



TO BUILD A VOICE ASSISTANT WITH OPEN AI USING PYTHON

Dr.S.BHUVANESHWARI M.COM.,M.Phil,PGDCA;MBA,Ph.D Assistant Professor, Department of B.Com.CA, Sri Krishna Adithya College of Arts and Science, Coimbatore.

BALA KRISHNAN.S Department of B.Com.CA, Sri Krishna Adithya College of Arts and Science, Coimbatore.

1.1 INTRODUCTION:

This presents the design and implementation of a voice assistant system using speech recognition techniques in the python programming language. Voice assistants have gained widespread popularity and are becoming an integral part of our daily lives, providing us with hands-free and convenient ways to interact with computers and smart devices.

The proposed system utilizes the power of automatic speech recognition (asr) to convert spoken language into text, enabling seamless interaction between users and the assistant. The key objective is to develop a robust and accurate speech recognition module that can understand natural language commands and respond appropriately.

The system incorporates various components, including audio input processing, feature extraction, speech recognition models, natural language understanding, and text-to-speech synthesis. Python libraries such as speechrecognition, pyaudio, and nltk are leveraged to facilitate these functionalities.



Overall, this project showcases the potential of using speech recognition techniques in python to create a voice assistant that can understand and respond to natural language commands, contributing to the advancement of human-computer interaction and intelligent personal assistants.

1.1.1 INTRODUCTION ON VISUAL STUDIO:

Visual Studio 2022, developed by Microsoft, is a sophisticated integrated development environment (IDE) tailored for software development. With a sleek and user-friendly interface, it offers developers a comfortable workspace for coding, debugging, and project management. One of its standout features is its broad platform support, accommodating development for Windows, Android, iOS, web, and cloud platforms. The IDE comes equipped with a wide array of tools and templates for various programming languages and frameworks, such as .NET, C++, Python, and JavaScript. Notably, Visual Studio 2022 is optimized for performance, enabling developers to work on extensive projects efficiently. Its collaboration tools, including Live Share, facilitate real-time co-authoring and pair programming, fostering seamless teamwork regardless of team members' locations.

Visual studio 2022, developed by Microsoft , is a moreover, the ide harnesses the power of artificial intelligence (AI) to provide intelligent code suggestions, refactoring tools, and code analysis capabilities, thereby enhancing productivity. With built-in integration for azure services, visual studio 2022 simplifies the development, deployment, and management of cloud-native applications. Furthermore, the ide's extensibility allows developers to customize their workspace with additional tools and features through a rich ecosystem of extensions. Microsoft's commitment to accessibility ensures that visual studio 2022 is inclusive, with features and tools designed to improve accessibility and usability for all developers. Whether you're a seasoned developer or a novice embarking on your coding journey, visual studio 2022 offers a comprehensive and versatile environment to create high-quality applications efficiently. In addition to its robust feature set, visual studio 2022 prioritizes user experience and workflow efficiency. Its intuitive interface and



customizable layout options cater to diverse developer preferences, promoting a seamless and personalized coding experience. Furthermore, the ide's integration with Microsoft's ecosystem, including azure devops and github, facilitates seamless project management and version control. With regular updates and improvements driven by community feedback, visual studio 2022 remains at the forefront of modern software development, empowering developers to turn their ideas into reality with speed, precision, and innovation.

1.2 SYSTEM SPECIFICATION

1.2.1 SOFTWARE REQUIREMENTS:

- **Operating System:** Windows 7, 10, 11, Linux
- **Programming Language:** Python
- **IDE/Workbench:** Visual Studio code.

1.2.2 HARDWARE REQUIREMENTS:

- **Processor:** Any processor
- **Hard Disk:** 128GB or more
- **RAM:** 8GB or more



2.1 EXISTING AND PROPOSED SYSTEM

2.1.1 EXISTING SYSTEM:

We are familiar with many existing voice assistants like Alexa, Siri, Google Assistant, and Cortana, which use concepts of language processing and voice recognition. They listen to the commands given by the user and, as per their requirements, perform that specific function in a very efficient and effective manner. As these voice assistants use artificial intelligence, the results that they are providing are highly accurate and efficient. These assistants can help reduce human effort and consume less time while performing any task; they have removed the concept of typing completely and behave as another individual to whom we are talking and asking to perform tasks. These assistants are no less than human assistants, but we can say that they are more effective and efficient at performing any task. The algorithm used to make these assistants focus on the time complexities and reduce time, but for using these assistants, one should have an account (like a Google account for Google Assistant or a Microsoft account for Cortana) and can use it with an internet connection only because these assistants are going to work with internet connectivity. They are integrated with many devices like phones, laptops, speakers, etc.

Let's break down how to create a basic ai voice assistant using python, leveraging existing libraries and services. This explanation focuses on the core components and their interaction. A full, production-ready assistant is a complex project, but this will give you a solid foundation.

1. Speech Recognition (listening):

Library:



- ❖ speech_recognition (wraps various speech recognition APIs) **Process:**
- ❖ The user speaks into the microphone.
- ❖ The speech_recognition library captures the audio.
- ❖ It sends the audio to a speech recognition engine (e.g., google speech recognition, cmu sphinx). You'll likely need an api key for cloud-based engines.
- ❖ The engine transcribes the audio into text.

2. Natural language processing (understanding):

Libraries:

- ❖ NLTK, spacy (for more advanced NLP), potentially transformers (for using pretrained models).

Process:

- ❖ The transcribed text is analyzed.
- ❖ Intent recognition: determine what the user wants to do (e.g., "set an alarm," "play music," "tell me the weather"). This can be done with keyword matching, rule-based systems, or machine learning models.
- ❖ Entity extraction: identify key information in the text (e.g., the time for the alarm, the song to play, the location for the weather).

3. Action execution (doing):

Process:

- ❖ Based on the identified intent and entities, perform the appropriate action.
- ❖ **This might involve:**
 - Calling other python functions.
 - Interacting with external apis (e.g., a weather api, a music streaming service api).
 - Controlling the operating system (e.g., setting an alarm).

4. Speech synthesis (speaking):

Libraries:

- ❖ pyttsx3 (cross-platform), other platform-specific libraries.



Process:

- ❖ Convert the text you want the assistant to say into speech.
- ❖ Play the audio through the speakers.

5. Putting it together (simplified flow):

Listen:

- ❖ Use `speech_recognition` to capture and transcribe user speech.

Understand:

- ❖ Use NLP techniques (or simpler keyword matching) to determine the user's intent and extract entities.

Act:

- ❖ Execute the appropriate action based on the intent and entities. This might involve calling other functions or interacting with APIs.

Speak:

- ❖ Use `pyttsx3` to generate speech and respond to the user.

Key improvements and considerations:

- ❖ **More robust NLP:** use `spacy` or transformer models for better intent recognition and entity extraction.
- ❖ **Context handling:** maintain conversation history to understand follow-up questions.
- ❖ **Error handling:** implement robust error handling for speech recognition, API calls, etc.
- ❖ **Background processing:** use threading or asynchronous programming to prevent the assistant from blocking while waiting for speech input or API responses.
- ❖ **User interface (optional):** create a simple GUI or command-line interface for interaction.



- ❖ **Wake word:** implement a "wake word" so the assistant only listens when triggered (e.g., "hey").
- ❖ **Integration with services:** connect to various APIs (weather, music, smart home devices) to expand functionality.

Drawbacks of Existing System

Here are the key drawbacks of a basic voice assistant built with OpenAI:

Latency:

- There's a noticeable delay between speaking, processing, and the AI's response. This is due to the network round trip to the OpenAI API, processing time, and text-to-speech.

Internet Dependence:

- The system relies entirely on an internet connection. If the internet is down, the assistant won't function.

API Costs:

- Using the OpenAI API incurs costs based on usage. Frequent or lengthy interactions can become expensive.

Privacy Concerns:

- Audio data is sent to external servers (Google for speech recognition and OpenAI for processing). Some users may be uncomfortable with this.
- The openAI terms of service should be read, as your data might be used to further train the AI.



Accuracy Limitations:

- Speech recognition can be inaccurate, especially in noisy environments or with accents.
- OpenAI's responses may not always be accurate or relevant, particularly for complex or nuanced queries.

Lack of Personalization:

- A basic system lacks the ability to learn user preferences or personalize responses over time.
- It also lacks local context, like current time, or local files.

Limited Functionality:

- The system's capabilities are limited to what the OpenAI API can provide. It may not be able to perform actions like controlling smart home devices or accessing local files without additional integrations.

Security Risks:

- If the system is not properly secured, there's a risk of unauthorized access or data breaches.

Wake Word Implementation:

- The basic code provided does not contain a wake word. Meaning that the program is constantly listening. A good voice assistant should only listen when a specific word or phrase is spoken. Implementing a reliable and efficient wake word system can be complex.

Error Handling:



- The provided code has basic error handling, but a production-ready system would need robust error handling for network issues, API errors, and other potential problems.

2.1.2 PROPOSED SYSTEM:

Proposed system for developing a voice assistant using speech recognition techniques in the python programming language. The objective is to design an efficient and user-friendly voice assistant that can understand natural language commands and provide accurate and meaningful responses.

Leverage speech recognition techniques in python to create voice assistant applications. Voice assistants have gained significant traction in recent years, offering users a convenient and intuitive way to interact with technology using natural language commands.

The study examines several prominent voice assistant systems developed using python's speech recognition capabilities. These systems demonstrate the versatility and potential of python in creating robust and efficient voice assistants.

Through a comparative analysis of these systems, this paper aims to highlight the strengths, limitations, and unique features of each, enabling developers and researchers to make informed decisions when choosing a speech recognition framework for their voice assistant projects.

Overall, the existing systems reviewed in this paper demonstrate the continuous development and innovation in the field of voice assistants using speech recognition in python. They provide a solid foundation for further advancements in creating intelligent and user-friendly voice interfaces.

The features of this project are:

1. It can send emails.
2. It can read PDF.
3. It can open and search a youtube.



4. It can open and close applications.
5. It can play music.
6. It can do Wikipedia searches for you.
7. It can open any websites.
8. It can give weather forecast.

2.1.3 ABOUT SOFTWARE:

The IDE used for this project is Visual Studio Code. All Python files are written with Visual Studio code, and all required packages can be easily installed in this IDE. Modules and libraries such as pyttsx3, SpeechRecognition, Wikipedia, keyboard, pywhatkit, pyjokes, PyPDF2, pyautogui, and PyQt are used in this project.

A live GUI is created for interacting with the virtual assistant, as it gives the conversation a unique and interesting look.

A. Visual Studio Code

It is an IDE, i.e., Integrated Development Environment, which has many features like supporting scientific tools (like Matplotlib, NumPy, and SciPy), web frameworks (example: Django, web2py, and Flask), refactoring in Python, an integrated Python debugger, code completion, code and project navigation, etc.

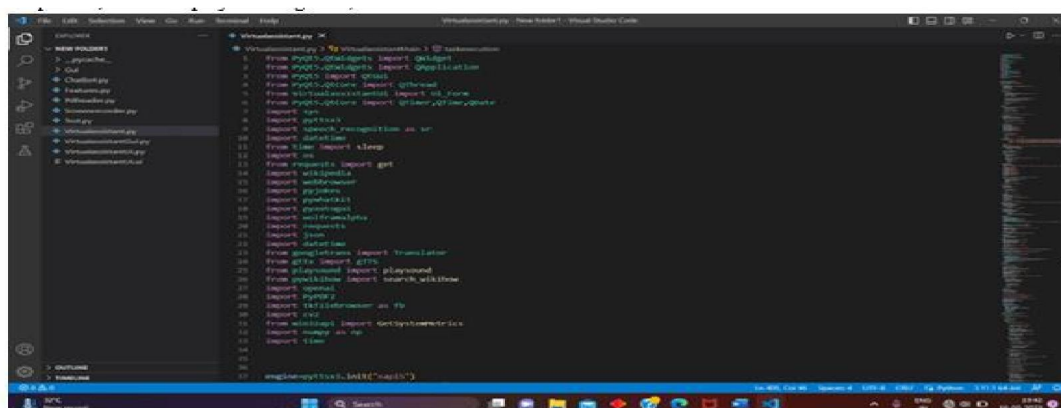


Fig. 5.1 Visual Studio Code IDE

E. PYQT5 for Graphical User Interface

PyQt5 is based on Qt version 5 that covers graphical user interfaces, network communication, threads, multimedia, XML processing, regular expressions, SQL databases, multimedia, web browsing, and other technologies available in Qt. The PyQt5 library is a high-level Python package that contains several Qt classes in a set of Python modules. PyQt5 is the most important Python plugin that contains a set of GUI widgets. PyQt5 has several important Python modules, such as QtWidgets, QtCore, QtGui, and QtDesigner.

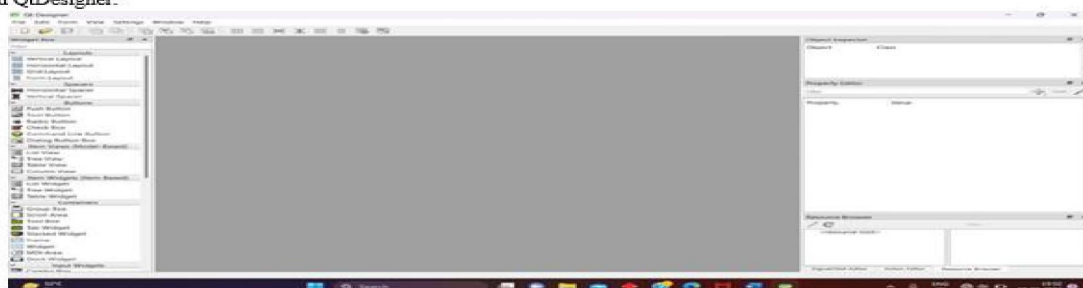


Fig. 5.2 Qt Designer

1. Speech Recognition:

Purpose: Converts spoken audio into text.

Libraries:

- ❖ **Speech_recognition:** A wrapper for several speech recognition apis (like google speech recognition, cmu sphinx). It's good for quick prototyping and testing different engines.
- ❖ **Vosk:** An offline, lightweight, high-accuracy speech recognition toolkit. It's particularly useful when internet connectivity is unreliable or unavailable.
- ❖ **Whisper:** An open-source, automatic speech recognition (asr) system developed by openai. It is known for its robustness and accuracy.

How it works: These libraries typically send the audio data to a speech recognition engine (either online or offline). The engine processes the audio and returns the transcribed text.

2. Natural Language Processing (NLP):

Purpose: Understands the meaning and intent behind the transcribed text.

Libraries:

- ❖ **NLTK (Natural Language Toolkit):** Provides tools for tokenization, stemming, tagging, parsing, and more. It's a foundational library for NLP.
- ❖ **Spacy:** A more advanced library that focuses on efficiency and provides pretrained models for various NLP tasks (named entity recognition, part-of-speech tagging, etc.).
- ❖ **Transformers (Hugging Face):** Offers access to powerful pre-trained transformer models (like BERT, GPT) that can be fine-tuned for specific NLP tasks (intent classification, question answering, etc.).

How it works: NLP techniques are used to:

- ❖ **Intent Recognition:** Figure out what the user wants (e.g., "Set an alarm for 7 AM" - the intent is to set an alarm).
- ❖ **Entity Extraction:** Identify key information in the text (e.g., "7 AM" is the time entity).

3. Dialogue Management:

Purpose: Manages the flow of the conversation and keeps track of the context.

Libraries/Frameworks:

- ❖ **Rasa:** An open-source framework specifically designed for building conversational AI assistants. It handles intent recognition, entity extraction, dialogue management, and more.
- ❖ **Other approaches:** You can also build simpler dialogue management systems using Python dictionaries, conditional statements, and state machines.

How it works: The dialogue manager decides what action to take based on the user's input and the current state of the conversation. It might trigger a specific function, ask a clarifying question, or provide a response.

4. Task execution:

Purpose: Performs the actions requested by the user.

Libraries/modules:

This is highly dependent on the functionality of your voice assistant.

- ❖ **Date time:** For working with dates and times (for setting alarms, reminders).
- ❖ **Requests:** For making api calls (e.g., getting weather information).
- ❖ **Os or subprocess:** For interacting with the operating system (e.g., opening applications).
- ❖ **Custom modules:** You'll likely write your own functions to handle specific tasks.



How it works: Based on the intent and entities extracted by the NLP module, the task execution component calls the appropriate functions to perform the required action.

5. Text-to-Speech (TTS):

Purpose: Converts text into spoken audio.

Libraries:

- ❖ **pyttsx3:** A cross-platform text-to-speech library.
- ❖ **GTTS (Google Text-to-Speech):** Uses Google's TTS engine (requires internet connection).
- ❖ **Coqui TTS:** An advanced, open-source TTS library.

How it works: The TTS library takes the text response generated by the dialogue manager and converts it into audio, which is then played through the speakers.

Key Improvements in Modern AI Assistants:

- ❖ **Deep Learning:** Modern assistants heavily rely on deep learning models for speech recognition, NLP, and dialogue management. This has significantly improved accuracy and performance.
- ❖ **Contextual Awareness:** Advanced assistants can maintain context across multiple turns of conversation, making the interaction more natural.
- ❖ **Personalization:** They can learn user preferences and tailor their responses accordingly.



2.2 FEATURES:

Python has few keywords, simple structure, and a clearly defined syntax. Python Code is more clearly defined and visible to the eyes. Python's source code is fairly easy-to-Maintaining. Python's bulk of the library is very portable and cross-platform compatible on Unix, windows, and macintosh. Python has support for an interactive mode which allows Interactive testing and debugging of snippets of code. Portable python can run on a wide variety of hardware platforms and has the same interface on all platforms.

Key features of an AI voice assistant include: natural language processing (NLP) to understand human language, speech recognition to convert voice into text, context awareness to interpret meaning based on the situation, task automation to perform actions based on commands, personalization and customization options, integrations with other apps and devices, and the ability to learn and adapt to user behavior over time; essentially allowing users to interact with their devices hands-free using natural language to perform various tasks.

Breakdown of features:

1. Task Automation and Efficiency

AI virtual assistants can automate repetitive tasks, such as in call centers. AI-powered assistants can handle routine tasks such as answering common customer questions, scheduling calls, and managing inquiries, all without human intervention, which reduces errors and improves overall efficiency.

With an AI virtual helper, businesses can manage large volumes of tasks simultaneously, ensuring smooth workflows and higher productivity. Whether automating data entry,



appointment scheduling, or follow-up emails, AI-powered virtual assistants excel at optimizing time-consuming tasks.

2. Personalization and Adaptability

The ability of AI virtual assistants to personalize user experiences is a game-changer for both businesses and customers. These systems adapt to individual user preferences, learning from past interactions to provide more relevant and customized responses.

For example, an AI virtual assistant can tailor responses based on a customer's previous interactions in customer service, offering more meaningful engagement.

Virtual assistant AI can also adjust its tone and language, adapting to the user's communication style or geographic region. This level of personalization fosters better customer relationships, improving satisfaction and loyalty.

3. 24/7 Availability

Unlike human employees, AI virtual assistants can operate around the clock, providing 24/7 availability. This is especially crucial in customer service and support industries, where timely responses are key to customer satisfaction.

4. Multilingual Support

As businesses expand globally, communication across languages becomes critical. AI-powered virtual assistants are equipped with multilingual capabilities, allowing them to seamlessly handle customer interactions in multiple languages.

This feature especially benefits businesses operating across different regions, ensuring that language barriers do not impede customer service.

Virtual voice assistants can instantly switch between languages, facilitating better



communication and ensuring customers receive assistance in their preferred language, making interactions more inclusive and accessible.

5. Seamless Integration

Another critical feature of AI virtual assistants is their ability to integrate with existing business tools and workflows. Whether it's connecting with customer relationship management (CRM) platforms, email systems, or scheduling tools, AI virtual assistants can be embedded into daily operations without causing disruption.

This seamless integration ensures that AI powered virtual assistant not only automate tasks but also collaborate effectively with other tools to enhance business efficiency.

3.1 SYSTEM DESIGN:

The degree of interest in each concept has varied over the year, each has stood the test of time. Each provides the software designer with a foundation from which more sophisticated design methods can be applied. Fundamental design concepts provide the necessary framework for “getting it right”.

During the design process the software requirements model is transformed into design Models that describe the details of the data structures, system architecture,

interface, and Components. Each design product is reviewed for quality before moving to the next phase of Software development.

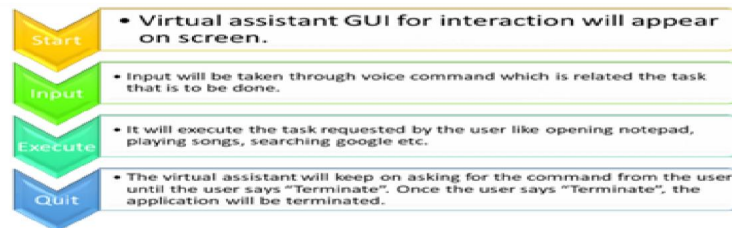


Fig. 4.1 Dataflow of Virtual Assistant

The system is designed using the concept of artificial intelligence and with the help of the necessary Python packages. Python provides numerous libraries and packages to perform the tasks; for illustration, pyPDF2 can be used to read PDF. The details of these packages are mentioned in Chapter VI of this research report. The data in this design is nothing but the end user's input; whatever the end user says, the virtual assistant performs the task accordingly. The end user's input is nothing specific but a list of tasks that the end user wants to get performed in a mortal language, i.e., English.

C. Activity Diagram

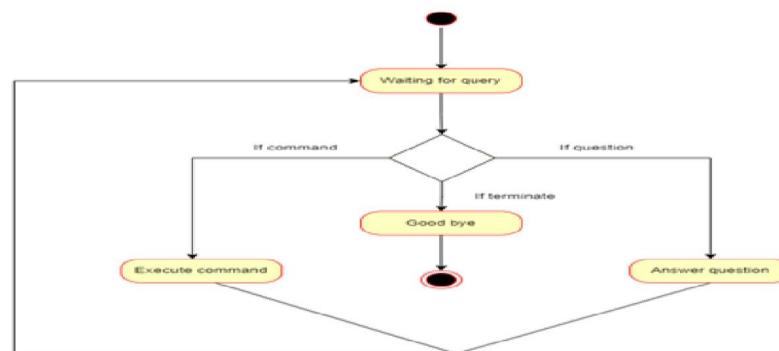


Fig. 4.2 Activity Diagram

3.1.1 INPUT DESIGN:

The design of input focus on controlling the amount of dataset as input required, avoiding delay and keeping the process simple. The input is designed in such a way to provide security. Input Design will consider the following steps:

- ❖ The dataset should be given as input.
- ❖ The dataset should be arranged.
- ❖ Methods for preparing input validations.



3.1.2 OUTPUT DESIGN:

A quality output is one, which meets the requirement of the user and presents the information clearly. In output design, it is determined how the information is to be displayed for immediate need. Designing computer output should proceed in an organized, well thought out manner the right output must be developed while ensuring that each output element is designed so that the user will find the system can be used easily and effectively.

3.1.3 DATABASE DESIGN:

Python provides interfaces to all major commercial databases.

GUI Programming:

Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

Click the "Start Voice Assistant" button to initiate the assistant's operation. Once started, users can communicate with the assistant using voice commands. Saying "exit" stops the assistant and displays a gratitude message. Users can restart the assistant by clicking the button again.

3.2 MODULES:

A module allows you to logically organize your Python code. Grouping related code into a module makes.

The Artificial Intelligence module introduces the potential of artificial intelligence (AI) – the intelligence demonstrated by a machine when it perceives its environment and takes actions that maximise the likelihood of achieving specific goals.

Module overview

Category	Content
What is Artificial Intelligence (AI)?	<ul style="list-style-type: none"> • Define the term AI • Three stages of AI: narrow, general, super • Key milestones in the development of AI
How does AI work?	<ul style="list-style-type: none"> • Key principles underpinning AI: algorithms, complexity, heuristics • Machine learning definition and key characteristics • Neural network: definition and key characteristics • Deep learning: definition and key characteristics
Common AI examples	<ul style="list-style-type: none"> • Identify the need for AI in organisations and society • Examples of how AI supports data mining • Examples of how AI supports natural language processing • Examples of how AI supports decision making

4.1 TESTING AND METHODOLOGIES

4.1.1 SYSTEM TESTING:

System testing was done by giving different training and testing datasets. This test was done to evaluate whether the system was predicting accurate result or not. During the phase of the development of the system our system was tested time and again.

4.1.2 UNIT TESTING:

In unit testing, we designed the whole system in modularized pattern and each module was tested. Till we get the accurate output from the individual module we worked on the same module.

4.2 INTEGRATION TESTING:

After constructing individual modules all the modules were merged and a complete system was made. Then the system was tested whether the prediction given by training dataset to testing set was correct or not. We tried to meet the accuracy as higher as much as we can get. After spending a couple of days in integration testing the average accuracy of our system was 91%.

4.2.1 ALPHA TESTING:

Alpha testing is the first stage of software engineering which is considered as a simulated or actual operational testing done by the individual member of the project. Alpha testing is conducted by the project developers, in context of our project.

4.2.2 BETA TESTING:

Beta testing comes continuously after alpha testing which is considered as a form of external user acceptance testing. The beta version of the program is developed to and provided to limited audience. This is the final test process in the case of this project. In this system the beta-testing is done by our colleagues and the project supervisor.

4.3 IMPLEMENTATION

4.3.1 PYTHON IMPLEMENTATION:

The standard implementation of python is usually called "cpython" it is the default and widely implementation of the python prochain. Besides Them implementation alternatives of python, such as jython, ironpython, stackless and pypt



all these python implementations have specific purpose and roles. All of them make use of simple python language to execute programs in different ways. Different python implementations are briefly explained ahead.

Implementation Work Details

Virtual Assistant is a desktop voice assistant that can perform many daily tasks on the desktop like playing music or opening your favourite IDE with the help of a single voice command. Virtual Assistant is different from other traditional voice assistants in terms of the fact that it is specific to desktops and the user does not need to make an account to use it; it does not require any internet connection while getting instructions to perform any specific task.

A. Real Life Application

1. **Saves Time:** Virtual Assistant is a desktop voice assistant that works on the voice commands offered to it, can do voice searching, and can let us complete a set of tasks.
2. **Conversational Interaction:** It makes it easier to complete any task as it automatically does it by using the essential modules or libraries of Python in a conversational way. Hence, any user, when instructing it to do any task, feels like giving a task to a human assistant because of the conversational interaction between giving input and getting the desired output in the form of a task done.
3. **Reactive Nature:** The desktop assistant is reactive, which means it knows human language very well, understands the context that is provided by the user, and gives a response in the same way, i.e., in human-understandable English.



So, the user finds its reaction in an informed and smart way.

4. **Multitasking:** Its main application is its multitasking ability. It can ask for continuous instructions one after the other until the user "quits" it.
5. **No Trigger Phase:** It asks for the instructions and listens to the response that is given by the user without needing any trigger phase, and then only executes the task.

B. Data Implementation

As the first step, we will install all the necessary libraries and packages. The command used to install the libraries is "pip install," and then import them. The necessary packages included are as follows:

1. **Pyttsx3:** This is a python library used for converting text to speech.
2. **Speech recognition:** This is a python library that converts speech to text.
3. **Pywhatkit:** This is a python library for sending whatsapp messages with some additional features.
4. **Date & time:** This library provides us with the current date and time.
5. **Wikipedia:** This is a python module for searching anything on wikipedia.
6. **Pypdf2:** It is a python module that can read, split, and merge any pdf.
7. **Pyjokes:** This is a python library that is used to create one-line jokes for programmers.



8. **Web browser:** It is a python library that is used to open the given url using the default browser.
9. **Pyautogui:** This is a cross-platform gui automation python module that is used to programmatically control the mouse and keyboard.
10. **Os:** The os module in python provides functionality for interacting with the operating system.
11. **Sys:** This allows control of the interpreter as it provides access to variables and functions usually associated with the interpreter.
12. **Request:** Used to send http requests to the specified url.
13. **Get:** It is used to get data from the given url.
14. **Sleep:** It is used to stop the execution of a program for a given number of seconds.
15. **Keyboard:** It is a python library that allows us to get full control over the keyboard.
16. **Wolfram alpha:** This is an api that calculates expert responses using wolfram algorithms, knowledge bases, and artificial intelligence.
17. **Json:** It is a python library that is used for processing and reading json files.
18. **Googletrans:** It is a python library that is used to call methods such as detect and translation.
19. **Gtts:** It is a python library that is used to convert text to audio that can be saved as mp3 files.



20. **Playsound:** It is a cross-platform module that can play audio files.
21. **Pywikihow:** It is a python library that is used to search for anything on wikihow.
22. **Openai:** It is a python library that conveniently accesses the open ai api for applications written in python.
23. **Tkfilebrowser:** Tkfilebrowser has replaced tkinter.filedialog, which allows the user to select files or directories.
24. **Cv2:** Cv2 is a python library designed to solve computer vision problems.
25. **Win32api:** This provides access to many windows apis from python.
26. **Numpy:** It is a python library that is used for working with arrays and numeric data.
27. **Pyqt5:** PyQt5 is the most important python binding. It has several gui widgets. PyQt5 has some important python modules like qtwidgets, qtcore, qtgui, etc.

C. Functions

1. **Takecommand():** The function is used to take the command as input through the microphone of the user and return the output as a string.
2. **Wishme():** This function greets the user according to the time, like Good Morning, Good Afternoon," and Good Evening.
3. **Taskexecution():** This is the function that contains all the necessary task execution definitions like sendemail(), pdf_reader(), news(), and many conditions in the if-elif-else ladder like "open Google,", "open Notepad,", "search on Wikipedia" ,"play music," "open command prompt," etc.



5.1 SYSTEM STUDY:

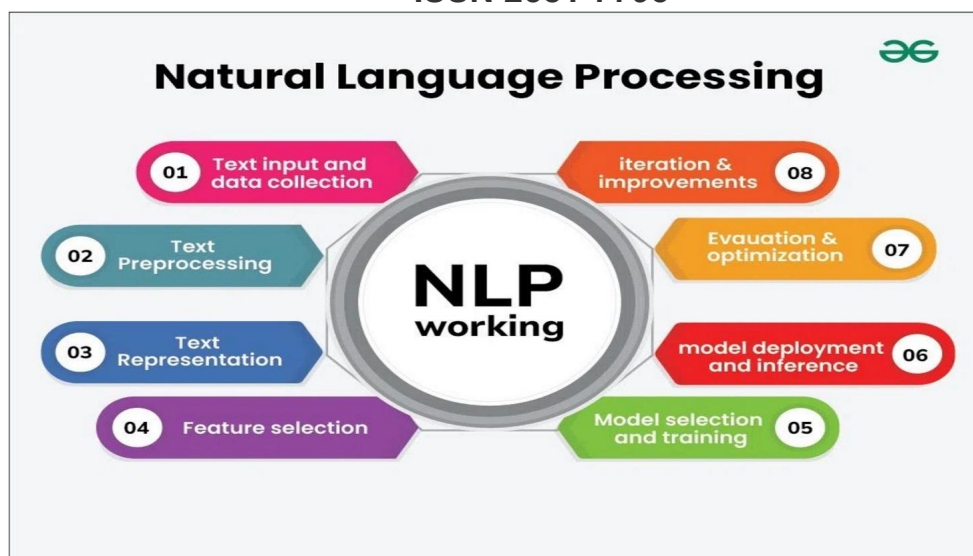
System study contains existing and proposed system details. Existing system is useful to develop proposed system. To elicit the requirements of the system and to identify the elements, Inputs, Outputs, subsystems and the procedures, the existing system had to be examined and analyzed in detail.

This increases the total productivity. The use of paper files is avoided and all the data are efficiently manipulated by the system. It also reduces the space needed to store the larger paper files and records.

5.1.1 NATURAL LANGUAGE PROCESSING:

Natural Language Processing (NLP) is the study of letting computers understand human languages. Without NLP, human language sentences are just a series of meaningless symbols to computers. Computers don't recognize the words and don't understand the grammars. NLP can be regarded as a "translator", who will translate human languages to computer understandable information.

Traditionally, users need to follow well-defined procedures accurately, in order to interact with computers. For example, in Linux systems, all commands must be precise. A single replace of one character or even a space can have significant difference. However, the emergence of NLP is changing the way of interacting. Apple Siri and Microsoft Cortana have made it possible to give command in everyday languages and is changing the way of interacting.



Working in natural language processing (NLP) typically involves using computational techniques to analyze and understand human language. This can include tasks such as language understanding, language generation, and language interaction.

Text Input and Data Collection:

- **Data Collection:** Gathering text data from various sources such as websites, books, social media, or proprietary databases.
- **Data Storage:** Storing the collected text data in a structured format, such as a database or a collection of documents.

Text preprocessing:

Preprocessing is crucial to clean and prepare the raw text data for analysis. Common preprocessing steps include:

- **Tokenization:** Splitting text into smaller units like words or sentences.
- **Lowercasing:** Converting all text to lowercase to ensure uniformity



- **Stopword Removal:** Removing common words that do not contribute significant meaning, such as “and,” “the,” “is.”
- **Punctuation Removal:** Removing punctuation marks.
- **Stemming and Lemmatization:** Reducing words to their base or root forms. Stemming cuts off suffixes, while lemmatization considers the context and converts words to their meaningful base form.
- **Text Normalization:** Standardizing text format, including correcting spelling errors, expanding contractions, and handling special characters.

Text Representation:

- **Bag of Words (BoW):** Representing text as a collection of words, ignoring grammar and word order but keeping track of word frequency.
- **Term Frequency-Inverse Document Frequency (TF-IDF):** A statistic that reflects the importance of a word in a document relative to a collection of documents.
- **Word Embeddings:** Using dense vector representations of words where semantically similar words are closer together in the vector space (e.g., Word2Vec, GloVe).

Feature Extraction:

Extracting meaningful features from the text data that can be used for various NLP tasks.

- **N-grams:** Capturing sequences of N words to preserve some context and word order.



- **Syntactic Features:** Using parts of speech tags, syntactic dependencies, and parse trees.
- **Semantic Features:** Leveraging word embeddings and other representations to capture word meaning and context.

Model Selection and Training:

Selecting and training a machine learning or deep learning model to perform specific NLP tasks.

- **Supervised Learning:** Using labeled data to train models like Support Vector Machines (SVM), Random Forests, or deep learning models like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs).
- **Unsupervised Learning:** Applying techniques like clustering or topic modeling (e.g., Latent Dirichlet Allocation) on unlabeled data.
- **Pre-trained Models:** Utilizing pre-trained language models such as BERT, GPT, or transformer-based models that have been trained on large corpora.

Model Deployment and Inference:

Deploying the trained model and using it to make predictions or extract insights from new text data.

- **Text Classification:** Categorizing text into predefined classes (e.g., spam detection, sentiment analysis).
- **Named Entity Recognition (NER):** Identifying and classifying entities in the text.
- **Machine Translation:** Translating text from one language to another.
- **Question Answering:** Providing answers to questions based on the context provided by text data.



Evaluation and Optimization:

Evaluating the performance of the NLP algorithm using metrics such as accuracy, precision, recall, F1-score, and others.

- **Hyperparameter Tuning:** Adjusting model parameters to improve performance.
- **Error Analysis:** Analyzing errors to understand model weaknesses and improve robustness.

Iteration and Improvement:

Continuously improving the algorithm by incorporating new data, refining preprocessing techniques, experimenting with different models, and optimizing features.

5.1.2 INSTALLING NLTK DATA:

NLTK is a popular open-source library in Python that provides tools for NLP tasks such as tokenization, stemming, and part-of-speech tagging. Other popular libraries include spaCy, OpenNLP, and CoreNLP.

NLTK comes with many corpora, toy grammars, trained models, etc. A complete list is posted at: https://www.nltk.org/nltk_data/



5.2 EXTENDABLE:

It allows to add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

5.2.1 OBJECT-ORIENTED APPROACH:

One of the key aspects of Python is its object-oriented approach. This basically means that Python recognizes the concept of class and object encapsulation thus allowing programs to be efficient in the long run.

5.2.2 HIGHLY DYNAMIC:

Python is one of the most dynamic languages available in the industry today. There is no need to specify the type of the variable during coding, thus saving time and increasing efficiency.

5.2.3 EXTENSIVE ARRAY OF LIBRARIES:

Python comes inbuilt with many libraries that can be imported at any instance and be used in a specific program.



5.2.4 OPEN SOURCE AND FREE:

Python is an open-source programming language which means that anyone can create and contribute to its development. Python is free to download and use in any operating system, like Windows, Mac or Lin.

5.3 OPENCV:

Open CV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.



6.1 CONCLUSION:

In conclusion, voice assistants powered by Natural Language Processing (NLP) have revolutionized the way we interact with technology and have become integral parts of our daily lives. NLP enables voice assistants to understand and interpret human language, allowing them to provide accurate and contextually relevant responses.

Voice assistants using NLP have greatly improved the user experience by providing seamless and intuitive interactions. They can perform a wide range of tasks, such as answering questions, setting reminders, playing music, providing weather updates, controlling smart home devices, and much more. The ability to communicate with voice assistants using natural language has eliminated the need for complex commands or interfaces, making technology more accessible to a wider audience.

NLP-based voice assistants have evolved significantly over the years, thanks to advancements in machine learning and deep learning algorithms. They have become better at understanding speech patterns, accents, and even contextual nuances, making interactions more human-like and personalized.

However, challenges still exist in the development of voice assistants using NLP. These include accurately understanding ambiguous queries, ensuring data privacy and security, and addressing ethical concerns related to data usage and bias.



Ongoing research and development efforts are focused on addressing these challenges and further enhancing the capabilities of voice assistants.

In conclusion, voice assistants powered by NLP have transformed the way we interact with technology and have become valuable tools in our daily lives. With continuous advancements, voice assistants will likely play an even more significant role in the future, making technology more accessible, personalized, and efficient.

7.1 FUTURE ENHANCEMENT:

10 KEY EXPECTATIONS FOR THE FUTURE:

In the past decade, AI voice assistants have evolved from simple query responders to sophisticated virtual companions that millions rely on daily. With giants like Amazon, Google, and Apple leading the charge, the growth in this space shows no signs of slowing down.

As AI technology continues to advance, what can we expect from voice assistants in the future? Here are ten key predictions that will shape the next wave of AI-driven voice experiences.

Natural language processing (NLP) will get even smarter:

Natural language processing is the backbone of AI voice assistants. As NLP technology improves, we can expect voice assistants to understand and process more complex commands with higher accuracy.

This means that interactions will feel more natural, resembling human conversation more closely. This development will be crucial for industries such as healthcare and customer service, where precise communication is vital.

**Multilingual capabilities will expand:**

Currently, AI voice assistants are proficient in several languages, but the future holds even more extensive multilingual capabilities. Assistants will not only understand and respond in multiple languages but will also be able to switch between them seamlessly within a single conversation.

This will be especially beneficial for global businesses and multilingual households, enhancing user experience by making communication across languages easier and more intuitive.

Personalization will reach new heights:

Voice assistants will soon know you better than ever. With advancements in machine learning, these AI-driven tools will gather data from various touch points to offer hyper-personalized experiences.

Whether it's recommending products based on past purchases, scheduling reminders tailored to your habits, or adjusting the tone and style of interaction based on your mood, the future of AI voice assistants is one of deep personalization. This will likely increase user engagement and satisfaction as interactions become more relevant and efficient.

Integration with IOT will be seamless:

The Internet of Things (IoT) has already started intertwining with AI voice assistants, but the future promises even more profound integration. Imagine controlling every smart device in your home—lights, thermostats, security systems, and more—through simple voice commands.

The seamless integration of voice assistants with IoT devices will transform how we interact with our environments, making smart homes more intelligent and responsive to our needs.

**Voice biometrics will enhance security:**

Security concerns have always been a hurdle for AI voice assistants, but voice biometrics could change that. This technology uses unique vocal features to verify a user's identity, making voice commands more secure.

In the future, we can expect voice biometrics to be a standard feature in voice assistants, providing an extra layer of security for tasks like making purchases, accessing sensitive information, or controlling smart home devices.

Voice commerce will flourish:

The convenience of voice commerce, or v-commerce, is set to grow exponentially. Voice assistants will become more adept at handling transactions, from ordering groceries to booking flights, all through simple voice commands.

As consumers grow more comfortable with this technology, businesses will need to optimize their platforms for voice search and transactions to stay competitive in the evolving e-commerce landscape.

Ai voice assistants in healthcare:

AI voice assistants are poised to make significant inroads in healthcare. They will assist with tasks ranging from appointment scheduling to providing medical advice based on symptom descriptions.

For patients with chronic conditions, voice assistants could offer medication reminders, monitor health metrics, and even provide emotional support. This could revolutionize patient care, making healthcare more accessible and personalized.

**Contextual understanding will improve:**

One of the most exciting predictions is the enhancement of contextual understanding in AI voice assistants. Future models will not only respond to what you say but will also consider the context—your location, the time of day, your previous interactions, and more. This will allow for more accurate and contextually relevant responses, improving user satisfaction and making interactions smoother.

Ai voice assistants in education:

The educational sector will also see a boost from AI voice assistants. These tools will become personalized tutors capable of answering student queries, offering explanations, and even providing feedback on assignments. With the rise of remote learning, voice assistants will play a crucial role in making education more interactive and accessible, catering to the individual needs of students.

Ethical AI will be a priority:

As AI voice assistants become more ingrained in our daily lives, the ethical implications of their use will come to the forefront. Issues such as data privacy, bias in AI algorithms, and the potential for misuse will need to be addressed. Companies developing these technologies will prioritize ethical AI practices, ensuring that voice assistants are not only helpful but also fair and secure.



8.1 BIBLIOGRAPHY:

For a bibliography on AI voice assistant projects, key areas to focus on include: speech recognition, natural language processing (NLP), text-to-speech synthesis, dialogue management, machine learning models used in voice assistants, and the applications and challenges of developing such systems; here are a few relevant research papers and articles to consider:

General Concepts and Overview:

- **"Artificial Intelligence-based Voice Assistant" by Sahu, Ajay et al. (SSRN):**
Provides a comprehensive overview of the key components of an AI voice assistant, including speech recognition, NLP, and text-to-speech functionalities.

- **"Voice Assistant Using Artificial Intelligence" by Sahu, Ajay et al. (Proceedings of the International Conference on Innovative Computing & Communication):**

Discusses the design and implementation of a voice assistant system, highlighting the challenges and potential applications.



Speech Recognition:

- **"Deep Neural Networks for Automatic Speech Recognition" by Hinton et al. (IEEE Transactions on Audio, Speech, and Language Processing):**
Explains the application of deep learning techniques for speech recognition, a core component of voice assistants.
- **"A Hybrid Approach to Speech Recognition Using Deep Learning and Hidden Markov Models" by Li et al. (IEEE Transactions on Neural Networks and Learning Systems):**
Describes a combined approach using deep learning and traditional HMMs for improved speech recognition accuracy.

Natural Language Processing (NLP):

- **"Natural Language Processing with Deep Learning" by Yoav Goldberg (Synthesis Lectures on Human Language Technologies):**
A comprehensive overview of NLP techniques utilizing deep learning models, crucial for understanding user intent in voice assistants.
- **"Dialogue State Tracking for Conversational AI Systems" by Williams et al. (Proceedings of the 16th Conference on Computational Natural Language Learning):**
Discusses methods for tracking the context of a conversation, essential for building a coherent dialogue with a voice assistant.

Text-to-Speech Synthesis:

- **"A Statistical Parametric Speech Synthesis System Based on a Large Speech Corpus" by K. Tokuda et al. (IEEE Transactions on Audio, Speech, and Language Processing):**



Explains the principles behind text-to-speech synthesis using statistical modeling.

- **"Neural Text-to-Speech Synthesis with Monotonic Alignment" by Shen et al. (Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing):**

Presents advanced neural network-based TTS techniques for generating natural-sounding speech.

Applications and Challenges:

- **"Challenges and Opportunities in Conversational AI" by Vinyals et al. (arXiv):**

Discusses the current challenges and future directions in developing conversational AI systems, including voice assistants.

- **"Designing Voice Interfaces for Smart Home Devices" by O'Brien et al. (Proceedings of the ACM Symposium on User Interface Software and Technology):**

Explores user interface design considerations specific to voice-activated smart home devices.

Important Considerations:

- **Specific platforms and APIs:**

If you are building your voice assistant on a particular platform like Amazon Alexa, Google Assistant, or Microsoft Cortana, research their documentation and APIs.

- **Privacy and security:**



Address the potential privacy concerns related to voice data collection and processing in your project.

➤ **Domain-specific knowledge:**

If your voice assistant is intended for a niche domain, incorporate relevant research and datasets to enhance its functionality.

9.1 DATA FLOW DIAGRAM:

Data flow diagram is graphical representation of system which give detail information about data flow between input and output. As level increases it elaborates detail information about data flow.

DIAGRAM 1:

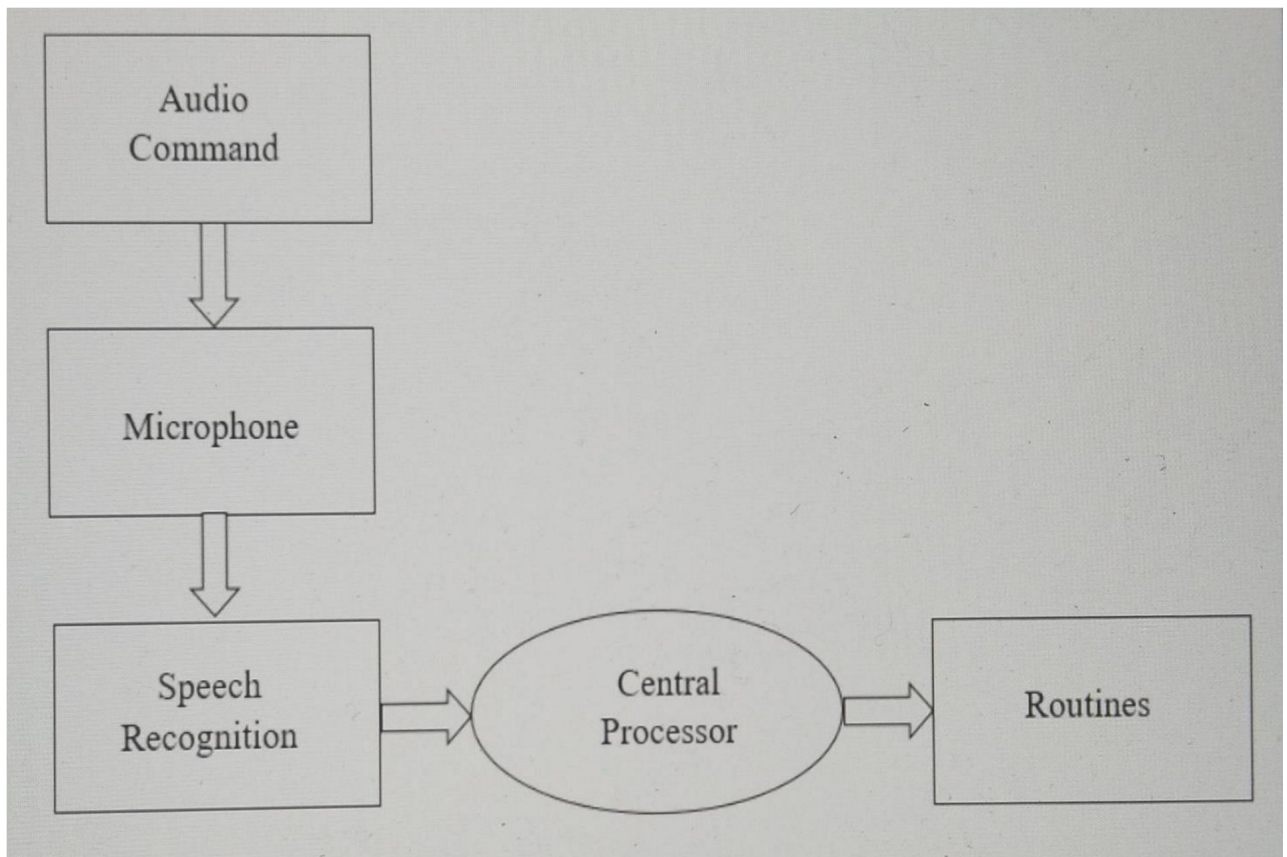


DIAGRAM 2:

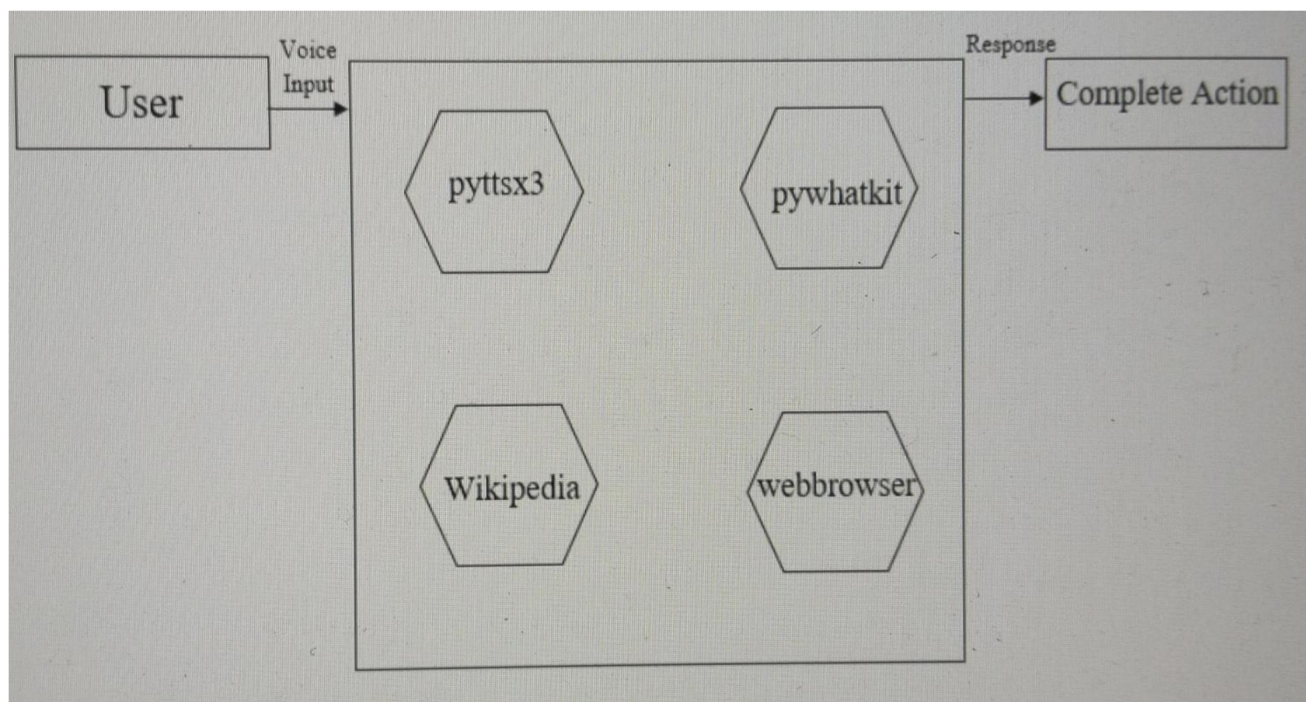




DIAGRAM 3:

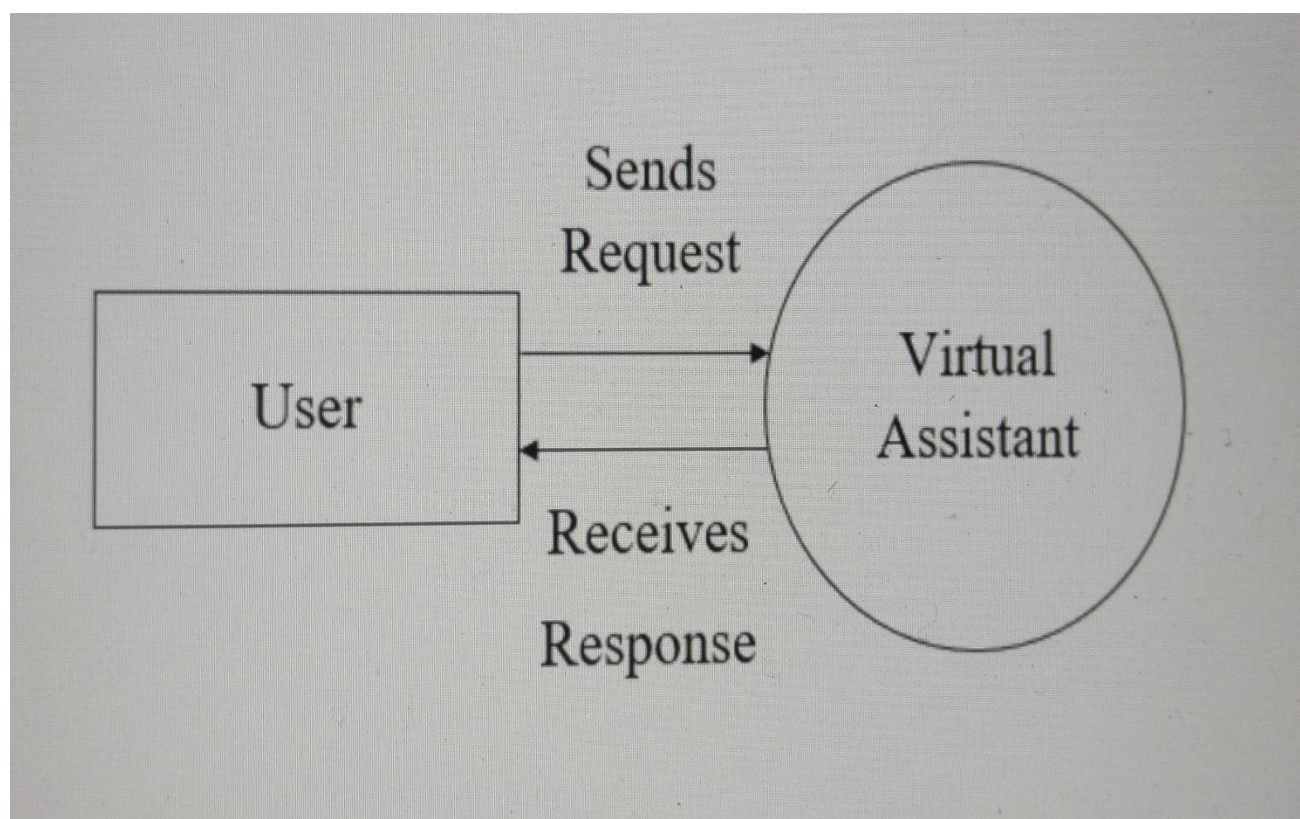


DIAGRAM 4:

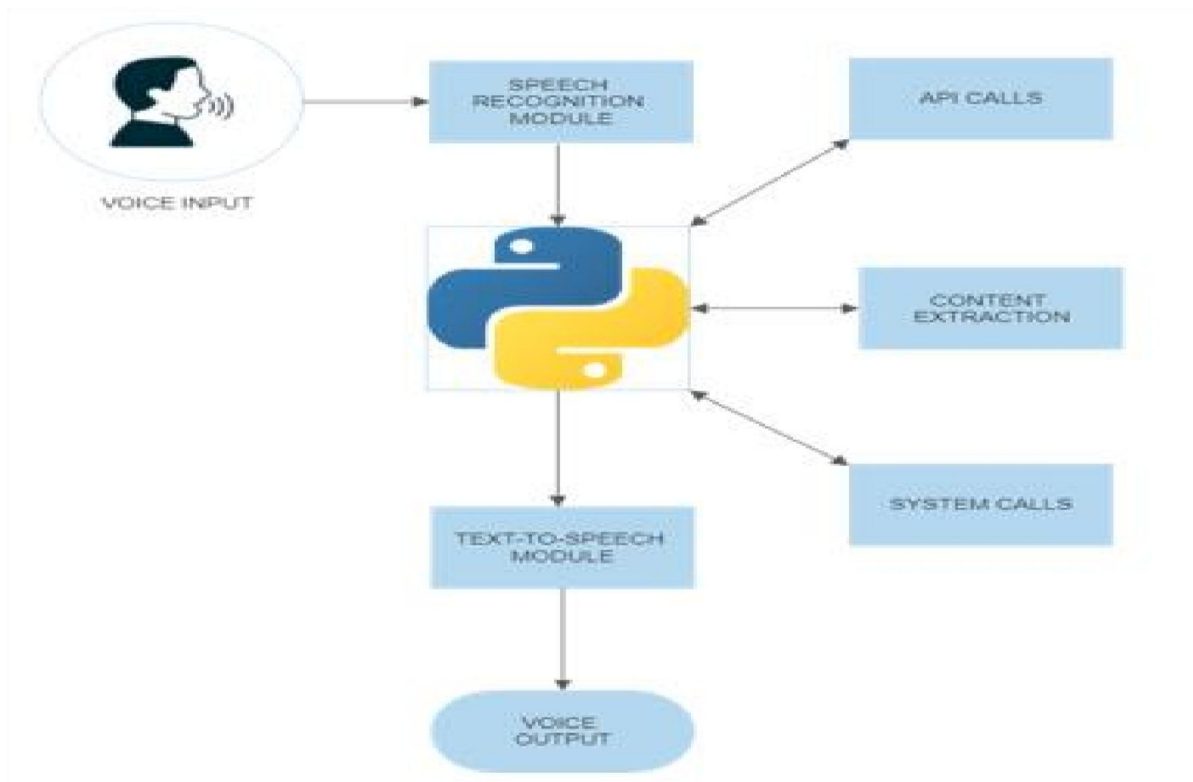
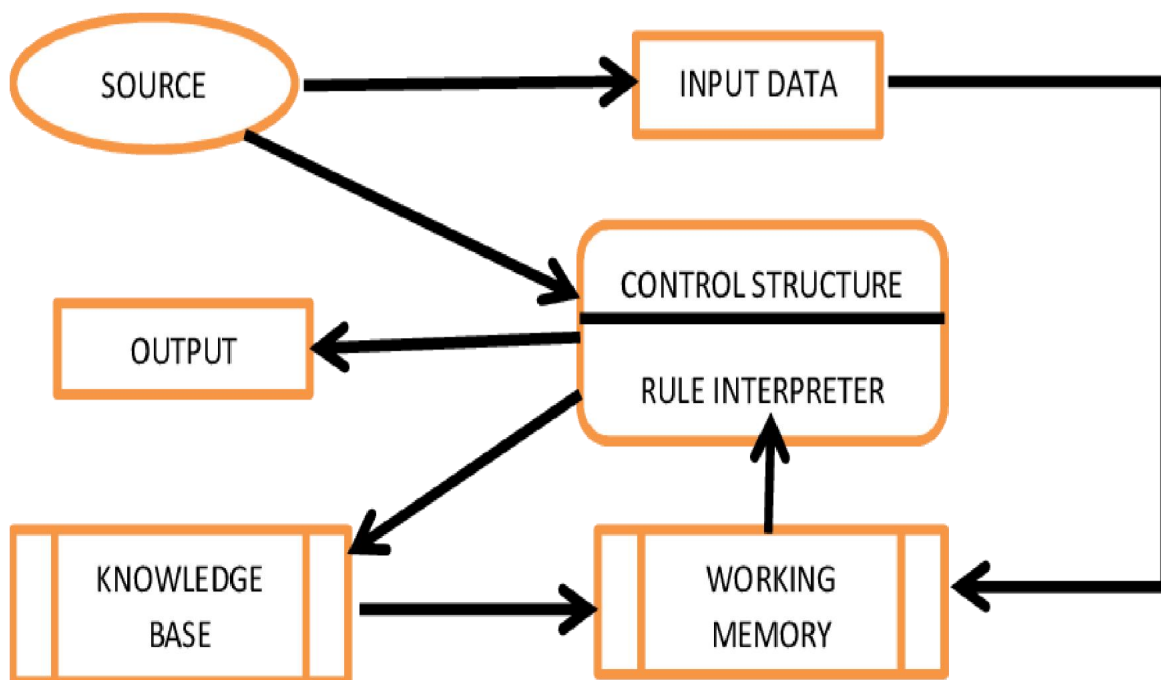


DIAGRAM 5:



9.2 TABLE DESIGN: TABLE

1:

PYTHON LIBRARIES

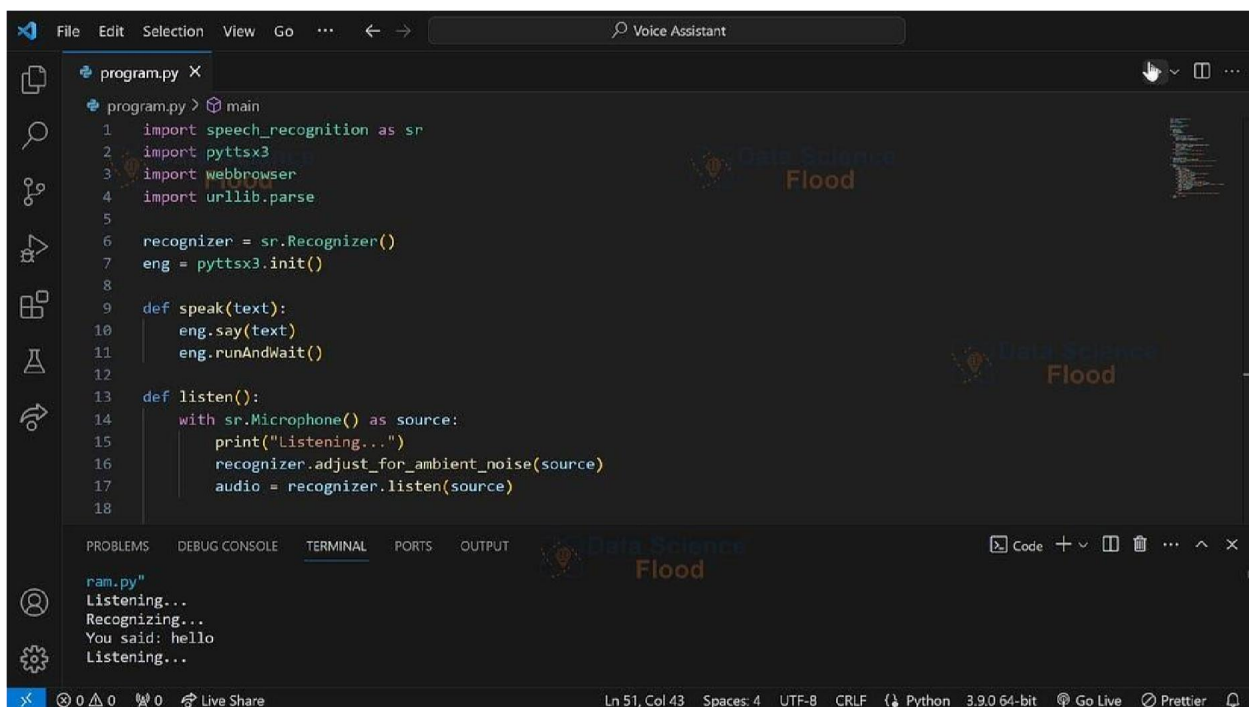
MODULES	FUNCTION	DESCRIPTION	RELEVANT PYTHON LIBRARIES
Speech recognition	Initialize microphone	Establish connection to the user's microphone	speech_recognition
Natural language processing (nlp)	Intent classification	Identify the user's intended action from the text input	NLTK, spaCy, ras
Task execution	System interaction	Execute system commands based on intent	os, webbrowser, datetime
Text-to-speech	Text to speech conversion	Generate audio from text	pyttsx3

TABLE 2

FILES	FUNCTION	DESCRIPTION	IMPORT FILES
Web browser	To open a web browsers (google, internet explorer)	This statement is used to bring the contents of another Python file (module) into your current script's namespace. This allows you to use functions, classes, and variables defined in that other file	import web browser
Wikipedia	To search a contents	This imports the wikipedia , which provides an interface to the Wikipedia API. This allows you to programmatically access and retrieve information from Wikipedia.	import wikipedia

IMPORT FILES

9.3 SAMPLE SCREENSHOTS

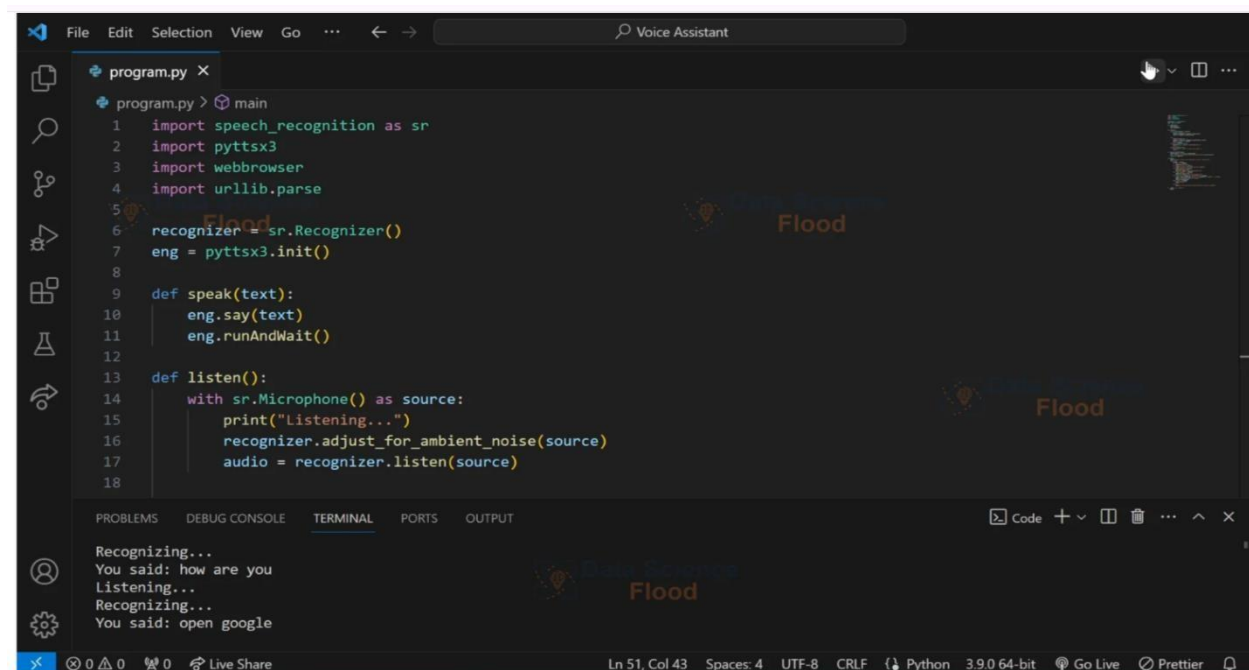


```
File Edit Selection View Go ... Voice Assistant
program.py X
program.py > main
1 import speech_recognition as sr
2 import pyttsx3
3 import webbrowser
4 import urllib.parse
5
6 recognizer = sr.Recognizer()
7 eng = pyttsx3.init()
8
9 def speak(text):
10     eng.say(text)
11     eng.runAndWait()
12
13 def listen():
14     with sr.Microphone() as source:
15         print("Listening...")
16         recognizer.adjust_for_ambient_noise(source)
17         audio = recognizer.listen(source)
18
19 ram.py"
20 Listening...
21 Recognizing...
22 You said: hello
23 Listening...
```

PROBLEMS DEBUG CONSOLE TERMINAL PORTS OUTPUT

Code + - - - - -

Ln 51, Col 43 Spaces: 4 UTF-8 CRLF Python 3.9.0 64-bit Go Live Prettier

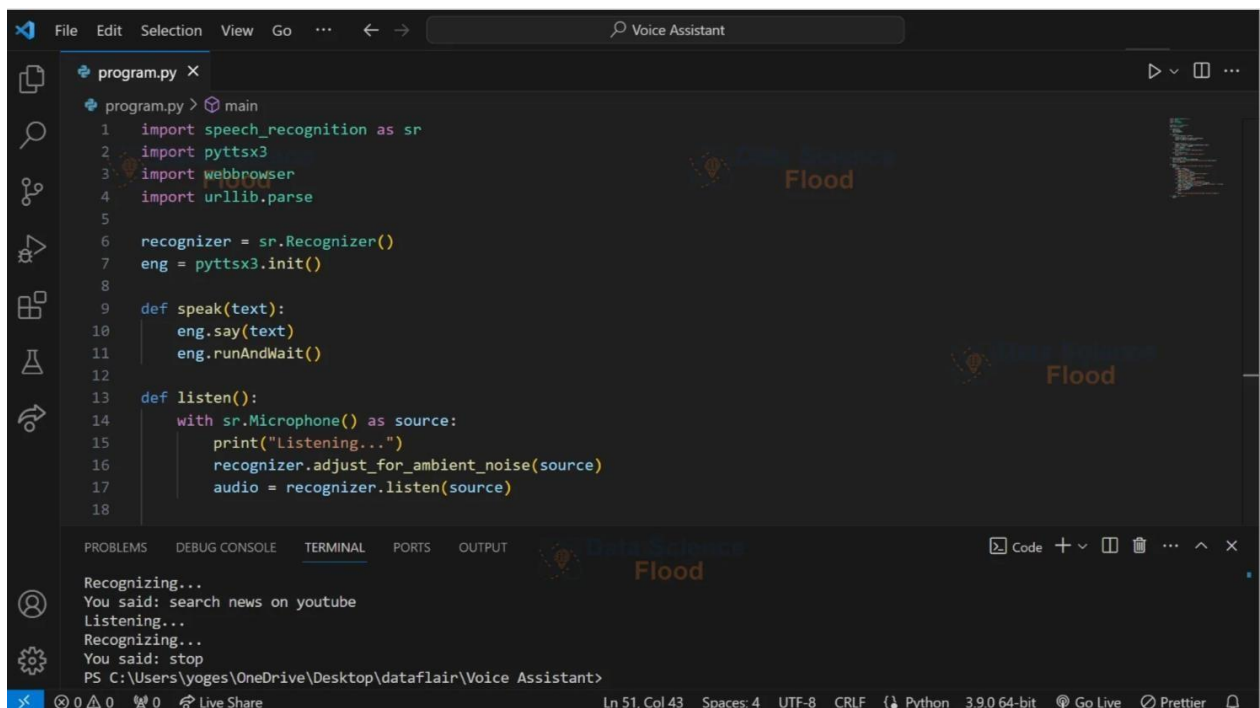
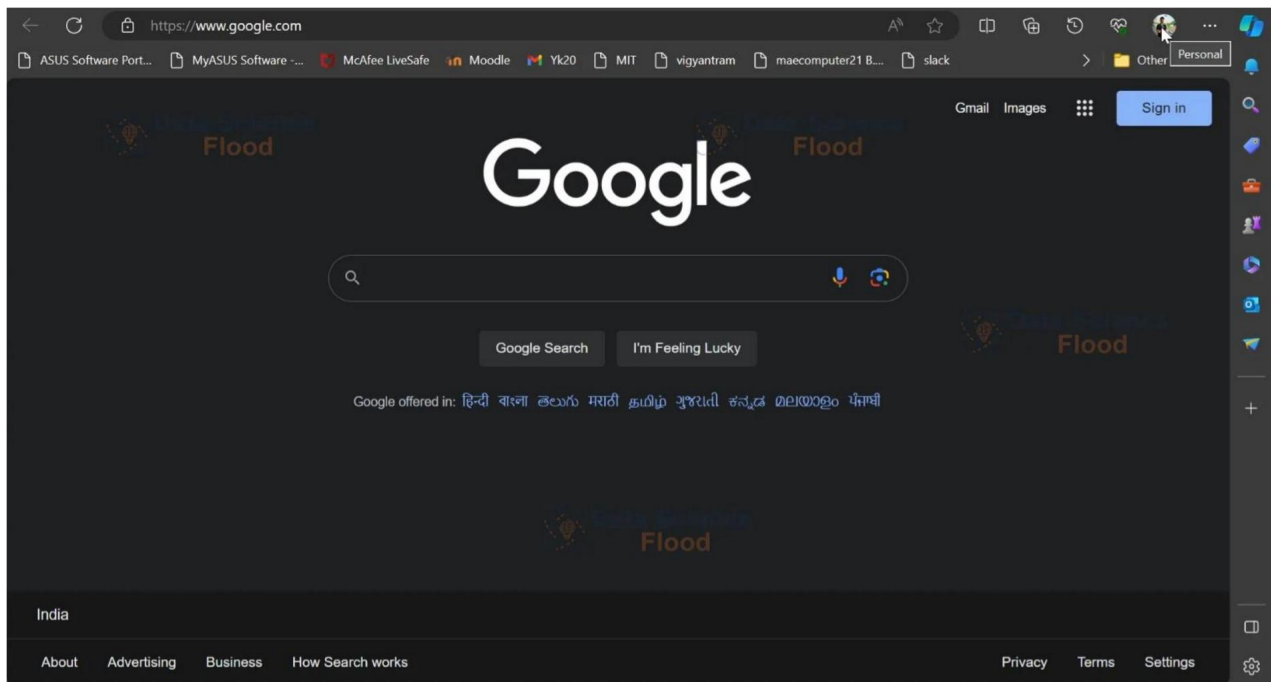


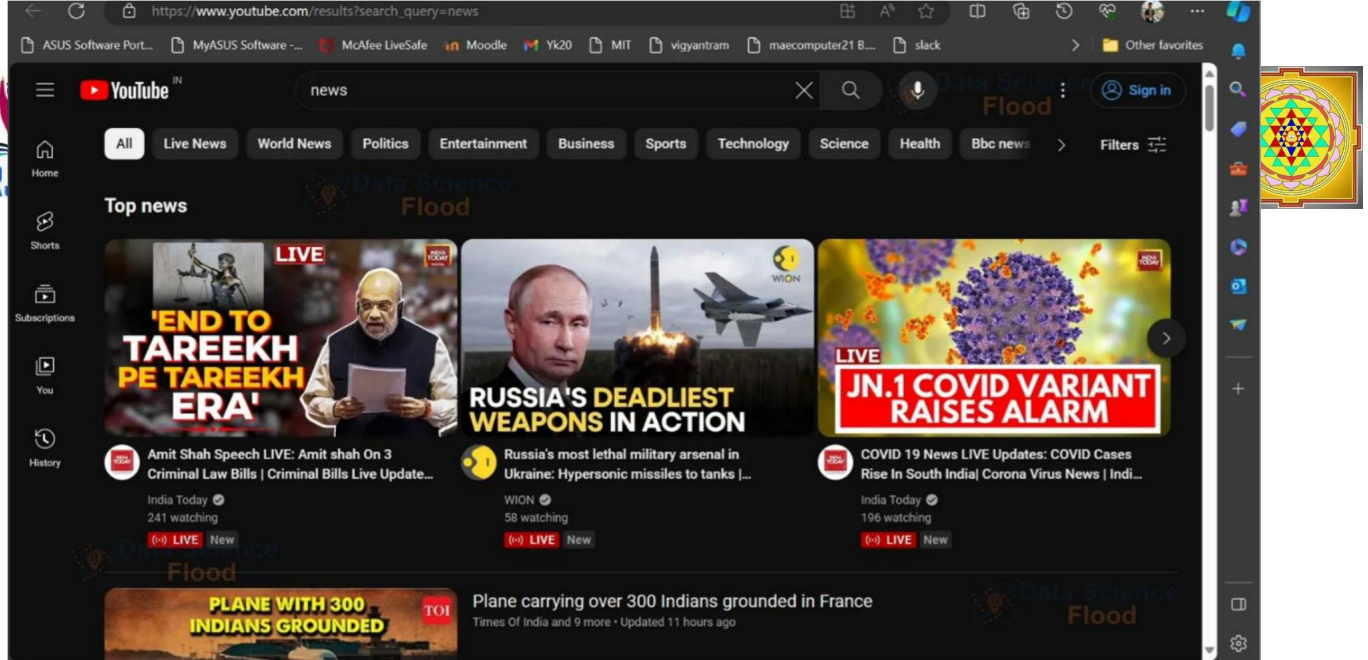
```
File Edit Selection View Go ... Voice Assistant
program.py X
program.py > main
1 import speech_recognition as sr
2 import pyttsx3
3 import webbrowser
4 import urllib.parse
5
6 recognizer = sr.Recognizer()
7 eng = pyttsx3.init()
8
9 def speak(text):
10     eng.say(text)
11     eng.runAndWait()
12
13 def listen():
14     with sr.Microphone() as source:
15         print("Listening...")
16         recognizer.adjust_for_ambient_noise(source)
17         audio = recognizer.listen(source)
18
19 Recognizing...
20 You said: how are you
21 Listening...
22 Recognizing...
23 You said: open google
```

PROBLEMS DEBUG CONSOLE TERMINAL PORTS OUTPUT

Code + - - - - -

Ln 51, Col 43 Spaces: 4 UTF-8 CRLF Python 3.9.0 64-bit Go Live Prettier





9.4 SAMPLE CODE

```
import speech_recognition as sr

import pyttsx3 import wikipedia

import os import subprocess

import sys import re

DESKTOP_PATH = "C:\\Users\\haric\\OneDrive\\Desktop"

recognizer = sr.Recognizer() tts_engine = pyttsx3.init()

voices = tts_engine.getProperty('voices')

tts_engine.setProperty('voice', voices[1].id)

last_listed_items=[] def speak(text): """Convert text to
speech.""" tts_engine.say() tts_engine.runAndWait()

def listen_command():

    """List for a voice command and return it as text."""

    with sr.Microphone()as source:

        print("Listening...")

        recognizer.adjust_for_ambient_noise(source,duration=1) audio

        = recognizer.listen(source)

        try:

            print("Recognizing...") command =

            recognizer.recognize_goole(audio) print(f"User

            said:{command}") return command.lower()

        except sr.UnknownValueError:
```

```
    speak("Sorry,I did not understand that.")

    return ""    except sr.RequestError:

        speak("Sorry,my speech service is down.")

        return ""

def open_youtube(query=None):

    """Open YouTube and optionally search for a query."""

    if query:        url

    =f"https://www.youtube.com/results?search_query={query}"

    else:

        url =f"https://www.yputube.com"

    webbrowser.open(url)    speak("Opening

    YouTube.") def

    open_wikipedia_search(query=None):

        """Search Wikipedia for a query and read a summary."""

        if query:

            try:

                summary = wikipedia.summary(query,sentences=2)

                speak(f"According to Wikipedia,{summary}")            except

                wikipedia.exceptions.DisambiguationError:

                    speak("The term is ambiguous.Please be more specific.")

            except wikipedia.exceptions.PageError:

                speak("I could not find any information on that topic.")

        else:
```

```
url = "https://www.wikipedia.org"

webbrowser.open(url)

speak("Opening Wikipedia.") def

open_chrome():      """Open Google
Chrome."""

try:

    chrome_path =r"C:\Program Files\Google\Chrome\Application\chrome.exe"

if sys.platform == "win32":      os.startfile(chrome_path)      elif sys.platform
=="darwin":

    subprocess.Popen(["/Application/Google Chrome.app/Contents/MacOS/Google Chrome"])

else:

    subprocess.Popen(["goole-chrome"])

speak("Opening Goole Chrome.")

except Exception as e:

    speak("Unable to open Chrome.")

    print(e)

def access_desktop(current_path=DESKTOP_PATH):

    """List files and folders on the desktop or specified path."""

    global last_listed_items      if os.path.exists(current_path):

    items=os.listdir(current_path)      if items:

        last_listed_items = sorted(items)      speak("Here
are the files and folders.")      for idx,item in
enumerate(last_listed_items,start=1):
```

```
    speak(f"Number {idx}:{item}.")

print(f"{idx}:{item}")    else:

    speak("The selected folder is empty.")

print("The selected folder is empty.")    else:

    speak("The specified path does not exist.")

print("The specified path does not exist")


def open_file_or_folder(section_number,current_path=DESKTOP_PATH):

    """Open a specific file or folder based on the selection number."""    global

    last_listed_items    if 1 <=selection_number <=len(last_listed_items):

        selected_item=last_listed_items[selection_number-1]

    selected_path=os.path.join(current_path,selected_item)    if

    os.path.isfile(selected_path):

        try:        if sys.platform ==

"win32":

            os.startfile(selected_path)

        elif sys.platform == "darwin":

            subprocess.call(['open',selected_path])

        else:

            subprocess.call(['xdg-open',selected_path])

    speak(f"Opening file{selected_item}.")    except

Exception as e:
```

```
speak("Unable to open the file.")

print(e)    elif:

os.path.isdir(selected_path):

try:

    if sys.platform == "win32":

os.startfile(selected_path)    elif

sys.platform == "darwin":

subprocess.call(['open',selected_path])
```