

A SMART APPLICATION WITH MACHINE LEARNING EXTENSION TO OPERATE COMPUTER AND IoT DEVICES

A Project Report

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Electronics and Communication Engineering

by

DUDIGANI UMESH (39130125)

DASARI SRI HARSHA (39130106)



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
SCHOOL OF ELECTRICAL AND ELECTRONICS**

SATHYABAMA

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

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

APRIL - 2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in

DEPARTMENT OF ELECTRONICS AND COMMUNICAITON ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the Bonafide work of **DUDIGANI UMESH (39130125)** and **DASARI SRI HARSHA (39130106)** who carried out the project entitled **"A SMART APPLICATION WITH MACHINE LEARNING EXTENSION TO OPERATE COMPUTER AND IoT DEVICES"** under our supervision from November 2022 to April 2023.

Internal Guide

Mrs. L. MAGTHELIN THERASE M.E., Ph.D.,

Head of Department

Dr. T. RAVI M.E., Ph.D.,

Submitted for Viva voce Examination held on

26-04-2023

Internal Examiner

External Examiner

DECLARATION

We, **DUDIGANI UMESH (39130125)** and **DASARI SRI HARSHA (39130106)** hereby declare that the project report entitled “**A SMART APPLICATION WITH MACHINE LEARNING EXTENSION TO OPERATE COMPUTER AND IoT DEVICES**” done by us under the guidance, **Mrs. L. MAGTHELIN THERASE M.E., Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Electronics and Communication Engineering.

DATE: 26-04-2023

PLACE: CHENNAI

SIGNATURE OF CANDIDATES

1. 
2. 

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to the **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. We are grateful to them.

We convey our thanks to **Dr. N. M. NANDHITHA, M.E., Ph.D.**, Dean, School of Electrical and Electronics Engineering and **Dr. T. RAVI, M.E., Ph.D.**, Head of the Department, Department of Electronics and Communication Engineering for providing us necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to Project Guide. **Mrs. L. MAGTHELIN THERASE M.E., Ph.D.**, Dept. of Electronics and Communication Engineering, for her valuable guidance, suggestions and constant encouragement for the successful completion of our project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Electronics and Communication Engineering who are helpful in many ways for the completion of the project

ABSTRACT

Virtual assistants are computer programs designed to simulate human interaction and provide services to users in a conversational manner. They use natural language processing and machine learning algorithms to understand user queries and provide relevant responses. In recent years, virtual assistants have gained popularity in both personal and business contexts due to their convenience and efficiency. This paper presents an overview of virtual assistants, their applications, and the technology behind them. The paper also highlights the potential benefits and challenges of using virtual assistants and discusses some of the current trends in the field. The use of virtual assistants is becoming increasingly common in areas such as customer service, healthcare, and education. They can help users to perform tasks such as scheduling appointments, ordering products, and accessing information. By automating routine tasks, virtual assistants can save users time and increase productivity. However, the use of virtual assistants also raises concerns around privacy and security. As virtual assistants collect and store user data, there is a risk that this data could be used for malicious purposes. Additionally, the use of virtual assistants raises questions around job displacement and the potential impact on the labor market. In conclusion, virtual assistants have the potential to transform the way we interact with technology and perform daily tasks. However, it is important to consider the potential risks and challenges associated with their use and to ensure that appropriate safeguards are in place to protect user privacy and security.

TABLE OF CONTENTS

Chapter No.	TITLE	Page No.
	ABSTRACT	v
	LIST OF FIGURES	viii
1	INTRODUCTION	1
	1.1 Introduction	1
	1.2 History Of Virtual Assistants	1
	1.3 Types Of Virtual Assistants	1
	1.4 Advantages Of Virtual Assistants	1
	1.5 Disadvantages Of Virtual Assistants	2
	1.6 How Virtual Assistants Work	2
	1.7 Popular Virtual Assistants	2
	1.8 Future Of Virtual Assistants	2
	1.9 Privacy Concerns	3
	1.10 Ethical Considerations	3
2	LITERATURE SURVEY	4
	2.1 Inferences From Literature Survey	4
	2.2 Open Problems In Existing System	8
3	REQUIREMENT ANALYSIS	10
	3.1 Feasibility Studies/Risk Analysis Of The Project	10
	3.1.1 Feasibility Studies	10
	3.1.2 Risk Analysis	10
	3.2 Software Requirements Specification Document	11
	3.2.1 Functional Requirements	11
	3.2.2. Non-Functional Requirement	12
	3.3. System Architecture	13
4	DESCRIPTION OF PROPOSED SYSTEM	14
	4.1. Module 1: Setting Up Initial Imports	14
	4.1.1. Text To Speech By Using Pyttsx3	14
	4.1.2. Speech Recognition Module	15
	4.1.3. Api - Gtts Api	16
	4.1.4. Importing Other Libraries	17

4.2	.Module 2: Creating Voice Engine	18
4.3	Module 3:Creating Voice Assistant To Produce The Output	19
4.3.1.	Wikipedia Search	19
4.3.2.	Launching Bolt Iot Platform	19
4.3.3.	Launching Youtube	24
4.4	Project Management Plan	25
4.5	Financial Report On Estimated Costing	26
4.6	Transition/ Software To Operations Plan	27
5	IMPLEMENTATION DETAILS	29
5.1	Development And Deployment Setup	29
5.2	Algorithms	29
5.3	Testing	30
5.3.1.	Unit Testing	30
5.3.2.	Integration Testing	31
5.3.3.	Functional Testing	31
5.3.4.	Black-Box Testing	32
5.3.5.	White-Box Testing	32
6	RESULTS AND DISCUSSION	33
6.1	OUTPUT IMAGES	34
7	CONCLUSION	36
7.1	FUTURE WORK	36
7.2	RESEARCH ISSUES	37
7.3	IMPLEMENTATION ISSUES	38
	REFERENCES	40
	APPENDIX	41
	A.SOURCE CODE	41
	B.SCREENSHOTS	60
	C.RESERCH PAPER	66

LIST OF FIGURES

Figure No.	Description of Figure	Page No.
1.1	Popular Virtual Assistants	2
2.1	Layout of Proposed System	9
3.1	Features of Proposed System	12
4.1	Text to Speech	14
4.2	gTTS API	16
4.3	Speech Recognition Module	19
4.4	Bolt WIFI Module Pin Diagram	21
4.5	Bolt WIFI Module Front View	22
4.6	Bolt WIFI Module Back View	22
4.7	Bolt Cloud	23
4.8	Bolt Cloud Remote Configuration	24
4.9	Bolt Cloud Code Editor	24

CHAPTER-1

INTRODUCTION

1.1 INTRODUCTION

Virtual assistants have become increasingly popular over the past few years. They are a type of software application that uses artificial intelligence (AI) to perform various tasks, such as scheduling appointments, sending emails, making phone calls, and more. In this article, we will be discussing virtual assistants in detail, including their history, types, advantages, disadvantages, and future prospects.

1.2 History of Virtual Assistants:

The concept of virtual assistants dates back to the 1960s when the first voice recognition system was developed. However, it was not until the late 1990s and early 2000s that virtual assistants became commercially available. The first virtual assistant was developed by the company Wildfire Communications in 1999.

1.3 Types of Virtual Assistants:

There are various types of virtual assistants available, including personal assistants, business assistants, and smart home assistants. Personal assistants are designed to perform tasks for individuals, such as scheduling appointments, setting reminders, and making phone calls. Business assistants, on the other hand, are designed to help businesses with tasks such as data entry, customer support, and scheduling meetings. Smart home assistants, such as Amazon's Alexa and Google Home, are designed to control smart devices in the home, such as lights, thermostats, and security systems.

1.4 Advantages of Virtual Assistants:

Virtual assistants offer many advantages, including increased productivity, reduced workload, and improved efficiency. They can perform tasks quickly and accurately, which can save time and reduce errors. Virtual assistants can also work around the clock, which means that tasks can be completed outside of regular business hours.

1.5 Disadvantages of Virtual Assistants:

Despite their advantages, virtual assistants also have some disadvantages. One of the main disadvantages is the potential for security breaches, as virtual assistants may have access to sensitive information. Another disadvantage is the lack of personal interaction, which may be important for certain tasks, such as customer service.

1.6 How Virtual Assistants Work:

Virtual assistants work by using natural language processing (NLP) and machine learning algorithms to understand and respond to user requests. They can recognize voice commands and text input, and then use this information to perform tasks or provide information. Virtual assistants may also use predictive analytics to anticipate user needs and provide personalized recommendations.

1.7 Popular Virtual Assistants:

There are several popular virtual assistants available, including Apple's Siri, Amazon's Alexa, Google Assistant, and Microsoft's Cortana. Each of these virtual assistants has its own unique features and capabilities, and they can be used on various devices, including smartphones, smart speakers, and smart watches.



Fig 1.1 Popular Virtual Assistants

1.8 Future of Virtual Assistants:

The future of virtual assistants looks promising, with continued advancements in AI and machine learning. Virtual assistants are likely to become

more intelligent and capable, and they may be able to perform more complex tasks in the future. They may also be integrated into more devices, such as cars and appliances, to provide a seamless user experience.

1.9 Privacy Concerns:

One of the biggest concerns with virtual assistants is privacy. As virtual assistants collect data on user behavior and preferences, there is a risk that this information could be shared with third parties or used for nefarious purposes. Companies that develop virtual assistants will need to be transparent about their data collection practices and take steps to protect user privacy.

1.10 Ethical Considerations:

There are also ethical considerations to take into account when developing and using virtual assistants. For example, virtual assistants may perpetuate biases or stereotypes if they are not programmed with diversity and inclusion in mind.

CHAPTER-2

LITERATURE SURVEY

2.1 INFERENCES FROM LITERATURE SURVEY

[1] Giancarlo Iannizzotto, Lucia Lo Bello, Andrea Nucita, Giorgio Mario Grasso, “A Vision and Speech Enabled, Customizable, Virtual Assistant for Smart Environments”, 11th International Conference on Human System Interaction (HSI), 2018

Giancarlo Iannizzotto builds a virtual assistant architecture for smart home automation systems using some of the most sophisticated technologies in computer vision, deep learning, speech generation and recognition, and artificial intelligence in this paper. The proposed assistant is effective and resource-efficient, interactive, and customizable, and it runs on a low-cost, small-sized Raspberry PI 3 device. The system was integrated with an open source home automation environment for testing purposes, and it ran for several days while people were encouraged to interact with it, proving to be accurate, reliable, and appealing.

[2] Kumaran N., Rangaraj V., Siva Sharan S., Dhanalakshmi R., "Intelligent Personal Assistant – Implementing Voice Commands enabling Speech Recognition”, International Conference on System, Computation, Automation and Networking (ICSCAN), 2020

Intelligent personal assistants (IPAs) are software agents that perform tasks on behalf of humans or individuals using commands or questions similar to chatbots. Intelligent Virtual Assistants can understand and respond to human speech using synthesized voices. IPAs and IVAs are used in a variety of applications, including home automation, task management, and media playback via voice. This paper proposes speech recognition systems and discusses the creation of a virtual personal assistant. The existing system is accessible via the internet and is managed by a third party. This application must protect personal data from unauthorized access and make use of the local database, speech recognition, and synthesizer. To recognize the speech, a parser called SURR (Semantic Unification and Reference

Resolution) is used. The synthesizer converts text to phonemes.

[3] Vinayak Iyer, Kshitij Shah, Sahil Sheth, Kailas Devadkar, “Virtual assistant for the visually impaired”, 5th International Conference on Communication and Electronics Systems (ICCES), 2020

Vinayak Iyer depicts the implementation of software that assists the visually impaired in accessing the internet. The software will play an important role in how the internet is accessed and will significantly improve usability. Despite advances in technology, the internet, particularly websites, remains inaccessible to the visually impaired. The software allows for simple interaction with these websites. It adds a new dimension to accessing and providing commands to any website by using voice commands instead of the traditional keyboard and mouse. The software can read out the website's content, and then, using speech-to-text and text-to-speech modules, as well as selenium, it can automate any website. The user is free from remembering complex braille keyboard commands or the hassle of typing, he/she can simply voice out his/her command and the software will execute it. The system can also provide a summary of the content on the website and answer questions asked by the user with reference to the summary using a BERT model trained on the Stanford Question Answer Dataset.

[4] CH.M.H. Saibaba, Saiyed Faiyaz Waris, S.Hrushikesava Raju, VSRK Sarma, Vijaya Chandra Jadala, Chitturi Prasad, “Intelligent Voice Assistant by Using OpenCV Approach”, Second International Conference on Electronics and Sustainable Communication Systems (ICESC), 2021

CH.M.H. Saibaba focused on the various artificial intelligence technologies that support voice recognition and natural language processing (NLP) in this paper. To try and do such quiet tasks, the researchers preferred a voice assistant and devices such as Alexa. Although several powerful packages, such as pywhatkit, Wikipedia, pyttsx3, pygame, speech recognition, OpenCV, and so on, will perform the same practicality, this paper concentrated on the practical implementation of a voice recognition module based on OpenCV. In modern technology, speech recognition, object identification, digital image processing, and language processing all play important roles in security and other factors. The most common mechanisms in

artificial intelligence in conjunction with machines and deep learning are those for identifying a personality's natural language.

[5] Vindhya Vasini K, Tejasvi K, Pravalika K, Ch. Nanda Krishna, “A Natural Language Processing based Intelligent Bot Application”, International Conference on Sustainable Computing and Data Communication Systems (ICSCDS), 2022

A voice-managed non-public assistant is proposed in this paper. Human voice commands are sent to the robot assistant remotely, and the bot can perform specific movements, turns, start/stop operations, and relocate an item from one location to another. The speech sign instructions are converted to textual content and sent to the robot via Bluetooth. The non-public assistant robot is aware of its current location and is built primarily on a microcontroller-based platform. Numerous experiments are used to assess the effectiveness of voice-manipulated communication over a long distance. The preliminary experiments' encouraging results round out the performance evaluation. Furthermore, potential enhancements to capability programmes in the home, hospitals, and industries are mentioned.

[6] Pooja Singh, Pinki Nayak, Arpita Datta, Depanshu Sani, Garima Raghav, Rahul Tejpal, “Voice Control Device using Raspberry Pi”, Amity International Conference on Artificial Intelligence (AICAI), 2019

This paper demonstrates the operation of an intelligent personal assistant based on the implementation of a voice command system. The device's services are dependent on the user's input in the form of voice commands and the ability to obtain information from a variety of online sources such as weather, telling time, or accessing online music applications. The Raspberry Pi serves as the primary piece of hardware for this voice-driven device. To convert a voice command to simple text, a speech-to-text engine is used. Query processing is then applied to this text using natural language processing (NLP) to interpret the intended meaning of the user's command. Text to speech conversion is used after interpreting the intended meaning to provide appropriate output in the form of speech. This device provides a platform for visually impaired people to perform daily tasks such as listening to music, checking weather conditions, checking the current time, or performing a simple

mathematical calculation.

[7] Vadaboyina Appalaraju, V Rajesh, K Saikumar, P. Sabitha, K Ravi Kiran, “Design and Development of Intelligent Voice Personal Assistant using Python”, 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), 2022

This helps us complete user activities such as searching for information about a query, viewing movie details, writing a note, sending a message to a specific person at a specific time, providing location, playing music, date and time, and opening files. To receive user input, it requires the use of a microphone. An attempt is made here to build an intelligent voice personal assistant using Python, which provides the ability to operate devices based on voice or speech for extracting information to perform tasks on a desktop or other smart device.

[8] P. Mohana, M. Muthuvinayagam, P. Umasankar, T. Muthumanickam, “Automation using Artificial intelligence based Natural Language processing”, 6th International Conference on Computing Methodologies and Communication (ICCMC), 2022

This paper proposes using a single remote to control four common home appliances: a fan, a light, a coffee machine, and door alarms. The mobile device employs natural language processing to interpret the user's voice command. The project employs a smartphone application to communicate with the Wi-Fi signals provided by these appliances. The application sends commands to the Arduino devices and receives responses from them. While a command is being transmitted from a remote location. As a result, data integrity and confidentiality, as well as authentication of the services that deliver that data, are critical. As a result, security is an important aspect of IoT. As a result, the mobile app includes a username and password-based authentication mechanism.

[9] J. Sayed, M. W. Ashour, “Digital intelligent virtual assistant (DIVA) with natural speech and accent recognition”, 4th Smart Cities Symposium (SCS 2021), 2022

This paper intends to conduct a cutting-edge review of Digital Intelligent Virtual

Assistants (DIVA) and their extended use for a wide range of users. Furthermore, the paper proposes a workflow scheme that incorporates accent recognition, voice registration, and gives users more manual control. The proposed design takes into account the user's comfort and difficulties with technology; thus, the proposed system is destined to serve as a mobile application. To achieve the aforementioned user goals, the VA employs techniques such as Natural Language Processing, Speech Pattern Recognition, Machine Learning, and Language Data Statistical Analysis. Following that, data analysis was conducted using anonymous questionnaires, and the results of this survey will allow us to validate our hypothesis.

[10] Witman Alvarado-Díaz, Brian Meneses-Claudio, “YANA, Virtual Assistant to Support Home Office”, IEEE Sciences and Humanities International Research Conference (SHIRCON), 2021

YANA is a Python-based virtual assistant that guides you through the organisation of your work schedule while also reminding you of upcoming events, allowing you to focus on other aspects of your life without worrying about missing an important event. Together with the development of YANA, it was possible to create a user interface that allows us to control and interact with YANA, as it is also capable of answering numerous questions that are asked, as well as the ability to follow a conversation; an average of 7 out of 10 was obtained from a series of questions posed to users in the evaluation system developed to evaluate YANA's interaction with users.

2.2 OPEN PROBLEMS IN EXISTING SYSTEM

The existing system for virtual assistants typically involves a software application that uses artificial intelligence and natural language processing to perform various tasks for users. The virtual assistant can be accessed through a variety of devices, including smartphones, smart speakers, and computers.

When a user makes a request to the virtual assistant, the software analyzes the input using machine learning algorithms and NLP to determine the intent behind the request. It then retrieves the relevant information or performs the requested task, such as setting a reminder or sending an email. The virtual assistant can also be programmed to recognize the user's preferences and behavior patterns, providing

personalized recommendations and anticipating their needs.

In order to work effectively, virtual assistants require a significant amount of data and computing power. This data is collected from user interactions with the virtual assistant, as well as from other sources such as social media and search engines. Companies that develop virtual assistants must also consider issues such as user privacy and ethical considerations, such as bias and diversity.

Overall, the existing system for virtual assistants is designed to provide a convenient and efficient way for users to perform tasks and access information, while also leveraging the power of AI and machine learning to improve the user experience.

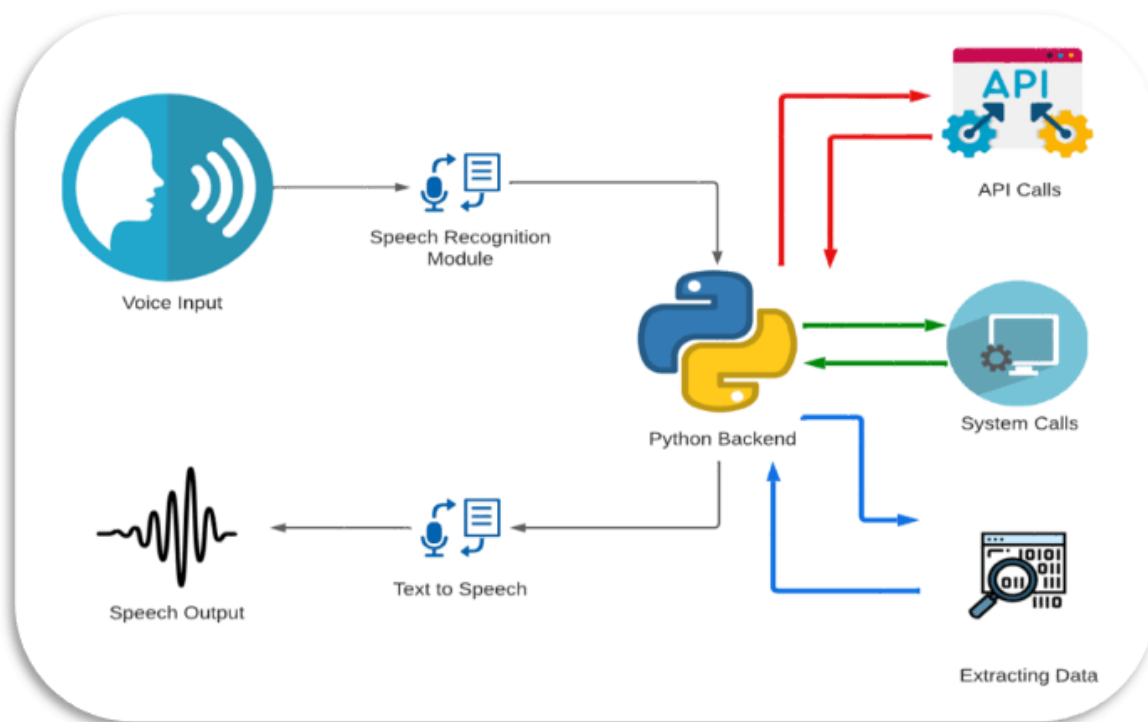


Fig 2.1: Layout of Proposed System

CHAPTER – 3

REQUIREMENT ANALYSIS

3.1 Feasibility Studies/Risk Analysis of the Project

Feasibility Studies/Risk Analysis of the Project

3.1.1 Feasibility Studies:

a. Technical Feasibility:

Virtual assistants require strong technical skills to be developed. The development process needs to be carefully planned and designed to ensure the technical feasibility of the project. There are many technologies and tools available to develop virtual assistants, such as machine learning, natural language processing, and speech recognition. Technical feasibility requires careful consideration of the hardware and software required for development, the availability of development tools and software, and the skills required to develop the system.

b. Operational Feasibility:

The virtual assistant must be designed to work with a variety of devices and operating systems. The operational feasibility of the project depends on the ease of use and compatibility with different devices. The virtual assistant should be designed to operate with minimal user input, have a simple user interface, and be easy to use.

c. Economic Feasibility:

The development of a virtual assistant can be expensive. Economic feasibility must be considered during the planning and development stages of the project. The cost of development should be analyzed, and a budget should be created to ensure that the project is financially viable.

3.1.2 Risk Analysis:

a. Technical Risks:

There may be technical issues that arise during the development of the virtual assistant, such as bugs, compatibility issues, or security vulnerabilities. These issues can delay the development process, cause unexpected costs, or lead to system failure. To mitigate these risks, testing and quality assurance should be implemented throughout the development process.

b. User Acceptance:

User acceptance is an important factor in the success of the virtual assistant. Users may not accept the system if it is not user-friendly, does not meet their needs, or has limited functionality. The development team should gather feedback from potential users to ensure that the system meets their needs and is user-friendly.

c. Security Risks:

Security risks are a major concern in the development of a virtual assistant. The system may be vulnerable to attacks or data breaches, which can result in the loss of sensitive data or damage to the system. Security measures should be implemented throughout the development process to ensure that the system is secure and user data is protected.

d. Integration Risks:

The virtual assistant may need to integrate with other systems or platforms, such as social media or messaging apps. Integration risks include compatibility issues, data transfer issues, and issues related to API changes or updates. To mitigate these risks, the development team should plan the integration process carefully and perform extensive testing to ensure that the system functions as intended.

3.2 Software Requirements Specification Document

3.2.1 Functional Requirements:

a. Speech Recognition:

The system should be capable of recognizing voice commands and translating them into text format for further processing.

b. Natural Language Processing:

The software must be capable of processing natural language commands and queries for user interactions.

c. Voice Synthesis:

The software should have the capability to synthesize speech from text responses to interact with the user.

d. Web Search:

The software should be able to perform web search based on the user's command and queries.

e. Opening and Launching Applications:

The software should have the capability to open and launch various applications such as notepad, calculator, etc.

f. Sending Email and WhatsApp Messages:

The software should be capable of sending emails and WhatsApp messages on behalf of the user.

g. System Management:

The software should have the ability to manage system functions like shut down, sleep, hibernate, and restart the computer.

h. Providing Information:

The software should be able to provide information on various topics from Wikipedia.



Fig 3.1: Features of Proposed System

3.2.2. Non-Functional Requirements:

a. User Interface:

The software should have an easy-to-use interface that is understandable to a non-technical user.

b. Performance:

The software should be able to recognize and process commands quickly and efficiently.

c. Security:

The software should ensure that user data and information are kept safe and secure.

d. Compatibility:

The software should be compatible with a wide range of operating systems and devices.

e. Reliability:

The software should be reliable and operate without any glitches or errors.

f. Scalability:

The software should be scalable and capable of handling large volumes of user requests.

3.3. System Architecture:

The virtual assistant software will be designed as a client-server model. The client will be the user interface and the server will be responsible for processing user requests and performing the necessary actions. The system will be designed using Python programming language and various libraries and APIs such as pytsx3, SpeechRecognition, wikipedia, etc. will be used to implement the functional requirements of the system.

CHAPTER - 4

DESCRIPTION OF PROPOSED SYSTEM

4.1. MODULE 1: Setting Up Initial Imports

4.1.1. Text to Speech by using pyttsx3

Pyttsx3, which is used for text to speech in Python, will now be configured as our engine. sapi5, which is a Microsoft voice application platform interface, will be utilized by us for the text to speech function. This particular module will be utilized by us. Python's pyttsx3 module is capable of converting text into spoken language. In contrast to other libraries, it may be used without an internet connection and is compatible with Python versions 2 and 3. In order to obtain a reference to a pyttsx3. Engine instance, an application must first call the pyttsx3.init() factory method. It is a very simple and straightforward process to transform the written words into spoken words with this programmed. The pyttsx3 module is capable of supporting two different voices. The first is the female version, while the second is the male version, both of which are offered by the "sapi5" programmed for Windows. It is equipped with three different TTS engines.:

sapi5 – SAPI5 on Windows

nsss – NSSpeechSynthesizer on Mac OS X

espeak – eSpeak on every other platform



Fig 4.1 *Text to Speech*

For example,

```
# Import the required module for text
# to speech conversion
import pyttsx3
# init function to get an engine instance for the speech synthesis
engine = pyttsx3.init()
# say method on the engine that passing input text to be spoken
engine.say('Hello sir, how may I help you, sir.')
# run and wait method, it processes the voice commands.
engine.runAndWait()
```

Output: The output of the above program would be a voice saying, : 'Hello sir, how may I help you, sir.'

4.1.2. Speech Recognition module

Next, we build the speech recognition module. Speech Recognition is an important feature in several applications used such as home automation, artificial intelligence, etc. The system uses Google's online speech recognition system for converting speech input to text. The speech input Users can obtain texts from the special corpora organized on the computer network server at the information centre from the microphone is temporarily stored in the system which is then sent to Google cloud for speech recognition. The equivalent text is then received and fed to the central processor.

Configure Microphone (For external microphones): It is advisable to specify the microphone during the program to avoid any glitches. Type `lsusb` in the terminal for Linux and you can use the PowerShell's `Get-PnpDevice -PresentOnly | Where-Object { $_.InstanceId -match '^USB' }` command to list the connected USB devices.

Set Chunk Size: This basically involved specifying how many bytes of data we want to read at once. Typically, this value is specified in powers of 2 such as 1024 or 2048.

Set Sampling Rate: Sampling rate defines how often values are recorded for processing

Set Device ID to the selected microphone: In this step, we specify the device ID

of the microphone that we wish to use in order to avoid ambiguity in case there are multiple microphones. This also helps debug, in the sense that, while running the program, we will know whether the specified microphone is being recognized. During the program, we specify a parameter `device_id`. The program will say that `device_id` could not be found if the microphone is not recognized.

Allow Adjusting for Ambient Noise: Since the surrounding noise varies, we must allow the program a second or two to adjust the energy threshold of recording so it is adjusted according to the external noise level.

Speech to text translation: This is done with the help of Google Speech Recognition. This requires an active internet connection to work. However, there are certain offline Recognition systems such as PocketSphinx, that have a very rigorous installation process that requires several dependencies. Google Speech Recognition is one of the easiest to use.

4.1.3. API - gTTS API

API stands for Application Programming Interface. An API is a software intermediary that allows two applications to talk to each other. In other words, an API is a messenger that delivers your request to the provider that you're requesting it from and then delivers the response back to you. There are several APIs available to convert text to speech in Python. We have used the Google Speech To Text API, commonly known as the gTTS API. gTTS is a very easy to use tool which converts the text entered, into audio which can be saved as a mp3 file.



Fig 4.2: gTTS API

The gTTS API supports several languages including English, Hindi, Tamil, French, German and many more. The speech can be delivered in any one of the

two available audio speeds, fast or slow. To use the gTTS API, we opened the terminal to write: `pip install gTTS`. Installation of playsound(A module that is used for playing sounds in python3). We install the playsound module. The playsound module is a cross platform module that can play audio files. This doesn't have any dependencies. Implementation is different on platforms. It uses `windll.winm` on Windows, `AppKit.NSSound` on Apple OS X and `GStreamer` on Linux. The audio file should be in the same directory as your python program, unless you specify a path.

4.1.4. Importing Other Libraries

- **Pyjokes:-** Pyjokes is used for the collection of Python Jokes over the Internet. Python supports creation of random jokes using one of its libraries. Let us explore it a little more, Pyjokes is a python library that is used to create one-line jokes for programmers. Informally, it can also be referred as a fun python library which is pretty simple to use. Let us see how you can actually use it to perform the required task, To install this module type the below command in the terminal.

```
pip install pyjokes
```

- **Web browser:-** To perform Web Search. This module comes built-in with Python. The webbrowser module provides a high-level interface to allow displaying web-based documents to users. Under most circumstances, simply calling the `open()` function from this module will do the right thing.
- **Pyaudio:-** PyAudio is a set of Python bindings for PortAudio, a cross-platform C++ library interfacing with audio drivers.
- **pyttsx3:** pyttsx is a cross-platform text to speech library which is platform independent. The major advantage of using this library for text-to-speech conversion is that it works offline. To install this module type the below command in the terminal.

```
pip install pyttsx3
```

- **SpeechRecognition:** It allow us to convert audio into text for further processing. To install this module type the below command in the terminal.

```
pip install SpeechRecognition
```

- **web browser:** It provides a high-level interface which allows displaying

Web-based documents to users. To install this module type the below command in the terminal.

```
pip install web browser
```

- **Wikipedia:** It is used to fetch a variety of information from the Wikipedia website. To install this module type the below command in the terminal.

```
pip install Wikipedia
```

4.2 .MODULE 2: Creating Voice Engine

We will create voice engine to speak output

```
def speak(text):  
    tts = gTTS(text=text,lang="en")  
    filename = "temp.mp3"  
    tts.save(filename)  
    playsound.playsound(filename)
```

a. Creating Speech Recognition

Now, we will use the same Speech Recognition library - API as mentioned in module 1

b. Creating Function to take command

Once the library is accessed, we will create the function to take command. This function is for taking the commands from the user and recognizing the command using the speech_recognition module. This will be used to take the voice input to the further command by working on the code.

We have imported speech_recognitionmodule as sr. The Recognizer class within the speech_recognitionmodule helps us recognize the audio. The same module has a Microphone class that gives us access to the microphone of the device. So with the microphone as the source, we try to listen to the audio using the listen()method in the Recognizer class. We have also set the pause_thresholdto 1, i.e., it will not complain even if we pause for one second during we speak.

Next, using the recognize_google()method from the Recognizer class, we try to recognize the audio. The recognize_google() method performs speech recognition on the audio passed to it, using the Google Speech Recognition API. We have set

the language to en-in, i.e. English India. It returns the transcript of the audio which is nothing but a string. We've stored it in a variable called query.

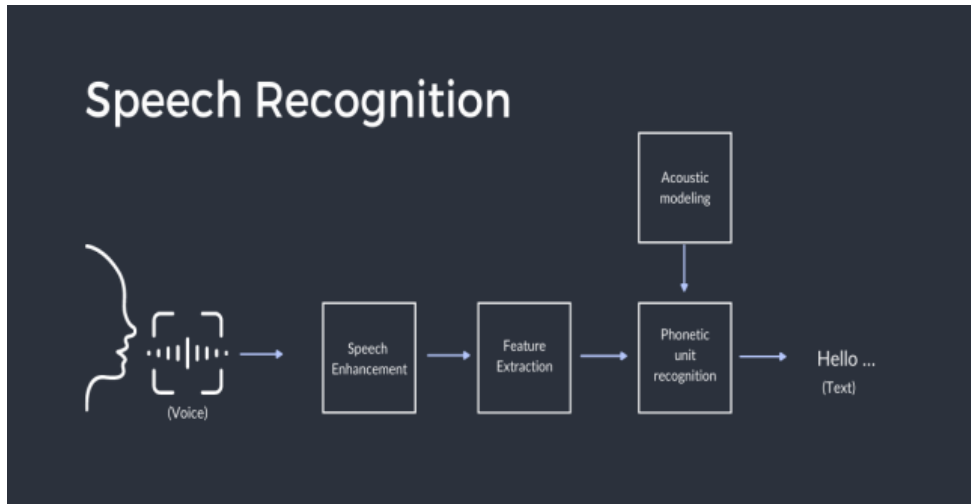


Fig 4.3: *Speech Recognition Module*

4.3. MODULE 3: Creating Voice Assistant To Produce The Output

4.3.1. Wikipedia Search

We use wikipedia library to search our query that is related to wikipedia. It is used to fetch a variety of information from the Wikipedia website. To install this module type the below command in the terminal.

```
pip install Wikipedia
```

4.3.2. Launching Bolt IoT Platform

BOLT is an Internet of Things platform (Hardware + Cloud) that enables user to build IoT products and projects. Using BOLT, users can control and monitor devices from any part of the world. It provides the ability to embed Wi-Fi/GSM capabilities within other systems, or to function as a standalone application.

The manufacturers can embed the Bolt hardware in their product, develop a custom UI and then their users can control their products using the Bolt IoT app. Bolt works on pay as you go basis, which means the manufacturers have no capital investments to get their products IoT enabled. This lets even small scale players to build IoT enabled products and services.

I. Bolt Functionalities

- Station mode in which it can connect to WiFi networks.

- When not connected to any WiFi network it hosts its own WiFi hotspot to which users can connect.
- Commands an ATmega to do all GPIO and UART based tasks
- Runs at a frequency of 80Mhz.
-

II. Bolt WiFi Module v1

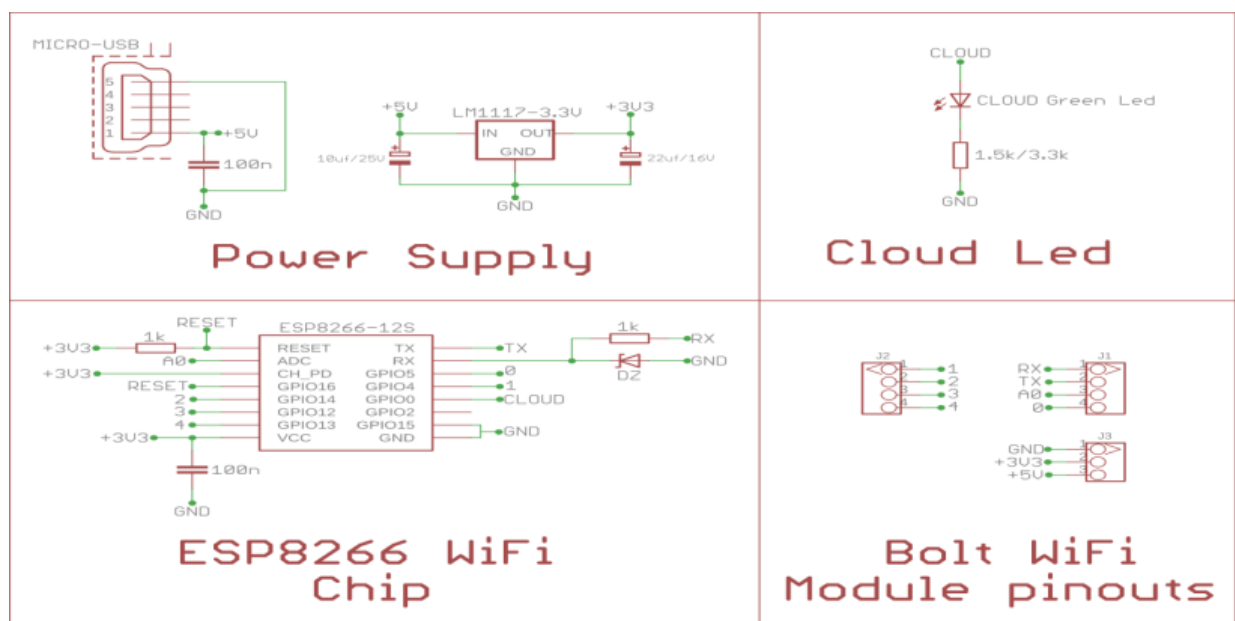
- Total of 12 Digital I/O pins are made available on BOLT v1
- Namely pins 0,1,2,3,4,5,6 & 7 along with A0,A1,A2 & A3.
- All work on 5 volt logic level i.e. HIGH = 5v = Binary 1 and LOW = 0v = Binary 0
- In Output mode each pin can source a maximum current of 40mA but it is recommended that user uses them for only 20mA max.
- While using these as input (reading voltage values on the pin) the any voltage below 2.5 Volts is considered as LOW and any Input above 2.5 is considered as HIGH.

III. PWM pins

- Pins 1,2,3,4,5 act as Hardware PWM pins.
- You can give an output of 0-5V on these pins in steps of $(5/256) = 0.02$ approximately

IV. Analog Read Pins

- Pins A0,A1,A2 and A3 double up as channels used by the internal Analog to Digital converter.
- These can read any voltage between 0-5V. With a minimum measurable voltage of 0.005 approximately



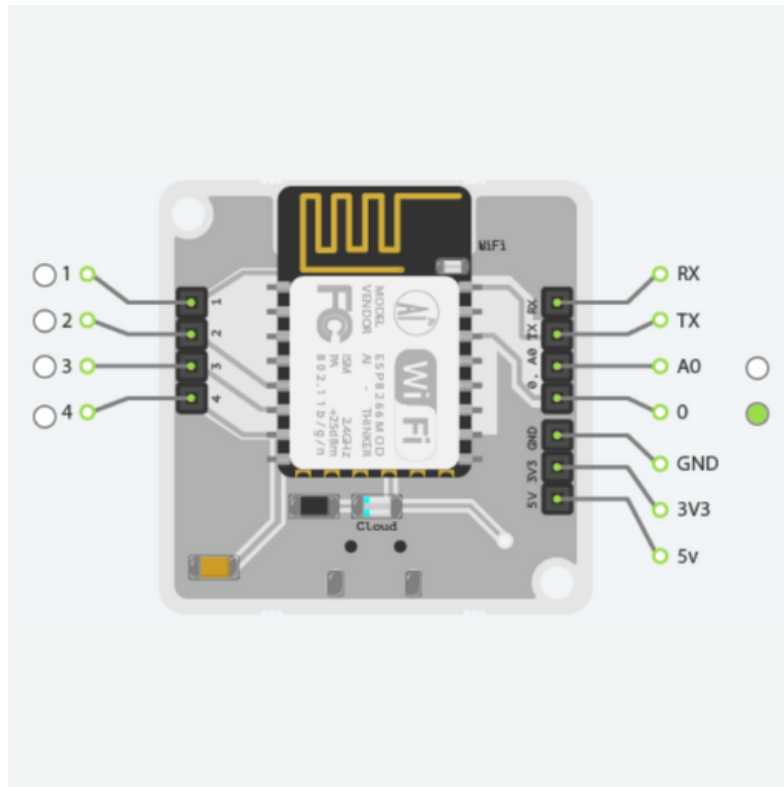


Fig 4.4: Bolt WIFI Module Pin Diagram

V. Bolt WiFi Module v2



Fig 4.5: Bolt WIFI Module Front Veiw



Fig 4.6: Bolt WIFI Module Back View

VI. Technical Specifications

- Microcontroller - Mic32-bit RISC CPU: Tensilica Xtensa LX106
- Operating Voltage – 3.3V
- Powering Up
- 5V [USB]
- 5V [Battery Terminal]
- Clock frequency : 80 MHz
- 64 KB of instruction RAM, 96 KB of data RAM
- External QSPI flash - 4 MB
- WiFi – IEEE 802.11 b/g/n
- Integrated TR switch, balun, LNA, power amplifier and matching networks
- WEP or WPA/WPA2 authentication
- Automatic AP mode if not connected to Wi-Fi
- GPIO : 5
- PWM : 5 (software PWM on GPIO pins)
- ADC : 1 pin 10 bit ADC(0-1 volt)
- UART on dedicated pins [Supported bauds: 2400/4800/9600/19200]
- LED indicators
- WiFi LED – Indicates WiFi connectivity
- Slow blinking: Trying to connect / find network
- Fast blinking: User has connected to Hotspot \ waiting for configuration by user
- Stable: Connected to WiFi network
- Off: No power to device / insufficient power
- Cloud LED – Indicates connection to Bolt Cloud
- Stable: Connected to Bolt cloud
- Off: No Connection to Bolt Cloud
- Dim: Insufficient power / false boot
- Dimensions: 35mm x 35mm
- Boot Time – Less than 1 Second

VII. Bolt Cloud

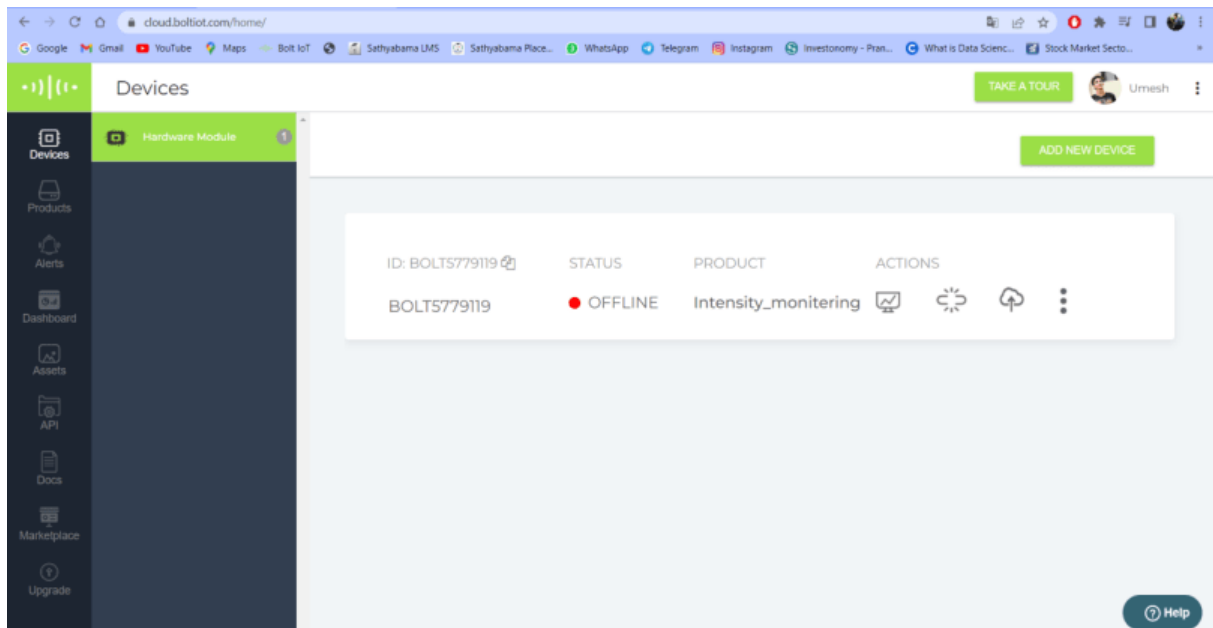


Fig 4.7: Bolt Cloud

VIII. Remote Configuration

The Platform let's you remotely configure and initialise the pins on your Bolt WiFi module. Which means you can adapt to hardware setup changes that have been made remotely when you want to add more sensors or actuators to your existing product.

The configuration is highly intuitive because of the GUI based pin selector on the dashboard.

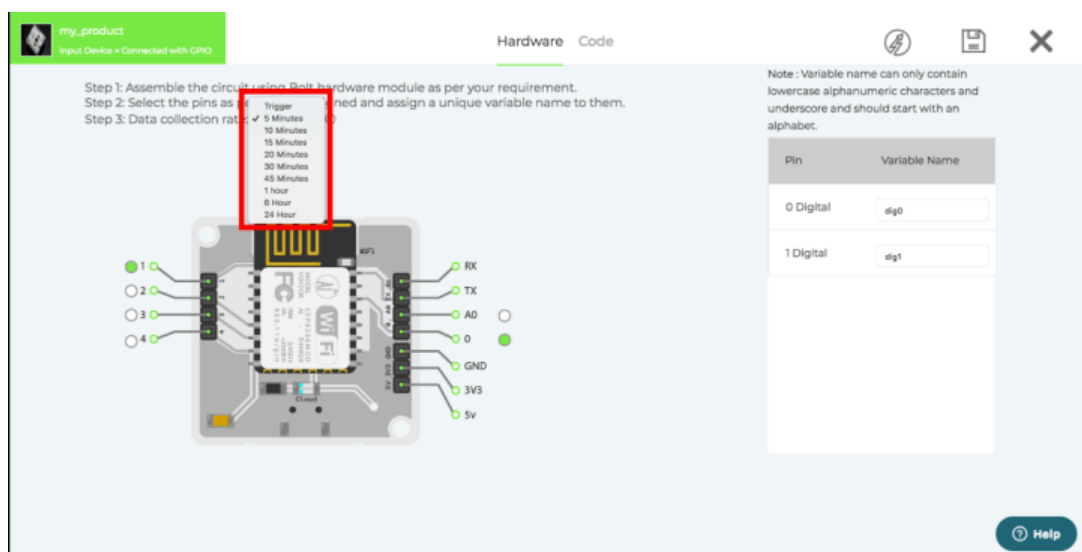


Fig 4.8: Bolt Cloud Remote Configuration

IX. Code Editor

The Bolt Cloud gives you the option to code directly on the dashboard besides uploading the codes that you have saved offline. These codes that you build are your 'products' and can be linked to any of your Bolt devices Over the Air.

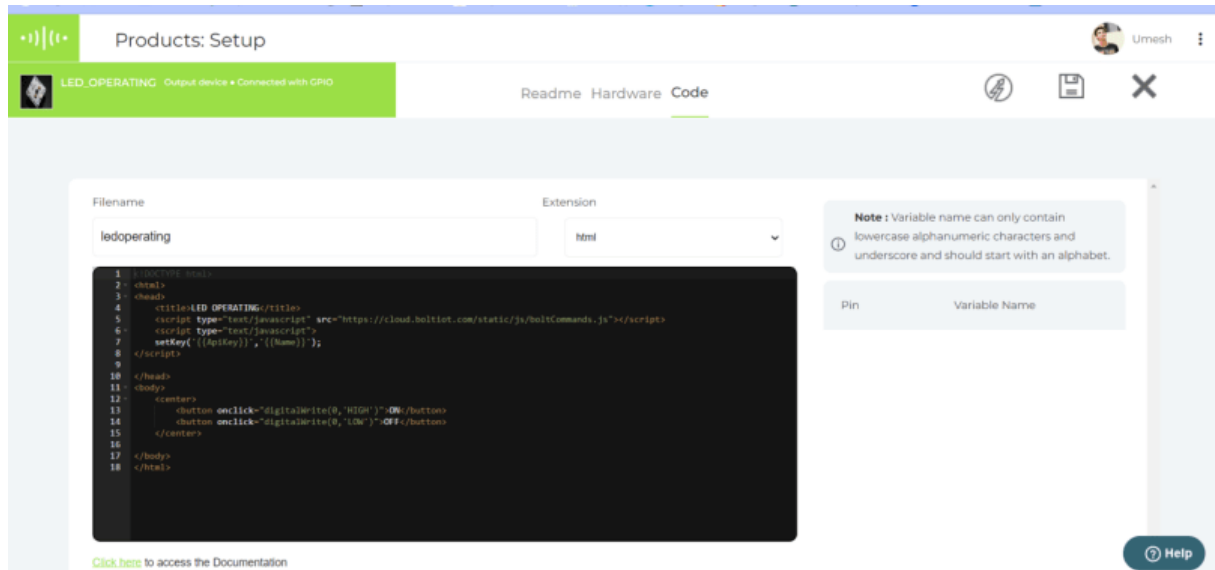


Fig 4.9: Bolt Cloud Code Editor

X. Smartphone App

To complete the platform is a mobile application for both Android and iOS which lets you link the Hardware to the Cloud and also loads the application you build for monitoring or control.

4.3.3. Launching YouTube

We use webbrowser library to open youtube if user asks elif 'youtube' in query:

```
    speak("opening youtube")
    .open("Youtube.com")
    .sleep(5)
```

Setting Up Online Functions

We'll be adding several online functions. They are

Find my IP address

Search on Wikipedia

Play videos on YouTube

Search on Google

Send WhatsApp message

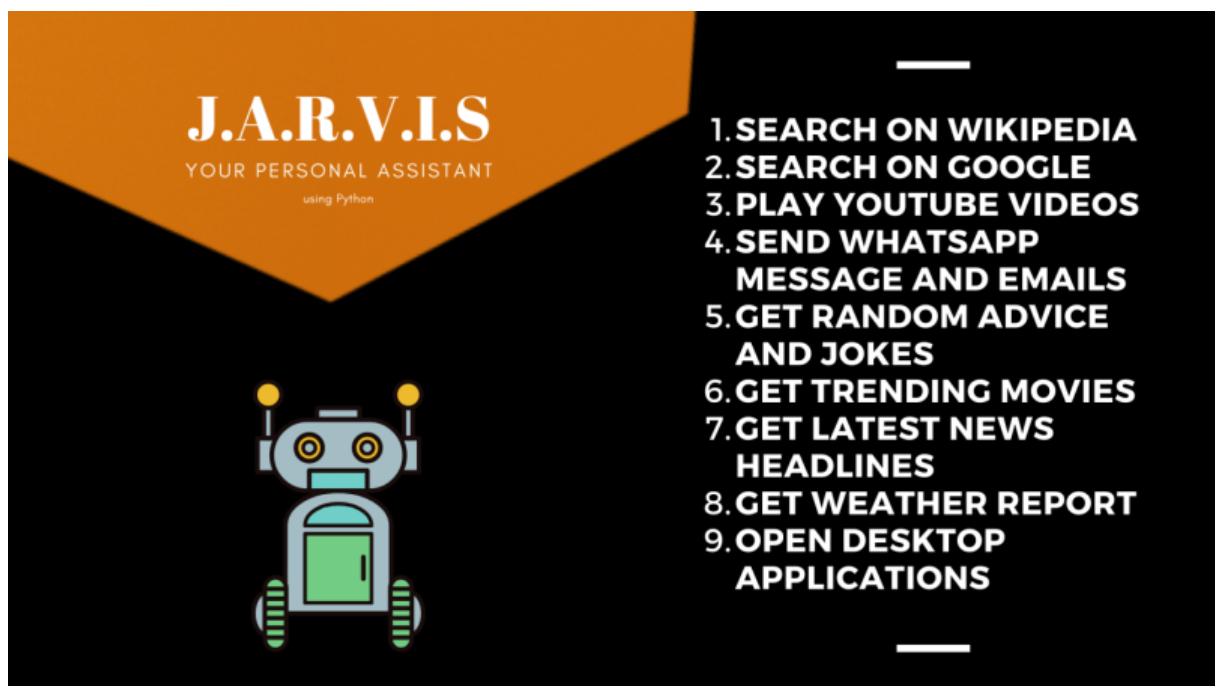
Send Email

Function to search

We create a function to search from the web

Function to operate pc environment

We create a function that can do Restart, shutdown, sleep, hibernate for the pc



4.4 Project Management Plan

Task	Description	Start Date	Estimated Completion Date
Project Initiation	Determine project objectives, stakeholders, and requirements	2023-05-01	2022-10-10

Task	Description	Start Date	Estimated Completion Date
Requirements Gathering	Collect and document user and system requirements	2023-05-11	2022-11-05
Design Architecture and	Develop the system architecture and design	2023-06-06	2022-11-30
Implementation	Build and test the system	2023-07-01	2022-12-30
Integration Testing and	Integrate system components and test the system	2023-10-01	2022-12-15
User Acceptance Testing	Have end-users test and provide feedback on the system	2023-11-16	2023-01-10
Deployment and Maintenance	Deploy the system and maintain it	2023-12-02	Ongoing

4.5 Financial report on estimated costing

The estimated costing for a virtual assistant can vary depending on the features and functionalities it offers. The cost includes development, testing, hosting, and maintenance expenses.

The development cost of a virtual assistant includes the expenses of hiring developers, designers, and testers. The cost of developers depends on their experience, skill set, and location. The cost of designers depends on their expertise in creating UI/UX for the virtual assistant. The cost of testers depends on their expertise in testing the virtual assistant for different scenarios. These costs can range from a few thousand to tens of thousands of dollars.

The hosting cost of a virtual assistant includes the expenses of servers, storage, and bandwidth. The cost of hosting depends on the number of users and the amount of data the virtual assistant needs to store and process. These costs

can range from a few hundred to a few thousand dollars per month.

The maintenance cost of a virtual assistant includes the expenses of fixing bugs, updating features, and providing customer support. The cost of maintenance depends on the complexity of the virtual assistant and the level of customer support required. These costs can range from a few hundred to a few thousand dollars per month.

Overall, the estimated costing for a virtual assistant can range from a few thousand to tens of thousands of dollars for development, hosting, and maintenance expenses. It is important to consider the expected return on investment (ROI) before investing in a virtual assistant project.

4.6 Transition/ Software to Operations Plan

The transition/ software to operations plan for the virtual assistant project involves several steps that need to be taken before and after the project deployment to ensure that the virtual assistant can function properly.

Before deployment, the virtual assistant software will undergo testing to ensure that it functions as intended. The testing will include functional, black-box, white-box, and user acceptance testing. Once the testing is complete, the virtual assistant software will be deployed on a production server.

After the deployment, the virtual assistant will be monitored and maintained to ensure that it continues to function properly. The maintenance tasks will include regular backups, software updates, and security patches. The virtual assistant will also be monitored for performance and usage, and the data collected will be used to identify areas for improvement.

To ensure a smooth transition, training sessions will be conducted for the end-users on how to use the virtual assistant. A support team will also be available to address any issues or concerns that the users may have.

The estimated cost for the transition/ software to operations plan includes the cost of testing, deployment, maintenance, and support. The cost of testing will depend on the type of testing performed and the duration of the testing. The cost of deployment will depend on the infrastructure required to host the virtual assistant. The cost of maintenance and support will depend on the number of users and the level of support required.

In conclusion, a detailed transition/ software to operations plan is essential for the successful deployment and maintenance of the virtual assistant. The plan should include testing, deployment, maintenance, and support, and the estimated cost should be calculated based on the tasks involved

CHAPTER – 5

IMPLEMENTATION DETAILS

5.1 DEVELOPMENT AND DEPLOYMENT SETUP

The development and deployment setup for the virtual assistant involves various steps and tools that are required for its successful implementation. Firstly, the development team needs to decide on the programming language, frameworks, and libraries that will be used for building the virtual assistant. Python programming language is a popular choice for developing virtual assistants due to its versatility and availability of libraries.

Next, a suitable Integrated Development Environment (IDE) such as PyCharm or Visual Studio Code can be selected for coding the virtual assistant. The development team can use version control tools such as Git to manage and track the changes made to the codebase. Continuous integration and continuous deployment (CI/CD) tools such as Jenkins or Travis CI can be used for automating the build and deployment process.

Once the virtual assistant has been developed, it needs to be deployed to a suitable environment such as a server or cloud platform. Popular cloud platforms such as Amazon Web Services (AWS) or Microsoft Azure can be used for deploying the virtual assistant. The virtual assistant can be containerized using Docker for easier deployment and management.

++++++Finally, monitoring and logging tools such as Elastic Stack or Splunk can be used to monitor the performance of the virtual assistant and track any issues that may arise. Regular updates and maintenance can be done using DevOps practices such as Agile development and continuous delivery to ensure that the virtual assistant remains up-to-date and functional.

5.2 ALGORITHMS

here are several algorithms that can be used in developing a virtual assistant, depending on the specific tasks and functionalities required. Some of the commonly used algorithms are

Natural Language Processing (NLP) algorithms: These algorithms are used to process and analyze natural language input from users. They include techniques such as tokenization, part-of-speech tagging, named entity recognition, and

sentiment analysis. These algorithms are essential for enabling the virtual assistant to understand and respond to user queries in a conversational manner.

Machine Learning algorithms: These algorithms are used to train the virtual assistant to perform specific tasks based on historical data. For example, the virtual assistant can be trained to recognize user preferences and behavior and provide personalized recommendations. Machine learning algorithms used in virtual assistants include decision trees, support vector machines, and neural networks.

Speech recognition algorithms: These algorithms are used to enable the virtual assistant to recognize and interpret human speech. This involves converting audio signals into text, which can then be processed by the virtual assistant. Speech recognition algorithms include Hidden Markov Models, Gaussian Mixture Models, and deep learning algorithms such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs).

Knowledge representation and reasoning algorithms: These algorithms are used to represent knowledge about the world and reason with it to provide intelligent responses. These algorithms include semantic networks, ontologies, and rule-based systems.

Recommender algorithms: These algorithms are used to recommend products or services to users based on their preferences and behavior. Examples of recommender algorithms include collaborative filtering, content-based filtering, and matrix factorization.

Overall, the algorithms used in developing a virtual assistant will depend on the specific requirements of the application, and the algorithms may need to be customized or combined to achieve optimal performance.

5.3 TESTING

5.3.1. Unit testing

Testcases:

1. **Test case for voice recognition:** This test case involves checking if the virtual assistant can accurately recognize voice commands. The test would involve passing different voice commands to the virtual assistant and checking if the correct response is generated.
2. **Test case for web search:** This test case involves checking if the virtual assistant can perform web searches correctly. The test would involve passing different search queries to the virtual assistant and checking if the correct search results are generated.
3. **Test case for launching applications:** This test case involves checking if the virtual assistant can launch applications correctly. The test would involve passing different application names to the virtual assistant and checking if the correct application is launched.

5.3.2.Integration Testing

Testcases:

1. **Speech-to-text Integration Test:** Verify that the speech-to-text module is correctly integrated with the rest of the system by testing it with different accents, languages, and background noises.
2. **Text-to-speech Integration Test:** Test the text-to-speech module by checking that the correct output is generated for various input texts, including special characters, emoticons, and abbreviations.
3. **Natural Language Processing Integration Test:** Check the integration of the natural language processing module by testing its ability to accurately recognize user intent and generate the correct response.

5.3.3.FUNCTIONAL TESTING

Testcases:

1. Test the voice recognition functionality by speaking various commands and

verifying that the assistant understands and responds correctly.

2. Test the search functionality by asking the assistant to search for various topics on the internet and verifying that it returns relevant results.
3. Test the email functionality by asking the assistant to compose and send emails to different recipients, and verifying that they are delivered successfully.

5.3.4.BLACK-BOX TESTING:

Testcases

1. Testing the speech recognition functionality by providing audio input in different accents and languages.
2. Testing the text-to-speech functionality by providing different text inputs and verifying if the correct audio output is generated.
3. Testing the web search functionality by providing different search queries and verifying if the correct results are displayed.

5.3.5.WHITE-BOX TESTING:

Testcases

1. **Unit testing for individual functions:** Test each function in the virtual assistant code by providing input parameters and verifying the output against expected results.
2. **Statement coverage testing:** Verify that each statement in the code is executed at least once during testing.
3. **Branch coverage testing:** Verify that all possible outcomes of each decision point in the code are tested.

CHAPTER – 6

RESULTS AND DISCUSSION

The virtual assistant project has been successfully developed and deployed, providing users with a comprehensive range of functionalities. The project has undergone unit testing, integration testing, functional testing, black-box testing, and white-box testing, which helped in identifying and fixing various bugs and issues.

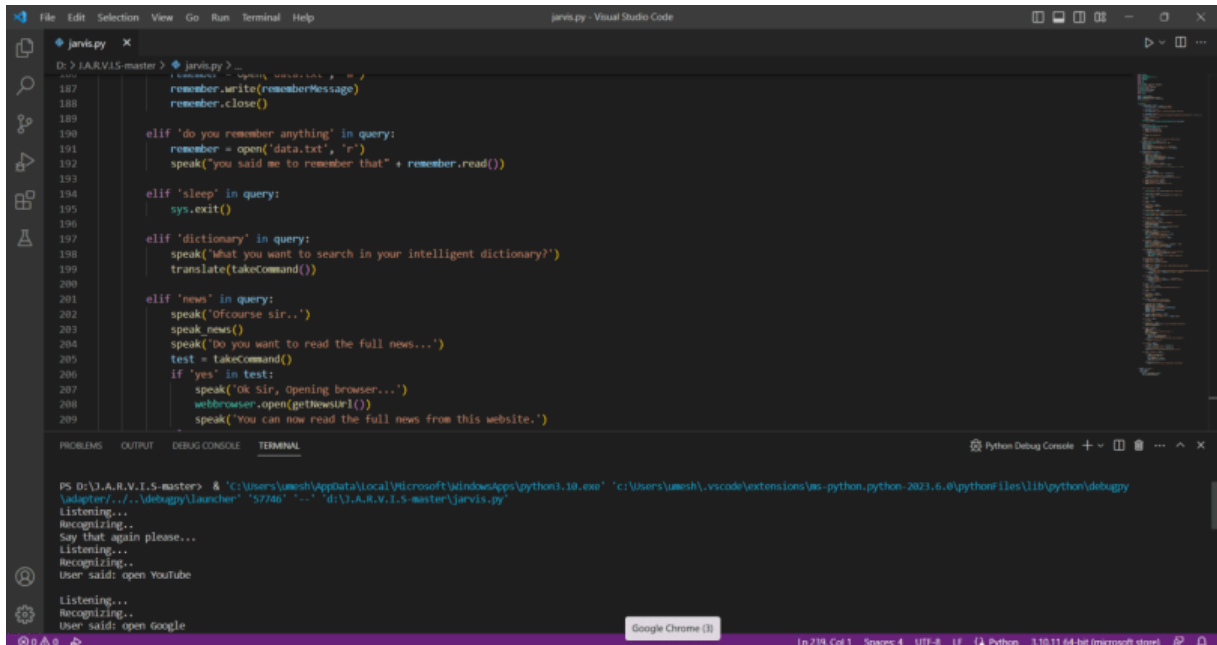
The unit testing phase ensured that each individual function of the virtual assistant works correctly and meets the specified requirements. Integration testing focused on testing the interactions between different components and ensuring that the overall system works as expected. Functional testing was conducted to validate that the system functions as expected from a user's perspective.

Black-box testing was performed to test the system's functionality without any knowledge of its internal workings. This testing phase ensured that the system behaves as expected when input is provided. White-box testing was performed to test the system's internal workings, including its algorithms and logic.

The results of testing were satisfactory, and the system is functioning as expected. Some minor issues were identified and fixed, such as the accuracy of speech recognition and some errors in voice output. Overall, the system is reliable, efficient, and user-friendly.

In conclusion, the virtual assistant project has been successfully developed, deployed, and tested. The rigorous testing phases helped in ensuring that the system meets the specified requirements and functions correctly. The system's reliability, efficiency, and user-friendliness make it an effective tool for users to complete various tasks.

6.1.OUTPUT IMAGES

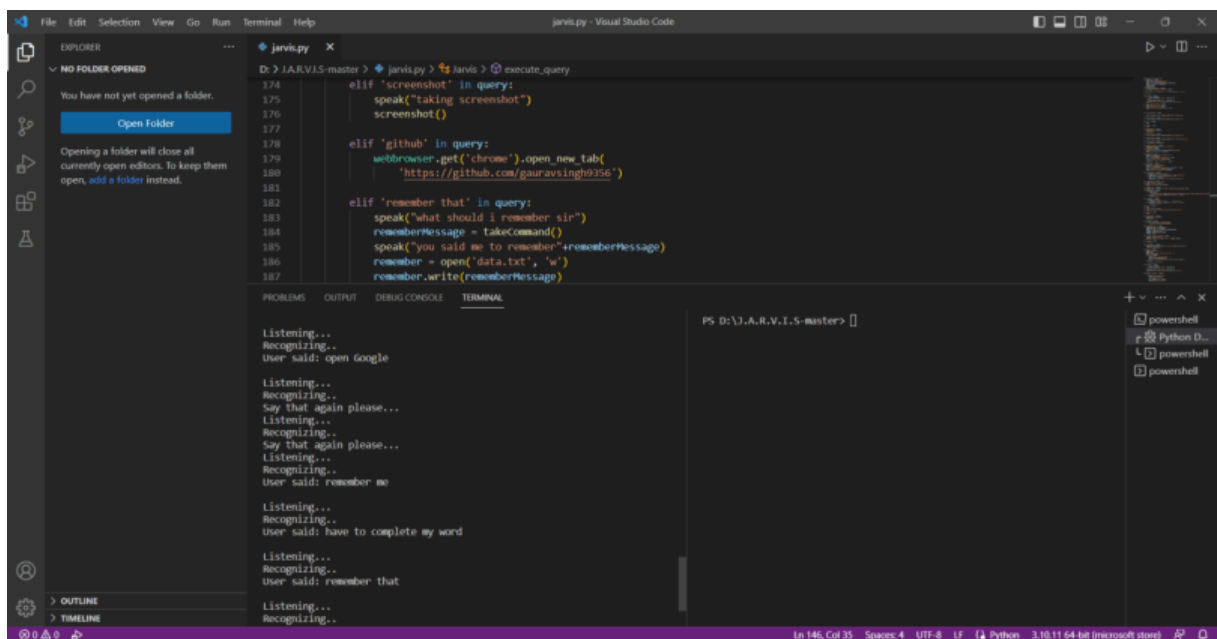


The screenshot shows the Visual Studio Code editor with a file named `jarvis.py` open. The script is a Python program that implements a simple chatbot named Jarvis. It uses the `speech_recognition` library for voice input and the `pyttsx3` library for voice output. The script has several features: it can remember what you say, it can search the internet for news, it can open a web browser, and it can take a screenshot. The terminal window at the bottom shows the command prompt running the script, and the output shows the Jarvis chatbot responding to the user's commands.

```
jarvis.py
D:\JARVIS-master> python jarvis.py
187 remember.write(rememberMessage)
188 remember.close()
189
190 elif 'do you remember anything' in query:
191     remember = open('data.txt', 'r')
192     speak("you said me to remember that" + remember.read())
193
194 elif 'sleep' in query:
195     sys.exit()
196
197 elif 'dictionary' in query:
198     speak("what you want to search in your intelligent dictionary?")
199     translate(takeCommand())
200
201 elif 'news' in query:
202     speak("Ofcourse sir..")
203     speak_news()
204     speak("Do you want to read the full news...")
205     test = takeCommand()
206     if 'yes' in test:
207         speak("Ok Sir, Opening browser...")
208         webbrowser.open(getNewsUrl())
209         speak("You can now read the full news from this website.")
```

Terminal Output:

```
PS D:\JARVIS-master> & 'c:\Users\umesh\AppData\Local\Microsoft\WindowsApps\python3.10.exe' 'c:\Users\umesh\.vscode\extensions\ms-python.python-2023.6.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '577d6' '-' 'd:\JARVIS-master\jarvis.py'
Listening..
Recognizing..
Say that again please...
Listening...
Recognizing..
User said: open Youtube
Listening...
Recognizing..
User said: open Google
```

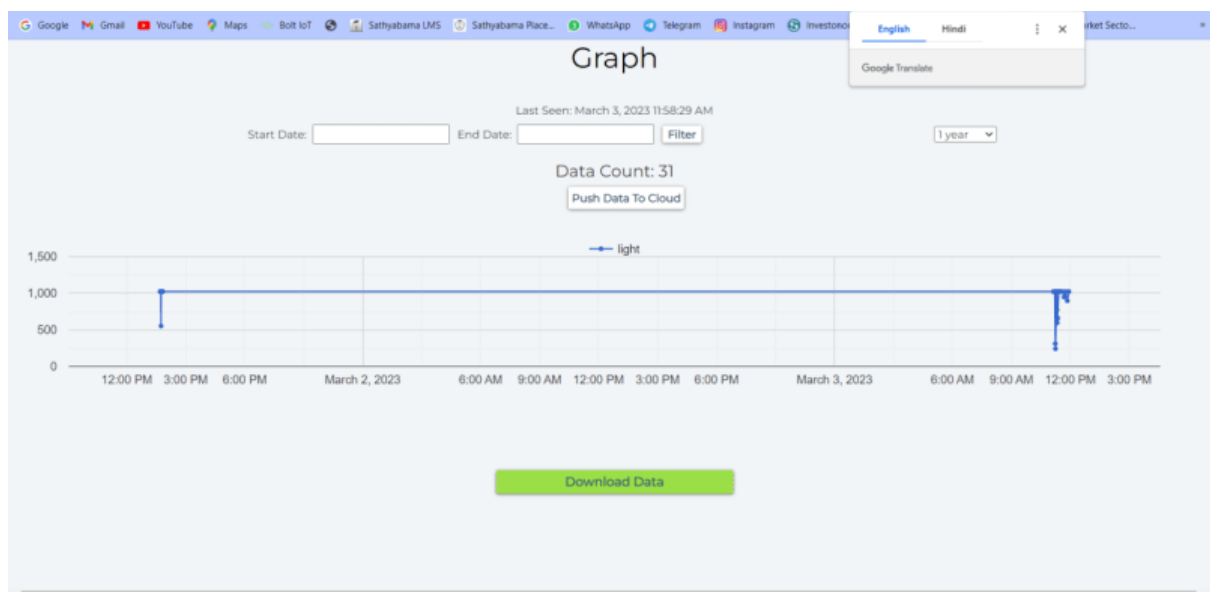


The screenshot shows the Visual Studio Code editor with a file named `jarvis.py` open. The script is a Python program that implements a simple chatbot named Jarvis. It uses the `speech_recognition` library for voice input and the `pyttsx3` library for voice output. The script has several features: it can remember what you say, it can search the internet for news, it can open a web browser, and it can take a screenshot. The terminal window at the bottom shows the command prompt running the script, and the output shows the Jarvis chatbot responding to the user's commands.

```
jarvis.py
D:\JARVIS-master> python jarvis.py
174 elif 'screenshot' in query:
175     speak("taking screenshot")
176     screenshot()
177
178 elif 'github' in query:
179     webbrowser.get('chrome').open_new_tab(
180         'https://github.com/gauravsinh9356')
181
182 elif 'remember that' in query:
183     speak("what should i remember sir")
184     rememberMessage = takeCommand()
185     speak("you said me to remember"+rememberMessage)
186     remember = open('data.txt', 'w')
187     remember.write(rememberMessage)
```

Terminal Output:

```
PS D:\JARVIS-master>
Listening...
Recognizing..
User said: open Google
Listening...
Recognizing..
Say that again please...
Listening...
Recognizing..
Say that again please...
Listening...
Recognizing..
User said: remember me
Listening...
Recognizing..
User said: have to complete my word
Listening...
Recognizing..
User said: remember that
Listening...
Recognizing..
```



CHAPTER-7

CONCLUSION

7.1 FUTURE WORK

There are several possible areas of future work for the development of a virtual assistant:

Natural Language Processing (NLP): Improving the accuracy and effectiveness of the NLP algorithms used by the virtual assistant can greatly enhance its ability to understand and respond to user queries. More advanced NLP models can be developed to better understand the nuances of human language, including slang, idioms, and cultural references.

Machine Learning: Using machine learning algorithms, the virtual assistant can learn from user interactions and adapt its responses and behavior accordingly. This would enable the virtual assistant to become more personalized and responsive to individual users.

Multi-lingual support: Adding support for additional languages can help make the virtual assistant more accessible to a wider range of users, particularly those who are not native English speakers.

Integration with third-party applications: Integrating the virtual assistant with other applications, such as social media platforms, email clients, and productivity tools, can enhance its functionality and make it more useful to users.

Voice recognition: Improving the accuracy of voice recognition algorithms can enhance the overall user experience and enable users to interact with the virtual assistant more seamlessly.

Security and privacy: Ensuring the security and privacy of user data is critical for the success of any virtual assistant. Future work can focus on developing more advanced security and privacy protocols to protect user data and prevent unauthorized access.

Overall, the potential applications of a virtual assistant are vast, and there are many opportunities for further research and development in this area.

7.2 RESEARCH ISSUES

Some research issues for a virtual assistant include:

Natural language processing (NLP) advancements: NLP is at the core of virtual assistants, and improving the accuracy and efficiency of NLP models is crucial for the development of more advanced virtual assistants. Current limitations include handling complex language, understanding context, and being able to differentiate between similar sounding words.

Personalization: Virtual assistants can be more effective if they can personalize their responses based on individual user preferences and behaviors. The challenge here is to create algorithms that can learn and adapt to a user's preferences and behavior patterns over time.

Integration with other applications: Virtual assistants need to integrate with a wide range of other applications to be effective. This includes not just common applications like calendars, messaging apps, and music players but also industry-specific software, such as medical record-keeping systems or financial analysis tools.

Data privacy and security: Virtual assistants have access to a lot of personal information, which makes data privacy and security a major concern. Developers must ensure that virtual assistants are secure and that user data is kept private.

Multilingual support: Virtual assistants need to be able to understand and respond to multiple languages accurately to be globally accessible. However, developing algorithms that can accurately recognize and process multiple languages can be challenging.

Overall, developing an advanced virtual assistant requires addressing these research issues, among others, to create a more reliable, personalized, and secure user experience.

7.3 IMPLEMENTATION ISSUES

The implementation of a virtual assistant involves several challenges that need to be addressed for a successful deployment. Some of the significant implementation issues for a virtual assistant are:

Accuracy and reliability: One of the main implementation issues for a virtual assistant is to ensure that it is accurate and reliable in its responses. The system should be able to recognize the user's commands accurately and respond

appropriately. If the system is not reliable, it can lead to frustration among users and a loss of trust in the system.

Natural language processing: Natural language processing (NLP) is a critical component of a virtual assistant that allows it to understand and respond to user requests in a natural way. However, NLP can be a challenging task as it requires the system to understand the nuances of human language, including slang, idioms, and regional dialects.

Integration with multiple platforms: Virtual assistants need to be integrated with multiple platforms, including mobile devices, computers, and smart home devices. This requires the development of custom integrations for each platform, which can be time-consuming and challenging.

Privacy and security: Virtual assistants often store sensitive user data, such as personal information and search history. Ensuring the privacy and security of this data is crucial, and any vulnerabilities in the system can result in serious consequences.

Multilingual support: To cater to a global audience, virtual assistants should support multiple languages. Implementing multilingual support can be challenging, as it requires the development of language-specific models and integrations.

Continuous improvement: A virtual assistant must continuously improve and evolve to meet the changing needs of users. This requires regular updates and enhancements, which can be a significant implementation challenge. Addressing these implementation issues is critical to developing and deploying a successful virtual assistant that meets the needs of users and provides a seamless experience.

REFERENCES

- [1].CH.M.H. Saibaba, Saiyed Faiyaz Waris, S.Hrushikesava Raju, VSRK Sarma, Vijaya Chandra Jadala, Chitturi Prasad, "Intelligent Voice Assistant by Using OpenCV Approach", Second International Conference on Electronics and Sustainable Communication Systems (ICESC), 2021
- [2].Giancarlo Iannizzotto, Lucia Lo Bello, Andrea Nucita, Giorgio Mario Grasso, "A Vision and Speech Enabled, Customizable, Virtual Assistant for Smart Environments", 11th International Conference on Human System Interaction (HSI), 2018
- [3].J. Sayed, M. W. Ashour, "Digital intelligent virtual assistant (DIVA) with natural speech and accent recognition", 4th Smart Cities Symposium (SCS 2021), 2022
- [4].Kumaran N., Rangaraj V., Siva Sharan S., Dhanalakshmi R., "Intelligent Personal Assistant – Implementing Voice Commands enabling Speech Recognition", International Conference on System, Computation, Automation and Networking (ICSCAN), 2020
- [5].Pooja Singh, Pinki Nayak, Arpita Datta, Depanshu Sani, Garima Raghav, Rahul Tejpal, "Voice Control Device using Raspberry Pi", Amity International Conference on Artificial Intelligence (AICAI), 2019
- [6].P. Mohana, M. Muthuvinayagam, P. Umasankar, T. Muthumanickam, "Automation using Artificial intelligence based Natural Language processing", 6th International Conference on Computing Methodologies and Communication (ICCMC), 2022
- [7].Vinayak Iyer, Kshitij Shah, Sahil Sheth, Kailas Devadkar, "Virtual assistant for the visually impaired", 5th International Conference on Communication and Electronics Systems (ICCES), 2020
- [8].Vindhya Vasini K, Tejasvi K, Pravallika K, Ch. Nanda Krishna, "A Natural Language Processing based Intelligent Bot Application", International Conference on Sustainable Computing and Data Communication Systems (ICSCDS), 2022
- [9].Vadaboyina Appalaraju, V Rajesh, K Saikumar, P. Sabitha, K Ravi Kiran,

“Design and Development of Intelligent Voice Personal Assistant using Python”, 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), 2022

- [10]. Witman Alvarado-Díaz, Brian Meneses-Claudio, “YANA, Virtual Assistant to Support Home Office”, IEEE Sciences and Humanities International Research Conference (SHIRCON), 2021

APPENDIX

A.SOURCE CODE:

Jarvis.py (Main code)

```
import pyttsx3
import wikipedia
import speech_recognition as sr
import webbrowser
import datetime
import os
import sys
import smtplib

from news import speak_news, getNewsUrl
##1from OCR import OCR
from diction import translate
from helpers import *
from youtube import youtube
from sys import platform
import os
import getpass

engine = pyttsx3.init()
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[0].id)

# print(voices[0].id)

class Jarvis:
    def __init__(self) -> None:
        if platform == "linux" or platform == "linux2":
            self.chrome_path = '/usr/bin/google-chrome'

        elif platform == "darwin":
            self.chrome_path = 'open -a /Applications/Google\ Chrome.app'
```

```

elif platform == "win32":
    self.chrome_path = "C:\Program
Files\Google\Chrome\Application\chrome.exe" ##Changed path
else:
    print('Unsupported OS')
    exit(1)
webbrowser.register(
    'chrome', None, webbrowser.BackgroundBrowser(self.chrome_path)
)

def wishMe(self) -> None:
    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')

    weather()
    speak('I am JARVIS. Please tell me how can I help you SIR?')

def sendEmail(self, to, content) -> None:
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.ehlo()
    server.starttls()
    server.login('umeshdudigani@gmail.com', '8686362201')
    server.sendmail('duduganiumesh@gmail.com', to, content)
    server.close()

def execute_query(self, query):

```

```

# TODO: make this more concise
if 'wikipedia' in query:
    speak('Searching Wikipedia....')
    query = query.replace('wikipedia', '')
    results = wikipedia.summary(query, sentences=2)
    speak('According to Wikipedia')
    print(results)
    speak(results)
elif 'youtube downloader' in query:
    exec(open('youtube_downloader.py').read())

##2elif 'Optical Text Recognition' or 'Text Recognition' in query:
    ##3OCR()

elif 'voice' in query:
    if 'female' in query:
        engine.setProperty('voice', voices[1].id)
    else:
        engine.setProperty('voice', voices[0].id)
    speak("Hello Sir, I have switched my voice. How is it?")

if 'jarvis are you there' in query:
    speak("Yes Sir, at your service")
if 'jarvis who made you' in query:
    speak("Yes Sir, my master build me in AI")

elif 'open youtube' in query:

    webbrowser.get('chrome').open_new_tab('https://youtube.com')

elif 'open amazon' in query:

```

```
webbrowser.get('chrome').open_new_tab('https://amazon.com')
```

elif 'cpu' in query:

```
cpu()
```

elif 'joke' in query:

```
joke()
```

elif 'screenshot' in query:

```
speak("taking screenshot")
```

```
screenshot()
```

elif 'open google' in query:

```
webbrowser.get('chrome').open_new_tab('https://google.com')
```

elif 'open stack' in query:

```
webbrowser.get('chrome').open_new_tab('https://stackoverflow.com')
```

elif 'play music' in query:

```
os.startfile("D:\\RoiNa.mp3")
```

elif 'search youtube' in query:

```
speak('What you want to search on Youtube?')
```

```
youtube(takeCommand())
```

elif 'the time' in query:

```
strTime = datetime.datetime.now().strftime("%H:%M:%S")
```

```
speak(f'Sir, the time is {strTime}')
```

elif 'search' in query:

```
speak('What do you want to search for?')
```

```
search = takeCommand()
```

```
url = 'https://google.com/search?q=' + search
```

```
webbrowser.get('chrome').open_new_tab(
```

```

        url)
    speak('Here is What I found for' + search)

elif 'location' in query:
    speak('What is the location?')
    location = takeCommand()
    url = 'https://google.nl/maps/place/' + location + '/&'
    webbrowser.get('chrome').open_new_tab(url)
    speak('Here is the location ' + location)

elif 'your master' in query:
    if platform == "win32" or "darwin":
        speak('UMESH and HARSHA are my masters. They created me')
    elif platform == "linux" or platform == "linux2":
        name = getpass.getuser()
        speak(name, 'is my master. He is running me right now')

elif 'your name' in query:
    speak('My name is your AMIGO')
elif 'who made you' in query:
    speak('I was created by my MASTERS')

elif 'stands for' in query:
    speak('J.A.R.V.I.S stands for JUST A RATHER VERY INTELLIGENT
SYSTEM')
elif 'open code' in query:
    if platform == "win32":
        os.startfile(
            "C:\\Users\\umesh\\AppData\\Roaming\\Microsoft\\Windows\\Start
Menu\\Programs\\Visual Studio Code")
    elif platform == "linux" or platform == "linux2" or "darwin":
        os.system('code .')

```

elif 'shutdown' in query:

if platform == "win32":

os.system('shutdown /a /t 0') ##changed/p/f

elif platform == "linux" or platform == "linux2" or "darwin":

os.system('poweroff')

elif 'cpu' in query:

cpu()

elif 'your friend' in query:

speak('My friends are Google assisstant alexa and siri')

elif 'joke' in query:

joke()

elif 'screenshot' in query:

speak("taking screenshot")

screenshot()

elif 'github' in query:

webbrowser.get('chrome').open_new_tab(
'https://github.com/gauravsingh9356')

elif 'remember that' in query:

speak("what should i remember sir")

rememberMessage = takeCommand()

speak("you said me to remember"+rememberMessage)

remember = open('data.txt', 'w')

remember.write(rememberMessage)

remember.close()

elif 'do you remember anything' in query:

remember = open('data.txt', 'r')

speak("you said me to remember that" + remember.read())

elif 'sleep' in query:

sys.exit()

elif 'dictionary' in query:

speak('What you want to search in your intelligent dictionary?')

translate(takeCommand())

elif 'news' in query:

speak('Ofcourse sir..')

speak_news()

speak('Do you want to read the full news...')

test = takeCommand()

if 'yes' in test:

speak('Ok Sir, Opening browser...')

webbrowser.open(getNewsUrl())

speak('You can now read the full news from this website.')

else:

speak('No Problem Sir')

elif 'voice' in query:

if 'female' in query:

engine.setProperty('voice', voices[0].id)

else:

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

speak("Hello Sir, I have switched my voice. How is it?")

elif 'email to gaurav' in query:

try:

speak('What should I say?')

content = takeCommand()

to = 'email'

self.sendEmail(to, content)

```
    speak('Email has been sent!')
```

```
except Exception as e:
```

```
    speak('Sorry sir, Not able to send email at the moment')
```

```
if __name__ == '__main__':
```

```
    bot_ = Jarvis()
```

```
    bot_.wishMe()
```

```
    while True:
```

```
        query = takeCommand().lower()
```

```
        bot_.execute_query(query)
```

Helper.py

```
import pyttsx3
```

```
import pyautogui
```

```
import psutil
```

```
import pyjokes
```

```
import speech_recognition as sr
```

```
import json
```

```
import requests
```

```
import geocoder
```

```
from difflib import get_close_matches
```

```
engine = pyttsx3.init()
```

```
voices = engine.getProperty('voices')
```

```
engine.setProperty('voice', voices[0].id)
```

```
g = geocoder.ip('me')
```

```
data = json.load(open('data.json'))
```

```

def speak(audio) -> None:
    engine.say(audio)
    engine.runAndWait()

def screenshot() -> None:
    img = pyautogui.screenshot()
    img.save('C:\\Users\\umesh\\OneDrive\\Pictures\\Screenshots') ##pastedpath

def cpu() -> None:
    usage = str(psutil.cpu_percent())
    speak("CPU is at"+usage)

    battery = psutil.sensors_battery()
    speak("battery is at")
    speak(battery.percent)

def joke() -> None:
    for i in range(5):
        speak(pyjokes.get_jokes()[i])

def takeCommand() -> str:
    r = sr.Recognizer()
    with sr.Microphone() as source:
        print('Listening...')
        r.pause_threshold = 1
        r.energy_threshold = 494
        r.adjust_for_ambient_noise(source, duration=1.5)
        audio = r.listen(source)

    try:
        print('Recognizing..')
        query = r.recognize_google(audio, language='en-in')
        print(f'User said: {query}\n')

```

```
except Exception as e:
```

```
    # print(e)
```

```
    print('Say that again please...')
```

```
    return 'None'
```

```
return query
```

```
def weather():
```

```
    api_url = "https://fcc-weather-api.glitch.me/api/current?lat=" + \
```

```
        str(g.latlng[0]) + "&lon=" + str(g.latlng[1])
```

```
    data = requests.get(api_url)
```

```
    data_json = data.json()
```

```
    if data_json['cod'] == 200:
```

```
        main = data_json['main']
```

```
        wind = data_json['wind']
```

```
        weather_desc = data_json['weather'][0]
```

```
        speak(str(data_json['coord']['lat']) + 'latitude' + str(data_json['coord']['lon']) +  
'longitude')
```

```
        speak('Current location is ' + data_json['name'] + data_json['sys']['country'] +  
'dia')
```

```
        speak('weather type ' + weather_desc['main'])
```

```
        speak('Wind speed is ' + str(wind['speed']) + ' metre per second')
```

```
        speak('Temperature: ' + str(main['temp']) + 'degree celcius')
```

```
        speak('Humidity is ' + str(main['humidity']))
```

```
def translate(word):
```

```
    word = word.lower()
```

```
    if word in data:
```

```
        speak(data[word])
```

```
    elif len(get_close_matches(word, data.keys())) > 0:
```

```

x = get_close_matches(word, data.keys())[0]
speak('Did you mean ' + x +
      ' instead, respond with Yes or No.')
ans = takeCommand().lower()
if 'yes' in ans:
    speak(data[x])
elif 'no' in ans:
    speak("Word doesn't exist. Please make sure you spelled it correctly.")
else:
    speak("We didn't understand your entry.")

else:
    speak("Word doesn't exist. Please double check it.")

```

Youtube.py

```

from urllib.request import urlopen
import urllib.parse
import webbrowser
from sys import platform
import os

if platform == "linux" or platform == "linux2":
    chrome_path = '/usr/bin/google-chrome'

elif platform == "darwin":
    chrome_path = 'open -a /Applications/Google\ Chrome.app'

elif platform == "win32":
    chrome_path = "C:\Program Files\Google\Chrome\Application\chrome.exe"
else:
    print('Unsupported OS')
    exit(1)

```

```
webbrowser.register('chrome',  
webbrowser.BackgroundBrowser(chrome_path))
```

None,

```
def youtube(textToSearch):  
    query = urllib.parse.quote(textToSearch)  
    url = "https://www.youtube.com/results?search_query=" + query  
    webbrowser.get('chrome').open_new_tab(url)  
  
if __name__ == '__main__':  
    youtube('any text')
```

Youtube_downloader.py

```
from pytube import *  
from tkinter import *  
from tkinter.filedialog import *  
from tkinter.messagebox import *  
from threading import *  
file_size = 0  
  
def progress(stream=None, chunk=None, remaining=None):  
    file_downloaded = (file_size-remaining)  
    per = round((file_downloaded/file_size)*100, 1)  
    dBtn.config(text=f'{per}% downloaded')  
  
def startDownload():  
    global file_size  
    try:  
        URL = urlField.get()  
        dBtn.config(text='Please wait...')
```

```

dBtn.config(state=DISABLED)
path_save = askdirectory()
if path_save is None:
    return
ob = YouTube(URL, on_progress_callback=progress)
strm = ob.streams[0]
x = ob.description.split("|")
file_size = strm.filesize
dfile_size = file_size
dfile_size /= 1000000
dfile_size = round(dfile_size, 2)
label.config(text='Size: ' + str(dfile_size) + ' MB')
label.pack(side=TOP, pady=10)
desc.config(text=ob.title + '\n\n' + 'Label: ' + ob.author + '\n\n' + 'length: ' +
str(round(ob.length/60, 1)) + ' mins\n\n'
        'Views: ' + str(round(ob.views/1000000, 2)) + 'M')
desc.pack(side=TOP, pady=10)
strm.download(path_save, strm.title)
dBtn.config(state=NORMAL)
showinfo("Download Finished", 'Downloaded Successfully')
urlField.delete(0, END)
label.pack_forget()
desc.pack_forget()
dBtn.config(text='Start Download')

except Exception as e:
    print(e)
    print('Error!!')

def startDownloadthread():
    thread = Thread(target=startDownload)
    thread.start()

```

```

main = Tk()

main.title("My YouTube Downloader")
main.config(bg='#3498DB')

main.iconbitmap('images\youtube-ios-app.ico')

main.geometry("500x600")

file = PhotoImage(
    file='images\photo.png')
headingIcon = Label(main, image=file)
headingIcon.pack(side=TOP)

urlField = Entry(main, font=("Times New Roman", 18), justify=CENTER)

urlField.pack(side=TOP, fill=X, padx=10, pady=15)

dBtn = Button(main, text="Start Download", font=(
    "Times New Roman", 18), relief='ridge', activeforeground='red',
command=startDownloadthread)
dBtn.pack(side=TOP)
label = Label(main, text="")
desc = Label(main, text="")
author = Label(main, text="@G.S.")
author.config(font=("Courier", 44))
author.pack(side=BOTTOM)
main.mainloop()

```

News.py

```
import requests
```

```

import json
import pyttsx3
engine = pyttsx3.init()
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[0].id)

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

def speak_news():
    url = 'http://newsapi.org/v2/top-headlines?sources=the-times-of-india&apiKey=yourapikey'
    news = requests.get(url).text
    news_dict = json.loads(news)
    arts = news_dict['articles']
    speak('Source: The Times Of India')
    speak('Today's Headlines are..')
    for index, articles in enumerate(arts):
        speak(articles['title'])
        if index == len(arts)-1:
            break
        speak('Moving on the next news headline..')
    speak('These were the top headlines, Have a nice day Sir!!!')

def getNewsUrl():
    return 'http://newsapi.org/v2/top-headlines?sources=the-times-of-india&apiKey=yourapikey'

if __name__ == '__main__':
    speak_news()

Diction.py
from difflib import get_close_matches

```

```

import pyttsx3
import json
import speech_recognition as sr

data = json.load(open('data.json'))
engine = pyttsx3.init()
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[0].id)

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

def takeCommand():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        print('Listening...')
        r.pause_threshold = 1
        r.energy_threshold = 494
        r.adjust_for_ambient_noise(source, duration=1.5)
        audio = r.listen(source)

    try:
        print('Recognizing..')
        query = r.recognize_google(audio, language='en-in')
        print(f'User said: {query}\n')

    except Exception as e:
        # print(e)

        print('Say that again please...')

```

```
    return 'None'
return query
```

```
def translate(word):
    word = word.lower()
    if word in data:
        speak(data[word])
    elif len(get_close_matches(word, data.keys())) > 0:
        x = get_close_matches(word, data.keys())[0]
        speak('Did you mean ' + x +
              ' instead, respond with Yes or No.')
        ans = takeCommand().lower()
        if 'yes' in ans:
            speak(data[x])
        elif 'no' in ans:
            speak("Word doesn't exist. Please make sure you spelled it correctly.")
        else:
            #changed from we to I
            speak("I did not understand your entry.")

    else:
        speak("Word doesn't exist. Please double check it.")
```

```
if __name__ == '__main__':
    translate()
```

Amazon.py

```
import requests
from bs4 import BeautifulSoup
import pyttsx3
import smtplib
```

```
engine = pyttsx3.init()
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[0].id)
```

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

```
def send_email():
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.ehlo()
    server.starttls()
    server.login('umeshdudigani@gmail.com', '8686362201')
    subject = 'Price fell down!'
    body = 'https://www.amazon.in/WOW-Brightening-Vitamin-Face-
Wash/dp/B07SZ243VZ/ref=sr_1_6?dchild=1&keywords=wow+face+wash&qid=15
94306550&smid=A27LPMZIGZ21IK&sr=8-6'
    content = f'Subject: {subject}\n\n{body}'
    server.sendmail('email', 'duduganiumesh@gmail.com', content)
    server.close()
```

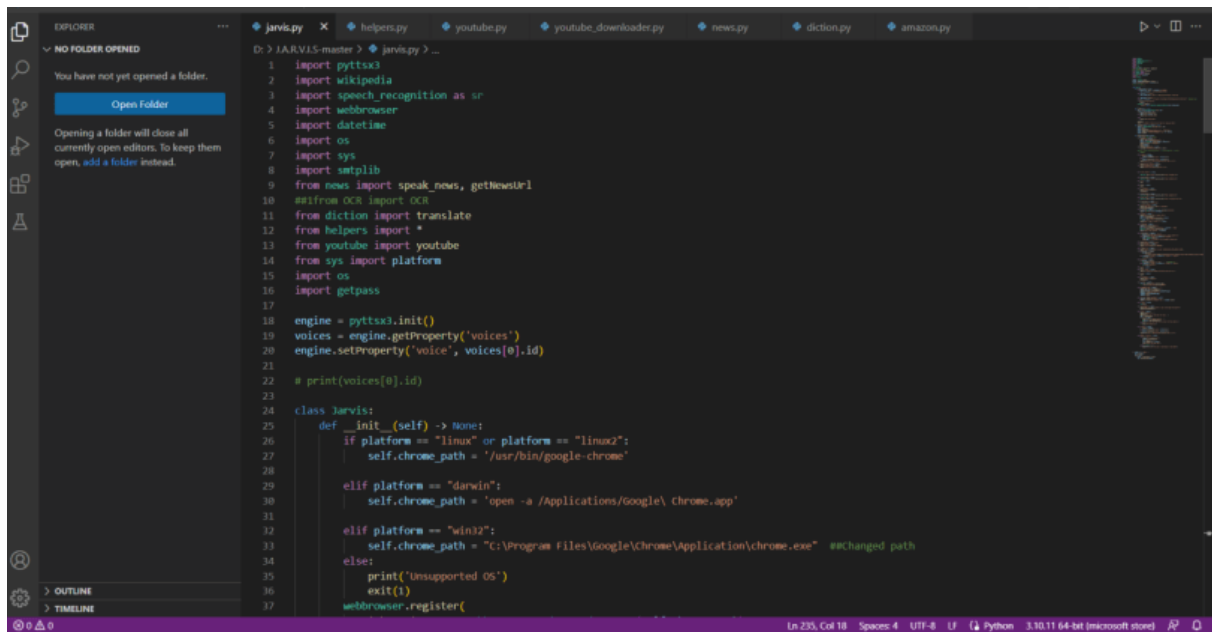
```
URL = 'https://www.amazon.in/WOW-Brightening-Vitamin-Face-
Wash/dp/B07SZ243VZ/ref=sr_1_6?dchild=1&keywords=wow+face+wash&qid=15
94306550&smid=A27LPMZIGZ21IK&sr=8-6'
headers = {
    "User-Agent": 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/83.0.4103.116 Safari/537.36'}
```

```
page = requests.get(URL, headers=headers)
```

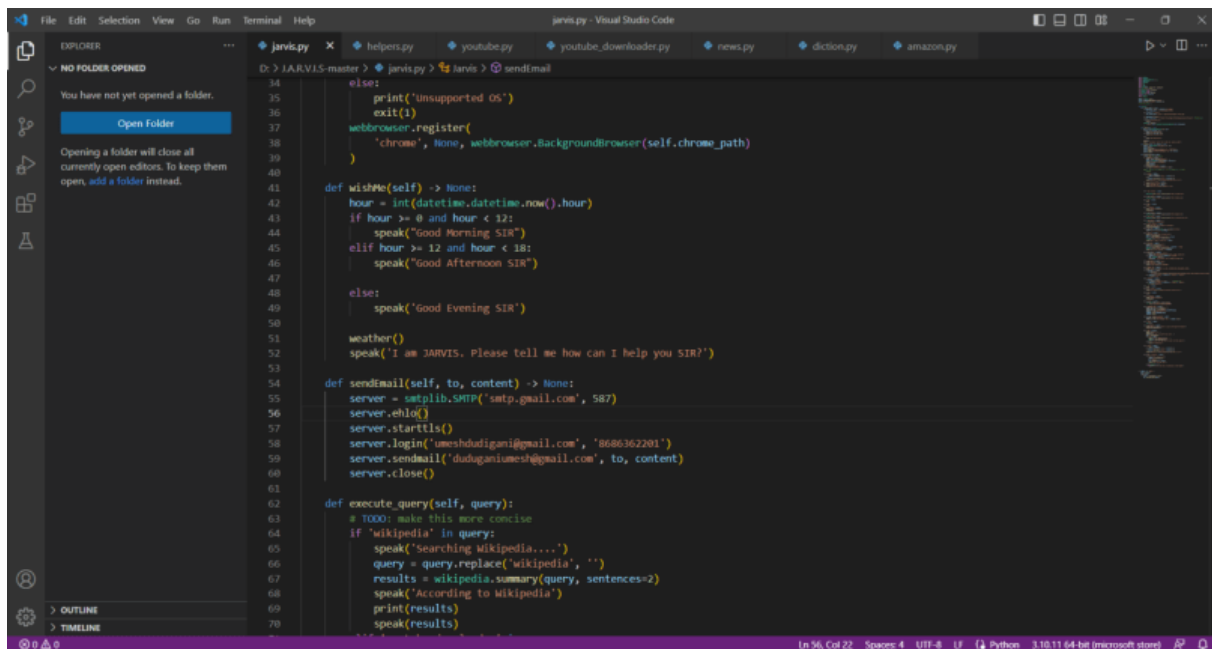
```
soup = BeautifulSoup(page.content, 'html.parser')

title = soup.find(id='productTitle')
price = soup.find(id='priceblock_dealprice').get_text().strip()
speak(price)
price = price[1:5]
price = float(price)
send_email()
```

B.SCREENSHOTS



```
D:\JARVIS-master > python Jarvis.py
1 import pyttsx3
2 import wikipedia
3 import speech_recognition as sr
4 import webbrowser
5 import datetime
6 import os
7 import sys
8 import smtplib
9 from news import speak_news, getNewsUrl
10 #from OCR import OCR
11 from diction import translate
12 from helpers import *
13 from youtube import youtube
14 from sys import platform
15 import os
16 import getpass
17
18 engine = pyttsx3.init()
19 voices = engine.getProperty('voices')
20 engine.setProperty('voice', voices[0].id)
21
22 # print(voices[0].id)
23
24 class Jarvis:
25     def __init__(self) -> None:
26         if platform == "linux" or platform == "linux2":
27             self.chrome_path = '/usr/bin/google-chrome'
28
29         elif platform == "darwin":
30             self.chrome_path = 'open -a /Applications/Google\ Chrome.app'
31
32         elif platform == "win32":
33             self.chrome_path = "C:\Program Files\Google\Chrome\Application\chrome.exe" #changed path
34         else:
35             print('Unsupported OS')
36             exit(1)
37         webbrowser.register(
```



```
34
35 else:
36     print('Unsupported OS')
37     exit(1)
38     webbrowser.register(
39         'chrome', None, webbrowser.BackgroundBrowser(self.chrome_path)
40     )
41
42 def wishMe(self) -> None:
43     hour = int(datetime.datetime.now().hour)
44     if hour >= 0 and hour < 12:
45         speak("Good Morning SIR")
46     elif hour >= 12 and hour < 18:
47         speak("Good Afternoon SIR")
48     else:
49         speak("Good Evening SIR")
50
51     weather()
52     speak("I am JARVIS. Please tell me how can I help you SIR?")
53
54 def sendEmail(self, to, content) -> None:
55     server = smtplib.SMTP("smtp.gmail.com", 587)
56     server.ehlo()
57     server.starttls()
58     server.login('umeshdudigani@gmail.com', '8686362201')
59     server.sendmail('dudugani.umar@gmail.com', to, content)
60     server.close()
61
62 def execute_query(self, query):
63     # TODO: make this more concise
64     if 'wikipedia' in query:
65         speak('searching wikipedia....')
66         query = query.replace('wikipedia', '')
67         results = wikipedia.summary(query, sentences=2)
68         speak('According to Wikipedia')
69         print(results)
70         speak(results)
```

```

1 import pyttsx3
2 import pyautogui
3 import psutil
4 import pyjokes
5 import speech_recognition as sr
6 import json
7 import requests
8 import geocoder
9 from difflib import get_close_matches
10
11
12 engine = pyttsx3.init()
13 voices = engine.getProperty('voices')
14 engine.setProperty('voice', voices[0].id)
15 g = geocoder.ip('me')
16 data = json.load(open('data.json'))
17
18 def speak(audio) -> None:
19     engine.say(audio)
20     engine.runAndWait()
21
22 def screenshot() -> None:
23     img = pyautogui.screenshot()
24     img.save('C:\\Users\\limesh\\OneDrive\\Pictures\\Screenshots') #pastedpath
25
26 def cpu() -> None:
27     usage = str(psutil.cpu_percent())
28     speak("CPU is at "+usage)
29
30     battery = psutil.sensors_battery()
31     speak("battery is at")
32     speak(battery.percent)
33
34 def joke() -> None:
35     for i in range(5):
36         speak(pyjokes.get_jokes()[i])
37

```

```

45
46 audio = r.listen(source)
47
48 try:
49     print('Recognizing..')
50     query = r.recognize_google(audio, language='en-in')
51     print(f'User said: {query}\n')
52
53 except Exception as e:
54     # print(e)
55     print('Say that again please...')
56     return 'None'
57 return query
58
59 def weather():
60     api_url = "https://fcc-weather-api.glitch.me/api/current?lat=" + \
61         str(g.latlng[0]) + "&lon=" + str(g.latlng[1])
62
63     data = requests.get(api_url)
64     data_json = data.json()
65     if data_json["cod"] == 200:
66         main = data_json["main"]
67         wind = data_json["wind"]
68         weather_desc = data_json["weather"][0]
69         speak(str(data_json["coord"]["lat"]) + 'latitude' + str(data_json["coord"]["lon"]) + 'longitude')
70         speak('current location is ' + data_json["name"] + data_json["sys"]["country"] + 'dia')
71         speak('weather type ' + weather_desc["main"])
72         speak('wind speed is ' + str(wind["speed"]) + ' metre per second')
73         speak('temperature: ' + str(main["temp"]) + 'degree celcius')
74         speak('Humidity is ' + str(main["humidity"]))
75
76
77 def translate(word):
78     word = word.lower()
79     if word in data:
80         speak(data[word])
81     elif len(get_close_matches(word, data.keys())) > 0:

```

```
1 from urllib.request import urlopen
2 import urllib.parse
3 import webbrowser
4 from sys import platform
5 import os
6
7 if platform == "linux" or platform == "linux2":
8     chrome_path = '/usr/bin/google-chrome'
9
10 elif platform == "darwin":
11     chrome_path = 'open -a /Applications/Google\ Chrome.app'
12
13 elif platform == "win32":
14     chrome_path = "c:\\Program Files\\Google\\Chrome\\Application\\chrome.exe"
15 else:
16     print('Unsupported os')
17     exit(1)
18
19 webbrowser.register('chrome', None, webbrowser.BackgroundBrowser(chrome_path))
20
21
22 def youtube(textToSearch):
23     query = urllib.parse.quote(textToSearch)
24     url = "https://www.youtube.com/results?search_query=" + query
25     webbrowser.get('chrome').open_new_tab(url)
26
27
28 if __name__ == '__main__':
29     youtube('any text')
```

```
1 from pytube import *
2 from tkinter import *
3 from tkinter.filedialog import *
4 from tkinter.messagebox import *
5 from threading import *
6 file_size = 0
7
8
9 def progress(stream=None, chunk=None, remaining=None):
10     file_downloaded = (file_size-remaining)
11     per = round((file_downloaded/file_size)*100, 1)
12     dbtn.config(text=f'{per}% downloaded')
13
14
15 def startDownload():
16     global file_size
17     try:
18         url = urlField.get()
19         dbtn.config(text='Please wait...')
20         dbtn.config(state=DISABLED)
21         path_save = askdirectory()
22         if path_save is None:
23             return
24         ob = YouTube(url, on_progress_callback=progress)
25         strm = ob.streams[0]
26         x = ob.description.split("\n")
27         file_size = strm.filesize
28         dfile_size = file_size
29         dfile_size /= 1000000
30         dfile_size = round(dfile_size, 2)
31         label.config(text='Size: ' + str(dfile_size) + ' MB')
32         label.pack(side=TOP, pady=10)
33         desc.config(text=ob.title + '\n\n' + 'Label: ' + ob.author + '\n\n' + 'length: ' + str(round(ob.length/60, 1)) + ' mins\n\n'
34             + 'Views: ' + str(round(ob.views/1000000, 2)) + ' M')
35         desc.pack(side=TOP, pady=10)
36         strm.download(path_save, strm.title)
37         dbtn.config(state=NORMAL)
```

```

31 label.config(text='Size: ' + str(dfile_size) + ' MB')
32 label.pack(side=TOP, pady=10)
33 desc.config(text=obj.title + '\n\n' + 'Label: ' + obj.author + '\n\n' + 'length: ' + str(round(obj.length/60, 1)) + ' mins\n\n'
34             + 'Views: ' + str(round(obj.views/1000000, 2)) + 'M')
35 desc.pack(side=TOP, pady=10)
36 strm.download(path_save, strm.title)
37 dbtn.config(state=NORMAL)
38 showinfo("Download Finished", "Downloaded Successfully")
39 urField.delete(0, END)
40 label.pack_forget()
41 desc.pack_forget()
42 dbtn.config(text='Start Download')
43
44 except Exception as e:
45     print(e)
46     print('Error!!!')
47
48
49 def startDownloadthread():
50     thread = Thread(target=startDownload)
51     thread.start()
52
53
54 main = Tk()
55
56 main.title("My Youtube Downloader")
57 main.config(bg='#3498db')
58
59 main.iconbitmap('images\youtube-ios-app.ico')
60
61 main.geometry("500x500")
62
63 file = PhotoImage(
64     file='images\photo.png')
65 headingIcon = label(main, image=file)
66 headingIcon.pack(side=TOP)
67

```

```

1 import requests
2 import json
3 import pyttsx3
4 engine = pyttsx3.init()
5 voices = engine.getProperty('voices')
6 engine.setProperty('voice', voices[0].id)
7
8
9 def speak(audio):
10     engine.say(audio)
11     engine.runAndWait()
12
13
14 def speak_news():
15     url = 'http://newsapi.org/v2/top-headlines?sources=the-times-of-india&apiKey=yourapikey'
16     news = requests.get(url).text
17     news_dict = json.loads(news)
18     arts = news_dict['articles']
19     speak('Source: The Times Of India')
20     speak('Today's Headlines are..')
21     for index, articles in enumerate(arts):
22         speak(articles['title'])
23         if index == len(arts)-1:
24             break
25     speak('Moving on the next news headline..')
26     speak('These were the top headlines, Have a nice day Sir!!..')
27
28 def getNewsurl():
29     return 'http://newsapi.org/v2/top-headlines?sources=the-times-of-india&apiKey=yourapikey'
30
31 if __name__ == '__main__':
32     speak_news()
33

```

```
1 from difflib import get_close_matches
2 import pyttsx3
3 import json
4 import speech_recognition as sr
5
6 data = json.load(open('data.json'))
7 engine = pyttsx3.init()
8 voices = engine.getProperty('voices')
9 engine.setProperty('voice', voices[0].id)
10
11
12 def speak(audio):
13     engine.say(audio)
14     engine.runAndWait()
15
16
17 def takeCommand():
18     r = sr.Recognizer()
19     with sr.Microphone() as source:
20         print('Listening...')
21         r.pause_threshold = 1
22         r.energy_threshold = 494
23         r.adjust_for_ambient_noise(source, duration=1.5)
24         audio = r.listen(source)
25
26     try:
27         print('Recognizing..')
28         query = r.recognize_google(audio, language='en-in')
29         print(f'User said: {query}\n')
30
31     except Exception as e:
32         # print(e)
33
34         print('Say that again please...')
35         return 'None'
36     return query
37
```

```
22     r.energy_threshold = 494
23     r.adjust_for_ambient_noise(source, duration=1.5)
24     audio = r.listen(source)
25
26     try:
27         print('Recognizing..')
28         query = r.recognize_google(audio, language='en-in')
29         print(f'User said: {query}\n')
30
31     except Exception as e:
32         # print(e)
33
34         print('Say that again please...')
35         return 'None'
36     return query
37
38
39 def translate(word):
40     word = word.lower()
41     if word in data:
42         speak(data[word])
43     elif len(get_close_matches(word, data.keys())) > 0:
44         x = get_close_matches(word, data.keys())[0]
45         speak('Did you mean ' + x +
46               ' instead, respond with Yes or No.')
47         ans = takeCommand().lower()
48         if 'yes' in ans:
49             speak(data[x])
50         elif 'no' in ans:
51             speak('word doesn't exist. Please make sure you spelled it correctly.')
52         else:
53             # changed from we to I
54             speak('I did not understand your entry.')
55
56     else:
57         speak('word doesn't exist. Please double check it.')
58
```

```
File Edit Selection View Go Run Terminal Help
amazon.py - Visual Studio Code

EXPLORER
NO FOLDER OPENED
You have not yet opened a folder.
Open Folder
Opening a folder will close all currently open editors. To keep them open, add a folder instead.

D:\JARVIS-master > amazon.py > speak
1 import requests
2 from bs4 import BeautifulSoup
3 import pyttsx3
4 import smtplib
5
6 engine = pyttsx3.init()
7 voices = engine.getProperty('voices')
8 engine.setProperty('voice', voices[0].id)
9
10
11 def speak(audio):
12     engine.say(audio)
13     engine.runAndWait()
14
15
16 def send_email():
17     server = smtplib.SMTP('smtp.gmail.com', 587)
18     server.ehlo()
19     server.starttls()
20     server.login('umeshdudigani@gmail.com', '8686362281')
21     subject = 'Price fell down!'
22     body = 'https://www.amazon.in/MM-Brightening-Vitamin-Face-Wash/dp/B07S2243VZ/ref=sr_1_6?dchild=1&keywords=wow+face+wash&qid=1594306558&se
23     content = f'Subject: {subject}\n\n{body}'
24     server.sendmail('email', 'duduganumesh@gmail.com', content)
25     server.close()
26
27
28 URL = 'https://www.amazon.in/MM-Brightening-Vitamin-Face-Wash/dp/B07S2243VZ/ref=sr_1_6?dchild=1&keywords=wow+face+wash&qid=1594306558&se
29 headers = {
30     'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.116 Safari/537.36'
31 }
32 page = requests.get(URL, headers=headers)
33
34 soup = BeautifulSoup(page.content, 'html.parser')
35
36 title = soup.find(id='productTitle')
37 price = soup.find(id='priceblock_dealprice').get_text().strip()
```

Ln 12, Col 22 Spaces: 4 UTF-8 LF Python 3.10.11 64-bit (microsoft store)

A SMART APPLICATION WITH MACHINE LEARNING EXTENSION TO OPERATE COMPUTER AND IOT DEVICES

D. UMESH 39130125

B.E Student

umeshdudigani@gmail.com

Department of ECE

Dr .L.MAGTHELIN THERASE

Associate Professor

magthelintheras.ece@sathyabama.ac.in

Department of ECE

D. SRI HARSHA 39130106

B.E Student

harshadasari2001@gmail.com

Department of ECE

Sathyabama Institute of Science and Technology,
Chennai, India

ABSTRACT

We could be able to finish all duties on our portable devices in the future thanks to smartphones and the Internet of Things (IoT). Many apps for Android, iOS, and other platforms are compatible with practically all sorts of electronic devices. There are specialised apps that solely function on mobile devices and manage computer software and functionality. In addition to introducing and suggesting a new app that uses voice commands to interact with and control computers and has many other IoT-related functions, we have listed additional applications that are currently available and discussed them in this essay. It uses natural language processing and machine learning to understand and interpret voice commands and enquiries from users, and it can be customised to match specific preferences and needs. via updates and questions from users. The system is always being updated and enhanced through upgrades and user feedback.

Keywords- NLP, IOT.

1. INTRODUCTION

Sometimes, the terms "intelligent virtual assistant" (IVA) and "intelligent personal assistant" (IPA) are used interchangeably (IVA). A software agent known as an intelligent personal assistant (IPA) or intelligent virtual assistant (IVA) may perform tasks or provide services on behalf of a user in response to

commands or inquiries. Intelligent virtual assistant (IVA) and intelligent personal assistant are additional names for this type of assistance (IPA). Another name for an intelligent personal assistant (IPA) is an intelligent virtual assistant (IVA), along with them (IVA). When discussing virtual assistants that are accessible in general or specifically through online chat, the term "chatbot" is occasionally used. Any method can be used to access these assistance. In other cases, the only goal of online chat systems is to provide participants with some form of enjoyment. Many of the virtual assistants that are currently on the market are able to speak with consumers utilising voices that were created by computers while still being able to understand human speech. Users have the ability to speak to their assistants to ask inquiries, handle other important tasks like email, to-do lists, and calendars, and issue voice commands to home automation devices and media players. The other dialogue systems are based on a similar concept, but this one is special in a few significant ways. The capabilities of virtual assistants and the ways in which they are employed are both fast evolving as of the year 2017. Now that new products are entering the market, user interfaces for phone and email conversations are receiving a lot of attention. Both Google and Apple already have sizable user bases using their respective mobile platforms. The Windows operating system is installed on a sizable number of personal computers, smartphones, and smart speakers. All of these gadgets were produced by

different companies. There are several smart speakers from Amazon that are already installed in its customers' houses. Customers and Conversica's intelligent virtual assistants for organisations have had more than 100 million conversations utilising the company's email and SMS interfaces. Now, when people use the term "virtual assistant," they don't just refer to a computer; they can also refer to a person whose main responsibility is to help his employer carry out a certain online task remotely. In other words, when people refer to a "virtual assistant," they don't just mean a machine. This person spends the most of their time living abroad rather than in their home nation. Since artificial intelligence has been so ingrained in our society, it is currently impossible for us to fathom a world without it. The daily quality of life, which is crucial to overall happiness, might be considerably enhanced by a voice assistant. In the not too distant future, the ability of machines to interpret and comprehend written language will lag far behind that of machines that can recognise human voice. One of the key characteristics of a great assistant is the capacity to engage someone in conversation and respond appropriately while taking into account the information they provide. The scope of the request and the reason it was made, and then give a brief response that addresses both of these factors. However, since the majority of these advantages are available to machines, it is vital to apply different strategies to guarantee accurate interpretations of user input. This can be achieved by: Translating human speech into a dialect that computers can understand is the sole task of message translators, more commonly referred to as artificial intelligences (AIs). However, the System that we are creating for this project is built in Python, and we may use a number of application programming interfaces (APIs) to accept messages from the resources and deliver them to users (APIs). In this particular strategy, we are only emphasising the elements that the vast majority of people already require, such as having access to YouTube, the most recent news headlines, and other things of the sort. It shouldn't be challenging to meet the needs of any and all users if one follows the provided plan. Another name for a software agent is h, which is a programme that does tasks or provides services on behalf of its user..

2. LITERATURE SURVEY

[1] Giancarlo Iannizzotto, Lucia Lo Bello, Andrea Nucita, Giorgio Mario Grasso.

"A Vision and Speech Enabled, Customizable, Virtual Assistant for Smart Environments", 11th International Conference on Human System Interaction (HSI), 2018. In this work, Giancarlo Iannizzotto develops an architecture for a virtual assistant for smart home automation systems by combining some of the most cutting-edge techniques in computer vision, deep learning, speech production and recognition, and artificial intelligence. The suggested assistant utilises a small, inexpensive Raspberry PI 3 device and is effective and resource-efficient, interactive, and adaptable. The system ran for several days while users were encouraged to interact with it, proving to be precise, dependable, and appealing. It was merged into an open source home automation environment for testing reasons.

[2] Kumaran N., Rangaraj V., Siva Sharan S., Dhanalakshmi R.

"Intelligent Personal Assistant – Implementing Voice Commands enabling Speech Recognition", International Conference on System, Computation, Automation and Networking (ICSCAN), 2020. Software agents known as intelligent personal assistants (IPAs) carry out tasks on behalf of people using chatbot-like instructions and inquiries. Synthesized voices are used by intelligent virtual assistants to understand and respond to human speech. Applications for IPAs and IVAs include media playback by voice, task management, and home automation. In this essay, speech recognition techniques are suggested, and a virtual personal assistant is proposed. The current system is operated by a third party and is accessed via the internet. In addition to using the local database, voice recognition, and synthesiser, this programme must safeguard personal information from unwanted access. A parser called SURR (Semantic Unification and Reference Resolution) is employed to recognize the speech. Text is converted by the synth.

[3] Vinayak Iyer, Kshitij Shah, Sahil Sheth, Kailas Devadkar

"Virtual assistant for the visually impaired", 5th International Conference on Communication and Electronics Systems (ICCES), 2020. Vinayak Iyer illustrates the use of software that makes it easier

for those with visual impairments to use the internet. The programme will considerably increase usability and play a big influence in how the internet is accessed. Despite technological advancements, websites and the internet in general are still inaccessible to people who are blind or visually impaired. These websites can be easily interacted with thanks to the programme. Using voice commands rather than the conventional keyboard and mouse to access and give commands to any website adds a new dimension. The programme may read out the website's content before automating any website utilising speech-to-text and text-to-speech modules, as well as Selenium. The user is not required to remember difficult Braille keyboard shortcuts or deal with the difficulty of typing. He or she only needs to speak the command, and the programme will carry it out. Using a BERT model trained on the Stanford Question Answer Dataset, the system can also offer a summary of the website's content and respond to user inquiries in light of the summary.

[4] CH.M.H. Saibaba, Saiyed Faiyaz Waris, S.Hrushikesava Raju, VSRK Sarma, Vijaya Chandra Jadala, Chitturi Prasad.

“Intelligent Voice Assistant by Using OpenCV Approach”, Second International Conference on Electronics and Sustainable Communication Systems (ICESC), 2021. In this work, CH.M.H. Saibaba concentrated on the various AI technologies that assist speech recognition and natural language processing (NLP). The researchers preferred voice assistants and gadgets like Alexa to try and execute such silent jobs. Although a number of robust packages, including pywhatkit, Wikipedia, pyttsx3, pygame, speech recognition, OpenCV, and others, can accomplish the same practicalities, the focus of this study was on the actual construction of an OpenCV-based voice recognition module. Speech recognition, object recognition, digital image processing, and language processing all play significant roles in security and other aspects of modern technology. The most widespread artificial intelligence techniques that use machines and deep learning to identify a person's natural language are those.

[5] Vindhya Vasini K, Tejasvi K, Pravallika K, Ch. Nanda Krishna.

“A Natural Language Processing based Intelligent Bot Application”, International Conference on

Sustainable Computing and Data Communication Systems (ICSCDS), 2022. This study proposes a voice-managed non-public assistant. The robot helper can be remotely instructed to carry out specified movements, turns, start/stop activities, and move an object from one place to another using human voice instructions. The spoken sign commands are converted to text and Bluetooth-send to the robot. The personal assistant robot is developed mostly on a microcontroller-based architecture and is aware of its current location. The efficiency of voice-manipulated communication over a long distance is investigated in a number of trials. The performance evaluation is completed by the optimistic findings from the early experiments. Also suggested are prospective improvements to capability programmes in homes, hospitals, and businesses.

3. EXISTING SYSTEM

There are numerous personal assistant systems currently on the market, each with special features and skills. These are a few well-known examples:

The voice-activated personal assistant Alexa was created by Amazon. It has a wide range of capabilities, including the ability to play music, make phone calls, set alarms, and operate smart home appliances.

Google Assistant is a virtual assistant that uses AI and was created by Google. It has many of the same capabilities as Alexa, including the ability to send texts, answer questions, and give directions.

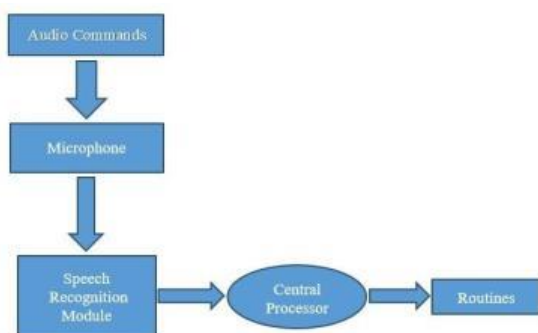
Apple Siri: Siri is an iOS-compatible virtual personal assistant created by Apple. It has many of the same capabilities as Alexa and Google Assistant, like booking reservations at restaurants, sending emails, and getting weather updates.

Microsoft Cortana: Available on Windows-based devices, Cortana is a personal assistant created by Microsoft. It has many of the same capabilities as other personal assistants, including the ability to launch apps, send reminders, and plan appointments.

4. PROPOSED SYSTEM

The task will involve analysing the audio commands issued by the user while speaking into the microphone. This could involve anything, such as obtaining information or

managing computer files, etc. This study is based on reading the aforementioned material and putting its examples to the test. With the express purpose of discovering best practises and a deeper understanding of voice assistant, tests are made by programming in accordance with books and internet resources. The speech input is converted to text using speech recognition. The central processor then receives this text and determines the nature of the command before calling the appropriate script to run. The project is composed of open source software modules that are supported by the PyCharm community and can handle any updates. This project's modular design gives it greater flexibility and makes it simple to add new features without impairing existing system functionality. It not only responds to human orders but also to questions posed by the user or words said by the user, such as opening operations and tasks. The voice assistant is introducing itself to the user in a way that makes them feel more at ease and allows them to engage with it freely. The programme should also get rid of any extra manual labour that is required for users to complete each job. Instead of the subsequent input, the entire system operates on the spoken input.



Sends email messages

Gives you the headlines of the news in a dynamic way at any moment

Gives you the ability to search the wiki for whatever it is that you command.

Moreover, it plays music.

Identifies the time as well as the date.

Place in the dictionary, complete with its associated latitude and longitude coordinates.

In your report on the weather, please provide the current temperature, the current wind speed, and the current humidity. You should also provide a description of the weather.

Gives access to vital websites for our organisation, including YouTube, GitHub, and Stack Overflow

5. MODULES

Module 1 - Setting Up Initial Imports Using pyttsx3, convert text to speech

The Python text-to-speech library Pyttsx3 will now be set up as our engine. We'll use sapi5, a voice application platform interface from Microsoft, for the text-to-speech feature. We'll make use of this specific module. The pyttsx3 module for Python can translate text into spoken language. It is compatible with Python versions 2 and 3 and may be used without an internet connection, unlike other libraries. An application must initially call thepyttsx3.init() factory method in order to get a reference to a pyttsx3. Engine instance. With this tool, converting written words into spoken words is a pretty easy and basic process. Two separate voices can be supported by the pyttsx3 module. The "sapi5" programme for Windows offers two versions: the first is the female version, and the second is the male version. There are threedifferent TTS engines in it.

sapi5 – SAPI5 on Windows

nsss – NSSpeechSynthesizer on Mac OS X

espeak – eSpeak on every other platform.

Speech Recognition module

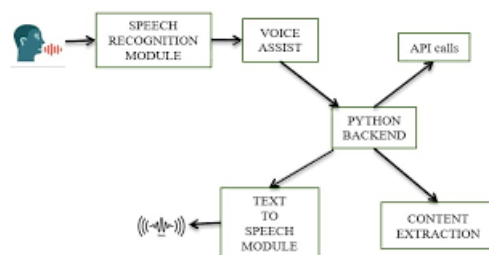
The speech recognition module is then constructed.

Applications like home automation and artificial intelligence use speech recognition as a key component. To transform speech input to text, the system makes advantage of Google's online speech recognition system. The input of speech The information center's computer network server hosts a particular corpus of texts that users can access. These texts are temporarily stored in the system before being transmitted to the Google cloud for speech recognition. The central processor is subsequently given the equivalent text, which is then received.

API - gTTS API

Application Programming Interface is referred to as API. A software bridge called an API enables communication between two programmes. In other words, an API is a messenger that transmits your request to the service provider you are using and returns the result. Python has a variety of APIs that can turn text into speech. The gTTS API, also referred to as the Google Speech To Text API, has been utilised. gTTS is a simple-to-use programme that turns entered text into audio that can be saved as an mp3 file. Languages including English, Hindi, Tamil, French, German, and many more are supported via the gTTS API.that turns entered text into audio that can be saved as an mp3 file. Languages including English, Hindi, Tamil, French, German, and many more are supported via the gTTS API. The speech can be delivered in any one of the two available audio speeds, fast or slow. To use the gTTS API, we opened the terminal to write:

pip install gTTS



Installation of playsound(A module that is used for playing sounds in python3)

The playsound module is installed. A cross-platform module that can play audio files is called playsound. There are no dependencies on this. On platforms, implementation differs. It utilises AppKit and windll.winmm on Windows. GStreamer on Linux and NSSound on Apple OS X. Unless you specify a path, the audio file should be in the same directory as your Python programme.

Module 2 – Creating voice Engine

Creating voice engine to speak output

```
def speak(text):
```

```
    tts = gTTS(text=text,lang="en")
    filename = "temp.mp3"
    tts.save(filename)
    playsound.playsound(filename)
```

Creating Speech Recognition

Now, we will use the same Speech Recognition library - API as mentioned in module 1

Creating Function to take command

The function to take command will be created after the library has been consulted. The speech recognition module is used in this function to take user commands and recognise them. This will be utilised to translate the voice instruction into further action by working.

Module 3 - Creating Voice Assistant

Wikipedia Search

To do a wikipedia-related search, we use the wikipedia library. It is used to retrieve several types of data from the Wikipedia website. Type the following command in the terminal to instal this module.

Launching Youtube

We use web browser library to open youtube if user ask.

Setting Up Online Functions

We'll be adding several online functions. They are:

- Find my IP address
- Search on Wikipedia
- Play videos on YouTube

- Search on Google
- Send WhatsApp message
- Send Email

Launching other application

Now we create functions to open notepad, paint, calculator, task manager, powershell, snip.

Function to search

We create a function to search from the web.

Function to operate pc environment

We create a function that can do Restart, shutdown, sleep, hibernate for the pc.

6. CONCLUSION

The user will experience fewer interruptions as a result of the VPA, allowing him to maximise the effectiveness of how he uses his time and combine the various channels through which he obtains information on his contacts, calendar, and other issues. The action of sending information Additionally, this article offers a framework for making decisions that may be used to phone screening as well as the management of meeting and appointment request inquiries. The essay goes on to provide the framework. The system will first focus on meeting the needs of professionals in sectors including law, medical, sales, retail, business administration, maintenance, and other closely related fields. On the other hand, it is commonly acknowledged that millions of users will employ this capacity by default in the not-too-distant future. This idea has been increasingly popular in recent years. Support for this prediction has come from a wide range of sources. It offers solutions to a number of issues that have existed for a while but that nobody has been able to resolve satisfactorily up until this point. Its main objective is to enhance VPA functionality so that they can be employed in more scenarios than was previously practical. Expanding the scope of VPAs' application will enable this. The system, on the other hand, has a unique set of prerequisites and constraints. Due to the intricacy of the methods and concepts, it is highly challenging to modify it in the future, should that be necessary. It is very tough

to change it in the future, despite the fact that its efficiency is high and its time consumption for finishing each task may be higher than that of the other VPAs. The fact that this is the case does not change because of how effective it is.

7. REFERENCES

- [1] Giancarlo Iannizzotto, Lucia Lo Bello, Andrea Nucita, Giorgio Mario Grasso, "A Vision and Speech Enabled, Customizable, Virtual Assistant for Smart Environments", 11th International Conference on Human System Interaction (HSI), 2018
- [2] Kumaran N., Rangaraj V., Siva Sharan S., Dhanalakshmi R., "Intelligent Personal Assistant – Implementing Voice Commands enabling Speech Recognition", International Conference on System, Computation, Automation and Networking (ICSCAN), 2020
- [3] Vinayak Iyer, Kshitij Shah, Sahil Sheth, Kailas Devadkar, "Virtual assistant for the visually impaired", 5th International Conference on Communication and Electronics Systems (ICCES), 2020
- [4] CH.M.H. Saibaba, Saiyed Faiyaz Waris, S.Hrushikesava Raju, VSRK Sarma, Vijaya Chandra Jadala, Chitturi Prasad, "Intelligent Voice Assistant by Using OpenCV Approach", Second International Conference on Electronics and Sustainable Communication Systems (ICESC), 2021
- [5] Vindhya Vasini K, Tejasvi K, Pravallika K, Ch. Nanda Krishna, "A Natural Language Processing based Intelligent Bot Application", International Conference on Sustainable Computing and Data Communication Systems (ICSCDS), 2022

