VOICE CRAFT DESKTOP ASSISTANT

A Project Partial Report submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

Gannamaneni Vignesh	122010308042
Usman Shariff	122010308044
Meghana Bandaru	122010308010
Viswanadhula Jaya Sri	122010333032

Under the esteemed guidance of

Dr. B. Srinivasa Rao

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

GITAM

(Deemed to be University)

VISAKHAPATNAM

MARCH 2024

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GITAM SCHOOL OF TECHNOLOGY

GITAM

(Deemed to be University)

DECLARATION

We, hereby declare that the project report entitled "**VOICE CRAFT DESKTOP ASSISTANT**" is an original work done in the Department of Computer Science and Engineering, GITAM School of Technology, GITAM (Deemed to be University) submitted in partial fulfillment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering. The work has not been submitted to any other college or University for the award of any degree or diploma.

Date: March 13, 2024

Registration No(s).	Name(s)	Signatures(s)
122010308042	Gannamaneni Vignesh	
122010308044	Usman Shariff	
122010308010	Meghana Bandaru	
122010333032	Viswanadhula Jaya Sri	

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GITAM SCHOOL OF TECHNOLOGY

GITAM

(Deemed to be University)

CERTIFICATE

This is to certify that the project report entitled "**VOICE CRAFT DESKTOP ASSISTANT**" is a bonafide record of work carried out by **Gannamaneni Vignesh (122010308042), Usman Shariff (122010308044), Meghana Bandaru (122010308010), Viswanadhula Jaya Sri (122010333032)** students submitted in partial fulfillment of requirement for the award of degree of Bachelor of Technology in Computer Science and Engineering.

Project Guide

Dr. B. Srinivasa Rao

Deputy HOD

TABLE OF CONTENTS

S. No	Topic	Page No
1	Abstract	5
2	Objectives	6
3	Introduction	7-9
4	Literature Review	10
5	Problem Statement	11
6	Requirement Analysis	12-15
7	System Analysis	16-17
8	System Methodology	18-20
9	System Design	21-31
10	Implementation	32-36
11	Analysis and Results	37-49
12	Conclusion	50-52
13	References	53
14	Plagiarism Report	54-56

1. Abstract

Natural Language Processing (NLP) systems represent a critical aspect of modern human-computer interaction, especially within voice-controlled interfaces. This paper investigates key components of NLP systems, including intent classification, entity recognition, and their integration with ChatGPT to handle uncertain queries. Intent classification forms the backbone of user interaction, allowing systems to discern user intentions accurately. Leveraging labelled training data and machine learning techniques, intent classification enables systems to provide effective responses to user commands. Similarly, entity recognition enhances user interactions by identifying and categorizing specific entities mentioned in user input.

The integration of ChatGPT introduces a new dimension to NLP systems, enabling them to respond to uncertain or ambiguous queries with intelligence and relevance. By leveraging natural language processing and advanced AI capabilities, ChatGPT enhances the overall user experience by providing informative responses to a wide range of user queries.

In practical terms, the combined capabilities of intent classification, entity recognition, and ChatGPT integration empower voice-controlled systems to interpret and respond to user commands and queries more effectively. This not only improves the efficiency of human-computer interaction but also offers users a more natural and intuitive experience.

Through a comprehensive understanding of these components and their integration, this project aims to explore their significance in advancing NLP systems and revolutionizing human-computer interaction. By harnessing the power of machine learning, NLP, and AI, voice-controlled interfaces can provide users with an intelligent and efficient means of interacting with technology.

Keywords:

Natural Language Processing, Intent Classification, Entity Recognition, ChatGPT Integration, Voice-Controlled Interfaces, Machine Learning, User Interaction, Speech Recognition, and Text Processing.

2. Objectives

The main aim of this project is to introduce an innovative method for desktop interaction by developing a sophisticated Desktop Voice Assistant integrated with Chat GPT. It enables users to interact with their computers hands-free, and the initiative promises smoother navigation and task completion. In addition to this, the project is geared towards enhancing productivity by streamlining workflow and saving time. Furthermore, it aims to ensure accessibility for all users, including those with physical disabilities, thereby fostering inclusivity in computing. Leveraging the capabilities of natural language processing and advanced AI, the Voice Assistant seeks to simplify desktop interactions, making them more intuitive and user-friendly. Through intelligent responses to user inquiries and seamless integration with desktop applications, the project aims to elevate the overall user experience, promoting satisfaction and engagement. Ultimately, desktop interaction, the project strives to establish a new standard for intuitive and user-centric computing experiences, driving forward the evolution of desktop computing.

- Develop a Desktop Voice Assistant that seamlessly integrates natural language processing (NLP) capabilities to enable efficient voice interaction with desktop environments.
- Implement intent classification and entity recognition algorithms to accurately understand user commands and queries.
- Integrate ChatGPT to handle uncertain or ambiguous queries effectively, providing intelligent responses based on context and user intent.
- Enhance productivity and user experience by enabling hands-free navigation and task execution through voice commands.
- Ensure compatibility with various desktop operating systems and applications to cater to a wide range of users and use cases.

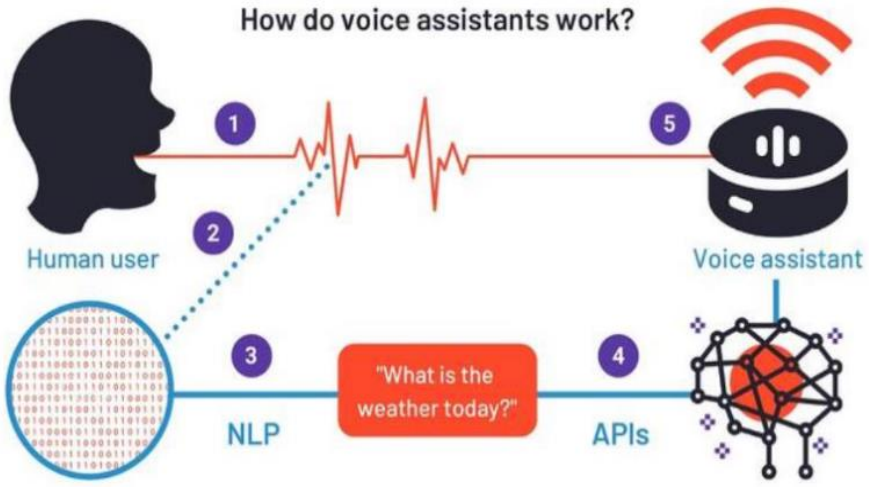
3. Introduction

A Voice assistant or intelligent personal assistant is a software agent designed to carry out tasks or provide services for users based on verbal commands, interpreting human speech, and responding through synthesized voice. These assistants have become ubiquitous in today's digital landscape, offering users a convenient and hands-free means of interacting with technology.

In our AI-based desktop assistant, users can utilize voice commands to perform a variety of tasks, ranging from basic inquiries to executing specific actions. For instance, users may request the assistant to search for information on Wikipedia, inquire about the current time, or launch applications such as Internet Explorer, YouTube, or Visual Studio Code. Additionally, users can instruct the assistant to undertake more intricate tasks, such as composing and sending emails to specific individuals.

3.1 How do voice assistants function?

Voice assistants operate through a series of intricate processes enabling them to comprehend and respond to user commands effectively. The process begins with speech recognition, converting spoken commands into text. Natural language processing algorithms then analyze this text to discern its meaning and intent. Backend processing executes the necessary actions based on this understanding, such as retrieving information from databases or controlling connected devices. Finally, the assistant formulates a response and communicates it back to the user, continuously learning and enhancing its capabilities over time.



3.2 Applications of Speech

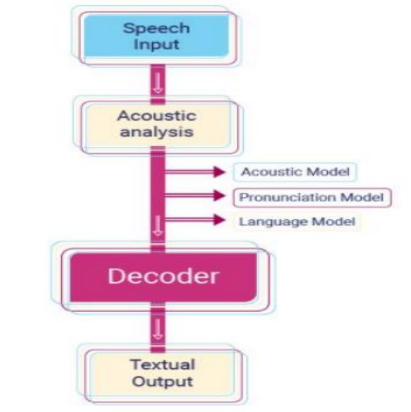
The Speech Application Programming Interface, commonly known as SAPI, is a software development tool created by Microsoft. Its primary purpose is to streamline the integration of speech recognition and synthesis capabilities into Windows applications. SAPI has seen several iterations over the years, often distributed either as part of the Speech Software Development Kit (SDK) or bundled with the Windows operating system. It has been widely utilized by various applications, with notable examples including Microsoft Office, Microsoft Agent, and Microsoft Speech Server.

3.2.1 Incorporating SAPI5 into Our Project:

Our project employs SAPI5, the most recent iteration of the API, to integrate speech recognition and synthesis capabilities into Windows applications. SAPI5, like its predecessors, offers a standardized set of interfaces accessible across various programming languages. This allows developers to seamlessly integrate speech-related features into their applications.

SAPI is crafted with versatility in mind, enabling third-party companies to create their own Speech Recognition and Text-To-Speech engines or modify existing ones to interface with the API. This adaptability fosters the development of a diverse array of speech-related applications customized to meet specific linguistic requirements and user needs. The Speech platform comprises application runtimes providing speech functionality, an Application Programming Interface (API) for managing the runtime, and Runtime Languages enabling speech recognition and synthesis in specific languages.

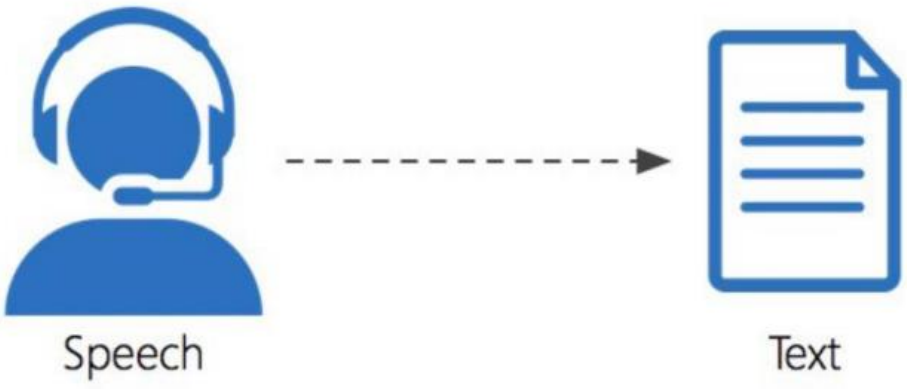
SAPI finds applications in diverse fields, from enhancing accessibility for users with disabilities to facilitating hands-free operation in various industries. Its robust speech recognition capabilities enable efficient communication and interaction with Windows applications, boosting productivity and user experience.



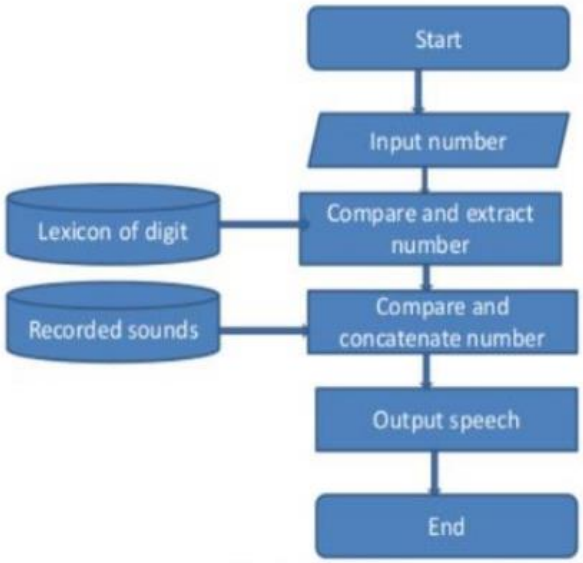
3.2.2 Speech-To-Text

Speech-to-text, also known as speech recognition, is the process of converting spoken words into text. It enables users to issue commands verbally, which are then transcribed into text form. This text can then be further processed to extract relevant information. Speech-to-text models need to be accent-agnostic, meaning they can accurately transcribe speech regardless of the user's accent or pronunciation.

This technology is invaluable for individuals who need to generate written content efficiently, as it eliminates the need for manual typing. Additionally, speech-to-text software is particularly beneficial for individuals with disabilities that may hinder their ability to use a keyboard effectively. By converting spoken words into written text, it enables easier communication and access to digital content for users with diverse needs.



❖ **Flowchart**



4. Literature Review

In today's era, there is a notable trend towards empowering machines to emulate human cognitive abilities and autonomously perform tasks traditionally carried out by humans. This shift has given rise to the concept of voice assistants, which can execute various tasks based on voice commands provided by users. These assistants excel at deciphering commands and retrieving relevant information. Globally, individuals are embracing cutting-edge technologies such as virtual reality, augmented reality, and voice interaction to enhance their digital experiences. Voice control represents a significant advancement in human-machine interaction, involving the conversion of analog speech signals into digital waves.

The widespread adoption of smartphones in recent years has fuelled the popularity of voice assistants like Apple's Siri, Google's Assistant, Microsoft's Cortana, and Amazon's Alexa. Leveraging technologies such as voice recognition, speech synthesis, and Natural Language Processing (NLP), these assistants offer an array of services aimed at simplifying users' lives.

Voice assistants provide a range of functionalities, including answering questions posed by users, playing music from streaming services, and even playing YouTube videos. They can set timers or alarms to help users manage their time effectively and send messages via platforms like WhatsApp or email, enabling seamless communication. Additionally, voice assistants offer real-time weather updates, providing users with valuable information about current conditions and forecasts.

Moreover, these assistants continuously evolve to meet the evolving needs of users. According to Deepak Shende and Ria Umabiya, the development of voice-controlled personal assistants like AIVA (2018) by Microsoft, Google Assistant, and AIVA itself has introduced novel features such as social media interactions and real-time internet searches. These assistants can perform tasks like posting comments on platforms like Facebook and Twitter, further enhancing user convenience and efficiency.

Furthermore, Tulshan highlights the potential risks of repetitive typing, which may lead to finger injuries. To mitigate such issues, there is a growing need to design systems that enable task completion through voice commands. Such systems utilize voice recognition to interpret user commands, which are then synthesized and displayed on the screen if deemed appropriate. Additionally, specific keywords can trigger program compilation and execution, further enhancing user convenience and efficiency.

5. Problem Statement

Traditional desktop interfaces lack efficient voice interaction capabilities, hindering users' ability to navigate their desktop environments seamlessly. While voice-controlled systems have gained popularity in recent years, they often struggle to handle uncertain queries effectively, limiting their usefulness in real-world scenarios.

Users encounter difficulties in accessing files, launching applications, and performing tasks using voice commands alone. Additionally, existing voice assistants often fail to provide satisfactory responses to ambiguous or contextually complex queries, leading to frustration and decreased user satisfaction. This gap in functionality hinders the potential of voice-controlled interfaces to revolutionize desktop computing and improve user productivity.

Therefore, there is a pressing need for a more sophisticated desktop voice assistant solution that integrates advanced natural language processing capabilities to enhance user interaction and provide intelligent responses to uncertain queries. Such an assistant would empower users to interact with their desktop environments more seamlessly, reducing reliance on traditional input methods and streamlining workflow. By addressing the limitations of current desktop interfaces, this solution aims to offer users a more intuitive and efficient computing experience, ultimately unlocking the full potential of voice-controlled technology in the desktop computing domain.

5.1 Scope of the Project

- The project will focus on developing a proof-of-concept Desktop Voice Assistant prototype with integrated NLP capabilities.
- Intent classification and entity recognition will be implemented to interpret user commands and extract relevant information.
- ChatGPT will be integrated to provide intelligent responses to uncertain queries, enhancing the assistant's conversational abilities.
- Compatibility testing will be conducted to ensure the assistant works seamlessly across different desktop operating systems and applications.
- The project scope does not include advanced features such as multi-language support or extensive customization options, which may be considered in future iterations.

6. Requirement Analysis

The objective of the Voice Assistant project is to create a resilient and intelligent system enabling users to interact with their computers through natural language voice commands. This system will utilize speech recognition, natural language processing (NLP), and assorted APIs to comprehend user commands, perform tasks, and furnish responses.

6.1 Stakeholders

1. Users: Individuals who engage with the voice assistant to carry out tasks and obtain information.

2. Developers: The team entrusted with the design, development, testing, and upkeep of the voice assistant system.

3. System Administrators: Those responsible for deploying and managing the voice assistant in organizational settings.

4. Third-party Service Providers: Providers of APIs and services integrated with the voice assistant for enhanced functionality.

6.2 Functional Requirements

1. Voice Command Recognition

- The system must accurately recognize voice commands spoken by users.
- It should support a wide range of natural language commands for performing tasks such as opening applications, searching the web, and controlling system settings.
- The voice recognition engine should have high accuracy and be able to handle variations in pronunciation and accents.
- It should provide feedback to the user to indicate when a command has been recognized or if clarification is needed.

2. Task Execution

- The system should be capable of executing various tasks based on user commands, including:
    1. Opening applications and files.
    2. Performing web searches and retrieving information.
    3. Setting reminders, alarms, and calendar events.
    4. Controlling system settings such as volume, brightness, and Wi-Fi connectivity.
- It should seamlessly integrate with the underlying operating system to perform system-level actions.
- The system should prioritize and execute tasks based on contextual relevance and user preferences.

3. Chat GPT Integration:

- The system should integrate with Chat GPT to handle uncertain or different questions asked by the user.
- It should utilize Chat GPT to generate appropriate responses for queries that cannot be directly fulfilled by the system's predefined commands.

4. Response Generation

- Generate appropriate responses to user queries using synthesized speech and/or text-based messages.
- Provide informative and contextually relevant responses that address the user's query or command.
- Integrate with external APIs and services to fetch real-time data and deliver up-to-date information to the user.

5. Error Handling

- Gracefully handle errors and exceptions that may occur during command recognition or task execution.
- Provide clear and concise error messages to the user, suggesting possible solutions or alternative commands.
- Implement logging and error-tracking mechanisms to monitor system performance and identify recurring issues.

6.3 Non-Functional Requirements

1. Performance

- The system should respond to voice commands and execute tasks within an acceptable timeframe to maintain user satisfaction.
- Minimize latency in voice recognition and response generation to provide a seamless user experience.
- Optimize resource utilization to ensure efficient performance, even under heavy usage or concurrent user interactions.

2. Security

- Implement robust authentication and authorization mechanisms to prevent unauthorized access to sensitive user data and system resources.
- Encrypt communication channels to protect user privacy and confidentiality.
- Regularly update and patch system components to mitigate security vulnerabilities and threats.

3. Usability

- Design an intuitive and user-friendly interface for interacting with the voice assistant, catering to users of all technical skill levels.
- Provide customizable settings and preferences to allow users to personalize their experience with the voice assistant.
- Offer contextual help and guidance to users when they encounter unfamiliar commands or functionalities.

4. Accessibility

- Ensure accessibility features are incorporated into the design to accommodate users with disabilities, including visual or motor impairments.
- Support alternative input methods such as keyboard shortcuts or gestures for users who may have difficulty with voice commands.
- Conduct usability testing with diverse user groups to identify and address accessibility barriers.

5. Constraints

- The system should be compatible with a wide range of hardware and operating systems to maximize accessibility for users.
- Consideration should be given to the limitations of speech recognition and natural language processing technologies, such as accuracy and language support.
- Ensure compliance with data privacy, security, and accessibility regulations and standards during the development and deployment of the voice assistant.

6. Assumptions

- Users have access to a microphone-enabled device capable of running the voice assistant application.
- The voice assistant will rely on internet connectivity to access external services and APIs for enhanced functionality.
- Users will receive appropriate training and guidance on how to interact with the voice assistant effectively.

7. System Analysis

7.1 Components

- **Voice Recognition Module:** This component is responsible for converting spoken commands from the user into text format that the system can understand. It utilizes external APIs or libraries for speech recognition, such as Google's Speech Recognition API or Python's speech recognition library.
- **Application/File Management Module:** The application/file management module handles tasks related to opening files, launching applications, and managing desktop resources. It interacts with the operating system's file system and application launch mechanisms to execute user commands.
- **Chat GPT Integration Module:** This module integrates OpenAI's ChatGPT model into the system, enabling natural language understanding and response generation. It interacts with external APIs to communicate with the ChatGPT model and process user queries.

7.2 Interactions

- **User Input:** User input consists of spoken commands directed at the voice assistant. These commands are captured by the voice recognition module and converted into text format.
- **Processing Logic:** Once the user input is transcribed, the system's processing logic determines the appropriate action based on the input. Simple commands, such as opening a file or launching an application, are handled by the application/file management module. More complex queries are passed to the Chat GPT integration module for analysis.
- **System Outputs:** The system generates outputs based on the processed input. For straightforward commands, such as opening a file, the output consists of executing the requested action. For more complex queries, the output includes generated responses from the ChatGPT model.

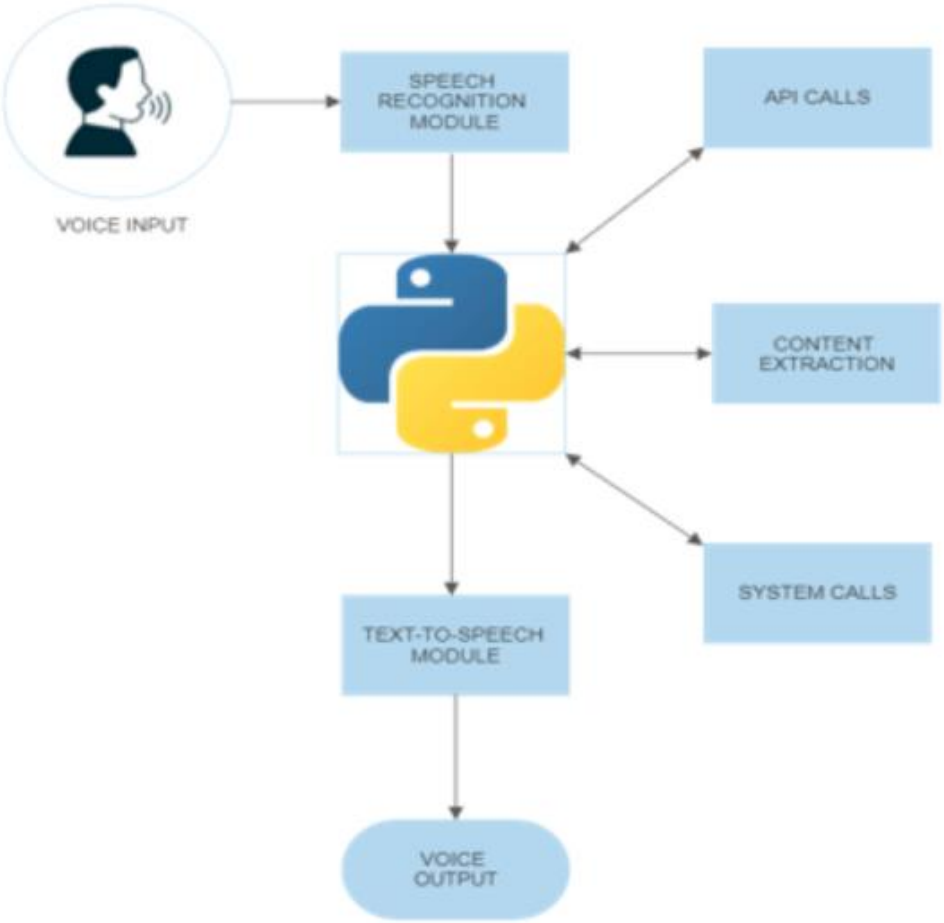
7.3 Dependencies

- **External APIs:** The system relies on external APIs for speech recognition and ChatGPT integration. These APIs provide the necessary functionality for transcribing user input and generating responses.
- **System Resources:** The voice assistant system consumes system resources such as CPU and memory during operation. Resource utilization may vary depending on factors such as the complexity of user commands and the processing requirements of the ChatGPT model. Efficient resource management is essential to ensure optimal system performance.

8. System Methodology

Voice assistants can provide relevant information based on specific commands, often referred to as intents, spoken by the user. By listening for predetermined keywords and filtering out ambient noise, these assistants retrieve and deliver requested data. Today, voice assistants are seamlessly integrated into various devices we use daily, including cell phones, computers, and smart speakers. Their widespread integration allows users to access their capabilities across different platforms.

Different voice assistants offer distinct feature sets; some are tailored to specific functions, while others maintain an open-ended approach to address a wide range of user needs. This diversity in functionality enables users to choose voice assistants that best suit their requirements and preferences, enhancing their overall experience with voice technology.



- **Speech Recognition:**

  The system utilizes Google's online speech recognition system to accurately transcribe speech input into text format. Users can access texts from specialized corpora stored on the network server at the information center, ensuring efficient data retrieval. These texts are temporarily stored within the system before being forwarded to Google Cloud for advanced speech recognition processing. Once recognized, the equivalent text is seamlessly integrated into the central processor for further analysis and action.

- **Python Backend:**

  The Python backend serves as the intelligent interpreter of the voice recognition module's output, making critical determinations regarding the nature of the user's command or speech output. Beyond categorizing these outputs as API Calls, Context Extraction, or System Calls, it orchestrates the flow of information, ensuring that user requests are efficiently processed and appropriate responses are generated. The backend plays a pivotal role in facilitating seamless communication between the user interface and underlying system functionalities.

- **API Calls:**

  APIs, or Application Programming Interfaces, establish a standardized means for applications to communicate and interact with one another. Acting as mediators, they facilitate the transmission of user requests to service providers and subsequent retrieval of responses. By serving as efficient messengers between different software components, APIs enable seamless integration and interoperability, fostering a cohesive and interconnected digital ecosystem.

- **Content Extraction:**

  Context extraction involves automatically extracting structured information from unstructured or semi-structured documents, often using natural language processing (NLP). This process can extend to multimedia document processing, including annotation and content extraction from images, audio, and video.

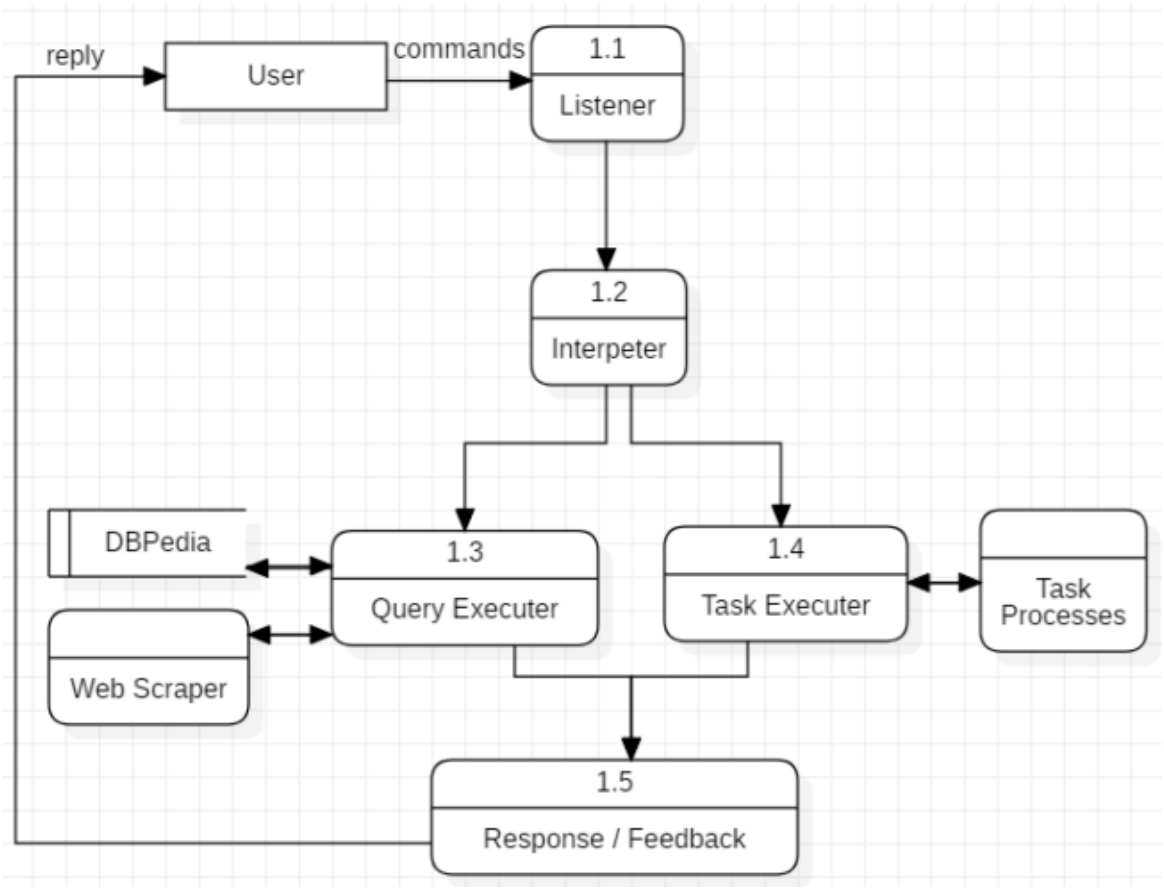
- **System Calls:**

  System calls serve as fundamental mechanisms through which computer programs solicit essential services from the underlying operating system kernel. These services encompass a wide spectrum of functionalities, including hardware-related tasks, process management, and communication with core kernel services. By bridging the gap between user-level processes and low-level system operations, system calls facilitate the seamless execution of diverse computational tasks, ensuring optimal performance and resource utilization.

- **Text-to-Speech:**

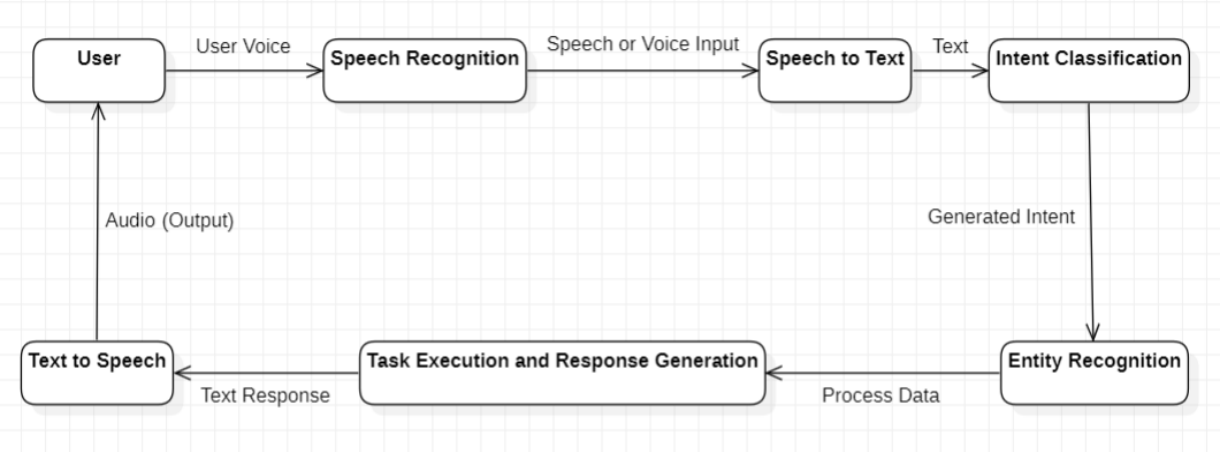
  Text-to-speech (TTS) enables computers to audibly read written text. Written text is converted into a phonemic representation, which is then synthesized into sound waves by a TTS engine. Various third-party TTS engines support multiple languages, dialects, and specialized vocabularies.

❖ Data Flow Diagram



9. System Design

❖ Working Model



i. Speech Recognition

Speech Recognition is a dynamic field merging computer science and linguistics to transcribe spoken words into text. It employs sophisticated algorithms and language models to decipher human language effectively. The process involves capturing audio input, processing it to extract features and filter noise, and analyzing speech sounds and patterns. Language models contextualize recognized words, considering grammar, syntax, and semantics, while machine learning algorithms enhance accuracy and adaptability.

Speech recognition facilitates hands-free device operation, automates transcription tasks, aids communication for individuals with disabilities, and enhances customer support and healthcare documentation.

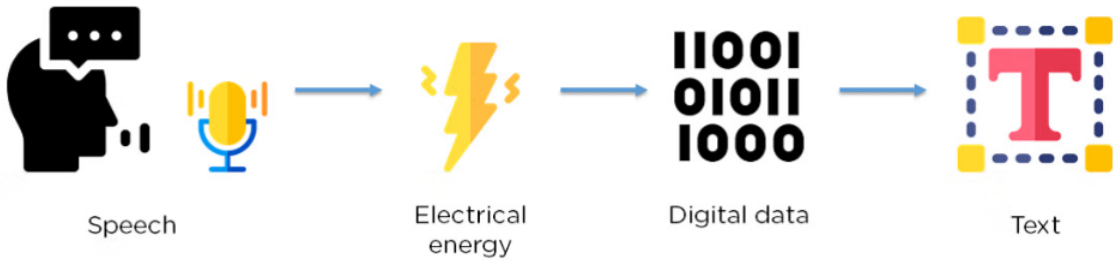
- Firstly we will install a speech_recognition package to record audio from the microphone and perform speech recognition using Google's speech recognition service.
- speech_recognition.Recognizer(): This function recognizes the speech input given by the user from the microphone.

```
def takeCommand():
    r = speech_recognition.Recognizer()
    with speech_recognition.Microphone() as source:
        print("Listening.....")
        r.pause_threshold = 1
        r.energy_threshold = 300
        audio = r.listen(source,0,4)
```

- Opens the microphone as a source for audio input.
- Prints "Listening....." to indicate that the program is ready to listen for audio input.
- Sets the pause threshold and energy threshold for the recognizer object. These thresholds determine when the recognizer should consider speech to have ended.
- Listens for audio input for up to 4 seconds, with no minimum pause between words (pause_threshold = 1 and energy_threshold = 300).

ii. Speech-To-Text

Speech recognition begins by capturing the sound energy produced by the speaker using a microphone. This sound energy is then transformed into electrical signals, transitioning from analog to digital format. Once digitized, sophisticated algorithms analyze the digital signal to identify patterns and features corresponding to spoken words. These patterns are compared against a vast database of linguistic models to interpret and transcribe the speech into text accurately.



- r.recognize_google(audio, language='en-in'): Recognize the speech in the captured audio (**audio**) and convert it to text

```
try:
    print("Understanding..")
    query   = r.recognize_google(audio,language='en-in')
    print(f"You Said: {query}\n")
```

- Once audio input is captured, prints "Understanding.." to indicate that the program is processing the captured audio.
- Uses Google's speech recognition service (recognize_google() method) to convert the captured audio into text. The recognized text is assigned to the variable query.
- Prints the recognized text.

iii. Intent Classification

Intent classification is the process of determining the underlying intention or purpose behind a user's input. In the context of the provided code, intent classification entails the identification of the user's intention by analyzing the content of their query and then executing the corresponding functionality. Here's how intent classification is implemented:

1. **User Input Processing:**

The user's input is captured through speech recognition and converted into text using the takeCommand() function.

The recognized text is stored in the query variable.

2. Intent Identification:

Conditional statements (if, elif, else) are used to analyze the content of the user's query and identify specific intents based on keywords or phrases present in the input.

Each conditional block corresponds to a different intent or task that Jarvis can perform.

Intents can include waking up Jarvis, searching the web, retrieving information, setting alarms, controlling media playback, etc.

3. Execution of Intent-specific Functionality:

Once the intent is identified, the code executes the corresponding functionality associated with that intent.

For example, if the intent is to wake up Jarvis, the code calls the greetMe() function to greet the user and initiate interaction.

- **Intent Classification Example:**

  Video Playback Control:

  - **Intent:** The user wants to control Video playback.
  - **Example Query:** "Pause the video."
  - **Implementation:** Recognize commands like "pause," "play," "mute," etc., and execute the appropriate action to control video playback.

iv. Entity Recognition

Entity recognition entails the identification and extraction of particular entities or information fragments from the user's input. In the provided code, entity recognition may not be explicitly implemented as a separate component. However, certain intents may require extracting additional information or parameters from the user's query. Here's how entity recognition could be incorporated:

1. Task-specific Functions:

Some task-specific functions may require additional parameters to perform their tasks effectively.

For example, when setting an alarm, the user may need to specify the time for the alarm to be set. In such cases, the code may extract relevant entities such as time, location, or other parameters from the user's query to provide context for the task execution.

- **Entity Recognition Example:**

  Web Search with Specific Query:

  - **Intent:** The user wants to perform a web search with a specific query.
  - **Example Query:** "Wikipedia on Artificial Intelligence."
  - **Implementation:** Extract the search query "Artificial Intelligence" from the user input and pass it as a parameter to the function responsible for searching Wikipedia.

v. Task Execution and Response Generation

1. **Task Execution:** Upon identifying and validating the user's intent and entities, the system proceeds to execute the relevant task or action. This process entails invoking specific functions, modules, or APIs to carry out various tasks, such as retrieving data from online sources, managing devices, scheduling alarms, or delivering requested information.

2. **Response Generation:** Following task execution, the system generates a response to provide feedback to the user. This response is crafted based on the outcome of the task and can take various forms, including spoken output using text-to-speech synthesis, visual feedback presented on a user interface, or other communication methods tailored to the application's interface design. The response aims to inform the user of the action taken and ensure clarity and satisfaction with the system's performance.

vi. Text-To-Speech

Text-to-speech (TTS) technology revolutionizes digital communication by enabling computers to transform written text into spoken words. With a wide array of applications, TTS enhances accessibility for visually impaired individuals, providing them with auditory access to digital content. Moreover, it facilitates the development of interactive voice-based systems, enriching user experiences across various platforms.

TTS operates by analyzing textual input and synthesizing it into natural-sounding audio output, mimicking human speech patterns and intonations. This seamless conversion process enables users to perceive and comprehend the content audibly, eliminating the need for reading.

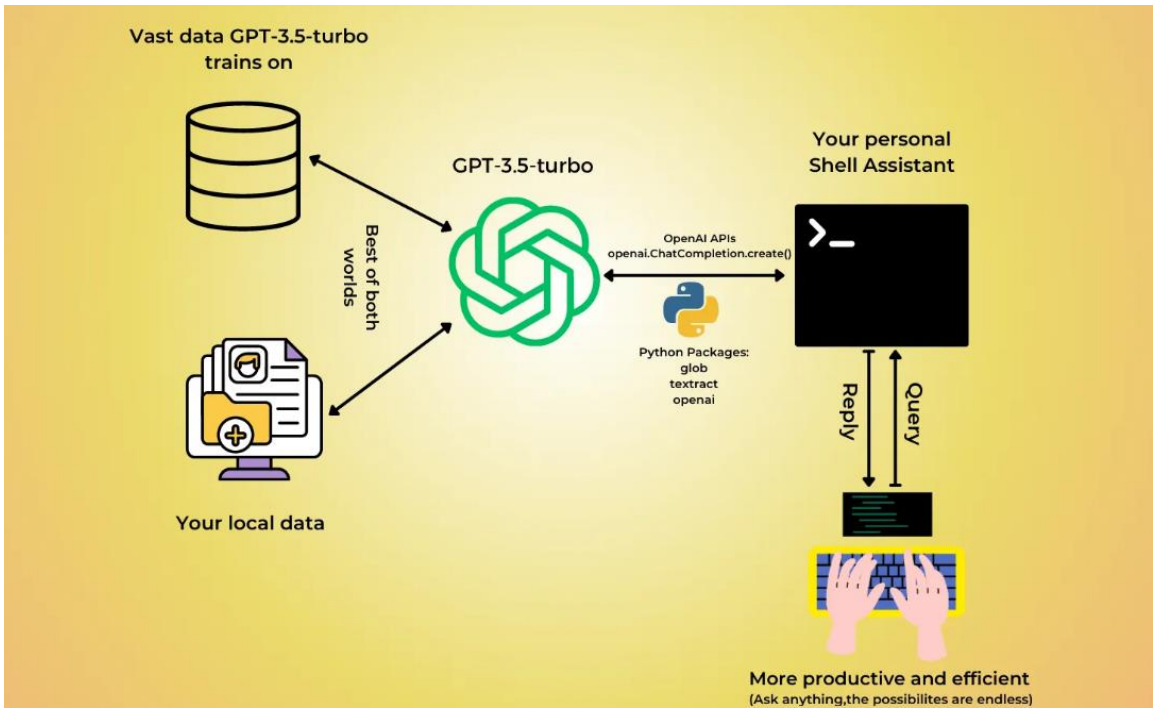
```
engine = pyttsx3.init("sapi5")
voices = engine.getProperty("voices")
engine.setProperty("voice", voices[0].id)
rate = engine.setProperty("rate",170)
```

- **Pyttsx3:** takes text as input and converts it to speech using the initialized text-to-speech engine.
- **pyttsx3.init("sapi5"):** Initializes the text-to-speech engine using Microsoft's Speech API (sapi5).
- **engine.setProperty("rate", 170):** Sets the speaking rate of the text-to-speech engine to 170 words per minute.

```
def speak(audio):
    engine.say(audio)
    engine.runAndWait()
```

we provide the text we want to convert to speech using the say() method, and finally, we run the text-to-speech conversion using runAndWait()

❖ Chat GPT Integration



1. Integration with GPT-3.5 Turbo:

- The voice assistant utilizes OpenAI's GPT-3.5 Turbo model, renowned for its advanced natural language processing capabilities.
- GPT-3.5 Turbo is utilized to understand user queries and generate contextually relevant responses, ensuring a conversational interaction experience.

2. Prompt Construction for GPT-3.5 Turbo:

- Before sending a query to GPT-3.5 Turbo, the system constructs a prompt that provides context for the model.
- The prompt format includes the user's query prefixed with "Chando:", indicating the user's input, followed by "\n Jarvis:", which marks the beginning of the assistant's response generation.

3. Response Generation with GPT-3.5 Turbo:

- GPT-3.5 Turbo processes the constructed prompt and generates responses based on the provided context and learned patterns from vast amounts of text data.
- The responses generated by GPT-3.5 Turbo aim to address the user's query or command in a conversational manner, mimicking human-like interaction.

4. Text-to-Speech Conversion:

- Once the responses are generated by GPT-3.5 Turbo, they are converted into speech using the `pyttsx3` library.
- This conversion process enables the system to audibly deliver the responses to the user, facilitating seamless interaction without the need for textual reading.

5. User Interaction Flow:

- The voice assistant follows a continuous interaction flow, where it constantly listens for user commands through speech recognition.
- Upon detecting a command, the system captures the audio input, transcribes it into text, and sends it to GPT-3.5 Turbo for processing and response generation.

6. Error Handling Mechanisms:

- Robust error-handling mechanisms are implemented to ensure the reliability and stability of the system.

- These mechanisms address potential issues during speech recognition, API connectivity, or response generation, providing a smooth user experience even in challenging conditions.

7. Integration with External APIs:

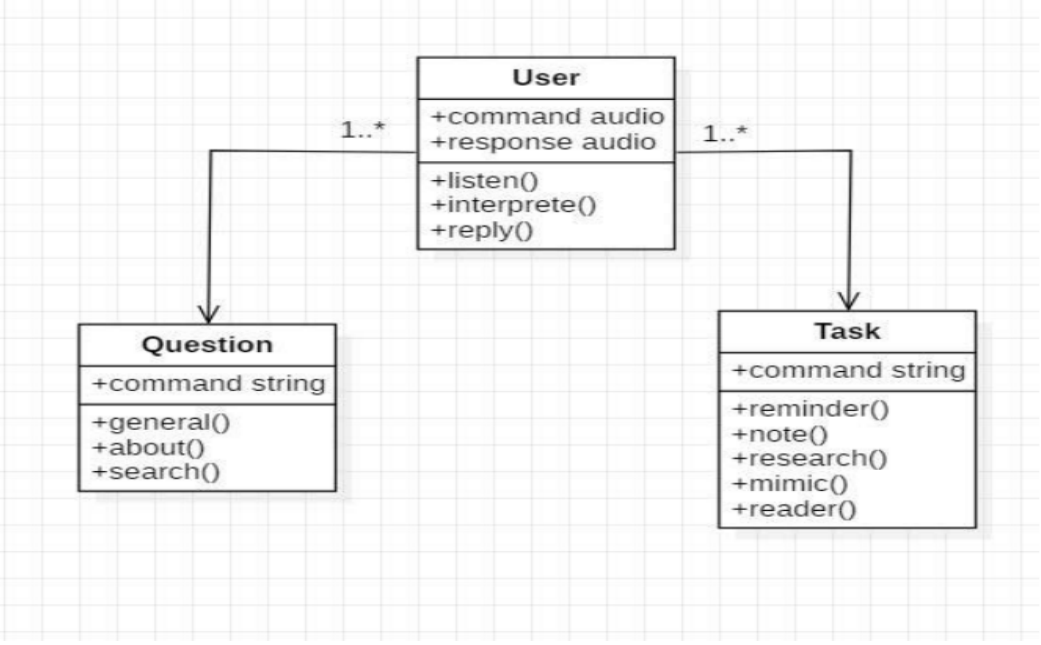
- The voice assistant integrates seamlessly with external APIs, including OpenAI's API for natural language processing and Google's speech recognition service.
- This integration enables the system to leverage advanced technologies for accurate transcription of user speech and sophisticated response generation.

8. Seamless Interaction Experience:

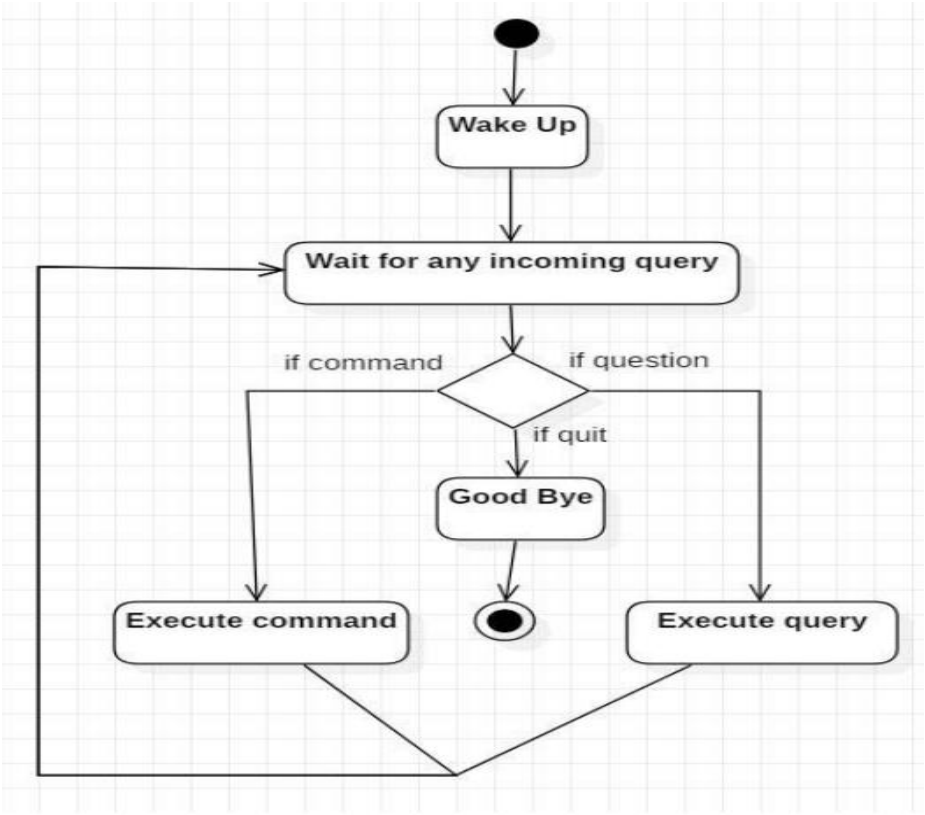
- Through the integration of GPT-3.5 Turbo, speech recognition, and text-to-speech conversion, the voice assistant offers a seamless interaction experience to users.
- Users can communicate naturally with the assistant, speaking commands or queries, and receiving informative responses audibly, enhancing user engagement and satisfaction.

❖ UML Diagrams

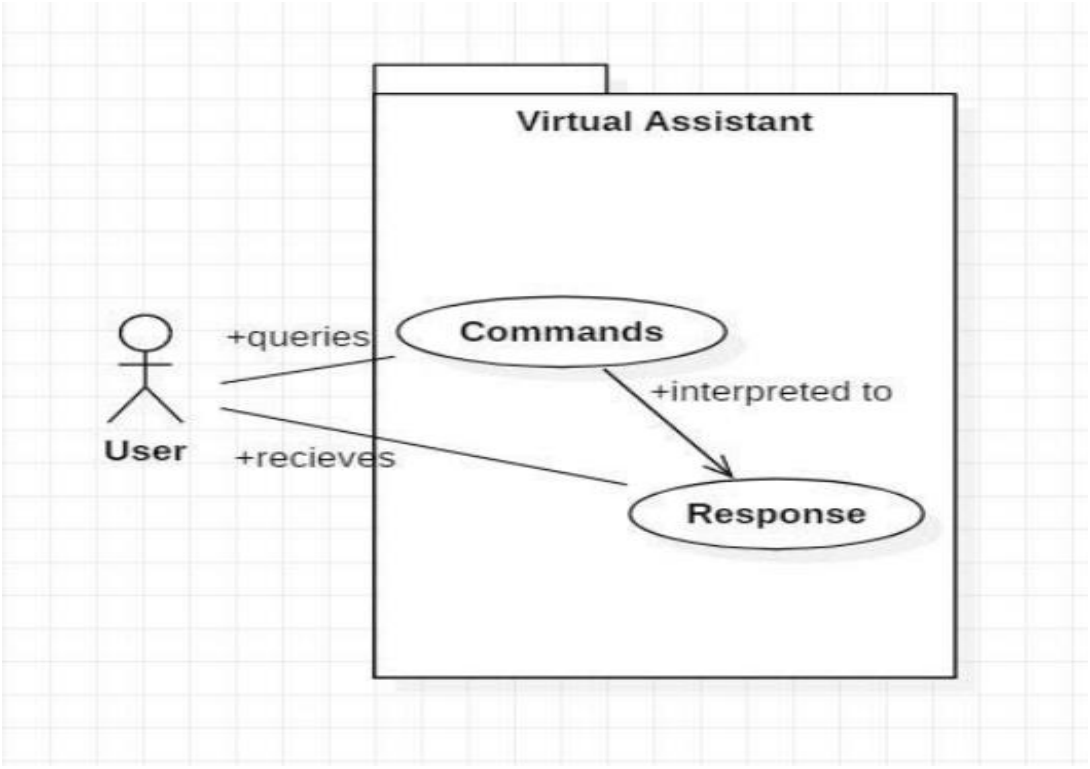
- **Class Diagram:**



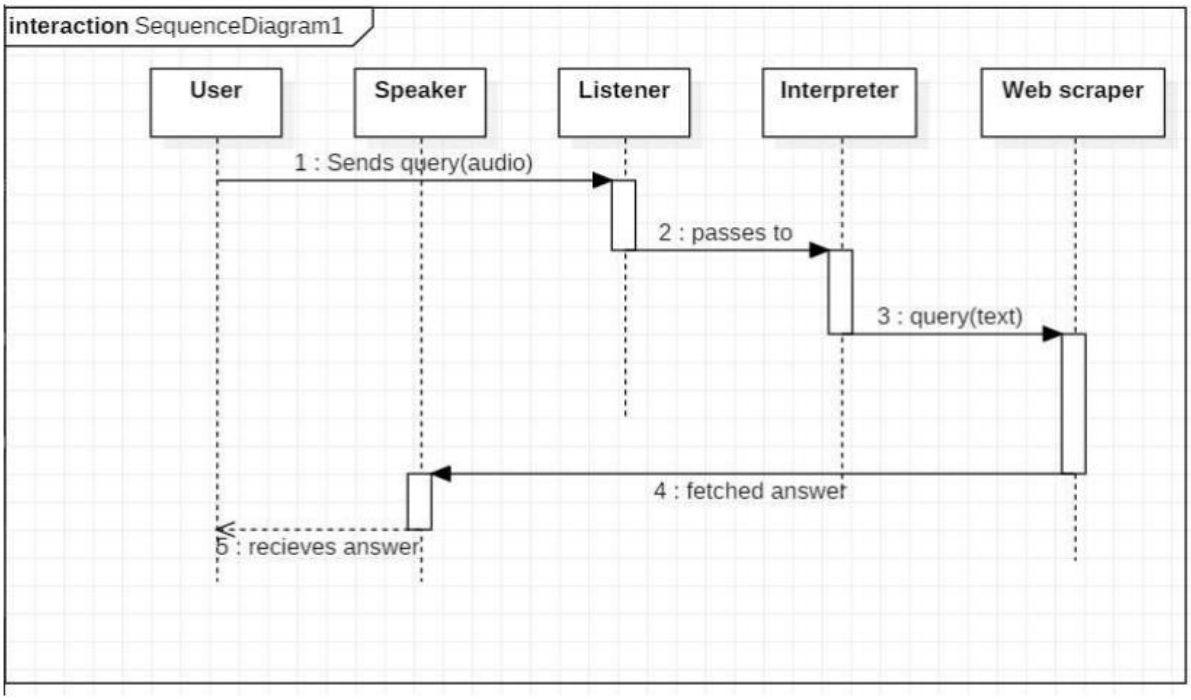
- **Activity Diagram:**

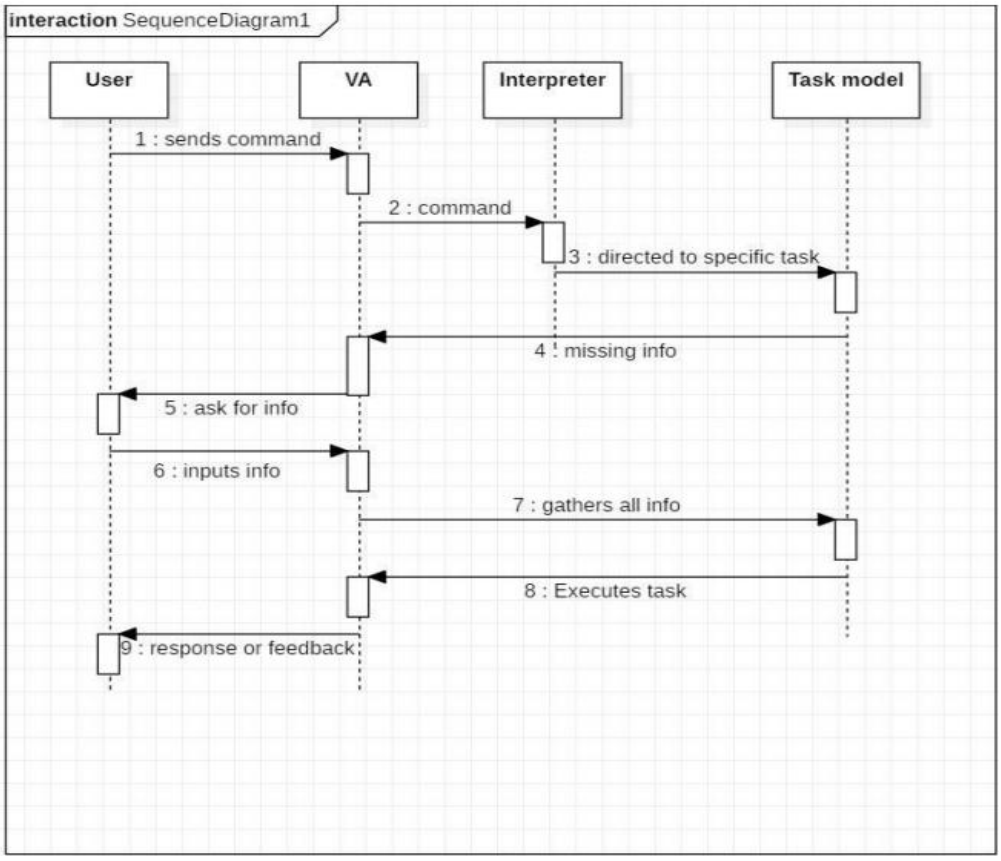


- **Use case Diagram:**

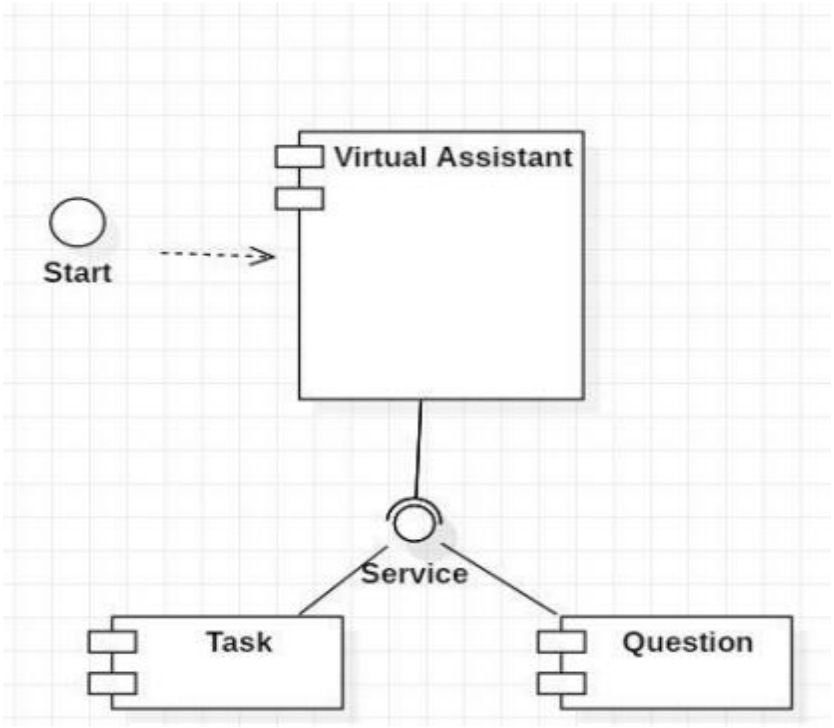


- **Sequence Diagram:**





- **Component Diagram**



10. Implementation

To develop a voice assistant, two core functions are vital: one to capture user commands and another to generate responses. These functions should be complemented by customized instructions for the assistant. Essentially, the assistant needs mechanisms to listen to user input and provide appropriate feedback. Customized instructions enable it to perform specific actions or provide relevant information. By integrating these elements effectively, developers can create versatile voice assistant's adept at assisting users in a variety of tasks. The required modules for a voice assistant in Python: are Speech Recognition, Wikipedia, Web browser, os, pyttsx3, and Datetime.

10.1 Modules Description

1. Pttsx3: A Python library facilitating text-to-speech conversion is available. It operates offline and supports both Python 2 and Python 3. Installation is straightforward via the "pip install pyttsx3" command. Applications utilize the "pyttsx3.init()" factory function to access a pyttsx3 Engine instance, simplifying the process. This tool seamlessly converts text to speech, offering user-friendly functionality. Additionally, the pyttsx3 module offers two voices: female and male, leveraging the "sapi5" feature for Windows platforms.

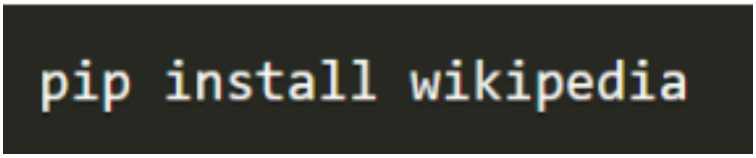
```
pip install pyttsx3
```

2. Speech Recognition: Speech recognition is a versatile library designed for executing speech recognition tasks using various engines and APIs, both online and offline. It specializes in identifying voice commands and transcribing them into text format.

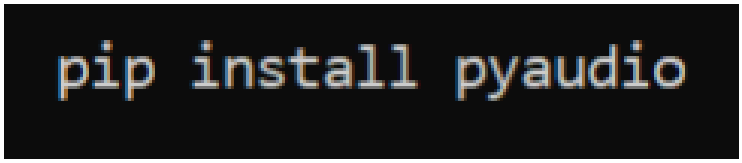
```
pip install speechRecognition
```

3. Wikipedia: The Wikipedia library enables access to information pertaining to user queries from Wikipedia. Wikipedia, a multilingual online encyclopedia, is curated through

collaborative efforts of volunteer editors utilizing a wiki-based editing system. To retrieve data from Wikipedia, installation of the Python Wikipedia library is required, as it serves as a wrapper for the official Wikipedia API.



5. Pyaudio: Pyaudio serves as a Python interface for PortAudio, a versatile library for audio input and output across multiple platforms. This integration enables recording and playback of sound on various operating systems including Windows, Mac, and Linux. PyAudio conveniently provides Python bindings for PortAudio, allowing seamless utilization of Python for audio-related tasks such as playing and recording audio across diverse platforms.



6. Datetime: This module plays a vital role in supporting date and time functionalities. Users can utilize it to retrieve the current date and time or schedule tasks for specific times, enhancing their productivity and organization.

7. PyAutoGUI: PyAutoGUI is a Python package that grants control over both the mouse and keyboard, allowing for the simulation of mouse cursor movements and button clicks. By specifying specific 2-D coordinates, precise clicking at exact screen locations is achievable.

8. PyWhatkit: PyWhatKit is a Python Library that has several features like Sending messages, and images through WhatsApp, playing YouTube videos, converting images to ASCII, sending emails, etc.

9. SpeedTest: SpeedTest: Speedtest library is essential to test internet bandwidth. It helps to evaluate the uploading as well as the downloading speed of the Internet. All the results that we get are in Megabits.

10. tk: This package provides a Tk GUI toolkit, which you might be using for building graphical user interfaces (GUIs) in your project.

11. Pillow: Pillow stands as a potent Python Imaging Library (PIL) fork, renowned for its prowess in image processing. It excels in tasks such as opening, editing, and saving various image file formats.

12. openai==0.28: OpenAI is an artificial intelligence research laboratory. This specific version might be used for accessing OpenAI's API and integrating AI capabilities into your project.

13. woframalpha: Wolfram Alpha is a computational knowledge engine. This package allows you to query the Wolfram Alpha service for answers to factual questions.

14. gtts: GTTS (Google Text-to-Speech) is both a Python library and a command-line interface tool, facilitating interaction with Google Translate's text-to-speech API. Its primary function is converting text into speech.

15. wheel: Wheel is a built-in package format for Python. It's used for distributing pre-built Python libraries, which can be installed more quickly than installing from a source.

16. Playsound: Playsound is a simple library for playing sound files. It's used for playing sound effects or audio prompts in your project.

17. Pygame: Pygame consists of a collection of Python modules specifically crafted for the development of video games. It's commonly used for multimedia applications, including game development and interactive media projects.

18. Pynput: pynput is a Python library for controlling and monitoring input devices such as mouse and keyboard. It allows your project to listen to keyboard and mouse events.

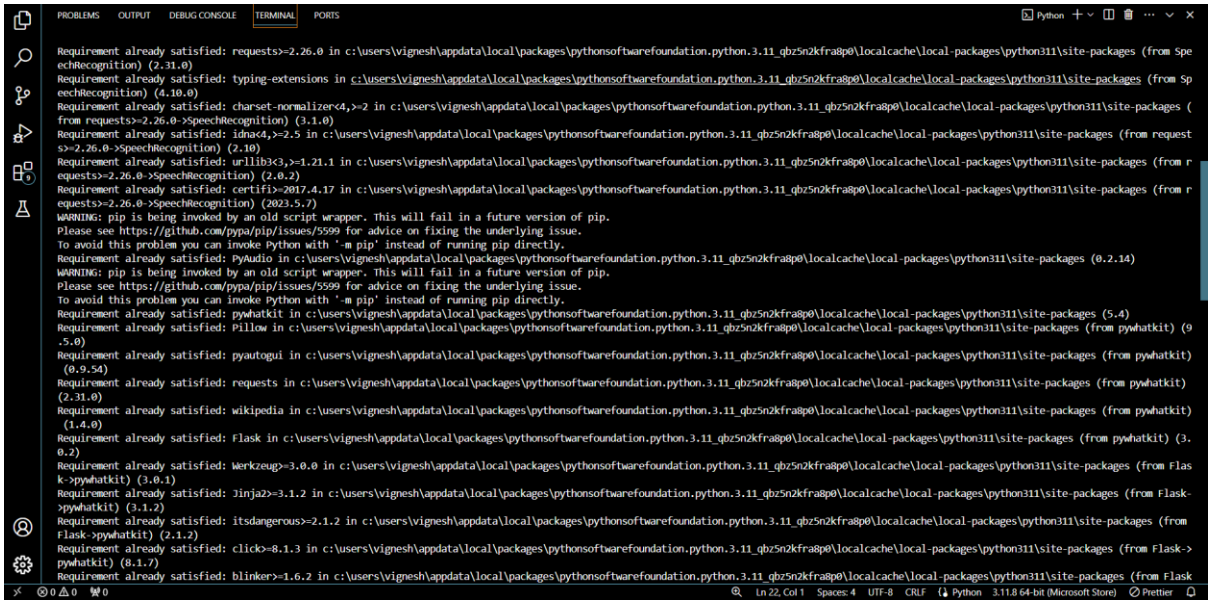
19. beautifulsoup4: Beautiful Soup serves as a Python library designed to extract data from HTML and XML files. It's commonly used for web scraping tasks, enabling your project to extract information from web pages.

11. Analysis and Results

- **Installer.py - (Installed Packages)**

```
import pip
pip.main(['install','wikipedia'])
pip.main(['install','pyttsx3'])
pip.main(['install','SpeechRecognition'])
pip.main(['install','PyAudio'])
pip.main(['install','pywhatkit'])
pip.main(['install','beautifulsoup4'])
pip.main(['install','dict'])
pip.main(['install','DateTime'])
pip.main(['install','pynput'])
pip.main(['install','pyautogui'])
pip.main(['install','wolframalpha'])
pip.main(['install','googletrans'])
pip.main(['install','gtts'])
pip.main(['install','wheel'])
pip.main(['install','playsound'])
pip.main(['install','pygame'])
pip.main(['install','speedtest-cli'])
pip.main(['install','tk'])
pip.main(['install','pillow'])
pip.main(['install','openai==0.28'])
```

➢ **Output**



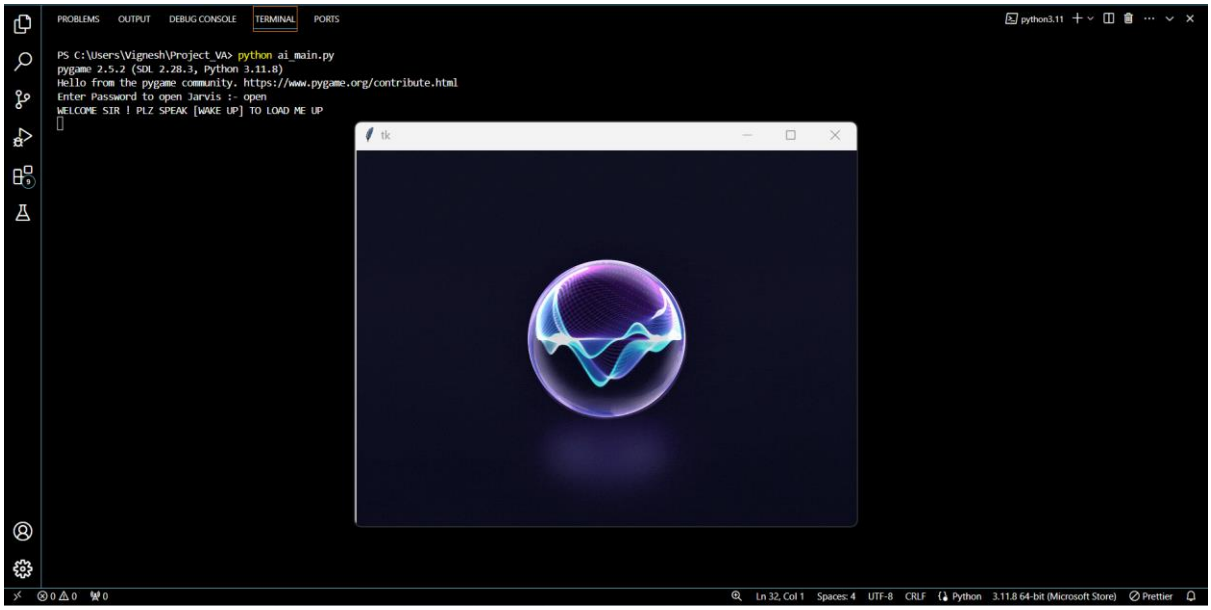
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Requirement already satisfied: requests>=2.26.0 in c:\users\vignesh\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from SpeechRecognition) (2.31.0)
Requirement already satisfied: typing-extensions in c:\users\vignesh\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from SpeechRecognition) (4.10.0)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\vignesh\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from requests>=2.26.0->SpeechRecognition) (3.1.0)
Requirement already satisfied: idna<4,>=2.5 in c:\users\vignesh\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from requests>=2.26.0->SpeechRecognition) (2.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\vignesh\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from requests>=2.26.0->SpeechRecognition) (2.0.2)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\vignesh\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from requests>=2.26.0->SpeechRecognition) (2023.5.7)
WARNING: pip is being invoked by an old script wrapper. This will fail in a future version of pip.
Please see https://github.com/pypa/pip/issues/5599 for advice on fixing the underlying issue.
To avoid this problem you can invoke Python with '-m pip' instead of running pip directly.
Requirement already satisfied: PyAudio in c:\users\vignesh\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (0.2.14)
WARNING: pip is being invoked by an old script wrapper. This will fail in a future version of pip.
Please see https://github.com/pypa/pip/issues/5599 for advice on fixing the underlying issue.
To avoid this problem you can invoke Python with '-m pip' instead of running pip directly.
Requirement already satisfied: pywhatkit in c:\users\vignesh\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (5.4)
Requirement already satisfied: Pillow in c:\users\vignesh\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from pywhatkit) (9.5.0)
Requirement already satisfied: pyautogui in c:\users\vignesh\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from pywhatkit) (0.9.54)
Requirement already satisfied: requests in c:\users\vignesh\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from pywhatkit) (2.31.0)
Requirement already satisfied: wikipedia in c:\users\vignesh\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from pywhatkit) (1.4.0)
Requirement already satisfied: Flask in c:\users\vignesh\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from pywhatkit) (3.0.2)
Requirement already satisfied: Werkzeug>=3.0.0 in c:\users\vignesh\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from Flask->pywhatkit) (3.0.1)
Requirement already satisfied: Jinja2>=3.1.2 in c:\users\vignesh\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from Flask->pywhatkit) (3.1.2)
Requirement already satisfied: itsdangerous>=2.1.2 in c:\users\vignesh\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from Flask->pywhatkit) (2.1.2)
Requirement already satisfied: click>=8.1.3 in c:\users\vignesh\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from Flask->pywhatkit) (8.1.7)
Requirement already satisfied: blinker>=1.6.2 in c:\users\vignesh\appdata\local\packages\pythonsoftwarefoundation.python.3.11_qbz5n2kfra8p0\localcache\local-packages\python311\site-packages (from Flask
```

- **Intro.py - (User Interface)**

```
from tkinter import *
from PIL import Image,ImageTk,ImageSequence
import time
from pygame import mixer
mixer.init()

root=Tk()
root.geometry("800x600")

def play_gif():
    root.lift()
    root.attributes("-topmost",True)
    global img
    img = Image.open("voice.gif")
    lbl = Label(root)
    lbl.place(x=0,y=0)
    i=0
    for img in ImageSequence.Iterator(img):
        img=img.resize((800,600))
        img = ImageTk.PhotoImage(img)
        lbl.config(image=img)
        root.update()
        time.sleep(0.05)
    root.destroy()

play_gif()
root.mainloop()
```

The code creates a desktop app that displays an animated GIF ("voice.gif") in a Tkinter window. It uses a loop to show each frame of the GIF, making it look like an animation. It provides visual feedback to the user.

➢ Output



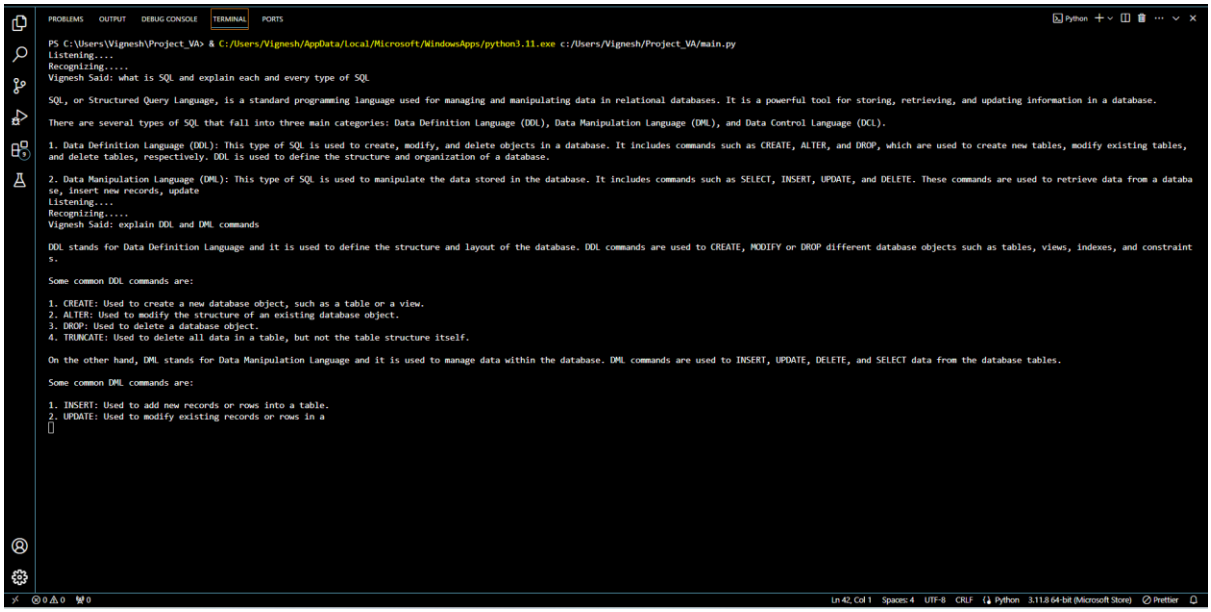
- **Main.py - (Chat GPT Integration)**

```python
import openai
from apikey import api_data
import pyttsx3
import speech_recognition as sr

openai.api_key=api_data
completion=openai.Completion()
def Reply(question):
    prompt=f'Vignesh: {question}\n Jarvis: '
    response=completion.create(prompt=prompt, engine="gpt-3.5-turbo-instruct", stop=['\Vignesh'], max_tokens=200)
    answer=response.choices[0].text.strip()
    return answer
engine=pyttsx3.init('sapi5')
voices=engine.getProperty('voices')
engine.setProperty('voice', voices[0].id)
def speak(text):
    engine.say(text)
    engine.runAndWait()
speak("Hello How Are You? ")
def takeCommand():
    r=sr.Recognizer()
    with sr.Microphone() as source:
        print('Listening....')
        r.pause_threshold = 1
        audio = r.listen(source)
    try:
        print("Recognizing.....")
        query=r.recognize_google(audio, language='en-in')
        print("Vignesh Said: {} \n".format(query))
    except Exception as e:
        print("Say That Again....")
        return "None"
    return query
if __name__ == '__main__':
    while True:
        query=takeCommand().lower()
        ans=Reply(query)
        print(ans)
        speak(ans)
```

The code implements a voice-controlled chatbot using OpenAI's GPT-3 model. It listens to the user's voice input, generates responses based on the input using GPT-3, and speaks them back using text-to-speech. This enables conversational interaction with the chatbot through voice commands, providing a seamless user experience.

1. Query: What is SQL and explain types of SQL commands

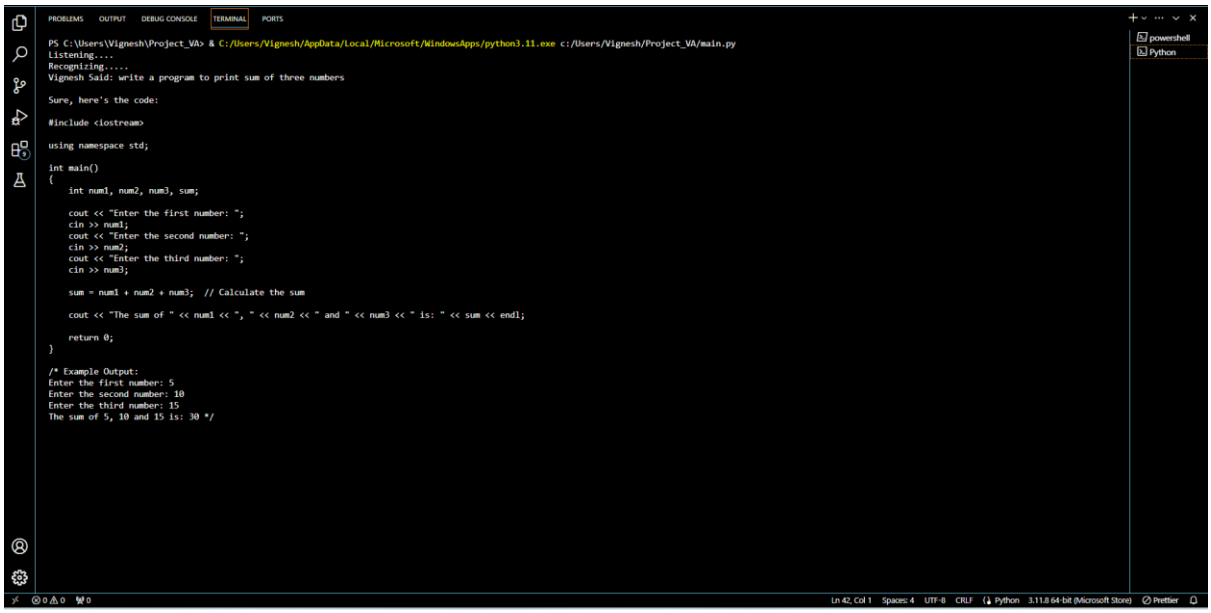
➢ Output



```
PS C:\Users\Vignesh\Project_VA> & C:/Users/Vignesh/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/Vignesh/Project_VA/main.py
Listening....
Recognizing.....
Vignesh Said: what is SQL and explain each and every type of SQL

SQL, or Structured Query Language, is a standard programming language used for managing and manipulating data in relational databases. It is a powerful tool for storing, retrieving, and updating information in a database.

There are several types of SQL that fall into three main categories: Data Definition Language (DDL), Data Manipulation Language (DML), and Data Control Language (DCL).

1. Data Definition Language (DDL): This type of SQL is used to create, modify, and delete objects in a database. It includes commands such as CREATE, ALTER, and DROP, which are used to create new tables, modify existing tables,
and delete tables, respectively. DDL is used to define the structure and organization of a database.

2. Data Manipulation Language (DML): This type of SQL is used to manipulate the data stored in the database. It includes commands such as SELECT, INSERT, UPDATE, and DELETE. These commands are used to retrieve data from a databa
se, insert new records, update
Listening....
Recognizing.....
Vignesh Said: explain DDL and DML commands

DDL stands for Data Definition Language and it is used to define the structure and layout of the database. DDL commands are used to CREATE, MODIFY or DROP different database objects such as tables, views, indexes, and constraint
s.

Some common DDL commands are:

1. CREATE: Used to create a new database object, such as a table or a view.
2. ALTER: Used to modify the structure of an existing database object.
3. DROP: Used to delete a database object.
4. TRUNCATE: Used to delete all data in a table, but not the table structure itself.

On the other hand, DML stands for Data Manipulation Language and it is used to manage data within the database. DML commands are used to INSERT, UPDATE, DELETE, and SELECT data from the database tables.

Some common DML commands are:

1. INSERT: Used to add new records or rows into a table.
2. UPDATE: Used to modify existing records or rows in a
```

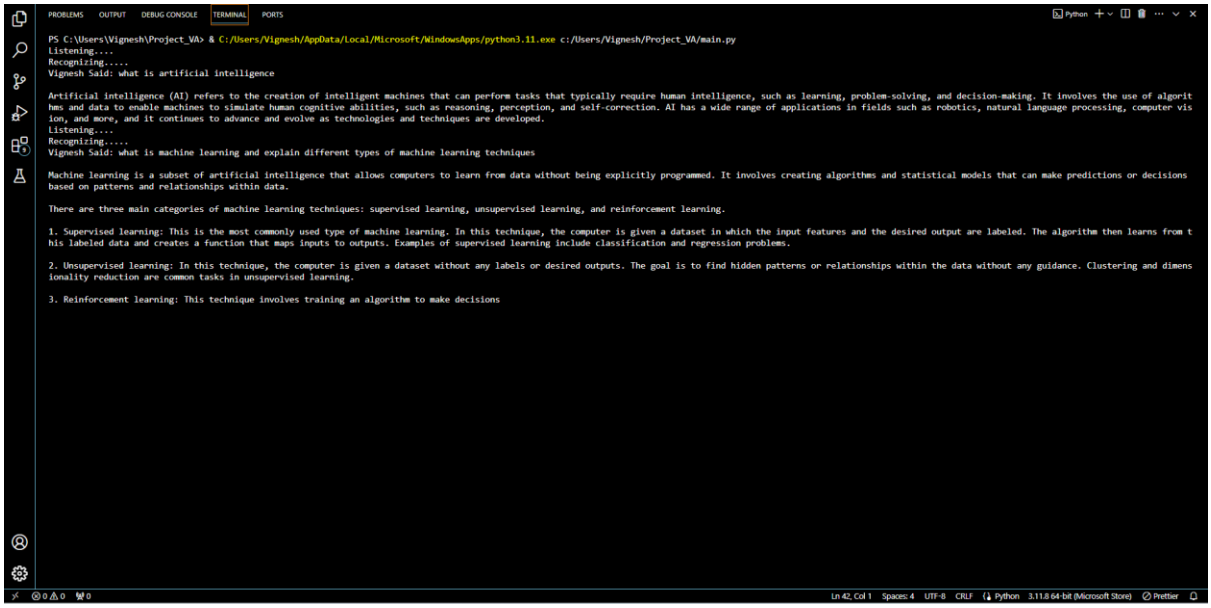
2. Query: Write a program to print the sum of three numbers

➢ Output



3. Query: What is artificial intelligence, Explain different types of machine learning

➢ Output



- **Ai_main.py – (Personal Desktop Assistant Code)**

```python
import pyttsx3
import speech_recognition
import requests
import pyautogui
import random
import datetime
import os
import webbrowser
from pygame import mixer
from plyer import notification
from bs4 import BeautifulSoup
import speedtest

for i in range(3):
    a = input("Enter Password to open Jarvis :- ")
    pw_file = open("password.txt","r")
    pw = pw_file.read()
    pw_file.close()
    if (a==pw):
        print("WELCOME SIR ! PLZ SPEAK [WAKE UP] TO LOAD ME UP")
        break
    elif (i==2 and a!=pw):
        exit()

    elif (a!=pw):
        print("Try Again")

from INTRO import play_gif
play_gif

engine = pyttsx3.init("sapi5")
voices = engine.getProperty("voices")
engine.setProperty("voice", voices[0].id)
rate = engine.setProperty("rate",170)

def speak(audio):
    engine.say(audio)
    engine.runAndWait()

def takeCommand():
    r = speech_recognition.Recognizer()
    with speech_recognition.Microphone() as source:
        print("Listening....")
        r.pause_threshold = 1
        r.energy_threshold = 300
        audio = r.listen(source,0,4)

    try:
        print("Understanding..")
        query  = r.recognize_google(audio,language='en-in')
        print(f"You Said: {query}\n")
    except Exception as e:
        print("Say that again")
        return "None"
```

```python
def takeCommand():
        print("Say that again")
        return "None"
    return query
def alarm(query):
    timehere = open("Alarmtext.txt","a")
    timehere.write(query)
    timehere.close()
    os.startfile("alarm.py")

if __name__ == "__main__":
    while True:
        query = takeCommand().lower()
        if "wake up" in query:
            from GreetMe import greetMe
            greetMe()
            while True:
                query = takeCommand().lower()
                if "go to sleep" in query:
                    speak("Ok sir , You can me call anytime")
                    break
                elif "change password" in query:
                    speak("What's the new password")
                    new_pw = input("Enter the new password\n")
                    new_password = open("password.txt","w")
                    new_password.write(new_pw)
                    new_password.close()
                    speak("Done sir")
                    speak(f"Your new password is{new_pw}")
                elif "open" in query:  #EASY METHOD
                    query = query.replace("open","")
                    query = query.replace("jarvis","")
                    pyautogui.press("super")
                    pyautogui.typewrite(query)
                    pyautogui.sleep(2)
                    pyautogui.press("enter")
                elif "internet speed" in query:
                    wifi  = speedtest.Speedtest()
                    upload_net = wifi.upload()/1048576         #Megabyte = 1024*1024 Bytes
                    download_net = wifi.download()/1048576
                    print("Wifi Upload Speed is", upload_net)
                    print("Wifi download speed is ",download_net)
                    speak(f"Wifi download speed is {download_net}")
                    speak(f"Wifi Upload speed is {upload_net}")
                elif "screenshot" in query:
                    import pyautogui #pip install pyautogui
                    im = pyautogui.screenshot()
                    im.save("ss.jpg")
                elif "click my photo" in query:
                    pyautogui.press("super")
                    pyautogui.typewrite("camera")
                    pyautogui.press("enter")
                    pyautogui.sleep(2)
                    speak("SMILE")
```

```python
                    speak("SMILE")
                    pyautogui.press("enter")
                elif "translate" in query:
                    from Translator import translategl
                    query = query.replace("jarvis","")
                    query = query.replace("translate","")
                    translategl(query)
                elif "hello" in query:
                    speak("Hello sir, how are you ?")
                elif "i am fine" in query:
                    speak("that's great, sir")
                elif "how are you" in query:
                    speak("Perfect, sir")
                elif "thank you" in query:
                    speak("you are welcome, sir")
                elif "google" in query:
                    from SearchNow import searchGoogle
                    searchGoogle(query)
                elif "youtube" in query:
                    from SearchNow import searchYoutube
                    searchYoutube(query)
                elif "wikipedia" in query:
                    from SearchNow import searchWikipedia
                    searchWikipedia(query)
                elif "temperature" in query:
                    search = "temperature in delhi"
                    url = f"https://www.google.com/search?q={search}"
                    r  = requests.get(url)
                    data = BeautifulSoup(r.text,"html.parser")
                    temp = data.find("div", class_ = "BNeawe").text
                    speak(f"current{search} is {temp}")
                elif "the time" in query:
                    strTime = datetime.datetime.now().strftime("%H:%M")
                    speak(f"Sir, the time is {strTime}")
                elif "finally sleep" in query:
                    speak("Going to sleep,sir")
                    exit()
                elif "open" in query:
                    from Dictapp import openappweb
                    openappweb(query)
                elif "close" in query:
                    from Dictapp import closeappweb
                    closeappweb(query)
                elif "set an alarm" in query:
                    print("input time example:- 10 and 10 and 10")
                    speak("Set the time")
                    a = input("Please tell the time :- ")
                    alarm(a)
                    speak("Done,sir")
                elif "pause" in query:
                    pyautogui.press("k")
                    speak("video paused")
                elif "play" in query:
```

This code creates a voice-controlled assistant named "Jarvis." It authenticates the user with a password and loads when the correct password is entered. The assistant listens for wake-up commands and responds to various voice commands, including opening applications, searching the internet, setting alarms, taking screenshots, controlling media playback, providing weather updates, reading news, performing calculations, sending WhatsApp messages, and shutting down the system. It utilizes libraries like pyttsx3, speech_recognition, requests, pyautogui, datetime, os, webbrowser, BeautifulSoup, speedtest, and plyer for different functionalities.

- **Personal Assistant Working:**
- **Newsread.py**

```python
import requests
import json
import pyttsx3
engine = pyttsx3.init("sapi5")
voices = engine.getProperty("voices")
engine.setProperty("voice", voices[0].id)
rate = engine.setProperty("rate",170)
def speak(audio):
    engine.say(audio)
    engine.runAndWait()
def latestnews():
    api_dict = {"business" : "https://newsapi.org/v2/top-headlines?country=in&category=business&apiKey=2f2114612a3f4002b1ec820d0a8a2c2b",
            "entertainment" : "https://newsapi.org/v2/top-headlines?country=in&category=entertainment&apiKey=2f2114612a3f4002b1ec820d0a8a2c2b",
            "health" : "https://newsapi.org/v2/top-headlines?country=in&category=health&apiKey=2f2114612a3f4002b1ec820d0a8a2c2b",
            "science" :"https://newsapi.org/v2/top-headlines?country=in&category=science&apiKey=2f2114612a3f4002b1ec820d0a8a2c2b",
            "sports" :"https://newsapi.org/v2/top-headlines?country=in&category=sports&apiKey=2f2114612a3f4002b1ec820d0a8a2c2b",
            "technology" :"https://newsapi.org/v2/top-headlines?country=in&category=technology&apiKey=2f2114612a3f4002b1ec820d0a8a2c2b"
}
    content = None
    url = None
    speak("Which field news do you want, [business] , [health] , [technology], [sports] , [entertainment] , [science]")
    field = input("Type field news that you want: ")
    for key ,value in api_dict.items():
        if key.lower() in field.lower():
            url = value
            print(url)
            print("url was found")
            break
        else:
            url = True
    if url is True:
        print("url not found")
    news = requests.get(url).text
    news = json.loads(news)
    speak("Here is the first news.")
    arts = news["articles"]
    for articles in arts :
        article = articles["title"]
        print(article)
        speak(article)
        news_url = articles["url"]
        print(f"for more info visit: {news_url}")
        a = input("[press 1 to cont] and [press 2 to stop]")
        if str(a) == "1":
```

The code initializes text-to-speech and fetches news articles from different categories using the News API. It reads out the titles of the articles and prompts the user to continue or stop after each one. The user selects categories, and the script retrieves news accordingly.

➢ Output

```
Listening.....
Understanding..
You Said: latest news

Type field news that you want: sports
https://newsapi.org/v2/top-headlines?country=in&category=sports&apiKey=2f2114612a3f4002b1ec820d0a8a2c2b
url was found
[Removed]
for more info visit: https://removed.com
[press 1 to cont] and [press 2 to stop]1
Ilkay Gundogan sends message to Lamine Yamal after Barcelona teen hits winner - Barca Blaugranes
for more info visit: https://www.barcablaugranes.com/2024/3/9/24095264/ilkay-gundogan-sends-message-to-lamine-yamal-after-barcelona-teen-hits-winner
[press 1 to cont] and [press 2 to stop]1
Hope money is not the incentive to play Tests: Rahul Dravid - Hindustan Times
for more info visit: https://www.hindustantimes.com/cricket/hope-money-is-not-the-incentive-to-play-tests-rahul-dravid-reacts-after-bcci-announces-incentive-scheme-101710003037885.html
[press 1 to cont] and [press 2 to stop]1
GZZ vs BSCR Live Score, Group E | Gozo Zalmi vs BSC Rehberge Score & Updates of Group E - CricTracker
for more info visit: https://www.crictracker.com/live-scores/gzz-vs-bscr-group-e-t10-european-cricket-league-09-mar-2024/
[press 1 to cont] and [press 2 to stop]1
'Ageing' Shami needs to take cue from Anderson: McGrath - The Times of India
for more info visit: https://timesofindia.indiatimes.com/sports/cricket/news/ageing-shami-needs-to-take-cue-from-anderson-mcgrath/articleshow/108354687.cms
[press 1 to cont] and [press 2 to stop]2
```

- **Calculations.py**

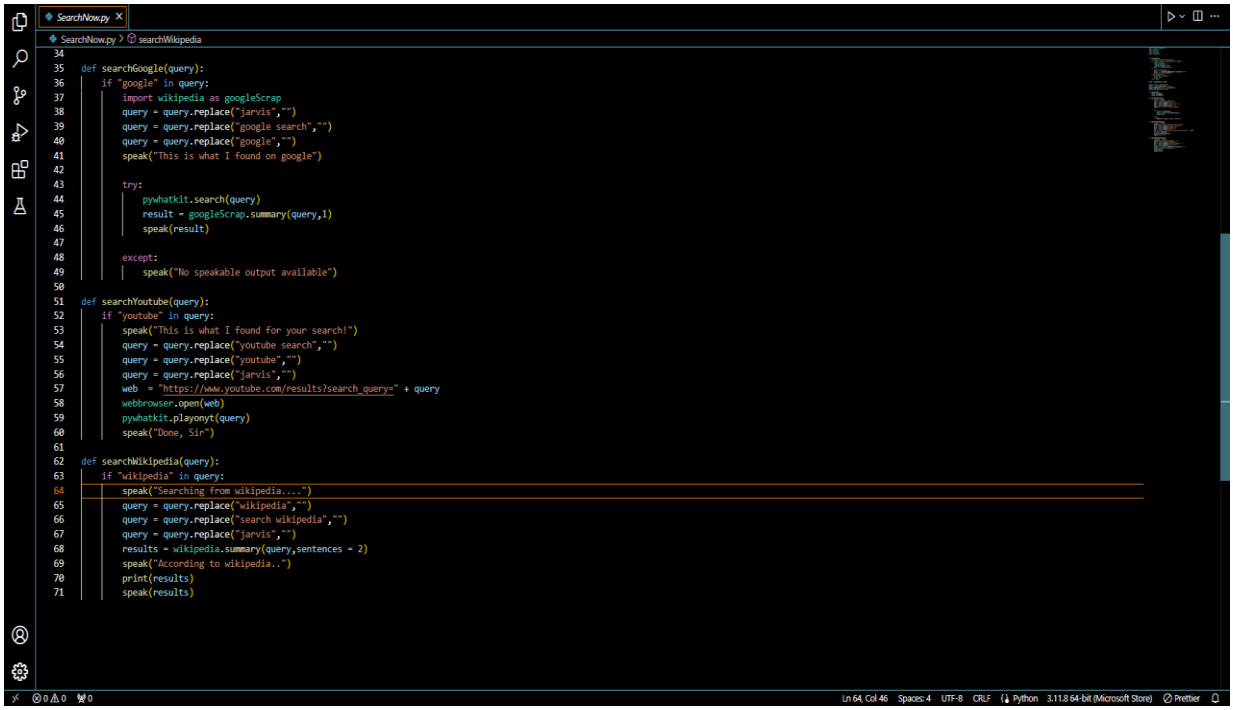
```python
import wolframalpha
import pyttsx3
import speech_recognition

engine = pyttsx3.init("sapi5")
voices = engine.getProperty("voices")
engine.setProperty("voice", voices[0].id)
rate = engine.setProperty("rate",170)

def speak(audio):
    engine.say(audio)
    engine.runAndWait()

def WolfRamAlpha(query):
    apikey = "U3HX5Q-VJJG6HHKJ2"
    requester = wolframalpha.Client(apikey)
    requested = requester.query(query)

    try:
        answer = next(requested.results).text
        return answer
    except:
        speak("The value is not answerable")

def Calc(query):
    Term = str(query)
    Term = Term.replace("jarvis","")
    Term = Term.replace("multiply","*")
    Term = Term.replace("plus","+")
    Term = Term.replace("minus","-")
    Term = Term.replace("divide","/")

    Final = str(Term)
    try:
        result = WolfRamAlpha(Final)
        print(f"{result}")
        speak(result)

    except:
        speak("The value is not answerable")
```

The code sets up voice interaction with Wolfram Alpha for questions and calculations. It converts speech to text, processes queries, and speaks back the answers.

➢ Output

```
Listening.....
Understanding..
You Said: calculate 55 + 35

90
Listening.....
Understanding..
You Said: calculate 55 X 22

1210
Listening.....
Understanding..
You Said: calculate 22 - 13

9
```

- **Searchnow.py**

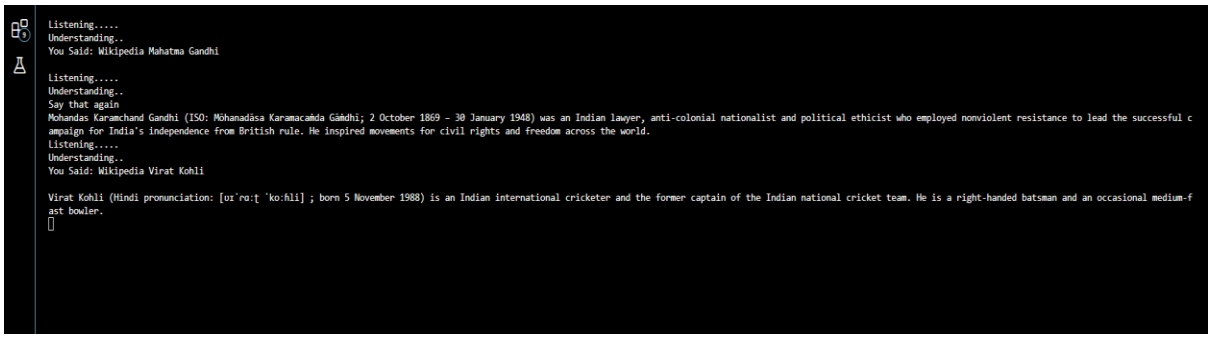


```python
def searchGoogle(query):
    if "google" in query:
        import wikipedia as googleScrap
        query = query.replace("jarvis","")
        query = query.replace("google search","")
        query = query.replace("google","")
        speak("This is what I found on google")

        try:
            pywhatkit.search(query)
            result = googleScrap.summary(query,1)
            speak(result)

        except:
            speak("No speakable output available")

def searchYoutube(query):
    if "youtube" in query:
        speak("This is what I found for your search!")
        query = query.replace("youtube search","")
        query = query.replace("youtube","")
        query = query.replace("jarvis","")
        web  = "https://www.youtube.com/results?search_query=" + query
        webbrowser.open(web)
        pywhatkit.playonyt(query)
        speak("Done, Sir")

def searchWikipedia(query):
    if "wikipedia" in query:
        speak("Searching from wikipedia....")
        query = query.replace("wikipedia","")
        query = query.replace("search wikipedia","")
        query = query.replace("jarvis","")
        results = wikipedia.summary(query,sentences = 2)
        speak("According to wikipedia..")
        print(results)
        speak(results)
```

This Python code enables a voice assistant named "Jarvis" to understand spoken commands, search Google, YouTube, and Wikipedia, and respond verbally using speech synthesis.

➢ Output

Example on Wikipedia:



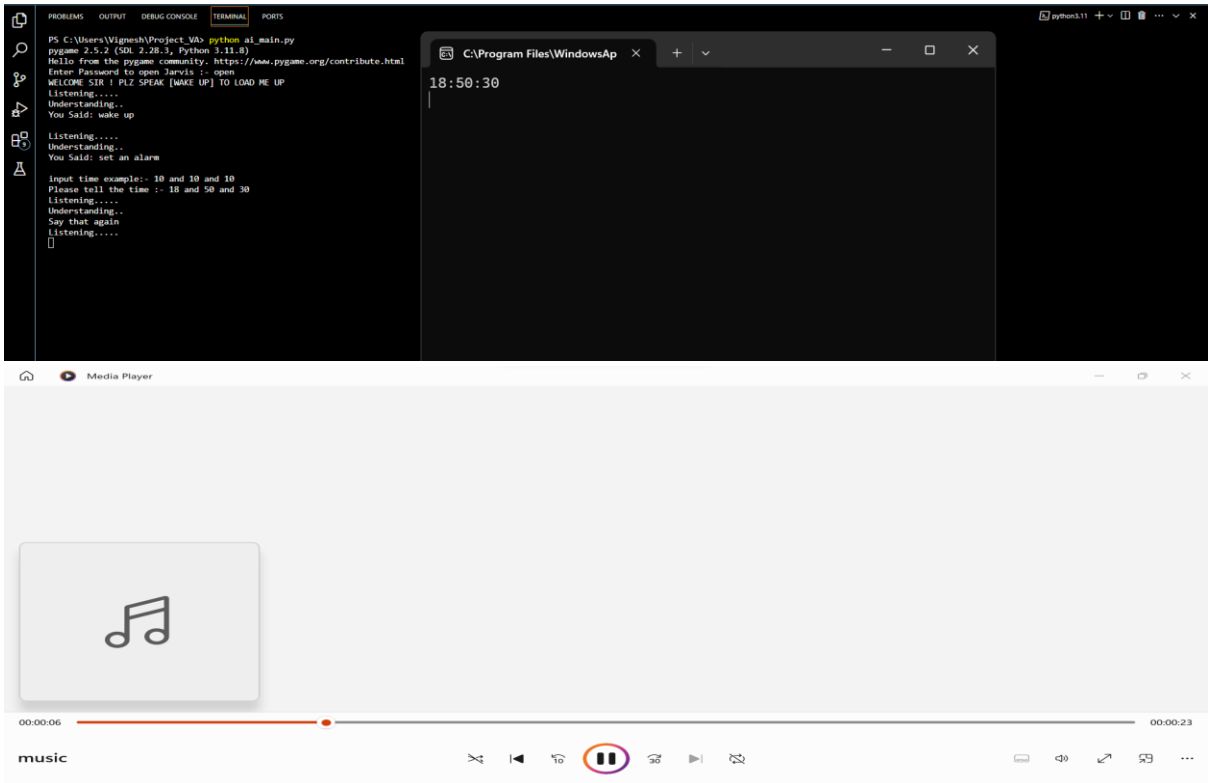
```
Listening.....
Understanding..
You Said: Wikipedia Mahatma Gandhi

Listening.....
Understanding..
Say that again
Mohandas Karamchand Gandhi (ISO: Mōhanadāsa Karamacaṁda Gāṁdhi; 2 October 1869 – 30 January 1948) was an Indian lawyer, anti-colonial nationalist and political ethicist who employed nonviolent resistance to lead the successful c
ampaign for India's independence from British rule. He inspired movements for civil rights and freedom across the world.
Listening.....
Understanding..
You Said: Wikipedia Virat Kohli

Virat Kohli (Hindi pronunciation: [ʊɪˈrɑːʈ ˈkoːɦlɪ] ; born 5 November 1988) is an Indian international cricketer and the former captain of the Indian national cricket team. He is a right-handed batsman and an occasional medium-f
ast bowler.
```

- **Alarm.py**

```python
import pyttsx3
import datetime
import os

engine = pyttsx3.init("sapi5")
voices = engine.getProperty("voices")
engine.setProperty("voice", voices[0].id)
engine.setProperty("rate",200)

def speak(audio):
    engine.say(audio)
    engine.runAndWait()

extractedtime = open("Alarmtext.txt","rt")
time = extractedtime.read()
Time = str(time)
extractedtime.close()

deletetime = open("Alarmtext.txt","r+")
deletetime.truncate(0)
deletetime.close()

def ring(time):
    timeset = str(time)
    timenow = timeset.replace("jarvis","")
    timenow = timenow.replace("set an alarm","")
    timenow = timenow.replace(" and ",":")
    Alarmtime = str(timenow)
    print(Alarmtime)
    while True:
        currenttime = datetime.datetime.now().strftime("%H:%M:%S")
        if currenttime == Alarmtime:
            speak("Alarm ringing,sir")
            os.startfile("music.mp3") #You can choose any music or ringtone
        elif currenttime + "00:00:30" == Alarmtime:
            exit()

ring(time)
```

The code reads a time value from a file, sets an alarm for that time, and plays a specified sound file when the alarm time matches the current time. It uses a text-to-speech engine to announce the alarm ringing. After the alarm rings, the program exits.

➢ Output

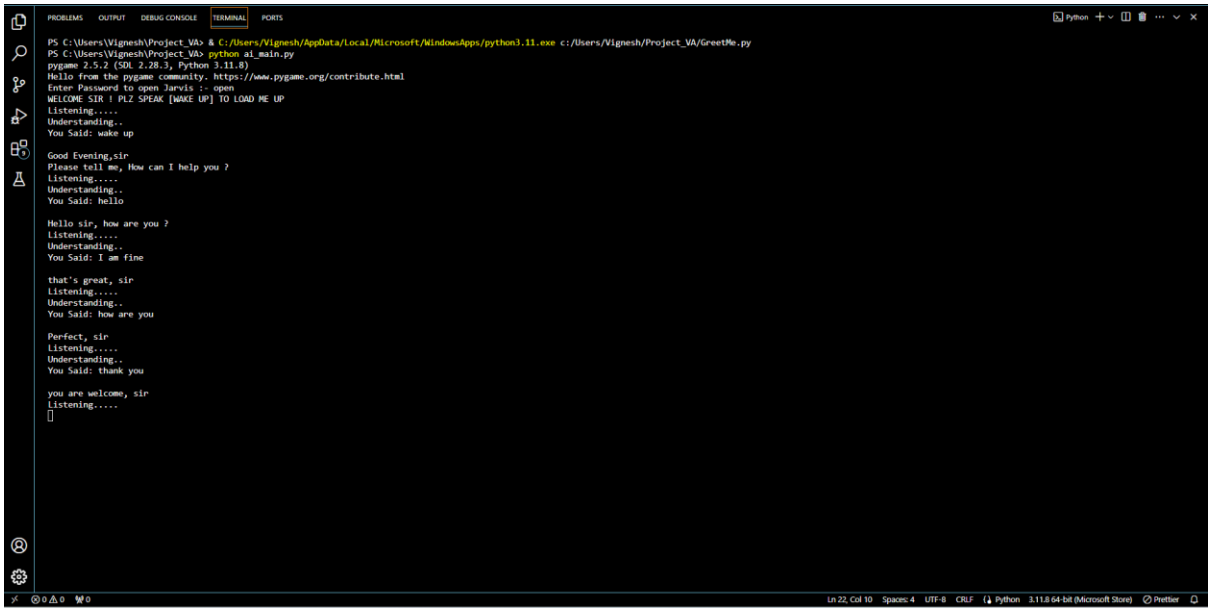


- **GreetMe.py**

```python
import pyttsx3
import datetime

engine = pyttsx3.init("sapi5")
voices = engine.getProperty("voices")
engine.setProperty("voice", voices[0].id)
engine.setProperty("rate",200)

def speak(audio):
    engine.say(audio)
    engine.runAndWait()

def greetMe():
    hour  = int(datetime.datetime.now().hour)
    if hour>=0 and hour<=12:
        speak("Good Morning,sir")
    elif hour >12 and hour<=18:
        speak("Good Afternoon ,sir")

    else:
        speak("Good Evening,sir")

    speak("Please tell me, How can I help you ?")
```

The code greets the user with a message based on the time of day and prompts them for assistance. It uses a text-to-speech engine to speak the greetings and instructions.

➢ Output



```
PS C:\Users\Vignesh\Project_VA> & C:/Users/Vignesh/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/Vignesh/Project_VA/GreetMe.py
PS C:\Users\Vignesh\Project_VA> python ai_main.py
pygame 2.5.2 (SDL 2.28.3, Python 3.11.8)
Hello from the pygame community. https://www.pygame.org/contribute.html
Enter Password to open Jarvis :- open
WELCOME SIR ! PLZ SPEAK [WAKE UP] TO LOAD ME UP
Listening.....
Understanding..
You Said: wake up

Good Evening,sir
Please tell me, How can I help you ?
Listening.....
Understanding..
You Said: hello

Hello sir, how are you ?
Listening.....
Understanding..
You Said: I am fine

that's great, sir
Listening.....
Understanding..
You Said: how are you

Perfect, sir
Listening.....
Understanding..
You Said: thank you

you are welcome, sir
Listening.....
```

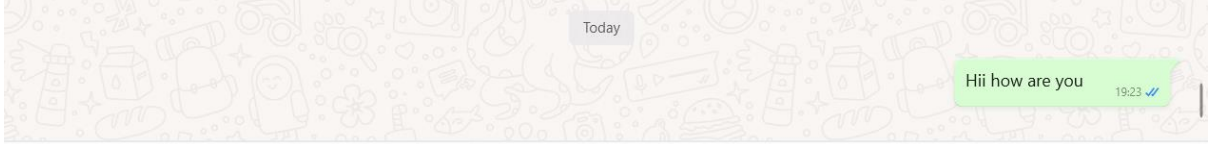
- **Whatsapp.py**

```python
import pywhatkit
import pyttsx3
import datetime
import speech_recognition
import os
from datetime import timedelta
from datetime import datetime

engine = pyttsx3.init("sapi5")
voices = engine.getProperty("voices")
engine.setProperty("voice", voices[0].id)
rate = engine.setProperty("rate",170)
def speak(audio):
    engine.say(audio)
    engine.runAndWait()
def takeCommand():
    r = speech_recognition.Recognizer()
    with speech_recognition.Microphone() as source:
        print("Listening.....")
        r.pause_threshold = 1
        r.energy_threshold = 300
        audio = r.listen(source,0,4)

    try:
        print("Understanding..")
        query  = r.recognize_google(audio,language='en-in')
        print(f"You Said: {query}\n")
    except Exception as e:
        print("Say that again")
        return "None"
    return query
strTime = int(datetime.now().strftime("%M"))
update = int((datetime.now()+timedelta(minutes = 2)).strftime("%M"))

def sendMessage():
    speak("Who do you wan to message")
    a = int(input('''Person 1 - 1
    Person 2 - 2'''))
    if a == 1:
        speak("Whats the message")
        message = str(input("Enter the message- "))
        pywhatkit.sendwhatmsg("+919502740178",message,time_hour-strTime,time_min-update)
    elif a==2:
        pass
```

The code utilizes voice recognition to take user commands and sends WhatsApp messages using PyWhatKit. It schedules messages for the current hour plus 2 minutes ahead, allowing users to specify recipients and message content through voice input.

➢ Output

```
Good Evening,sir
Please tell me, How can I help you ?
Listening.....
Understanding..
You Said: WhatsApp

Person 1 - 1
    Person 2 - 2 1
Enter the message- Hiii how are you
In 30 Seconds WhatsApp will open and after 15 Seconds Message will be Delivered!
```

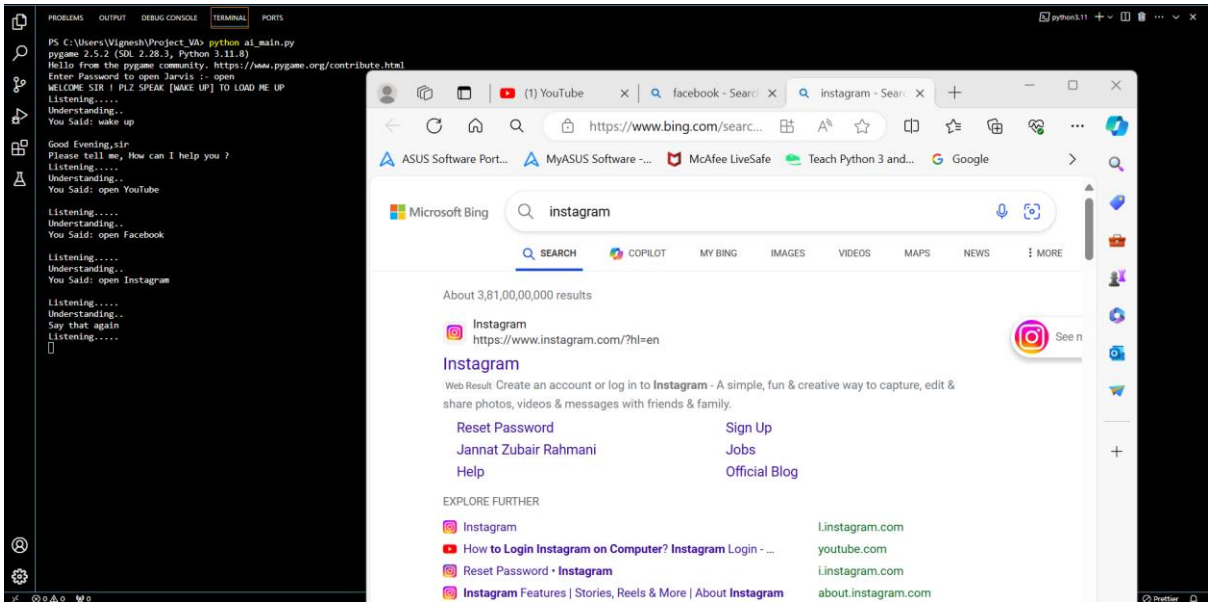


- **Dictapp.py**

```
import os
import pyautogui
import webbrowser
import pyttsx3
from time import sleep

engine = pyttsx3.init("sapi5")
voices = engine.getProperty("voices")
engine.setProperty("voice", voices[0].id)
engine.setProperty("rate",200)

def speak(audio):
    engine.say(audio)
    engine.runAndWait()

dictapp = {"commandprompt":"cmd","paint":"paint","word":"winword","excel":"excel","chrome":"chrome","vscode":"code","powerpoint":"powerpnt"}

def openappweb(query):
    speak("Launching, sir")
    if ".com" in query or ".co.in" in query or ".org" in query:
        query = query.replace("open","")
        query = query.replace("jarvis","")
        query = query.replace("launch","")
        query = query.replace(" ","")
        webbrowser.open(f"https://www.{query}")
    else:
        keys = list(dictapp.keys())
        for app in keys:
            if app in query:
                os.system(f"start {dictapp[app]}")

def closeappweb(query):
    speak("Closing,sir")
    if "one tab" in query or "1 tab" in query:
        pyautogui.hotkey("ctrl","w")
        speak("All tabs closed")
    elif "2 tab" in query:
        pyautogui.hotkey("ctrl","w")
        sleep(0.5)
        pyautogui.hotkey("ctrl","w")
        speak("All tabs closed")
    elif "3 tab" in query: ...

    elif "4 tab" in query: ...
    elif "5 tab" in query: ...

    else:
        keys = list(dictapp.keys())
        for app in keys:
            if app in query:
                os.system(f"taskkill /f /im {dictapp[app]}.exe")
```

The code enables voice commands to open and close applications and websites, using a dictionary to map application names to their commands. It launches and closes applications based on user input and closes browser tabs by simulating keyboard shortcuts.

➢ Output



12. Conclusion

The development of the desktop assistant project represents a significant advancement in the realm of human-computer interaction, leveraging cutting-edge technologies to create a sophisticated and user-centric interface. Through robust voice command recognition, natural language understanding, and seamless integration with external services, the assistant has demonstrated its ability to streamline user interactions and enhance productivity in computing environments.

12.1 Achievements

1. Functional Excellence: The desktop assistant showcases exceptional performance in accurately recognizing voice commands and executing a wide array of tasks, ranging from simple file operations to complex web searches and system manipulations. Its intuitive interface and responsive feedback mechanisms contribute to a seamless user experience.

2. Intelligent Interaction: By harnessing the power of natural language processing, the assistant exhibits a remarkable level of comprehension and context awareness, enabling it to engage in meaningful conversations with users. Its ability to understand user intents and adapt responses accordingly enhances the user experience and fosters a sense of natural interaction.

3. Enhanced Functionality: Through integration with external APIs and services, such as weather forecasts, news updates, and multimedia playback, the assistant expands its capabilities and delivers real-time information tailored to user preferences. This breadth of functionality transforms the assistant into a versatile and indispensable tool for everyday use.

4. Robust Error Handling: The project incorporates robust error handling mechanisms, ensuring graceful recovery from unexpected scenarios and providing informative feedback to users in case of unrecognized commands or system failures.

This resilience enhances user confidence and fosters trust in the reliability of the assistant.

12.2 Implications and Impact

1. User Empowerment: The desktop assistant empowers users to accomplish tasks more efficiently and effectively, reducing cognitive load and freeing up time for higher-level cognitive activities. Its intuitive interface and hands-free operation cater to users of all skill levels, democratizing access to technology and promoting digital inclusivity.

2. Productivity Enhancement: By automating routine tasks and streamlining workflow processes, the assistant significantly enhances user productivity and workflow efficiency. Its ability to handle multiple concurrent tasks and provide timely information boosts productivity across diverse domains and use cases.

3. Technological Advancement: The project underscores the transformative potential of voice-controlled interfaces in shaping the future of human-computer interaction. As technology continues to evolve, there is immense potential for further innovation and refinement in voice recognition, natural language understanding, and contextual computing.

4. Ethical Considerations: As with any technology-driven endeavor, the project acknowledges the importance of ethical considerations, including privacy, security, and data governance. By adhering to best practices and ethical guidelines, the project promotes responsible development and deployment of intelligent systems.

12.3 Future Directions

Moving forward, the desktop assistant project opens up exciting opportunities for future development and exploration. Key areas for future enhancement include:

1. Advanced AI Capabilities: Further integration of artificial intelligence and machine learning algorithms can enhance the assistant's ability to learn from user interactions, personalize responses, and anticipate user needs proactively.

2. Multi-Modal Interfaces: Exploration of multi-modal interfaces, combining voice commands with gesture recognition, touch inputs, and visual feedback, can enrich the user experience and cater to diverse user preferences and accessibility needs.

3. Domain-Specific Applications: Customization of the assistant for specific domains, such as healthcare, education, and entertainment, can unlock new use cases and drive innovation in targeted industries.

4. User-Centric Design Iterations: Continuous user feedback and iterative design cycles are essential for refining the assistant's user interface, improving usability, and addressing user pain points effectively.

13. References

- https://towardsdatascience.com/how-to-build-your-own-ai-personal-assistant-using-pythonf57247b4494b

- https://www.analyticsvidhya.com/blog/2020/11/build-your-own-desktop-voice-assistant-inpython/

14. Plagiarism Report

Sem2project

ORIGINALITY REPORT

4%	3%	2%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	digital.lib.usu.edu Internet Source	1%
2	S.A. Nayana Thilanka, G.C.H. Vihanga Dimantha Jinadasa, Prasad M. Bandara, W.M.M. Tharindu Weerakoon. "Vision-Based Real-Time Object Detection and Voice Alert for Blind Assistance System", 2023 IEEE 17th International Conference on Industrial and Information Systems (ICIIS), 2023 Publication	<1%
3	"Soft Computing for Security Applications", Springer Science and Business Media LLC, 2023 Publication	<1%
4	finance.dailyherald.com Internet Source	<1%
5	Wei Emma Zhang, Quan Z. Sheng, Ahoud Alhazmi, Chenliang Li. "Adversarial Attacks on Deep-learning Models in Natural Language Processing", ACM Transactions on Intelligent Systems and Technology, 2020	<1%

	Publication	
6	medium.com Internet Source	<1 %
7	messiahx2h7l.post-blogs.com Internet Source	<1 %
8	www.ijraset.com Internet Source	<1 %
9	umlbasic.blogspot.com Internet Source	<1 %
10	www.uni-mannheim.de Internet Source	<1 %
11	github.com Internet Source	<1 %
12	Abdelrahman Atef, Fahd Seddik, Abdulrahman Elbedewy. "AGS: Arabic GPT Summarization Corpus", 2023 International Conference on Electrical, Communication and Computer Engineering (ICECCE), 2023 Publication	<1 %
13	pypi.org Internet Source	<1 %
14	wntsapp.eu Internet Source	<1 %
15	Gerhard Paaß, Sven Giesselbach. "Foundation Models for Natural Language Processing",	<1 %

Springer Science and Business Media LLC, 2023
Publication

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		