**Conversational Bot powered by LLM**

**OCR:**

First step is to convert the PDF catalogue of AMM, EMM, CMM and DMM connector into LLM understandable markdown format. LLMs cannot understand complex structures of PDFs like titles of a block, paragraphs, tables, lists and more. Transforming this data into Markdowns and JSON will make it easier for LLM to know the context and relationship between the different text elements. Here I used Llama-parse which had a pipeline to convert PDF data into markdown using multiple elements like OCR and contextual reconstruction of data.

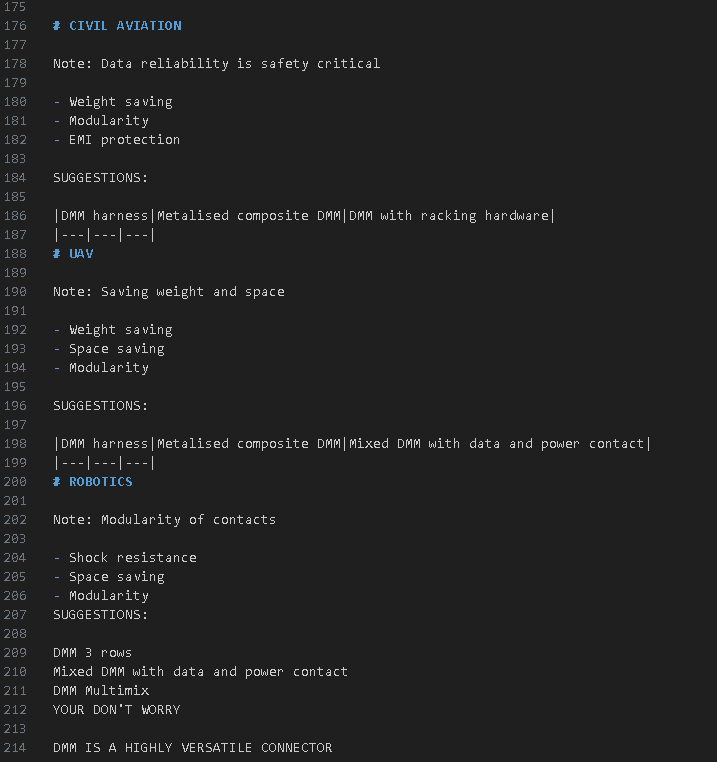
Code snippet:

A screen shot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generatedOutput:



Load data the parsed data:

Here we load the parsed data onto a variable for further processing. We also add more structured which were not parsed by the Llama parse or any other additional data. After this we combine the parsed and added data into a single corpus of data. After this the variable is passed for preprocessing.

A screen shot of a computer program

Description automatically generatedCode snippet:

A black background with white and red text

Description automatically generatedOutput:

Node Parsing:

Searching and breaking down the huge document into smaller chunks of information. The chunking is done based on nodes logic or context of information.

Code snippet:  
A screen shot of a computer

Description automatically generated

Output:

A black background with white and red lines

Description automatically generated

Creating vector embeddings:

In this step the nodes are converted into numbers using an embedding model which in this case is "nomic-embed-text-v1" (opted for this model because of its high dimensions). After the embedding model converts the nodes into numbers (vectors) the vectors are stored into a vector database which is “VectorStoreIndex” in the code.

Code:

A screen shot of a computer program

Description automatically generated

A screen shot of a computer

Description automatically generatedOutput:

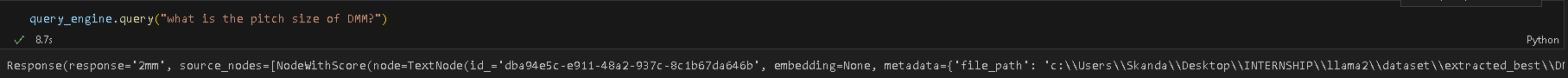
Creating a query engine to access this vector database:

LLM cannot directly retrieve the relevant data from the vector database. This is where a retriever or query engine comes in. These retrievers take in query embedding from the LLM access the vector database and compares the query embeddings with the embeddings in the vector database to find the more suitable or similar embeddings (powered by the embeddings model) and outputs a list of top similar embeddings.

Code snippet:

A black screen with white text

Description automatically generated

Output:  
Creating retriever tool:

The Chatbot will be connected to the internet as well. For this, we define the retriever as a tool in a tool set for the LLM to use whenever necessary.

Code:

A screen shot of a computer code

Description automatically generated

Create tool set:

We have a retriever tool already, now we need a tool to connect the LLM with the internet as well to fetch information. To make this possible we define a tool to access the internet for additional up-to-date information using web searches. In this code we use Tavily an LLM searching tool. We add this tool to the tool set as well.

A screenshot of a computer program

Description automatically generatedCode:

Conversational memory:

Context must be maintained for us to ask progressive questions to the LLM. For this we use session-based memory. Here we define an empty dictionary with key as the session ID and the value as the conversation. The conversation is split into two human message to store user’s responses and the AI message to store the LLM’ s responses.

Code:

A screen shot of a computer program

Description automatically generated

Output:



Create the RAG Agent:

We need the LLM to decide what tool to use and when to use this tool by thinking on its own. This can be done by creating an agent with specific instructions on deciding when to access the tools and to choose the right tool. This instruction is given in the prompt “intern/test”. The agent executer is used to initialize the agent and give it tools and chat history now the workflow is ready

A screen shot of a computer program

Description automatically generated

Prompt template:

A screenshot of a black screen

Description automatically generatedThe prompt is carefully crafted to use the tools and think about whether to use a tool or just answer using the LLM. The LLM is directed to ask progressive questions to shortlist the right connector and ask the right questions to the user based on the conversation tone and context.

[LangSmith (langchain.com)](https://smith.langchain.com/hub/intern/test)

Main loop:

The final loop which keeps the agent executor running infinitely until a stop word is used. We can also define session ID here

Code:

A screenshot of a computer program

Description automatically generated

Output:

A black background with green text

Description automatically generated