

**Project report**  
**on**  
**Design of a Smart Memory Device for Speech and Text Processing**  
*Bachelor of Technology (Hons.)*  
*in*  
*Electronics & Communication Engineering*  
*by*  
**SOMESH GHOSH (2022UGEC013)**  
*Under the supervision of*  
**Dr. BASUDEBA BEHERA**  
(Assistant Professor, Department of ECE, NIT Jamshedpur)



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**  
**NATIONAL INSTITUTE OF TECHNOLOGY JAMSHEDPUR 2024**

## **CANDIDATE’S DECLARATION**

We hereby declare that

- A. The work contained in this report is original and has been done by me under the guidance of our supervisor Dr Basudeba Behera, Assistant professor, Department of Electronics, NIT Jamshedpur.
- B. The work has not been submitted to any other Institute for any degree or diploma.
- C. I have followed all the guidelines provided by the Institute in preparing the report.
- D. I have conformed to the norms and guidelines given in the Ethical Code of Conduct of my Institute.
- E. Wherever I have used materials (data, theoretical analysis, figures and texts) from other sources, I have given due credit to them by citing them in the project report and giving their details in the reference. Further I have taken permission from the copyright owners of the sources, wherever necessary.

### **Signature of the Student**

Somesh Ghosh  
2022UGEC013

# CONTENTS

Title Page.....	1
Content.....	3
Abstract.....	4
Chapter 1      Overview	
1.1 Introduction.....	5
1.2 Motivation.....	5
1.3 Literature Review.....	6
Chapter 2      Methodology and Work done So Far	
2.1 Introduction to Smart Memory Devices.....	6
2.2 Core Components and Technologies.....	6
2.3 System Architecture and Modular Design.....	7
2.4 Use of Python.....	8
2.5 Role of Raspberry Pi.....	8
2.6 Wifi Module of Raspberry Pi.....	8
2.6 Data Storage and Security... ..	9
Chapter 3      Working and Methodology.....	10
Chapter 4      4.1 Project Reflections and Learnings.....	17
4.2 Conclusion.....	17
Chapter 5      References.....	18

## **Abstract**

This project focuses on the design and implementation of a Smart Memory Device utilizing Python and Large Language Models (LLMs) to perform advanced speech and text processing. The system efficiently converts speech to text and text to speech, storing processed data securely for future use. It leverages pre-trained models like SpeechBrain's speaker recognition system to ensure high accuracy and robustness in speech-based applications.

The device is designed to enhance user interaction by providing seamless conversions with minimal latency, ensuring an optimized user experience. It prioritizes efficient memory management, power-saving techniques, and scalability, making it suitable for diverse applications. The system's modular design allows easy integration with other technologies and software frameworks.

**Keywords:** Speech to Text, Text to Speech, Speaker Recognition, Large Language Models, Data Storage, Vector Database, Python, User Interaction.

**Software Used:** Python 3.x, SpeechRecognition Library, pyttsx3, SpeechBrain, PyCharm IDE, Langchain

**LLMs and Vector DB used:** Ollama (llama3), FAISS, *all-MiniLM-L6-v2* embeddings

**Supervisor**

Dr. Basudeba Behera  
(Asst. Professor)

# Chapter 1

## OVERVIEW

### 1.1 Introduction

Memory-related challenges, such as those faced by people with dementia, require innovative solutions to enhance communication and day-to-day functioning. This project focuses on designing a Smart Memory Device that leverages Python and advanced technologies like speech recognition and text-to-speech conversion. The device aims to bridge communication gaps by converting spoken words to text and vice versa, while securely storing information for future reference. Such a system can help individuals with dementia recall important conversations, access stored information, and interact more effectively with their surroundings.

### 1.2 Motivation

Individuals with dementia often struggle with memory loss, making it difficult to recall events, conversations, or important details. This project is motivated by the need to provide an accessible tool that supports their memory and communication. By using advanced speaker recognition and efficient memory management, this system offers a reliable way to store and retrieve data, empowering individuals and their caregivers to maintain a better quality of life.

## 1.3 Literature Review

### Literature Review

The Smart Memory Device leverages cutting-edge technologies and methodologies, drawing inspiration and insights from various references to enhance communication and memory recall for individuals with cognitive challenges. The integration of Large Language Models (LLMs) like Llama3, as discussed in the article *"Llama3 RAG on Google Colab"*, provides robust retrieval-augmented generation (RAG) capabilities for efficient data retrieval and context-aware responses, essential for intelligent memory systems. LangChain, as introduced in its documentation, serves as a powerful framework for building applications that utilize LLMs and vector databases, enabling seamless orchestration of the device's memory management and text-processing functionalities.

In addition, the practical implementation of generative AI on edge devices, as described in *"Llama in a Box: Running Generative AI out of a Raspberry Pi"*, highlights the feasibility of deploying resource-intensive AI models on compact hardware like the Raspberry Pi. This aligns with the project's goal of developing a portable and scalable memory device. The guide *"Setting up Raspberry Pi"* provides foundational steps for configuring the Raspberry Pi, ensuring a smooth hardware integration process. Furthermore, *"Configuring Microphone and Speaker to Raspberry Pi"* offers valuable instructions for enabling audio input and output, critical for the speech-to-text and text-to-speech functionalities.

These references collectively inform the design and development of the Smart Memory Device, combining the latest in AI-driven text and speech processing with practical implementation strategies on hardware platforms like Raspberry Pi to create a robust and user-friendly assistive tool.

# Chapter 2

## THEORETICAL STUDY

### 1. Introduction to Smart Memory Devices

Smart Memory Devices are electronic systems designed to help users remember, store and recall information effectively. These devices focus on storing data from user interactions (like conversations) and providing easy access later. Our Smart Memory Device uses speech processing to capture spoken words, convert them to text and securely save them for later use. This approach enhances user interaction by allowing people to talk to the device and retrieve information verbally, making it a helpful tool for personal memory assistance.

### 2. Core Components and Technologies

Our project relies on specific components and technologies to achieve smooth and accurate speech and text processing:

- **Speech-to-Text Conversion:** This component allows the device to "listen" to spoken words and convert them to written text. Using the SpeechRecognition library in Python, the device captures audio through a microphone and processes it into text that can be stored or used for further functions. This feature is useful for capturing quick notes, ideas or reminders without needing to type.
- **Text-to-Speech Conversion:** The Text-to-Speech (TTS) function, powered by the `pyttsx3` library in Python, allows the device to "speak" the text. For example, when the user wants to retrieve stored information, the device reads the stored text aloud. The TTS function makes the device interactive and user-friendly, as it can respond verbally to the user's requests.

- **Speaker Recognition:** Speaker recognition, enabled by SpeechBrain's pre-trained models, allows the device to identify individual users based on their voice. This technology helps ensure that personal data is secure and can be associated with specific users, adding an extra layer of security. It also allows the device to adapt to different users' preferences or settings.

### 3. System Architecture and Modular Design

The architecture of the device is modular, meaning that each function (speech-to-text, text-to-speech, speaker recognition) works as an independent part within the system. This modular design allows flexibility, making it easy to add new features or connect to other systems in the future.

The device's components are arranged in a sequence:

1. **Audio Input:** The user speaks into a microphone.
2. **Speech Processing:** The audio is processed into text.
3. **Data Storage:** The text is stored securely.
4. **Output (if requested):** The device can read out the stored information.

Each module communicates with others but can function separately, ensuring easy upgrades or feature expansions.

### 4. Use of Python

Python was chosen because it has libraries for everything this project needs: speech recognition, text-to-speech and speaker identification. Python is also highly compatible with Raspberry Pi and is widely used for projects involving artificial intelligence and machine learning. Using Python 3.x with PyCharm as the IDE helped in writing, testing and debugging the code efficiently, ensuring a smooth development process.



## 5. Role of Raspberry Pi

Raspberry Pi serves as the hardware foundation of the project, functioning as the main "brain" of the device. This compact, affordable computer is capable of running Python programs efficiently, making it ideal for portable, low-power applications like this one. It provides the following features and benefits:

- **Flexibility:** The Raspberry Pi allows seamless integration with external sensors and hardware components, enabling future expansions such as connecting new sensors or cloud storage services.
- **Efficiency:** Its ability to handle computational tasks at a low cost and power makes it a perfect fit for IoT projects and edge computing.

## 6. WiFi Module of Raspberry Pi

The Raspberry Pi, particularly models such as the Raspberry Pi 3, 4, and Zero W, comes equipped with built-in WiFi functionality. This plays a crucial role in the project:

- **Wireless Connectivity:** Enables the device to connect to the internet or local networks without requiring additional hardware, ensuring smooth communication with APIs and cloud services.
- **Remote Access and Control:** Facilitates remote interaction with the device for monitoring, updating code, or troubleshooting via SSH or VNC.
- **Data Transfer:** Supports the transfer of data, such as embeddings, to and from cloud storage or external databases, ensuring scalability and accessibility.

The WiFi model of the Raspberry Pi eliminates the need for Ethernet cables, increasing portability and ease of deployment in various environments. This wireless capability complements the flexibility of the project and ensures real-time functionality for the smart memory device

## 7. Data Storage and Security

The system securely stores the processed text data. Secure storage means data is protected against unauthorized access, ensuring user privacy. Different storage options could be considered:

- **Local Storage on the Raspberry Pi:** Data is stored directly on the Raspberry Pi's memory, suitable for small data sizes.
- **Cloud Storage:** If more storage is needed, data could be uploaded securely to cloud storage, making it accessible from other devices as well.

## Chapter 3

### WORKING AND METHODOLOGY:

#### 1. Wrote and tested the code in PyCharm.

- Installed and configured Python in the PyCharm IDE to create the development environment.
- Used the terminal in PyCharm to install required libraries
- Divided the project into individual modules and tested each one separately:

**Speech-to-Text:** Wrote code to capture audio input and convert it to text using [SpeechRecognition](#).

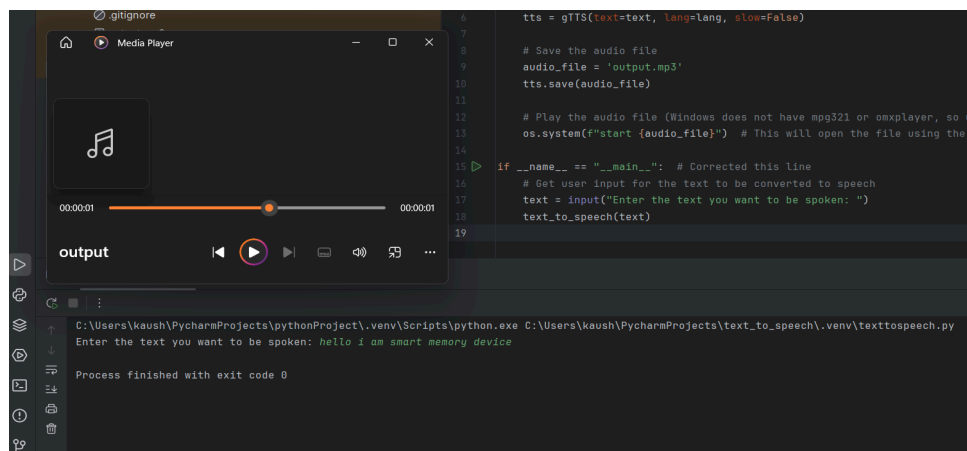
**Voice Matching:** Developed a voice-matching feature using [SpeechBrain](#) (if applicable) for speaker verification.

**Text-to-Speech:** Implemented a function using [gTTS](#) to convert text to spoken audio.

```
Please say your name.
Listening...
Verification score: tensor([0.9425])
User verified successfully.
Listening for commands... (Speak within 5 seconds or say 'stop listening')
You said: hello hello we are in room
Listening for commands... (Speak within 5 seconds or say 'stop listening')
Sorry, I could not understand the audio.
No speech detected for 5 seconds. Stopping.

Process finished with exit code 0
```

FIG 1: SPEECH TO TEXT OUTPUT



```
tts = gTTS(text=text, lang=lang, slow=False)
# Save the audio file
audio_file = 'output.mp3'
tts.save(audio_file)

# Play the audio file (Windows does not have mpg321 or omxplayer, so use
os.system(f"start {audio_file}") # This will open the file using the d

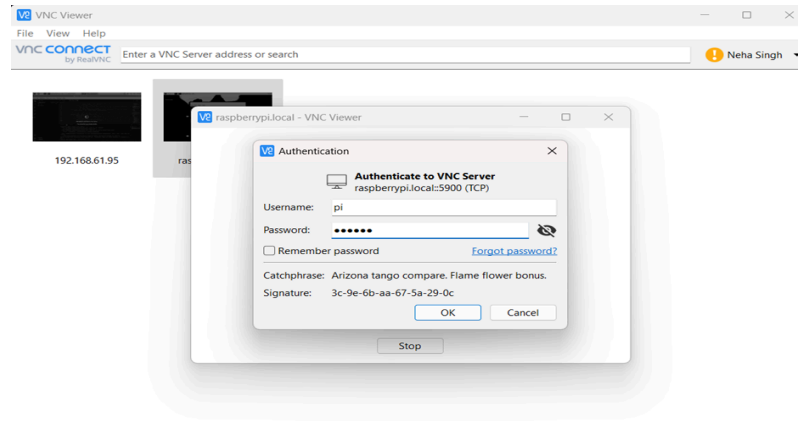
if __name__ == "__main__": # Corrected this line
# Get user input for the text to be converted to speech
text = input("Enter the text you want to be spoken: ")
text_to_speech(text)
```

C:\Users\kaush\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\kaush\PycharmProjects\text\_to\_speech\venv\texttospeech.py  
Enter the text you want to be spoken: hello i am smart memory device  
Process finished with exit code 0

FIG 2: TEXT TO SPEECH OUTPUT

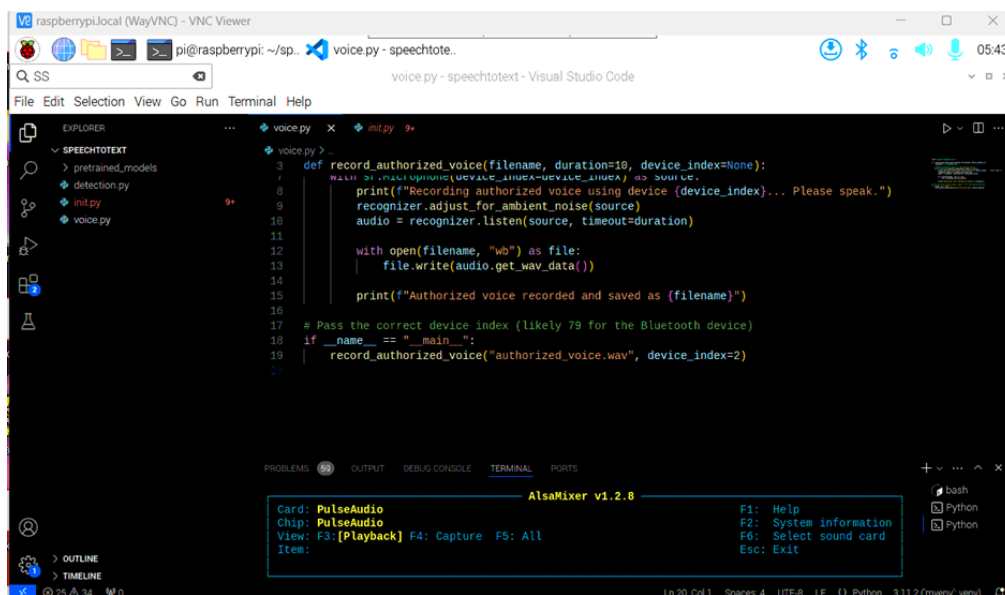
- Debugged each module in PyCharm using its debugging tools and fixed syntax or logic errors as needed.

## 2. Transferred the tested code to the Raspberry Pi.

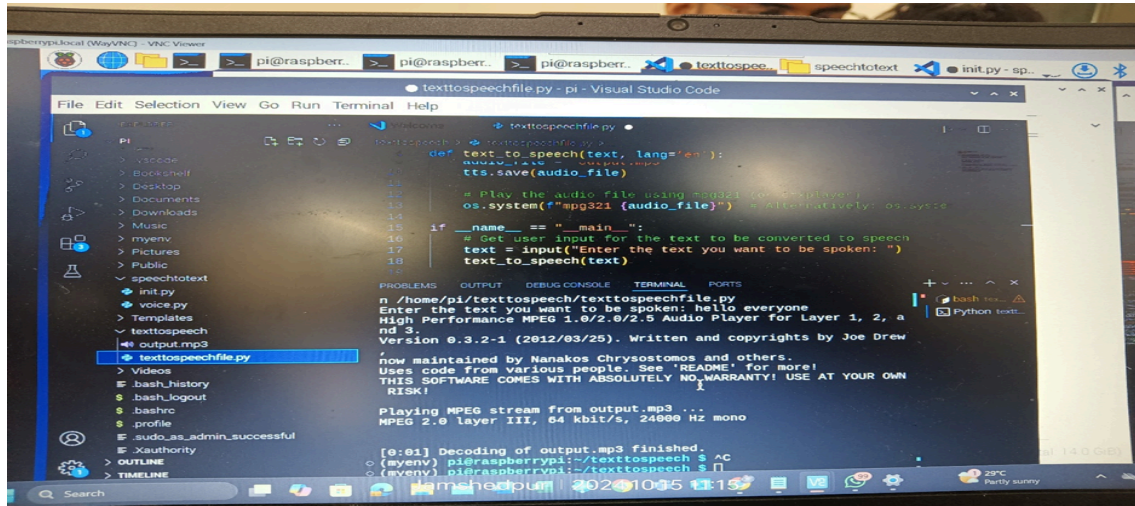


**FIG 3: AUTHENTICATION AND SETUP OF RASPBERRY PI OS**

- Used `scp` to transfer the Python files from the development machine to the Raspberry Pi over SSH.
- Alternatively, copied the files using a USB drive and moved them to the appropriate directory on the Pi.
- Installed the necessary Python libraries on the Raspberry Pi using `pip3`:
- Ensured the Python version on the Raspberry Pi matched the one used during development.



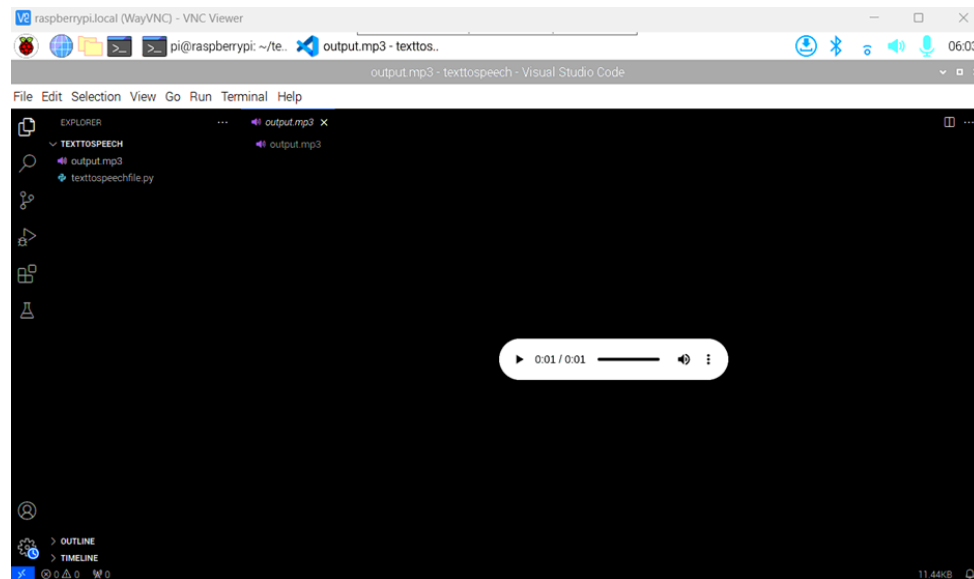
**FIG 4: SPEECH TO TEXT IN RASPBERRY PI OS**



**FIG 5 : TEXT TO SPEECH OUTPUT IN RASPBERRY PI OS**

### 3. Tested the hardware components on the Raspberry Pi.

- Verified microphone functionality by recording audio using the following
- Checked Raspberry Pi GPIO settings (if needed) to ensure correct connections.
- Ran and debugged the code on the Raspberry Pi.
- Executed the Python code using the terminal on the Raspberry Pi
- Encountered and tried to fix any platform-specific errors, such as incorrect file paths, permission issues for hardware access or missing dependencies.
- Added print statements and logging throughout the code to trace its execution and debug effectively.



**FIG : GENERATION OF AUDIO FILE AS OUTPUT**

#### 4. Working of the LLM model in this project

- User Interaction:

The user interacts with the system via prompts or questions. If the input is a statement or information, it is stored. If it is a question, the system processes it to generate a response.

- Storing Information:

The provided information is divided into smaller parts or chunks for efficient storage and retrieval. Each chunk is converted into a vector embedding (a numerical representation in floating-point format) using a pre-trained embedding model. These embeddings are stored in the Faiss database, a highly efficient vector database developed by Meta, optimized for similarity search and retrieval tasks.

- Retrieving Information:

When a question is asked, the input is also converted into a vector embedding. The Faiss database performs a similarity search to find the most relevant stored chunks based on the question embedding.

- Generating a Response:

The retrieved chunks, along with their context, are passed to the LLM (Ollama in your case) as input. The LLM generates a response by using the provided context and relevant chunks.

- Handling Unknowns:

If the Faiss database cannot find relevant information, the system admits it does not know the answer.

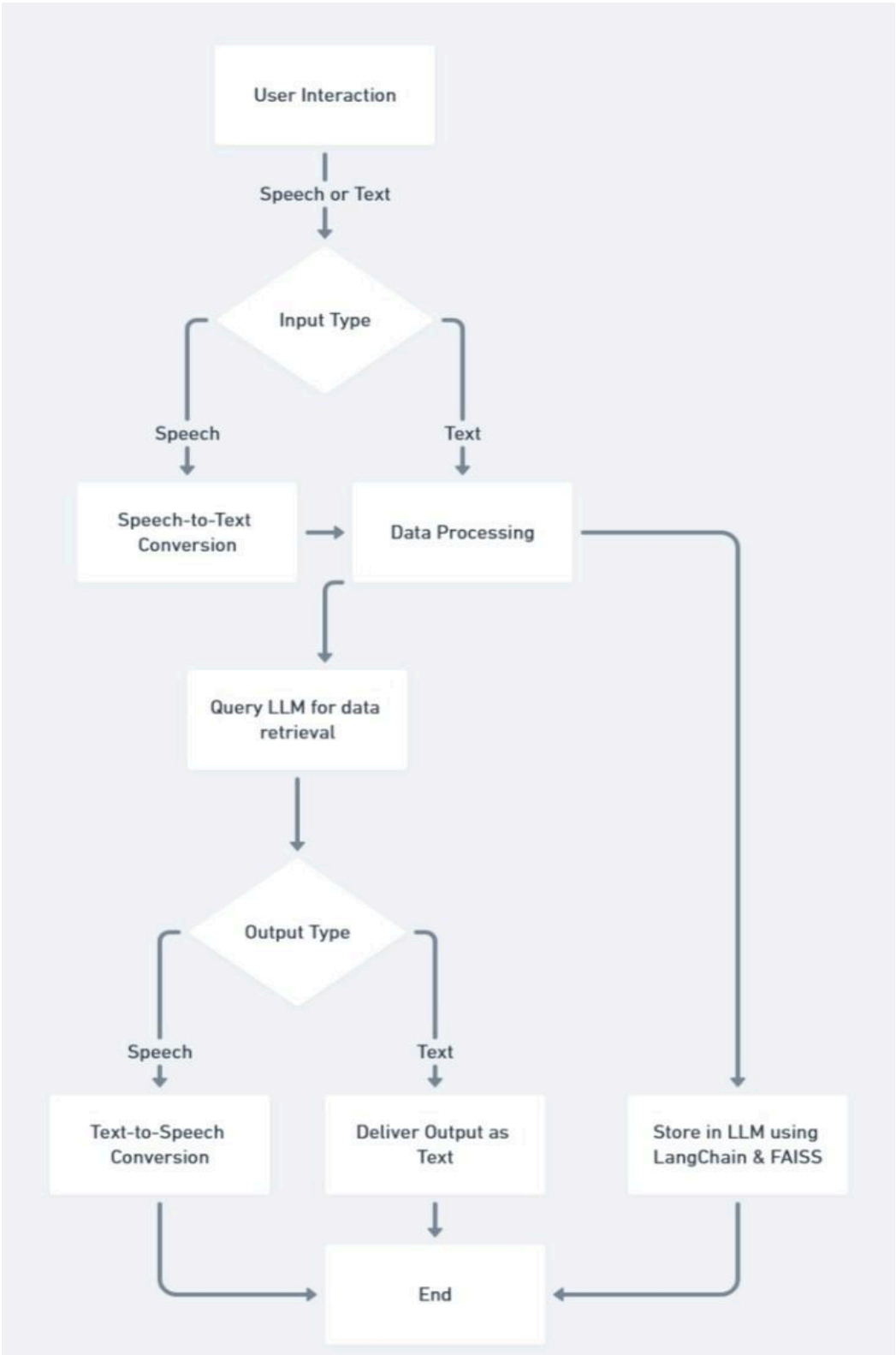
- LangChain Integration:

LangChain acts as a framework to manage interaction between the vector database, the embedding model and the LLM. It simplifies workflows such as chunking, vectorization, retrieval and context injection into the LLM.

This architecture allows your system to dynamically store, retrieve and generate responses in a context-aware manner, combining the efficiency of Faiss with the capabilities of the Ollama LLM.



# FLOWCHART





## Chapter 4

### Project Reflections and Learnings:

Through this project, we gained valuable insights into the practical application of language models and vector databases. Despite facing initial challenges with speech input on the Raspberry Pi, we successfully implemented our model on Google Colab. This experience improved our skills in prompt engineering, enabling us to effectively communicate with the language model and receive desired responses. Additionally, we delved into the intricacies of vector database operations, learning how to optimize storage and retrieval for efficient performance.

We developed a deeper understanding of the trade-offs between model complexity, computational resources, and performance. We also explored the challenges of deploying AI models on edge devices, such as power consumption and latency. By iteratively refining our approach and troubleshooting technical issues, we cultivated a problem-solving mindset essential for IoT and AI development.

This project has equipped us with a solid foundation in Internet of Things (IoT), AI and natural language processing, inspiring us to further explore the exciting possibilities of these technologies and their potential to revolutionize various industries.

### Conclusion:

While implementing AI, particularly large language models, on resource-constrained devices like the Raspberry Pi 4 presents challenges, such as speech input and computational limitations, cloud-based platforms like Google Colab offer a viable solution. By leveraging the computational power and resources of cloud environments, we can effectively train and deploy AI models, overcoming the constraints of local hardware. As AI continues to advance and cloud infrastructure becomes more accessible, we can expect to see even more innovative applications at the intersection of IoT and AI.

## Chapter 5

### References

1. <https://medium.com/@tharindumadhusanka99/llama3-rag-on-google-colab-73c43aa53281> [ Llama 3 RAG on Google Collab ]
2. <https://python.langchain.com/docs/introduction/> [ Lang Chain ]
3. <https://geeksheek.medium.com/llama-in-a-box-running-generativeai-out-of-a-raspberry-pi-4-81d9dfc7630e> [ Running GenerativeAI out of Raspberry Pi ]
4. <https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up/3> [ Setting up Raspberry Pi ]
5. <https://github.com/microsoft/IoT-For-Beginners/blob/main/6-consumer%2Flessons%2F1-speech-recognition%2Fpi-microphone.md> [ Configuring Microphone and Speaker to Raspberry Pi ]