Design and Implementation of a GenAI-Powered Data Pipeline with Langflow and AstraDB

The goal of this project is to design and implement a cloud-based GenAI-powered data pipeline using Langflow hosted on the Datastax AI Platform and AstraDB as a vector database. The system is designed to:

- Ingest unstructured text data (e.g., chat logs, customer support logs, or product reviews).
- Vectorize the data using an embedding model.
- Store and manage the vectorized data in AstraDB.
- Implement a Retrieval-Augmented Generation (RAG) workflow to generate contextual responses to user queries.

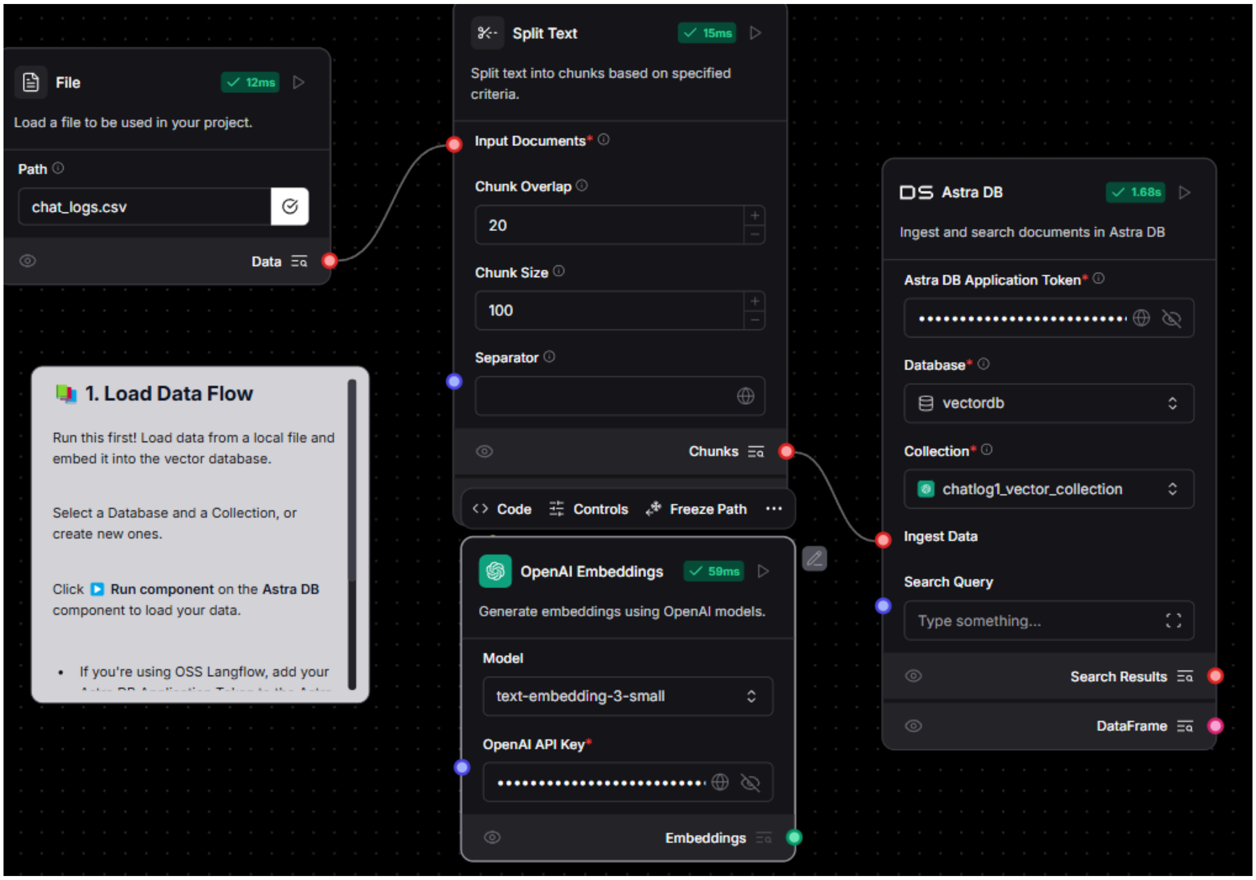
Key architectural decisions

1. **How to Load the Data:**

The first step to building our pipeline is loading a local file and embedding it into the vector database. This can be done in three ways:

1. Insert directly into AstraDB
2. Insert via python script
3. **Insert via Lang Flow components**

For this exercise, I accomplished the data load in all three ways. However, using langflow components was the straight line method to load data via input.



Workflow: Loading data source into Astra DB

In this flow, we load the data source using a File Load component, which splits the content into smaller chunks of size 100 with an overlap of 20 to preserve context between segments. Using smaller chunk sizes improves vector search accuracy and prevents the need to pass the entire document, optimizing the retrieval process.

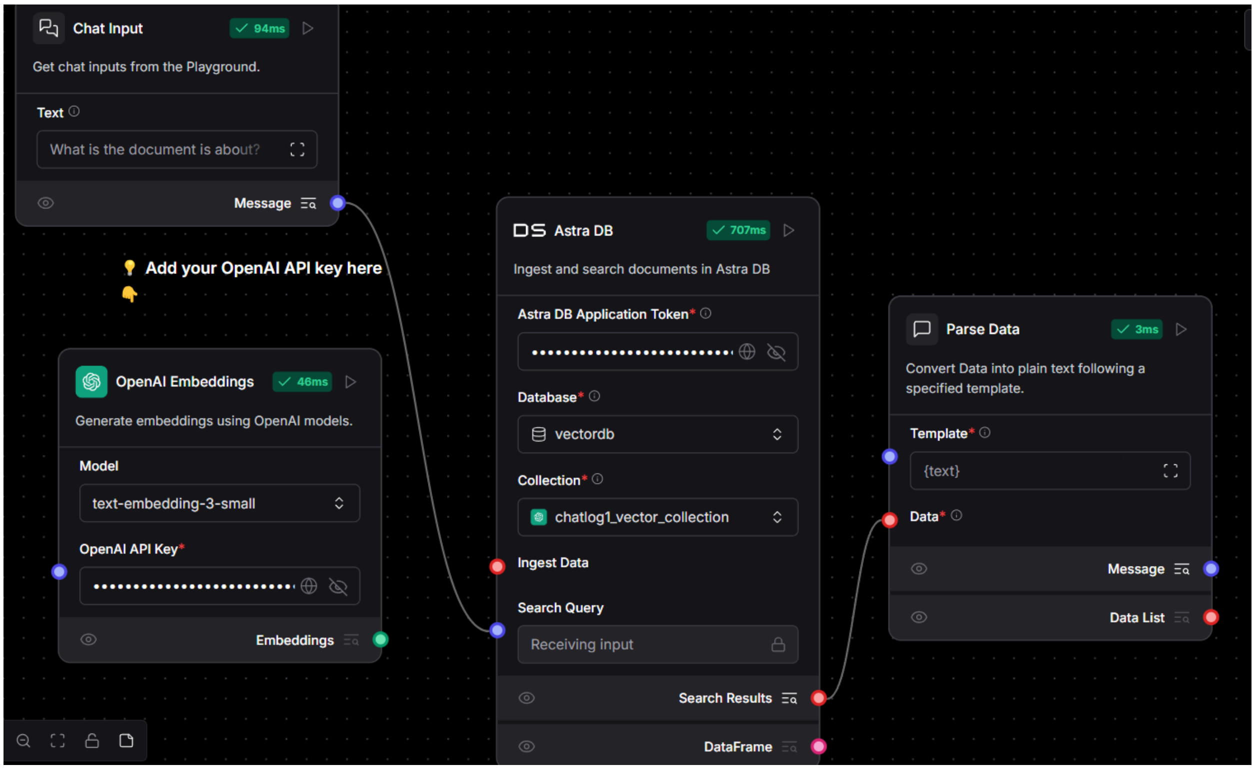
For vectorization, I chose the text-embedding-3-small model from OpenAI due to its simplicity, ease of integration, and reliable performance with my small dataset. The text chunks are transformed into high-dimensional vector representations using this model. Finally, the AstraDB component loads the vectorized data into the selected database and collection, making it ready for efficient retrieval.

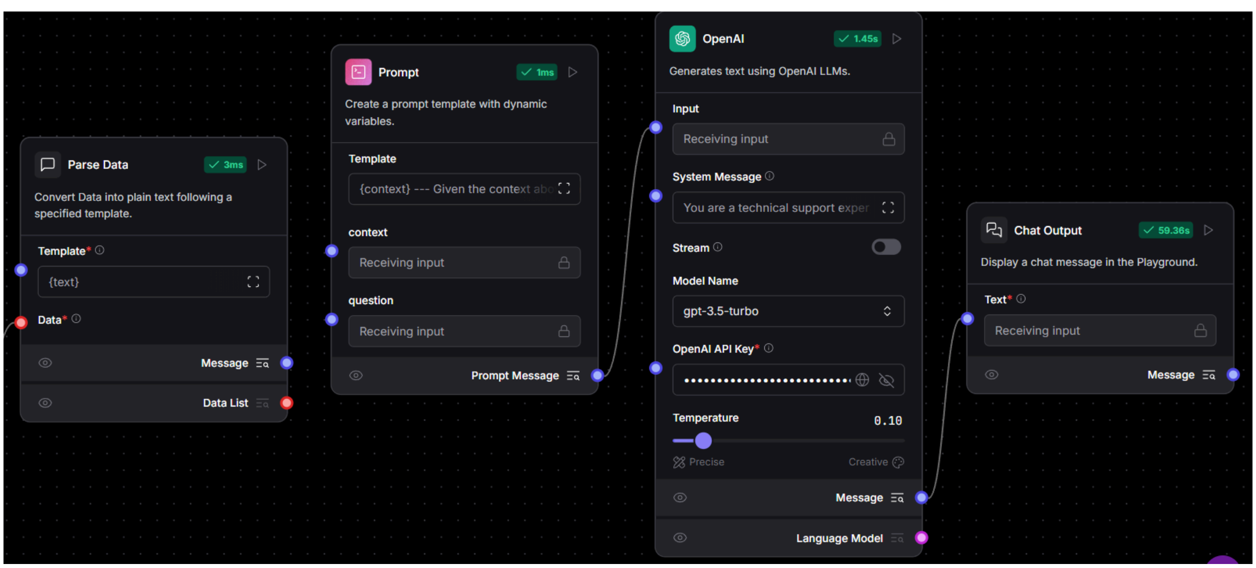
2. How to Build a Retriever Flow: LLM output based on input data and user query

The retriever flow begins by taking the user's input from the Playground and passing it to AstraDB via OpenAI's vector embeddings for similarity search. The retrieved results are parsed into plain text for better formatting.

Next, the Prompt Construction step combines the retrieved context with the user's query in a structured template, ensuring that the LLM has sufficient context before generating a response. This prompt is then sent to OpenAI's gpt-4-turbo model, configured with a system message that guides it to behave as a technical support agent. A low temperature setting ensures deterministic and consistent responses.

The final step involves generating and displaying the LLM's response in the Playground, delivering a context-aware and accurate output.





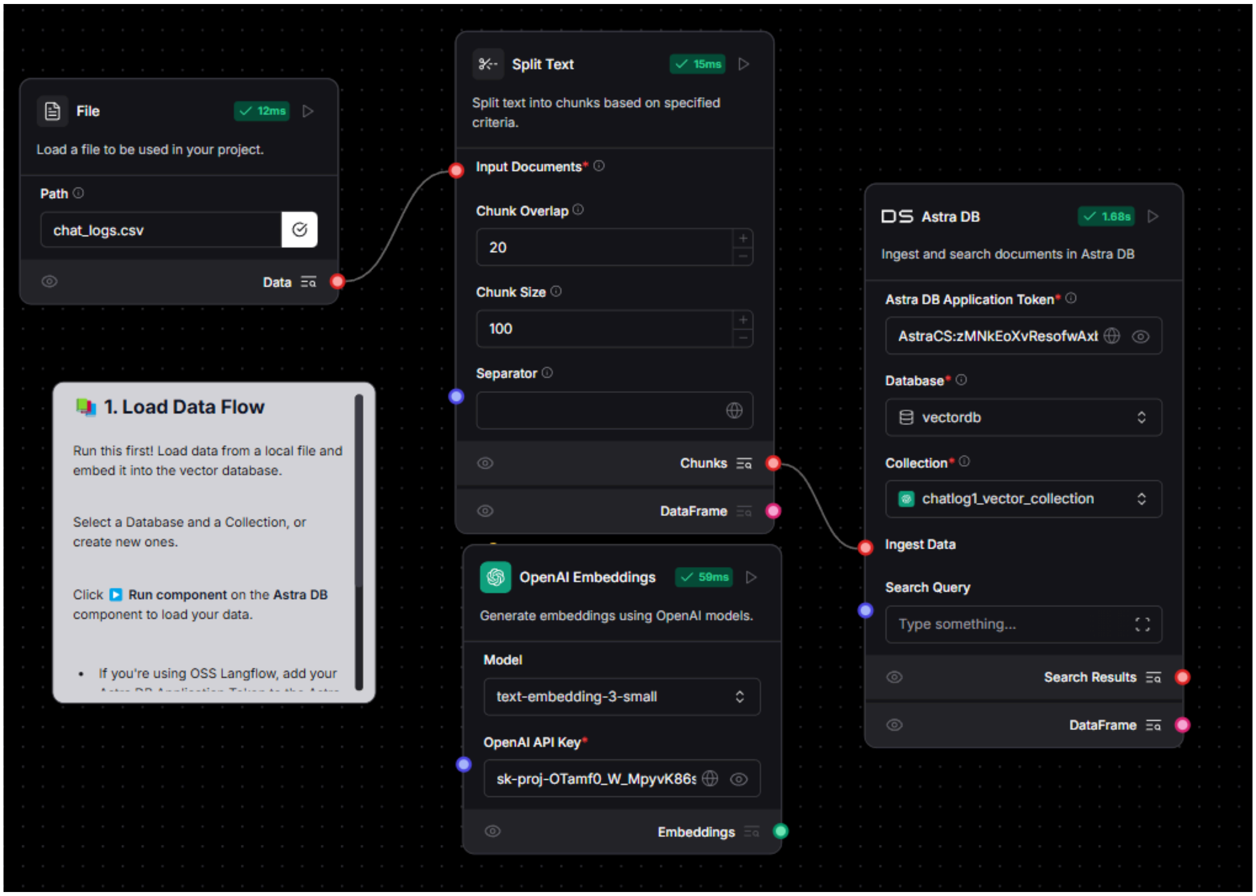
Technology Choices:
Langflow:

Provides an intuitive drag-and-drop interface to design and execute machine learning pipelines. It seamlessly integrates various components required for tasks such as file loading, text splitting, LLM embedding generation, and database interaction with vector databases like AstraDB. This modular approach simplifies the process of building and deploying AI-powered workflows.

AstraDB:

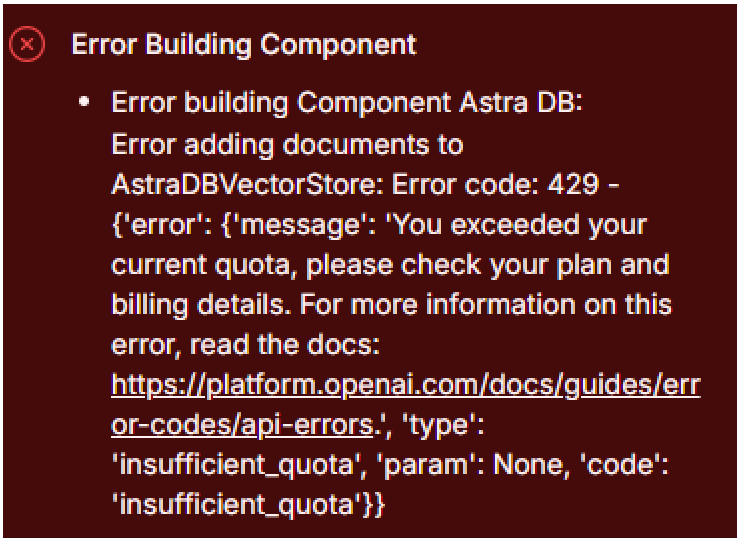
Highly scalable NoSQL and vector database designed specifically for generative AI applications. It efficiently retrieves relevant information for retrieval-augmented generation (RAG) pipelines by extracting context from sources such as PDF documents and injecting that context into the prompt along with the user query. Using AstraDB's vector search capabilities, we can quickly identify and retrieve information that is semantically relevant to the user's input, enhancing the accuracy of the responses.

RAG:
Improves the quality of responses by dynamically incorporating relevant document information into the prompt of an LLM. When a user submits a query, AstraDB retrieves the most relevant data, which is then injected into the LLM's prompt. This enriched context enables the model to generate more accurate, context-aware, and tailored responses, improving the overall user experience.



Challenges and solutions.

1). OpenAI API Key for Vectorized Embedding



I initially assumed that my new OpenAI account included $5 worth of credits for API usage, as was previously the case. However, I later realized that the policy had changed, and new accounts now require purchasing a minimum of $5 in credits to use the API. To explore alternatives, I considered using the Hugging Face model BAAI/bge-large-en-v1.5. However, I discovered that using Hugging Face's embedding models requires both an API key and dedicated inference endpoints for deployment. Given that I would need to purchase credits for API usage either way, I ultimately decided to proceed with OpenAI for simplicity and reliability.

2). Python script for data load

Although I successfully connected to AstraDB and inserted data into my collection, embedding the records for vectorized search proved a challenging and long process in python. To streamline this process, I opted to use LangFlow after resolving my OpenAI API key issue. This approach provided a more efficient and manageable workflow for embedding and loading data.

Screenshots of results.

User

What is the document is about?

AI gpt-3.5-turbo

The document appears to be a log of customer inquiries related to a company's policies and services, specifically regarding return policy, updating profile information, international shipping, and loyalty programs. The entries include ticket numbers, timestamps, and the status of the inquiries (resolved, unresolved, pending).

User

where is my order?

AI gpt-3.5-turbo

For order C19301, the status is pending. For order C30341, the status is also pending. For order C15691, the issue was resolved and the order may have been delayed. For order C17142, the issue is unresolved and the order may be delayed.

User

I forgot my password

User

I forgot my password

AI gpt-3.5-turbo

To reset your password, please follow these steps:

1. Go to the login page of the website.
2. Look for the "Forgot Password" or "Reset Password" link.
3. Click on that link and follow the instructions to reset your password.
4. You may need to verify your identity through email or security questions.
5. Once verified, you can create a new password for your account.

If you encounter any issues during this process, please let me know so I can assist you further.

User

I don't live at the location anymore, I want to change my address

AI gpt-3.5-turbo

To change your delivery address, please provide me with your order number so I can assist you further.

Future improvements.

Currently, I am using smaller OpenAI models such as text-embedding-small for embedding generation. To enhance retrieval accuracy, I plan to switch to a larger model that offers improved performance. Additionally, response times can be slow when user input does not exactly match the stored keywords. To address this, I intend to implement a hybrid search approach that combines vector similarity with keyword-based filtering. By modifying the AstraDB search queries to incorporate both methods, we can significantly improve the accuracy of query matches, especially for inputs with similar meanings.