🚀 Excited to Share My Latest Project: "CognifyX" 🚀

I’m thrilled to unveil my newest project, CognifyX, a state-of-the-art Retrieval-Augmented Generation (RAG) application designed to revolutionize how we interact with and retrieve information from the web!

Project Overview:

CognifyX is a cutting-edge solution that combines several advanced technologies to provide an enhanced information retrieval experience:

- LangChain and Streamlit frameworks:

- **LangChain:** This powerful framework enables the creation of sophisticated language models and chains, allowing for complex tasks involving natural language understanding and generation. It ensures that CognifyX can manage and orchestrate various AI components efficiently.

- **Streamlit:** Known for its ease of use, Streamlit is employed to build an interactive web application. Users can input URLs and questions, and receive responses seamlessly through an intuitive interface. It ensures a smooth and user-friendly experience.

- Groq API:

- The Groq API provides access to the robust `Llama3` language model, which is used to generate precise and contextually relevant answers. By leveraging Groq’s advanced model, CognifyX can effectively interpret user queries and produce high-quality responses based on the given content or supplementary web search results.

- FireCrawl API:

- FireCrawl API specializes in web scraping and content extraction. It can crawl entire websites and convert their content into structured formats like markdown. This feature is crucial for CognifyX as it extracts and processes information from specified URLs, enriching the application’s data source for answering user queries.

How It Works:

1. User Input:

- Users input a URL and their question into the CognifyX application.

2. Content Extraction:

- FireCrawl API scrapes the content from the provided URL, converting it into a structured format.

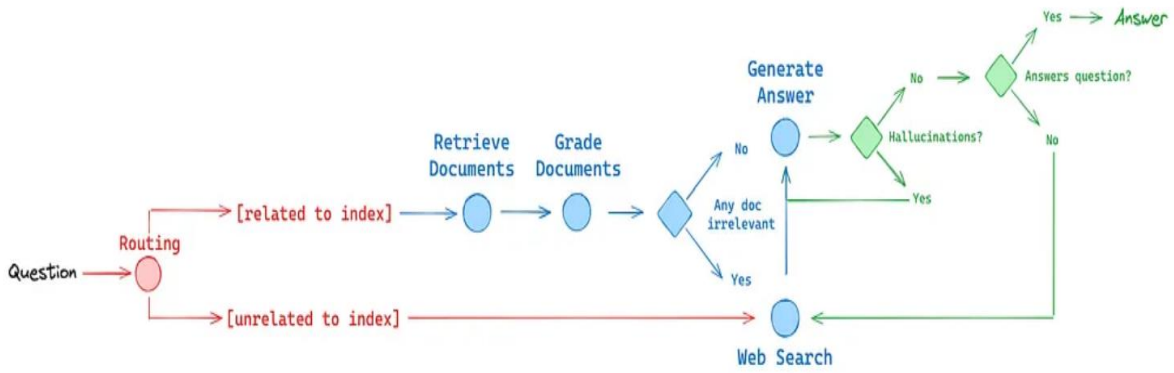
3. Answer Generation:

- Groq API uses the `Llama3` model to generate an answer based on the extracted content. If the content does not sufficiently address the query, the application performs a web search to find additional relevant information.

4. Answer Presentation:

- The generated answer is presented to the user via the Streamlit interface, ensuring a clear and user-friendly experience.

CognifyX represents a significant leap in enhancing information retrieval systems by integrating sophisticated language models with powerful web scraping and search capabilities. It ensures that users receive accurate, comprehensive, and contextually relevant answers to their questions.



There are some step that you have to follow to complete this project:

1.Go to Groq Api documentation website and make your self login and get the Api key and copy that key.

2. Now same for Firecrawl Api make your self login and get the Api key

3. Go to the environment variables advance in system properties and in the user variables section for your login account and system variables for the whole system (all the login accounts) on your requirement.

4. on the bottom of the both section there is a button NEW hit it.

5. and then name the Api as 'GROQ_API_KEY' and enter the copied Api key of Groq Api and then it ok.

6. and same for the FIRECRAWL_API_KEY' click the new and then enter the name and key that you have copy form the firecrawl Api key and paste it and hit enter.

7.then ok button.

8. a good practice is to restart the pc.

Install:

```
pip install langchain streamlit groq tavily firecrawl requests beautifulsoup4
```

api_clients.py

```
import os
from groq import Groq
from firecrawl import FirecrawlApp

class GroqClient:
    def __init__(self, api_key_env_var, model):
        self.api_key = os.environ.get(api_key_env_var)
        self.model = model
        self.client = Groq(api_key=self.api_key)

    def generate_answer(self, content, query):
        chat_completion = self.client.chat.completions.create(
            messages=[
                {"role": "user", "content": query}
```

```
            ],
            model=self.model
        )
        return chat_completion.choices[0].message.content


class FireCrawlClient:
    def __init__(self, api_key):
        self.api_key = api_key
        self.app = FirecrawlApp(api_key=self.api_key)

    def crawl_url(self, url, options={}):
        crawl_result = self.app.crawl_url(url, options)
        return crawl_result

    def check_crawl_status(self, job_id):
        status = self.app.check_crawl_status(job_id)
        return status

    def scrape_url(self, url):
        content = self.app.scrape_url(url)
        return content
```

context_extraction.py

```
# content_extraction.py
import requests
from bs4 import BeautifulSoup

def scrape_url(url):
    response = requests.get(url)
    soup = BeautifulSoup(response.content, 'html.parser')
    content = soup.get_text(separator=' ')
    return content
```

web_search.py

```
# web_search.py
def web_search(query):
    # Mock implementation for the purpose of this example
    # Replace with actual web search implementation
    search_results = [
        {"content": "Sample content from web search result."}
    ]
```

```
    return search_results
```

ap.py

```
import streamlit as st
import os
from api_clients import GroqClient, FireCrawlClient
from content_extraction import scrape_url
from web_search import web_search

# Initialize API clients
GROQ_API_KEY = os.getenv('GROQ_API_KEY', 'YOUR_GROQ_API_KEY')
FIRECRAWL_API_KEY = os.getenv('FIRECRAWL_API_KEY', 'YOUR_FIRECRAWL_API_KEY')

groq_client = GroqClient(api_key_env_var='GROQ_API_KEY', model='llama3-8b-
8192')
firecrawl_client = FireCrawlClient(api_key=FIRECRAWL_API_KEY)

def main():
    st.title("Corrective RAG Application")

    # User inputs
    url = st.text_input("Enter URL:")
    query = st.text_input("Enter your question:")

    if st.button("Get Answer"):
        if url:
            # Scrape content from URL
            content = scrape_url(url)
        else:
            content = ''

        # Generate answer using Llama3 (Groq API)
        answer = groq_client.generate_answer(content, query)

        # If answer is insufficient, perform web search
        if not answer:
            search_results = web_search(query)
            combined_content = " ".join([result['content'] for result in
search_results])
            answer = groq_client.generate_answer(combined_content, query)

        st.write("Answer:", answer)

if __name__ == "__main__":
    main()
```


Follow me!!