

LLM Retrieval Augmented Generation Application

RAG Customer Support LLM Application

Objective of the practical is to create an RAG Customer Support LLM Chabot application to answer customer questions without Hallucinations.

What are Hallucinations?

LLMs like GPT or other transformer-based models generate text by predicting the next most likely word based on patterns learned from massive text datasets.

However, they do not inherently verify facts — they rely on statistical associations rather than a true understanding of reality.

As a result, when asked a question or given an incomplete prompt, an LLM may:

- Invent facts, numbers, or references that don't exist.
- Confidently present incorrect information in a coherent and authoritative tone.
- Fill in gaps with “best guesses” that are linguistically logical but factually wrong.

What is Retrieval-Augmented Generation (RAG)?

Retrieval-Augmented Generation (RAG) is an advanced technique that enhances Large Language Models (LLMs) by combining **information retrieval** with **text generation**.

Instead of relying solely on the model's internal knowledge, RAG retrieves relevant external documents or data from a **knowledge base or vector database** and uses them to generate more accurate, up-to-date, and context-specific responses.

Key Components

Retriever:

Searches external data sources (e.g., documents, databases, or embeddings) to find the most relevant information for the user query.

Generator (LLM):

Uses the retrieved information to produce a coherent, factual, and context-aware response.

Benefits

Reduces hallucinations by grounding responses in real data

Provides access to domain-specific or private knowledge

Improves factual accuracy and contextual relevance

Example

When a user asks a question about company policies, a RAG system retrieves related documents from the company's internal database and then generates an answer based on that content — ensuring the response is accurate and source-based.

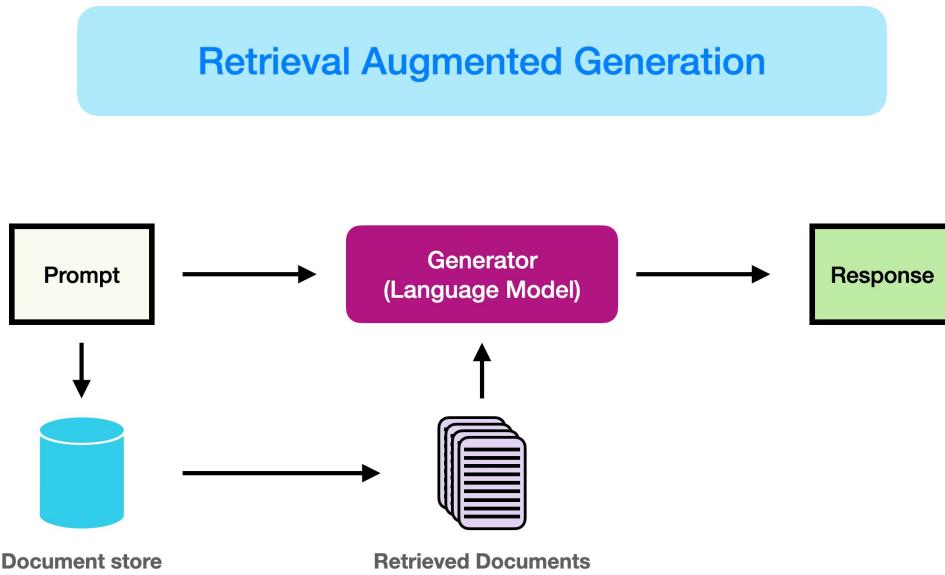


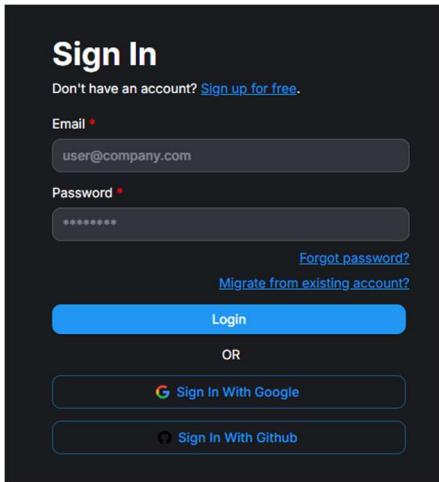
Image source: promptingguide.ai

1. Flowise Website

Goto this Website : <https://cloud.flowiseai.com/signin>

Sign up for free->complete signup of a free Account

Sign In with the new account created



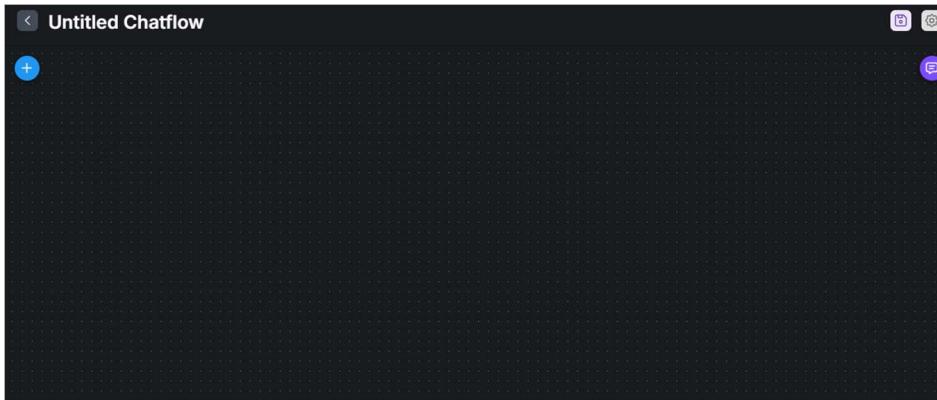
2. Flowise Development Interface

The image shows the Flowise development interface. The top navigation bar includes the Flowise logo, a star icon with the number 42,696, and dropdown menus for "nyprojectwork's Organization" and "Default Workspace". There is also an "Upgrade" button and a settings gear icon. The main content area has a dark background. On the left is a sidebar with various categories: Chatflows (selected), Agentflows, Executions, Assistants, Marketplaces, Tools, Credentials, Variables, API Keys, and Document Stores. Below this is another section titled "Evaluations" with sub-options: Datasets, Evaluators, and Evaluations. The main workspace is titled "Chatflows" and contains the sub-instruction "Build single-agent systems, chatbots and simple LLM flows". It features a search bar with the placeholder "Search Name or Category [Ctrl]", a "No Chatflows Yet" message, and a "Add New" button. There is also a small decorative illustration of a computer monitor with two cartoon characters.

3. Chatflows

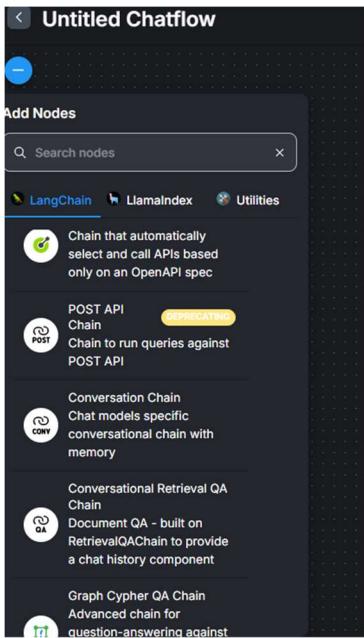
Select Chatflows- > [+Add New](#)

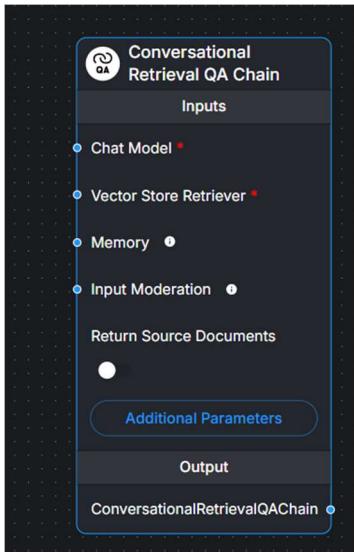
You will see an empty canvas with Untitled Chatflow



4. Build a ChatBot application

Select Add Nodes -> LangChain ->Chains ->**Conversational Retrieval QA Chain**, then drag into the canvas.

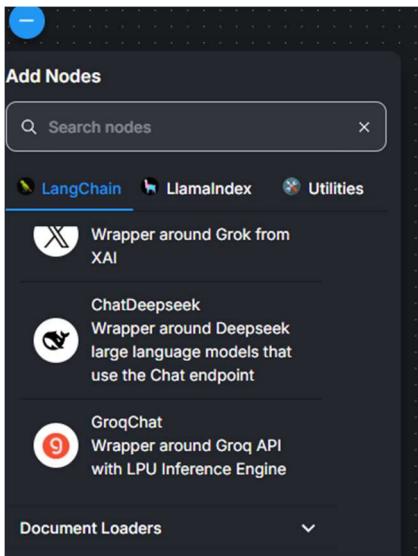




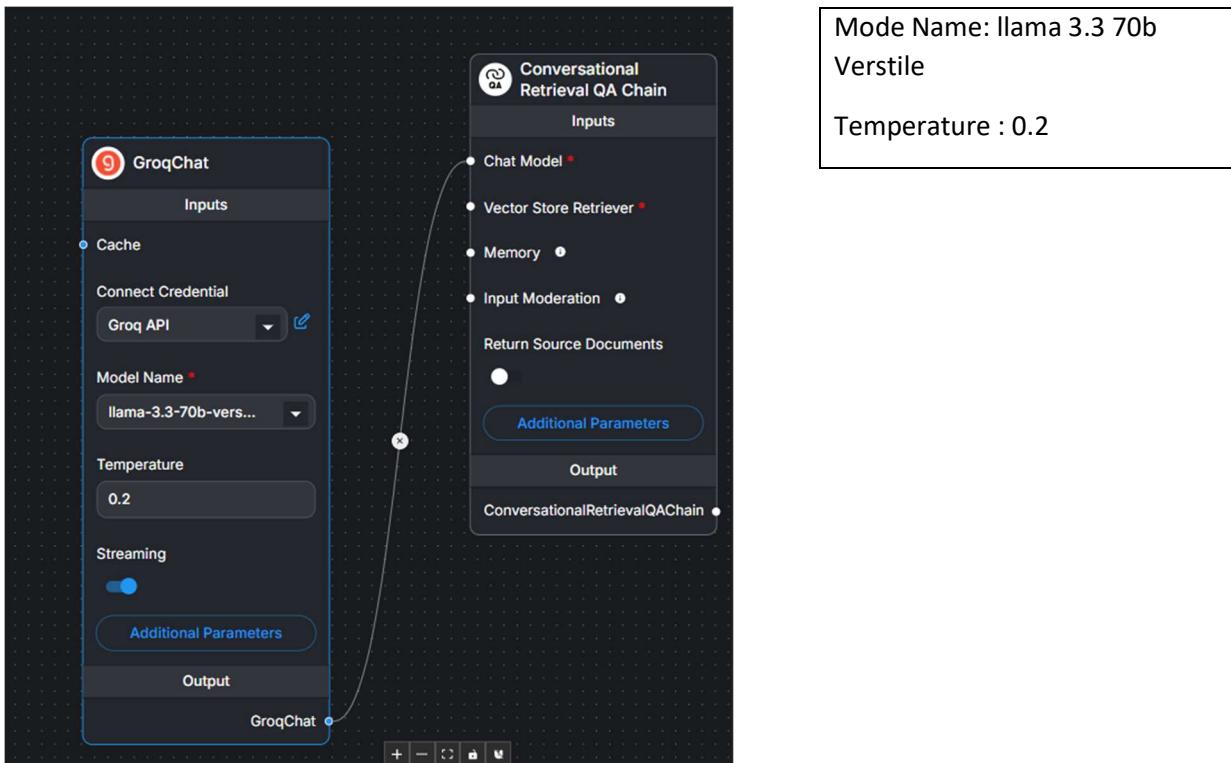
Any items with red dot are mandatory.

5. Add LLM Chat Model to Conversation Chain

Select Add Nodes->LangChain->Chat Model-> **GroqChat**, drag to the canvas. Connect it to the Conversation Chain

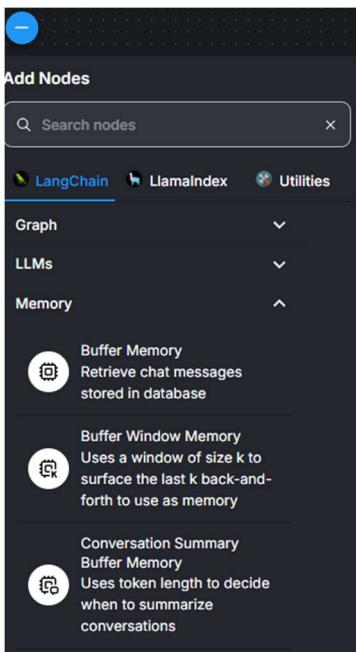


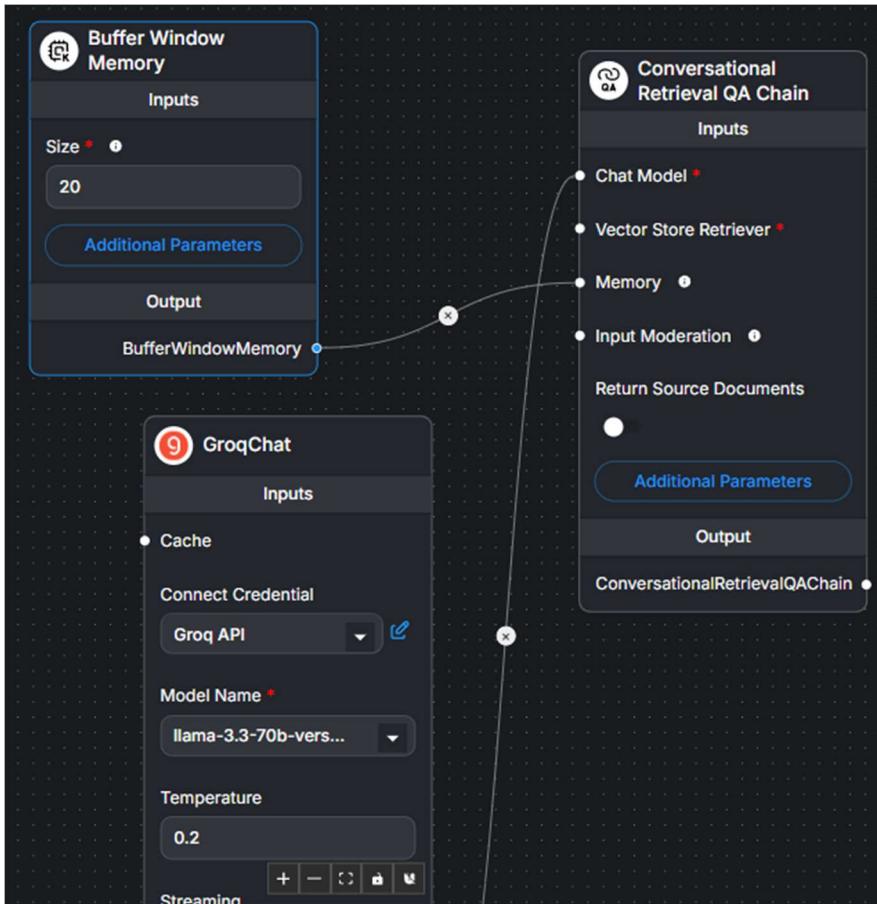
Connect Credential: Groq API



6. Add LLM Memory

Select Add Nodes->LangChain->Memory-> **Buffer Memory Window**, drag to the canvas. Connect it to the Conversation Chain. Set Buffer Window Memory Size to 20.





7. Add data into Document Store

Select main menu->Document Stores

The screenshot shows the Flowise platform interface with the following elements:

- Left Sidebar**: Includes "Chatflows", "Agentflows", "Executions", "Assistants", "Marketplaces", "Tools", "Credentials", "(x) Variables", "(x) API Keys", and a highlighted "Document Stores" button.
- Header**: Shows "Flowise", "Star 42,703", and "niprojectwork's Organization / Default Workspace".
- Document Store Section**: Titled "Document Store" with the sub-instruction "Store and upsert documents for LLM retrieval (RAG)". It features a "Add New" button and an illustration of a computer monitor with documents. Below it, a message says "No Document Stores Created Yet".

Select Document Stores-> Add New

Document Store

Store and upsert documents for LLM retrieval (RAG)

+ Add New Document Store

Name *

Description

Cancel Add

Document Store

Store and upsert documents for LLM retrieval (RAG)

Retro Bites Diner

o EMPTY

0 flow 0 chars 0 chunks

Search Name [Ctrl + F]
grid icon
list icon
+ Add New

Items per page: 12 ▼

Items 1 to 1 of 1

< 1 >

Retro Bites Diner e

o EMPTY



No Document Added Yet

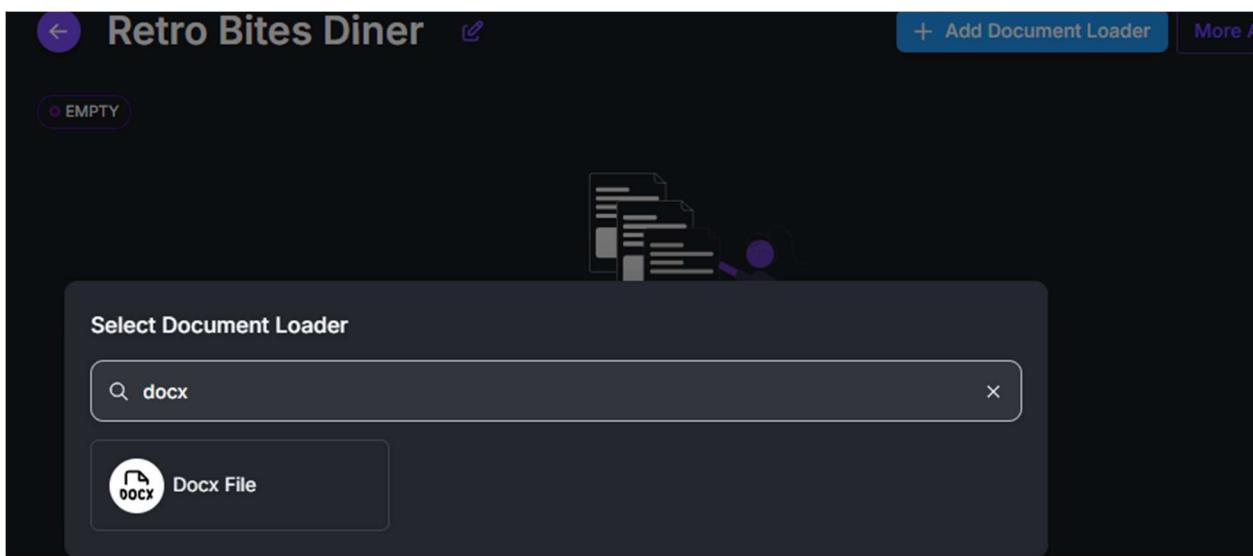
+ Add Document Loader

+ Add Document Loader

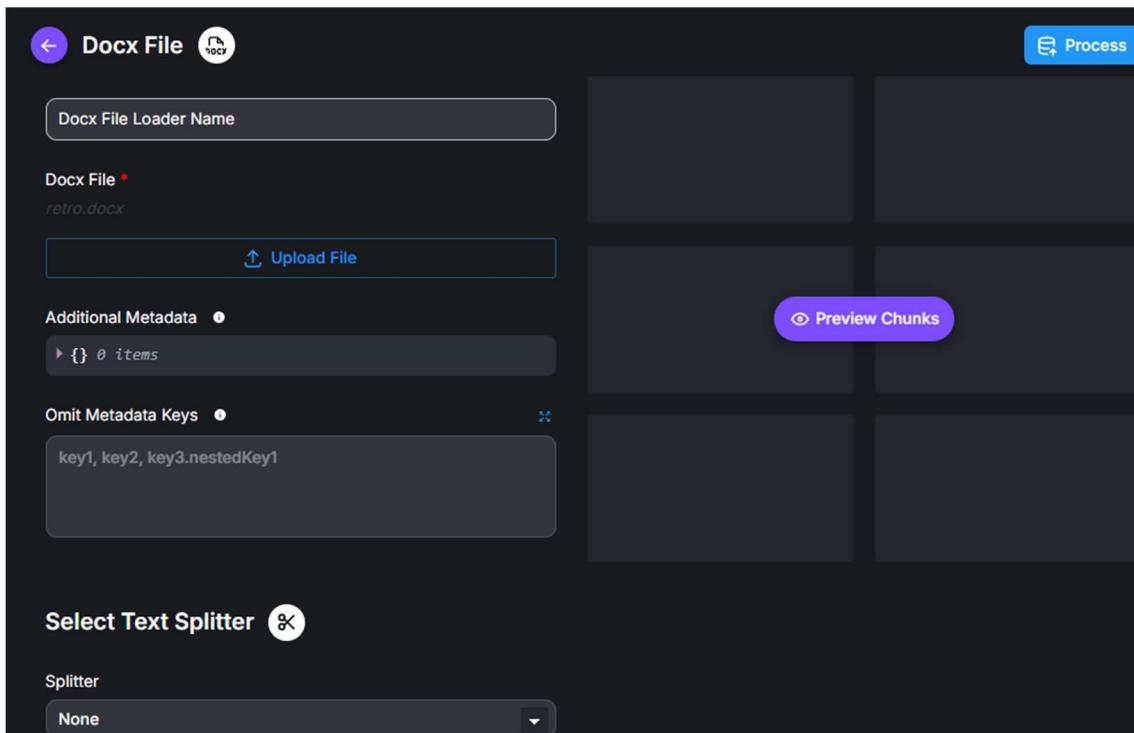
More Actions ▾

Select Add Document Loader.

At Select Document Loader- Search for Docx File.



Upload file **Retro.docx**



Set the Splitter-> Recursive Character Text Splitter

The screenshot shows the Flowise platform interface. On the left sidebar, under 'Document Stores', 'Recursive Character Text Splitter' is selected. The main panel displays the configuration for this splitter:

- Omit Metadata Keys:** key1, key2, key3.nestedKey1
- Splitter:** Recursive Character Text Splitter
- Chunk Size:** 1000
- Chunk Overlap:** 200
- Custom Separators:** ["\n", "\#\#\#"]

Select Process

The screenshot shows the Flowise platform interface. A 'Docx File' process is selected. The configuration includes:

- Docx File Loader Name:** Docx File
- Docx File:** retro.docx
- Upload File:** (button)
- Additional Metadata:** 0 items
- Omit Metadata Keys:** key1, key2, key3.nestedKey1

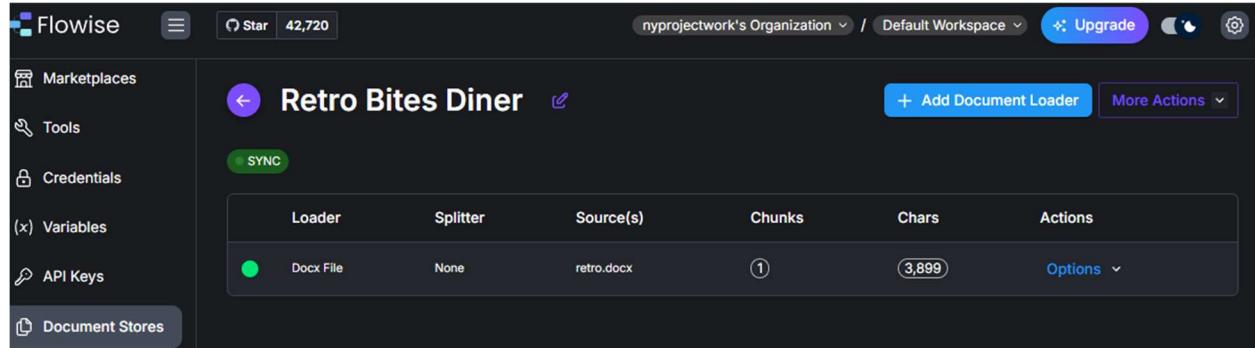
The main panel shows the results of the splitting process:

5 of 5 Chunks

Character Count	Content
#1. Characters: 997	Retro Bites Diner- FAQ General Q: What type of cuisine do you serve? A: The Retro Bites Diner offers a contemporary take on
#2. Characters: 908	Q: How can I make a reservation? A: You can make a reservation online through our website or by calling us at 555-123-4567.
#3. Characters: 874	Q: What is your cancellation policy for
#4. Characters: 816	Q: Do you have non-alcoholic beverages? A: Yes, we offer a variety of non-alcoholic beverages, including house-made juices, organic teas, and craft sodas.
#5. Characters: 783	Q: Do you have valet

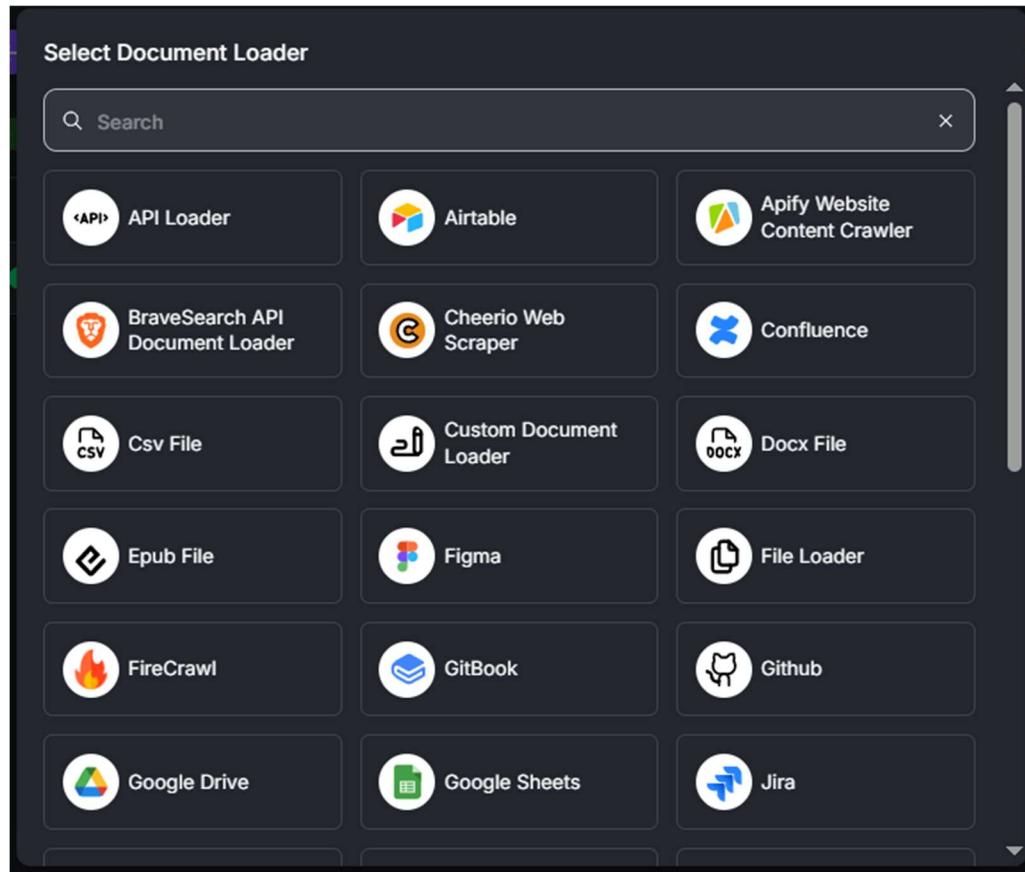
Character Text Splitter: Character Text Splitter

Upon completion of the document store for the docx file. Select +Add Document Loader once again. This time we will upload a csv file contain of the food prices.



The screenshot shows the Flowise platform interface. On the left, there's a sidebar with options: Marketplaces, Tools, Credentials, Variables, API Keys, and Document Stores (which is currently selected). The main area displays a document store titled "Retro Bites Diner". It shows a table with one row: Loader (Docx File), Splitter (None), Source(s) (retro.docx), Chunks (1), and Chars (3,899). There are "Options" and "Actions" buttons. At the top right, there are "Add Document Loader" and "More Actions" buttons. The top navigation bar includes "Star 42,720", "nypyprojectwork's Organization / Default Workspace", and "Upgrade".

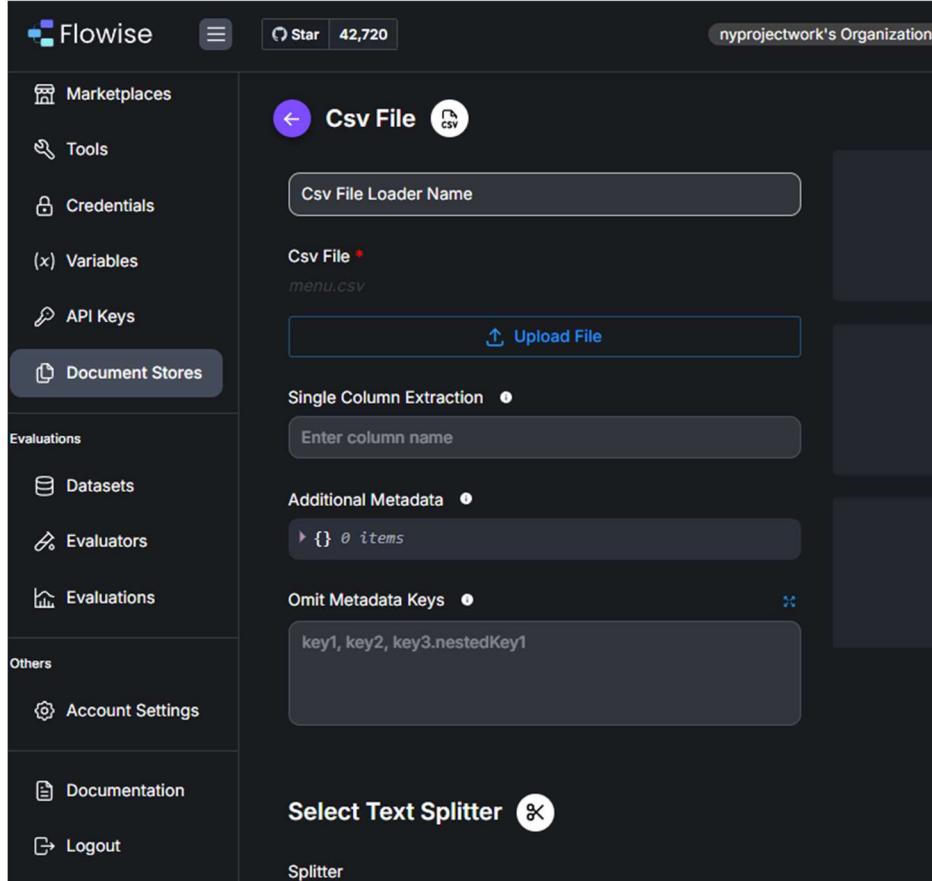
Select Csv File



The screenshot shows a modal window titled "Select Document Loader". It features a search bar at the top. Below it is a grid of 16 document loader options, each with an icon and a name. The options are arranged in four rows of four:

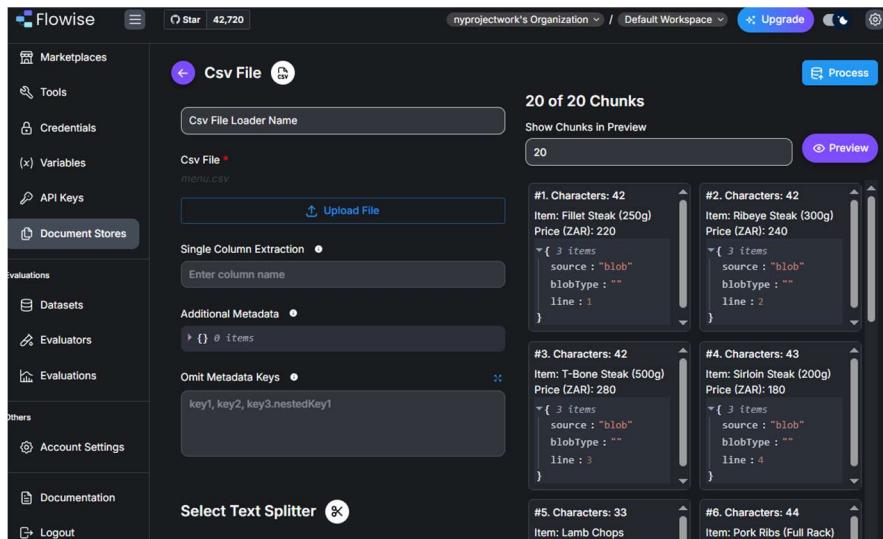
- Row 1: API Loader, Airtable, Apify Website Content Crawler
- Row 2: BraveSearch API Document Loader, Cheerio Web Scraper, Confluence
- Row 3: Csv File, Custom Document Loader, Docx File
- Row 4: Epub File, Figma, File Loader
- Row 5: FireCrawl, GitBook, Github
- Row 6: Google Drive, Google Sheets, Jira

Upload the CSV file -> menu.csv. And no Text Splitter is required because the information is organized row by row.



The screenshot shows the Flowise application interface. On the left sidebar, under 'Document Stores', the 'Csv File' section is selected. In the main area, there is a 'Csv File Loader Name' input field containing 'menu.csv'. Below it is an 'Upload File' button. Under 'Single Column Extraction', there is an 'Enter column name' input field. In 'Additional Metadata', there is a list with one item: '[] 0 items'. In 'Omit Metadata Keys', the value 'key1, key2, key3.nestedKey1' is listed. At the bottom, there is a 'Select Text Splitter' section with a 'Splitter' button.

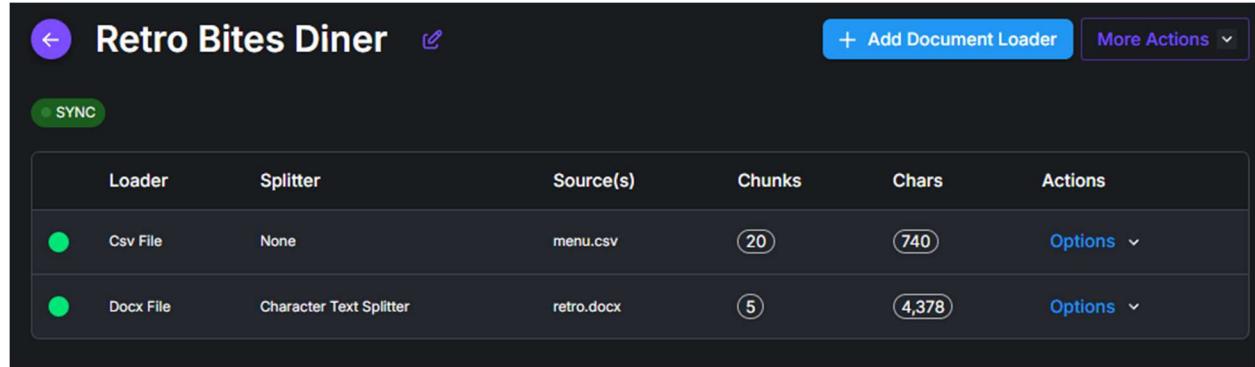
Select Preview Chunk



The screenshot shows the Flowise application interface after processing the CSV file. The main area displays '20 of 20 Chunks' with a 'Preview' button. Each chunk is numbered and has a preview of its content. The first few chunks are:

- #1. Characters: 42
Item: Fillet Steak (250g)
Price (ZAR): 220
[JSON snippet]
- #2. Characters: 42
Item: Ribeye Steak (300g)
Price (ZAR): 240
[JSON snippet]
- #3. Characters: 42
Item: T-Bone Steak (500g)
Price (ZAR): 280
[JSON snippet]
- #4. Characters: 43
Item: Sirloin Steak (200g)
Price (ZAR): 180
[JSON snippet]
- #5. Characters: 33
Item: Lamb Chops
- #6. Characters: 44
Item: Pork Ribs (Full Rack)

Select Process to store data in document store. All processes are complete there will be green dot at each loader.



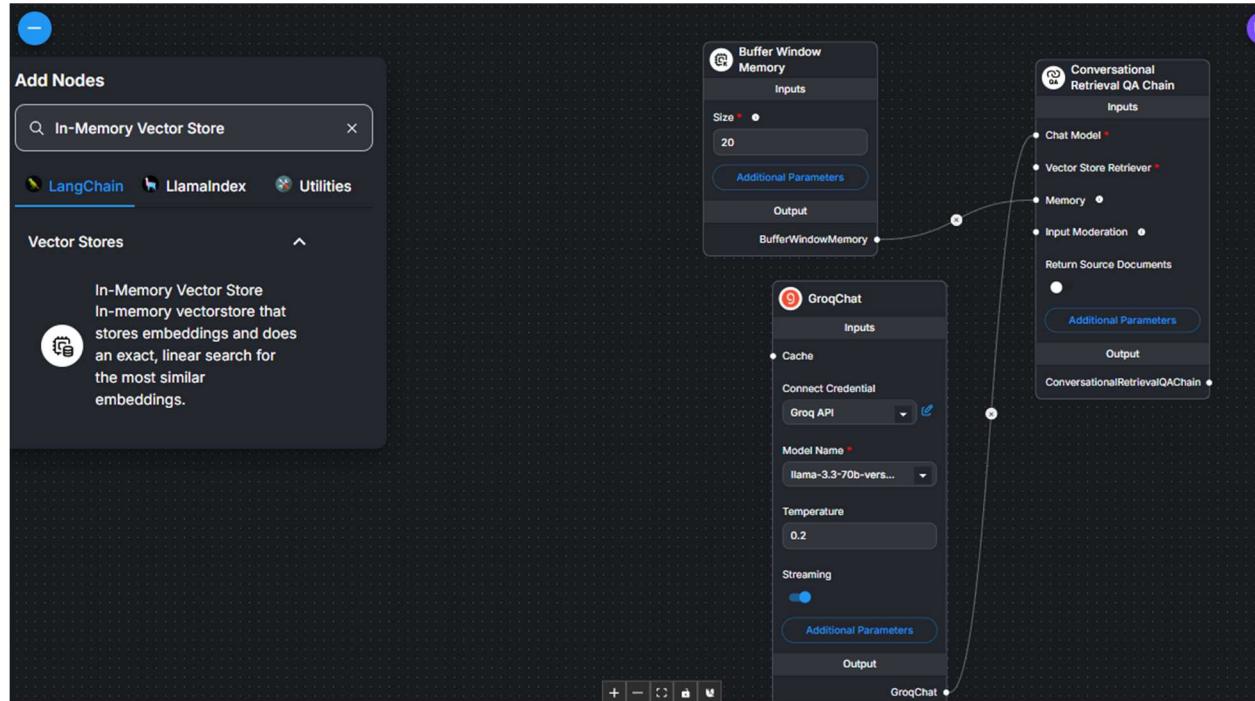
The screenshot shows a dashboard titled "Retro Bites Diner" with a "SYNC" button. Below it is a table with columns: Loader, Splitter, Source(s), Chunks, Chars, and Actions. Two rows are listed:

- Csv File**: Splitter is "None". Source is "menu.csv". Chunks: 20. Chars: 740. Actions: Options
- Docx File**: Splitter is "Character Text Splitter". Source is "retro.docx". Chunks: 5. Chars: 4,378. Actions: Options

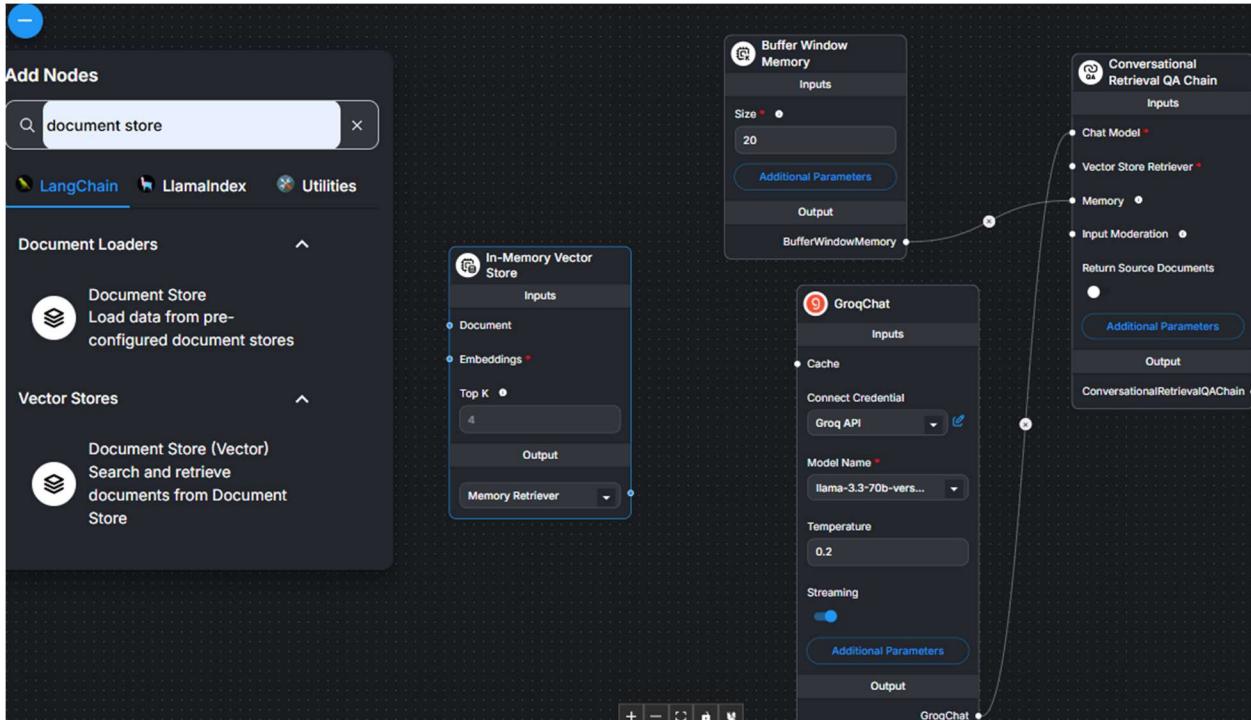
8. Setup Vector store Retrieval

The chunked data need to store as the vector store in order to used as the retrieval of information based on the customer query.

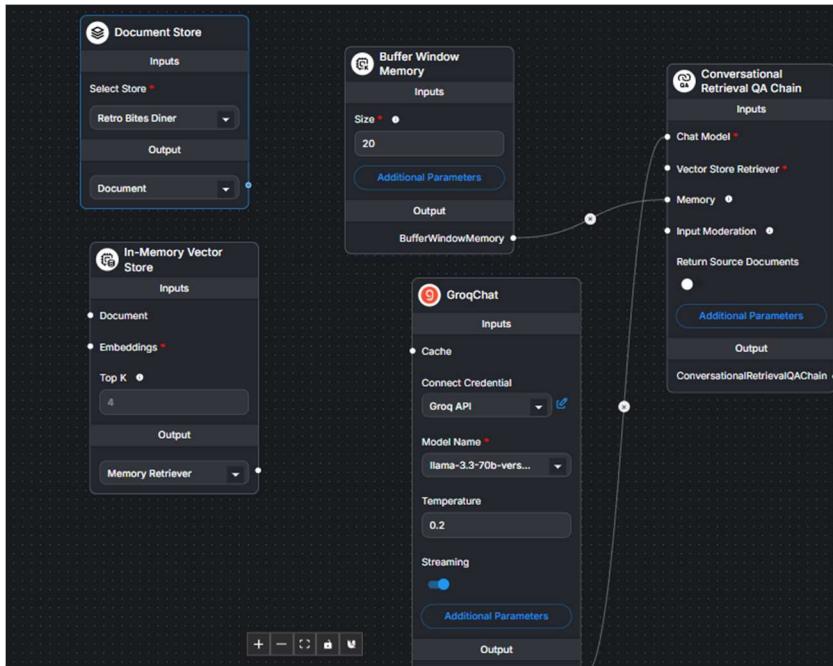
Back to Chatflow, search for **In-Memory Vector Store**, drag into canvas



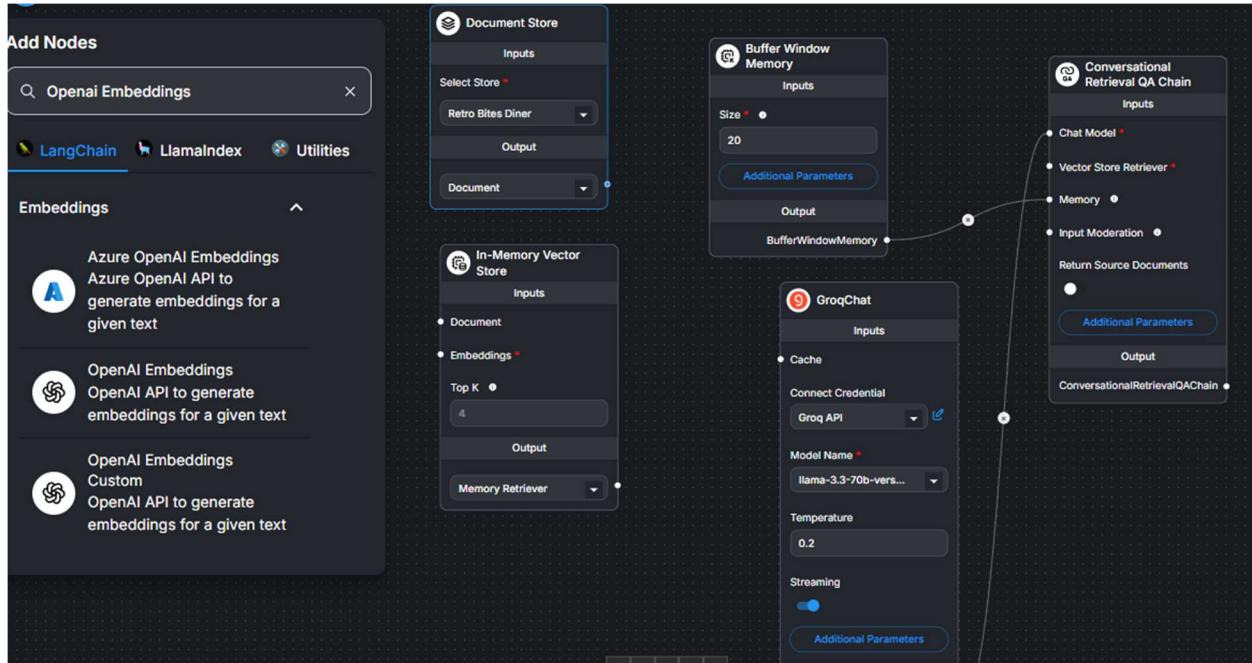
Search for document store. Select Document Loader-> Document Store, drag into canvas.



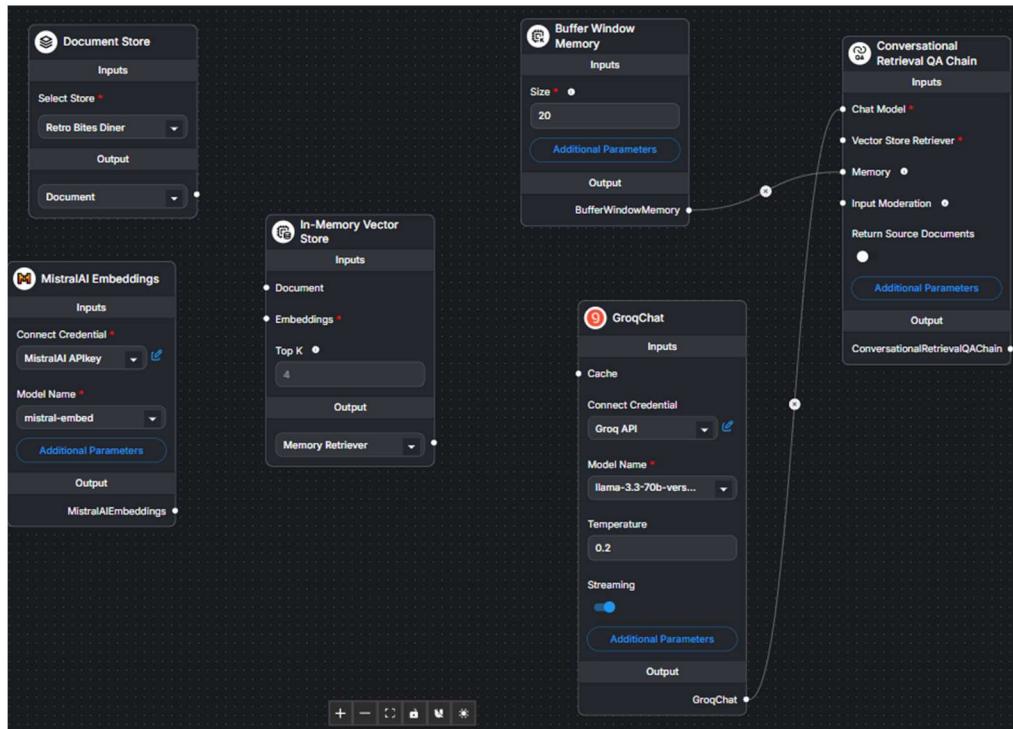
In the Document Store, Select Store->**Retro Bites Diner** (this the document store we created previously)



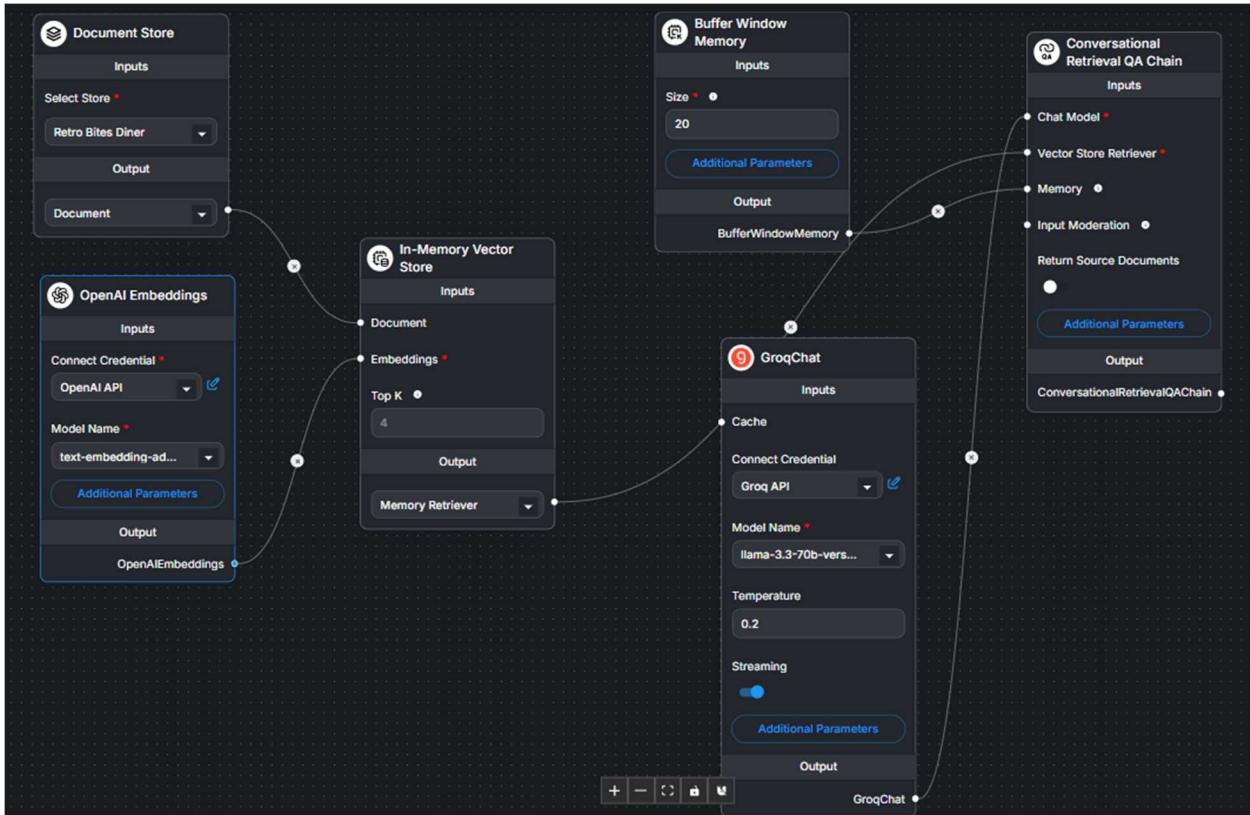
Next Search for Openai Embeddings. Drag into canvas



At the MistralAI Embeddings, Connection Credential -> MistralAI API. And Model Name-> mistral-embed.



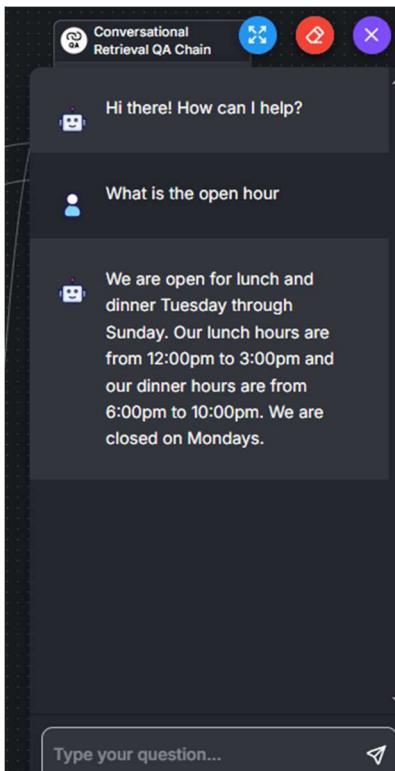
Complete all the connections as seen below



9. Test the Chatbot

Select the Chatbot button at top right.

Enter a question: What is open hour



Enter question: What is your specialty dish?

