# Retrieval-Augmented Generation RAG

1. To Implement the Retrieval-Augmented Generation, I first transform the user's natural language query into a vector using embeddings in the api of Langchain . This step is crucial as it translates the human-readable text into a numerical form that can be processed by the computer.

2. Once the query is vectorized, the application calculates the cosine similarity between this query vector and the vectors in a pre-built database of document embeddings. Cosine similarity is a metric used to measure how similar the documents are to the query in the vector space.

3. Based on the similarity scores, the system retrieves the most relevant chunk from the indexed documents. This step ensures that the language model has access to the most pertinent information before generating an answer.

4. Finally, the application displays the original documents from which the information was retrieved. This feature helps users see the source of the information and provides additional context.

# Document Indexing

PDF documents are loaded from a specified directory using the PyPDFDirectoryLoader class from the LangChain library. The loaded documents are split into manageable chunks using the RecursiveCharacterTextSplitter class. This splitter divides the documents into smaller segments based on a specified chunk size and overlap, making it easier to handle and store them. The

document chunks are then stored in the Chroma database. Before storage, unique IDs are assigned to each chunk to avoid duplication and ensure easy identification. Unique IDs are generated based on the source and page number of each chunk. If multiple chunks originate from the same page, an index is used to differentiate them.

# Process User Question

I first prompt the user to input their question. It checks if the user provided an empty question and offers a fallback response, prompting the user to try again. The user-provided question is processed by displaying it and initializing the necessary components for embedding and database interaction.An embedding function is initialized using get_embedding_function(). The Chroma database is accessed to perform a similarity search based on the user's question. The top 5 relevant contexts are retrieved. The search results are compiled into a single context string. If no relevant context is found, the script informs the user and prompts them to ask a different question. A prompt template is prepared using the retrieved context and the user's question. This template is then formatted accordingly.The formatted prompt is sent to the Ollama model (mistral) to generate a response. Error handling is in place to catch any issues during this process. The response generated by the AI model is formatted and displayed along with the source documents from the Chroma database that were used to compile the context.

# Voice Interaction

In this session, I intend to capture user queries via voice, processing the queries to generate appropriate answers, and then responding back to the user with synthesized speech. First of all, I use the speech_recognition library to capture voice input from the user. This library convert the captured audio into text using a speech recognition service (e.g., Google Web Speech API). Use the pyttsx3 library to convert the text response back into speech.

# Error Handling

1. Check for Empty User Input: If the user does not provide any input (user_question is empty), the script notifies the user and exits early without processing further.
2. Handle No Search Results: If the chroma_db.similarity_search_with_score method returns no results, the script informs the user that no relevant context was found. Check for Empty Context: Before generating the prompt, the script checks if the compiled_context is effectively empty (no useful content) and advises the user to try a different question.
3. Error Handling in Response Generation: Adds a try-except block around the model invocation to catch and handle any errors that occur when generating a response, which prevents the script from crashing and provides feedback to the user.

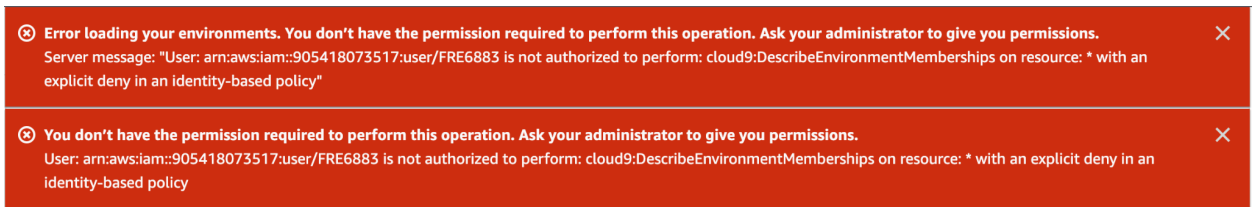
# Challenges faced

When I run the application in the AWS terminal, it requires me to install Python 3.9. Initially, I failed to install it because it required Homebrew. After installing Homebrew, I used `brew link

python@3.9` to link Python 3.9. However, after completing all these installation processes, I encountered an issue where the AWS credentials I'm using do not have the necessary permissions to perform the `InvokeModel` operation. Specifically, the error message indicates that my credentials do not have access to the model I am trying to invoke.

To resolve this, I went to the IAM policy to allow the `sagemaker:InvokeEndpoint` action. Even though I added this permission, the error message indicated that my IAM user FRE6883 is explicitly denied permission to perform the `bedrock:InvokeModel` action on the resource `arn:aws:bedrock:us-east-1::foundation-model/amazon.titan-embed-text-v1`. I then searched for statements with `"Effect": "Deny"`.

After these steps, my Cloud 9 environment did not function as expected, and I could not see any instances I had created before.




I couldn't fix this problem, so I decided to stop using AWS and run the application on my local device instead. To install Ollama, I followed the documentation from their website: [Ollama Blog on Embedding Models](https://ollama.com/blog/embedding-models). Here are the steps I followed:

1. ollama pull mxbai-embed-large

2. ollama pull mistral

3. pip install ollama chromadb

4. python query_data.py

```
(env) (base) xinrancheng@MacBook-Air-5 linvest_source_code %  python process_use
r_questions.py

Hello! What's your question? what is business ethics
```

```
(env) (base) xinrancheng@MacBook-Air-5 linvest_source_code %  python process_use
r_questions.py

Hello! What's your question? what is business ethics
Got it! Your question is:   what is business ethics
Human:
Answer the question based on the above context:  what is business ethics

Here is the response for your question
Response:  Business ethics refers to the moral principles and values that busine
sses, organizations, and their employees are expected to uphold in their operati
ons and interactions with stakeholders. These include principles such as honesty
, fairness, respect for others, responsibility, and respect for the law. Busines
s ethics aims to promote trust between businesses and society by ensuring that b
usiness practices are transparent, equitable, and socially responsible. It also
helps organizations manage potential conflicts between personal values, organiza
tional goals, and societal expectations.
```

In the process of voice interaction, I wrote a source code of voice interaction. However, during the integration process, I encountered a persistent error which prevented the application from running successfully. This error occurs despite ensuring that pyobjc is installed in the virtual environment. Unfortunately, I am not able to solve this issue.

```
    class NSSpeechDriver(NSObject):
  File "/Users/xinrancheng/Desktop/voice_version/venv/lib/python3.11/site-packages/py
ttsx3/drivers/nsss.py", line 13, in NSSpeechDriver
    @objc.python_method
     ^^^^
NameError: name 'objc' is not defined. Did you mean: 'object'?
```

# Improvements

In the process of voice interaction, exploring alternative text-to-speech libraries that are more compatible with macOS. It might resolve the current issues with pyttsx3. Additionally, implementing a fallback mechanism that defaults to text output if the voice synthesis fails would ensure a more robust user experience. Expanding the range of supported languages for both voice recognition and speech synthesis could make the application more accessible to a broader audience. Lastly, incorporating advanced natural language processing techniques to better understand and respond to user queries could significantly improve the accuracy and relevance of the provided answers. These enhancements would collectively elevate the overall functionality and user satisfaction of the voice-enabled application.