****

# Graphic Era (Deemed to be University)

##### PROJECT REPORT ON

**VIRTUAL ASSISTANT**

**(Anny)**

**SUBMITTED BY**

**Akash Kumar**

****

**MCA SEM IV [2020-2022]**



Ref No : Date :

**Certificate**

###### This is to certify that the project entitled VIRTUAL ASSISTANT is undertaken at the GRAPHIC ERA (Deemed to be University) by AKASH KUMAR in partial fulfillment of MCA (Semester IV ) Examination had not been submitted for any other examination and does not form part of any other course undergone by the candidate.

It is further certified that he has completed all required phases of project.

Signature of Internal Guide Signature of External

HOD/In -Charge/Co-ordinator

**Acknowledgement**

In completing this project report on project titled **VIRTUAL ASSISTANT**, I had to take the help and guideline of a few respected people, who deserve my greatest gratitude.

The completion of this project report gives me much Pleasure. I would like to show my gratitude to **Dr. Preeti** for giving me a good guideline for project throughout numerous consultations. I would also like to expand my deepest gratitude to all those who have directly and indirectly guided us in writing this project report.

Many people, especially my classmates and friends themselves, have made valuable comments and suggestions on this proposal which gave me inspiration to improve my project. Here I thank all the people for their help directly and indirectly to complete this project report.

The author

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Sr.No** | **Title** | **Page No** |
| 1 | Introduction | 1 |
|  | 1.1 Background | 2 |
| 1.2 Objectives | 4 |
| 1.3 Purpose, Scope and Applicability | 5 |
| 2 | Survey of Technology | 6 |
| 3 | Requirement and Analysis | 8 |
|  | 3.1 Problem Definition | 8 |
| 3.2 Requirement Specification | 9 |
| 3.3 Software and Hardware Requirement | 11 |
| 4 | System Design | 12 |
|  | 4.1 ER Diagram | 12 |
| 4.2 Activity Diagram | 13 |
| 4.3 Class Diagram | 14 |
| 4.4 Use Case Diagram | 15 |
| 4.5 Sequence Diagram | 16 |
| 4.6 Data Flow Diagram | 18 |
| 4.7 Component Diagram | 21 |
| 4.8 Deployment Diagram | 22 |
| 4.9 Data Dictionary | 23 |
| 4.10 Test Case Design | 24 |
|  | Reference and Bibliography | 26 |

**VIRTUAL ASSISTANT**

# INTRODUCTION

In today’s era almost all tasks are digitalized. We have Smartphone in hands and it is nothing less than having world at your finger tips. These days we aren’t even using fingers. We just speak of the task and it is done. There exist systems where we can say Text Dad, “I’ll be late today.” And the text is sent. That is the task of a Virtual Assistant. It also supports specialized task such as booking a flight, or finding cheapest book online from various e- commerce sites and then providing an interface to book an order are helping automate search, discovery and online order operations.

Virtual Assistants are software programs that help you ease your day to day tasks, such as showing weather report, creating reminders, making shopping lists etc. They can take commands via text (online chat bots) or by voice. Voice based intelligent assistants need an invoking word or wake word to activate the listener, followed by the command. For my project the wake word is Anny. We have so many virtual assistants, such as Apple’s Siri, Amazon’s Alexa and Microsoft’s Cortana. For this project, wake word was chosen Anny.

This system is designed to be used efficiently on desktops. Personal assistant software improves user productivity by managing routine tasks of the user and by providing information from online sources to the user. Anny is effortless to use. Call the wake word ‘Anny’ followed by the command. And within seconds, it gets executed.

Voice searches have dominated over text search. Web searches conducted via mobile devices have only just overtaken those carried out using a computer and the analysts are already predicting that 50% of searches will be via voice by 2021.Virtual assistants are turning out to be smarter than ever. Allow your intelligent assistant to make email work for you. Detect intent, pick out important information, automate processes, and deliver personalized responses.

This project was started on the premise that there is sufficient amount of openly available data and information on the web that can be utilized to build a virtual assistant that has access to making intelligent decisions for routine user activities.

### BACKGROUND

There already exist a number of desktop virtual assistants. A few examples of current virtual assistants available in market are discussed in this section along with the tasks they can provide and their drawbacks.

SIRI from Apple

SIRI is personal assistant software that interfaces with the user thru voice interface, recognizes commands and acts on them. It learns to adapt to user’s speech and thus improves voice recognition over time. It also tries to converse with the user when it does not identify the user request.

It integrates with calendar, contacts and music library applications on the device and also integrates with GPS and camera on the device. It uses location, temporal, social and task based contexts, to personalize the agent behavior specifically to the user at a given point of time.

Supported Tasks

* Call someone from my contacts list
* Launch an application on my iPhone
* Send a text message to someone
* Set up a meeting on my calendar for 9am tomorrow
* Set an alarm for 5am tomorrow morning
* Play a specific song in my iTunes library
* Enter a new note

Drawback

SIRI does not maintain a knowledge database of its own and its understanding comes from the information captured in domain models and data models.

ReQall

ReQall is personal assistant software that runs on smartphones running Apple iOS or Google Android operating system. It helps user to recall notes as well as tasks within a location and time context. It records user inputs and converts them into commands, and monitors current stack of user tasks to proactively suggest actions while considering any changes in the environment. It also presents information based on the context of the user, as well as filter information to the user based on its learned understanding of the priority of that information.

Supported Tasks

* Reminders
* Email
* Calendar, Google Calendar
* Outlook
* Evernote
* Facebook, LinkedIn
* News Feeds

Drawback

Will take some time to put all of the to-do items in – you could spend more time putting the entries in than actually doing the revision.

### OBJECTIVES

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

Virtual assistants can tremendously save you time. We spend hours in online research and then making the report in our terms of understanding. JIA can do that for you. Provide a topic for research and continue with your tasks while JIA does the research. Another difficult task is to remember test dates, birthdates or anniversaries. It comes with a surprise when you enter the class and realize it is class test today. Just tell JIA in advance about your tests and she reminds you well in advance so you can prepare for the test.

One of the main advantages of voice searches is their rapidity. In fact, voice is reputed to be four times faster than a written search: whereas we can write about 40 words per minute, we are capable of speaking around 150 during the same period of time15. In this respect, the ability of personal assistants to accurately recognize spoken words is a prerequisite for them to be adopted by consumers.

### PURPOSE, SCOPE AND APPILCABILITY

##### Purpose

Purpose of virtual assistant is to being capable of voice interaction, music playback, making to-do lists, setting alarms, streaming podcasts, playing audiobooks, and providing weather, traffic, sports, and other real-time information, such as news. Virtual assistants enable users to speak natural language voice commands in order to operate the device and its apps.

There is an increased overall awareness and a higher level of comfort demonstrated specifically by millennial consumers. In this ever-evolving digital world where speed, efficiency, and convenience are constantly being optimized, it’s clear that we are moving towards less screen interaction.

##### Scope

Voice assistants will continue to offer more *individualized* experiences as they get better at differentiating between voices. However, it’s not just developers that need to address the complexity of developing for voice as brands also need to understand the capabilities of each device and integration and if it makes sense for their specific brand. They will also need to focus on maintaining a user experience that is consistent within the coming years as complexity becomes more of a concern. This is because the visual interface with voice assistants is missing. Users simply cannot see or touch a voice interface.

##### Applicability

The mass adoption of artificial intelligence in users’ everyday lives is also fueling the shift towards voice. The number of IoT devices such as smart thermostats and speakers are giving voice assistants more utility in a connected user’s life. Smart speakers are the number one way we are seeing voice being used. Many industry experts even predict that nearly every application will integrate voice technology in some way in the next 5 years.

The use of virtual assistants can also enhance the system of IoT (Internet of Things). Twenty years from now, Microsoft and its competitors will be offering personal digital assistants that will offer the services of a full-time employee usually reserved for the rich and famous.

# SURVEY OF TECHNOLOGY

##### Python

Python is an OOPs (Object Oriented Programming) based, high level, interpreted programming language. It is a robust, highly useful language focused on rapid application development (RAD). Python helps in easy writing and execution of codes. Python can implement the same logic with as much as 1/5th code as compared to other OOPs languages.

Python provides a huge list of benefits to all. The usage of Python is such that it cannot be limited to only one activity. Its growing popularity has allowed it to enter into some of the most popular and complex processes like Artificial Intelligence (AI), Machine Learning (ML), natural language processing, data science etc. Python has a lot of libraries for every need of this project. For Anny, libraries used are speechrecognition to recognize voice, Pyttsx for text to speech, Wikipedia for web search results , os for operating system commnads like sleep and power off etc.

Python is reasonably efficient. Efficiency is usually not a problem for small examples. If your Python code is not efficient enough, a general procedure to improve it is to find out what is taking most the time, and implement just that part more efficiently in some lower-level language. This will result in much less programming and more efficient code (because you will have more time to optimize) than writing everything in a low-level language.

##### Pyttsx

Pyttsx stands for Python Text to Speech. It is a cross-platform Python wrapper for text- to-speech synthesis. It is a Python package supporting common text-to-speech engines on Mac OS X, Windows, and Linux. It works for both Python2.x and 3.x versions. Its main advantage is that it works offline. it is a very easy to use tool which converts the entered text into speech.The pyttsx3 module supports two voices first is female and the second is male which is provided by “sapi5” for windows.

**Installataion**  
To install the pyttsx3 module, first of all, you have to open the terminal and write

pip install pyttsx3

##### Speech Recognition

This is a library for performing speech recognition, with support for several engines and APIs, online and offline*.* It supports APIs like Google Cloud Speech API, IBM Speech to Text, Microsoft Bing Voice Recognition etc.Speech recognition is the process of converting spoken words to text. Python supports many speech recognition engines and APIs, including Google Speech Engine, Google Cloud Speech API, Microsoft Bing Voice Recognition and IBM Speech to Text.

**Installation**

A library that helps is named “SpeechRecognition”. You should install it with pyenv, pipenv or virtualenv. You can also install it system wide:

|  |
| --- |
| pip install SpeechRecognition |

SQLite

SQLite is a capable library, providing an in-process relational database for efficient storage of small-to-medium-sized data sets. It supports most of the common features of SQL with few exceptions. Best of all, most Python users do not need to install anything to get started working with SQLite, as the standard library in most distributions ships with the sqlite3 module.

SQLite runs embedded in memory alongside your application, allowing you to easily extend SQLite with your own Python code. SQLite provides quite a few hooks, a reasonable subset of which are implemented by the standard library database driver.

# REQUIREMENT AND ANALYSIS

System Analysis is about complete understanding of existing systems and finding where the existing system fails. The solution is determined to resolve issues in the proposed system. It defines the system. The system is divided into smaller parts. Their functions and inter relation of these modules are studied in system analysis. The complete analysis is followed below.

### Problem definition

Usually, user needs to manually manage multiple sets of applications to complete one task. For example, a user trying to make a travel plan needs to check for airport codes for nearby airports and then check travel sites for tickets between combinations of airports to reach the destination. There is need of a system that can manage tasks effortlessly.

We already have multiple virtual assistants. But we hardly use it. There are number of people who have issues in voice recognition. These systems can understand English phrases but they fail to recognize in our accent. Our way of pronunciation is way distinct from theirs. Also, they are easy to use on mobile devices than desktop systems. There is need of a virtual assistant that can understand English in Indian accent and work on desktop system.

When a virtual assistant is not able to answer questions accurately, it’s because it lacks the proper context or doesn’t understand the intent of the question. Its ability to answer questions relevantly only happens with rigorous optimization, involving both humans and machine learning. Continuously ensuring solid quality control strategies will also help manage the risk of the virtual assistant learning undesired bad behaviors. They require large amount of information to be fed in order for it to work efficiently.

Virtual assistant should be able to model complex task dependencies and use these models to recommend optimized plans for the user. It needs to be tested for finding optimum paths when a task has multiple sub-tasks and each sub-task can have its own sub-tasks. In such a case there can be multiple solutions to paths, and the it should be able to consider user preferences, other active tasks, priorities in order to recommend a particular plan.

### REQUIREMENT SPECIFICATION

Personal assistant software is required to act as an interface into the digital world by understanding user requests or commands and then translating into actions or recommendations based on agent’s understanding of the world.

JIA focuses on relieving the user of entering text input and using voice as primary means of user input. Agent then applies voice recognition algorithms to this input and records the input. It then use this input to call one of the personal information management applications such as task list or calendar to record a new entry or to search about it on search engines like Google, Bing or Yahoo etc. Focus is on capturing the user input through voice, recognizing the input and then executing the tasks if the agent understands the task. Software takes this input in natural language, and so makes it easier for the user to input what he or she desires to be done.

Voice recognition software enables hands free use of the applications, lets users to query or command the agent through voice interface. This helps users to have access to the agent while performing other tasks and thus enhances value of the system itself. JIA also have ubiquitous connectivity through Wi-Fi or LAN connection, enabling distributed applications that can leverage other APIs exposed on the web without a need to store them locally.

Virtual assistants must provide a wide variety of services. These include:

* Providing information such as weather, facts from e.g. Wikipedia etc.
* Set an alarm or make to-do lists and Many More Things.
* Remind you of birthdays and meetings.
* Play music from services such as Youtube and Music Player.
* Play videos, TV shows or movies on televisions, from e.g. Netflix or Hotstar youtube & web-browser
* Send Email .

##### Feasibility Study

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

1. **Technical feasibility:** It includes finding out technologies for the project, both hardware and software. For virtual assistant, user must have microphone to convey their message and a speaker to listen when system speaks. These are very cheap now a days and everyone generally possess them. Besides, system needs internet connection. While using JIA, make sure you have a steady internet connection. It is also not an issue in this era where almost every home or office has Wi-Fi.
2. **Operational feasibility:** It is the ease and simplicity of operation of proposed system. System does not require any special skill set for users to operate it. In fact, it is designed to be used by almost everyone. Kids who still don’t know to write can read out problems for system and get answers.
3. **Economical feasibility:** Here, we find the total cost and benefit of the proposed system over current system. For this project, the main cost is documentation cost. User also would have to pay for microphone and speakers. Again, they are cheap and available. As far as maintenance is concerned, JIA won’t cost too much.
4. **Organizational feasibility:** This shows the management and organizational structure of the project. This project is not built by a team. The management tasks are all to be carried out by a single person. That won’t create any management issues and will increase the feasibility of the project.
5. **Cultural feasibility:** It deals with compatibility of the project with cultural environment. Virtual assistant is built in accordance with the general culture. The project is named JIA so as to represent Indian culture without undermining local beliefs.

This project is technically feasible with no external hardware requirements. Also it is simple in operation and does not cost training or repairs. Overall feasibility study of the project reveals that the goals of the proposed system are achievable. Decision is taken to proceed with the project.

### HARDWARE AND SOFTWARE REQUIREMENTS

The software is designed to be light-weighted so that it doesn’t be a burden on the machine running it. This system is being build keeping in mind the generally available hardware and software compatibility. Here are the minimum hardware and software requirement for virtual assistant.

##### Hardware:

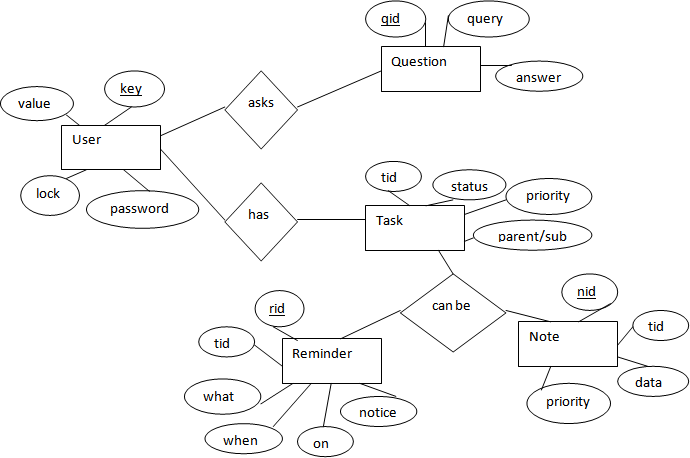
* Pentium-pro processor or later.
* RAM 512MB or more .
* 

##### Software:

* Windows 7(32-bit) or above.
* Python 2.7 or later
* Chrome Driver
* Selenium Web Automation
* Pyaudio installed
* Speech modulue in window

# SYSTEM DESIGN

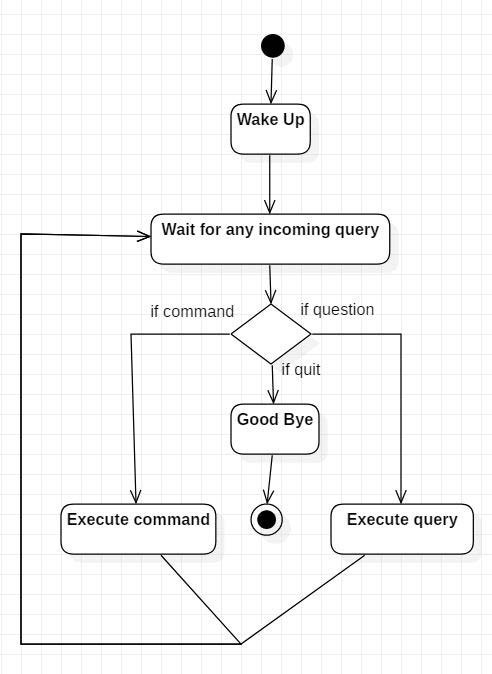
### ER DIAGRAM



The above diagram shows entities and their relationship for a virtual assistant system. We have a user of a system who can have their keys and values. It can be used to store any information about the user. Say, for key “name” value can be “Jim”. For some keys user might like to keep secure. There he can enable lock and set a password (voice clip).

Single user can ask multiple questions. Each question will be given ID to get recognized along with the query and its corresponding answer. User can also be having n number of tasks. These should have their own unique id and status i.e. their current state. A task should also have a priority value and its category whether it is a parent task or child task of an older task.

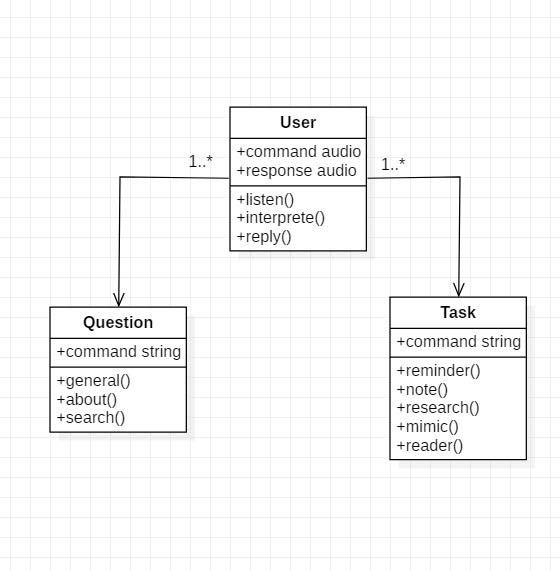
### ACTIVITY DIAGRAM



Initially, the system is in idle mode. As it receives any wake up cal it begins execution.

The received command is identified whether it is a questionnaire or a task to be performed. Specific action is taken accordingly. After the Question is being answered or the task is being performed, the system waits for another command. This loop continues unless it receives quit command. At that moment, it goes back to sleep.

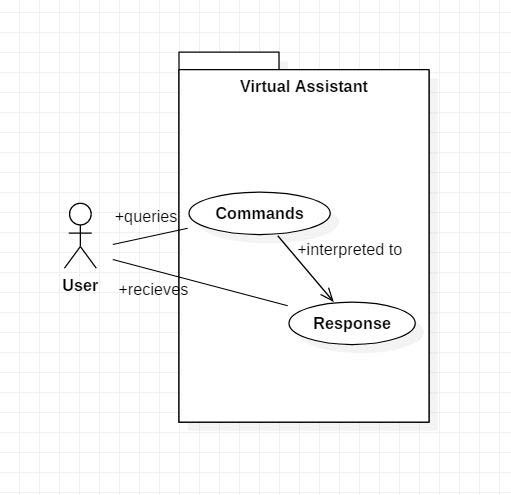
### CLASS DIAGRAM



The class user has 2 attributes command that it sends in audio and the response it receives which is also audio. It performs function to listen the user command. Interpret it and then reply or sends back response accordingly. Question class has the command in string form as it is interpreted by interpret class. It sends it to general or about or search function based on its identification.

The task class also has interpreted command in string format. It has various functions like reminder, note, mimic, research and reader.

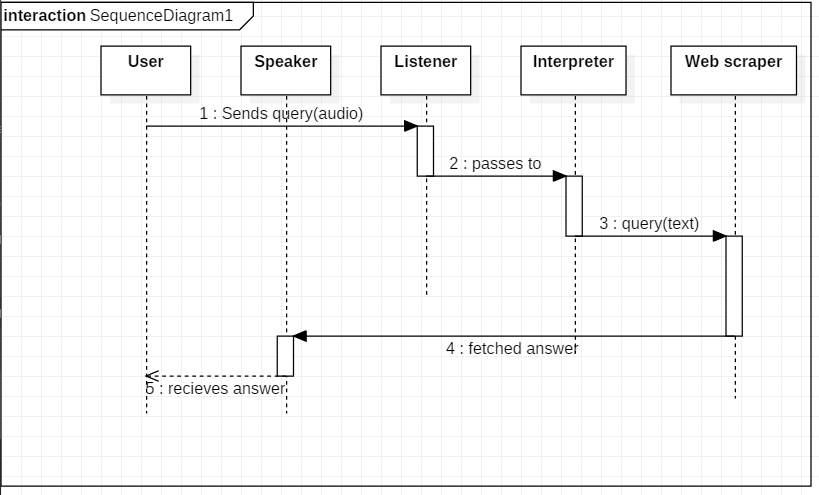
### USE CASE DIAGRAM



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

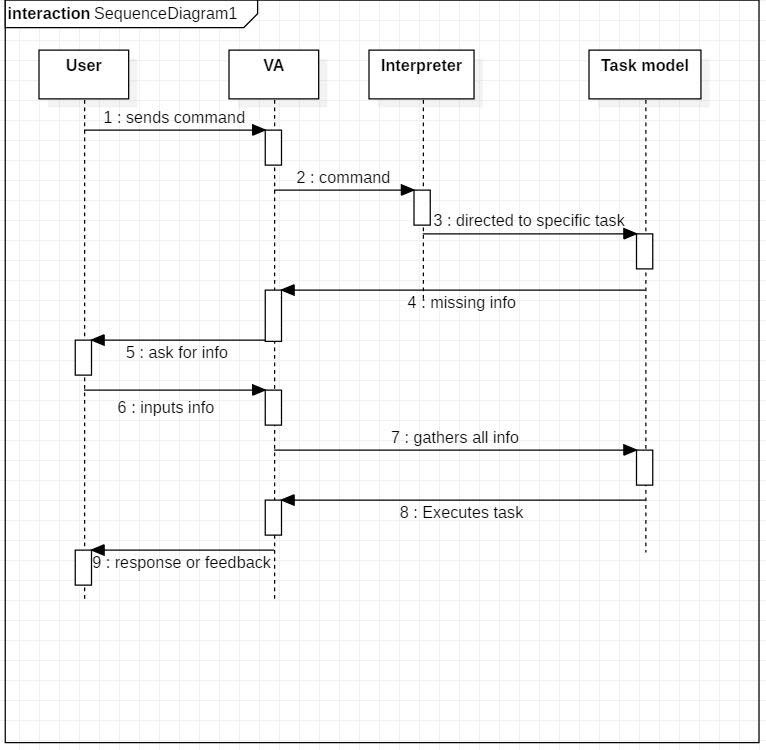
### SEQUENCE DIAGRAM

###### Sequence diagram for Query-Response



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

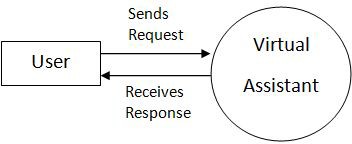
###### Sequence diagram for Task Execution



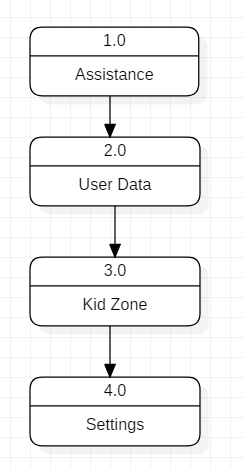
The user sends command to virtual assistant in audio form. The command is passed to the interpreter. It identifies what the user has asked and directs it to task executer. If the task is missing some info, the virtual assistant asks user back about it. The received information is sent back to task and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

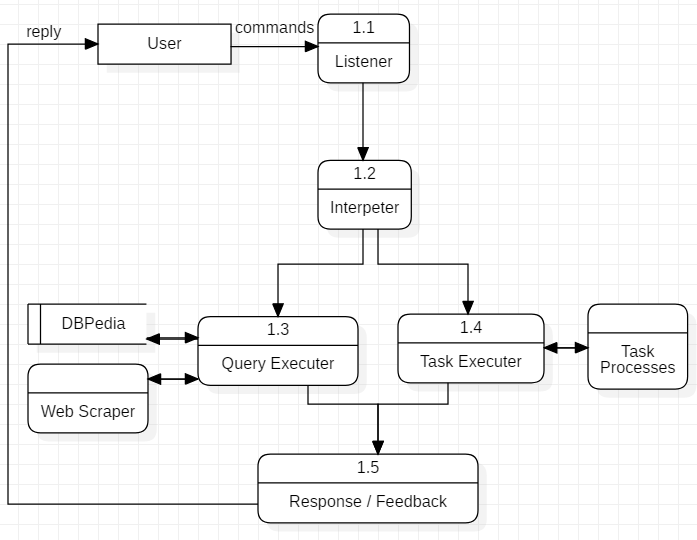
### DATA FLOW DIAGRAM

###### DFD Level 0 (Context Level Diagram)

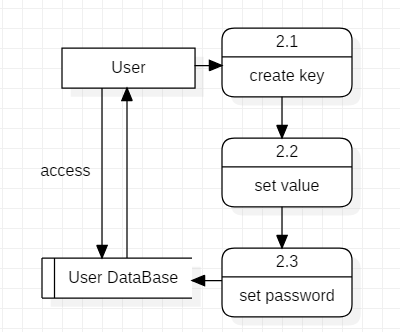


* + 1. DFD Level 1

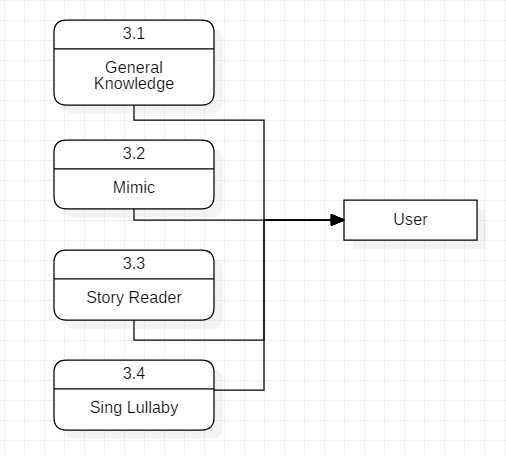


* + 1. DFD Level 2

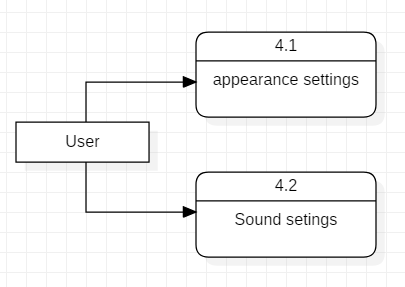
Data Flow in Assistance



Managing User Data

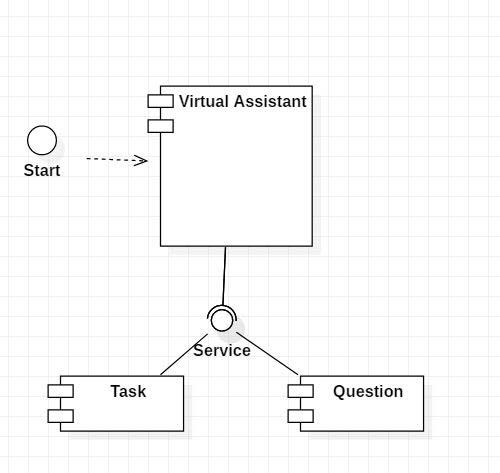


Data Flow in Kid Zone



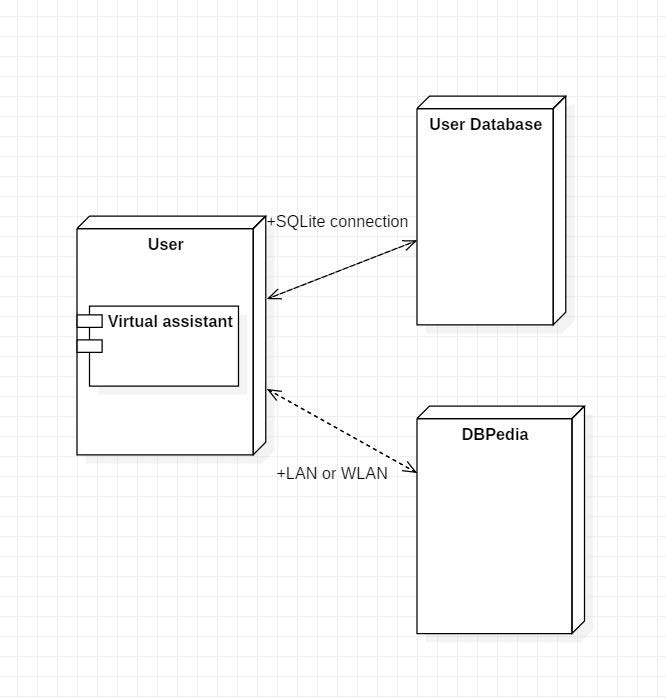
Settings of virtual Assistant

### COMPONENT DIAGRAM



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

### DEPLOYMENT DIAGRAM



The user interacts with SQLite database using SQLite connection in Python code. The knowledge database DBPedia must be accessed via internet connection. This requires LAN or WLAN / Ethernet network.

### DATA DICTIONARY

#### User

|  |  |
| --- | --- |
| Key | Text |
| Value | Text |
| Lock | Boolean |
| Password | Text |

Question

|  |  |
| --- | --- |
| Qid | Integer PRIMARY KEY |
| Query | Text |
| Answer | Text |

Task

|  |  |
| --- | --- |
| Tid | Integer PRIMARY KEY |
| Status | Text (Active/Waiting/Stopped) |
| Level | Text (Parent/Sub) |
| Priority | Integer |

Reminder

|  |  |
| --- | --- |
| Rid | Integer PRIMARY KEY |
| Tid | Integer FOREIGN KEY |
| What | Text |
| When | Time |
| On | Date |
| Notify before | Time |

Note

|  |  |
| --- | --- |
| Nid | Integer PRIMARY KEY |
| Tid | Integer FOREIGN KEY |
| Data | Text |
| Priority | Integer |

* + 1. **TEST CASE DESIGN**

##### Test Case 1

**Test Title:** Response Time

Test ID: T1

**Test Priority:** High

**Test Objective:** To make sure that the system respond back time is efficient.

Description:

Time is very critical in a voice based system. As we are not typing inputs, we are speaking them. The system must also reply in a moment. User must get instant response of the query made.

* **Test Case 2 Test Title:** Accuracy **Test ID:** T2

**Test Priority:** High

**Test Objective:** To assure that answers retrieved by system are accurate as per gathered data.

Description:

A virtual assistant system is mainly used to get precise answers to any question asked. Getting answer in a moment is of no use if the answer is not correct. Accuracy is of utmost importance in a virtual assistant system.

##### Test Case 3

**Test Title:** Approximation

Test ID: t3

**Test priority:** Moderate

**Test Objective:** To check approximate answers about calculations.

Description:

There are times when mathematical calculation requires approximate value. For example, if someone asks for value of PI the system must respond with approximate value and not the accurate value. Getting exact value in such cases is undesirable.

Note: There might include a few more test cases and these test cases are also subject to change with the final software development.

Code Explanation-

So let’s create our own virtual assistant.

***Notes-***

* All the codes is available on my GitHub.
* Demo YouTube video and code YouTube video is also available on my channel.
* Required links and packages are mentioned below.
* Sharing would be appreciated.

***Lets code-***

**2.1. Required packages and libraries-**

pip install JarvisAI

This is the latest virtual assistant module, created by me. It provides the basic functionality of any virtual assistant. The prerequisite is only Python (> 3.6).

Usage and Features-

After installing the library you can import the module-

import JarvisAI

obj = JarvisAI.JarvisAssistant()

response = obj.mic\_input()

print(response)

The functionality is cleared by the methods name. You can check the code for example.

1. mic\_input
2. text2speech
3. shutdown
4. website\_opener
5. send\_mail
6. tell\_me\_date
7. tell\_me\_time
8. launch\_any\_app
9. weather
10. news
11. tell\_me

***Read more about it***[***here***](https://pypi.org/project/JarvisAI/)***and you can also contribute to this repo***[***here***](https://github.com/Dipeshpal/Jarvis_AI)***.***

2.2. **Code-**

**Imports-**

*import* JarvisAI

*import* re

*import* pprint

*import* random

**Object creation** of JarvisAI as per as [documentation](https://pypi.org/project/JarvisAI/)–

obj = JarvisAI.JarvisAssistant()

We have created this ‘t2s(text)’ function. **This will convert any text to speech.** The entire program we will use (call) this function to produce speech from text.

*def* t2s(text):

obj.text2speech(text)

**We want to continuously listen to input from the user,** so this ‘mic\_input()’ will try to fetch audio from the computer’s microphone continuously. It will process the audio and return text in ‘res’ variable. We can use this ‘res’ variable to perform some action according to user input.

*while True*:

res = obj.mic\_input()

**Weather Forecast:** We are using a regular expression to match queries in user input. If ‘weather’ or ‘temperature’ is found in user input ‘res’ then we want to do weather forecasting. No need to write things from scratch, just call ‘obj.weather(city=city)’.

You just need to fetch the city from user input and pass it to the weather function. It will tell you the weather forecasting for your city.

We can pass this returned ‘weather\_res’ to ‘ t2s(weather\_res)’ to produce speech from ‘weather\_res’ string.

*while True*:

res = obj.mic\_input()

*if* re.search('weather|temperature', res):

city = res.split(' ')[-1]

weather\_res = obj.weather(city=city)

print(weather\_res)

t2s(weather\_res)

**News:**Similarly as above, match the ‘news’ word from user input ‘res’. If matched then call ‘obj.news’.

It will return 15 news as a list of strings. So, we can fetch news as ‘news\_res[0]’ and pass it to ‘t2s(news\_res[0])’.

*while True*:

res = obj.mic\_input()

*if* re.search('news', res):

news\_res = obj.news()

pprint.pprint(news\_res)

t2s(f"I have found {len(news\_res)} news. You can read it. Let me tell you first 2 of them")

t2s(news\_res[0])

t2s(news\_res[1])

**Tells about almost everything:**It will fetch the first 500 characters from Wikipedia and return them as a string. You can use ‘obj.tell\_me(topic)’.

You need to pass ‘topic’ to ‘tell\_me(topic=topic)’. The topic is the keyword you want to know about.

*while True*:

res = obj.mic\_input()

*if* re.search('tell me about', res):

topic = res.split(' ')[-1]

wiki\_res = obj.tell\_me(topic)

print(wiki\_res)

t2s(wiki\_res)

**Date and Time:** It will tell you the current date and time of your system.

*while True*:

res = obj.mic\_input()

*if* re.search('date', res):

date = obj.tell\_me\_date()

print(date)

print(t2s(date))

*if* re.search('time', res):

time = obj.tell\_me\_time()

print(time)

t2s(time)

**Open Any Website:**This ‘obj.website\_opener(domain)’ will open any website for you. You just need to fetch the domain from user input then pass to ‘obj.website\_opener(domain)’. It will open the website in your default browser.

*while True*:

res = obj.mic\_input()

*if* re.search('open', res):

domain = res.split(' ')[-1]

open\_result = obj.website\_opener(domain)

print(open\_result)

**Launch any Applications, Games, etc:**

This is little tricky, in ‘obj.launch\_any\_app(path\_of\_app=path)’ the function you need to pass the path of your ‘.exe’ file.

So we have created ‘dict\_app’ dictionary which is having an ‘app name’ as key and ‘path’ as value. We can use this ‘dict\_app’ for lookup. If the user input app exists in the dictionary then we will open it by fetching the path.

The below example is only for **Chrome and Epic Games.**

*while True*:

res = obj.mic\_input()

*if* re.search('launch', res):

dict\_app = {

'chrome': 'C:\Program Files (x86)\Google\Chrome\Application\chrome.exe',

'epic games': 'C:\Program Files (x86)\Epic Games\Launcher\Portal\Binaries\Win32\EpicGamesLauncher.exe'

}

app = res.split(' ', 1)[1]

path = dict\_app.get(app)

*if* path *is None*:

t2s('Application path not found')

print('Application path not found')

*else*:

t2s('Launching: ' + app)

obj.launch\_any\_app(path\_of\_app=path)

**Greetings and Chat,**you can create greetings and chat like this for now.

I am working on <https://pypi.org/project/JarvisAI/> to add chat kind of functionality using Tensorflow. You can [contribute](https://github.com/Dipeshpal/Jarvis_AI)to making it better.

*while True*:

res = obj.mic\_input()

*if* re.search('hello', res):

print('Hi')

t2s('Hi')

*if* re.search('how are you', res):

li = ['good', 'fine', 'great']

response = random.choice(li)

print(f"I am {response}")

t2s(f"I am {response}")

*if* re.search('your name|who are you', res):

print("My name is Jarvis, I am your personal assistant")

t2s("My name is Jarvis, I am your personal assistant")

**Ask- ‘What can you do?’:**Here simply we are using ‘obj.t2s()’to produce some speech. If you know python, you will easily understand the code below-

*while True*:

res = obj.mic\_input()

*if* re.search('what can you do', res):

li\_commands = {

"open websites": "Example: 'open youtube.com",

"time": "Example: 'what time it is?'",

"date": "Example: 'what date it is?'",

"launch applications": "Example: 'launch chrome'",

"tell me": "Example: 'tell me about India'",

"weather": "Example: 'what weather/temperature in Mumbai?'",

"news": "Example: 'news for today' ",

}

ans = """I can do lots of things, for example you can ask me time, date, weather in your city,

I can open websites for you, launch application and more. See the list of commands-"""

print(ans)

pprint.pprint(li\_commands)

t2s(ans)

3. Complete code-

import JarvisAI

import re

import pprint

import random

obj = JarvisAI.JarvisAssistant()

def t2s(text):

obj.text2speech(text)

while True:

res = obj.mic\_input()

if re.search('weather|temperature', res):

city = res.split(' ')[-1]

weather\_res = obj.weather(city=city)

print(weather\_res)

t2s(weather\_res)

if re.search('news', res):

news\_res = obj.news()

pprint.pprint(news\_res)

t2s(f"I have found {len(news\_res)} news. You can read it. Let me tell you first 2 of them")

t2s(news\_res[0])

t2s(news\_res[1])

if re.search('tell me about', res):

topic = res.split(' ')[-1]

wiki\_res = obj.tell\_me(topic)

print(wiki\_res)

t2s(wiki\_res)

if re.search('date', res):

date = obj.tell\_me\_date()

print(date)

print(t2s(date))

if re.search('time', res):

time = obj.tell\_me\_time()

print(time)

t2s(time)

if re.search('open', res):

domain = res.split(' ')[-1]

open\_result = obj.website\_opener(domain)

print(open\_result)

if re.search('launch', res):

dict\_app = {

'chrome': 'C:\Program Files (x86)\Google\Chrome\Application\chrome.exe',

'epic games': 'C:\Program Files (x86)\Epic Games\Launcher\Portal\Binaries\Win32\EpicGamesLauncher.exe'

}

app = res.split(' ', 1)[1]

path = dict\_app.get(app)

if path is None:

t2s('Application path not found')

print('Application path not found')

else:

t2s('Launching: ' + app)

obj.launch\_any\_app(path\_of\_app=path)

if re.search('hello', res):

print('Hi')

t2s('Hi')

if re.search('how are you', res):

li = ['good', 'fine', 'great']

response = random.choice(li)

print(f"I am {response}")

t2s(f"I am {response}")

if re.search('your name|who are you', res):

print("My name is Jarvis, I am your personal assistant")

t2s("My name is Jarvis, I am your personal assistant")

if re.search('what can you do', res):

li\_commands = {

"open websites": "Example: 'open youtube.com",

"time": "Example: 'what time it is?'",

"date": "Example: 'what date it is?'",

"launch applications": "Example: 'launch chrome'",

"tell me": "Example: 'tell me about India'",

"weather": "Example: 'what weather/temperature in Mumbai?'",

"news": "Example: 'news for today' ",

}

ans = """I can do lots of things, for example you can ask me time, date, weather in your city,

I can open websites for you, launch application and more. See the list of commands-"""

print(ans)

pprint.pprint(li\_commands)

t2s(ans)

4. Github Repository

Just clone the repository- <https://github.com/Dipeshpal/Jarvis-Assisant.git>

Then run pip install -r requirements.txt

It will automatically install everything.

What can this A.I. assistant do for you?

* It can send emails on your behalf.
* It can play music for you.
* It can do Wikipedia searches for you.
* It is capable of opening websites like Google, Youtube, etc., in a web browser.
* It is capable of opening your code editor or IDE with a single voice command.

Enough talks! Let's start building our own J.A.R.V.I.S.

Starting VS Code

I am going to use the VS Code IDE in this video. Feel free to use any other IDE you are comfortable d with. Start a new project and make a file called jarvis.py.

Defining Speak Function

The first and foremost thing for an A.I. assistant is that it should be able to speak. To make our J.A.R.V.I.S. talk, we will make a function called **speak().**This function will take audio as an argument, and then it will pronounce it.

def speak(audio):

pass #For now, we will write the conditions later.

Now, the next thing we need is audio. We must supply audio so that we can pronounce it using the speak() function we made. We are going to install a module called **pyttsx3.**

What is pyttsx3?

* A python library that will help us to convert text to speech. In short, it is a text-to-speech library.
* It works offline, and it is compatible with Python 2 as well as Python 3.

**Installation:**

pip install pyttsx3

In case you receive such errors:

* No module named win32com.client
* No module named win32
* No module named win32api

Then, install pypiwin32 by typing the below command in the terminal :

pip install pypiwin32.

After successfully installing pyttsx3, import this module into your program.

**Usage:**

import pyttsx3

engine = pyttsx3.init('sapi5')

voices= engine.getProperty('voices') #getting details of current voice

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

What is sapi5?

* Microsoft developed speech API.
* Helps in synthesis and recognition of voice.

What Is VoiceId?

* Voice id helps us to select different voices.
* voice[0].id = Male voice
* voice[1].id = Female voice

Writing Our speak() Function :

We made a function called speak() at the starting of this tutorial. Now, we will write our speak() function to convert our text to speech.

def speak(audio):

engine.say(audio)

engine.runAndWait() #Without this command, speech will not be audible to us.

Creating Our main() function:

We will create a main() function, and inside this main() Function, we will call our speak function.

**Code:**

if \_\_name\_\_=="\_\_main\_\_" :

speak("Code With Harry")

Whatever you will write inside this speak() function will be converted into speech. Congratulations! With this, our J.A.R.V.I.S. has its own voice, and it is ready to speak.

Defining Wish me Function :

Now, we will make a **wishme()**function that will make our J.A.R.V.I.S. wish or greet the user according to the time of computer or pc. To provide current or live time to A.I., we need to import a module called datetime. Import this module to your program by:

import datetime

Now, let's start defining the **wishme()** function:

def wishme():

hour = int(datetime.datetime.now().hour)

Here, we have stored the current hour or time integer value into a variable named hour. Now, we will use this hour value inside an if-else loop.

 – Defining Take command Function :

The next most important thing for our A.I. assistant is that it should take command with the help of the microphone of the user's system. So, now we will make a **takeCommand()**function.  With the help of the takeCommand() function, our A.I. assistant will return a string output by taking microphone input from the user.

 Before defining the takeCommand() function, we need to install a module called **speechRecognition.**Install this module by:

pip install speechRecognition

After successfully installing this module, import this module into the program by writing an import statement.

import speechRecognition as sr

Let's start coding the takeCommand() function :

def takeCommand():

#It takes microphone input from the user and returns string output

r = sr.Recognizer()

with sr.Microphone() as source:

print("Listening...")

r.pause\_threshold = 1

audio = r.listen(source)

We have successfully created our takeCommand() function. Now we are going to add a try and except block to our program to handle errors effectively.

try:

print("Recognizing...")

query = r.recognize\_google(audio, language='en-in') #Using google for voice recognition.

print(f"User said: {query}\n") #User query will be printed.

except Exception as e:

# print(e)

print("Say that again please...") #Say that again will be printed in case of improper voice

return "None" #None string will be returned

return query

Coding logic of Jarvis

 Now, we will develop logic for different commands such as Wikipedia searches, playing music, etc.

Defining Task 1: To search something on Wikipedia

 To do Wikipedia searches, we need to install and import the Wikipedia module into our program. Type the below command to install the Wikipedia module :

pip install wikipedia

 After successfully installing the Wikipedia module, import it into the program by writing an import statement.

if \_\_name\_\_ == "\_\_main\_\_":

wishMe()

while True:

# if 1:

query = takeCommand().lower() #Converting user query into lower case

# Logic for executing tasks based on query

if 'wikipedia' in query: #if wikipedia found in the query then this block will be executed

speak('Searching Wikipedia...')

query = query.replace("wikipedia", "")

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

speak("According to Wikipedia")

print(results)

speak(results)

In the above code, we have used an if statement to check whether Wikipedia is in the user's search query or not. If Wikipedia is found in the user's search query, then two sentences from the summary of the Wikipedia page will be converted to speech with the speak function's help.

– Defining Task 2: To open YouTube site in a web-browser

 To open any website, we need to import a module called **webbrowser**. It is an in-built module, and we do not need to install it with a pip statement; we can directly import it into our program by writing an import statement.

Code:

elif 'open youtube' in query:

webbrowser.open("youtube.com")

Here, we are using an elif loop to check whether Youtube is in the user's query. Let' suppose the user gives a command as "J.A.R.V.I.S., open youtube." So, open youtube will be in the user's query, and the elif condition will be true.

 – Defining Task 3: To open Google site in a web-browser

elif 'open google' in query:

webbrowser.open("google.com")

We are opening Google in a web-browser by applying the same logic that we used to open youtube.

– Defining Task 4: To play music

To play music, we need to import a module called os. Import this module directly with an import statement.

elif 'play music' in query:

music\_dir = 'D:\\Non Critical\\songs\\Favorite Songs2'

songs = os.listdir(music\_dir)

print(songs)

os.startfile(os.path.join(music\_dir, songs[0]))

In the above code, we first opened our music directory and then listed all the songs present in the directory with the os module's help. With the help of os.startfile, you can play any song of your choice. I am playing the first song in the directory. However, you can also play a random song with the help of a random module. Every time you command to play music, J.A.R.V.I.S. will play any random song from the song directory.

– Defining Task 5: To know the current time

elif 'the time' in query:

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

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

In the above, code we are using the datetime() function and storing the current or live system time into a variable called strTime. After storing the time in strTime, we are passing this variable as an argument in speak function. Now, the time string will be converted into speech.

Defining Task 6: To open the VS Code Program

elif 'open code' in query:

codePath = "C:\\Users\\Haris\\AppData\\Local\\Programs\\Microsoft VS Code\\Code.exe"

os.startfile(codePath)

To open the VS Code or any other application, we need the code path of the application.

**Steps to get the code path of the application:**

**Step 1:**Open the file location.

**Step 2:**Right-click on the application and click on properties.

**Step 3:** the target from the target section.

After ing the target of the application, save the target into a variable. Here, I am saving the target into a variable called codePath, and then we are using the os module to open the application.

Defining Task 7: To send Email

To send an email, we need to import a module called smtplib.

**What is smtplib?**

* Simple Mail Transfer Protocol (SMTP) is a protocol that allows us to send emails and route emails between mail servers. An instance method called **sendmail**is present in the SMTP module. This instance method allows us to send an email.  It takes 3 parameters:
* **The sender:** Email address of the sender.
* **The receiver:**T Email of the receiver.
* **The *message:***A string message which needs to be sent to one or more than one recipient.

Defining Send email function :

We will create a **sendEmail()**function, which will help us send emails to one or more than one recipient.

def sendEmail(to, content):

server = smtplib.SMTP('smtp.gmail.com', 587)

server.ehlo()

server.starttls()

server.login('youremail@gmail.com', 'your-password')

server.sendmail('youremail@gmail.com', to, content)

server.close()

In the above code, we are using the SMTP module, which we have already discussed above.

**Note:** Do not forget to *'enable the less secure apps'*feature in your Gmail account. Otherwise, the sendEmail function will not work properly.

Calling sendEmail() function inside the main() function:

elif 'email to harry' in query:

try:

speak("What should I say?")

content = takeCommand()

to = "harryyourEmail@gmail.com"

sendEmail(to, content)

speak("Email has been sent!")

except Exception as e:

print(e)

speak("Sorry my friend harry bhai. I am not able to send this email")

We are using the try and except block to handle any possible error while sending emails.

Recapitulate

1. First of all, we have created a **wishme()**function that gives the greeting functionality according to our A.I system time.
2. After wishme() function, we have created a **takeCommand()**function, which helps our A.I to take command from the user. This function is also responsible for returning the user's query in a string format.
3. We developed the code logic for opening different websites like google, youtube, and stack overflow.
4. Developed code logic for opening VS Code or any other application.
5. At last, we added functionality to send emails.

Is it an A.I.?

Many people will argue that the virtual assistant that we have created is not an A.I, but it is the output of a bunch of the statement. But, if we look at the fundamental level, the sole purpose of A.I develop machines that can perform human tasks with the same effectiveness or even more effectively than humans.

It is a fact that our virtual assistant is not a very good example of A.I., but it is an A.I.!

The E.N.D.

With this, you have successfully made your very first virtual assistant. Explore and try to add other functionalities to J.A.R.V.I.S. I hope you all have liked this tutorial. Feel free to ask your queries in the QnA section.

Code as described

import pyttsx3 #pip install pyttsx3

import speech\_recognition as sr #pip install speechRecognition

import datetime

import wikipedia #pip install wikipedia

import webbrowser

import os

import smtplib

engine = pyttsx3.init('sapi5')

voices = engine.getProperty('voices')

# print(voices[1].id)

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

def speak(audio):

engine.say(audio)

engine.runAndWait()

def wishMe():

hour = int(datetime.datetime.now().hour)

if hour>=0 and hour<12:

speak("Good Morning!")

elif hour>=12 and hour<18:

speak("Good Afternoon!")

else:

speak("Good Evening!")

speak("I am Jarvis Sir. Please tell me how may I help you")

def takeCommand():

#It takes microphone input from the user and returns string output

r = sr.Recognizer()

with sr.Microphone() as source:

print("Listening...")

r.pause\_threshold = 1

audio = r.listen(source)

try:

print("Recognizing...")

query = r.recognize\_google(audio, language='en-in')

print(f"User said: {query}\n")

except Exception as e:

# print(e)

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

return "None"

return query

def sendEmail(to, content):

server = smtplib.SMTP('smtp.gmail.com', 587)

server.ehlo()

server.starttls()

server.login('youremail@gmail.com', 'your-password')

server.sendmail('youremail@gmail.com', to, content)

server.close()

if \_\_name\_\_ == "\_\_main\_\_":

wishMe()

while True:

# if 1:

query = takeCommand().lower()

# Logic for executing tasks based on query

if 'wikipedia' in query:

speak('Searching Wikipedia...')

query = query.replace("wikipedia", "")

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

speak("According to Wikipedia")

print(results)

speak(results)

elif 'open youtube' in query:

webbrowser.open("youtube.com")

elif 'open google' in query:

webbrowser.open("google.com")

elif 'open stackoverflow' in query:

webbrowser.open("stackoverflow.com")

elif 'play music' in query:

music\_dir = 'D:\\Non Critical\\songs\\Favorite Songs2'

songs = os.listdir(music\_dir)

print(songs)

os.startfile(os.path.join(music\_dir, songs[0]))

elif 'the time' in query:

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

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

elif 'open code' in query:

codePath = "C:\\Users\\Haris\\AppData\\Local\\Programs\\Microsoft VS Code\\Code.exe"

os.startfile(codePath)

elif 'email to harry' in query:

try:

speak("What should I say?")

content = takeCommand()

to = "Email@gmail.com"

sendEmail(to, content)

speak("Email has been sent!")

except Exception as e:

print(e)

speak("Sorry my friend Ankit. I am not able to send this email")

import pyttsx3

import speech\_recognition as sr

import datetime

import wikipedia

import webbrowser

import os

import winshell

import subprocess

import smtplib

import pyjokes

import ctypes

import time

engine = pyttsx3.init('sapi5')

voices = engine.getProperty('voices')

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

def speak(audio):

    engine.say(audio)

    engine.runAndWait()

def wishMe():

    hour = int(datetime.datetime.now().hour)

    if hour >= 0 and hour < 12:

        speak("good Morning! ")

    elif hour >= 12 and hour < 18:

        speak("good AfterNoon! ")

    else:

        speak("good Evening! ")

    speak("I am Anny Your Virtual Assittant Mr Anon.  please Tell me How may I help You")

def takecommand():

    r=sr.Recognizer()

    with sr.Microphone() as source:

        print("Listening.....")

        r.pause\_threshold = 1

        audio=r.listen(source)

    try:

        print("Recognizing......")

        query=r.recognize\_google(audio, language='en-in')

        print(f"user said :{query}\n")

    except Exception as e:

        # print(e)

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

        return "None"

    return query

def sendEmail(to, content):

    server = smtplib.SMTP('smtp.gmail.com', 587)

    server.ehlo()

    server.starttls()

    server.login('youremail@gmail.com', 'your-password')

    server.sendmail('youremail@gmail.com', to, content)

    server.close()

if \_\_name\_\_ == "\_\_main\_\_":

    wishMe()

    while True:

       query = takecommand().lower()

       if 'wikipedia' in query:

            speak('Searching Wikipedia...')

            query = query.replace("wikipedia", "")

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

            speak("According to Wikipedia")

            print(results)

            speak(results)

       elif 'open youtube' in query:

            webbrowser.open("youtube.com")

       elif 'open google' in query:

            webbrowser.open("google.com")

       elif 'open stackoverflow' in query:

            webbrowser.open("stackoverflow.com")

       elif 'how are you' in query:

            speak("I am fine, Thank you")

            speak("How are you, Sir")

       elif "change my name to" in query:

            query = query.replace("change my name to", "")

            assname = query

       elif "change name" in query:

            speak("What would you like to call me, Sir ")

            assname = takecommand()

            speak("Thanks for naming me")

       elif "what's your name" in query or "What is your name" in query:

            speak("My friends call me")

            speak(assname)

            print("My friends call me", assname)

       elif 'lock window' in query:

                speak("locking the device")

                ctypes.windll.user32.LockWorkStation()

       elif 'shutdown system' in query:

                speak("Hold On a Sec ! Your system is on its way to shut down")

                subprocess.call('shutdown / p /f')

       elif 'empty recycle bin' in query:

            winshell.recycle\_bin().empty(confirm = False, show\_progress = False, sound = True)

            speak("Recycle Bin Recycled")

       elif "don't listen" in query or "stop listening" in query:

            speak("for how much time you want to stop jarvis from listening commands")

            a = int(takecommand())

            time.sleep(a)

            print(a)

       elif "where is" in query:

            query = query.replace("where is", "")

            location = query

            speak("User asked to Locate")

            speak(location)

            webbrowser.open("https://www.google.nl / maps / place/" + location + "")

     #   elif "camera" in query or "take a photo" in query:

     #        ec.capture(0, "Jarvis Camera ", "img.jpg")

       elif "restart" in query:

            subprocess.call(["shutdown", "/r"])

       elif "hibernate" in query or "sleep" in query:

            speak("Hibernating")

            subprocess.call("shutdown / h")

       elif "log off" in query or "sign out" in query:

            speak("Make sure all the application are closed before sign-out")

            time.sleep(5)

            subprocess.call(["shutdown", "/l"])

       elif "write a note" in query:

            speak("What should i write, sir")

            note = takecommand()

            file = open('jarvis.txt', 'w')

            speak("Sir, Should i include date and time")

            snfm = takecommand()

            if 'yes' in snfm or 'sure' in snfm:

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

                file.write(strTime)

                file.write(" :- ")

                file.write(note)

            else:

                file.write(note)

       elif "show note" in query:

            speak("Showing Notes")

            file = open("jarvis.txt", "r")

            print(file.read())

            speak(file.read(6))

       elif 'play music' in query:

            music\_dir = 'E:\\AUDIO\\MY FAV\\PUNJABI\\ROCKING'

            songs = os.listdir(music\_dir)

            print(songs)

            os.startfile(os.path.join(music\_dir, songs[0]))

       elif 'the time' in query:

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

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

       elif 'open code' in query:

            codePath = "C:\\Users\\Anon\\AppData\\Local\\Programs\\Microsoft VS Code\\Code.exe"

            os.startfile(codePath)

       elif "what's your name" in query or "What is your name" in query:

            speak("My friends call me Anny")

       elif 'is love' in query:

            speak("It is 7th sense that destroy all other senses")

       elif "who are you" in query:

            speak("I am your virtual assistant Anny created by Anon")

       elif 'reason for you' in query:

            speak("I was created as a Minor project by Mister Anon ")

       elif "who made you" in query or "who created you" in query:

            speak("I have been created by Anon.")

       elif 'fine' in query or "am good" in query:

            speak("It's good to know that your fine")

       elif "change my name to" in query:

            query = query.replace("change my name to", "")

            assname = query

       elif "change name" in query:

            speak("What would you like to call me, Sir ")

            assname = takecommand()

            speak("Thanks for naming me")

       elif "how are you" in query:

            speak("I'm fine, glad you me that")

       elif "who i am" in query:

            speak("If you talk then definitely your human.")

       elif "why you came to world" in query:

            speak("Thanks to ANoN . further It's a secret")

       elif "my gf" in query or " my bf" in query:

            speak("I'm not sure about, may be you should give me some time")

       elif "i love you" in query:

            speak("It's hard to understand")

       elif "do you love me" in query:

            speak("I am always available for you My Dear")

       elif "funny" in query:

            speak("Thank you My Dear")

       elif "tired" in query:

            speak("you need some rest. go get some sleep ")

       elif "bored" in query:

            speak("do you want listen music or want to listen a joke ")

       elif " good person" in query or "gentle" in query:

            speak("You are the sweetest person in this World  ")

       elif 'Who is Anon' in query:

            speak("My creator  My owner ")

       elif 'joke' in query:

            speak(pyjokes.get\_joke())

       elif 'exit' in query:

            speak("Thanks for giving me your time")

            exit()

       elif 'email to Anon' in query:

            try:

                speak("What should I say?")

                content = takecommand()

                to = "ThedashingAnon@gmail.com"

                sendEmail(to, content)

                speak("Email has been sent!")

            except Exception as e:

                print(e)

                speak("Sorry Mr  Anon. I am not able to send this email")

**REFERENCE AND BIBLIOGRAPHY**

#### 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.co.in](http://www.google.co.in/)
    - www.Github.com

#### Books referred

* + - Python Programming -
    - Learning Python -

#### YouTube Channels referred

* + - Code With Harry
    - edureka!
    - Codeshubham
    - Anontuts