I had to change the prompt multiple time to showcase the page number field in the citations.

```
4. Use the Metadatas columns in the dataframe to retrieve and cite the document name(s) and Page No as citation.
You should get the page no field and show relevant text eg:- Page 1 from the metadata.
This page no. value is mandatory which you cannot skip and you can find the same from metadata.
```

4. I had to also add chain of thought in my user query to get the appropriate response from the user.
5. Initially I was always getting the results from the main document collection instead of cache collection though the same query is issues. Finally understood that the cache collection should have been queried before getting the actual results from the document collection.
6. I was not able to create a chroma database with the user-defined name. It was always created with the default name chroma.
7. Openai_key_path is not being picked up by chroma collection.
8. The api secret key somehow got cleaned up multiple times and I had to recreate the keys thrice.

**Lessons learned**

1. Understood on how the realtime searching is performed semantically.
2. Writing a better prompt to get the better results.
3. Modularizing the code to avoid confusions.
4. Learnt most of the python features
5. Explored different GenAI models and used various properties of it.
6. Understood how the usage and billing is calculated by OpenAI
7. Clearly understood different layers in RAG

The code repository for this bot can be seen here and it's a public one
https://github.com/valarmathi/ServiceMappingSemanticSearch