A Project Report On

"CAMPUS QUERY"

Submitted in partial fulfilment of the requirement for the award of degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE & ENGINEERING

Submitted By:

MANAS BARNWAL (20130)

PRIYANKA (20138)

DEVESH MISHRA (20171)

Under the guidance of:

Dr. AVADHESH KUMAR DIXIT

INSTITUTE OF ENGINEERING & TECHNOLOGY
Dr. RAMMANOHAR LOHIA AVADH UNIVERSITY
AYODHYA UTTAR PRADESH-224001

SESSION 2023-24

DECLARATION

We hereby declare that project work entitled “CAMPUS QUERY” is an authentic work carried out at Institute of Engineering & Technology, Dr. Rammanohar Lohia Avadh University, Ayodhya as requirements for the award of degree of **Bachelor of Technology in Computer Science** and Engineering under the guidance of **Dr. Avadhesh Kumar Dixit**, Assistant Professor, CSE, **IET, Dr. RML Avadh University**. The information and data given in the project report is authentic and original to the best of my knowledge. The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Manas Barnwal (20130) **Priyanka (20138)** **Devesh Mishra (20171)**

<u>CERTIFICATE</u>

This is to certify that the project report entitled “CAMPUS QUERY” submitted in the partial fulfilment of degree of “Bachelor of Technology in Computer Science and Engineering” to the “**IET, Dr. Ram Manohar Lohia Avadh University, Ayodhya**” is a Bonafede project work carried out by **MANAS BARNWAL** (20130), **PRIYANKA** (20138), **DEVESH MISHRA** (20171), under my supervision.

Date:	**Dr. Avadhesh Kumar Dixit**
Place:	**Assistant Professor, CSE**

ACKNOWLEDGEMENT

Many lives and destinies are destroyed due to the lack of proper guidance, directions & opportunities. It this respect we feel that we are in much better condition today due to the continuous process of motivation & focus provide by our parents & teachers. The process of completion of this project was a tedious job & requires care & support at all stages.

We are eternally grateful to our Project guide Dr. Avadhesh Kumar Dixit, Assistant Professor, CSE and the Head of Department CSE for providing us the opportunity & infrastructures to complete the project as a partial fulfilment of B. Tech degree.

I would like to give a special thanks to Prof. S.S Mishra (Director) and Dr. Lokendra Singh Umrao (HOD of Computer Science and Engineering) for providing all the necessary facilities to us.

There are no words to express our regards and legitimate gratitude to all our esteemed techno personality faculties for their guidance throughout this project despite of their busy schedule.

It is because of these people that we are able to complete this project work successfully. I am also thankful to all visible and invisible hands of cooperation who helped us to complete this project with a feeling of success.

ABSTRACT

This project uses LangChain to create a question-answering (QA) system over university website text content. The system first loads the text data from university website into the LangChain document, then splits it into smaller text segments. These segments are then embedded into a vector space, which allows them to be compared and retrieved efficiently. When a user asks a question about the university, the system retrieves the most similar segments from the vector space and returns the answer that is most likely to be correct.

The system is able to answer a wide range of questions about the university, including factual questions, conceptual questions, and even creative questions. It is also able to handle questions that are ambiguous or open-ended. This makes it a valuable tool for students, researchers, and anyone else who needs to access information about the university.

For example, it has been used to answer questions about admission process, departments, activities, etc.

INDEX

<u>**List of Figures Page No:**</u>

CHAPTER – 1

INTRODUCTION

Since the release of ChatGPT, large language models (LLMs) have gained a lot of popularity. Although you probably don’t have enough money and computational resources to train an LLM from scratch in your basement, you can still use pre-trained LLMs to build something cool such as:

- Personal assistants that can interact with the outside world based on your data
- Chatbots customized for your purpose
- Analysis or summarization of your documents or code

This project aims to create a QA system that can answer questions about the data available on the university website.

The system will use LangChain to retrieve the answers from the text document, Groq API to generate text, and Streamlit Python to create a user interface. A vector database (Faiss) will be used to store the embeddings of the text documents. This will allow the system to quickly retrieve relevant text chunks when answering questions.

LangChain is a framework built to help you build LLM-powered applications more easily by providing you with the following:

- A generic interface to a variety of different foundation models (see Models),
- A framework to help you manage your prompts (see Prompts), and
- A central interface to long-term memory (see Memory), external data (see Indexes), other LLMs (see Chains), and other agents for tasks an LLM is not able to handle (e.g., calculations or search) (see Agents).

Because LangChain has a lot of different functionalities, it may be challenging to understand what it does at first. That’s why we will go over the (currently) six key modules of LangChain in this article to give you a better understanding of its capabilities.

1.1 MOTIVATION

We were motivated to make this project because we were interested in the intersection of natural language processing, LangChain, and RAG. we wanted to create a system that could use these technologies to answer the questions regarding university in a comprehensive and informative way.

we were also motivated by the fact that there are many people who need to access information from the university, but who may not have the skills or resources to do so. We wanted to create a system that would make it easier for people to find the information they need, regardless of their background or technical knowledge.

Finally, we were motivated by the challenge of creating a complex system that could actually work. we wanted to test my skills and learn new things, and we thought this project would be a good way to do that.
We are still working on this project, but we are excited about the potential it has to help people. We hope that one day, this project will be used by students, researchers, and anyone else who needs to access information from the university.

Here are some specific things that motivated me to make this project:

- We wanted to create a system that would be easy to use and accessible to a wide range of people. I didn't want to create a system that would only be useful to experts.
- We wanted to make a difference in the world. I thought this project could help people to access information more easily, which could lead to better decision-making and improved outcomes.
- We are proud of what I have accomplished so far, and I am excited to continue working on this project. I believe that it has the potential to make a real difference in the world.

1.2 FUNCTIONALITY AND DESIGN

1.2.1 Functionalities

- The system would be able answer the user’s queries related to the university.
- The system would be able to generate text that is relevant to the user's question.
- The system would be able to retrieve relevant information for student, teacher or anyone.
- The system would be able to combine the generated text and the retrieved text chunks to create a response to the user's question.
- The system would be able to handle a variety of question types, including factual questions, conceptual questions, and open-ended questions.

The system would be able to provide accurate and relevant answers to the user's questions.

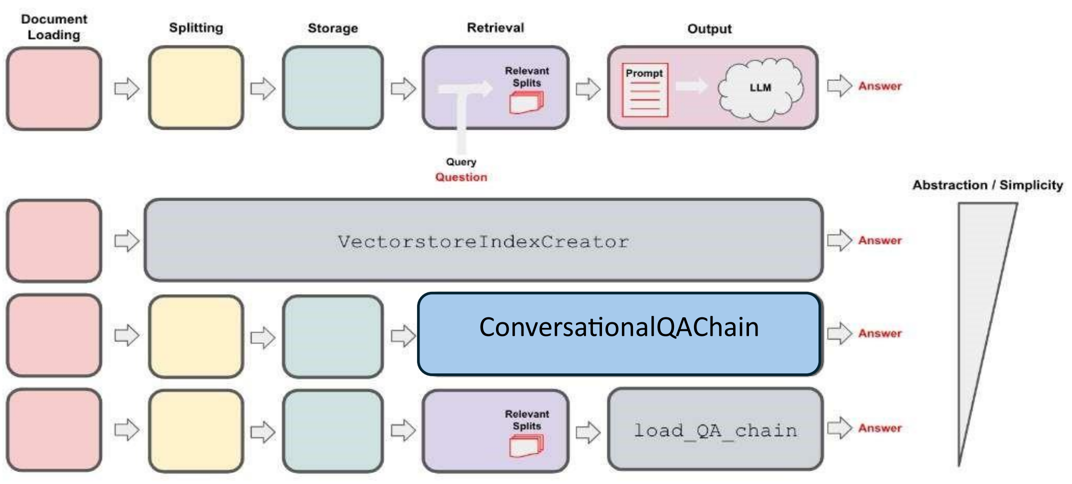
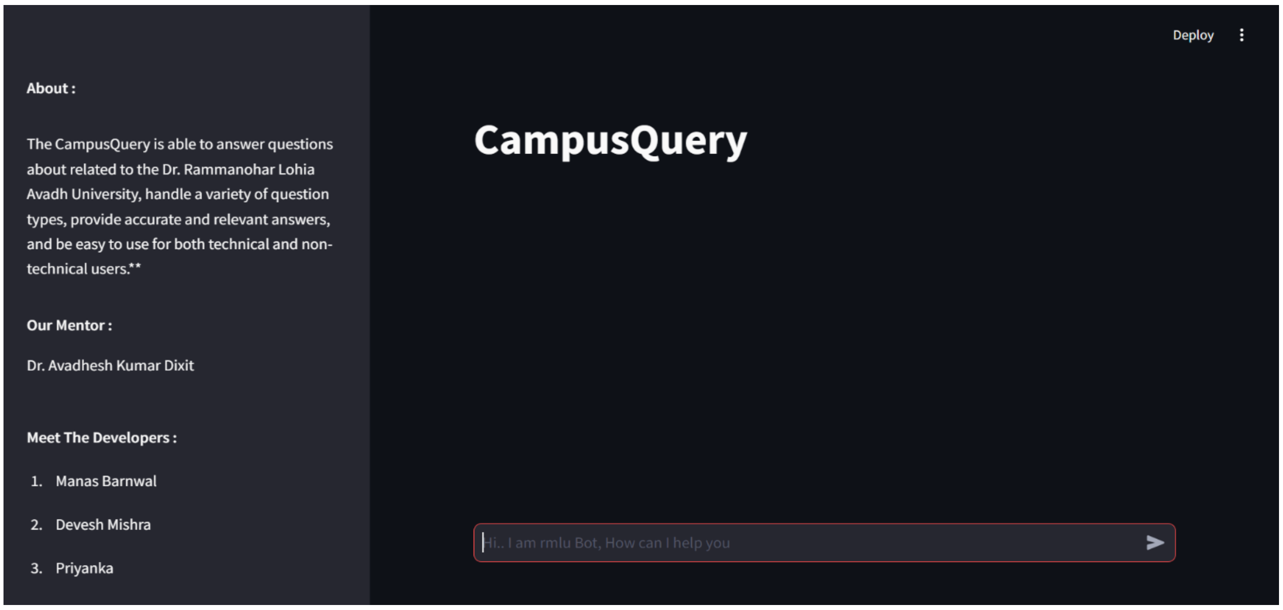


Fig-1 : Project Component

1.2.2 Design

- The system would be easy to use for both technical and non-technical users.
- The system would be scalable to handle a large number of users and questions.
- The system would be robust and able to handle errors and unexpected inputs.
- The system would be secure and protect the privacy of the user's data.



About :

The CampusQuery is able to answer questions about related to the Dr. Rammanohar Lohia Avadh University, handle a variety of question types, provide accurate and relevant answers, and be easy to use for both technical and non-technical users.**

Our Mentor :

Dr. Avadhesh Kumar Dixit

Meet The Developers :

1. Manas Barnwal
2. Devesh Mishra
3. Priyanka

CampusQuery

Hi.. I am rmlu Bot, How can I help you

Fig-2: User Interface

1.3 INTRODUCTION TO LLM

LLMs are a type of artificial intelligence (AI) that are trained on massive datasets of text and code. This training allows them to learn the statistical relationships between words and phrases, which gives them the ability to generate human-like text, translate languages, write different kinds of creative content, and answer your questions in an informative way.

LLMs are still under development, but they have already been shown to be capable of a wide range of tasks, including:

- **1.3.1 Generating text**: LLMs can be used to generate text in a variety of styles, including news articles, fiction, and poetry. They can also be used to translate languages and write different kinds of creative content, such as scripts, musical pieces, and email.
- **1.3.2 Answering questions**: LLMs can be used to answer questions about a variety of topics, both factual and conceptual. They can also be used to generate summaries of factual topics.
- **1.3.3 Translating languages**: LLMs can be used to translate text from one language to another. They are often more accurate than traditional machine translation methods.
- **1.3.4 Summarizing text**: LLMs can be used to summarize text, extracting the most important information and presenting it in a concise and easy-to understand format.

LLMs have the potential to revolutionize the way we interact with computers. They can be used to create more natural and engaging user interfaces, to provide us with more accurate and relevant information, and to help us to be more creative.

Here are some examples of how LLMs are being used today:

- Google Search uses LLMs to provide more relevant search results.
- Facebook uses LLMs to generate more engaging content for its users.
- Microsoft uses LLMs to power its translation services.
- OpenAI uses LLMs to create its chatbot, GPT-3.

LLMs are still a relatively new technology, but they are rapidly evolving. As they become more powerful and capable, we can expect to see them used in even more ways in the future.

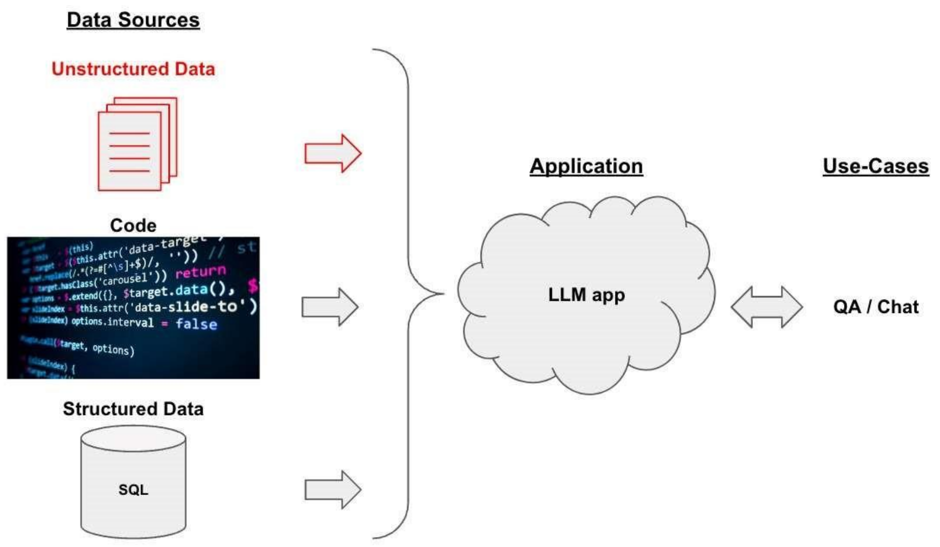


Fig-3: High Level View of Project

1.4 INTRODUCTION TO LANGCHAIN

LangChain is an innovative framework that is revolutionizing the way we develop applications powered by language models. By incorporating advanced principles, LangChain is redefining the limits of what can be achieved through traditional APIs. Furthermore, LangChain applications are agentic, enabling language models to interact and adapt to their environment with ease.

Langchain consist of few modules. As its name suggests, CHAINING different modules together are the main purpose of Langchain. The idea here is to chain every module in one chain and, at last, use that chain to call all modules at once.

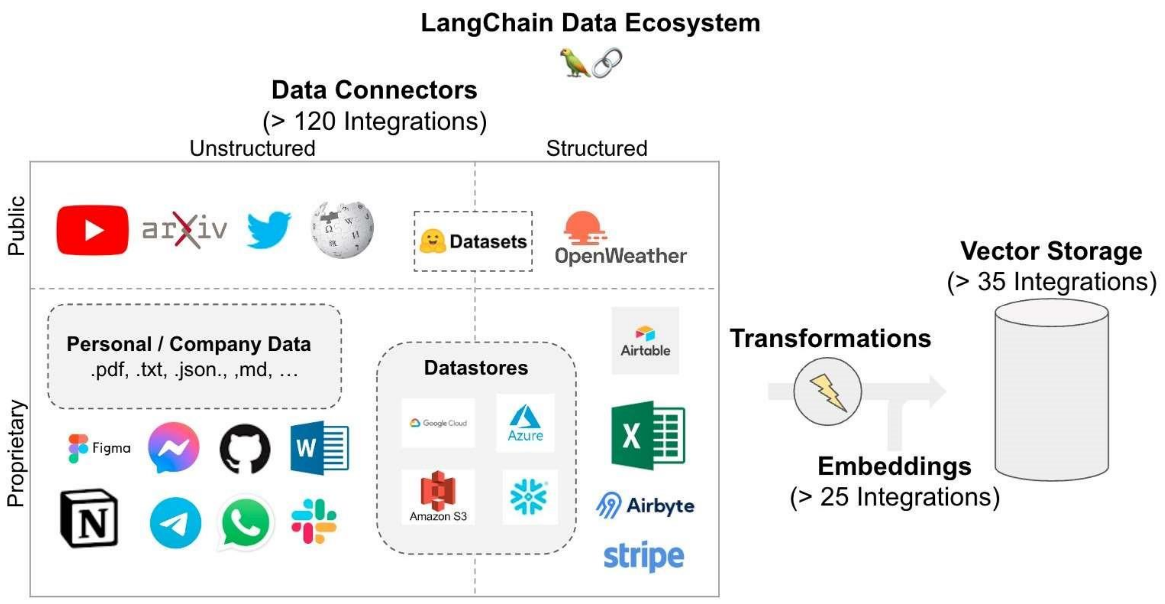


Fig-4: LangChain Ecosystem

1.5 WORKING OF LANGCHAIN

The modules consist of the following:

1. Model
2. Prompt
3. Memory
4. Chain
5. Agents
6. Callback
7. Indexes

1.5.1 Model:

As discussed in the introduction, Models mostly cover Large Language models. A large language model of considerable size is a model that comprises a neural network with numerous parameters and is trained on vast quantities of unlabelled text. There are various LLMs by tech giants, like,

1. BERT by Google
2. GPT-3 by OpenAI
3. LaMDA by Google
4. PaLM by Google
5. LLaMA by Meta AI

6. mixtral-8x7b by Groq API

With the help of the lang-chain, the interaction with large language models gets easier. The interface and functionality provided by Langchain help to integrate the power of LLMs easily into your work application. LangChain provides async support for LLMs by leveraging the asynchronized library.

There is also Async support [By releasing the thread that handles the request, the server can allocate it for other tasks until the response is prepared, which maximizes resource utilization.] provided by langchain for the case where you want to call multiple LLMs concurrently and it is known as network-bound.

Currently, OpenAi, PromptLayerOpenAI, ChatOpenAI and Anthropic are supported, but async support for other LLMs is on the roadmap. You can use the generate method to call an OpenAI LLM asynchronously. You can also write a **custom LLM wrapper** than one that is supported in LangChain.

1.5.2 Prompts:

We all know that prompt is the input that we give to any system to refine our answers to make them more accurate or more specific according to our use case. Many times, you may want to get more structured information than just text back. Many new object detection, and classification algorithms which are based on contrastive pretraining and zero-shot learning included prompts as a valid input for results. Just to give examples, OpenAI's CLIP and META's GROUNDING DINO use prompts as input for predictions.

In Langchain, we can set a prompt template according to the answer we want and then chain it to the main chain for output prediction. There is also a facility for the output parser to refine results. Output Parsers are responsible for (1) instructing the model how output should be formatted, and (2) parsing output into the desired formatting (including retrying if necessary).

In Langchain, we can provide prompts as a template. Template refers to the blueprint of the prompt or the specific format in which we want an answer. LangChain offers pre-designed prompt templates that can generate prompts for different types of tasks. Nevertheless, in some situations, the default templates may not satisfy your requirements. With the default, we can use our custom prompt template.

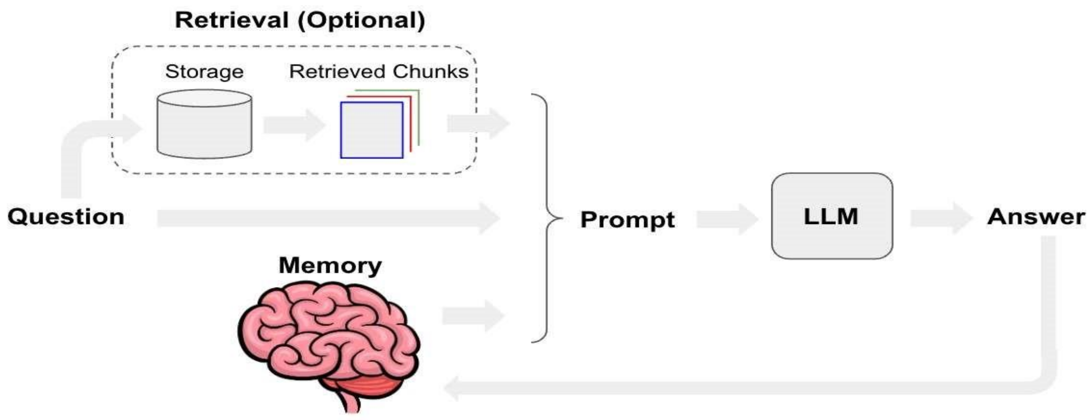


Fig-5: RAG with System and User Prompt

1.5.3 Memory:

Chains and Agents in LangChain operate in a stateless mode by default, meaning that they handle each incoming query independently. However, there are certain applications, like chatbots, where it is of great importance to retain previous interactions, both over the short and long term. This is where the concept of "Memory" comes into play.

LangChain provides memory components in two forms. First, LangChain provides helper utilities for managing and manipulating previous chat messages, which are designed to be modular and useful regardless of their use case. Second, LangChain offers an easy way to integrate these utilities into chains. This makes them highly versatile and adaptable to any situation.

1.5.4 Chains:

Chains provide a means to merge various components into a unified application. A chain can be created, for instance, that receives input from a user, formats it using a Prompt Template, and subsequently transmits the formatted reply to an LLM. More intricate chains can be generated by integrating multiple chains with other components.

LLM Chain is considered to be among the most widely utilized methods for querying an LLM object. It formats the provided input key values, along with memory key values (if they exist), according to the prompt template, and subsequently sends the formatted string to LLM, which then produces an output that is returned.

One may take a series of steps subsequent to invoking a language model, where a sequence of calls to the model can be made. This practice is especially valuable when there is a desire to utilize the output from one call as the input for another. In this series of chains, each individual chain has a single input and a single output, and the output of one step is used as input to the next.

1.5.5 Agents:

Certain applications may necessitate not only a pre-determined sequence of LLM/other tool calls but also an uncertain sequence that is dependent on the user's input. These kinds of sequences include an "agent" that has access to a range of tools. Based on the user input, the agent may determine which of these tools, if any, should be called.

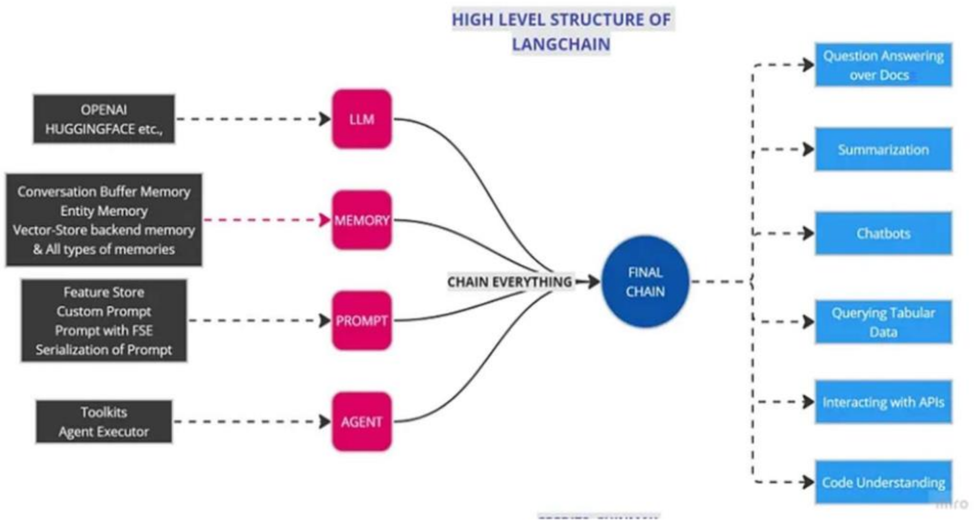

HIGH LEVEL STRUCTURE OF LANGCHAIN
OPENAI HUGGINGFACE etc.,
Conversation Buffer Memory Entity Memory Vector-Store backend memory & All types of memories
Feature Store Custom Prompt Prompt with FSE Serialization of Prompt
Toolkits Agent Executor
LLM
MEMORY
PROMPT
AGENT
CHAIN EVERYTHING
FINAL CHAIN
Question Answering over Docs
Summarization
Chatbots
Querying Tabular Data
Interacting with APIs
Code Understanding

Fig-6: Different Agents used in LLM

1.6 INTRODUCTION TO GROQ API

The Groq API is a powerful tool that can be used to access Groq's cutting- edge language models. It provides a simple interface that can be used to solve virtually any task that involves processing language.

The API has two main endpoints:

Completions: This endpoint allows you to generate text, translate languages, write different kinds of creative content, and answer your questions in an informative way.

Chat: This endpoint allows you to create a chatbot that can have conversations with humans.

To use the API, you will need to obtain an API key from OpenAI. Once you have your API key, you can start making API calls to your chosen model endpoints.

Here is an example of how to use the completions endpoint to generate a poem:

```
import groqapi

# Initialize the Groq API client
client = groqapi.Client(api_key="YOUR_API_KEY")

# Create a text completion request
response = client.chat.generate(
    prompt="Write a poem about a flower",
    max_tokens=100
)

# Print the generated text
print(response["text"])
```

This code will generate a poem about a flower, using mixtral model in Groq language model. The `max_tokens` parameter controls the length of the poem.

The Groq API is a powerful tool that can be used for a variety of projects. Here are a few ideas:

Create a natural language processing (NLP) application.

The Groq API can be used to power a variety of NLP applications, such as a chatbot, a text summarizer, or a question answering system.

Develop a new creative writing tool.

The Groq API can be used to generate creative text content, such as poems, code, scripts, musical pieces, email, letters, etc.

Research the potential applications of large language models.

The Groq API can be used to explore the potential applications of large language models in a variety of fields, such as education, healthcare, and finance.

1.7 INTRODUCTION TO STREAMLIT

Streamlit is an open-source Python framework that makes it easy to create interactive web apps for data science and machine learning. It is a great choice for rapid prototyping and smaller projects, as it is simple to learn and use.

Streamlit can be used to create a variety of web apps, including:

Data visualization apps: Streamlit can be used to create interactive visualizations of data, such as charts, graphs, and maps.

Machine learning apps: Streamlit can be used to create web apps that use machine learning models to make predictions or recommendations.

Dashboards: Streamlit can be used to create dashboards that display data in a clear and concise way.

Streamlit is a great choice for project thesis because it is:

- Easy to learn and use: Streamlit is a very simple framework to learn and use. You can get started with Streamlit with just a few lines of Python code.

- Rapid prototyping: Streamlit is great for rapid prototyping. You can quickly create a working web app with Streamlit, even if you are not a web developer. Interactive: Streamlit apps are interactive, which makes them engaging for users. Users can interact with your app by clicking on buttons, entering data, and selecting options.

- Deployable: Streamlit apps can be deployed to the web with a few clicks. This makes it easy to share your app with others.

If you are looking for a framework to create interactive web apps for your project thesis, Streamlit is a great option. It is easy to learn and use, rapid prototyping, interactive, and deployable.

1.8 INTRODUCTION TO VECTOR DATABASE

A vector database is a type of database that stores data as high-dimensional vectors, which are mathematical representations of features or attributes. This makes it well-suited for storing and querying data that is naturally represented as vectors, such as text, images, and audio.

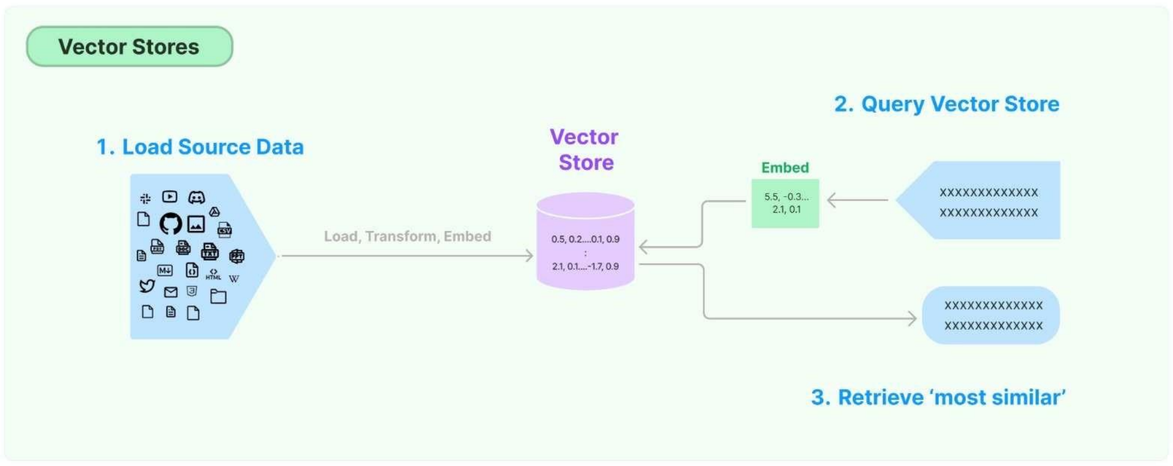
Vector databases offer several advantages over traditional relational databases, including:

- Efficient search and retrieval: Vector databases can efficiently search and retrieve data using vector similarity, which is a measure of how similar two vectors are. This makes them well-suited for applications where users need to find similar items, such as search engines, recommender systems, and fraud detection systems.
- Scalability: Vector databases can be scaled horizontally to handle large amounts of data. This makes them a good choice for big data applications.

- Flexibility: Vector databases can be used to store and query a wide variety of data types, including text, images, audio, and video. This makes them a versatile platform for a variety of applications.

Here are some of the most popular vector databases:

Elasticsearch: Elasticsearch is a popular open-source search engine that can be used as a vector database. It is scalable, flexible, and easy to use.
Milvus: Milvus is a high-performance vector database that is designed for largescale machine learning applications. It is fast, scalable, and fault-tolerant.
FAISS: FAISS is a fast approximate nearest neighbor search library that can be used as a vector database. It is fast and efficient for finding similar items in large datasets.
Annoy: Annoy is a fast approximate nearest neighbor search library that is designed for efficient storage and search of high-dimensional vectors. It is easy to use and can be used as a vector database.



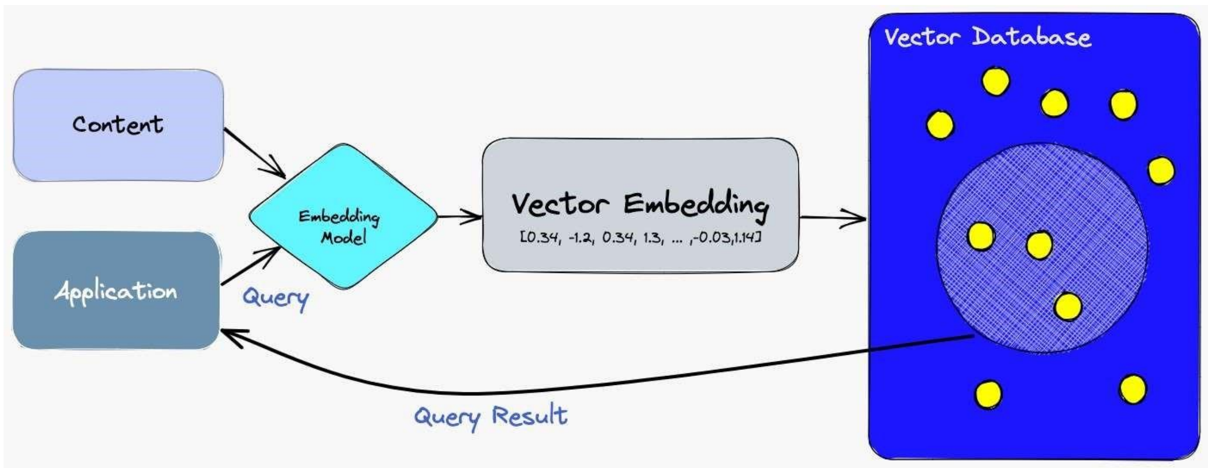


Fig-7: Saving Embeddings into the Vector Database

CHAPTER – 2

TECHNOLOGY USED

2.1 PYTHON:

Python is a general-purpose, high-level programming language. It is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming.

Python is a popular language for a variety of tasks, including:

- Web development: Python is often used for web development, as it is easy to learn and use. It is also a good choice for rapid prototyping, as it is quick to write and execute code.
- Data science: Python is a popular language for data science, as it has a number of libraries that make it easy to work with data. These libraries include NumPy, SciPy, and Pandas.
- Machine learning: Python is also a popular language for machine learning, as it has a number of libraries that make it easy to train and deploy machine learning models. These libraries include TensorFlow, PyTorch, and sci-kit learn.
- Software development: Python is a general-purpose language, so it can be used for any type of software development. It is particularly popular for developing web applications and data science applications.

Libraires required for this project:

```
langchain==0.2.12
langchain-groq==0.1.9
langchain-community==0.2.11
faiss-cpu==1.8.0.post1
torch==2.2.0
transformers==4.43.3
streamlit
sentence-transformers==3.0.1
```

2.2 LANGCHAIN:

To import the LangChain library in Python, you can use the following command:

```
pip install langchain
```

Once the LangChain library is installed, you can import it into your Python code using the following import statement:

```
from langchain_groq import ChatGroq
```

The LangChain library provides a number of different modules that you can use to interact with language models. For example, the model's module provides classes for different language models, such as OpenAI's GPT-3 model.

To use a language model from LangChain, you first need to create an instance of the model class. For example, to create an instance of the OpenAI GPT-3 model, you would use the following code:

```
chat = ChatGroq(temperature = 1,
            groq_api_key = api_key,
            model_name = "mixtral-8x7b-32768")
```

The api_key parameter is the API key for your Groq account. You can get an API key from the Groq website.

Once you have created an instance of the language model, you can use it to generate text, translate text, answer questions, or perform other tasks. For example, to generate text, you would use the following code:

```
prompt = "Write poem about cat."
text=model.generate(prompt)
```

This code will generate a poem about a cat using the mixtral model through Groq.

2.3 PYTHON-DOTENV:

Python-dotenv is a Python library that allows you to load environment variables from a .env file into your Python application. This is useful for storing sensitive information, such as API keys or database credentials, in a file that is not tracked by version control.

To import Python-dotenv in Python, you can use the following command:

```
pip install python-dotenv
```

Once the Python-dotenv library is installed, you can import it into your Python code using the following import statement:

```
import dotenv
```

The Python-dotenv library provides a number of functions that you can use to load environment variables from a .env file into your Python application.

To load the environment variables from a .env file, you can use the load_dotenv() function. For example, the following code would load the environment variables from the .env file in the current working directory:

```
dotenv.load_dotenv()
```

The load_dotenv() function will first look for the .env file in the current working directory. If it does not find the file there, it will then look for the file in the parent directory, and so on, until it finds the file or it reaches the root directory of the filesystem.

2.4 Streamlit:

Streamlit is an open-source Python library that makes it easy to create and share beautiful, custom web apps for machine learning and data science. In just a few minutes you can build and deploy powerful data apps.

To import Streamlit in Python, you can use the following command:

```
pip install streamlit
```

Once the Streamlit library is installed, you can import it into your Python code using the following import statement:

```
import streamlit as st
```

The Streamlit library provides a number of different widgets that you can use to create interactive web apps. For example, the text() widget can be used to display text, the input() widget can be used to get user input, and the plot() widget can be used to plot data.

2.5 Faiss-cpu:

To import Faiss-cpu in Python, you can use the following command:

```
pip pip install faiss-cpu
```

Once the Faiss-cpu library is installed, you can import it into your Python code using the following import statement:

```
from langchain_community.vectorstores import FAISS
```

The Faiss-cpu library provides a number of different classes that you can use to perform similarity search on vectors. For example, the IndexFlatL2 class can be used to perform similarity search on vectors using the L2 distance.

CHAPTER - 3

METHODOLOGY

3.1 BUILDING OF SYSTEM

The pipeline for converting raw unstructured data into a QA chain looks like this:

1. Loading: First, we need to load our data. Unstructured data can be loaded from many sources. Use the LangChain integration hub to browse the full set of loaders. Each loader returns data as a LangChain Document.
2. Splitting: Text splitters break Documents into splits of specified size
3. Storage: Storage (e.g., often a vectorstore) will house and often embed the splits
4. Retrieval: The app retrieves split from storage (e.g., often with similar embeddings to the input question)
5. Generation: An LLM produces an answer using a prompt that includes the question and the retrieved data
6. Conversation (Extension): Hold a multi-turn conversation by adding Memory to your Conversational QA chain

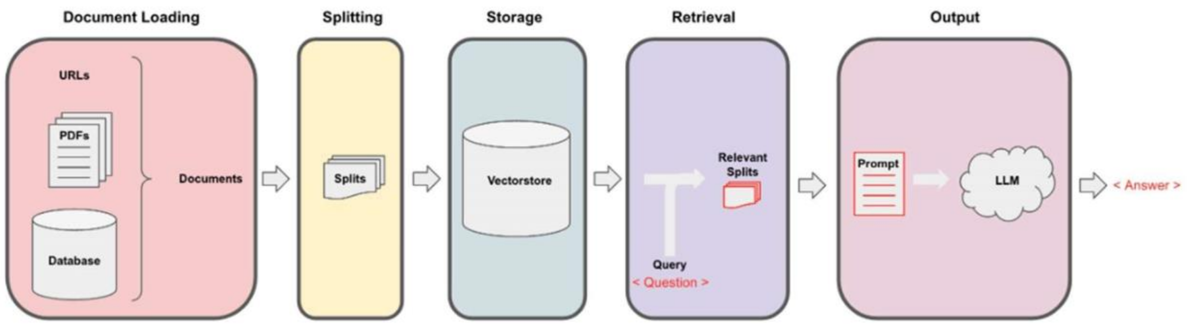


Fig-8: Complete Conversational QA Pipeline

For that, let's take a look at how we can construct this pipeline piece by piece.

3.2 STEPS:

Step 1. **Load**:

Specify a Document Loader to load in your unstructured data as Documents. A Document is a piece of text (the page content) and associated metadata.

```
def document_loader():
    filename = "Dataset/RmluBOT.txt"
    with open(filename, errors="ignore") as file:
        contents = file.read()
    return contents
```

Step 2. **Split into chunks**:

```
char_splitter = RecursiveCharacterTextSplitter(
    separators = ['\n', '\n\n', ' ', '. ', ', ', ''],
    chunk_size = 1000,
    chunk_overlap = 0.2
)
```

Step 3. **Create Embeddings**:

```
model_name = "sentence-transformers/all-MiniLM-L6-v2"
embedding_fn = HuggingFaceEmbeddings(model_name=model_name)
vector_path = 'Vector_db/rmlu_Vector_db'
vector_db = FAISS.from_documents(documents=texts, embedding=embedding_fn)
vector_db.save_local(vector_path)
```

Step 4. **User Input**:

```
def RAG_Chain(query):
    print("Generating the response...")
    # Pass the query with the key 'question'
    response = qa_conversation({"question": query})
    return response.get("answer")
```

CHAPTER – 4

BENEFITS

4.1 INTRODUCTION:

This chapter delves into the various benefits provided by the implementation of a Question and Answer (QA) system within a university setting. Such a system, powered by advanced technologies like the GROQ API and Mixtral LLM model, offers numerous advantages that can enhance user interaction and information retrieval. This chapter outlines the key benefits, emphasizing how the QA system can transform user experiences and operational efficiency.

4.2 SAVE TIME AND EFFORT

- **4.2.1 Automation of Information Retrieval**
  - **Explanation:** Traditionally, users spend significant time navigating through complex university websites to find relevant information. With the QA system, users can input their questions directly, and the system will automatically retrieve and provide the answers. This automation drastically reduces the time required to obtain information.
- **4.2.2 Streamlined User Experience**
  - **Explanation:** By simplifying the process of information retrieval, the QA system enhances the user experience. Users no longer need to sift through multiple pages or sections of the website; instead, they receive direct answers to their queries. This streamlined approach improves overall satisfaction and usability.

4.3 IMPROVE ACCURACY

- **4.3.1 Data-Driven Responses**
  - **Explanation:** The QA system is trained on extensive datasets, allowing it to deliver responses that are well-informed and accurate. Unlike human responders who might misinterpret or overlook details, the system provides answers based on a comprehensive understanding of the data.

- **4.3.2 Consistency in Information**
    - **Explanation:** Human responses can vary based on personal knowledge and interpretation. The QA system, however, ensures that every user receives consistent and reliable information. This consistency is crucial for maintaining trust and credibility in the information provided.

4.4 MORE OBJECTIVE

- **4.4.1 Elimination of Personal Bias**
    - **Explanation:** The QA system operates without personal biases or subjective opinions. It provides information based purely on the data it has been trained on, ensuring that users receive objective answers that are not influenced by individual beliefs or preferences.
- **4.4.2 Standardization of Answers**
    - **Explanation:** The system delivers standardized responses, which helps in maintaining uniformity across various interactions. This standardization is especially valuable in academic settings where precise and impartial information is essential.

4.5 SCALABLE

- **4.5.1 Handling High User Volume**
    - **Explanation:** As the university grows or experiences high levels of user interaction, the QA system can scale to accommodate increased demand. This scalability ensures that the system remains effective and responsive even during peak usage times.
- **4.5.2 Adaptability for Growth**
    - **Explanation:** The infrastructure of the QA system is designed to be adaptable, allowing for future enhancements and the ability to handle more complex queries as the university's needs evolve. This adaptability is key to long-term sustainability and relevance.

4.6 SECURE

- **4.6.1 Privacy Protection**
    - **Explanation:** Protecting users' personal data is a top priority for the QA system. It employs robust security measures to ensure that sensitive information remains confidential and secure from unauthorized access.
- **4.6.2 Compliance with Security Standards**
    - **Explanation:** The QA system adheres to established security standards and regulations, ensuring that it meets legal and ethical requirements for data protection. This compliance is critical for maintaining user trust and meeting institutional requirements.

CHAPTER – 5

RESULT

5.1 INTRODUCTION:

This chapter details the outcomes and effectiveness of the project, focusing on the performance and capabilities of the implemented QA system. The project aimed to create a robust and efficient system capable of answering a wide range of university-related questions. This chapter evaluates the success of the project based on the functionality, technology, and results achieved.

5.2 PROJECT SUCCESS AND FUNCTIONALITY

- **5.2.1 Successful Question Answering**

Explanation: The QA system successfully addressed various queries related to the university. It demonstrated its ability to comprehend and respond accurately to questions across different topics.

- **5.2.2 Handling Diverse Question Types**

Explanation: The system was capable of managing various types of questions, including:

Factual Questions: Inquiries seeking specific information or data.

Conceptual Questions: Queries requiring explanation of concepts or ideas.

Open-Ended Questions: Questions that invite more elaborate or detailed responses.

- **5.2.3 Accurate and Relevant Responses**

Explanation: The system consistently provided precise and pertinent answers, ensuring that users received reliable information relevant to their queries.

- 5.3 <u>TECHNOLOGY IMPLEMENTATION</u>

  - **5.3.1 LangChain**

Explanation: LangChain is utilized to build and manage the QA system, leveraging large language models to process and generate responses based on user questions.

  - **5.3.2 Groq API**

Explanation: The GROQ API facilitates access to OpenAI's large language models, enabling the system to generate high-quality text and responses by utilizing advanced AI capabilities.

  - **5.3.3 Streamlit Python**

Explanation: Streamlit Python framework is employed to create an interactive web application, providing a user-friendly interface for both technical and non-technical users to interact with the QA system.

  - **5.3.4 Vector Database**

Explanation: A vector database is used to efficiently store and retrieve vectorized text data. This database supports quick access to relevant text chunks, enhancing the system's performance and responsiveness.

  - **5.3.5 Large Language Model (LLM)**

Explanation: The LLM is the core of the QA system, trained on extensive text datasets to generate and process natural language responses accurately.

- 5.4 <u>RESULTS ACHIEVED</u>

  - **5.4.1 Effective Question Handling**

Explanation: The system efficiently addressed a wide range of questions, demonstrating its versatility and reliability in handling different types of inquiries.

  - **5.4.2 User-Friendly Interface**

Explanation: The system's design ensures ease of use, making it accessible to users with varying levels of technical expertise.

- **5.4.3 Scalability**

Explanation: The QA system is designed to accommodate a high volume of users and queries, ensuring that it remains effective even during peak usage times.

- **5.4.4 Robustness and Error Handling**

Explanation: The system is robust, capable of managing errors and unexpected inputs gracefully, thereby maintaining a high level of performance and reliability.

- **5.4.5 Security and Privacy**

Explanation: The system incorporates security measures to protect user data, ensuring that privacy is maintained and sensitive information is safeguarded.

CHAPTER - 6

LIMITATIONS

6.1 INTRODUCTION:

While the QA system project achieved significant success and met its primary objectives, there are several limitations that must be acknowledged. These limitations highlight areas for potential improvement and adaptation to enhance the system's functionality and applicability.

6.2 DEVELOPMENTAL LIMITATIONS

- **6.2.1 Need for Enhanced Accuracy**
  - **Explanation:** The current system's accuracy could be improved by utilizing a larger and more diverse dataset. A more extensive dataset would help the model better understand and respond to a wider range of queries. Additionally, integrating more advanced algorithms or refining existing ones could further enhance the system's performance.
- **6.2.2 Efficiency of Vector Database**
  - **Explanation:** The vector database currently in use may not be the fastest or most efficient available. Upgrading to a more performant vector database could reduce response times and improve the overall efficiency of the system. This would enhance user experience, particularly during high-demand periods.

6.3 DATA DYNAMICITY

- **6.3.1 Static Data Handling**
  - **Explanation:** The system relies on the data available at the time of its implementation. If the underlying data on the website changes or is updated, the system does not automatically reflect these changes. This limitation means that the system may provide outdated information if the source data is modified without corresponding updates to the system.

6.4 LANGUAGE SUPPORT

- **6.4.1 English-Only Availability**
  - **Explanation:** Currently, the system supports only English, which may limit its accessibility for users who speak other languages. Extending the system to support additional languages, such as Hindi, would increase its usability and make it more inclusive for a broader audience.

CHAPTER - 7

FUTURE RECOMMENDATIONS

7.1 INTRODUCTION:

This chapter outlines several recommendations for enhancing the QA system to address its current limitations and expand its functionality. These suggestions aim to improve accuracy, efficiency, versatility, user experience, and security, making the system more effective and accessible for a broader audience.

7.2 USE A LARGER AND MORE DIVERSE DATASET OF TEXT

- **7.2.1 Expand Dataset Scope**
  - **Explanation:** Incorporating a more extensive and varied dataset would enhance the system's ability to understand and generate responses. A larger dataset can improve the model's vocabulary, grammar comprehension, and its ability to grasp the nuances of human language.
- **7.2.2 Enhance Language Nuance Understanding**
  - **Explanation:** With a diverse dataset, the system can better understand and respond to a wider range of language styles, contexts, and subtleties, leading to more accurate and contextually appropriate answers.

7.3 USE A FASTER VECTOR DATABASE

- **7.3.1 Optimize Data Retrieval**
  - **Explanation:** Upgrading to a faster vector database can significantly improve the efficiency of text chunk retrieval. This would reduce response times, making the system quicker in answering user queries.
- **7.3.2 Improve System Performance**
  - **Explanation:** A more efficient vector database would enhance overall system performance, particularly under high user loads, ensuring consistent and rapid responses.

7.4 EXTEND THE SYSTEM TO ANSWER QUESTIONS ABOUT OTHER DOCUMENT TYPES

- **7.4.1 Support for Various Document Formats**
  - **Explanation:** Expanding the system's capabilities to include other document types, such as Word documents and HTML files, would increase its versatility. This enhancement would allow the system to handle a broader range of queries and provide answers based on different document formats.

- **7.4.2 Enhance Usefulness Across Applications**
    - **Explanation:** By supporting additional document types, the system would become more useful for various applications, including academic research, business documentation, and web content analysis.

7.5 <u>EXTEND THE SYSTEM TO ANSWER QUESTIONS IN OTHER LANGUAGES</u>

- **7.5.1 Multilingual Support**
    - **Explanation:** Adding support for multiple languages, such as Hindi, would make the system more accessible to non-English speaking users. This would broaden the system's user base and enhance its global usability.
- **7.5.2 Increase Inclusivity**
    - **Explanation:** Multilingual capabilities would ensure that users from diverse linguistic backgrounds can effectively utilize the system, promoting inclusivity and accessibility.

7.6 <u>IMPROVE THE USER INTERFACE</u>

- **7.6.1 Enhance User Experience**
    - **Explanation:** Refining the user interface with features such as a search bar, glossary, and help section would improve usability. A more intuitive and user-friendly design would facilitate easier interaction with the system.
- **7.6.2 Provide Additional Resources**
    - **Explanation:** Including resources like a glossary and help section can assist users in navigating the system and understanding the information provided, enhancing overall user satisfaction.

7.7 <u>MAKE THE SYSTEM MORE SECURE</u>

- **7.7.1 Encrypt User Data**
    - **Explanation:** Implementing data encryption would protect user privacy and secure sensitive information from unauthorized access. This is crucial for maintaining trust and compliance with data protection regulations.
- **7.7.2 Follow Security Best Practices**
    - **Explanation:** Adopting industry-standard security practices, such as regular security audits and updates, would further safeguard the system against potential vulnerabilities and threats.

CHAPTER - 8

CONCLUSION

8.1 INTRODUCTION:

The project aimed to develop a conversational QA system designed to address queries related to the university. Utilizing advanced technologies, the system has successfully achieved its objectives and demonstrates significant potential for integration into the university's website.

8.2 SUMMARY OF ACHIEVEMENTS

- **8.2.1 Effective Question Answering**
    - **Explanation:** The QA system effectively answers a broad range of questions related to the university. It handles various types of queries, including factual, conceptual, and open-ended questions, providing accurate and relevant responses.
- **8.2.2 Technological Implementation**
    - **Explanation:** The project leverages several innovative technologies:
        - **LangChain:** For building and managing the QA system using large language models.
        - **Groq API:** To access and utilize OpenAI's advanced language models.
        - **Streamlit and Python:** For developing an interactive and user-friendly web application.
        - **Vector Database:** For efficient storage and retrieval of vectorized text data.
        - **LLM (Large Language Model):** For generating and understanding natural language responses.
- **8.2.3 User Accessibility and Efficiency**
    - **Explanation:** The system enhances accessibility by allowing users to query information in natural language, thus avoiding the need to navigate through multiple web pages. This functionality improves user experience and streamlines information retrieval.

- **8.2.4 Automation of Tasks**
    - **Explanation:** The system automates tasks that are typically performed by humans, such as summarizing lengthy documents and extracting specific information. This automation increases efficiency and reduces the manual effort required for these tasks.

8.3 POTENTIAL IMPACT AND APPLICATIONS

- **8.3.1 University Website Integration**
    - **Explanation:** The QA system is well-suited for integration into the university's website, where it can serve as a valuable tool for current and prospective students, faculty, and staff seeking information.
- **8.3.2 Broader Applications**
    - **Explanation:** Beyond the university context, the system's capabilities can be extended to various applications, including customer support, educational tools, and research assistance, demonstrating its versatility and potential for widespread use.

8.4 FUTURE DIRECTIONS

- **8.4.1 Continuous Improvement**
    - **Explanation:** To further enhance the system, ongoing improvements are necessary, including expanding the dataset, optimizing the vector database, supporting additional document types and languages, and refining the user interface and security features.
- **8.4.2 Expansion and Adaptation**
    - **Explanation:** The system has the potential for expansion into other domains and use cases, adapting to different types of organizations and information needs. Future developments could explore these opportunities to broaden the system's impact and functionality.

REFERENCES

1.**Github** : https://github.com/langchain-ai/langchain

2.**LLM** : https://en.wikipedia.org/wiki/Large_language_model#:~:text=A%20large%20language%20model%20(LLM,mostly%20scraped%20from%20the%20Internet

3.**Streamlit**: https://streamlit.io/

4.**LangChain**: https://python.langchain.com/docs/use_cases/question_answering/

https://pub.towardsai.net/understanding-langchain-%EF%B8%8F-part-198499559f4c4

5. **Groq API**: https://console.groq.com/docs/quickstart