

**InfiniteVoix - An AI based voice
assistant for Desktop**

Submitted in partial fulfillment of the
requirements for the award of
Bachelor of Engineering Degree in Computer Science and Engineering

By

**KEVIN SAMSON R (Reg.No - 40110588)
KARTHIKEYAN S (Reg.No - 40110559)**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Category-1 University by UGC

**Accredited with Grade "A++" by NAAC | 12B Status by UGC | Approved by
AICTE**

**JEPPIAAR NAGAR, RAJIV GANDHISALAI,
CHENNAI - 600119**

APRIL - 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Category - I University by UGC

Accredited "A++" by NAAC | Approved by AICTE

www.sathyabama.ac.in

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **KARTHIKEYAN S(40110559)** and **KEVIN SAMSON R(40110588)** who carried out the Project entitled "**InfiniteVoix - An AI based voice assistant for Desktop**" under my supervision from November 2023 to April 2024.

Internal Guide

Ms.V.GOWRI MANOHARI, M.E.

HEAD OF THE DEPARTMENT
Head of the Department
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
DR. JAKSHYANAN, M.E., Ph.D.
600034, Chennai, Tamil Nadu, India
Cell No: 94421 41015

Submitted for Viva voce Examination held on 12/04/2024

Internal Examiner

External Examiner

DECLARATION

I **KARTHIKEYAN S** (Reg.No- 40110559) hereby declare that the Project Report entitled **InfiniteVoix - An AI based voice assistant for Desktop** done by me under the guidance of **Ms.V.GOWRI MANOHARI, M.E.**, is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

DATE: 13.4.2024

PLACE: Chennai

s.karthikeyan

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of Sathyabama Institute Of Science And Technology** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph. D, Dean**, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D.**, Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Ms.V.GOWRI MANOHARI, M.E.** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

InfiniteVoix stands as a groundbreaking AI-based voice assistant, meticulously crafted to cater specifically to desktop environments, thus reshaping the landscape of user-computer interaction. Departing from its predecessors confined to the realms of smartphones and laptops, InfiniteVoix extends its reach far beyond, seamlessly embedding itself into desktop systems, fostering unparalleled accessibility and convenience across a spectrum of platforms. At its core lies the transformative power of Natural Language Processing (NLP), an intricate application of AI technology that enables InfiniteVoix to adeptly decipher user commands captured through the device's microphone, effortlessly transmuting spoken words into actionable text prompts. Leveraging a sophisticated suite of algorithms, InfiniteVoix swiftly processes these requests, executing tasks with unparalleled efficiency and precision. Beyond its technical prowess and efficiency, InfiniteVoix carries profound implications for differently-abled individuals, offering a lifeline to those navigating the digital landscape with unique challenges. For individuals with mobility impairments, InfiniteVoix serves as an indispensable tool, empowering them to navigate their computers, access information, and perform tasks with unprecedented ease and independence, simply through the power of their voice. Moreover, for those with visual impairments, InfiniteVoix transcends barriers, providing a seamless interface that relies not on sight, but on the innate ability to communicate verbally. Through its intuitive design and robust functionality, InfiniteVoix embodies inclusivity, ensuring that every user, regardless of their abilities, can harness the full potential of desktop computing with unparalleled ease and dignity. In doing so, InfiniteVoix not only revolutionizes the way we interact with technology but also fosters a more equitable and inclusive digital ecosystem for all.

TABLE OF CONTENTS

Chapter No	TITLE	Page No.
	ABSTRACT	v
	LIST OF FIGURES	viii
	LIST OF ABBREVIATIONS	ix
1	INTRODUCTION	1
	1.1 Aim	1
	1.2 Scope	2
2	LITERATURE SURVEY	
	2.1 Inferences from Literature Survey	4
	2.2 Open problems in Existing System	5
3	REQUIREMENTS ANALYSIS	
	3.1 Introduction	6
	3.2 Feasibility Studies/Risk Analysis of the Project	7
	3.3 Software Requirements Specification Document	8
4	DESCRIPTION OF PROPOSED SYSTEM	
	4.1 Selected Methodology or process model	16
	4.2 Architecture / Overall Design of Proposed System	17
	4.3 Description of Software for Implementation and Testing plan of the Proposed Model/System	18
5	IMPLEMENTATION DETAILS	
	5.1 Introduction	20
	5.2 Implementation Of Modules	21

6	RESULTS AND DISCUSSIONS	
	6.1 Results	27
	6.2 Testing and Discussion	28
7	CONCLUSION	
	7.1 Conclusion	29
	REFERENCES	30
	APPENDIX	
	A Source code	31
	B Screenshot	46
	C Research paper	48
	D Certificate	53

LIST OF FIGURES

FIGURE NO	FIGURE NAME	Page No.
3.1	Uses of Python	10
4.1	Overall Architecture	15
4.2	System Architecture	17
1	GUI of InfiniteVoix	46
2	Weather command	46
3	Asking a Question	47

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
AI	Artificial Intelligence
API	Application programming interface
IDE	Integrated development environment
NLP	Natural Language Processing
NLU	Natural Language Understanding
OS	Operating System
TTS	Text-to-Speech
STT	Speech-to-Text
VA	Virtual Assistant

CHAPTER 1

INTRODUCTION

Computers have become a very important devices and as well as less expensive over time. The idea behind the Personal Virtual Assistant is creation of an inexpensive, reliable and easy to use assistant. Virtual Assistant (VA) is a term that applies to computer-simulated environments that can simulate physical presence in places in the real world, as well as in imaginary worlds. Virtual Assistant is a real-time and interactive technology. It means that the computer can detect user inputs and modify the virtual world instantaneously. Interactivity and its captivating power contribute to the feeling of being part of the action on the environment that the user experience. All human sensorial channels can be used to have a high-level interaction. Most current virtual assistant environments are primarily visual experiences, displayed either on a computer screen, but some simulations include additional sensory information, such as sound through speakers or headphones. Virtual Assistant technology has been a promising technology applicable in various domains of application such as training simulators, medical and health care, rehabilitation, education, engineering, scientific visualization, and the entertainment industry. The application works similar like Siri/ Google Assistant. But the application deals with the computer itself mainly. A voice assistant is a digital assistant that uses voice recognition, natural language processing and speech synthesis to provide aid to users through devices and voice recognition applications. This project is based on basic concepts of Artificial Intelligence such as speech recognition, natural language processing.

1.1. AIM

The aim of an AI voice assistant for desktop is to enhance user productivity and streamline daily tasks through the integration of advanced artificial intelligence. By leveraging natural language processing and machine learning, such an assistant can understand and execute a wide range of commands, from searching for information and controlling devices. The goal is to create a seamless interaction

between the user and their computer, allowing for a more intuitive and efficient workflow. The technology not only empowers users with physical disabilities by allowing them to navigate through digital spaces with ease but also supports those with visual impairments through audio feedback and voice-guided navigation. This technology is designed to adapt to individual user preferences and learn from interactions to provide personalized assistance, making it an invaluable tool for both personal and professional use. Furthermore, the potential of AI voice assistants extends beyond individual use. They can be instrumental in educational settings, providing students with disabilities the means to participate fully in academic activities. Ultimately, the AI voice assistant aims to act as a reliable and intelligent companion that augments human capabilities and simplifies the complexities of modern digital life.

1.2. SCOPE

The scope of an AI voice assistant for desktop encompasses a wide range of functionalities aimed at enhancing user productivity and interaction with their computer. These assistants are designed to understand natural language commands and perform tasks such as searching the internet, and controlling devices. They can also provide real-time information like weather updates, news, and traffic alerts. Advanced AI assistants are integrated with machine learning algorithms to learn from user interactions and preferences, thereby offering personalized experiences. These tools are designed to overcome a range of challenges, from physical to cognitive, offering tailored assistance through voice commands. They can automate repetitive tasks, facilitate hands-free operation, and assist users with disabilities through voice commands. Moreover, AI voice assistants for desktops are increasingly becoming more context-aware, capable of understanding the user's current task and providing relevant assistance. They are also being designed to seamlessly integrate with various applications and services, ensuring a cohesive and intuitive user experience. With the integration of AI, traditional assistive devices are evolving into more intelligent and adaptive tools. These advancements promise to expand access and abilities for over a billion people with disabilities worldwide, making everyday tasks more manageable and

fostering greater independence. As technology advances, the scope of these assistants will continue to expand, potentially incorporating more sophisticated features such as emotion recognition, proactive assistance, and complex problem-solving capabilities.

CHAPTER 2

LITERATURE SURVEY

- In this paper entitled “Study of Voice Controlled Personal Assistant Device” author proposed the famous application of iPhone is “SIRI” which helps the end user to communicate end user mobile with voice and it also responds to the voice commands of the user. Same kind of application is also developed by the Google that is “Google Voice Search” which is used for in Android Phones. But this Application mostly works with Internet Connections. This Proposed System has capability to work with and without Internet Connectivity. It is named as Personal Assistant with Voice Recognition Intelligence, which takes the user input in form of voice or text and process it and returns the output in various forms like action to be performed or the search result is dictated to the end user.
- In this paper entitled “AI Based Voice Assistant Using Python” author developed an economically effective and performance wise efficient virtual assistant using Raspberry Pi for home automation based on the concepts of Internet of Things, Speech Recognition, Natural Language Processing and Artificial Intelligence. People who are using it can give voice inputs and the device itself responds through voice commands by itself. It can fetch the date, time, weather, play your favourite music and fetch search results from the internet along with controlling the home appliances.

2.1 INFERENCES FROM LITREATURE SURVEY

- The system should partially work without the internet connection. As internet may not be available all the time, the system should recognize the user’s command and respond to it accordingly.
- The system should implement an effective Natural Language Processing system, so that it can effectively understand what the user trying to do and help them with their task, thus making an effective personal AI assistant.

2.2 OPEN PROBLEMS IN EXISTING SYSTEM

- The system doesn't work without internet. The internet will be mostly available all the time, but the system should handle the user's response even if the internet is not available.
- The system does not have any kind of Artificial intelligence associated with it. Artificial intelligence helps the system to better understand the needs of the user and give the result intelligently. Integrating AI with voice assistant makes a powerful personal assistant which can do many things that a normal voice assistant cannot.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 INTRODUCTION

AI-powered voice assistant for desktops promises to redefine workplace interaction, streamline workflows, and boost user productivity by harnessing the natural power of human speech.

Let's embark on a journey into the core functionalities and capabilities that will underpin this revolutionary assistant, paving the way for a future where voice commands orchestrate digital symphonies:

- **Seamless Natural Language Processing (NLP):** The ability to understand and respond to natural language, akin to conversing with a human colleague, forms the bedrock of this technology. Think dictating emails, scheduling meetings, and managing files, all hands-free.
- **Robust Task Automation:** From controlling applications to launching complex workflows, the assistant should flawlessly execute user's requests, reducing manual input and saving precious time. Imagine initiating data analysis or generating reports with a simple voice command.
- **Advanced Information Retrieval:** Accessing information should be effortless. This requires the assistant to harness the power of search engines and internal databases, providing accurate and relevant responses to user queries, be it market trends, competitor analysis, or project data.
- **Personalized User Experience:** Learning user preferences and adapting to their work style is crucial. The assistant should evolve alongside the user, anticipating needs and proactively offering contextual suggestions. Imagine automatically prioritizing tasks based on deadlines or suggesting relevant documents for ongoing projects.

- **Security and Privacy:** In the professional landscape, data security is paramount. The assistant must be built with robust security protocols to ensure user privacy and data integrity.

3.2 FEASIBILITY STUDIES/RISK ANALYSIS OF THE PROJECT

A feasibility study is a crucial step in determining the viability of a project or system, before committing significant resources to its development. Below is a feasibility study for the InfiniteVoix AI-based voice assistant in a desktop environment:

1. Technical Feasibility:

- **Hardware Requirements:** Evaluate the hardware capabilities of typical desktop computers to ensure they can handle the voice recognition and NLP processing requirements without significant performance degradation. Consider system compatibility and resource constraints.
- **Software Requirements:** Assess the availability of necessary software tools and libraries for NLP, speech recognition, and voice synthesis on desktop platforms. Ensure that existing libraries and APIs are compatible with your project.
- **NLP Accuracy:** Ensure that the NLP technology and voice recognition technology you plan to use are accurate and capable of understanding a wide range of commands and accents. Test for the accuracy of responses and adaptability to user commands.
- **Scalability:** Investigate whether the system can scale effectively to accommodate a growing user base and potential updates or expansions.

2. Operational Feasibility:

- **User Acceptance:** Conduct surveys, focus groups, or user interviews to gauge user interest and acceptance of a desktop voice assistant. Identify potential use cases and gather user feedback for feature development.
- **User Training:** Assess the need for user training to effectively use the voice assistant. Consider the ease of use and intuitive user interface.

Maintenance and Support: Define the operational procedures for system maintenance and user support. Evaluate the cost and resources required for ongoing technical support.

3.3 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT

The AI-powered desktop voice assistant is a symphony of software components playing in perfect harmony. To fulfill its potential, robust and well-defined software requirements are essential. They are:

- Python
- Speech_recognition module
- pyttsx3 module

3.3.1 PYTHON

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library. Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming

language and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000. Python 3.0, released in 2008, was a major revision not completely backward-compatible with earlier versions. Python 2.7.18, released in 2020, was the last release of Python 2. Python consistently ranks as one of the most popular programming languages, and has gained widespread use in the machine learning community.

Since 2003, Python has consistently ranked in the top ten most popular programming languages in the TIOBE Programming Community Index where as of December 2022 it was the most popular language (ahead of C, C++, and Java). It was selected Programming Language of the Year (for "the highest rise in ratings in a year") in 2007, 2010, 2018, and 2020 (the only language to have done so four times as of 2020).

An empirical study found that scripting languages, such as Python, are more productive than conventional languages, such as C and Java, for programming problems involving string manipulation and search in a dictionary, and determined that memory consumption was often "better than Java and not much worse than C or C++"

Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block. Thus, the program's visual structure accurately represents its semantic structure. This feature is sometimes termed the off-side rule. Some other languages use indentation this way; but in most, indentation has no semantic meaning. The recommended indent size is four spaces.

Methods on objects are functions attached to the object's class; the syntax `instance.method(argument)` is, for normal methods and functions, syntactic sugar for `Class.method(instance, argument)`. Python methods have an explicit `self` parameter to access instance data, in contrast to the implicit `self` (or `this`) in some other object-oriented programming languages (e.g., C++, Java, Objective-C,

Ruby). Python also provides methods, often called dunder methods (due to their names beginning and ending with double-underscores), to allow user-defined classes to modify how they are handled by native operations including length, comparison, in arithmetic operations and type conversion.

It is used mainly in:

- scripting language for web applications
- scientific computing
- computer vision
- machine learning
- artificial intelligence

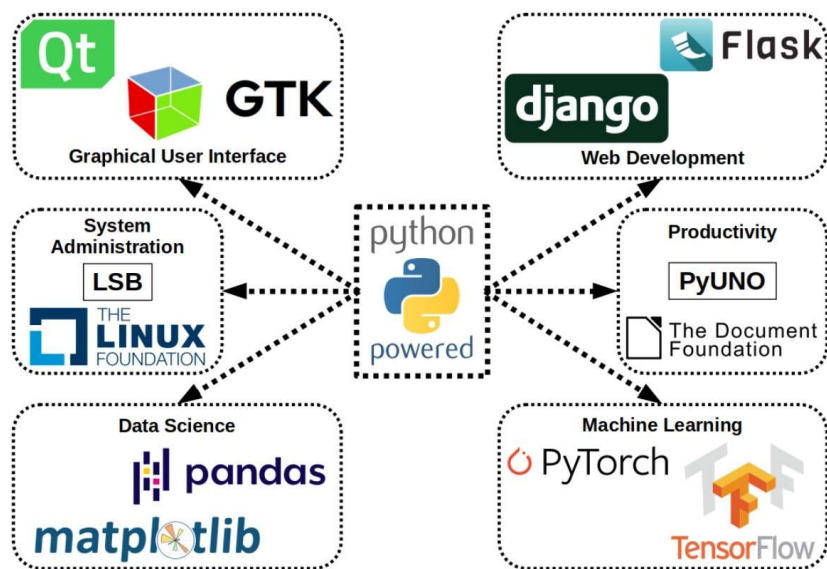


Fig 3.1: Uses of python

3.2.1 *Speech_recognition module*

The `speech_recognition` module empowers Python developers to effortlessly incorporate speech-to-text capabilities into their applications. It offers a user-friendly interface for accessing various speech recognition engines and APIs, both online and offline. This versatility enables developers to select the most suitable engine for their specific task and environment.

Key features of `speech_recognition` include:

- **Engine flexibility:** Supports a range of engines, including Google Speech Recognition, Microsoft Bing Speech, CMU Sphinx, and more.
- **Offline functionality:** Offers offline recognition using CMU Sphinx, allowing for use even without an internet connection.
- **Simple usage:** Provides straightforward methods for recognizing speech from microphones, audio files, and other sources.
- **Noise handling:** Includes features for adjusting to ambient noise, enhancing recognition accuracy in real-world environments.
- **Language support:** Works with multiple languages, catering to diverse user needs.

Common applications of `speech_recognition` include:

- **Voice assistants and chatbots:** Understand and respond to spoken commands and queries.
- **Transcription:** Convert audio recordings to text for documentation, analysis, or accessibility purposes.
- **Voice-controlled applications:** Facilitate hands-free control of software and devices.
- **Accessibility tools:** Enable voice interaction for individuals with physical or visual impairments.

- To harness the power of speech recognition in your Python projects, the `speech_recognition` module provides a comprehensive and accessible toolkit.

3.2.3 *pyttsx3* MODULE

`pyttsx3` shines as a Python's text-to-speech (TTS) champion, offering offline voice magic for both Python 2 and 3. Forget internet jitters, it leverages diverse engines like Sapi5, nsss, and espeak to breathe life into your text, even on the go. This versatile tool lets you choose your perfect voice from a range of tones and languages, then personalize things further with speech rate and pitch adjustments.

Features:

- **Offline Text-to-Speech:** Works without an internet connection, utilizing pre-installed speech engines like Sapi5 (Windows), nsss (Linux), and espeak (all platforms).
- **Multiple Voice Options:** Choose from different voices available on your system, often with variations in gender, accent, and language.
- **Customization:** Adjust speech rate, pitch, and volume to personalize the output.
- **Audio Output:** Generate audio files in various formats (e.g., MP3, WAV) or play the speech directly through your speakers.
- **Simple API:** Easy-to-use functions for initializing the engine, setting properties, and speaking text.
- **Cross-Platform:** Works on Windows, macOS, and Linux.

common applications of the pyttsx3 module:

- Accessibility:
 - Reading text aloud for those with visual impairments or reading difficulties.
 - Providing audio feedback for screen reader software.
 - Creating audio instructions or tutorials for tasks that require visual attention.
- Education:
 - Developing language learning tools for pronunciation practice and comprehension.
 - Creating interactive audio books or flashcards.
 - Building virtual tutors or study aids that can read text aloud and provide feedback.
- Entertainment:
 - Creating audiobooks or narrated stories.
 - Adding voice prompts or narration to games.
 - Designing interactive audio experiences for museums or exhibits.
- Productivity:
 - Building voice-controlled assistants for tasks like setting reminders, checking emails, or controlling smart home devices.

- Creating text-to-speech features for messaging apps or social media platforms.
 - Developing tools for hands-free note-taking or dictation.
- Assistive Technologies:
 - Aiding communication for individuals with speech impairments.
 - Providing audio cues for those with cognitive disabilities.
 - Creating tools for rehabilitation and therapy.
- Additional applications:
 - Generating audio notifications or alerts.
 - Creating spoken CAPTCHAs for website security.
 - Building voice-based interfaces for robots or virtual agents.

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

In this proposed concept effective way of implementing an AI based voice assistant, Speech Recognition library has many in-built functions, that will let the assistant understand the command given by user and the response will be sent back to user in voice, with Text to Speech functions. When assistant captures the voice command given by user, the underlying algorithms will convert the voice into text.

Proposed Architecture

The system design consists of

- Taking the input as speech patterns through microphone.
- Audio data recognition and conversion into text.
- Comparing the input with predefined commands.
- Giving the desired output.

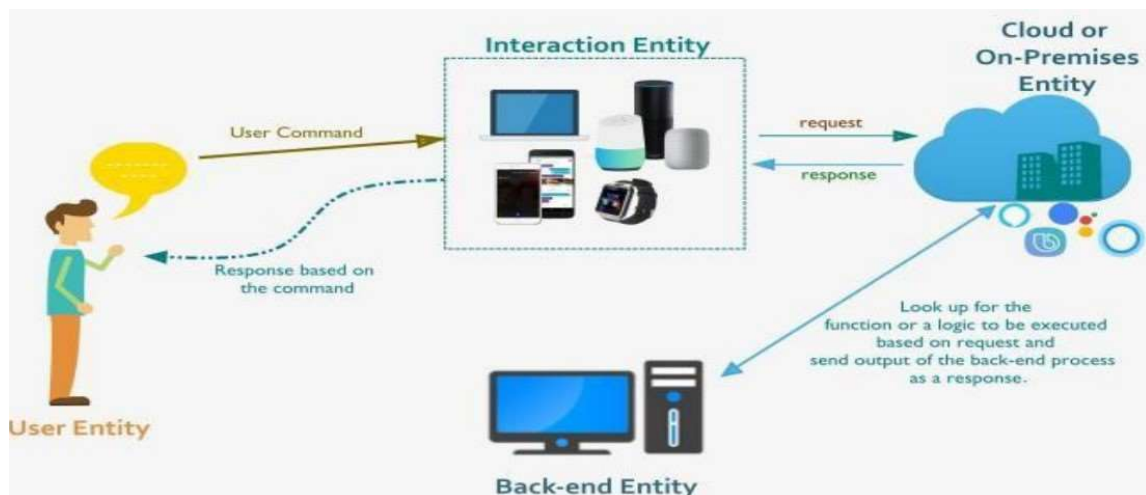


Fig 4.1: Overall Architecture

The initial phase includes the data being taken in as speech patterns from the microphone. In the second phase the collected data is worked over and transformed into textual data using NLP. In the next step, this resulting string the data is manipulated through Python Script to finalize the required output process. In the last phase, the produced output is presented either in the form of text or converted from text to speech using TTS.

Features the system shall be developed to offer the following features:

- It keeps listening continuously in inaction and wakes up into action when called with a particular predetermined functionality.

4.1 SELECTED METHODOLOGY OR PROCESS MODEL

We are going to use python language and google text to speech API for this project, speech recognition module can be used to recognize the voice of used, and based on its query will be fired. Many different modules i.e., web browser, YouTube, Wikipedia, etc. are used to interact with the internet. the os module is used to interact with operating system related queries.

For Learning purposes, users can search any information related to a certain topic on Wikipedia, Google or in text documents. We are using some concepts related to AI and NLP for the processing of text into voice. Our project's main goal is to develop a virtual voice assistant for blind people so they may use it to communicate with emerging technologies, manage their devices, and learn from them.

4.2 ARCHITECTURE / OVERALL DESIGN OF PROPOSED SYSTEM

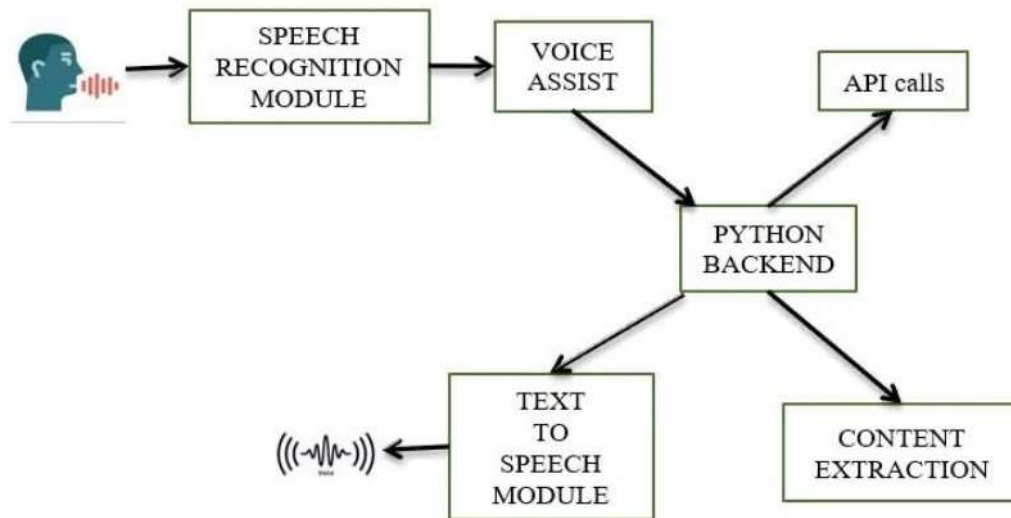


Fig 4.2: System Architecture

- **Speech recognition:** The user speaks their request, which is then captured by the assistant's microphone and converted to text using speech recognition technology.
- **Natural Language Understanding (NLU):** The assistant analyses the user's request to understand the intent behind it. This involves identifying the keywords and context of the request to determine the user's desired action.
- **Intent mapping:** The assistant maps the user's request to a specific action or set of actions. For example, if the user asks "What's the weather like today?", the assistant would map this intent to a weather app and retrieve the relevant information.
- **Action execution:** The assistant executes the mapped action(s) and retrieves the requested information, which is then presented to the user through text or speech.
- **Response generation:** The assistant generates a response to the user's request, which is either spoken aloud or displayed on the screen. This response may include the requested information, confirmation of a

completed action, or an error message if the assistant was unable to fulfil the request.

4.3 DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION AND TESTING PLAN OF THE PROPOSED MODEL/SYSTEM

The system has to:

- Take the input as speech patterns through microphone.
- Audio data recognition and conversion into text.
- Comparing the input with predefined commands.
- Giving the desired output.

These steps are implemented using modules such as Speech recognition module, Natural Language processing module, Text-to-speech conversion module.

4.3.1 *SPEECH RECOGNITION MODULE*

- Speech recognition, or speech-to-text, is the ability of a machine or program to identify words spoken aloud and convert them into readable text.
- A software program turns the sound a microphone records into written language that computers and humans can understand, following these four steps:
 - analyze the audio
 - break it into parts
 - digitize it into a computer-readable format
 - use an algorithm to match it to the most suitable text representation.

4.3.2 NATURAL LANGUAGE PROCESSING MODULE

- Natural Language Processing (NLP) is a branch of artificial intelligence that focuses on enabling computers to understand, interpret, and manipulate human language.
- It combines computational linguistics, machine learning, and deep learning models to process human language in the form of text or voice data.
- NLP allows computers to comprehend the full meaning of text or speech, including the intent and sentiment of the speaker or writer. This technology is used in various applications such as voice-operated GPS systems, digital assistants, speech-to-text dictation software, customer service chatbots, and more.

4.3.3 TEXT TO SPEECH ENGINE

- Text-to-speech (TTS) is a technology that converts text into spoken audio.
- It can read aloud PDFs, websites, and books using natural AI voices.
- TTS technology can be helpful for anyone who needs to access written content in an auditory format, providing a more inclusive and accessible way of communication.

This engine helps the system to communicate with the user.

CHAPTER 5

IMPLEMENTATION DETAILS

5.1. INTRODUCTION

We are going to use python language and google text to speech API for this project, speech recognition module can be used to recognize the voice of user, and based on its query will be fired. Many different modules i.e., web browser, YouTube, Wikipedia, etc. are used to interact with the internet. the os module is used to interact with operating system related queries.

Here's a more detailed breakdown of implementing an AI desktop assistant:

1. Natural Language Processing (NLP) Core: This is the brain of the assistant. It breaks down spoken or typed input into its grammatical components, identifies keywords and phrases, and extracts the intended meaning.
2. Intent Recognition: Machine learning takes center stage here. Trained on vast datasets of user queries and corresponding actions, algorithms like decision trees or neural networks analyze the processed language to determine the user's goal.
3. Action Execution: Once the intent is clear, the assistant needs to take action. This might involve interacting with the user's computer system by opening applications, managing files, or manipulating the desktop. Additionally, it can utilize APIs to perform web searches or interact with online services.
4. Natural User Interface (NUI): To create a seamless experience, the assistant should provide natural ways to interact. Text-to-speech (TTS) functionality allows the assistant to respond to queries in a human-like voice, while speech-to-text (STT) converts spoken instructions into text for processing.

5.2 IMPLEMENTATION OF MODULES

InfiniteVoix is implemented using python and python libraries like speech_recognition module, pyttx3 module etc. To develop the application, we need an IDE to write and execute the code.

5.2.1. IDE SETUP

To develop InfiniteVoix, we are going to use Visual Studio Code as the IDE of the project. Visual Studio Code is a code editor developed by Microsoft, which is embedded with lots of useful extensions and some features to speed up the development process. Using Visual Studio Code has some advantages such as:

Here are some of the advantages of Visual Studio Code:

- **Lightweight and Customizable:** Unlike bulkier IDEs, VS Code starts as a basic editor, making it fast and responsive. You can extend its functionality through a vast marketplace of extensions, tailoring the code editor to your specific needs and workflows.
- **Multi-Language Support:** Out of the box, VS Code offers built-in support for popular languages like JavaScript, Python, and TypeScript. With extensions, you can gain syntax highlighting, code completion, and other language-specific tools for a wide range of programming languages.
- **Cross-Platform:** No matter your operating system (Windows, macOS, or Linux), VS Code runs seamlessly, providing a consistent development environment across your devices.
- **Built-in Git Integration:** For version control, VS Code integrates directly with Git, allowing you to manage your code commits, branches, and merges efficiently within the editor itself.

- **Intelligent Features:** Intellisense, a core feature, provides smart code completion suggestions as you type, helping you write code faster and with fewer errors. Additionally, VS Code offers debugging tools to help identify and fix bugs in your code.
- **Large and Active Community:** A vibrant developer community surrounds VS Code, creating and maintaining extensions that enhance the editor's functionality. This vast pool of resources ensures you can find extensions to fit almost any development need.

5.2.2. LIBRARIES USED

Before we can use the libraries, we should install all the libraries using the pip command:

```
pip install pyttsx3
```

Similarly, we should install the following libraries for the project:

- pyttsx3
- datetime
- speechRecognition
- tkinter
- ecapture
- cv2
- webbrowser
- requests

After installing the above-mentioned libraries, we can proceed to implementing the modules.

5.2.3. INPUT MODULE

The input module consists of speech to text engine, which is used to take the command from the user. Also the user can type the command in the given command box. The speech to text engine is implemented using the `speech_recognition` library. We can get the user's command in the form of voice by:

1. **Library Setup:** We'll use the `SpeechRecognition` library for core functionality and potentially `PyAudio` for microphone access. Install them using `pip install SpeechRecognition` and `pip install pyaudio` (if needed on your system).
2. **Recognizer Instance:** Import the libraries and create a `Recognizer` object from `SpeechRecognition`. This object handles the speech recognition process.
3. **Microphone Access:** To capture live speech, utilize the `Microphone` class from `SpeechRecognition`. Encapsulate the microphone source within a `with` statement for proper resource management.
4. **Listening and Recognizing:** Inside the `with` block, use the recognizer's `listen()` method. This method listens for a set amount of time (or until silence is detected) and stores the captured audio data.
5. **Speech-to-Text Conversion:** Now that we have the audio data, the `recognize_google()` method of the recognizer utilizes Google Speech-to-Text for online conversion (an internet connection is required). This method returns the recognized text, which you can then use for further actions in your Python program.

This recognized text is then fed to the NLP module to decide the action that is to be executed.

5.2.4. NLP MODULE

Natural Language Processing (NLP) is a critical component of AI voice assistants, enabling them to understand and process human language. Implementing NLP in a desktop voice assistant involves several key steps. First, speech recognition technology is used to convert spoken words into text. This is followed by natural language understanding (NLU), where the assistant interprets the user's intent and context from the transcribed text. The system then uses intent recognition to determine the action the user wants to perform, such as setting a reminder or searching the web.

5.2.4.1 INTENT MAPPING

Intents represent a task Assistant needs our Action to carry out, such as some user input that needs processing or a system event that we need to handle. We use intents to help build our invocation and conversation models. When these events occur, the Assistant runtime matches it to the corresponding intent and sends the intent to your Action to process.

When we build Actions, we create user intents that contain training phrases, which extends Assistant's ability to understand even more. Assistant uses our training phrases to augment its NLU when it delegates user requests to our Actions.

When this occurs, Assistant brokers the communication between the user and your Actions, mapping user input to an intent that has a matching language model. Assistant then notifies our Actions of the matched intent, so we can process it within a scene.

When building user intents, we would specify the following elements:

- A Global intent designation defines whether or not the Assistant runtime can match the specified user intent at invocation time as well as during a

conversation. By default, Assistant can match user intents only during a conversation. Only intents that we mark as global are eligible for deep link invocation.

- Training phrases are examples of what a user might say to match the intent. The Assistant NLU (natural language understanding) engine naturally expands these training phrases to include other, similar phrases. Providing a large set of high-quality examples increases the intent's quality and matching accuracy.
- Parameters are typed data that we want to extract from user input. To create a parameter, you annotate training phrases with types to notify the NLU engine that we want portions of user input to be extracted. We can use system types or create our own custom types for parameters.

When the NLU engine detects a parameter match in user input, it extracts the value as a typed parameter, so we can carry out logic with it in a scene. If an intent parameter has the same name as a scene slot, the Assistant runtime automatically fills the scene slot with the value from the intent parameter.

Intent parameters also support "partial" matches. For example, if you specify a type of `DateTime` and the user only provides a date, the NLU still extracts the partial value as a parameter.

5.2.5. *COMMAND EXECUTION*

After successfully finding the intent of the user, we should be able to execute the necessary command. InfiniteVoix will be able to:

- Open camera and take a photo
- Open browser
- Tell the weather
- Answer any question asked by the user

- Take a screenshot

These commands are called according to the user intent.

5.2.6. OUTPUT MODULE

The output of the Assistant is both displayed on the screen and read out aloud. This enables differently abled users to understand what's happening in their system. The output is got from the action run by the assistant whether it's opening an application or answering the question asked by the user. The output is read out aloud using the text to speech engine built into the python.

It is implemented by:

1. Library Selection: We have chose `pyttsx3` for offline speech synthesis. Install the chosen library using `pip install pyttsx3`.
2. Engine Initialization: For `pyttsx3`, import the library and create an engine object using `init()`. This initializes the text-to-speech engine on your system.
3. Text Preparation: Define the text you want to convert to speech as a Python string variable, which is nothing but the output from the action executed.
4. Speech Setup (pyttsx3): With `pyttsx3`, set the property of the engine object. We can adjust properties like speech rate and volume.
5. Text-to-Speech Conversion: For `pyttsx3`, use the `say()` method of the engine object, passing the text string as an argument. This queues the text for speech generation.
6. Playback (pyttsx3): Finally, call the `runAndWait()` method of the engine. This plays the generated speech through your system's default audio device.

CHAPTER 6

RESULTS AND DISCUSSIONS

6.1.RESULTS

Artificial Intelligence (AI) has been a game-changer in creating more inclusive environments, especially with the advent of AI desktop voice assistants designed for differently-abled individuals. These innovative tools are not just about convenience; they are about empowerment and independence.

These assistants utilize advanced natural language processing technologies to understand spoken commands and execute tasks accordingly. They can perform a wide array of actions, from simple ones like opening camera and playing music to more complex tasks such as searching the web. The development of InfiniteVoix uses Python, which provides libraries for speech recognition and text-to-speech conversion, enabling the assistant to listen and respond to users effectively.

The creation of InfiniteVoix assistant not only simplifies daily tasks but also provides accessibility for individuals with disabilities, allowing them to operate computers and other devices using voice commands. This technology is not just limited to personal use; it has implications for business efficiency as well, streamlining operations and enhancing productivity. With the integration of AI, these voice assistants are continuously learning and improving, offering personalized experiences and anticipating user needs over time.

InfiniteVoix aims to bring the comfort of voice assistant to Desktop. We have successfully implemented Natural Language Processing (NLP), an application of AI, to process the commands given to it and executes the commands. The user is able to ask the voice assistant to complete their task using the device's microphone, and the command is subsequently translated into text. The text request is processed after that, and a text response is provided along with any voice assistant work.

6.2. TESTING AND DISCUSSION

The whole system will interpret the inputs from the users, it constructs queries to cloud servers and knowledge sources in order to perform tasks and retrieve information to be output by the response generation models. For testing the whole system, we separated the stages of testing the system into several stages, based on the difficulty and complexity of the new models.

For testing the `speech_recognition` module, we tried speaking at different distances from the microphone and measured the accuracy of the system. Also different persons used the `speech_recognition` system to test how well the system is able to pick each persons' voice and accurately convert to text.

Next to test the NLP module, we tried framing different sentences with same meaning and fed it to system and observed whether the system maps all the sentence to the same intent. Also, we tried giving commands which is not trained on and checked whether it handles such sentences properly.

We also tested whether all the command defined in the system works properly and if there is any mismatch in the actions. The `text_to_speech` engine is also tested by feeding many sentences and testing whether every sentence is read out aloud correctly.

InfiniteVoix has been able to pass all the test and could accurately translate speech to text, map the correct intent, perform the intended action and give the required output.

As InfiniteVoix relies on stable internet connection, the tests results varied upon the speed of the internet. But the 5G technology seems promising and will increase the internet speed and, in the future, it won't be an issue.

CHAPTER 7

CONCLUSION

7.1. CONCLUSION

In conclusion, InfiniteVoix, AI desktop voice assistant represent a significant step forward in human-computer interaction. It embodies the remarkable advancements in AI and machine learning, providing a glimpse into a future where technology is an even more seamless part of our daily lives. Whether for personal convenience, aiding those with disabilities, or boosting business productivity, these assistants are a testament to human ingenuity and the endless possibilities that AI brings. Through this voice assistant, we have automated various services using a single line command. It eases most of the tasks of the user like, opening camera, retrieving weather forecast details, vocabulary help and searching the web.

REFERENCES:-

- [1] Desktop Assistant using Python, Journal of Emerging Technologies and Innovative Research, 6(4), 521-526.
- [2] Haseeb. A, & Rizvi, S. A. R. A Review of Voice Assistant and Its Impact on the Healthcare Industry, International Research Journal of Engineering and Technology, 9(1), 299-303, (2022).
- [3] Kumar. A, Kumar. A, & Kumar. P, Development of Voice Assistant using Python. International Journal of Research in Engineering and Science, 10(2), 43-50, (2021)
- [4] Mukherjee, A. Survey on Virtual Assistant (Google Assistant, Siri, Cortana, Alexa). 4th International Symposium SIRS 2018, 65-74, (2019)
- [5] Muneer. S, & Khan. F. A, Artificial Intelligence and Its Applications in Healthcare, International Research Journal of Modernization in Engineering, Technology and Science, 1(4), 23-27, (2022).
- [6] Nallamothu. M & Mukkamala, R, A Study on Speech Recognition (2019)
- [7] Oracle, Advantages and Disadvantages of Voice Assistants for Marketers. Oracle Marketing Cloud Blog, (2018).
- [8] Palleti, V, SpeechRecognition. PyPI, (2021)
- [9] Sweigart, A, PyAutoGUI. PyPI, (2021)
- [10] Tripathy. S, Build Your Own Desktop Voice Assistant in Python, Analytics Vidhya (2020)
- [11] Upadhyay. A, PyDictionary. PyPI, (2021)
- [12] Yadav. M, pyttsx3. PyPI, (2021)

APPENDIX

A. SOURCE CODE

1.GUI code

```
from tkinter import *
import time
import pyttsx3
import tkinter.messagebox
import threading
import speech_rec as sr
import open_ai as ai
import main_logic as ml

saved_username = ["InfiniteVoix"]
ans=["InfiniteVoix"]
window_size = "400x400"

class ChatInterface(Frame):

    def __init__(self, master=None):
        Frame.__init__(self, master)
        self.master = master

        # sets default bg for top level windows
        self.tl_bg = "#EEEEEE"
        self.tl_bg2 = "#EEEEEE"
        self.tl_fg = "#000000"
        self.font = "Verdana 10"
```

```

menu = Menu(self.master)
self.master.config(menu=menu, bd=5)
# Menu bar

# File
file = Menu(menu, tearoff=0)
menu.add_cascade(label="File", menu=file)
# file.add_command(label="Save Chat Log", command=self.save_chat)
file.add_command(label="Clear Chat", command=self.clear_chat)
# file.add_separator()
file.add_command(label="Exit", command=self.chatexit)

# Options
options = Menu(menu, tearoff=0)
menu.add_cascade(label="Options", menu=options)

# username

# font
font = Menu(options, tearoff=0)
options.add_cascade(label="Font", menu=font)
font.add_command(label="Default", command=self.font_change_default)
font.add_command(label="Times", command=self.font_change_times)
font.add_command(label="System", command=self.font_change_system)
font.add_command(label="Helvetica", command=self.font_change_helvetica)
font.add_command(label="Fixedsys", command=self.font_change_fixedsys)

# color theme
color_theme = Menu(options, tearoff=0)
options.add_cascade(label="Color Theme", menu=color_theme)
color_theme.add_command(label="Default",
command=self.color_theme_default)
color_theme.add_command(label="Grey", command=self.color_theme_grey)

```

```
        color_theme.add_command(label="Blue",
command=self.color_theme_dark_blue)
        color_theme.add_command(label="Torque",
command=self.color_theme_turquoise)
        color_theme.add_command(label="Hacker",
command=self.color_theme_hacker)
```

```
self.text_frame = Frame(self.master, bd=6)
self.text_frame.pack(expand=True, fill=BOTH)
```

```
# scrollbar for text box
```

```
self.text_box_scrollbar = Scrollbar(self.text_frame, bd=0)
self.text_box_scrollbar.pack(fill=Y, side=RIGHT)
```

```
# contains messages
```

```
self.text_box = Text(self.text_frame,
yscrollcommand=self.text_box_scrollbar.set, state=DISABLED,
                    bd=1, padx=6, pady=6, spacing3=8, wrap=WORD, bg=None,
font="Verdana 10", relief=GROOVE,
                    width=10, height=1)
self.text_box.pack(expand=True, fill=BOTH)
self.text_box_scrollbar.config(command=self.text_box.yview)
```

```
# frame containing user entry field
```

```
self.entry_frame = Frame(self.master, bd=1)
self.entry_frame.pack(side=LEFT, fill=BOTH, expand=True)
```

```
# entry field
```

```
self.entry_field = Entry(self.entry_frame, bd=1, justify=LEFT)
self.entry_field.pack(fill=X, padx=6, pady=6, ipady=3)
# self.users_message = self.entry_field.get()
```

```

# frame containing send button and emoji button
self.send_button_frame = Frame(self.master, bd=0)
self.send_button_frame.pack(fill=BOTH)

self.mic_button = Button(self.send_button_frame, text="mic", width=5,
relief=GROOVE, bg='white',
                        bd=1, command=lambda: self.mic_message_insert(),
activebackground="#FFFFFF",
                        activeforeground="#000000")
self.mic_button.pack(side=LEFT, ipady=7)
self.master.bind("<Return>", self.mic_message_insert)

# send button
self.send_button = Button(self.send_button_frame, text="Send", width=5,
relief=GROOVE, bg='white',
                        bd=1, command=lambda: self.send_message_insert(None),
activebackground="#FFFFFF",
                        activeforeground="#000000")
self.send_button.pack(side=LEFT, ipady=8)
self.master.bind("<Return>", self.send_message_insert)

#t2 = threading.Thread(target=self.send_message_insert(name='t1'))
#t2.start()

def playResponce(self, response):
    x = pyttsx3.init()
    print(response)
    li = []
    if len(response) > 100:
        if response.find('--') == -1:
            b = response.split('--')
            print(b)

```

```

x.setProperty('rate', 200)
x.setProperty('volume', 100)
x.say(responce)
x.runAndWait()
print("Played Successfully.....")

def last_sent_label(self, date):

    try:
        self.sent_label.destroy()
    except AttributeError:
        pass

    self.sent_label = Label(self.entry_frame, font="Verdana 7", text=date,
bg=self.tl_bg2, fg=self.tl_fg)
    self.sent_label.pack(side=LEFT, fill=X, padx=3)

def clear_chat(self):
    self.text_box.config(state=NORMAL)
    self.last_sent_label(date="No messages sent.")
    self.text_box.delete(1.0, END)
    self.text_box.delete(1.0, END)
    self.text_box.config(state=DISABLED)

def chatexit(self):
    exit()

def msg(self):
    tkinter.messagebox.showinfo("InfiniteVoix v1.0",
                                'InfiniteVoix is a chatbot for answering questions\nIt is based
on NLP using pythons NLTK tool-kit module\nGUI is based on Tkinter')

def about(self):

```

```
tkinter.messagebox.showinfo("InfiniteVoix Developers","Karthikeyan and  
Kevin samson")
```

```
def send_message_insert(self, message):  
    user_input = self.entry_field.get()  
    pr1 = "Human : " + user_input + "\n"  
    self.text_box.configure(state=NORMAL)  
    self.text_box.insert(END, pr1)  
    self.text_box.configure(state=DISABLED)  
    self.text_box.see(END)  
    #t1 = threading.Thread(target=self.playResponce, args=(user_input,))  
    #t1.start()  
    #time.sleep(1)  
    #ob = chat(user_input)  
  
    ob = ml.decideAction(user_input)  
    pr = "BOT: " + ob + "\n"  
    self.text_box.configure(state=NORMAL)  
    self.text_box.insert(END, pr)  
    self.text_box.configure(state=DISABLED)  
    self.text_box.see(END)  
    self.last_sent_label(str(time.strftime("Last message sent: " + '%B %d, %Y' + '  
at ' + '%l:%M %p'))))  
    self.entry_field.delete(0, END)  
    time.sleep(0)  
    t2 = threading.Thread(target=self.playResponce, args=(ob,))  
    t2.start()  
    #return ob  
  
def font_change_default(self):  
    self.text_box.config(font="Verdana 10")  
    self.entry_field.config(font="Verdana 10")  
    self.font = "Verdana 10"
```

```

def font_change_times(self):
    self.text_box.config(font="Times")
    self.entry_field.config(font="Times")
    self.font = "Times"

def font_change_system(self):
    self.text_box.config(font="System")
    self.entry_field.config(font="System")
    self.font = "System"

def font_change_helvetica(self):
    self.text_box.config(font="helvetica 10")
    self.entry_field.config(font="helvetica 10")
    self.font = "helvetica 10"

def font_change_fixedsys(self):
    self.text_box.config(font="fixedsys")
    self.entry_field.config(font="fixedsys")
    self.font = "fixedsys"

def color_theme_default(self):
    self.master.config(bg="#EEEEEE")
    self.text_frame.config(bg="#EEEEEE")
    self.entry_frame.config(bg="#EEEEEE")
    self.text_box.config(bg="#FFFFFF", fg="#000000")
    self.entry_field.config(bg="#FFFFFF", fg="#000000",
insertbackground="#000000")
    self.send_button_frame.config(bg="#EEEEEE")
    self.send_button.config(bg="#FFFFFF", fg="#000000",
activebackground="#FFFFFF", activeforeground="#000000")
    # self.emoji_button.config(bg="#FFFFFF", fg="#000000",
activebackground="#FFFFFF", activeforeground="#000000")
    self.sent_label.config(bg="#EEEEEE", fg="#000000")

```

```

self.tl_bg = "#FFFFFF"
self.tl_bg2 = "#EEEEEE"
self.tl_fg = "#000000"

# Dark
def color_theme_dark(self):
    self.master.config(bg="#2a2b2d")
    self.text_frame.config(bg="#2a2b2d")
    self.text_box.config(bg="#212121", fg="#FFFFFF")
    self.entry_frame.config(bg="#2a2b2d")
    self.entry_field.config(bg="#212121", fg="#FFFFFF",
insertbackground="#FFFFFF")
    self.send_button_frame.config(bg="#2a2b2d")
    self.send_button.config(bg="#212121", fg="#FFFFFF",
activebackground="#212121", activeforeground="#FFFFFF")
    self.emoji_button.config(bg="#212121", fg="#FFFFFF",
activebackground="#212121", activeforeground="#FFFFFF")
    self.sent_label.config(bg="#2a2b2d", fg="#FFFFFF")

    self.tl_bg = "#212121"
    self.tl_bg2 = "#2a2b2d"
    self.tl_fg = "#FFFFFF"

# Grey
def color_theme_grey(self):
    self.master.config(bg="#444444")
    self.text_frame.config(bg="#444444")
    self.text_box.config(bg="#4f4f4f", fg="#ffffff")
    self.entry_frame.config(bg="#444444")
    self.entry_field.config(bg="#4f4f4f", fg="#ffffff", insertbackground="#ffffff")
    self.send_button_frame.config(bg="#444444")
    self.send_button.config(bg="#4f4f4f", fg="#ffffff", activebackground="#4f4f4f",
activeforeground="#ffffff")

```

```
self.emoji_button.config(bg="#4f4f4f", fg="ffffff", activebackground="#4f4f4f",
activeforeground="ffffff")
```

```
self.sent_label.config(bg="#444444", fg="ffffff")
```

```
self.tl_bg = "#4f4f4f"
```

```
self.tl_bg2 = "#444444"
```

```
self.tl_fg = "ffffff"
```

```
def color_theme_turquoise(self):
    self.master.config(bg="#003333")
    self.text_frame.config(bg="#003333")
    self.text_box.config(bg="#669999", fg="FFFFFF")
    self.entry_frame.config(bg="#003333")
    self.entry_field.config(bg="#669999", fg="FFFFFF",
insertbackground="FFFFFF")
    self.send_button_frame.config(bg="#003333")
    self.send_button.config(bg="#669999", fg="FFFFFF",
activebackground="#669999", activeforeground="FFFFFF")
    self.emoji_button.config(bg="#669999", fg="FFFFFF",
activebackground="#669999", activeforeground="FFFFFF")
    self.sent_label.config(bg="#003333", fg="FFFFFF")
```

```
self.tl_bg = "#669999"
```

```
self.tl_bg2 = "#003333"
```

```
self.tl_fg = "FFFFFF"
```

```
# Blue
```

```
def color_theme_dark_blue(self):
    self.master.config(bg="#263b54")
    self.text_frame.config(bg="#263b54")
    self.text_box.config(bg="#1c2e44", fg="FFFFFF")
    self.entry_frame.config(bg="#263b54")
```

```

        self.entry_field.config(bg="#1c2e44", fg="#FFFFFF",
insertbackground="#FFFFFF")
        self.send_button_frame.config(bg="#263b54")
        self.send_button.config(bg="#1c2e44", fg="#FFFFFF",
activebackground="#1c2e44", activeforeground="#FFFFFF")
        self.emoji_button.config(bg="#1c2e44", fg="#FFFFFF",
activebackground="#1c2e44", activeforeground="#FFFFFF")
        self.sent_label.config(bg="#263b54", fg="#FFFFFF")

```

```

self.tl_bg = "#1c2e44"
self.tl_bg2 = "#263b54"
self.tl_fg = "#FFFFFF"

```

Torque

```

def color_theme_turquoise(self):
    self.master.config(bg="#003333")
    self.text_frame.config(bg="#003333")
    self.text_box.config(bg="#669999", fg="#FFFFFF")
    self.entry_frame.config(bg="#003333")
    self.entry_field.config(bg="#669999", fg="#FFFFFF",
insertbackground="#FFFFFF")
    self.send_button_frame.config(bg="#003333")
    self.send_button.config(bg="#669999", fg="#FFFFFF",
activebackground="#669999", activeforeground="#FFFFFF")
    self.emoji_button.config(bg="#669999", fg="#FFFFFF",
activebackground="#669999", activeforeground="#FFFFFF")
    self.sent_label.config(bg="#003333", fg="#FFFFFF")

```

```

self.tl_bg = "#669999"
self.tl_bg2 = "#003333"
self.tl_fg = "#FFFFFF"

```

Hacker

```

def color_theme_hacker(self):

```

```

self.master.config(bg="#0F0F0F")
self.text_frame.config(bg="#0F0F0F")
self.entry_frame.config(bg="#0F0F0F")
self.text_box.config(bg="#0F0F0F", fg="#33FF33")
self.entry_field.config(bg="#0F0F0F", fg="#33FF33",
insertbackground="#33FF33")
self.send_button_frame.config(bg="#0F0F0F")
self.send_button.config(bg="#0F0F0F", fg="FFFFFF",
activebackground="#0F0F0F", activeforeground="FFFFFF")
self.emoji_button.config(bg="#0F0F0F", fg="FFFFFF",
activebackground="#0F0F0F", activeforeground="FFFFFF")
self.sent_label.config(bg="#0F0F0F", fg="#33FF33")

self.tl_bg = "#0F0F0F"
self.tl_bg2 = "#0F0F0F"
self.tl_fg = "#33FF33"

# Default font and color theme
def default_format(self):
    self.font_change_default()
    self.color_theme_default()

def speech_to_text(self):
    mic_input = sr.takeCommand()
    return mic_input

def nlp(self,prompt):
    self.response = ai.processCommand(prompt)

def mic_message_insert(self):
    user_input = self.speech_to_text()
    pr1 = "Human : " + user_input + "\n"
    self.text_box.configure(state=NORMAL)

```

```

self.text_box.insert(END, pr1)
self.text_box.configure(state=DISABLED)
self.text_box.see(END)
#t1 = threading.Thread(target=self.playResponce, args=(user_input,))
#t1.start()
#time.sleep(1)
#ob = chat(user_input)
ob = ml.decideAction(user_input)
pr = "Bot: " + ob + "\n"
self.text_box.configure(state=NORMAL)
self.text_box.insert(END, pr)
self.text_box.configure(state=DISABLED)
self.text_box.see(END)
self.last_sent_label(str(time.strftime("Last message sent: " + '%B %d, %Y' + '
at ' + '%l:%M %p'))))
self.entry_field.delete(0, END)
time.sleep(0)
t2 = threading.Thread(target=self.playResponce, args=(ob,))
t2.start()

```

```

root = Tk()
a = ChatInterface(root)
root.geometry(window_size)
root.title("InfiniteVoix")
root.mainloop()

```

2. Main logic code

```
import Text_speech_converter as ts
```

```

import cv
import mail_skill as m
import open_ai as ai
import weather_pred as weather

camera_words = ['open camera','take a picture','take a photo']
timed_words = ['in', 'after']

open_pattern = r"open (.+)"
mail_pattern = r"to (.+)"

def decideAction(statement):
    for i in camera_words:
        if i in statement:
            for j in timed_words:
                if j in statement:
                    number_str = ""

                    for char in statement:
                        if char.isdigit():
                            number_str += char

                    if number_str:
                        number = int(number_str)

                        ts.speak(f"I have opened camera for you and will take a picture in
{number} seconds")
                        cv.takePicture(number)
                        return "I have opened camera for you and will take a picture in
"+str(number)+"seconds"

                    ts.speak("I have opened camera for you, press spacebar to click a picture
and q to exit")

```

```
        cv.takePicture(0)
        return "I have opened camera for you, press spacebar to click a picture and
q to exit"
```

if 'open youtube' in statement:

```
    webbrowser.open_new_tab("https://www.youtube.com")
    ts.speak("youtube is open now")
    return "youtube is open now"
```

elif 'open google' in statement:

```
    webbrowser.open_new_tab("https://www.google.com")
    ts.speak("Google chrome is open now")
    time.sleep(5)
```

elif 'open gmail' in statement:

```
    webbrowser.open_new_tab("gmail.com")
    ts.speak("Google Mail open now")
    time.sleep(5)
```

elif 'send mail' in statement:

```
    # if re.match(mail_pattern, statement):
    #     entity = re.match(mail_pattern, statement).group(1)
    m.sendEmail("", "", "")
```

elif 'screenshot' in statement:

```
    cv.takeSS()
    return "screenshot has been taken"
```

elif 'weather' in statement:

```
    # weather_pattern = r"in (.+)"
    # city = re.match(weather_pattern, statement).group(1)
    city="chennai"
    lst = statement.split()
    for i in range(len(lst)):
```

```
if lst[i] == 'in':  
    city = lst[i+1]
```

```
res = weather.tellWeather(city)  
ans = ai.processCommand(res)  
return ans
```

```
else:  
    res = ai.processCommand(statement)  
    return res
```

B. SCREENSHOTS

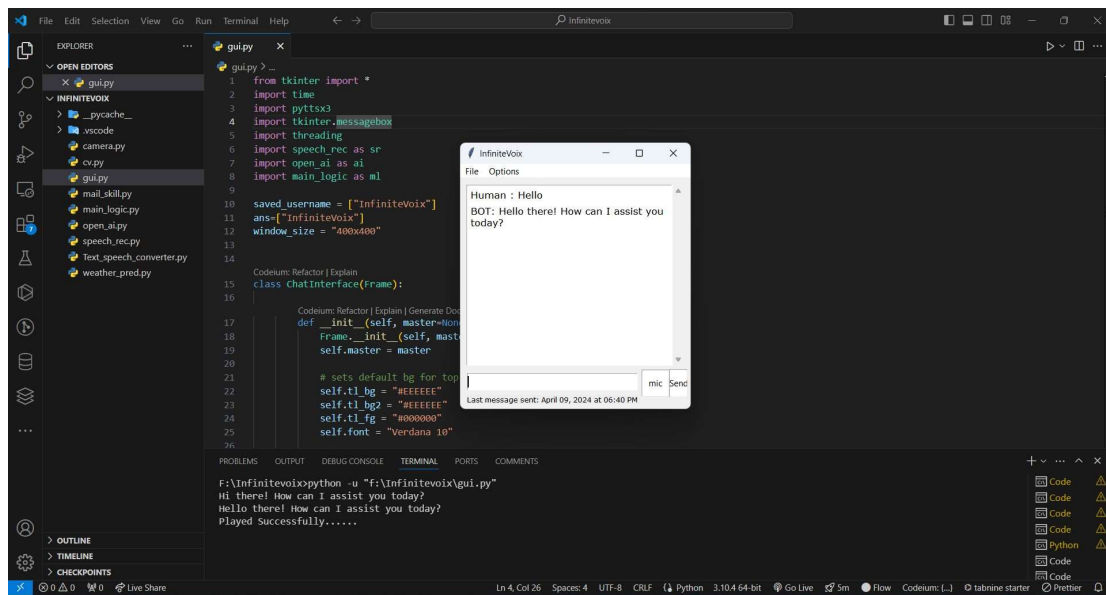


Fig B.1: GUI of InfiniteVoix

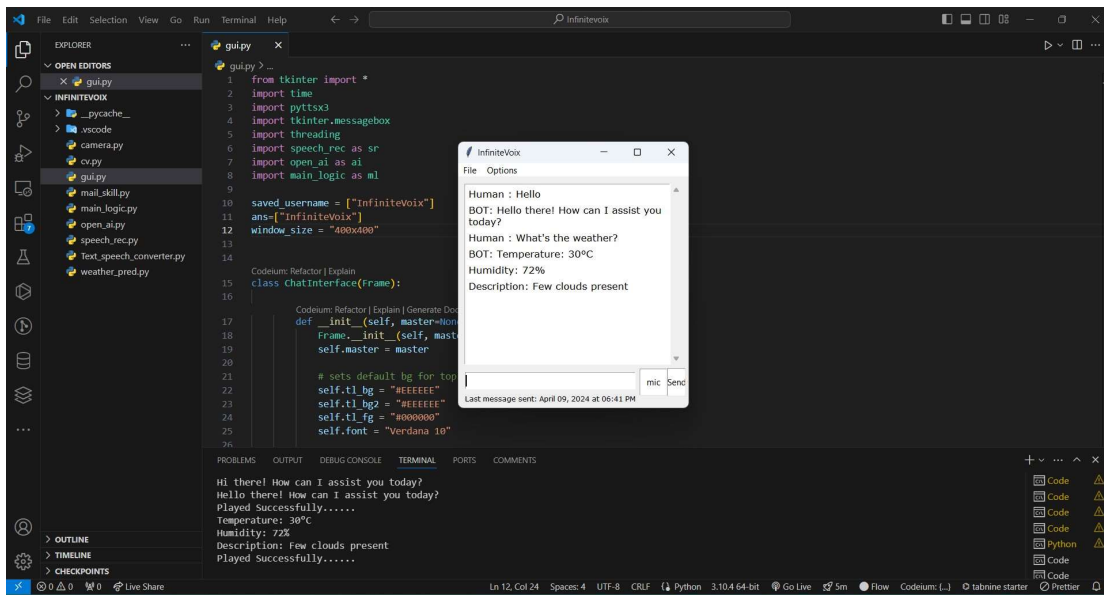


Fig B.2: Weather command

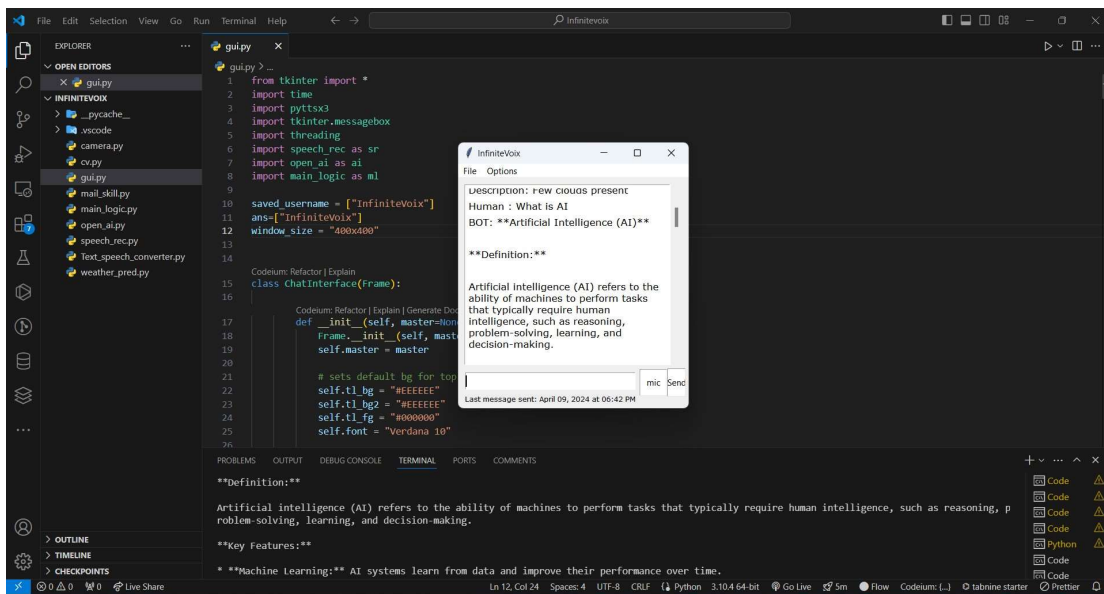


Fig B.3: Asking a Question

C. RESEARCH PAPER

InfiniteVoix - An Ai Desktop Voice Assistant for Differently abled people

Karthikeyan S¹

*"Dept.of Computer Science and Engineering"
"Sathyabama Institute of Science and Technology"
"Chennai, Tamilnadu, India".
karthi2003ind@gmail.com*

Kevin samson R²

*"Dept.of Computer Science and Engineering"
"Sathyabama Institute of Science and Technology"
"Chennai, Tamilnadu, India."
kevin007samson@gmail.com*

Ms.V.Gowri manohari³

*"Dept.of Computer Science and Engineering"
"Sathyabama Institute of Science and Technology"
"Chennai, Tamil Nadu,India."
Assistant professor
gowrimanohari.cse@sathyabama.ac.in*

Ms.P.Nandhini⁴

*"Dept.of Computer Science and Engineering"
"Sathyabama Institute of Science and Technology"
"Chennai, Tamil Nadu,India."
Assistant professor
nandhini.p.cse@sathyabama.ac.in*

Ms.J.Joan Niveda⁵

*"Dept.of Computer Science and Engineering"
"Sathyabama Institute of Science and Technology"
"Chennai, Tamil Nadu,India."
Assistant professor
joanniveda.cse@sathyabama.ac.in*

Abstract: InfiniteVoix, an Ai Desktop Voice assistant designed for differently abled people, acts as a proficient software agent capable of executing a wide range of tasks or services for individual users. While traditionally associated with smartphones and laptops, voice assistants have now expanded their reach to include home automation systems and smart speakers. InfiniteVoix aims to bring the convenience of voice assistants to desktop platforms. Leveraging Natural Language Processing (NLP), a branch of AI, the assistant interprets spoken commands and carries out the corresponding tasks. At the heart of our project lies the disabled people's ability to guide the voice assistant using the device's microphone. InfiniteVoix can be incredibly beneficial for individuals with disabilities, offering hands-free interaction that enhances independence. They enable control over devices and access to information without physical touch, which is particularly helpful for those with mobility or visual impairments. Additionally, InfiniteVoix can assist in performing everyday tasks, making technology more accessible and inclusive. The spoken command is transcribed into text, initiating further processing of the request. The system then executes the command and provides a text response, accompanied by the necessary actions performed by the voice assistant.

Keywords: Artificial Intelligence, Machine Learning, Speech-to-text, NLP, Infinite Voix

I.INTRODUCTION

The landscape of computing has undergone a significant transformation, giving rise to computers as indispensable and cost-effective tools. The inception of the Personal Virtual Assistant aims to create an economical, dependable, and user-friendly aide. A Virtual Assistant (VA) goes beyond the realm of traditional computing, embodying a simulated presence in both real and imaginary contexts. It operates in real-time, interpreting user inputs and dynamically adjusting the virtual environment. This interactive nature elevates user engagement, leveraging all human sensory channels for a heightened interactive experience.

Contemporary virtual assistant platforms primarily offer visual interfaces on computer screens, occasionally integrating additional sensory elements like sound through speakers or headphones. The potential applications of VA technology span diverse domains, including training simulations, healthcare, rehabilitation, education, engineering, scientific visualization, and entertainment. This application, akin to Siri or Google Assistant, concentrates on interactions within the computer environment. The voice assistant within this project harnesses advanced technologies, including speech recognition, natural language processing, and speech synthesis, all deeply

rooted in fundamental Artificial Intelligence principles.

II.LITERATURE SURVEY

In the paper titled "Study of Voice Controlled Personal Assistant Device," the author explores the application of iPhone's renowned "SIRI" and Google's "Google Voice Search" for voice communication and command response on mobile devices. While these applications generally rely on internet connectivity, the proposed system, named Personal Assistant with Voice Recognition Intelligence, distinguishes itself by its capability to function both with and without an internet connection. This system is designed to accept user inputs in the form of either voice or text, processing them and providing output in various formats, such as actionable commands or dictated search results. In a pertinent work titled "AI Based Voice Assistant Using Python," the author presents a virtual assistant for home automation that is both affordable and very effective, built on the Raspberry Pi platform. This voice assistant lets users command the device with their voices by using the principles of AI, Natural Language Processing, Internet of Things, and Speech Recognition. Its capabilities extend to retrieving date and time information, providing weather updates, playing music, fetching internet search results, and controlling home appliances.

Key findings from the literature survey underscore the system's significance in operating partially without internet connectivity, recognizing and responding to user commands effectively. Furthermore, the successful implementation of Natural Language Processing emerges as a critical factor, facilitating an understanding of user intent and enabling efficient task assistance, ultimately establishing a robust personal AI assistant.

III.RELATED WORKS

Open Problems in Existing System:

Lack of Offline Functionality: The current system heavily relies on internet connectivity. While internet access is generally available, the system should be designed to handle user interactions even in the absence of an internet connection.

Absence of Artificial Intelligence (AI): The existing system lacks integration with artificial intelligence, a crucial component for understanding user needs and providing intelligent results. Incorporating AI into the voice assistant enhances its capabilities, enabling it to perform tasks beyond the scope of a conventional voice assistant.

Risk Analysis of the Project:

Technical Feasibility:

Hardware Requirements: Evaluate the capabilities of typical desktop computers to ensure they can handle voice recognition and Natural Language Processing (NLP) without significant performance degradation. Consider system compatibility and resource constraints.

Software Requirements: Assess the availability of necessary software tools and libraries for NLP, speech recognition, and voice synthesis on desktop platforms. Verify compatibility with existing libraries and APIs.

NLP Accuracy: Verify the accuracy of the planned NLP and voice recognition technologies in understanding a diverse range of commands and accents. Test for response accuracy and adaptability to user commands.

Scalability: Investigate the system's scalability to accommodate a growing user base and potential updates or expansions.

Operational Feasibility:

User Acceptance: Conduct surveys, focus groups, or user interviews to assess user interest and acceptance of a desktop voice assistant. Identify potential use cases and gather feedback for feature development.

User Training: Evaluate the need for user training to effectively use the voice assistant. Consider the system's ease of use and the intuitiveness of its user interface.

Maintenance and Support: Define operational procedures for system maintenance and user support. Evaluate the cost and resources required for ongoing technical support.

Proposed System:

In this conceptualization of an AI-based voice assistant, the implementation involves an efficient utilization of a Speech Recognition library with a multitude of built-in functions. These functions facilitate the assistant in comprehending user commands, and the response is then delivered back to the user in the form of voice using Text-to-Speech functions. The process begins with the assistant capturing the voice command from the user, and the underlying algorithms are responsible for converting this voice input into text.

IV.METHODOLOGY

Proposed Architecture:

The proposed system design encompasses the following key components:

Input Reception: The system takes input in the form of speech patterns through a microphone, capturing the user's vocal commands.

Audio Data Recognition and Conversion: The captured audio data undergoes a recognition process, with algorithms converting the voice input into textual form for further analysis.

Command Comparison: The converted text is then compared with predefined commands within the system, determining the appropriate action or response based on the recognized input.

Output Generation: The voice assistant's interactive loop concludes when the system creates the required output and voices it to the

user.

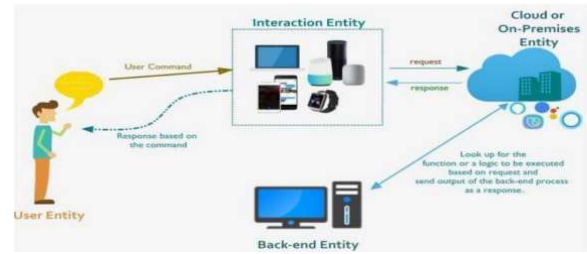


Fig.1 Proposed Architecture

The initial stage involves capturing data in the form of speech patterns through a microphone. In the subsequent phase, the gathered data undergoes processing and transformation into textual data using Natural Language Processing (NLP). Following this, a Python script is employed to manipulate the resulting text for the final output process. In the concluding phase, the generated output is presented either as text or converted from text to speech using Text-to-Speech (TTS).

The system is designed to offer the following features:

Continuous listening in an inactive state, awakening into action when called with a specific predetermined functionality.

Process Model:

In the context of this project, Python is the chosen programming language, and the implementation utilizes the Google Text-to-Speech API. Voice recognition is achieved through the Speech Recognition module, enabling the system to generate queries based on user input. Interactions with the internet involve various modules such as web browsers, YouTube, Wikipedia, etc., while operating system-related queries are managed by the os module.

Designed with educational purposes in mind, users can seamlessly search for information on specific topics through platforms like Wikipedia, Google, or within text documents. The initiative harmonizes fundamental tenets of Artificial Intelligence (AI) and Natural Language Processing (NLP) to transform textual information into spoken language. The primary goal is to create an AI voice assistant tailored

to the needs of visually impaired people. This aims to empower them to interact with evolving technologies, oversee device management, and enhance learning opportunities.

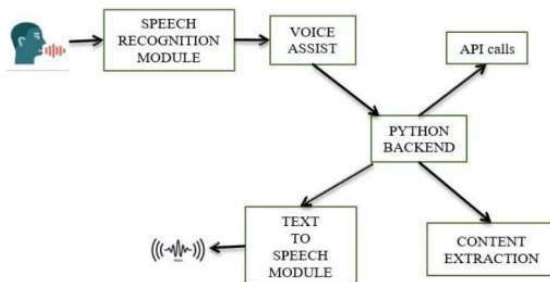


Fig.2 Architecture diagram

Speech recognition is integral to the smooth functioning of virtual assistants, serving as the linchpin in their operations. Through the use of sophisticated voice recognition technology, the assistant is able to convert the user's spoken commands into text. The assistant's workflow unfolds methodically through distinct stages:

Natural Language Understanding (NLU): A comprehensive analysis of the user's spoken request takes place, unraveling its intent. This involves discerning keywords and context to precisely determine the desired action.

Intent Mapping: The assistant deftly translates the user's request into concrete actions by using NLU. This step connects the user's intent with the relevant functionalities of the assistant. For instance, a weather-related query might be seamlessly directed to a weather app.

Action Execution: The helper then efficiently carries out the designated activity or actions after intent mapping. With deftness, it finds what the user needs and gives it to them, verbally or in writing.

Response Generation: The last step is for the assistant to skillfully come up with a reply for the user. You may hear or see this answer, which includes the requested data, confirmation of successful actions, or an error message if the request was unsuccessful. In the implementation and testing

plan of the proposed model/system, the following steps are delineated:

Input Reception: Capturing speech patterns through a microphone.

Audio Data Recognition and Conversion: Transforming the captured audio data into text.

Command Comparison: Comparing the converted text with predefined commands.

Output Generation: Creating the desired output based on the comparison is accomplished through specialized modules, including the Speech Recognition module, Natural Language Processing module, and Text-to-Speech conversion module.

Speech Recognition Module:

The speech recognition module, often referred to as speech-to-text, is a crucial component that identifies spoken words and converts them into readable text. This intricate process involves the analysis of audio, segmentation, digitization into a computer-readable format, and the application of algorithms to find the most suitable text representation.

Natural Language Processing Module:

The Natural Language Processing (NLP) module, a subset of artificial intelligence, empowers computers to comprehend, interpret, and manipulate human language. By integrating computational linguistics, machine learning, and deep learning models, this module processes both text and voice data, ensuring a comprehensive understanding of meaning, intent, and sentiment.

Text-to-Speech Engine:

The Text-to-Speech (TTS) engine is a transformative technology that converts written text into spoken audio. This feature facilitates seamless communication with users, allowing the system to read aloud content from PDFs, websites, and books using natural AI voices.

In synergy, these modules work cohesively to empower the system, providing a comprehensive and user-friendly voice assistant experience. In

conclusion, the proposed virtual assistant seamlessly integrates cutting-edge technologies, including speech recognition, natural language processing, and text-to-speech, ensuring a robust and user-friendly interaction for enhanced user experience.

V.CONCLUSION

Through meticulous stages, including intent mapping and action execution, the assistant adeptly processes user requests, generating responses in either text or speech. The implementation plan outlines key steps, from input reception to output generation, facilitated by specialized. This cohesive integration empowers the system to offer comprehensive functionality, making it a versatile and inclusive voice assistant catering to diverse user needs.

REFERENCES

- [1] Palleti, V. (2021). SpeechRecognition. PyPI.
- [2] Upadhyay, A. (2021). PyDictionary. PyPI.
- [3] Sweigart, A. (2021). PyAutoGUI. PyPI.
- [4] Yadav, M. (2021). pyttsx3. PyPI.
- [5] Nallamothe, M., & Mukkamala, R. (2019). A Study on Speech Recognition and
- [6] Desktop Assistant using Python. Journal of Emerging Technologies and Innovative Research, 6(4), 521-526.
- [7] Mukherjee, A. (2019). Survey on Virtual Assistant (Google Assistant, Siri, Cortana, Alexa). 4th International Symposium SIRS 2018, Revised Selected Papers, 65-74.
- [8] Haseeb, A., & Rizvi, S. A. R. (2022). A Review of Voice Assistant and Its Impact on the Healthcare Industry. International Research Journal of Engineering and Technology, 9(1), 299-303.
- [9] Oracle. (2018). Advantages and Disadvantages of Voice Assistants for Marketers. Oracle Marketing Cloud Blog.
- [10] Muneer, S., & Khan, F. A. (2022). Artificial Intelligence and Its Applications in Healthcare.
- [11] Tripathy, S. (2020). Build Your Own Desktop Voice Assistant in Python. Analytics Vidhya.
- [12] Kumar, A., Kumar, A., & Kumar, P. (2021). Development of Voice Assistant using Python. International Journal of Research in Engineering and Science, 10(2), 43-50.

D. CERTIFICATES

