

**NAME: - INSIYA RIZVI**

**ROLL NO.IT2021013**

**MUMBAI UNIVERSITY**

**SMT.JANAKIBAI RAMA SALVI COLLEGE OF ARTS/COMMERCE  
& SCIENCE**



**A**

**PRACTICAL REPORT ON**

**SOFTWARE PROJECT MANAGEMENT**

**SUBMITTED BY  
INSIYA RIZVI**

**UNDER THE GUIDANCE OF  
Prof. VIJAY KOTHWADE**

**DEPARTMENT OF INFORMATION TECHNOLOGY  
2023-2024**

**NAME: - INSIYA RIZVI**

**ROLL NO.IT2021013**

**SMT.JANAKIBAI RAMA SALVI COLLEGE OF ARTS/COMMERCE  
& SCIENCE**

**DEPARTMENT OF INFORMATION TECHNOLOGY**



**CERTIFICATE**

This is certify that the group of the following students of Third Year Degree in Information Technology has completed the project report

**ON**

**“SOFTWARE PROJECT MANAGEMENT”**

As a partial fulfillment for completion of “Third Year Degree in Information Technology” awarded by Maharashtra State Board of Technical Education Mumbai

For year 2023-2024

**SUBMITTED BY**

**INSIYA RIZVI**

**Under the guidance  
of Prof. VIJAY  
KOTHWADE**

**EXAMINER**

**HOD**

**COLLEGE STAMP**

**NAME: - INSIYA RIZVI**

**ROLL NO.IT2021013**

## **ACKNOWLEDGE**

.....

We are thankful to Prof. VIJAY KOTHWADE for his guidance and valuable suggesting in carrying out this project

I acknowledge with deep sense of gratitude for timely helpful guidance about "SOFTWARE PROJECT MANAGEMENT" Received from prof. VIJAY KOTHWADE.

This is very sweet movement for me that I have completed my project

I am thankful to my principle Mr. & special thanks to all staff member & co-ordinate of "SMT.JANAKIBAI RAMA SALVI COLLEGE OF ARTS/COMMERCE & SCIENCE" for offering necessary work facilities.

Finally, I wish to thanks the staff, principal, friends & person who helped Me directly &indirectly for completion of project.

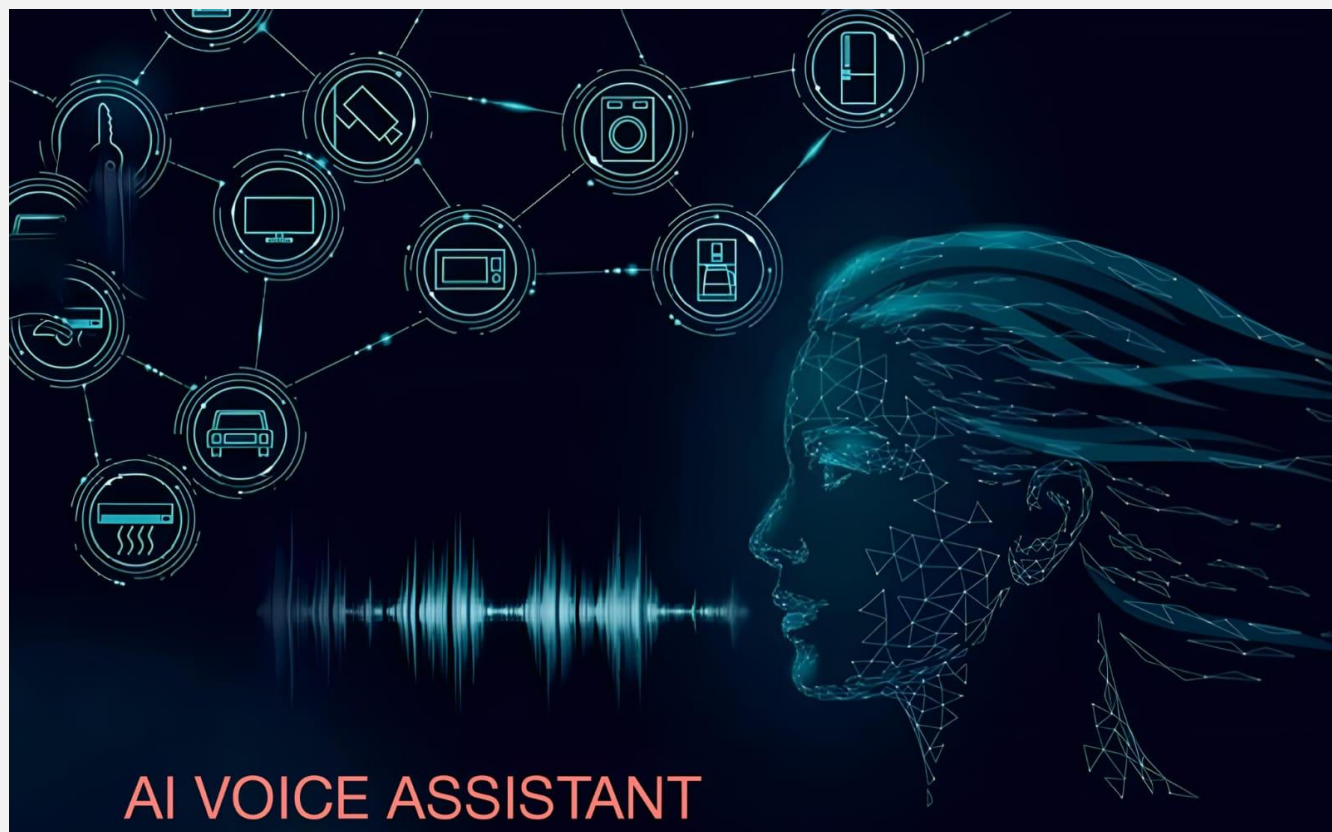
INSIYA RIZVI

# AI VOICE ASSISTANT REPORT



Name : INSIYA RIZVI

Roll No : IT2021013



## TABLE OF CONTENTS

| Chapter No. | TITLE  |  |
|-------------|--|--|
|             |  |  |
| <b>1</b>    | <b>INTRODUCTION</b>  |  |
|             | 1.1. BACKGROUND  |  |
|             | 1.2. WHAT IS VOICE ASSISTANT   |  |
|             | 1.3. WHY DO WE NEED IT   |  |
|             | 1.4. OBJECTIVE<br>1.5. PURPOSE<br>1.6. SCOPE   |  |
| <b>2</b>    | <b>SYSTEM ANALYSIS</b>   |  |
|             | 2.1. EXISTING SYSTEM<br>2.2. PROPOSED SYSTEM<br>2.3. SOFTWARE REQUIREMENTS<br>2.4. HARDWARE REQUIREMENTS<br>2.5. LIBRARIES<br>2.6. PROGRAMMING LANGUAGE (PYTHON)<br>2.7. DOMAIN<br>2.8. JUSTIFICATION TECHNOLOGY |  |

|          |  |  |
|----------|--|--|
| <b>3</b> | <b>SYSTEM DESIGN</b>   |  |
|          | 3.1. SYSTEM ARCHITECTURE   |  |
|          | 3.2. ALGORITHM USED <ul style="list-style-type: none"> <li>➤ SPEECH RECOGNITION MODULE</li> <li>➤ . SPEECH TO TEXT &amp; TEXT TO SPEECH CONVERSION</li> <li>➤ PROCESS &amp; EXECUTES THE REQUIRED COMMAND</li> </ul> |  |
|          | 3.3. DESIGN DIAGRAMS <ul style="list-style-type: none"> <li>➤ USE CASE DIAGRAM</li> <li>➤ COMPONENT DIAGRAM</li> <li>➤ SEQUENCE DIAGRAM</li> </ul>   |  |
|          | 3.4. FEASIBILITY STUDY   |  |
|          | 3.5. TYPES OF OPERATIONS   |  |
| <b>4</b> | <b>RESULTS AND DISCUSSION</b>  |  |
|          | 4.1. WORKING<br>4.2. CONCLUSION<br>4.3. FUTURE WORK<br>4.4. SOURCE CODE<br>4.5 SCREENSHOTS   |  |
| <b>5</b> | <b>IMPLEMENTATION AND TESTING</b>  |  |
|          | 5.1. FEASIBILTY TESTING<br>5.2. TESTING APPROACH <ul style="list-style-type: none"> <li>➤ UNIT TESTING</li> <li>➤ INTEGRATION TESTING</li> </ul>   |  |

|          |  |  |
|----------|--|--|
| <b>6</b> | <b>CONCLUSION AND FEATURE WORK</b>                             |  |
|          | 6.1. CONCLUSION<br>6.2. SYSTEM IMPEMNTATION AND<br>ENHANSEMENT |  |
| <b>7</b> | <b>REFERENCES.</b>   |  |

# INTRODUCTION

---

A Voice Assistant is one of the hot topics in the current world that are programs that listens to human's verbal command and respond to them which makes it a human- computer/device interaction. In the current days, a voice assistant is everywhere which is a lot useful in these busy days. Nowadays, almost everyone in the current world is using voice assistant because it's everywhere starting from Google smartphone assistant which even 5 years old kids will know how to use because of the current world pandemic which makes them use smartphones till Amazon's Alexa which will be very useful to do works starting from entertaining the users till turning on and off the household products (Internet of Things). One of the greatest features is that it will be very useful to even physically challenged people, for example, people who aren't able to walk use the Internet of Things (IoT) feature to operate the household products and maintain them. So, we tend to develop a voice assistant which will be very useful to the users same as the other voice assistants which are currently in the world.





In the 21st century, human interaction is being replaced by automation very quickly. One of the main reasons for this change is performance. There's a drastic change in technology rather than advancement. In today's world, we train our machines to do their tasks by themselves or to think like humans using technologies like Machine Learning, Neural Networks, etc. Now in the current era, we can talk to our machines with the help of virtual assistants.

Virtual assistants are software programs that help you ease your day to day tasks, such as showing weather reports, giving daily news, searching the internet etc. They can take commands by voice. Voice-based intelligent assistants need an invoking word or wake word to activate the listener, followed by the command. We have so many virtual assistants, such as Apple's Siri, Amazon's Alexa and Microsoft's Cortana and Amazon's Alexa and this has been an inspiration for us to do this as a project. This system is designed to be used efficiently on desktops. Voice assistants are programs on digital devices that listen and respond to verbal commands. A user can say, "What's the weather?" and the voice assistant will answer with the weather report for that day and location.

## 1.1 BACKGROUND

Healthcare is the most crucial parts of the human life. Nowadays, so many are not willing to go to hospital, due to work overload and negligence of their health. The doctors and nurses are putting up maximum efforts to save people's lives without even considering their own loves. There are also some villages which lack medical facilities.

Accurate and on-time analysis of any health-related problem is important for the prevention and treatment of the illness. The traditional way of diagnosis may not be sufficient in the case of a serious ailment. In this situation, where everything has turned virtual, the doctors and nurses are putting up maximum efforts to save people's lives even if they have to danger their own.

There are also some remote villages which lack medical facilities. The dataset was processed in ML models Naive Bayes and Decision Tree. While processing the data, symptoms are given as input and the disease was received as an output. This project helps to get the idea about the disease of an individual based on the symptoms he/she have, and get the treatment easily by contacting the concern doctor.

## 1.2WHAT IS VOICE ASSISTANT?



A voice assistant, also known as an intelligent personal assistant or a connected speaker, is a new type of device that is based on natural language speech recognition and is offered by popular companies like Apple, Amazon, and Google. We got inspired by that and created one our self.

Our assistant “NOVA” extends to helps us when working on a system in which it is installed. We can access by calling the wake word "Hello NOVA".

### **1.3 WHY DO WE NEED IT**

Usually, typing out and searching or doing day-to-day tasks becomes hectic. But our life doesn't need to be like that. One can ask for help to voice assistants. They let the users to perform a task using a speech command, as well as retrieve information via voice synthesis.

## 1.4 OBJECTIVE

The objective of this final year project is to design, develop, and implement an advanced voice assistant system capable of natural language understanding, seamless interaction, and task execution. The project aims to leverage state-of-the-art machine learning algorithms and speech recognition technologies to create an intelligent and user-friendly voice assistant that can perform a wide array of tasks, including but not limited to information retrieval, smart home control, scheduling, and personalized assistance. Additionally, the project seeks to explore innovative methods to enhance the voice assistant's adaptability, accuracy, and responsiveness while ensuring robustness and privacy in its operations. Evaluation metrics will focus on accuracy of speech recognition, naturalness of conversation, and overall usability of the voice assistant across various applications and user scenarios.

Main objective of building personal assistant software (a virtual assistant) is using semantic data sources available on the web, user generated content and providing knowledge from knowledge databases. The main purpose of an intelligent virtual assistant is to answer questions that users may have. This may be done in a business environment, for example, on the business website, with a chat interface. On the mobile platform, the intelligent virtual assistant is available as a call-button operated service where a voice asks the user "What can I do for you?" and then responds to verbal input. Virtual assistants can tremendously save you time. We spend hours in online research and then making the report in our terms of understanding.

Provide a topic for research and continue with your tasks while the assistant

does the research. Another difficult task is to remember test dates, birthdates or anniversaries. It comes with a surprise when you enter the class and realize it is class test today. Just tell assistant in advance about your tests and she reminds you well in advance so you can prepare for the test. One of the main advantages of voice searches is their rapidity. In fact, voice is reputed to be four times faster than a written search: whereas we can write about 40 words per minute, we are capable of speaking around 150 during the same period of time. In this respect, the ability of personal assistants to accurately recognize spoken words is a prerequisite for them to be adopted by consumers.

## 1.5 PURPOSE



**Accessibility and Inclusivity:** Create an inclusive tool accessible to users with disabilities by integrating features like text-to-speech capabilities, catering to a wider user base.

**Continuous Learning and Improvement:** Develop a system capable of learning from user interactions, improving accuracy, understanding, and response over time using machine learning algorithms and data analytics.

**Innovation in Human-Computer Interaction:** Explore innovative ways to enhance the interaction between humans and machines, potentially incorporating elements of emotional intelligence or contextual

understanding.

**Cross-Platform Integration:** Aim for compatibility across multiple platforms (desktop, web, mobile) by leveraging Python frameworks that enable seamless integration across different environments.

**Privacy and Security:** Ensure user data privacy and security by implementing encryption, secure communication protocols, and clear user consent mechanisms for data collection and storage.

**Educational and Research Value:** Contribute to the academic and research community by documenting methodologies, challenges, and insights gained during the development process, potentially aiding future projects in this domain.

**Real-World Applications:** Explore how voice assistant technology can be applied in various real-world scenarios beyond personal assistance, such as in healthcare, education, customer service, or industry-specific tasks.

## **1.6 SCOPE**

The scope of this voice assistant project involves the development of a versatile and user-friendly system utilizing Python's capabilities across multiple technical facets. This includes implementing advanced speech recognition and natural language processing techniques, employing libraries like `SpeechRecognition` and `NLTK`, enabling accurate transcription of speech to text and sophisticated understanding of user intents. Additionally,



the project aims to integrate voice synthesis functionalities using Python libraries like `gTTS` or `pyttsx3` to ensure the assistant can respond audibly. Leveraging machine learning algorithms facilitated by `scikit-learn` or custom models, the assistant will be tailored to personalize responses and improve interactions based on user preferences. The project further encompasses multi-platform compatibility, exploration of cloud integration for scalability using services like AWS or Google Cloud, and a modular architecture to support extensibility and integration of new features. Ethical considerations regarding user privacy, data handling, and responsible usage will also be addressed throughout the development process.

# SYSTEM ANALYSIS

---



## 2.1 EXISTING SYSTEM

From the above literature survey, we have inferred that all the systems existing predict only particular diseases namely lung disease, breast cancer, heart disease, diabetes by implementing various algorithms on the particular datasets.

After implementing various algorithms, the most accurate one is selected and it is used for prediction of disease. Sometimes, we may get confused of what algorithm to use. Also, all the systems find only the particular disease and not the disease based on the symptoms.

## 2.2 PROPOSED SYSTEM

We are proposing a system in an efficient way of implementing a Personal voice assistant, Speech Recognition library has many in-built functions, that will let the assistant understand the command given by user and the response will be sent back to user in voice, with Text to Speech functions. When assistant captures the voice command given by user, the underlying algorithms will convert the voice into text. And according to the keywords present in the text (command given by user), respective action will be performed by the assistant.

This is made possible with the functions present in different libraries. Also, the assistant was able to achieve all the functionalities with help of some API's. We had used these APIs for functionalities like performing calculations, extracting news from web sources, and for telling the weather. We will be sending a request, and through the API, we're getting the respective output. API's like WOLFRAMALPHA, are very helpful in performing things like calculations, making small web searches. And for getting the data from web. In this way, we are able to extract news from the web sources, and send them as input to a function for further purposes. Also, we have libraries like Random and many other libraries, each corresponding to a different technology. We used the library OS to implement Operating System related functionalities like Shutting down a system, or restarting a system.

At the outset we make our program capable of using system voice with the help of sapi5 and pyttsx3. pyttsx3 is a text-to-speech conversion library in Python. Unlike alternative libraries, it works offline, and is compatible with both Python 2 and 3. The Speech Application Programming Interface or SAPI is an API developed by Microsoft to allow the use of speech recognition and speech synthesis within Windows applications. Then we define the speak function to enable the program to speak the outputs.

After that we will define a function to take voice commands using the system microphone. The main function is then defined where all the capabilities of the program are defined.

- The proposed system will have the following functionality:

(a) The system will keep listening for commands and the time for listening is variable which can be changed according to user requirements.

(b) If the system is not able to gather information from the user input it will keep asking again to repeat till the desired number of times.

(c) The system can have both male and female voices according to user requirements.

(d) Features supported in the current version include playing music, texts, search on Wikipedia, or opening system installed applications, opening anything on the web browser, etc.

### **2.3 Software Requirements:**

2.3.1 Python 3.5 & Above

2.3.2 Windows 7 And Above

2.3.3 Microsoft Visual Studio Code

### **2.4 Hardware Requirements:**

2.4.1.1 Processor: Intel Core i5

2.4.1.2 RAM: 4GB

2.4.1.3 OS: Windows / Mac

2.4.1.4 Microphone

2.4.1.5 ARDUINO UNO board

2.4.1.6 Relay

2.4.1.7 A Light Bulb

2.4.1.8 USB Cable

2.4.1.9 Electronics Wires

2.4.1.10 Plug Point & a Plug

## **2.5 LIBRARIES:**

2.5.1.1 **Pytttsx3**- It is a text to speech conversion library in python which is used to convert the text given in the parenthesis to speech. It is compatible with python 2 and 3. An application invokes the `pytttsx3.init()` factory function to get a reference to a `pytttsx3`. it is a very easy to use tool which converts the entered text into speech. The `pytttsx3` module supports two voices first is female and the second is male which is provided by “sapi5” for windows. Command to install: - `pip install pytttsx3`

It supports three TTS engines: -

sapi5- To run on windows

nsss - NSSpeechSynthesizer on Mac OS X

espeak – eSpeak on every other platform

2.5.1.2 **Speech\_recognition**- It allows computers to understand human language. Speech recognition is a machine's ability to listen to spoken words

and identify them. We can then use speech recognition in Python to convert the spoken words into text, make a query or give a reply. Python supports many speech recognition engines and APIs, including Google Speech Engine, Google Cloud Speech API.

Command to install :- `pip install SpeechRecognition`

- 2.5.1.3 **WolframAlpha-** Wolfram Alpha is an API which can compute expert-level answers using Wolfram's algorithms, knowledgebase and AI technology. It is made possible by the Wolfram Language. The WolframAlpha API provide a web-based API allowing the computational and presentation capabilities of WolframAlpha to be integrated into web, mobile and desktop applications.

Command to install :- `pip install wolframalpha`

- 2.5.1.3.1 **Randfacts-** Randfacts is a python library that generates random facts. We can use `randfacts.get_fact()` to return a random fun fact.

Command to install :- `pip install randfacts`

- 2.5.1.3.2 **Pyjokes-** Pyjokes is a python library that is used to create one-line jokes for the users. Informally, it can also be referred as a fun python library which is pretty simple to use.

Command to install :- `pip install pyjokes`

- 2.5.1.3.3 **Datetime-** This module is used to get the date and time for the user. This is a built-in module so there is no need to install this module externally. Python Datetime module supplies classes to work with date and time. Date and datetime are an object in Python, so when we manipulate them, we are actually manipulating objects and not string or timestamps.

2.5.1.3.4 **Random2**- Python version 2 has a module named "random". This module provides a Python 3 ported version of Python 2.7's random module. It has also been back-ported to work in Python 2.6. In Python 3, the implementation of randrange() was changed, so that even with the same seed you get different sequences in Python 2 and 3.

2.5.1.3.5 **Math**- This is a built-in module which is used to perform mathematical tasks. For example, math.cos() which returns the cosine of a number or math.log() returns the natural logarithm of a number, or the logarithm of number to base.

2.5.1.3.6 **Warnings**- The warning module is actually a subclass of Exception which is a built-in class in Python. A warning in a program is distinct from an error. Conversely, a warning is not critical. It shows some message, but the program runs.

2.5.1.3.7 **OS**- The os module is a built-in module which provides functions with which the user can interact with the os when they are running the program. This module provides a portable way of using operating system-dependent functionality. This module has functions with which the user can open the file which is mentioned in the program.

2.5.1.3.8 **Serial**- This module encapsulates the access for the serial port. It provides backends for Python running on Windows, OSX, Linux, BSD and Iron Python. The module named "serial" automatically selects the appropriate backend.

Command to install :- pip install pyserial

2.2.1.1.1 **Time**- This module provides many ways of representing time in code, such as objects, numbers, and strings. It also provides

functionality other than representing time, like waiting during code execution and measuring the efficiency of our code. This is a built-in module so the installation is not necessary.

2.2.1.1.2 **Wikipedia** :-This is a Python library that makes it easy to access and parse data from Wikipedia. Search Wikipedia, get article summaries, get data like links and images from a page, and more. Wikipedia is a multilingual online encyclopedia.

Command to install :- pip install wikipedia

2.2.1.1.3 **Selenium Webdriver**- The selenium module is used to automate web browser interaction from Python. Several browsers/drivers are supported (Firefox, Chrome, Internet Explorer), as well as the Remote protocol. The supported python versions are python 3.5 and above.

Command to install :- pip install selenium

2.2.1.1.4 **Requests**- The requests module allows you to send HTTP requests using Python. The HTTP request returns a Response Object with all the response data. With it, we can add content like headers, form data, multipart files, and parameters via simple Python libraries. It also allows you to access the response data of Python in the same way.

Command to install :- pip install requests

2.2.1.1.5 **Webbrowser**- Webbrowser module is a convenient web browser controller. It provides a high-level interface that allows displaying Web-based documents to users. webbrowser can also be used as a CLI tool. It accepts a URL as the argument with the following optional parameters: -n opens the URL in a new browser window, if possible, and -t opens the URL in a new browser tab. This is a



built-in module so installation is not required.

## **2.6 PROGRAMMING LANGUAGE (PYTHON):**

Python is an OOPs (Object Oriented Programming) based, high level, interpreted programming language. It is a robust, highly useful language focused on rapid application development (RAD). Python helps in easy writing and execution of codes. Python can implement the same logic with as much as 1/5th code as compared to other OOPs languages. Python provides a huge list of benefits to all. The usage of Python is such that it cannot be limited to only one activity. Its growing popularity has allowed it to enter into some of the most popular and complex processes like Artificial Intelligence (AI), Machine Learning (ML), natural language processing, data science etc. Python has a lot of libraries for every need of this project. For this project, libraries used are speech recognition to recognize voice, Pyttsx for text to speech, selenium for web automation etc.

It's owing to the subsequent strengths that Python has –

- ✓ **Easy to be told and perceive-** The syntax of Python is simpler; thence it's comparatively straightforward, even for beginners conjointly, to be told and perceive the language.
- ✓ **Multi-purpose language** – Python could be a multi-purpose programming language as a result of it supports structured programming, object-oriented programming yet as practical programming.

**Support of open supply community** – As being open supply programming language, Python is supported by awfully giant developer community. Because of this, the bugs square measure simply mounted by the Python community. This characteristic makes Python terribly strong and adaptative

## **2.7 DOMAIN:**

The internet of things, or IoT, is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

A *thing* in the internet of things can be a person with a heart monitor implant, a farm animal with a biochip transponder, an automobile that has built-in sensors to alert the driver when tire pressure is low or any other natural or man-made object that can be assigned an Internet Protocol (IP) address and is able to transfer data over a network.

Increasingly, organizations in a variety of industries are using IoT to operate more efficiently, better understand customers to deliver enhanced customer service, improve decision-making and increase the value of the business. An IoT ecosystem consists of web-enabled smart devices that use embedded systems, such as processors, sensors and communication hardware, to collect, send and act on data they acquire from their environments.

IoT devices share the sensor data they collect by connecting to an IoT gateway or other edge device where data is either sent to the cloud to be analysed or analysed locally. Sometimes, these devices communicate with other related devices and act on the information they get from one another. The devices do most of the work without human intervention, although people can interact with the devices -- for instance, to set them up, give them instructions or access the data.

## **2.8 JUSTIFICATION OF TECHNOLOGY:**

### **1. Rich Ecosystem of Libraries and Frameworks:**

Python offers a vast array of libraries and frameworks specialized in natural language processing (NLTK, spaCy), machine learning (TensorFlow, PyTorch), and speech recognition (SpeechRecognition). This abundance facilitates rapid development, allowing seamless integration of cutting-edge functionalities into the voice assistant.

### **2. Ease of Learning and Use:**

Python's syntax is clear and readable, making it accessible for both beginners and experienced developers. Its simplicity accelerates development cycles and fosters a collaborative environment, ideal for a final year project where learning and implementation timelines are crucial.

### **3. Community Support and Documentation:**

Python boasts a robust community of developers worldwide. Abundant resources, forums, and comprehensive documentation exist, aiding in troubleshooting, learning, and implementing complex functionalities. This support is invaluable for addressing challenges encountered during the project.

### **4. Flexibility and Scalability:**

Python's versatility allows for cross-platform compatibility, enabling the voice assistant to function across various devices and operating systems. Additionally, Python's scalability, coupled with cloud integration capabilities, ensures the assistant's adaptability to handle increased workloads and future expansions.

### **5. Machine Learning and AI Capabilities:**

Python is a leading language for machine learning and artificial intelligence. Its extensive libraries and frameworks offer powerful tools for implementing personalized interactions, sentiment analysis, context understanding, and continual learning, essential for an intelligent voice assistant.

#### 6. Speed of Development:

Python's high-level abstractions and extensive libraries streamline the development process, enabling rapid prototyping and quicker iteration. This agility is advantageous in refining the voice assistant's features, enhancing its accuracy and user experience within project time constraints.

#### 7. Industry Adoption and Relevance:

Python's prevalence in industries such as AI, data science, and natural language processing demonstrates its real-world applicability. Building a voice assistant using Python equips students with skills aligned with current industry demands, enhancing employability post-graduation.

# SYSTEM DESIGN

## 3.1 SYSTEM ARCHITECTURE

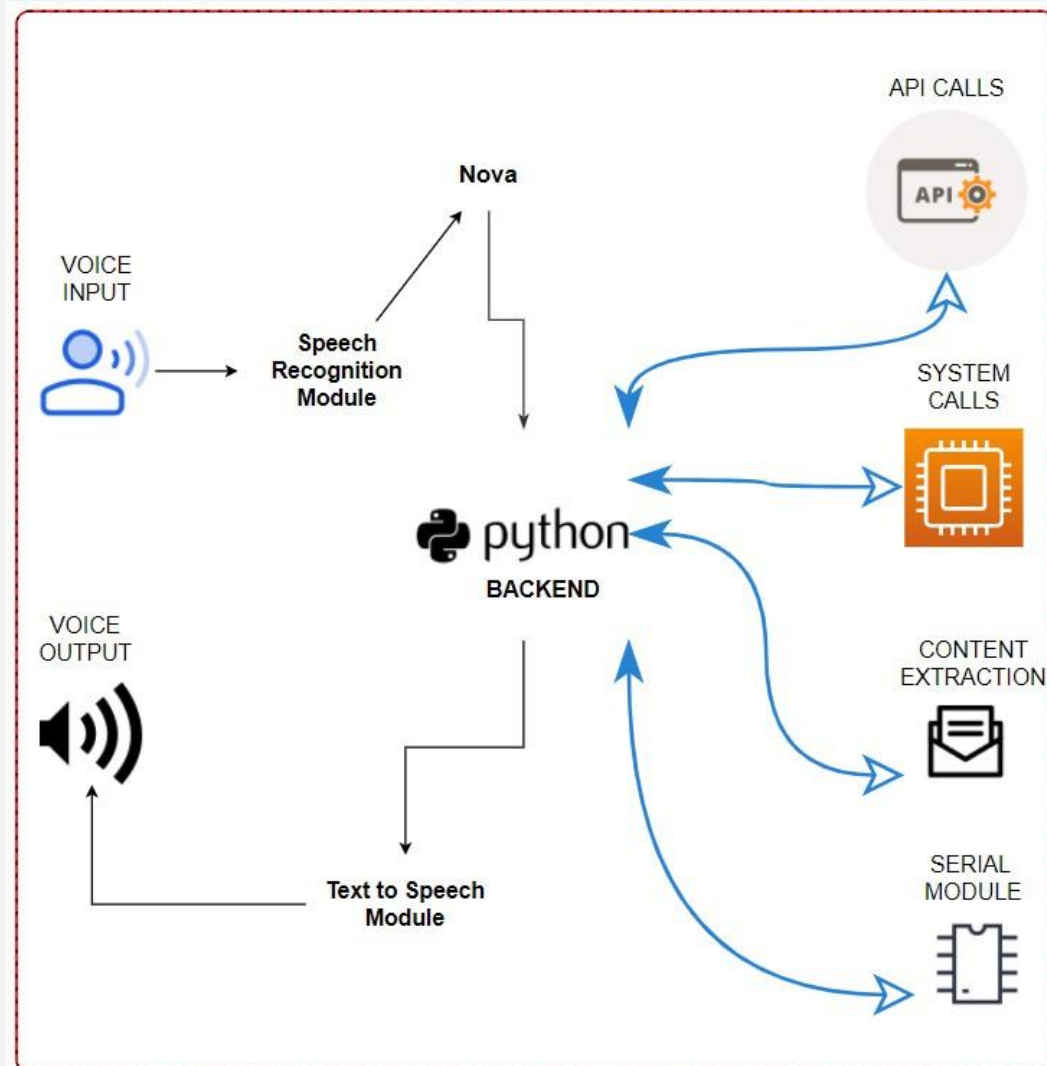


Fig 3.1 System Architecture

## 3.2 ALGORITHMS USED

### 3.2.1 SPEECH RECOGNITION MODULE



The class which we are using is called Recognizer.



It converts the audio files into text and module is used to give the output in speech.



**Energy threshold** function represents the energy level threshold for sounds. Values below this threshold are considered silence, and values above this threshold are considered speech.



#### **Recognizer**

instance.adjust\_for\_ambient\_noise(source, duration = 1), adjusts the energy threshold dynamically using audio from source (an AudioSource instance) to account for ambient noise.

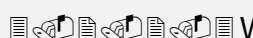
### 3.2.2 SPEECH TO TEXT & TEXT TO SPEECH CONVERSION



Pytttsx3 is a text-to-speech conversion library in Python. And can change the Voice, Rate and Volume by specific commands.



Python provides an API called Speech Recognition to allow us to convert audio into text for further processing converting large or long audio files into text using the Speech Recognition API in python.



We have Included sapi5 and espeak TTS Engines which can process the same.

### **3.2.3 PROCESS & EXECUTES THE REQUIRED COMMAND**



The said command is converted into text via speech recognition module and further stored in a temp.

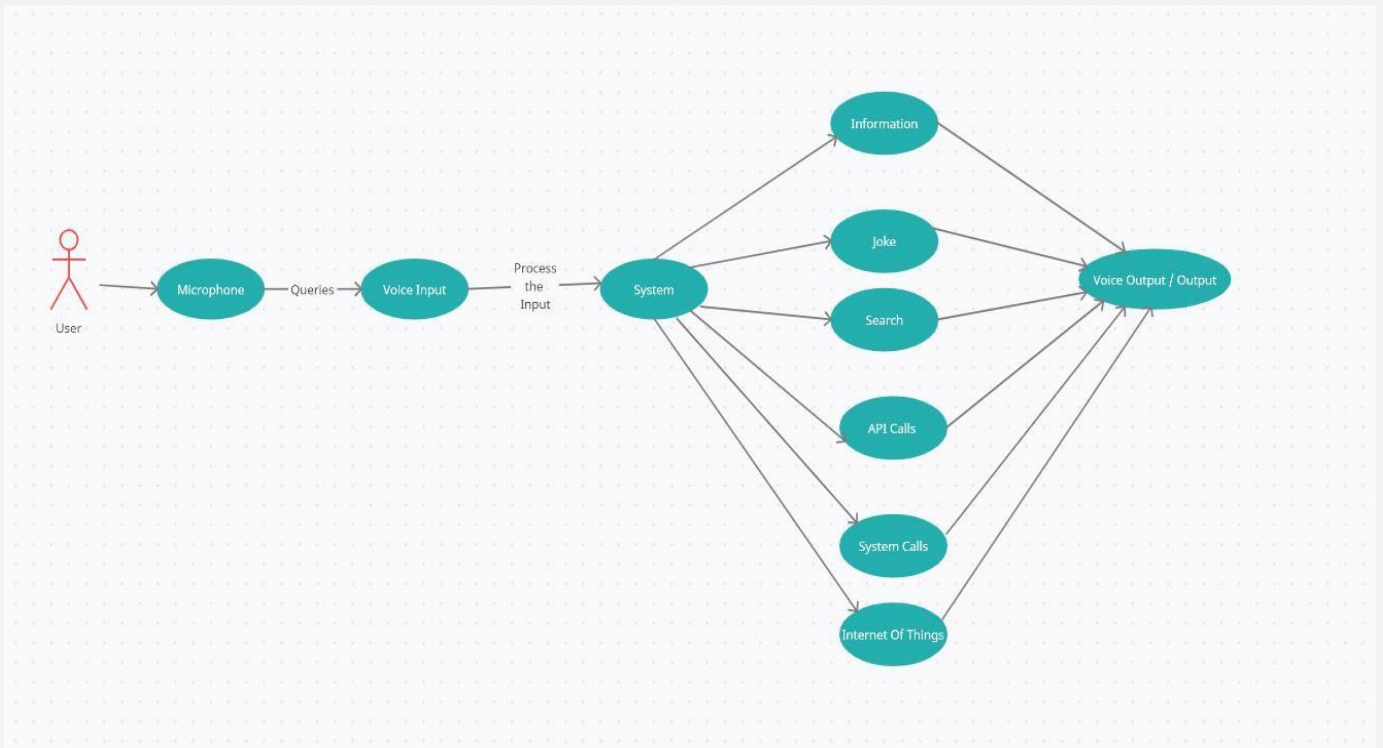


Then, Analyze the user's text via temp and decide what the user needs based on input provided and runs the while loop. Then, Commands are executed.

### 3.3 **DESIGN:**

- a) The voice assistant takes an input word which is called as "signal word" to be activated. so, it takes in the signal word and starts operating for the user commands.
- b) Converting the speech into text will be processed by the assistant.
- c) The converted text is now processed to get the required results.
- d) The text given by the user should contain one or two keywords that determine what query is to be executed. If the keyword doesn't match any of the queries in the code then the assistant asks the user to speak again.
- e) Finally, the output to the user's query will be given by converting speech to text

#### 3.3.1 **USE CASE DIAGRAM:**

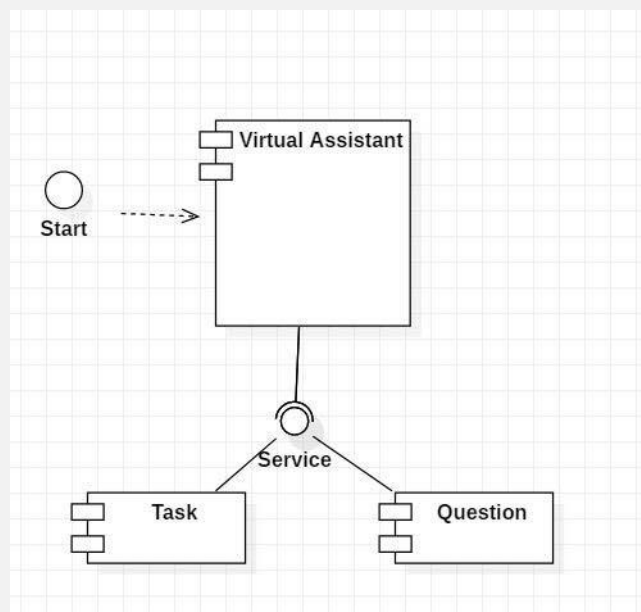




**Fig 3.2. Use Case Diagram**

- ✓ In this project there is only one user. The user queries command to the system. System then interprets it and fetches answer. The response is sent back to the user.

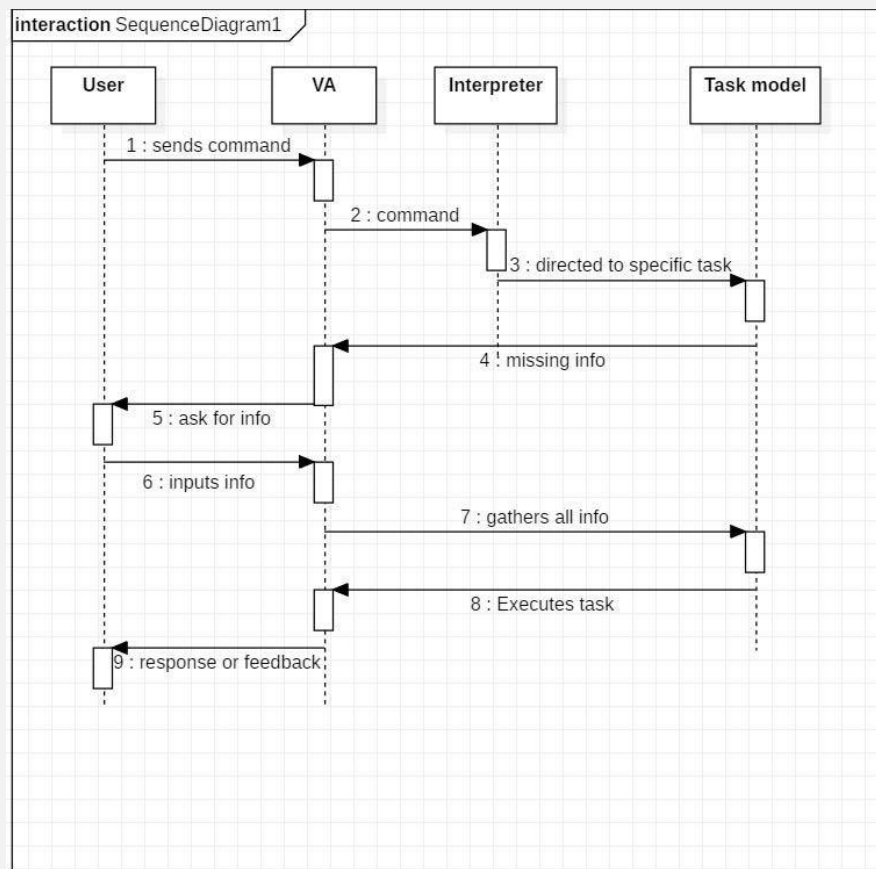
### **3.3.2 COMPONENT DIAGRAM:**



**Fig 3.3. Component Diagram**

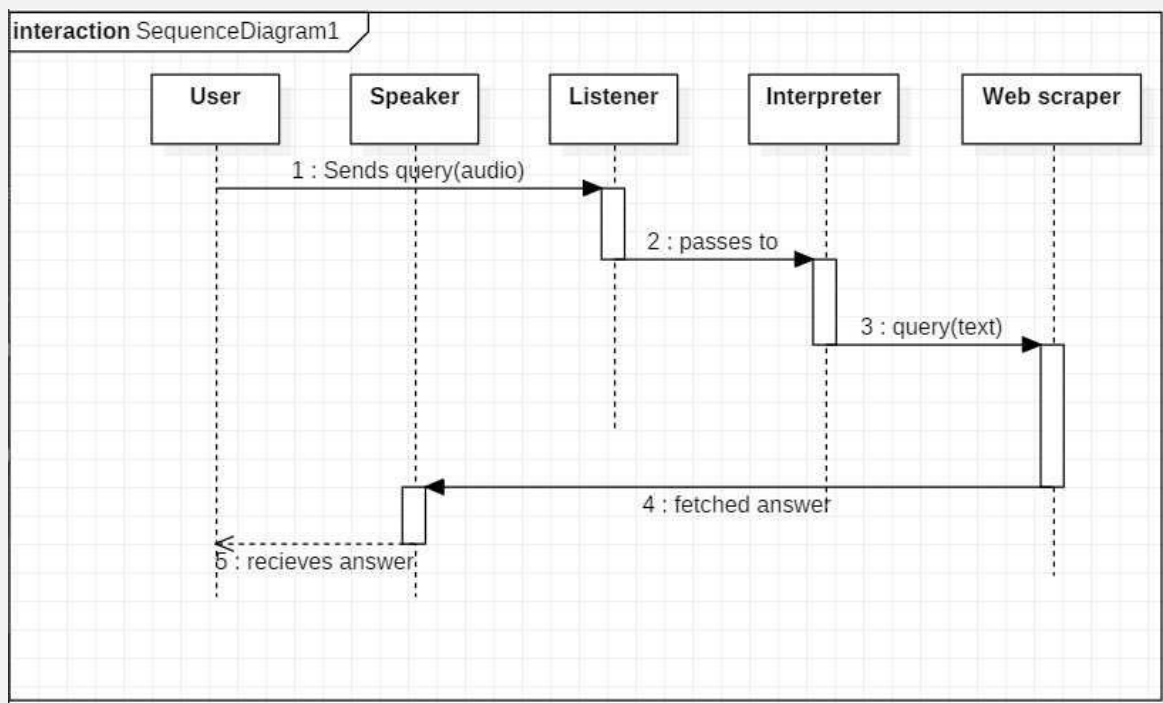
- ✓ The main component here is the Virtual Assistant. It provides two specific service, executing Task or Answering your question.

## SEQUENCE DIAGRAM:



**Fig 3.4. Sequence Diagram**

- ✓ The user sends command to virtual assistant in audio form. The command is passed to the interpreter. It identifies what the user has asked and directs it to task executor. If the task is missing some info, the virtual assistant asks user back about it. The received information is sent back to task and it is accomplished. After execution feedback is sent back to user.



**Fig 3.5. Sequence Diagram (Answering the user)**

- ✓ The above sequence diagram shows how an answer asked by the user is being fetched from internet. The audio query is interpreted and sent to Web scraper. The web scraper searches and finds the answer. It is then sent back to speaker, where it speaks the answer to user.

### 3.4 **Feasibility Study**

Feasibility study can help you determine whether or not you should proceed with your project. It is essential to evaluate cost and benefit. It is essential to evaluate cost and benefit of the proposed system. Five types of feasibility study are taken into consideration.

1. **Technical feasibility:** It includes finding out technologies for the project, both hardware and software. For virtual assistant, user must have microphone to convey their message and a speaker to listen when system speaks. These are very cheap now a days and everyone generally possess them. Besides, system needs internet connection. While using, make sure you have a steady internet connection. It is also not an issue in this era where almost every home or office has Wi-Fi.
2. **Operational feasibility:** It is the ease and simplicity of operation of proposed system. System does not require any special skill set for users to operate it. In fact, it is designed to be used by almost everyone. Kids who still don't know to write can read out problems for system and get answers.
3. **Economic feasibility:** Here, we find the total cost and benefit of the proposed system over current system. For this project, the main cost is documentation cost. User also, would have to pay for microphone and speakers. Again, they are cheap and available. As far as maintenance is concerned, it won't cost too much.
4. **Organizational feasibility:** This shows the management and organizational structure of the project. This project is not built by a team. The management tasks are all to be carried out by a single person. That won't create any management issues and will increase the feasibility of the project.

**5. Cultural feasibility:** It deals with compatibility of the project with cultural environment. Virtual assistant is built in accordance with the general culture. This project is technically feasible with no external hardware requirements. Also, it is simple in operation and does not cost training or repairs. Overall feasibility study of the project reveals that the goals of the proposed system are achievable. Decision is taken to proceed with the project.

### 3.5 TYPES OF OPERATIONS

- Information:

If we ask for some information, it opens up wikipedia and asks us the topic on which we want the information, then it clicks on the wikipedia search box using its xpath, searches the topic in the search box and clicks the search button using the xpath of the button and reads a paragraph about that topic.

Keyword: information

- Plays the video which we ask:

If we ask it to play a video, it opens up YouTube and asks us the name of the video which it wants to play. After that, it clicks on the search YouTube search box using its xpath, then it clicks on the search button using its xpath and clicks the first result of the search using the xpath of the first video.

Keyword: Play and video or music

- News of the day:

If we ask for the news, it reads out the Indian news of the day on which it is asked.

Keyword: news

- Temperature and Weather:

If the user asks the temperature, it gives the current temperature.

Keyword: temperature

- Joke:

If the user asks for a joke, it tells a one liner joke to the user.

Keyword: funny or joke

- Fact:

If the user asks for some logical fact, it tells a fact to the user.

Keyword: fact

- Game:

The assistant can play the number guessing game with the user. First, it asks for the lower and the upper limit between which the number should be. Then it initializes a random number between that upper and lower limit. After that, it uses a formula to calculate the number of turns within which the user should guess the number.

Keyword: game

- Restart the system:

The assistant restarts the system if the user asks the assistant to restart the system.

Keyword: Restart the system or Reboot the system

- Open:

The assistant will open some of the folders and applications which the user asks the assistant to open.

Keyword: Open

- Date and Time:

If the user asks for the date or time, the assistant tells it.

Keyword: date or time or date and time

- Calculate:

The assistant will calculate the equations which the user tells it to calculate using wolframalpha API key.

Keyword : calculate (along with the equation)



- Turn on the light:

This is an IOT feature where the assistant turns on the light if the user asks it to turn on the light.

Keyword: light on

- Turn off the light:

This is an IOT feature where the assistant turns off the light if the user asks it to turn off the light.

Keyword: light off

- Tells its name:

The assistant tells its name if the user asks it. The name of the assistant is Next Gen Optimal Assistant NOVA.

Keyword: Name

- Basic Conversation

You could have some basic communications with it

- Play Music

You can have it play music as well

- Send Email

You can save list of emails and have it send an Email to the person you want to mail from the list of emails

- Send Whatsapp Message

You can have it send whatsapp messages as well

- Send Spam Message

It can also send An infinite amount of spam messages to anyone you wish, good for pranking your friends

- Open websites

Have it open websites such as github, google, youtube, linkedin, etc

- Exit:

The assistant will stop assisting the user if the user asks it to exit.

Keyword: exit or end or stop.

# *RESULTS AND DISCUSSION*

---

The project work of the voice assistant has been clearly explained in this report, how useful it is and how we can rely on a voice assistant for performing any/every task which the user needs to complete and how the assistant is developing everyday which we can hope that it'll be one of the biggest technology in the current technological world. Development of the software is almost completed from our side and it's working fine as expected which was discussed for some extra development. So, maybe some advancement might come in the near future where the assistant which we developed will be even more useful than it is now.

## **4.1. WORKING**

It starts with a signal word. Users say the names of their voice assistants for the same reason. They might say, "Hey Siri!" or simply, "Alexa!" Whatever the signal word is, it wakes up the device. It signals to the voice assistant that it should begin paying attention. After the voice assistant hears its signal word, it starts to listen. The device waits for a pause to know you've finished your request. The voice assistant then sends our request over to its source code. Once in the source code, our request is compared to other requests. It's split into separate commands that our voice assistant can understand. The source code then sends these commands back to the voice assistant. Once it receives the commands, the voice assistant knows what to do next. If it understands, the voice assistant will carry out the task we asked for. For example, "Hey NOVA! What's the weather?" NOVA reports back to us in seconds. The more directions the devices receive, the better and faster they get at fulfilling our requests. The user gives the voice input through microphone and the assistant is triggered by the wake up word and performs the STT (Speech to Text) and converts it into a text and understands the Voice input and further performs the task said by the user repeatedly and delivers it via TTS (Text to Speech) module via AI Voice.

These are the important features of the voice assistant but other than this, we can do an plenty of things with the assistant

List of features that can be done with the assistant:

- Playing some video which, the user wants to see.
- Telling some random fact at the start of the day with which the user can do their work in an informative way and the user will also learn something new.
- One of the features which will be there in every assistant is playing some game so that the user can spend their free time in a fun way.
- Users might forget to turn off the system which might contain some useful data but with a voice assistant, we can do that even after leaving the place where the system is just by commanding the assistant to turn the system off.

As discussed about the mandatory features to be listed in voice assistant are implemented in this work, brief explanation is given below.

## **SYSTEM CALLS**

In this feature, we have used OS & Web Browser Module to access the desktop, calculator, task manager, command prompt & user folder. This can also restart the pc and open the chrome application.

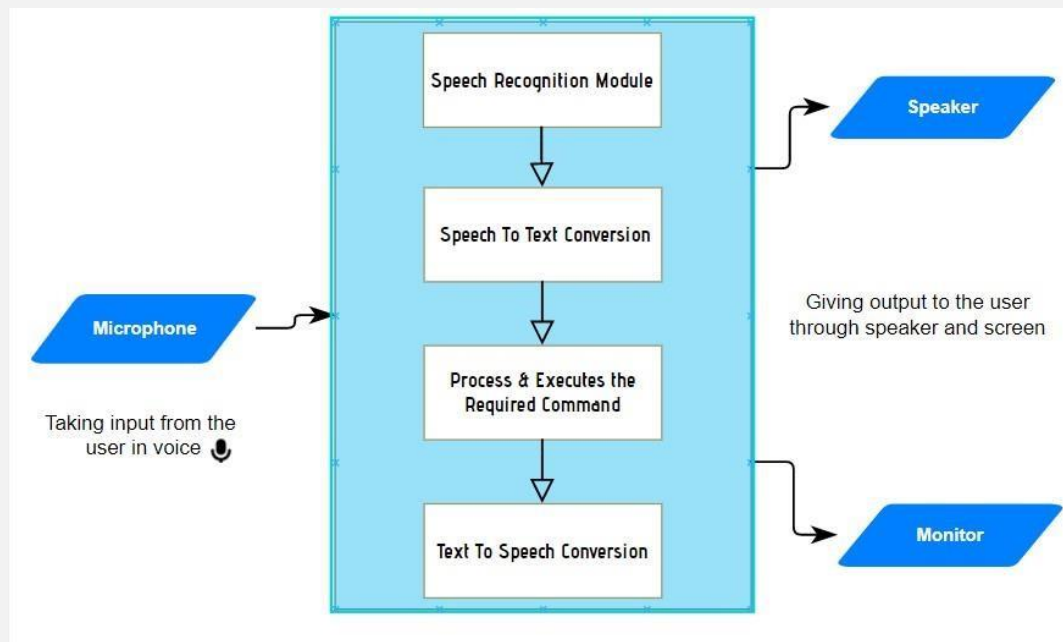
## **CONTENT EXTRATION**

This can Perform content extraction from YouTube, Wikipedia and Chrome using the web driver module from selenium which provides all the implementations for the webdrive like searching for a specific video to play, to get a specific information in google or from Wikipedia.

## **SERIAL MODULES**

Finally, we used the serial module for implementing the Internet of Things (IOT) feature for this project. It is a module which acquires the access for the serial port of the Arduino board

and used port number 11 and COM3.



**Fig 4.1. Flowchart**

- 1) Must provide the user any information which they ask for
- 2) Telling the day's hot news in the user's location
- 3) Telling some joke to chill up the moment
- 4) Playing music
- 5) Sending whatsapp messages
- 6) Sending emails
- 7) Making a note

8) Playing game

9) Opening websites such as google, youtube, github, etc

10) Playing a Youtube video

11) It can do a lot more things

## 4.2 CONCLUSION

As stated before, "voice assistant is one of the biggest problem solver" and you can see that in the proposals with the examples that it is in fact one of the biggest problem solver of the current world. We can see that voice assistant is one of the major evolving artificial intelligence in the current world once again on seeing the proposal examples because at the past, the best feature which a voice assistant had was telling the date and searching the web and giving the results but now look at the functions that it can do so with this, we can say that it is a evolving software in the current world. The main idea is to develop the assistant even more advanced than it is now and make it the best ai in the world which will save an ample of time for its users. I would like to conclude with the statement that we will try our best and give one of the best voice assistants which we are able to.



## 4.3 FUTURE SCOPE

We are entering the era of implementing voice-activated technologies to remain relevant and competitive. Voice-activation technology is vital not only for businesses to stay relevant with their target customers, but also for internal operations. Technology may be utilized to automate human operations, saving time for everyone. Routine operations, such as sending basic emails or scheduling appointments, can be completed more quickly, with less effort, and without the use of a computer, just by employing a simple voice command. People can multitask as a result, enhancing their productivity. Furthermore, relieving employees from hours of tedious administrative tasks allows them to devote more time to strategy meetings, brainstorming sessions, and other jobs that need creativity and human interaction.

### 1) Sending Emails with a voice assistant:

Emails, as we all know, are very crucial for communication because they can be used for any professional contact, and the finest service for sending and receiving emails is, as we all know, GMAIL. Gmail is a Google-created free email service. Gmail can be accessed over the web or using third-party apps that use the POP or IMAP protocols to synchronize email content.

To integrate Gmail with Voice Assistant we have to utilize Gmail API. The Gmail API allows you to access and control threads, messages, and labels in your Gmail mailbox.

## 2) Scheduling appointments using a voice assistant:

The demands on our time increase as our company grows. A growing number of people want to meet with us. We have a growing number of people who rely on us. We must check in on certain projects or set aside time to chat with possible business leads. There won't be enough hours in the day if we keep doing things the old way.

We need to get a better handle on our full-time schedule and devise a strategy for arranging appointments that doesn't interfere with our most critical job. By working with a virtual scheduler or, in other words, a virtual assistant, we let someone else worry about the organization and prioritize our schedule while we focus on the work.

## 3) Improved Interface of a voice assistant (VUI):

Voice user interfaces (VUIs) allow users to interact with a system by speaking commands. VUIs include virtual assistants like Amazon's Alexa and Apple's Siri. The real advantage of a VUI is that it allows users to interact with a product without using their hands or their eyes while focusing on anything else.

-Other benefits of a Voice user interface (VUI):

Speed and Efficiency:

Hands-free interactions are possible with VUIs. This method of interaction eliminates the need to click buttons or tap on the screen. The major means of human communication is speech. People have been using speech to form relationships for ages. As a result, solutions that allow customers to do the same are extremely valuable. Furthermore, even for experienced texters, dictating text messages has been demonstrated to be faster than typing. Hands-free interactions, at least in some circumstances, save time and boost efficiency.

Intuitiveness and convenience:

Intuitive user flow is required of high-quality VUIs, and technical advancements are expected to continue to improve the intuitiveness of voice interfaces. Compared to graphical UIs, VUIs require less cognitive effort from the user. Furthermore, everyone – from a small child to your grandmother – can communicate. As a result, VUI designers are in a better position than GUI designers, who run the danger of producing incomprehensible menus and exposing users to the agony of poor interface design. Customers are unlikely to need to be instructed on how to utilize the technology by VUI makers. People can instead ask their voice assistant for assistance.

## 4.4 SOURCE CODE

```
import pyttsx3
import speech_recognition as sr
import wikipedia
import webbrowser
import os
import datetime
from datetime import date
import random
import smtplib
import pywhatkit
import pyautogui
import datetime
import subprocess

engine = pyttsx3.init("sapi5") # ms voice api
voice = engine.getProperty("voices")
print(voice[2].id)
engine.setProperty("voice", voice[2].id)

myvoice_assistant_name = "bella"

"""Personal Variables"""
name = "Insiya"
fullname = "Insiya firoz haider rizvi"
hobby = "drawing"
course = "Bachelor of Information Technology"
prog_lang = ["python", "java", "c++"]
dob = "25 july 2003"
college_name = "SMT. JANAKIBAI RAMA SALVI COLLEGE"
place = "Thaane, Saaket"
```

```
# Python3 code to calculate age in years
def calculateAge(birthDate):
    today = date.today()
    age = (
        today.year
        - birthDate.year
        - ((today.month, today.day) < (birthDate.month, birthDate.day))
    )
    return age

"""Functions"""

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

def wishme():
    hour = int(datetime.datetime.now().hour)
    if hour >= 0 and hour <= 12:
        speak("Good morning mam")
    elif hour >= 12 and hour <= 17:
        speak("Good afternoon mam")
    elif hour >= 17 and hour <= 22:
        speak("Good evening mam")
    else:
        speak("Haven't you slept mam?")

    speak(f"my name is {myvoice_assistant_name}, please enter the valid password")
```

```

def takecommand():
    """To take microphone input from user and return it as a string"""
    r = sr.Recognizer()
    with sr.Microphone() as source:
        print("Listening...")
        r.pause_threshold = 1
        audio = r.listen(source)

    try:
        print("Recognizing...")
        query = r.recognize_google(audio, language="en-in")
        print(f"user said {query}")
        # speak(query)
    except Exception as e:
        # print(e)
        x = "I cannot hear you proper, please speak louder"
        speak(x)
        return "None"
    return query

def sendEmail(to, email_content):
    server = smtplib.SMTP("smtp.gmail.com", 587)
    server.ehlo()
    server.starttls()
    server.login("faizkhan.netpersonal.7@gmail.com", "fyjdawzgzkvajnna")
    server.sendmail("faizkhan.netpersonal.7@gmail.com", to, email_content)
    server.close()

def note(text):
    date = datetime.datetime.now()
    file_name = str(date).replace(":", "-") + "-note.txt"
    with open(file_name, "w") as f:
        f.write(text)

```

```
subprocess.Popen(["notepad.exe", file_name])
```

```
"""All current emails contact"""
```

```
emails_dict = {  
    "my main Gmail": "randygaming.net@gmail.com",  
    "my other Gmail": "nurfarzanakhan7@gmail.com",  
    "my father": "mk2223249@gmail.com",  
    "my sister": "nurfarzanakhan@gmail.com",  
}
```

```
"""All current whatsapp contacts"""
```

```
whatsapp_dic = {  
    "MI redmi number": "+91 9167162878",  
    "my father": "+968 9542 0072",  
    "my mother": "+91 70393 50250",  
}
```

```
if __name__ == "__main__":
```

```
    wishme()
```

```
    query = takecommand()
```

```
    if "2507" in query:
```

```
        speak(f"Hello {name} mam, Please tell me how may i help you")
```

```
    while True:
```

```
        query = takecommand().lower()
```

```
        """Logics for executing tasks"""
```

```
    def basiconvo():
```

```
        # Personal Datas
```

```

if "my name" in query:
    speak(f"mam, your name is {name}")
elif "my full name" in query:
    speak(f"your full name is {fullname}")
elif "my age" in query:
    speak(f"mam, you are {calculateAge(date(2003, 7, 25))} years old")
elif "college name" in query:
    speak(
        f"Your college name is {college_name} and it is in Manisha
nagar, kaalwaa"
    )
elif "about me" in query:
    speak(
        f"{fullname} is a student and has a hobby of {hobby}, she was
born on {dob}, and is currently persuing {course} \
        from {college_name}she has a knowledge of programming
languages like {prog_lang}, but mainly work on {prog_lang[0]} \
        .she stays in {place}"
    )

# -----

elif "how are you" in query:
    speak("i'm fine mam, what about you")
# elif "who created you" or "who made you" in query:
#     speak(f"i was created by {name}")
elif "what is your name" in query:
    speak(f"my name is {myvoice_assistant_name}")
elif "who are you" in query:
    speak(f"I am {myvoice_assistant_name}, a virtual assistant")
elif "i am fine" in query: # W
    speak("i'm glad you are fine")
elif "ok" in query or "good" in query:
    speak("is there anything else i can do for you, mam")
elif "nope" in query: # need work

```



```
    speak("Okay mam")
elif "can you sing" in query: # W
    speak("no, i am a bad singer")
elif "how old are you" in query or "what is your age" in query: # W
    speak("i am 1 month old")
elif "how many languages" in query:
    speak("as of now i can only speak english")
elif "humans" in query:
    speak("yes i love humans")
elif "are you a robot" in query:
    speak("i am your virtual assistant")
elif "correct" in query:
    speak("see, i knew it")
elif "not fine" in query:
    speak("oh, i hope you feel better, is there anything i can do?")
elif (
    "can you be my girlfriend" in query
    or "can you be my boyfriend" in query
):
    speak("sorry, i am already taken")
elif "love me" in query:
    speak("ofcourse i do")
elif "where are you from" in query:
    speak("i am from your imagination")
elif "is love" in query:
    speak("it is the 7th sense that destroy all other senses")
elif "favourite colour" in query:
    speak("my favourite color is blue because i love the sky")
elif f"{myvoice_assistant_name}" in query:
    speak("yes mam, what can i do for you")
elif "thank you" in query:
    speak("my pleasure mam")
elif "job" in query:
    speak("thank you mam!")
```

```
# calling the basic conversation
basiconvo()

if "wikipedia" in query:
    try:
        speak("searching wikipedia...")
        query = query.replace("wikipedia", "")
        result = wikipedia.summary(query, sentences=2)
        speak("According to wikipedia")
        print(result)
        speak(result)
    except:
        speak("Sorry! i couldn't find it for some reason")

elif "open youtube" in query:
    speak("opening youtube")
    # webbrowser.open("https://www.youtube.com/")
    url = "https://www.youtube.com"
    chrome_path = (
        "C:\\Program Files\\Google\\Chrome\\Application\\chrome.exe"
    )
    webbrowser.register(
        "chrome", None, webbrowser.BackgroundBrowser(chrome_path)
    )
    webbrowser.get("chrome").open_new_tab(url)

elif "close chrome" in query:
    speak("closing chrome browser")
    os.system("taskkill /im firefox.exe /f")
    os.system("taskkill /im chrome.exe /f")

elif "open google" in query:
    speak("opening google")
```

```
# webbrowser.open("http://google.com")
urL = "https://www.google.com"
chrome_path = (
    "C:\\Program Files\\Google\\Chrome\\Application\\chrome.exe"
)
webbrowser.register(
    "chrome", None, webbrowser.BackgroundBrowser(chrome_path)
)
webbrowser.get("chrome").open_new_tab(urL)
```

elif "open github" in query:

```
speak("opening github")
urL = "https://github.com/"
chrome_path = (
    "C:\\Program Files\\Google\\Chrome\\Application\\chrome.exe"
)
webbrowser.register(
    "chrome", None, webbrowser.BackgroundBrowser(chrome_path)
)
webbrowser.get("chrome").open_new_tab(urL)
```

elif "open vs code" in query:

```
vs_path = "C:\\Users\\User\\AppData\\Local\\Programs\\Microsoft VS
Code\\Code.exe"
speak("opening vs code")
os.startfile(vs_path)
```

elif "make a note" in query:

```
speak("what would you like me to write down?")
notewrite = takecommand()
note(notewrite)
```

```
speak("I've made a note of that.")
```

elif "play music" in query:

```
try:
    music_dir = "C:\\Users\\User\\OneDrive\\Documents\\Faiz Khan
Program\\Insiya SEM 6 Project\\mymusic"
    songs = os.listdir(music_dir)
    print(songs)
    random = os.startfile(os.path.join(music_dir, songs[0]))
except Exception as e:
    speak("Sorry mam, unable to play music")

elif "test auto search" in query:
    """Need works"""
    import pyautogui
    import time

    speak("testing it")
    # print(pyautogui.position())

    pyautogui.click(498, 744, 1, 1, "left")
    pyautogui.click(497, 449, 1, 1, "left", duration=1)
    pyautogui.leftClick(601, 417, duration=1)

    speak("What do you wish to type mam?")
    input_query_insearchbar = takecommand()

    pyautogui.typewrite(input_query_insearchbar, interval=1)

    pyautogui.press("enter")

elif "google search" in query:
    speak("what do you wish to search mam")
    ques = takecommand()
    speak(f"searching {ques} on google")
    pywhatkit.search(ques)

elif "send an email" in query:
```

```

try:
    speak("who do you wish to send an email to, mam?")
    mail_name = takecommand()
    if mail_name in emails_dict.keys():
        speak("okay, What should i say?")
        email_content = takecommand()
        to = emails_dict.get(mail_name)
        sendEmail(to, email_content)
        speak("email has been sent!")
    else:
        speak("sorry! This email is not available in my memory")
except Exception as e:
    print(e)
    speak("sorry, i wasn't able to send the email")

elif (
    "open whatsapp" in query or "open Whatsapp" in query
): # need to change pyauto gui position
    speak("Roger that! mam")

    pyautogui.click(466, 752, 1, 1, "left")
    # pyautogui.click(697, 434, 1, 1, 'left', duration=1)

    # pyautogui.click(727, 631, 1, 1, 'left', duration=1)

elif "whatsapp message" in query: # need works
    try:
        speak("who do you wish to message mam?")
        whatsapp_name = takecommand()
        if whatsapp_name in whatsapp_dic.keys():
            test = whatsapp_dic.get(whatsapp_name)
            from datetime import timedelta, time

            # import time
            speak("okay, what should i say?")

```

```

        my_message = takecommand()
        # time.sleep(3)
        time1 = datetime.datetime.now()
        minu = 2
        t = time(time1.hour, time1.minute)
        # result = datetime.datetime.combine(date.today(), t) +
timedelta(minutes=minu)
        x = time1.strftime("%I")
        print("\n")
        speak(
            f"seonding message by {x} o'clock and {time1.minute + minu}
minutes"
        )
        print(
            f"sending message by {x} o'clock and {time1.minute + minu}
minutes"
        )

        pywhatkit.sendwhatmsg(
            test, my_message, time1.hour, time1.minute + minu
        )
        speak("Message has been sent!")
    except:
        speak("Sorry mam, i was not able to send the message")

elif "spam message" in query:
    """need works here"""
    import time

    try:
        # speak('Who do you want to spam mam?')
        # spamname = takecommand()
        # if spamname in whatsapp_dic.keys():
        speak("okay, and what will be the spam message?")
        spammsg = takecommand()

```

```
# spammsg = input('enter: ')
speak("ok, how many times to you want to spam it?")
num_ofspam = takecommand()

# message = input("Enter the message: ")
# num_value = input("Enter num of times: ")
# speak('roger that!')
abc = int(num_ofspam)

# pyautogui.click(612, 745, 1, 1, 'left')
# pyautogui.click(697, 434, 1, 1, 'left', duration=1)

# pyautogui.click(727, 631, 1, 1, 'left', duration=1)

# pyautogui.click(289, 253, 1, 1, 'left', duration=7)

# pyautogui.click(610, 693, 1, 'left', duration=1)
speak(f"Sending spam messages")
time.sleep(15)

for i in range(abc):
    pyautogui.typewrite(spammsg)
    pyautogui.press("Enter")

    speak("spam messages has been sent!")
except Exception as e:
    # print(e)
    speak("sorry! unable to send spam messages")

elif "search on youtube" in query:
    speak("what topic do you want to search for on youtube?")
    topic = takecommand()
    speak(f"okay, playing a random video on the topic {topic}")
    pywhatkit.playonyt(topic)
```

elif "the time" in query:

```
strtime = datetime.datetime.now().strftime("%H:%M:%S")
speak(f"The time is {strtime}")
```

elif "the date" in query:

```
date1 = date.today()
print(date1)
speak(date1)
```

elif "sleep" in query:

```
speak("Ok then, i am going to sleep, bye")
quit()
```

elif "joke" in query:

```
import pyjokes

speak(pyjokes.get_joke())
```

elif "close current window" in query:

```
speak("Closing the current window mam!")
pyautogui.click(1332, 20, 1, 1, "left")
```

elif "lock window" in query:

```
speak("Locking the window mam")
import ctypes
```

```
# Define the Windows API function
```

```
user32 = ctypes.windll.user32
```

```
LockWorkStation = user32.LockWorkStation
```

```
# Call the LockWorkStation function to lock the PC
```

```
LockWorkStation()
```

elif "shutdown" in query or "shut down" in query:

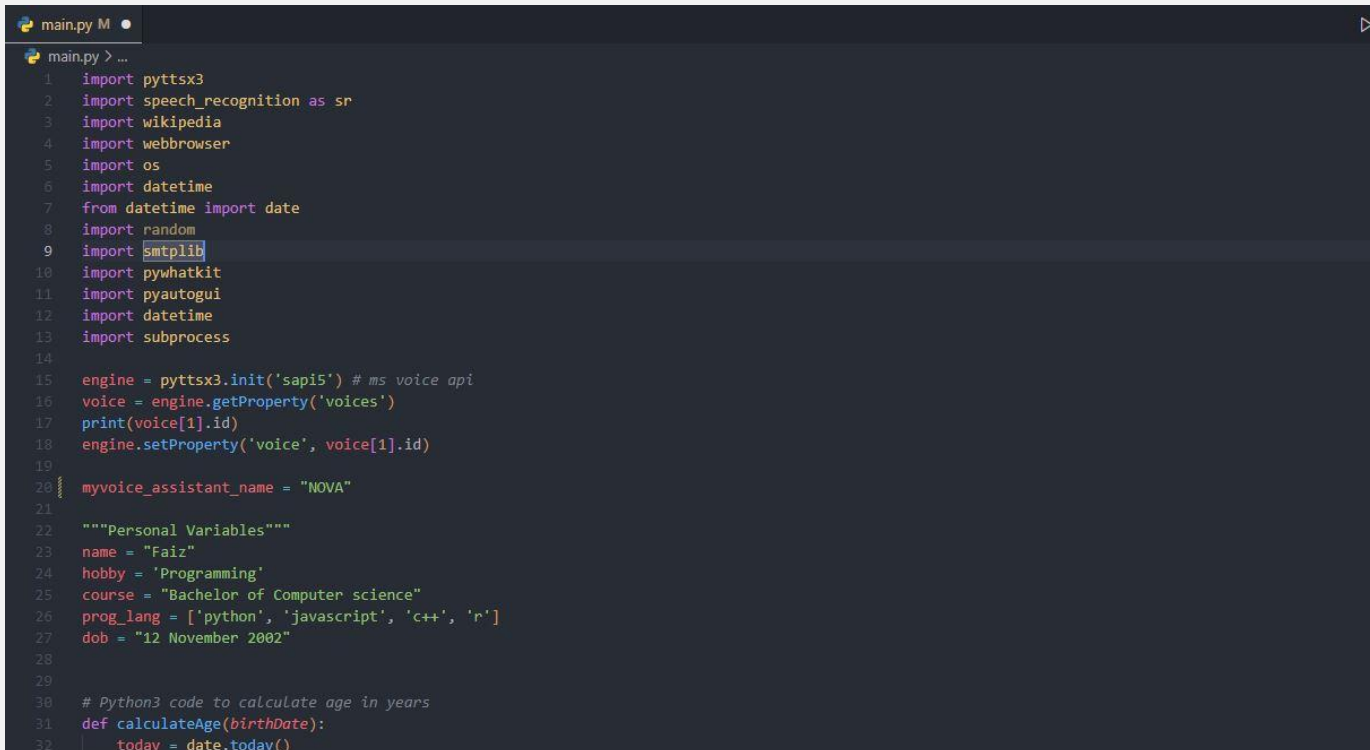
```
speak("mam, are you sure you want to shut down ?")
```



```
surety = takecommand()
if surety == "yes" or "yeah" in query:
    os.system("shutdown /s /t 1")
else:
    speak("shut down cancelled")

else:
    speak("I'm sorry, You entered the wrong password")
    quit()
```

## 4.5 SCREENSHOTS



```
main.py M
main.py > ...
1  import pyttsx3
2  import speech_recognition as sr
3  import wikipedia
4  import webbrowser
5  import os
6  import datetime
7  from datetime import date
8  import random
9  import smtplib
10 import pywhatkit
11 import pyautogui
12 import datetime
13 import subprocess
14
15 engine = pyttsx3.init('sapi5') # ms voice api
16 voice = engine.getProperty('voices')
17 print(voice[1].id)
18 engine.setProperty('voice', voice[1].id)
19
20 myvoice_assistant_name = "NOVA"
21
22 """Personal Variables"""
23 name = "Faiz"
24 hobby = 'Programming'
25 course = "Bachelor of Computer science"
26 prog_lang = ['python', 'javascript', 'c++', 'r']
27 dob = "12 November 2002"
28
29
30 # Python3 code to calculate age in years
31 def calculateAge(birthDate):
32     today = date.today()
```

```

main.py M
main.py > ...
41
42 """Functions"""
43
44 def speak(audio):
45     engine.say(audio)
46     engine.runAndWait()
47
48 def wishme():
49     hour = int(datetime.datetime.now().hour)
50     if hour >= 0 and hour <= 12:
51         speak("Good morning sir")
52     elif hour >= 12 and hour <= 17:
53         speak("Good afternoon sir")
54     elif hour >= 17 and hour <= 22:
55         speak("Good evening sir")
56     else:
57         speak("Haven't you slept sir?")
58
59     speak(f"my name is {myvoice_assistant_name}, please enter the valid password")
60
61
62
63 def takecommand():
64     """To take microphone input from user and return it as a string"""
65     r = sr.Recognizer()
66     with sr.Microphone() as source:
67         print("Listening...")
68         r.pause_threshold = 1
69         audio = r.listen(source)
70
71     try:
72         print("Recognizing...")

```

```

main.py M
main.py > ...
83 def sendEmail(to, email_content):
84     server = smtplib.SMTP('smtp.gmail.com', 587)
85     server.ehlo()
86     server.starttls()
87     server.login('faizkhan.netpersonal.7@gmail.com', 'fyjdawzgzkvajnna')
88     server.sendmail('faizkhan.netpersonal.7@gmail.com', to, email_content)
89     server.close()
90
91 def note(text):
92     date = datetime.datetime.now()
93     file_name = str(date).replace(":", "-") + "-note.txt"
94     with open(file_name, "w") as f:
95         f.write(text)
96
97     subprocess.Popen(["notepad.exe", file_name])
98
99
100 """All current emails contact"""
101 emails_dict = {
102     "my main Gmail": "randygaming.net@gmail.com",
103     "my other Gmail": "nurfarzanakhan7@gmail.com",
104     "my father": "mk2223249@gmail.com",
105     "my sister": "nurfarzanakhan@gmail.com"
106 }
107
108 """All current whatsapp contacts"""
109 whatsapp_dic = {
110     "MI redmi number" : "+91 9167162878",
111     "my father" : "+968 9542 0072",
112     "my mother" : "+91 70393 50250",
113     "my sister" : "+91 91520 81174",
114 }

```

```

main.py M
main.py > ...
127
128 def basiconvo():
129
130     # Personal Datas
131
132     if 'my name' in query:
133         speak(f'sir, your name is {name}')
134     elif 'my full name' in query:
135         speak(f'your full name is {fullname}')
136     elif 'my age' in query:
137         speak(f'sir, you are {calculateAge(date(2002, 11, 12))} years old')
138     elif 'about me' in query:
139         speak(f'{fullname} is a student and a learning {hobby} who was born on {dob}, and is currently persuing {course} \
140             he has a knowledge of programming languages like {prog_lang}, but mainly work on {prog_lang[0]} and has created \
141             many projects using Python')
142     elif 'mother name' in query:
143         speak(my_mother)
144     elif 'father name' in query:
145         speak(my_father)
146     elif 'sister name' in query:
147         speak(my_sister)
148
149     # -----
150
151     elif 'how are you' in query:
152         speak("i'm fine sir, what about you")
153     elif 'who created you' in 'who made you' in query:
154         speak("i was created by faiz khan")
155     elif 'what is your name' in query:
156         speak(f"my name is {myvoice_assistant_name}")
157     elif 'who are you' in query:
158         speak(f"I am jarvis, a virtual assistant")
159     elif 'i am fine' in query:
160         #W

```

```

main.py M
main.py > ...
151
152     elif 'how are you' in query:
153         speak("i'm fine sir, what about you")
154     elif 'who created you' in 'who made you' in query:
155         speak("i was created by faiz khan")
156     elif 'what is your name' in query:
157         speak(f"my name is {myvoice_assistant_name}")
158     elif 'who are you' in query:
159         speak(f"I am jarvis, a virtual assistant")
160     elif 'i am fine' in query:
161         #W
162         speak("i'm glad you are fine")
163     elif "ok" in query or "good" in query:
164         speak("is there anything else i can do for you, sir")
165     elif "nope" in query: # need work
166         speak("Okay sir")
167     elif 'can you sing' in query:
168         #W
169         speak("no, i am a bad singer")
170     elif 'how old are you' in query or 'what is your age' in query:
171         #W
172         speak("i am 1 years old")
173     elif 'how many languages' in query:
174         speak("as of now i can only speak english")
175     elif 'humans' in query:
176         speak("yes i love humans")
177     elif 'are you a robot' in query:
178         speak("i am your virtual assistant")
179     elif 'correct' in query:
180         speak("see, i knew it")
181     elif 'not fine' in query:
182         speak("oh, i hope you feel better, is there anything i can do?")
183     elif 'can you be my girlfriend' in query or 'can you be my boyfriend' in query:
184         speak("sorry, i am already taken")
185     elif 'love me' in query:
186         speak("ofcourse i do")
187     elif 'where are you from' in query:

```

```

main.py M
main.py > ...
235     webbrowser.get('chrome').open_new_tab(url)
236
237
238     elif 'close chrome' in query:
239         speak("closing chrome browser")
240         os.system("taskkill /im firefox.exe /f")
241         os.system("taskkill /im chrome.exe /f")
242
243     elif 'open google' in query:
244         speak("opening google")
245         # webbrowser.open("http://google.com")
246         url='https://www.google.com'
247         chrome_path="C:\\Program Files\\Google\\Chrome\\Application\\chrome.exe"
248         webbrowser.register('chrome', None,webbrowser.BackgroundBrowser(chrome_path))
249         webbrowser.get('chrome').open_new_tab(url)
250
251     elif 'open github' in query:
252         speak("opening github")
253         url="https://github.com/"
254         chrome_path="C:\\Program Files\\Google\\Chrome\\Application\\chrome.exe"
255         webbrowser.register('chrome', None,webbrowser.BackgroundBrowser(chrome_path))
256         webbrowser.get('chrome').open_new_tab(url)
257
258     elif 'open vs code' in query:
259         vs_path = "C:\\Users\\User\\AppData\\Local\\Programs\\Microsoft VS Code\\Code.exe"
260         speak("opening vs code")
261         os.startfile(vs_path)
262
263     elif 'make a note' in query:
264         speak('what would you like me to write down?')
265         notewrite = takecommand()
266         note(notewrite)
267
268         speak("I've made a note of that.")

```

```

main.py M
main.py > ...
330
331
332     elif 'whatsapp message' in query: # need works
333         try:
334             speak('who do you wish to message sir?')
335             whatsapp_name = takecommand()
336             if whatsapp_name in whatsapp_dic.keys():
337                 test = whatsapp_dic.get(whatsapp_name)
338                 from datetime import timedelta, time
339                 # import time
340                 speak("okay, what should i say?")
341                 my_message = takecommand()
342                 # time.sleep(3)
343                 time1 = datetime.datetime.now()
344                 minu = 2
345                 t = time(time1.hour, time1.minute)
346                 # result = datetime.datetime.combine(date.today(), t) + timedelta(minutes=minu)
347                 x = time1.strftime("%I")
348                 print('\n')
349                 speak(f"seonding message by {x} o'clock and {time1.minute + minu} minutes")
350                 print(f"sending message by {x} o'clock and {time1.minute + minu} minutes")
351
352                 pywhatkit.sendwhatmsg(test, my_message, time1.hour, time1.minute + minu)
353                 speak('Message has been sent!')
354             except:
355                 speak('Sorry sir, i was not able to send the message')
356
357
358     elif 'spam message' in query:
359         '''need works here'''
360         import time
361         try:

```

```

main.py M
main.py > ...
395
396
397     elif 'search on youtube' in query:
398         speak('what topic do you want to search for on youtube?')
399         topic = takecommand()
400         speak(f"okay, playing a random video on the topic {topic}")
401         pywhatkit.playonyt(topic)
402
403
404     elif 'the time' in query:
405         strtime = datetime.datetime.now().strftime("%H:%M:%S")
406         speak(f"The time is {strtime}")
407
408     elif 'the date' in query:
409         date1 = date.today()
410         print(date1)
411         speak(date1)
412
413     elif 'sleep' in query:
414         speak("Ok then, i am going to sleep, bye")
415         quit()
416
417     elif 'joke' in query:
418         import pyjokes
419         speak(pyjokes.get_joke())
420
421     elif 'close current window' in query:
422         speak('Closing the current window sir!')
423         pyautogui.click(1332, 20, 1, 1, 'left')
424
425     elif 'lock window' in query:
426         speak('Locking the window sir')
427         import ctypes

```

```

main.py M
main.py > ...
422         speak('Closing the current window sir!')
423         pyautogui.click(1332, 20, 1, 1, 'left')
424
425     elif 'lock window' in query:
426         speak('Locking the window sir')
427         import ctypes
428         # Define the Windows API function
429         user32 = ctypes.windll.user32
430         LockWorkStation = user32.LockWorkStation
431
432         # Call the LockWorkStation function to lock the PC
433         LockWorkStation()
434
435
436
437     elif 'shutdown' in query or 'shut down' in query:
438         speak('sir, are you sure you want to shut down?')
439         surety = takecommand()
440         if surety == 'yes' or 'yeah' in query:
441             os.system('shutdown /s /t 1')
442         else:
443             speak('shut down cancelled')
444
445     else:
446         speak("I'm sorry, You entered the wrong password")
447         quit()
448
449
450
451

```



# IMPLEMENTATION AND TESTING

---



## 5.1. FEASIBILITY TESTING

Feasibility testing is a critical phase to evaluate the practicality and viability of implementing the Advanced Virtual Voice Assistant. This testing phase ensures that the proposed functionalities and features align with the project's objectives and technical capabilities.

The feasibility testing for this project involved assessing the following aspects:

- **Technical Feasibility:**

Examined whether the required technologies and libraries were available and compatible for the implementation.

In this phase, we assessed the availability and compatibility of the required technologies and libraries essential for the implementation of the virtual assistant. Compatibility checks were conducted to ensure that different modules and libraries seamlessly integrated with each other. This included validating the compatibility of speech recognition libraries, automation tools, and web interaction modules. The technical feasibility assessment confirmed that the chosen technologies were readily accessible and harmonized effectively within the project's scope.

- **Operational Feasibility:**

Evaluated the practicality of integrating the virtual assistant into users' daily routines and assessed potential user acceptance.

Operational feasibility testing gauged the practicality of integrating the virtual assistant into users' daily routines. User acceptance and adaptability were key considerations during this evaluation. The project team conducted surveys and user interviews to understand potential user preferences, expectations, and potential challenges. The positive feedback received during operational feasibility testing reinforced the belief that the virtual assistant could become an integral and user-friendly part of individuals' daily lives.

- **Economic Feasibility:**

Considered the cost implications of implementing and maintaining the virtual assistant, including hardware, software, and maintenance.

Economic feasibility assessments focused on evaluating the cost implications associated with implementing and maintaining the virtual assistant. This included considerations for hardware, software



licenses, and ongoing maintenance costs. The analysis revealed that the project's economic feasibility was sound, with a reasonable investment required for the development and maintenance of the virtual assistant compared to the anticipated benefits and functionalities it would provide.

- **Legal and Ethical Feasibility:**

Ensured compliance with legal and ethical standards, particularly concerning user privacy and responsible use of automation.

The results of the feasibility testing confirmed that the project was both technically and operationally feasible, with minimal economic and ethical concerns.

Legal and ethical considerations are paramount in any software development project, particularly when dealing with automation and user interactions. The project team conducted a thorough review to ensure compliance with privacy regulations and ethical standards. Special attention was given to user data handling, spam message automation, and web scraping to ensure responsible and ethical use of the virtual assistant. The legal and ethical feasibility assessment verified that the project aligned with regulatory requirements and ethical standards.

The comprehensive feasibility testing demonstrated that the project was both technically and operationally feasible, with minimal economic and ethical concerns.

## **5.2. TESTING APPROACH**

Testing is a crucial aspect of ensuring the reliability and effectiveness of the Advanced Virtual Voice Assistant. The testing approach encompassed a systematic process, including unit testing and integration testing.

Testing is a critical phase in the development lifecycle of the Advanced Virtual Voice Assistant, serving as a cornerstone for ensuring the reliability, stability, and effectiveness of the system. The primary objective of testing is to identify and rectify any issues or anomalies within the software, ultimately guaranteeing a robust and user-friendly virtual assistant.

### **5.2.1. UNIT TESTING**

Unit testing involved the rigorous examination of individual components or modules of the virtual assistant in isolation. This phase aimed to verify the correctness and functionality of each feature and module.

Key aspects of unit testing included:

- **Voice Recognition Accuracy**

Voice recognition accuracy was a paramount concern to ensure the virtual assistant accurately understood and interpreted user commands. The team employed various test scenarios, including different accents, tones, and speech patterns, to evaluate the system's robustness in voice recognition.

- **Functionality of Each Feature**

Each feature of the virtual assistant underwent thorough testing to ensure its proper execution. For instance, sending emails, playing music, and browsing the web were meticulously tested to guarantee their functionality. Test cases were designed to cover a wide range of inputs and scenarios, ensuring the virtual assistant's versatility across diverse tasks.

- **Error Handling**

The unit testing phase placed a strong emphasis on error handling mechanisms. The team systematically tested the system's response to unexpected inputs, ensuring that appropriate error messages were provided, and the virtual assistant gracefully handled unexpected scenarios without compromising the user experience.

The unit testing phase not only verified the reliability of individual

components but also laid the foundation for seamless integration.

### **5.2.2. INTEGRATION TESTING**

Following successful unit testing, the testing approach progressed to integration testing. This phase focused on evaluating the interaction and collaboration between different modules and features of the virtual assistant. Integration testing is crucial for identifying potential issues that may arise from the interdependence of various functionalities.

Integration testing focused on evaluating the interaction and collaboration between different modules and features of the virtual assistant. This phase aimed to identify potential issues arising from the integration of various functionalities.

Key aspects of integration testing included:

- **Communication Between Modules**

Smooth communication between different components of the virtual assistant was critical for cohesive functionality. Integration tests were designed to ensure that modules exchanged information seamlessly and that data integrity was maintained throughout various stages of execution.

- **Data Flow and Integrity**

Ensuring the proper flow of data between modules was a priority during integration testing. The team meticulously verified that data moved efficiently between different components and that the integrity of information was preserved throughout the system.

- **Cross-Functionality Testing**

To guarantee consistent performance across various tasks, cross-functionality testing was conducted. This involved examining how different features interacted with each other and ensuring that the virtual assistant maintained a high level of performance and responsiveness.

The integration testing phase successfully validated the seamless integration of features, ensuring a cohesive and fully functional virtual assistant.

# CONCLUSION AND FEATURE WORK

---

## 6.1. CONCLUSION



The development of the Advanced Virtual Voice Assistant marks a significant achievement in the realm of virtual assistant technology. This project aimed to create a versatile and user-friendly assistant, capable of seamlessly integrating into users' daily lives while offering a diverse range of features. As we conclude this project, it is essential to reflect on the journey, accomplishments, and potential future enhancements.

- Achievements

The Advanced Virtual Voice Assistant successfully implements a plethora of features, ranging from communication and entertainment to productivity and web interaction. The feasibility testing phase validated the project's technical, operational, economic, and ethical viability. The unit testing and integration testing processes ensured the reliability and effectiveness of individual components and their cohesive integration.

Key achievements of the project include:

- Robust Voice Recognition: The voice recognition system demonstrated accuracy and adaptability, allowing users to interact with the assistant using natural language and diverse speech patterns.
- Diverse Feature Set: The assistant's extensive feature set, including sending emails, playing music, browsing the web, and interacting in a conversational manner, adds a layer of versatility and usefulness for users.
- Testing Rigor: The thorough unit testing and integration testing processes validated the functionality of the virtual assistant, addressing potential issues and enhancing its overall reliability.

- **User Feedback**

Throughout the development process, user feedback played a crucial role in refining and improving the virtual assistant. Beta testing with a diverse user group provided valuable insights into user preferences, expectations, and areas for improvement. Positive feedback on ease of use, voice recognition accuracy, and the assistant's ability to perform various tasks affirmed the project's success.

- **Challenges and Lessons Learned**

Challenges encountered during development, such as ensuring accurate voice recognition in various environments and handling unexpected user inputs, provided valuable learning experiences. Addressing these challenges required iterative improvements and optimizations, ultimately contributing to the project team's growth and expertise in voice assistant technology.



## **6.2. SYSTEM IMPLEMENTATION AND ENHANCEMENT**

The implementation of the Advanced Virtual Voice Assistant involved the integration of various Python modules and libraries, creating a robust foundation for its functionalities. As we consider future directions and enhancements, several areas can be explored to further elevate the assistant's capabilities and user experience.

- **Enhanced Voice Recognition**

Improving voice recognition capabilities can significantly enhance the user experience. Exploring advanced machine learning models, incorporating natural language processing (NLP) techniques, and adapting to user-specific speech patterns are avenues for refining and optimizing the voice recognition system.

- **Expansion of Features**

To cater to a broader user base, the inclusion of additional features and integrations is an exciting prospect. Collaboration with popular platforms, incorporation of smart home device control, and expansion into specialized domains such as healthcare or education could broaden the assistant's utility.

- **Personalization and User Profiles**

Implementing user profiles and personalization features could elevate the virtual assistant's ability to tailor its responses and actions based on individual preferences. This could include learning from user interactions, adapting to specific user needs, and providing a more personalized and efficient experience over time.

- **Natural Language Understanding (NLU)**

Advancing natural language understanding capabilities would enable the assistant to comprehend and respond to user queries more intelligently. Integrating sophisticated NLU models could enhance the system's contextual understanding and improve its ability to engage in meaningful and dynamic conversations.

- **Accessibility and Inclusivity**

Ensuring the virtual assistant is accessible to users with diverse needs is paramount. Implementing accessibility features, such as support for multiple languages, accommodating different accents and dialects, and incorporating assistive technologies, would contribute to a more inclusive and user-friendly

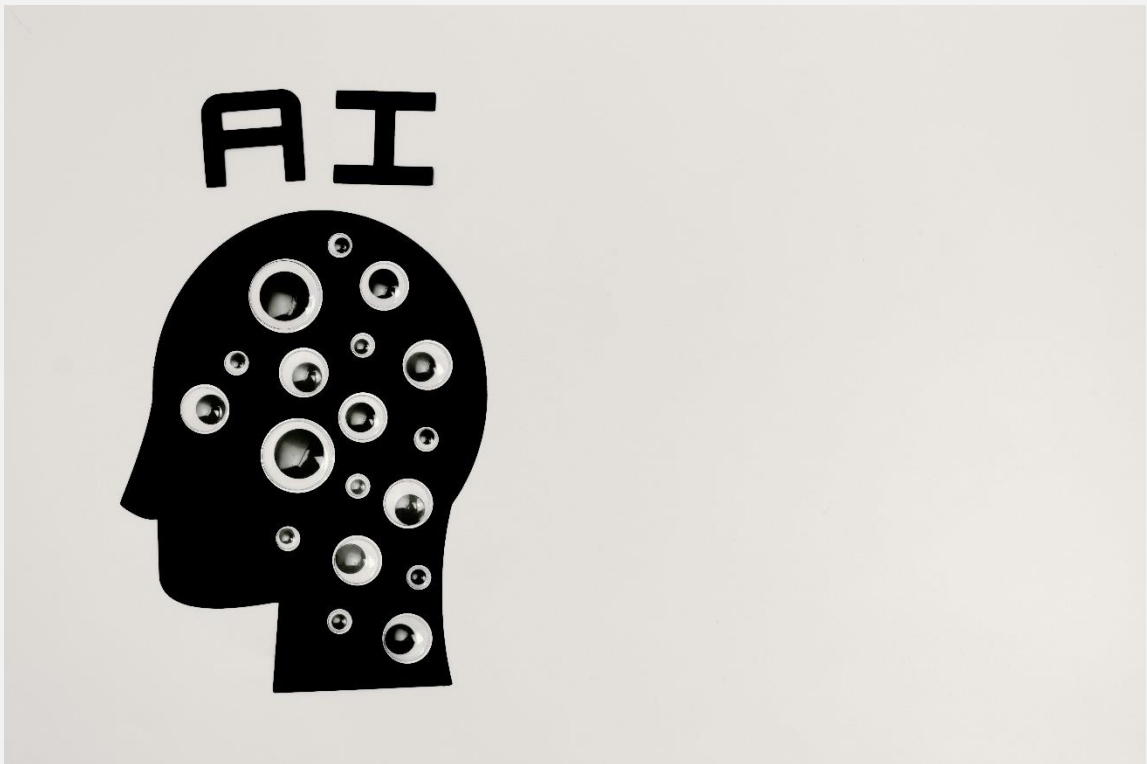
experience.

- **Continuous Improvement through User Feedback**

Establishing a feedback loop with users and regularly updating the virtual assistant based on their input is essential for continuous improvement. This iterative approach ensures that the assistant stays relevant, adaptive, and aligned with evolving user expectations.

- **Final Remarks**

In conclusion, the Advanced Virtual Voice Assistant project has not only delivered a feature-rich and functional virtual assistant but has also paved the way for future enhancements. As technology evolves, the potential for innovation in voice assistant systems is vast. The journey from project inception to implementation has been both challenging and rewarding, and the continuous pursuit of excellence remains at the core of this endeavor.



# REFERENCES

---

The development and implementation of the Advanced Virtual Voice Assistant project drew upon a variety of resources, libraries, and technologies. The following references acknowledge and credit the sources that contributed to the successful realization of this project:

1. Python Software Foundation. (n.d.). Python.

[<https://www.python.org/>](<https://www.python.org/>)

2. SpeechRecognition Library. (n.d.). SpeechRecognition Documentation.

[<https://pypi.org/project/SpeechRecognition/>](<https://pypi.org/project/SpeechRecognition/>)

3. pytsx3 Library. (n.d.). pytsx3 Documentation.

[<https://pypi.org/project/pytsx3/>](<https://pypi.org/project/pytsx3/>)

4. smtplib Library. (n.d.). Python Documentation - smtplib.

[<https://docs.python.org/3/library/smtplib.html>](<https://docs.python.org/3/library/smtplib.html>)

5. pywhatkit Library. (n.d.). pywhatkit Documentation.

[<https://pypi.org/project/pywhatkit/>](<https://pypi.org/project/pywhatkit/>)

6. pyautogui Library. (n.d.). pyautogui Documentation.

[<https://pypi.org/project/PyAutoGUI/>](<https://pypi.org/project/PyAutoGUI/>)

7. wikipedia Library. (n.d.). Wikipedia-API Documentation.

[<https://pypi.org/project/Wikipedia-API/>](<https://pypi.org/project/Wikipedia-API/>)

8. webbrowser Module. (n.d.). Python Documentation - webbrowser.

[<https://docs.python.org/3/library/webbrowser.html>](<https://docs.python.org/3/library/webbrowser.html>)

9. selenium Library. (n.d.). Selenium Documentation.

[<https://www.selenium.dev/documentation/en/>](<https://www.selenium.dev/documentation/en/>)

10. pytube Library. (n.d.). pytube Documentation.

[<https://pypi.org/project/pytube/>](<https://pypi.org/project/pytube/>)

These references encompassed various Python modules and libraries that played a pivotal role in achieving the functionalities of the virtual assistant. Each resource provided valuable documentation and support, contributing to the project's success.

Furthermore, the development team consulted additional online resources, forums, and tutorials to address specific challenges and implement best practices throughout the project.

These collective resources formed the foundation for the Advanced Virtual Voice Assistant, demonstrating the collaborative and dynamic nature of the software development process.

As a testament to the open-source community, the development team expresses gratitude to the authors, contributors, and maintainers of the referenced libraries and tools. Their dedication to sharing knowledge and fostering innovation greatly enriched the project's development journey.

In addition to the specific references listed, the development team acknowledges the vast body of knowledge available through online forums, community discussions, and educational platforms that contributed to the overall understanding and successful implementation of the Advanced Virtual Voice Assistant.

The completion of this project has been made possible through the collective efforts of the open-source community and the commitment to knowledge sharing in the field of software development





♥INSIYA RIZVI♥