VIRTUAL DESKTOP AI

ABSTRACT

The project aims to develop a personal-assistant for Linux-based systems. Jarvis draws its inspiration from virtual assistants like Cortana for Windows, and Siri for iOS. It has been designed to provide a user-friendly interface for carrying out a variety of tasks by employing certain well-defined commands. Users can interact with the assistant either through voice commands or using keyboard input.

As a personal assistant, Jarvis assists the end-user with day-to-day activities like general human conversation, searching queries in google, bing or yahoo, searching for videos, retrieving images, live weather conditions, word meanings, searching for medicine details, health recommendations based on symptoms and reminding the user about the scheduled events and tasks. The user statements/commands are analysed with the help of Arificial Intelligence to give an optimal solution.

CONTENTS

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
IOT	Internet of Things
GUI	Graphical User Interface
AI	Artificial Intelligence
IDE	Integrated Development Environment

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 General Theory

In today's era almost all tasks are digitalized. We have Smartphone in hands and it is nothing less than having world at Wer 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 ecommerce 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 We ease Wer 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 JIA. 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 JIA.

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. JIA is effortless to use. Call the wake word 'JIA' 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 2020.Virtual assistants are turning out to be smarter than ever. Allow We are intelligent assistant to make email work for We. 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.

1.2 Implementation Overview

We can use any IDE like Anaconda, Visual Studio etc. So the first step is to create a function for a voice which has arguments. Also, we are using Speech API to take voice as input. There are two default voices in our computer I.e. Male and Female. So We can use either any from these. Also, check the voice function by giving some text input and it will be converted into voice.

We can create a new function for wishing the user. Use if_else condition statement for allocating the wished output. E.g. If its time between 12 to 18 then the system will say "Good Evening". Along with this, We can get a welcome voice also. E.g. "Welcome, What can I do for us".After that, we have to install the speech recognition model and then import it

Define a new function for taking command from a user. Also mention class for speech recognition and input type like microphone etc. Also mention pause threshold. Set a query for voice recognition language. We can use Google engine to convert War voice input to the text.

We have to install and import some other packages like pyttsx3, Wikipedia etc. Pyttsx3 helps We to convert text input to speech conversion. If We ask for any information then it will display the result in textual format. We can convert it very easily in the voice format as we have already defined a function in our code previously.

Else We ask to open We Tube in the query then it will go to the We Tube address automatically. For that, We have to install a web browser package and import it. In the same way, We can add queries for many websites like Google, Instagram, Facebook etc. The next task is to play the songs. It is same as We have done before. Add a query for "play songs". Also, add the location of songs folder so that assistant will play the song from that only.

So the main question is that how the queries work? So here we are using conditional statements. If the computer hears voice command which contains word We tube then it will go to the We Tube page address, if the voice contains google command then it will go to the Google search page and so many accordingly.

We can add so many pages and commands for Web desktop assistant.

Have We ever wondered how cool it would be to have Wer own A.I. assistant? Imagine how easier it would be to send emails without typing a single word, doing Wikipedia searches without opening web browsers, and performing many other daily tasks like playing music with the help of a single voice command.

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

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

1.3 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 us?" and then responds to verbal input.Virtual assistants can tremendously save We time. We spend hours in online research and then making the report in our terms of understanding. JIA can do

that for We. Provide a topic for research and continue with Wer tasks while JIA does the research. Another difficult task is to remember test dates, birthdates or anniversaries.

It comes with a surprise when We enter the class and realize it is class test today. Just tell JIA in advance about Wer tests and she reminds We well in advance so We 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.

1.4 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.

CHAPTER 2

LITERATURE SURVEY

2.1 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

2.2 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

2.3.1 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 $1/5^{th}$ 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 JIA, libraries used are speechrecognition to recognize voice, Pyttsx for text to speech, selenium for web automation etc.

Python is reasonably efficient. Efficiency is usually not a problem for small examples. If Wer 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 lowerlevel language. This will result in much less programming and more efficient code (because We will have more time to optimize) than writing everything in a low-level language

2.3.4 Pyttsx

Pyttsx stands for Python Text to Speech. It is a cross-platform Python wrapper for textto-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.

2.3.5 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.

CHAPTER 3

MATERIALS AND METHODS

3.1 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.

3.2 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.

Jarvis 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

CHAPTER 4
SYSTEM REQUIREMENT AND DESIGN

4.1 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

4.2 System Design & Block Diagrams

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

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.

4.2.5 SEQUENCE DIAGRAM

Sequence diagram for Query-Response

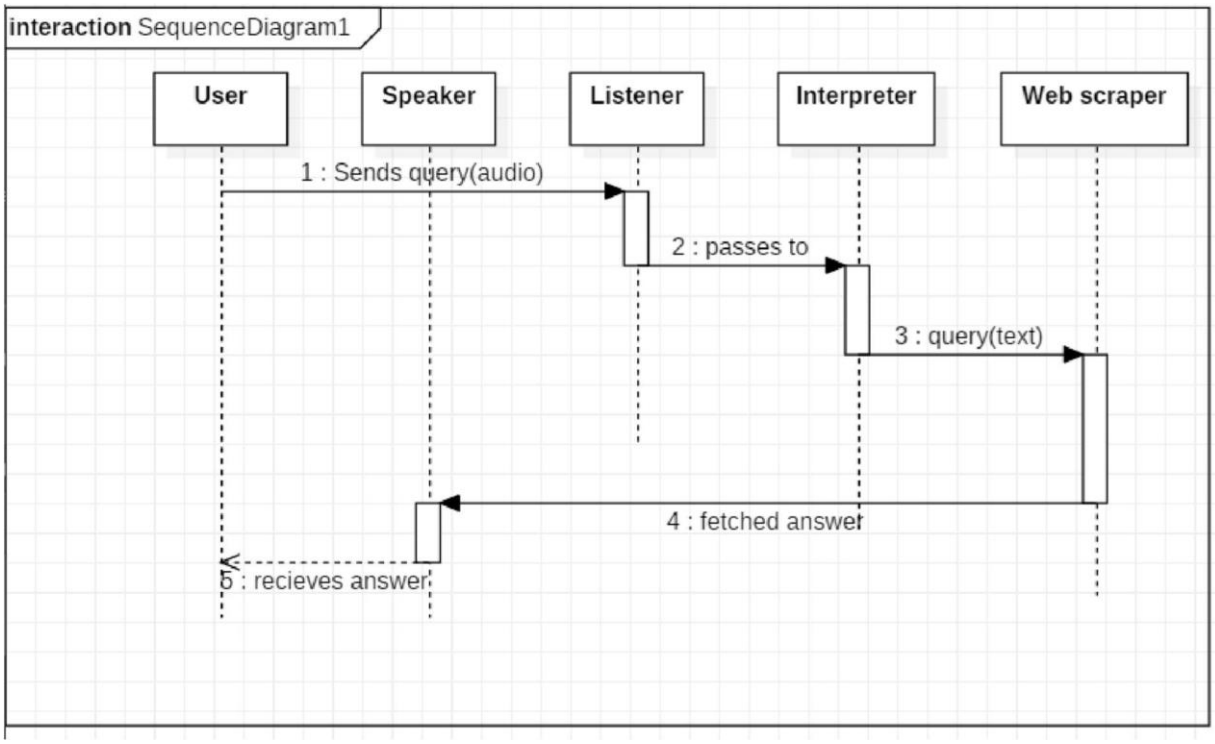

Fig 4.2.5.1 Sequence diagram for Query-Response

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.

4.2.6 DATA FLOW DIAGRAM

DFD Level 0 (Context Level Diagram)

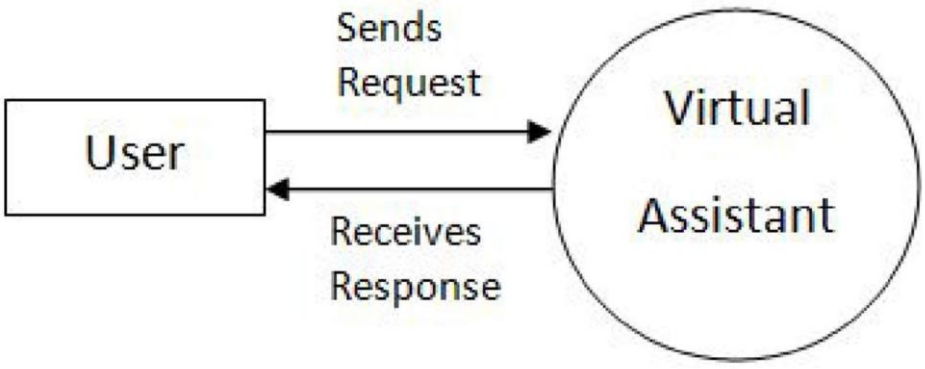

Fig 4.2.6.1 DFD Level 0

DFD Level 1

CHAPTER 5

RESULTS AND DISCUSSION

5.1 TEST CASE DESIGN

5.1.1 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.

5.1.2 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.

5.1.3 Test Case 3

Test Title: Approximation

Test ID: t3

Test priority: Moderate

Test Objective: To check approximate answers about calculations.

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-tospeech library.
- It works offline, and it is compatible with Python 2 as well as Python 3.

In case We 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.
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 Naman & Imteyaz")
```

Whatever We 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 Wer 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
```

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 search query of the user 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 WeTube 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 Wetube' in query:
        webbrowser.open("Wetube.com")
```

Here, we are using the elif loop to check whether Wetube is in the user's query. Let' suppose the user gives a command as "J.A.R.V.I.S., open Wetube." So, open Wetube 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 Wetube.

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)
```

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, We can play any song of Wer choice. I am playing the first song in the directory. However, We can also play a random song with the help of a random module. Every time We 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 with using datetime() function and storing the current or live system 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)
```

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: Copy the target from the target section.
After copying 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 to 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 :

```
def sendEmail(to, content):
    server = smtplib.SMTP('smtp.gmail.com', 587)    server.ehlo()    server.starttls()
server.login('Weremail@gmail.com', 'Wer-password')
server.sendmail('Weremail@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 Wer 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 = "harryWerEmail@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

5.3 Recapitulate

First of all, we have created a wishme() function that gives the greeting functionality according to our A.I system time.

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.

We developed the code logic for opening different websites like google, Wetube, and stack overflow.

Developed code logic for opening VS Code or any other application.

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. !

5.4 Code as inscribed

```
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[1].id)


def speak(audio):

   engine.say(audio) #engine will speak the given audio string     engine.runAndWait()


def wishMe():

   hour = int(datetime.datetime.now().hour)     if hour>=0 and hour<12:

       speak("Good Morning Mam, Welcome to Our Final Year Project!")

    elif hour>=12 and hour<18:

       speak("Good Afternoon Mam, Welcome to Our Final Year Project!")

    else:

       speak("Good Evening Mam, Welcome to Our Final Year Project!")


   speak("I am Jarvis. This project is been made my developers Naman Mittal and

Imteyaz Mallick. Please tell me how may I help We")
```

```
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)


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[1].id)


def speak(audio):
    engine.say(audio) #engine will speak the given audio string     engine.runAndWait()


def wishMe():
    hour = int(datetime.datetime.now().hour)     if hour>=0 and hour<12:
        speak("Good Morning Mam, Welcome to Our Final Year Project!")
     elif hour>=12 and hour<18:
        speak("Good Afternoon Mam, Welcome to Our Final Year Project!")
     else:
        speak("Good Evening Mam, Welcome to Our Final Year Project!")


    speak("I am Jarvis. This project is been made my developers Naman Mittal and
Imteyaz Mallick. Please tell me how may I help We")
 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)  #if i wnt to see error in my console, print it or else comm ent it  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('mittalnaman659@gmail.com', 'naman06051999')
server.sendmail('mittalnaman659@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=1)  speak("According to Wikipedia")  print(results)
speak(results)

elif 'open Wetube' in query:

webbrowser.open("Wetube.com")

elif 'open google' in query:

webbrowser.open("google.com")

elif 'open stackoverflow' in query:

webbrowser.open("stackoverflow.com")



elif 'open sathyabama website' in query:  webbrowser.open("sathyabama.ac.in")


elif 'play music' in query:

music_dir = 'D:\\All MP3 songs\\Selected mp3'  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 'thank We' in query:

print("please sir, do not say thank We. In friendship, no thank We an d no sorry")
speak("please sir, do not say thank We. In friendship, no thank We an d no sorry")

elif 'open best restaurants' in query:

webbrowser.open("https://www.topranker4u.com/city-wise-bestrestaurants-india/")

elif 'open list of colleges' in query:

webbrowser.open("http://www.studyguideindia.com/Colleges/Citywisecolleges-india.asp")

elif 'open result of last semester' in query:

webbrowser.open("http://cloudportal.sathyabama.ac.in/sist_semester_ja n_2021/login.php")

elif 'open amazon' in query:

webbrowser.open("http://amazon.com")

elif 'open flipkart' in query:

webbrowser.open("http://flipkart.com")

elif 'open book my show' in query:

webbrowser.open("https://in.bookmyshow.com/")

elif 'send mail' in query:  try:

speak("What should I say?")  content = takeCommand()  to =
"mittalnaman6590@gmail.com"  sendEmail(to, content)  speak("Email has been
sent!")  except Exception as e:

print(e)  speak("Sorry my friend Naman & Imteyaz Sir. I am not able to send  this
email")
```

CHAPTER 6

CONCLUSION

Through this voice assistant, we have automated various services using a single line command. It eases most of the tasks of the user like searching the web, retrieving weather forecast details, vocabulary help and medical related queries. We aim to make this project a complete server assistant and make it smart enough to act as a replacement for a general server administration. The future plans include integrating Jarvis with mobile using React Native to provide a synchronised experience between the two connected devices. Further, in the long run, Jarvis is planned to feature auto deployment supporting elastic

beanstalk, backup files, and all operations which a general Server Administrator does. The functionality would be seamless enough to replace the Server Administrator with Jarvis.

REFERENCES

1. Rabiner Lawrence, Juang Bing-Hwang. Fundamentals of Speech Recognition Prentice Hall , New Jersey, 1993, ISBN 0-13-015157-2

2. Deller John R., Jr., Hansen John J.L., Proakis John G. ,Discrete-Time Processing of Speech Signals, IEEE Press, ISBN 0-7803-5386-2

3. Hayes H. Monson,Statistical Digital Signal Processing and Modeling, John Wiley & Sons Inc. , Toronto, 1996, ISBN 0-471-59431-8

4. Proakis John G., Manolakis Dimitris G.,Digital Signal Processing, principles, algorithms, and applications, Third Edition, Prentice Hall , New Jersey, 1996, ISBN 0-13- 394338-9

5. Ashish Jain,Hohn Harris,Speaker identification using MFCC and HMM based techniques,university Of Florida,April 25,2004.

6. http://www.cse.unsw.edu.au/~waleed/phd/html/node38.html , downloaded on 2 Oct 2012