CHAPTER 1: INTRODUCTION

Artificial Intelligence when used with machines, it shows us the capability of thinking like humans. In this, a computer system is designed in such a way that typically requires interaction from human. As we know Python is an emerging language so it becomes easy to write a script for Voice Assistant in Python. The instructions for the assistant can be handled as per the requirement of user. Speech recognition is the Alexa, Siri, etc. In Python there is an API called Speech Recognition which allows us to convert speech into text. It was an interesting task to make my own assistant. It became easier to send emails without typing any word, searching on Google without opening the browser, and performing many other daily tasks like playing music, opening your favourite IDE with the help of a single voice command. In the current scenario, advancement in technologies is such that they can perform any task with same effectiveness or can say more effectively than us. By making this project, we both realized that the concept of AI in every field is decreasing human effort and saving time.

As the voice assistant is using Artificial Intelligence hence the result that it is providing are highly accurate and efficient. The assistant can help to reduce human effort and consumes time while performing any task, they removed the concept of typing completely and behave as another individual to whom we are talking and asking to perform task. The assistant is no less than a human assistant but we can say that this is more effective and efficient to perform any task. The libraries and packages used to make this assistant focuses on the time complexities and reduces time.

The functionalities include, it can send emails, it can read PDF, it can send text on WhatsApp, it can open command prompt, your favourite IDE, notepad etc., It can play music,

it can do Wikipedia searches for you, it can open websites like Google, YouTube, etc., in a web browser, it can give weather forecast, it can give desktop reminders of your choice. It can have some basic conversation.

Tools and technologies used are PyCharm IDE for making this project, and we created all .py files in PyCharm. Along with this we used following modules and libraries in my project. pyttsx3, SpeechRecognition, Datetime, Wikipedia, Smtplib, pywhatkit, pyjokes, pyPDF2, pyautogui, pyQt etc. we have created a live GUI for interacting with the JARVIS as it gives a design and interesting look while having the conversation.

1.1 PRESENT SYSTEM

We are familiar with many existing voice assistants like Alexa, Siri, Google Assistant, Cortana which uses concept of language processing, and voice recognition. They listen the command given by the user as per their requirements and performs that specific function in a very efficient and effective manner.

As these voice assistants are using Artificial Intelligence hence the result that they are providing are highly accurate and efficient. These assistants can help to reduce human effort and consumes time while performing any task, they removed the concept of typing completely and behave as another individual to whom we are talking and asking to perform task. These assistants are no less than a human assistant but we can say that they are more effective and efficient to perform any task. The algorithm used to make these assistant focuses on the time complexities and reduces time.

But for using these assistants one should have an account (like Google account for Google assistant, Microsoft account for Cortana) and can use it with internet connection only

because these assistants are going to work with internet connectivity. They are integrated with many devices like, phones, laptops, and speakers etc.

1.2 PROPOSED SYSTEM

It was an interesting task to make my own assistant. It became easier to send emails without typing any word, searching on Google without opening the browser, and performing many other daily tasks like playing music, opening your favourite IDE with the help of a single voice command. Jarvis is different from other traditional voice assistants in terms that it is specific to desktop and user does not need to make account to use this, it does not require any internet connection while getting the instructions to perform any specific task.

The IDE used in this project is PyCharm. All the python files were created in PyCharm and all the necessary packages were easily installable in this IDE. For this project following modules and libraries were used i.e., pyttsx3, Speech Recognition, Datetime, Wikipedia, Smtplib, pywhatkit, pyjokes, pyPDF2, pyautogui, pyQt etc. I have created a live GUI for interacting with the JARVIS as it gives a design and interesting look while having the conversation.

With the advancement JARVIS can perform any task with same effectiveness or can say more effectively than us. By making this project, we realized that the concept of AI in every field is decreasing human effort and saving time. Functionalities of this project include, it can send emails, it can read PDF, it can send text on WhatsApp, it can open command prompt, your favourite IDE, notepad etc., It can play music, it can do Wikipedia searches for you, it can open websites like Google, YouTube, etc., in a web browser, it can give weather forecast, it can give desktop reminders of your choice. It can have some basic conversation.

CHAPTER 2: LITRATURE SURVEY

2.1. SPEECH RECOGNITION

Speech recognition, frequently known as speech-to-textbook, is the capacity of a machine or program to fete and transfigure spoken words into comprehendible textbook. The vocabulary of rudimentary voice recognition software is confined, and it can only fete words and rulings when pronounced easily. More advanced software can deal with natural speech, multitudinous accentuations, and several languages Computer wisdom, linguistics, and computer engineering exploration are all used in speech recognition. Speech recognition functions are included into numerous current widgets and textbook-concentrated program to make using them easier or hands-free. Speech and voice recognition are two distinct technologies that must not be confused

- Speech recognition is a technology that recognizes words in spoken language.
- Voice recognition is a biometric technology for relating an existent’s voice.

2.2. ARTIFICIAL INTELLIGENCE

Artificial intelligence (AI) refers to a computers or a computer-controlled robot's ability to negotiate tasks normally performed by intelligent beings. The expression is considerably used to relate to a design aimed at creating systems with mortal- suchlike cognitive capacities, analogous as the capability to reason, discern meaning, generalize, and learn from formerly exploits. Since the invention of the digital computer in the 1940s, it has been proved that computers can be programmed to perform extremely complicated jobs with ease, analogous as chancing evidences for fine theorems or playing chess. Despite ongoing increases in computer processing speed and memory capacity, no program can yet match mortal severity across

broader fields or in conditioning taking a great deal of common knowledge. still, certain programmes have surpassed the performance situations of mortal specialists and professionals in executing specific tasks, and artificial intelligence in this limited sense can be set up in operations as different as medical opinion, computer quest machines, and voice or handwriting recognition.

2.3. VIRTUAL ASSISTANT

A virtual adjunct is an independent contractor that works for a customer and provides executive support while working from a position other than the client's Store. A virtual adjunct generally works from home, but may pierce important planning accoutrements similar as participated timetables from anywhere. Virtual sidekicks constantly have times of experience working as an executive adjunct or Store director. Virtual sidekicks with chops in social media, content operation, blog post jotting, graphic design, and online marketing are chancing new jobs. The demand for educated virtual sidekicks is projected to rise as working from home becomes further accepted by both workers and businesses.

- A virtual assistant is a self-employed professional who provides administrative help to clients from a remote location, usually a home Store.
- Scheduling appointments, making phone calls, planning vacations, and managing email accounts are all common responsibilities of a virtual assistant.
- Graphic design, blog authoring, bookkeeping, social media management, and marketing are some of the specialties of virtual assistants.

2.4. TEXT-TO-SPEECH

The capacity of computers to read text aloud is referred to as text-to-speech (TTS). Written text is converted to a phonemic representation, which is subsequently converted to waveforms that can be generated as sound by a TTS Engine. Third-party publishers offer TTS engines in a variety of languages, dialects, and specialist vocabularies.

2.5. CONTEXT EXTRACTION

Context Extraction (CE) is the process of obtaining structured data from machine-readable materials that are unstructured or semi-structured. Most of the time, this activity entails using natural language processing to process human language texts (NLP). TEST RESULTS for context extraction can be seen in recent activities in multimedia document processing, such as automatic annotation and content extraction from images/audio/video.

2.6. SYSTEM CALLS

The mechanism through which a computer software requests a service from the kernel of the operating system on which it is running is known as a system call. Hardware-related services (for example, accessing a hard disc drive), the creation and execution of new processes, and communication with core kernel services such as process scheduling are all examples of this. A process's interface with the operating system is provided by system calls.

CHAPTER 3: SYSTEM DESIGN

3.1. DATA FLOW

The data flow for JARVIS is as follow:

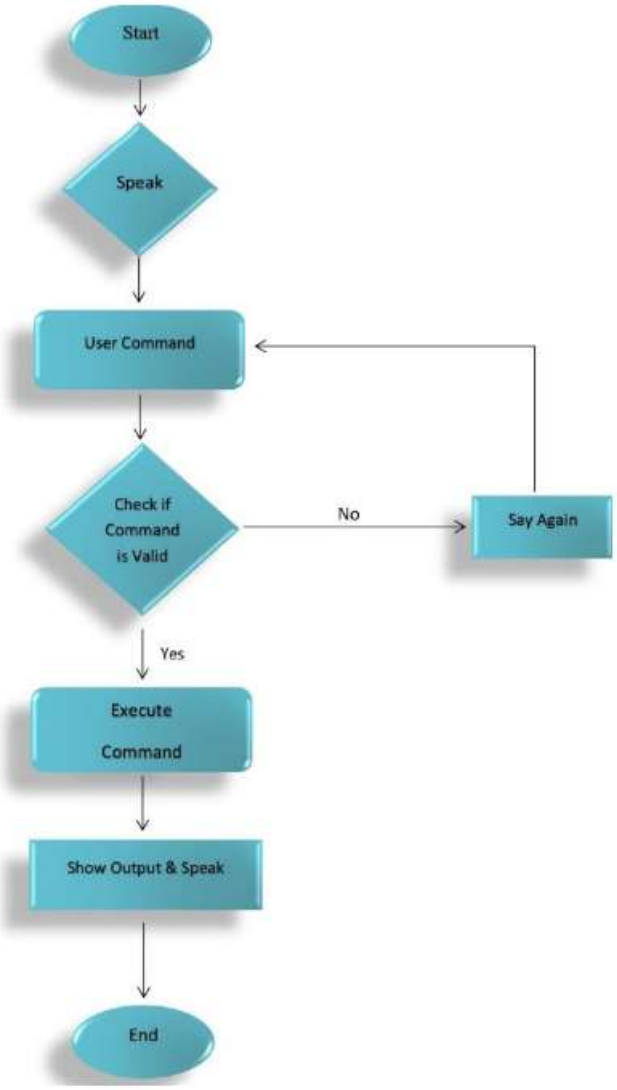

Figure 3.1 Data flow for JARVIS

The system is designed using the concept of Artificial Intelligence and with the help of necessary packages of Python. Python provides many libraries and packages to perform the tasks, for example pyPDF2 can be used to read PDF. The details of these packages are mentioned in CHAPTER 4 of this report.

The data in this project is nothing but user input, whatever the user says, the assistant performs the task accordingly. The user input is nothing specific but the list of tasks which a user wants to get performed in human language i.e., English.

3.2. ER DIAGRAM

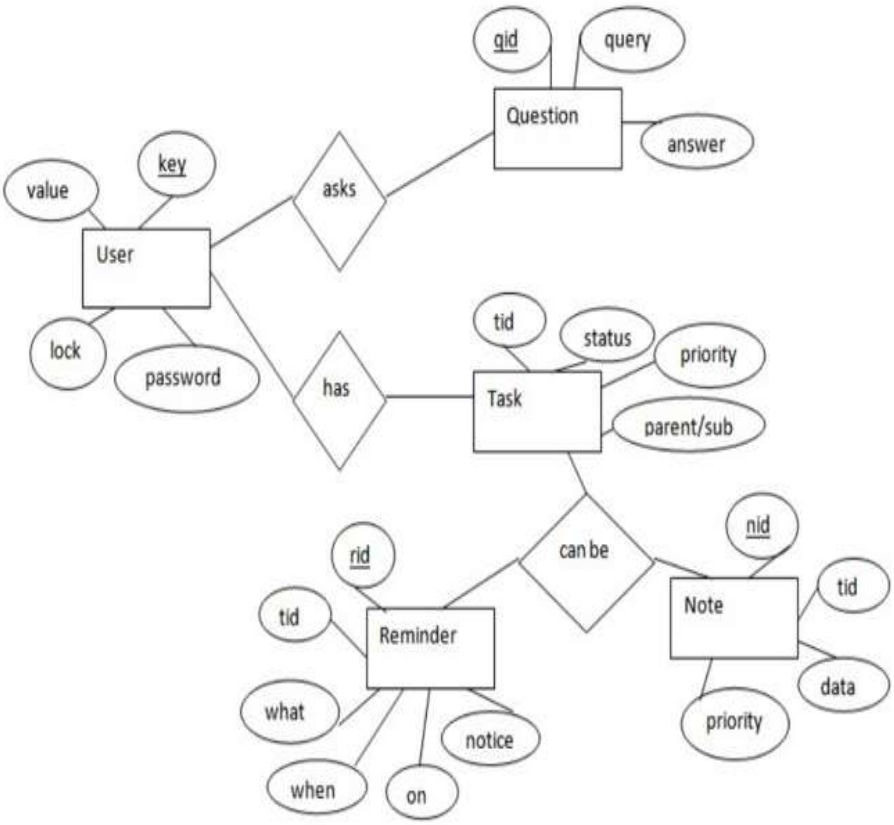

Figure 3.2 ER Diagram for JARVIS

The above diagram shows entities and their relationship for a virtual system. We have a user of system who can have their keys and values it can be used to store any information about the user. See, for key “name” value can be “Jim.” For some keys user might like to keep secure. There we 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 that is their current state.

The task should also have a priority value and its category whether it is a parent cost or child task of an older task.

3.3. ACTIVITY DIAGRAM

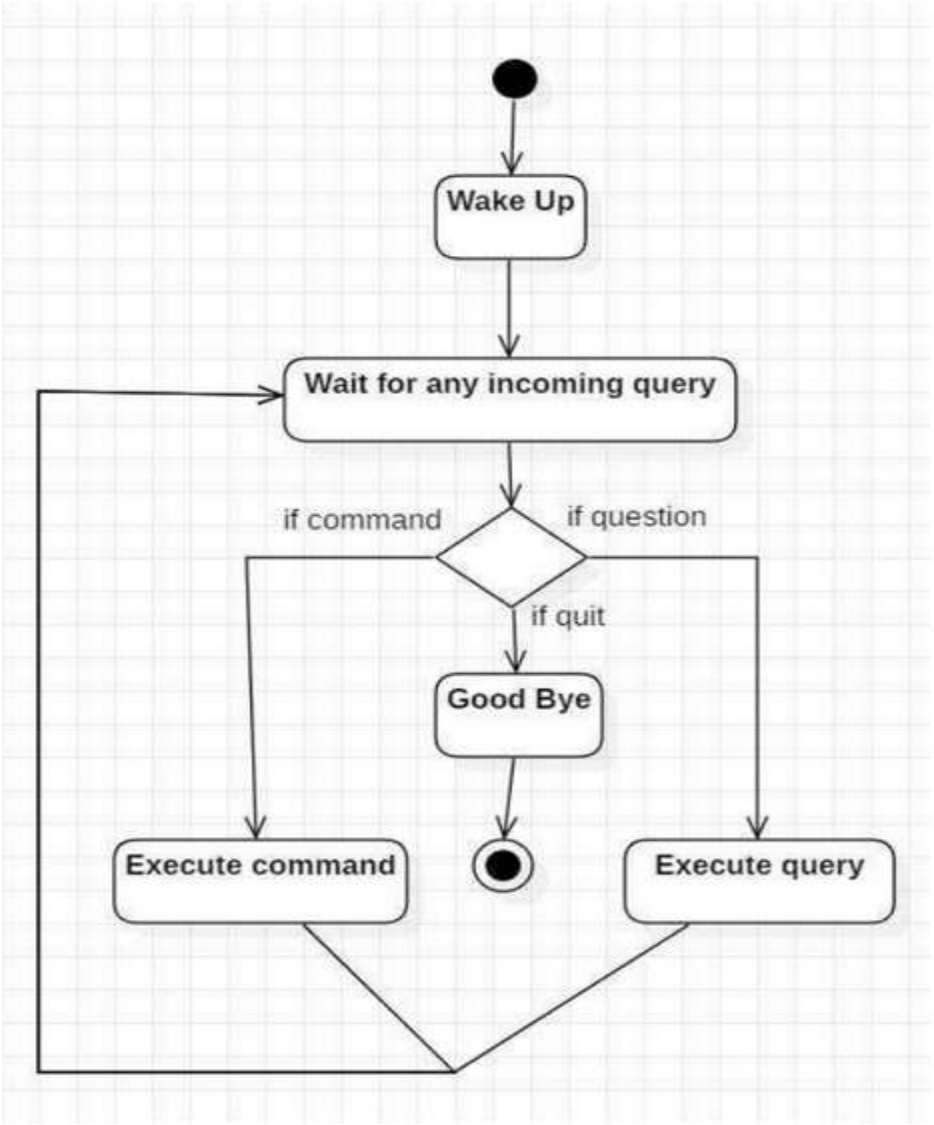

Figure 3.3 Activity Diagram for JARVIS

Initially, the system is in idle mode. As it receives any wakeup call it begins execution. The received command is identified whether it is a question 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 quick command. At that moment, it goes back to sleep.

3.4. SEQUENCE DIAGRAM

3.4.1. Sequence Diagram for Query-Response

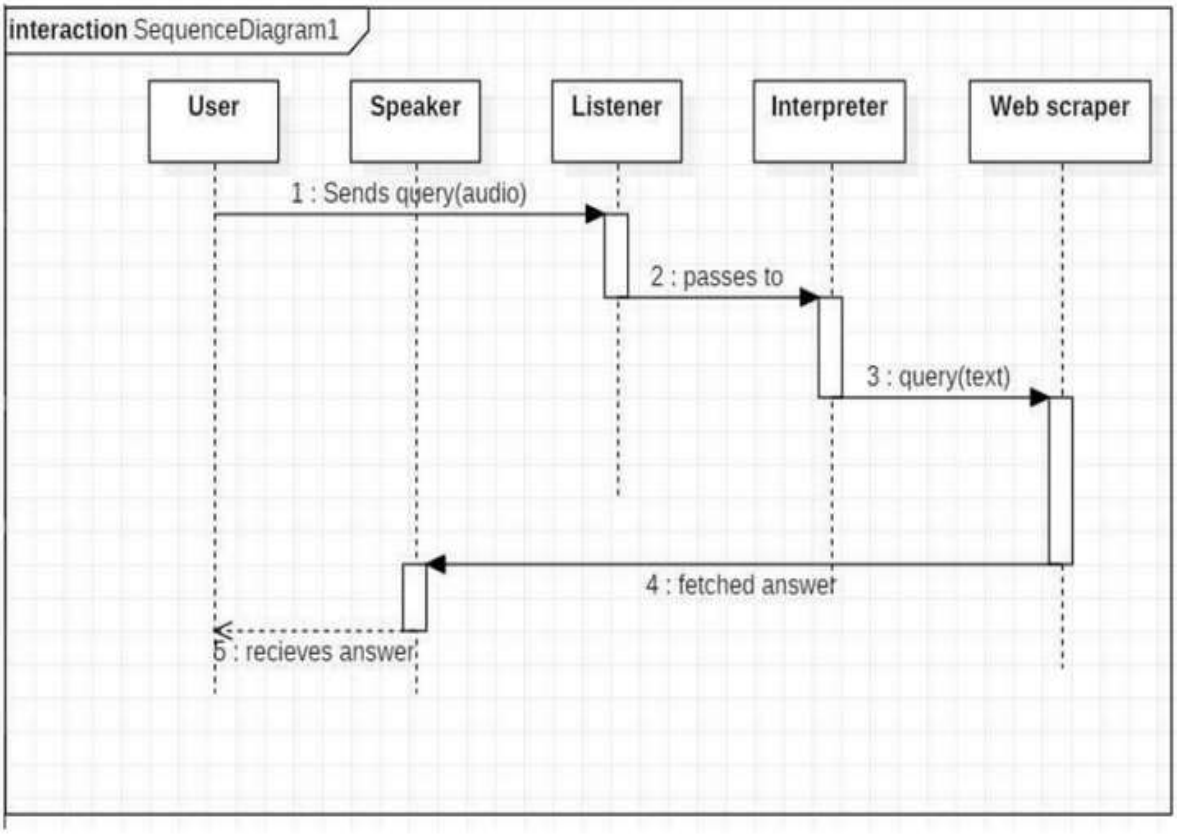

Figure 3.4 Sequence Diagram for Query-Response in JARVIS

The above sequence diagram shows how and answer ask why 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 back to speaker, where it speaks the answer to the user.

3.4.2. Sequence Diagram for Task Execution

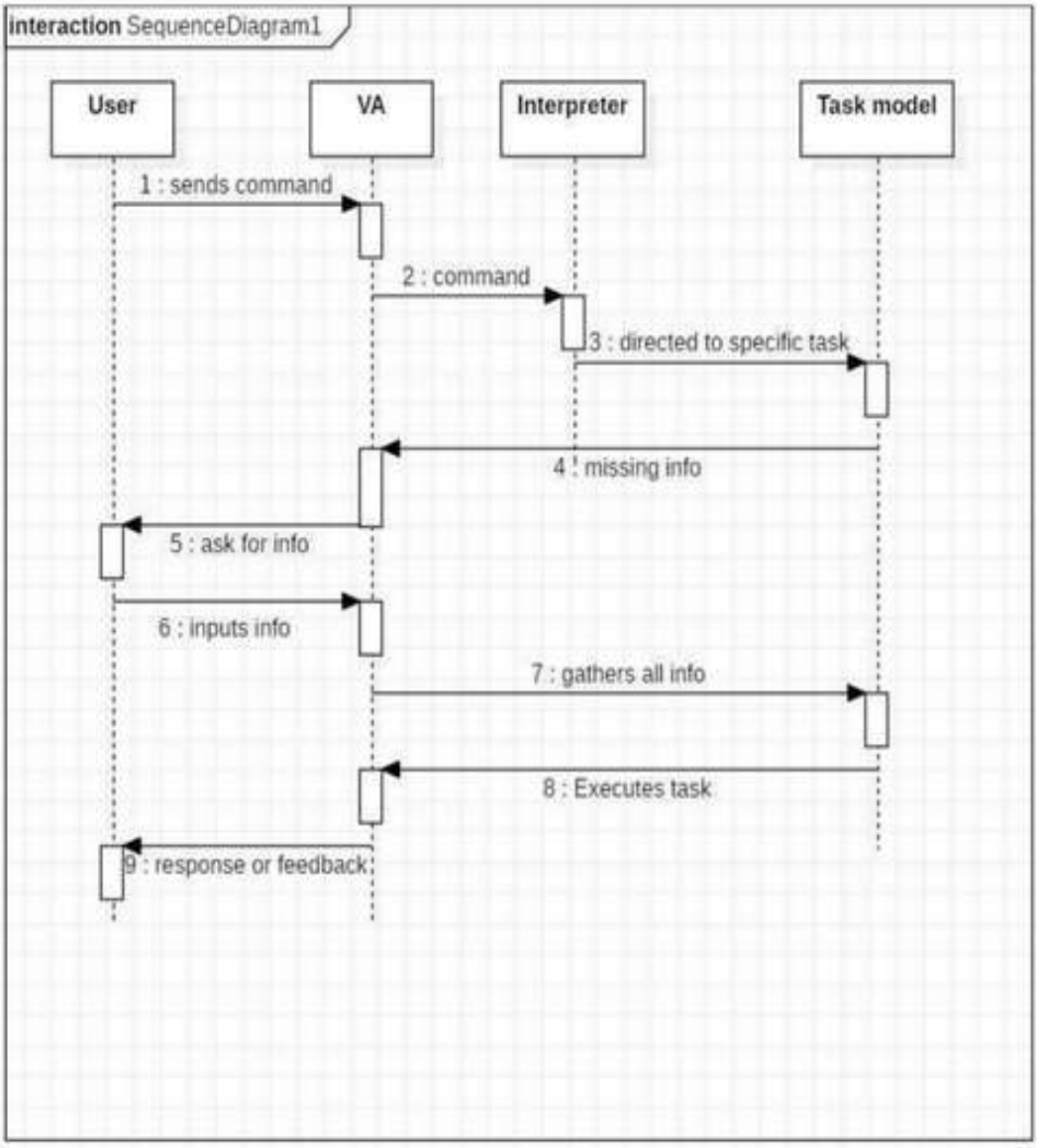

Figure 3.5 Sequence Diagram for Task Execution in JARVIS

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

3.5. DATA FLOW DIAGRAM (DFD)

3.5.1. DFD Level 0 (Context Level Diagram)

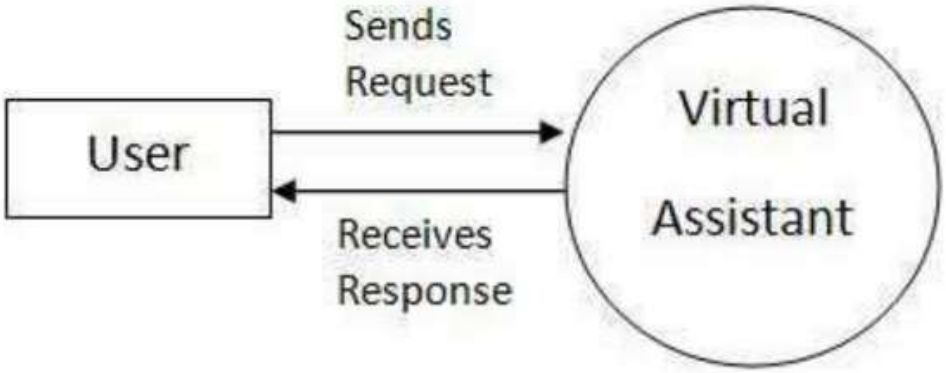

Figure 3.6 DFD Level 0

3.5.2. DFD Leve 1

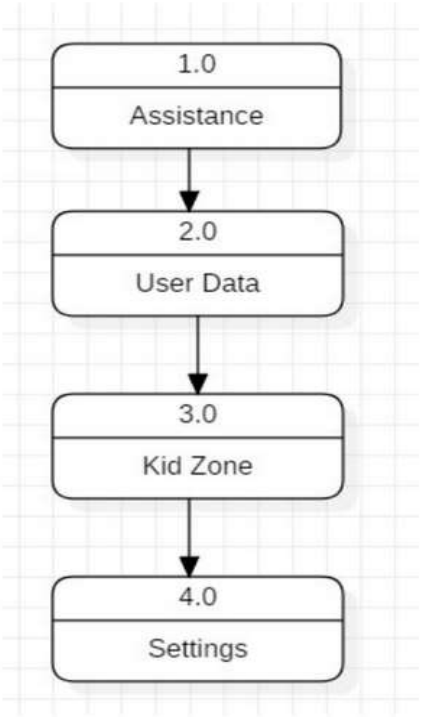

Figure 3.7 DFD Level 1

3.5.3. DFD Level 2

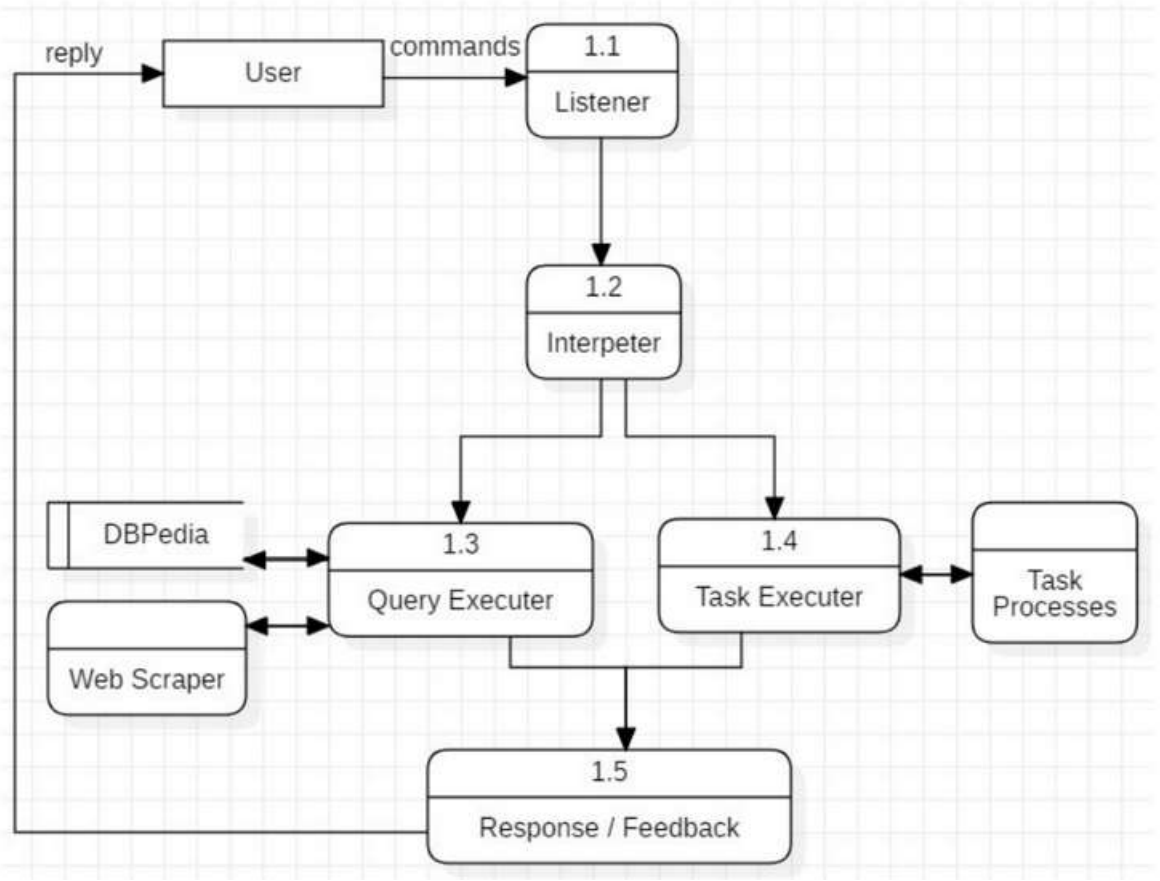

Figure 3.8 DFD Level 2

CHAPTER 4: SOFTWARE DETAILS

The IDE used in this project is PyCharm. All the python files were created in PyCharm and all the necessary packages were easily installable in this IDE. For this project following modules and libraries were used i.e., pyttsx3, SpeechRecognition, Datetime, Wikipedia, Smtplib, pywhatkit, pyjokes, pyPDF2, pyautogui, pyQt etc. We have created a live GUI for interacting with the JARVIS as it gives a design and interesting look while having the conversation.

4.1. PYCHARM

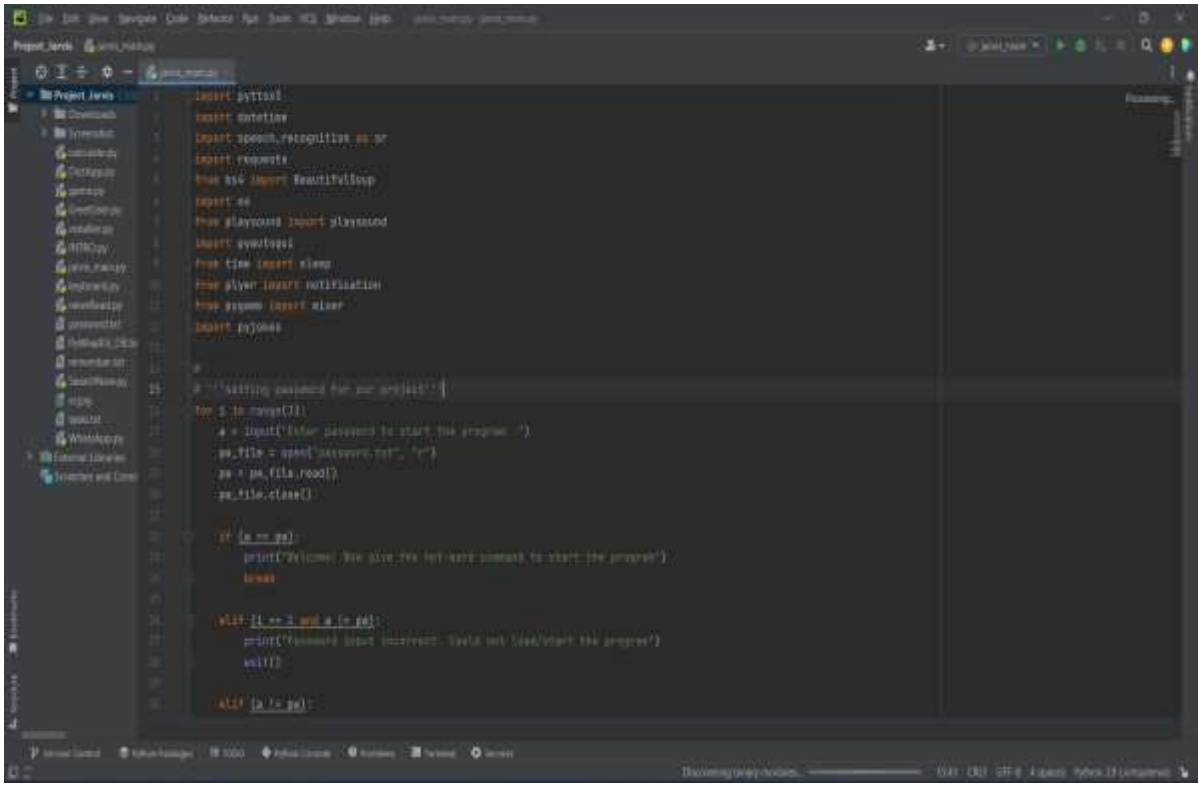

Figure 4.1 PyCharm IDE

It is an IDE i.e., Integrated Development Environment which has many features like it supports scientific tools (like matplotlib, numpy, scipy etc) web frameworks (example Django, web2py and Flask) refactoring in Python, integrated python debugger, code completion, code, and project navigation etc. It also provides Data Science when used with Anaconda.

4.2. PYQT5 FOR LIVE GUI

PyQt5 is the most important python binding. It contains set of GUI widgets. PyQt5 has some important python modules like QTWidgets, QtCore, QtGui, and QtDesigner etc.

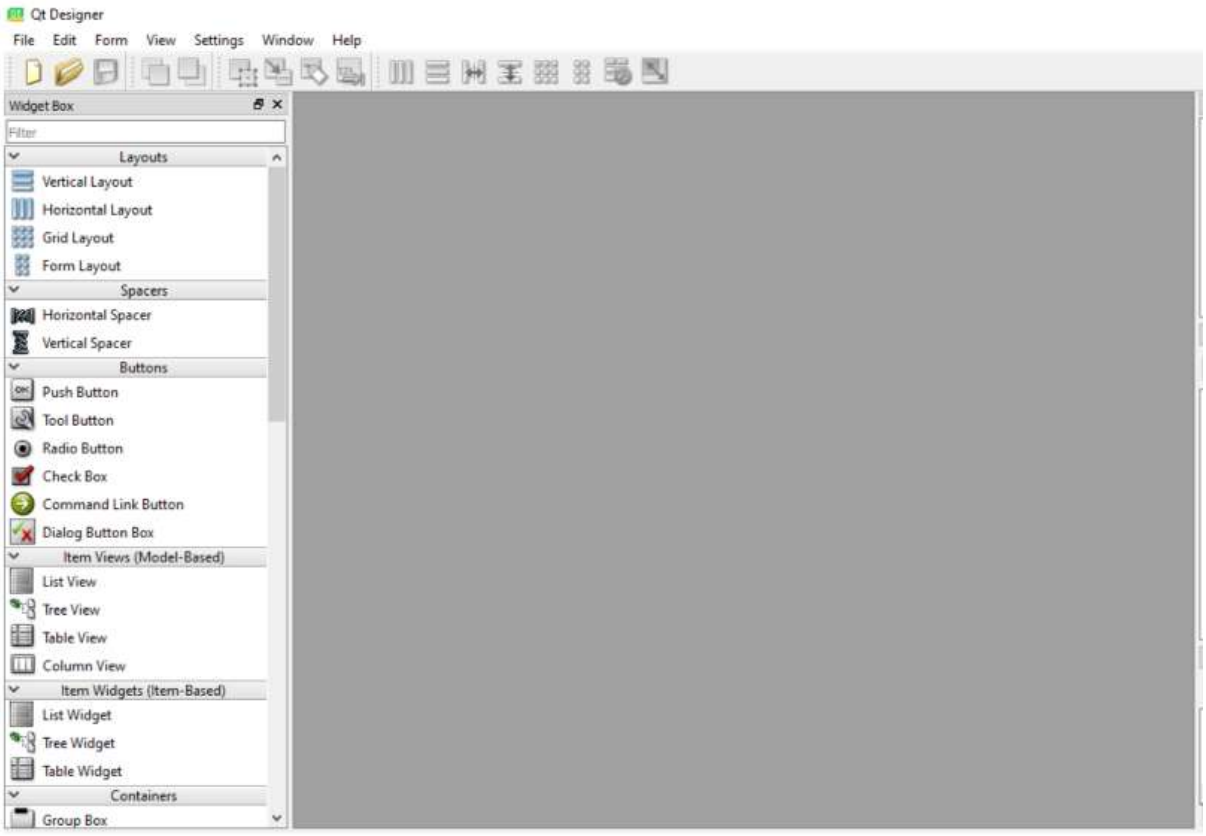

Figure 4.2 PyQt5

4.3. PYTHON LIBRARIES

In JARVIS following python libraries were used:

4.3.1. pyttsx3: It is a python library which converts text to speech.

4.3.2. Speech Recognition: It is a python module which converts speech to text.

4.3.3. pywhatkit: It is python library to send WhatsApp message at a particular time with some additional features.

4.3.4. Datetime: This library provides us the actual date and time.

4.3.5. Wikipedia: It is a python module for searching anything on Wikipedia.

4.3.6. Smtplib: Simple mail transfer protocol that allows us to send mails and to route mails between mail servers.

4.3.7. pyPDF2: It is a python module which can read, split, merge any PDF.

4.3.8. Pyjokes: It is a python library which contains lots of interesting jokes in it.

4.3.9. Webbrowser: It provides interface for displaying web-based documents to users.

4.3.10. Pyautogui: It is a python library for graphical user interface.

4.3.11. os: It represents Operating System related functionality.

4.3.12. sys: It allows operating on the interpreter as it provides access to the variables and functions that usually interact strongly with the interpreter.

```
import pyttsx3
import datetime
import speech_recognition as sr
import requests
from bs4 import BeautifulSoup
import os
from playsound import playsound
import pyautogui
from time import sleep
from plyer import notification
from pygame import mixer
import pyjokes
```

Figure 4.3 Imported Modules

CHAPTER 5: IMPLEMENTATION WORK DETAILS

JARVIS, a desktop assistant is a voice assistant that can perform many daily tasks of desktop like playing music, opening your favourite IDE with the help of a single voice command. Jarvis is different from other traditional voice assistants in terms that it is specific to desktop and user does not need to make account to use this, it does not require any internet connection while getting the instructions to perform any specific task.

5.1. REAL LIFE APPLICATION

5.1.1. Saves time: JARVIS is a desktop voice assistant which works on the voice command offered to it, it can do voice searching, voice-activated device control and can let us complete a set of tasks.

5.1.2. Conversational interaction: It makes it easier to complete any task as it automatically does it by using the essential module or libraries of Python, in a conversational interaction way. Hence any user when instruct any task to it, they feel like giving task to a human assistant because of the conversational interaction for giving input and getting the desired output in the form of task done.

5.1.3. Reactive nature: The desktop assistant is reactive which means it know human language very well and understand the context that is provided by the user and gives response in the same way, i.e., human understandable language, English. So, user finds its reaction in an informed and smart way.

5.1.4. Multitasking: The main application of it can be its multitasking ability. It can ask for continuous instruction one after other until the user "QUIT" it.

5.1.5. No Trigger phase: It asks for the instruction and listen the response that is given by user without needing any trigger phase and then only executes the task.

5.2. DATA IMPLEMENTATION AND PROGRAM EXECUTION

As the first step, install all the necessary packages and libraries. The command used to install the libraries is "*pip install*" and then import it. The necessary packages included are as follows:

5.2.1. LIBRARIES AND PACKAGES

5.2.2.1. pyttsx3: It is a python library which converts text to speech.

5.2.2.2. speechrecognition: It is a python module which converts speech to text.

5.2.2.3. pywhatkit: It is python library to send WhatsApp message at a particular time with some additional features.

5.2.2.4. datetime: This library provides us the actual date and time.

5.2.2.5. wikipedia: It is a python module for searching anything on Wikipedia.

5.2.2.6. Smtplib: Simple mail transfer protocol that allows us to send mails and to route mails between mail servers.

5.2.2.7. pyPDF2: It is a python module which can read, split, merge any PDF.

5.2.2.8. pyjokes: It is a python library which contains lots of interesting jokes in it.

5.2.2.9. webbrowser: It provides interface for displaying web-based documents to users.

5.2.2.10. pyautogui: It is a python library for graphical user interface.

5.2.2.11. os: It represents Operating System related functionality.

5.2.2.12. sys: It allows operating on the interpreter as it provides access to the variables and functions that usually interact strongly with the interpreter.

5.2.2. FUNCTIONS

5.2.2.1. takeCommand(): The function is used to take the command as input through microphone of user and returns the output as string.

5.2.2.2. GreetMe(): This function greets the user according to the time like

Good Morning, Good Afternoon and Good Evening.

5.2.2.3. taskExecution(): This is the function which contains all the necessary task execution definition like sendEmail(), pdf_reader(), news() and many conditions in if condition like "open google", "open notepad", "search on Wikipedia", “play music" and "open command prompt" etc.

CHAPTER 6: SOURCE CODE AND COMMANDS

6.1. installer.py

```
import pip
pip.main(['install', 'speechRecognition', 'requests', 'plyer',
'pywhatkit', 'pyaudio', 'wikipedia', 'bs4', 'playsound',
'python-decouple', 'pyautogui', 'pynput', 'wolframalpha',
'pygame', 'vosk', 'googletrans', 'gtts', 'pykeyboard',
'pyjokes'])
```

6.2. GreetUser.py

```
import pyttsx3
import datetime
engine = pyttsx3.init("sapi5")
voices = engine.getProperty("voices")
engine.setProperty("voice", voices[0].id)
engine.setProperty("rate",200)
def speak(audio):
    engine.say(audio)
    engine.runAndWait()
def greetMe():
    hour  = int(datetime.datetime.now().hour)
    if hour>=5 and hour<=12:
        speak("Good Morning, sir")
    elif hour >12 and hour<=16:
        speak("Good Afternoon ,sir")
    else:
        speak("Good Evening, sir")
    speak("I am Jarvis your personal ai assistant, please tell
me, how may I help you ?")
```

6.3. SearchNow.py

```
import webbrowser
import speech_recognition as sr
import pyttsx3
import pywhatkit
import wikipedia
def takeCommand():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        print("Listening...")
        r.pause_threshold = 1
        r.energy_threshold = 400
        audio = r.listen(source, 0, 4)
        try:
            print("Recognizing...")
            query = r.recognize_google(audio, language='en-
in')
            print(f"User Said: {query}\n")
        except Exception as e:
            return "None"
        return query
query = takeCommand().lower()
engine = pyttsx3.init("sapi5")
voices = engine.getProperty("voices")
engine.setProperty('voice', voices[0].id)
engine.setProperty("rate", 200)
def speak(audio):
    engine.say(audio)
    engine.runAndWait()
def searchGoogle(query):
    if "google" in query:
```

```
        import wikipedia as googleScrap
        speak("working on that, sir")
        query = query.replace("jarvis", "")
        query = query.replace("google search", "")
        query = query.replace("search", "")
        query = query.replace("on", "")
        query = query.replace("google", "")
        query = query.replace("wikipedia", "")
        speak(f"{query}")
        speak("This is what, I found on, Google")
        try:
            pywhatkit.search(query)
            result = googleScrap.summary(query, 2)
            speak(result)
        except:
            speak("No speakale output available")
def searchYouTube(query):
    if "youtube" in query:
        speak("working on that, sir")
        query = query.replace("jarvis", "")
        query = query.replace("youtube", "")
        query = query.replace("youtube search", "")
        query = query.replace("search", "")
        query = query.replace("on", "")
        web = "https://www.youtube.com/results?search_query="
+ query
        speak(f"searching {query} on youtube")
        webbrowser.open(web)
        pywhatkit.playonyt(query)
        speak("This is what i found for your search on
youtube")
```

```
        speak("Done, sir")
def searchWikipedia(query):
    if "wikipedia" in query:
        speak("Searching from wikipedia...")
        query = query.replace("jarvis", "")
        query = query.replace("wikipedia", "")
        query = query.replace("search wikipedia", "")
        query = query.replace("search", "")
        query = query.replace("on", "")
        results = wikipedia.summary(query, sentences = 2)
        speak("According to wikipedia")
        print(results)
        speak(results)
```

6.4. keyboard.py

```
from pynput.keyboard import Key, Controller
from time import sleep
keyboard = Controller()
def volumeUp():
    for i in range(5):
        keyboard.press(Key.media_volume_up)
        keyboard.release(Key.media_volume_up)
        sleep(0.1)
def volumeDown():
    for i in range(5):
        keyboard.press(Key.media_volume_down)
        keyboard.release(Key.media_volume_down)
        sleep(0.1)
```

6.5. DictApp.py

```
import pyttsx3
```

```
import os
import pyautogui
import webbrowser
from time import sleep
# start engine property
engine = pyttsx3.init("sapi5")
# providing voice to the assistant
voices = engine.getProperty("voices")
engine.setProperty('voice', voices[0].id)
# setting the rate of voice
engine.setProperty("rate", 200)
# talk to the user through device's speaker
def speak(audio):
    engine.say(audio)
    engine.runAndWait()
dictApp = {
    "command prompt" : "cmd",
    "word" : "winword",
    "excel" : "EXCEL",
    "google chrome" : "chrome",
    "pycharm" : "pycharm64",
    "edge" : "msedge",
    "browser" : "firefox",
    "powerpoint" : "POWERPNT",
    "powershell" : "powershell",
    "one note" : "onenote",
    "media player" : "VLC",
    "sublime text" : "C:\\Program Files\\Sublime
Text\\sublime_text"
}
def openAppWeb(query):
```

```
    speak("Working on that, sir")
    if ".com" in query or ".co.in" in query or ".org" or
".ac.in" in query:
        query = query.replace("open", "")
        query = query.replace("jarvis", "")
        query = query.replace("launch", "")
        query = query.replace("slash", "/")
        query = query.replace(" ", "")
        webbrowser.open(f"https://www.{query}")
        speak(f"Opening {query}")
    else:
        keys = list(dictApp.keys())
        for app in keys:
            if app in query:
                speak(f"Opening {app}")
                os.system(f"start {dictApp[app]}")
def closeAppWeb(query):
    speak("Closing, sir")
    if "one tab" in query or "1 tab" in query:
        pyautogui.hotkey("ctrl","w")
        speak("All tabs are closed, sir")
    elif "to tabs" in query or "2 tabs" in query or "2 tab" in
query or "to tab" in query or "too tabs" in query or "too tab"
in query:
        pyautogui.hotkey("ctrl","w")
        sleep(0.5)
        pyautogui.hotkey("ctrl","w")
        speak("All tabs are closed, sir")
    else:
        keys = list(dictApp.keys())
        for app in keys:
```

```
            if app in query:
                os.system(f"taskkill /f /im
{dictApp[app]}.exe")
```

6.6. newsRead.py

```
import requests
import json
import pyttsx3
import speech_recognition as sr
# start engine property
engine = pyttsx3.init("sapi5")
# providing voice to the ssistant
voices = engine.getProperty("voices")
engine.setProperty('voice', voices[0].id)
# setting the rate of voice
engine.setProperty("rate", 200)
# talk to the user through device's speaker
def speak(audio):
    engine.say(audio)
    engine.runAndWait()
def takeCommand():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        print("Listening.....")
        r.pause_threshold = 1
        r.energy_threshold = 300
        audio = r.listen(source, 0, 4)
    try:
        print("Recognising...")
        query  = r.recognize_google(audio, language='en-in')
        print(f"You Said: {query}\n")
```

```
    except Exception as e:
        print()
        return "None"
    return query
def latestNews():
    apidict = {
        "top headlines" : "https://newsapi.org/v2/top-
headlines?country=in&apiKey=f0acff549fdd4964a3b19eccfa93087f",
        "politics" : "https://newsapi.org/v2/top-
headlines?country=in&apiKey=f0acff549fdd4964a3b19eccfa93087f",
        "international" :
"https://newsapi.org/v2/everything?q=tesla&from=2022-08-
23&sortBy=publishedAt&apiKey=f0acff549fdd4964a3b19eccfa93087f"
,
        "business" : "https://newsapi.org/v2/top-
headlines?country=in&category=business&apiKey=f0acff549fdd4964
a3b19eccfa93087f",
        "health" : "https://newsapi.org/v2/top-
headlines?country=in&category=health&apiKey=f0acff549fdd4964a3
b19eccfa93087f",
        "science" : "https://newsapi.org/v2/top-
headlines?country=in&category=science&apiKey=f0acff549fdd4964a
3b19eccfa93087f",
        "sports" : "https://newsapi.org/v2/top-
headlines?country=in&category=sports&apiKey=f0acff549fdd4964a3
b19eccfa93087f",
        "tech crunch" : "https://newsapi.org/v2/top-
headlines?sources=techcrunch&apiKey=f0acff549fdd4964a3b19eccfa
93087f",
```

```
        "technology" : "https://newsapi.org/v2/top-
headlines?country=in&category=technology&apiKey=f0acff549fdd49
64a3b19eccfa93087f",
        "entertainment" : "https://newsapi.org/v2/top-
headlines?country=in&category=entertainment&apiKey=f0acff549fd
d4964a3b19eccfa93087f",
        "google news" :
"https://newsapi.org/v2/everything?q=bitcoin&from=2022-08-
23&sortBy=publishedAt&apiKey=f0acff549fdd4964a3b19eccfa93087f"
,
        "tesla" :
"https://newsapi.org/v2/everything?q=tesla&from=2022-08-
24&sortBy=publishedAt&apiKey=f0acff549fdd4964a3b19eccfa93087f"
,
        "apple" :
"https://newsapi.org/v2/everything?q=apple&from=2022-09-
23&to=2022-09-
23&sortBy=popularity&apiKey=f0acff549fdd4964a3b19eccfa93087f",
        "google news" : "https://newsapi.org/v2/top-
headlines?sources=google-news-
in&apiKey=f0acff549fdd4964a3b19eccfa93087f",
        "finance" : "https://newsapi.org/v2/top-
headlines?country=in&category=business&apiKey=f0acff549fdd4964
a3b19eccfa93087f"
        }
    content = None
    url = None
    speak("Which field news do you want hear, sir")
    field = takeCommand()
    for key,value in apidict.items():
        if key.lower() in field.lower():
```

```
            url = value
            print(url)
            break
        else:
            url = True
    if url is True:
        print("url is not found")
    news = requests.get(url).text
    news = json.loads(news)
    speak("here is the first news.")
    arts = news["articles"]
    for articles in arts:
        article = articles["title"]
        print(article)
        speak(article)
        news_url = articles["url"]
        print(f"for more info visit: {news_url}")
        speak("do you wish to listen more news, sir")
        print("[Speak Yes to continue] and [No to stop]")
        a = takeCommand()
        if str(a) == "yes":
            pass
        elif str(a) == "no":
            break
    speak("that's all, sir")
```

6.7. calculate.py

```
import wolframalpha
import pyttsx3
import speech_recognition as sr
# start engine property
```

```
engine = pyttsx3.init("sapi5")
# providing voice to the assistant
voices = engine.getProperty("voices")
engine.setProperty('voice', voices[0].id)
# setting the rate of voice
engine.setProperty("rate", 200)
# talk to the user through device's speaker
def speak(audio):
    engine.say(audio)
    engine.runAndWait()
def WolfRamAlpha(query):
    apikey = "EVX5V8-LE8YAKW2UT"
    requester = wolframalpha.Client(apikey)
    requested = requester.query(query)
    try:
        answer = next(requested.results).text
        return answer
    except:
        speak("Value is not answerable, sir")
def Calculator(query):
    term = str(query)
    term = term.replace("jarvis", "")
    term = term.replace("Jarvis", "")
    term = term.replace("multiply", "*")
    term = term.replace("into", "*")
    term = term.replace("plus", "+")
    term = term.replace("minus", "-")
    term = term.replace("divided by", "/")
    final = str(term)
    try:
        result = WolfRamAlpha(final)
```

```
        print(f"{result}")
        speak(f"Sir, the answer is: {result}")
    except:
        speak("Sorry sir, The value is not answerable")
```

6.8. game.py

```
import pyttsx3
import speech_recognition as sr
import random
engine = pyttsx3.init('sapi5')
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[0].id)
engine.setProperty("rate", 170)
def speak(audio):
    engine.say(audio)
    engine.runAndWait()
def takeCommand():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        print("Listening.....")
        r.pause_threshold = 1
        r.energy_threshold = 300
        audio = r.listen(source, 0, 4)
    try:
        print("Recognizing..")
        query = r.recognize_google(audio, language='en-in')
        print(f"You Said : {query}\n")
    except Exception as e:
        print("Say that again")
        return "None"
    return query
```

```
def game_play():
    speak("Lets Play ROCK PAPER SCISSORS !!")
    print("LETS PLAYYYYYYYYYYYYY")
    i = 0
    Me_score = 0
    Com_score = 0
    while (i < 5):
        choose = ("rock", "paper", "scissors", "thread")  #
Tuple
        com_choose = random.choice(choose)
        query = takeCommand().lower()
        if query == "rock":
            if com_choose == "rock":
                speak("ROCK")
                print(f"Score:\nME: {Me_score}\nJARVIS:
{Com_score}")
            elif com_choose == "paper":
                speak("paper")
                Com_score += 1
                print(f"Score:\nME: {Me_score}\nJARVIS:
{Com_score}")
            elif com_choose == "scissors":
                speak("Scissors")
                Me_score += 1
                print(f"Score:\nME: {Me_score}\nJARVIS:
{Com_score}")
            else:
                speak("thread")
                Com_score += 1
                print(f"Score:\nME: {Me_score}\nJARVIS:
{Com_score}")
```

```
        elif query == "paper":
            if com_choose == "rock":
                speak("ROCK")
                Me_score += 1
                print(f"Score:\nME: {Me_score}\nJARVIS:
{Com_score}")
            elif com_choose == "paper":
                speak("paper")
                print(f"Score:\nME: {Me_score}\nJARVIS:
{Com_score}")
            elif com_choose == "thread":
                speak("thread")
                Me_score += 1
                print(f"Score:\nME: {Me_score}\nJARVIS:
{Com_score}")
            else:
                speak("Scissors")
                Com_score += 1
                print(f"Score:\nME: {Me_score}\nJARVIS:
{Com_score}")
        elif query=="scissors" or query=="scissor" or
query=="caesar":
            if com_choose == "rock":
                speak("ROCK")
                Com_score += 1
                print(f"Score:\nME: {Me_score}\nJARVIS:
{Com_score}")
            elif com_choose == "paper":
                speak("paper")
                Me_score += 1
```

```
                print(f"Score:\nME: {Me_score}\nJARVIS:
{Com_score}")
            elif com_choose == "thread":
                speak("thread")
                Me_score += 1
                print(f"Score:\nME: {Me_score}\nJARVIS:
{Com_score}")
            else:
                speak("Scissors")
                print(f"Score:\nME: {Me_score}\nJARVIS:
{Com_score}")
        i += 1
    print()
    print(f"FINAL SCORE:\nME: {Me_score}\nJARVIS:
{Com_score}")
    if Me_score > Com_score:
        print("YOU WON THE GAME")
        speak("Congratulation sir, you won the game")
    elif Me_score < Com_score:
        print("JARVIS WON THE GAME")
        speak("Hurray!! I won the game")
    else:
        print("GAME ENDS IN A DRAW!!!")
        speak("That was a nice game, sir")
```

6.9. INTRO.py

```
from tkinter import *  # pip install tkinter
from PIL import Image, ImageTk, ImageSequence  # pip install
Pillow
import time
import pygame  # pip install pygame
```

```
from pygame import mixer
mixer.init()
root = Tk()
root.geometry("1000x500")
def play_gif():
    root.lift()
    root.attributes("-topmost", True)
    global img
img=Image.open(r"C:\Users\Mukund\PycharmProjects\Project_Jarvi
s\Downloads\jarvis.gif)"
    lbl = Label(root)
    lbl.place(x=0, y=0)
    i = 0
mixer.music.load(r"C:\Users\Mukund\PycharmProjects\Project_Jar
vis\Downloads\jarvis_Sound.mp3") #enter the music file address
    mixer.music.play()
    for img in ImageSequence.Iterator(img):
        img = img.resize((1000, 500))
        img = ImageTk.PhotoImage(img)
        lbl.config(image=img)
        root.update()
        time.sleep(0.05)
    root.destroy()
play_gif()
root.mainloop()
```

6.10. jarvis_main.py

```
import pyttsx3
import datetime
import speech_recognition as sr
import requests
from bs4 import BeautifulSoup
```

```
import os
from playsound import playsound
import pyautogui
from time import sleep
from plyer import notification
from pygame import mixer
import pyjokes
# '''setting password for our project'''
for i in range(3):
    a = input("Enter password to start the program: ")
    pw_file = open("password.txt", "r")
    pw = pw_file.read()
    pw_file.close()
    if (a == pw):
        print("Welcome! Now give the hot-word command to start
the program")
        break
    elif (i == 2 and a != pw):
        print("Password input incorrect. Could not load/start
the program")
        exit()
    elif (a != pw):
        print("Try Again")
# INTRO GIF
from INTRO import play_gif
play_gif
engine = pyttsx3.init("sapi5")
voices = engine.getProperty("voices")
engine.setProperty("voice", voices[0].id)
rate = engine.setProperty("rate", 170)
def speak(audio):
```

```
    engine.say(audio)
    engine.runAndWait()
def takeCommand():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        print("Listening.....")
        r.pause_threshold = 1
        r.energy_threshold = 300
        audio = r.listen(source, 0, 4)
    try:
        print("Recognising...")
        query = r.recognize_google(audio, language='en-in')
        print(f"You Said: {query}\n")
    except Exception as e:
        print()
        return "None"
    return query
if __name__ == "__main__":
    while True:
        query = takeCommand().lower()
        if "wake up" in query:
            from GreetUser import greetMe
            greetMe()
            while True:
                query = takeCommand().lower()
                if "go to sleep" in query:
                    speak("Ok sir , You can me call anytime")
                    break
                # tell me the time
                if 'time' in query:
```

```
                    strTime =
datetime.datetime.now().strftime("%H:%M")
                    print(strTime)
                    speak(f"Sir, the time is {strTime}")
                # tell me the date
                elif 'date' in query:

strDate=datetime.datetime.now().strftime("%m-%d-%Y")
                    print(strDate)
                    speak(f"Sir, the date is {strDate}")
                # normal conversation
                elif "hello" in query:
                    speak("Hello sir, how are you ?")
                elif "i am fine" in query:
                    speak("that's great, sir")
                elif "how are you" in query:
                    speak("Perfect, sir")
                elif "thank you" in query:
                    speak("you are welcome, sir")
                # easy method to open application
                elif "open" in query:
                    query = query.replace("open", "")
                    query = query.replace("jarvis", "")
                    pyautogui.press("super")
                    pyautogui.typewrite(query)
                    pyautogui.sleep(2)
                    pyautogui.press("enter")
                    speak(f"Opening {query}")
                # open and close apps and website
                elif "open" in query:
                    from DictApp import openAppWeb
```

```
        openAppWeb(query)
    elif "close" in query:
        from DictApp import closeAppWeb
        closeAppWeb(query)
    # find something on browser
    elif "google" in query:
        from SearchNow import searchGoogle
        searchGoogle(query)
    elif "youtube" in query:
        from SearchNow import searchYouTube
        searchYouTube(query)
    elif "wikipedia" in query:
        from SearchNow import searchWikipedia
        searchWikipedia(query)
    # news api
    elif "news" in query:
        from newsRead import latestNews
        latestNews()
    # youtube controls
    elif "pause" in query:
        pyautogui.press("k")
        speak("Video paused, sir")
    elif "play" in query:
        pyautogui.press("k")
        speak("video played, sir")
    elif "mute" in query:
        pyautogui.press("m")
        speak("Video Muted, Sir")
    elif "full screen" in query:
        pyautogui.press("f")
    elif "mini player" in query:
```

```
                pyautogui.press("f")
            elif "normal screen" in query:
                pyautogui.press("i")
            elif "theatre mode" in query:
                pyautogui.press("t")
            elif "subtitle" in query:
                pyautogui.press("c")
            elif "next" in query:
                pyautogui.hotkey("shift", "n")
                sleep(0.5)
                speak("Playing new video, sir")
            elif "previous" in query:
                pyautogui.hotkey("shift", "p")
                sleep(0.5)
                speak("Playing previous video, sir")
            elif "unmute" in query:
                pyautogui.press("m")
                speak("Video Un Muted, Sir")
            elif "volume up" in query:
                from keyboard import volumeUp
                speak("Turning volume up, sir")
                volumeUp()
            elif "volume down" in query:
                from keyboard import volumeDown
                speak("Turning volume down, sir")
                volumeDown()
            elif "temperature" in query:
                search = "temperature at my location"
                url =
f"https://www.google.com/search?q={search}"
                r = requests.get(url)
```

```
                    data = BeautifulSoup(r.text,
"html.parser")
                    temp = data.find("div",
class_="BNeawe").text
                    print(f"Temperature: {temp}")
                    speak(f"current temperature at your
location is {temp}")
                # making calculator
                elif "calculate" in query:
                    from calculate import WolfRamAlpha
                    from calculate import Calculator
                    query = query.replace("calculate", "")
                    query = query.replace("jarvis", "")
                    Calculator(query)
                # reminder program
                elif "remember that" in query:
                    rememberMessage = query.replace("rememebr
that", "")
                    rememberMessage = query.replace("jarvis",
"")
                    rememberMessage =
query.replace("rememebr", "")
                    rememberMessage = query.replace("that",
"")
                    speak("Sir, You told me to  " +
rememberMessage)
                    remember = open("remember.txt", "w")
                    remember.write(rememberMessage)
                    remember.close()
                elif "what do you remember" in query:
                    rememebr = open("remember.txt", "r")
```

```
                    speak("Sir, You told me to " +
rememebr.read())
                # sleep jarvis
                elif "break" in query:
                    speak("Ok Sir, You can call me anytime")
                    break
                # shutdown jarvis
                elif "goodbye" in query:
                    speak("Good Bye Sir. Hope we will meet
again")
                    exit()
                elif "shutdown" in query:
                    speak("Do you really want to shut down the
system, sir")
                    print("0: No")
                    print("1: Yes")
                    shutdown = input("Do you really wish so?
(0/1): ")
                    if shutdown == "1":
                        speak("Shutting down the system, sir,
good bye")
                        os.system("shutdown /s /t 1")
                    elif shutdown == "0":
                        break
                # change password
                elif "change password" in query:
                    speak("What is the new password, sir")
                    new_pw = input("Enter New Password: ")
                    new_password = open("password.txt", "w")
                    new_password.write(new_pw)
                    new_password.close()
```

```
                    speak("Password changed successfully,
Sir")
                    print("Password changed successfully")
                # schedule my day function
                elif "schedule my day" in query:
                    tasks = []
                    speak("Do you want to clear old tasks (Yes
or No)")
                    query = takeCommand().lower()
                    if "yes" in query:
                        file =
open(r"C:\Users\Mukund\PycharmProjects\Project_Jarvis\tasks.tx
t", 'w')
                        file.write(f"")
                        file.close()
                        no_tasks=int(input("Enter the number
of tasks: "))
                        i = 0
                        for i in range(no_tasks):
                            tasks.append(input(f"Enter task
{i}: "))
                            file =
open(r"C:\Users\Mukund\PycharmProjects\Project_Jarvis\tasks.tx
t", 'a')
                            file.write(f"{i}. {tasks[i]} \n")
                            file.close()
                    elif "no" in query:
                        no_tasks = int(input("Enter the number
of tasks: "))
                        i = 0
                        for i in range(no_tasks):
```

```
                            tasks.append(input(f"Enter task
{i}: "))
                            file =
open(r"C:\Users\Mukund\PycharmProjects\Project_Jarvis\tasks.tx
t", 'a')
                            file.write(f"{i}. {tasks[i]} \n")
                            file.close()
                # show schedule
                elif "show my schedule" in query:
                    file =
open(r"C:\Users\Mukund\PycharmProjects\Project_Jarvis\tasks.tx
t", 'r')
                    content = file.read()
                    file.close()
                    mixer.init()
mixer.music.load(r"C:\Users\Mukund\PycharmProjects\Project_Jar
vis\Downloads\notification.mp3")
                    mixer.music.play()
                    notification.notify(
                        title="My Schedule:",
                        message=content,
                        timeout=15
                    )
                # cricket score function
                elif "score" in query:
                    url = "https://www.cricbuzz.com/"
                    resp = requests.get(url)
                    soup = BeautifulSoup(resp.content,
"html.parser")
                    team1 = soup.find_all(class_="cb-ovr-flo
cb-hmscg-tm-nm")[0].get_text()
```

```
                    team2 = soup.find_all(class_="cb-ovr-flo
cb-hmscg-tm-nm")[1].get_text()
                    team1_score = soup.find_all(class_="cb-
ovr-flo")[8].get_text()
                    team2_score = soup.find_all(class_="cb-
ovr-flo")[10].get_text()
                    # live match
                    print("Live Match")
                    speak("Live Match")
                    a = print(f"{team1}: {team1_score}")
                    b = print(f"{team2}: {team2_score}")
                    speak(f"Current match is going on between
{team1} and {team2}")
                    mixer.init()
mixer.music.load(r"C:\Users\Mukund\PycharmProjects\Project_Jar
vis\Downloads\notification.mp3")
                    mixer.music.play()
                    notification.notify(
                        title="CURRENT SCORE:",
                        message=f"{team1} : {team1_score}\n
{team2} : {team2_score}",
                        timeout=15,
                    )
                    print()
                elif "screenshot" in query:
                    im = pyautogui.screenshot()
im.save(r"C:\Users\Mukund\PycharmProjects\Project_Jarvis\ss.jp
g")
                elif "take a picture" in query:
                    pyautogui.press("super")
                    pyautogui.typewrite("camera")
```

```
        pyautogui.press("enter")
        pyautogui.sleep(2)
        speak("Smile Please, sir")
        pyautogui.press("enter")
    elif "game" in query:
        from game import game_play
        game_play()
    elif "joke" in query:
        get = pyjokes.get_joke()
        speak(get)
```

CHAPTER 7: INPUT/OUTPUT SCREENSHOT

Figure 7.1 Live GUI of JARVIS

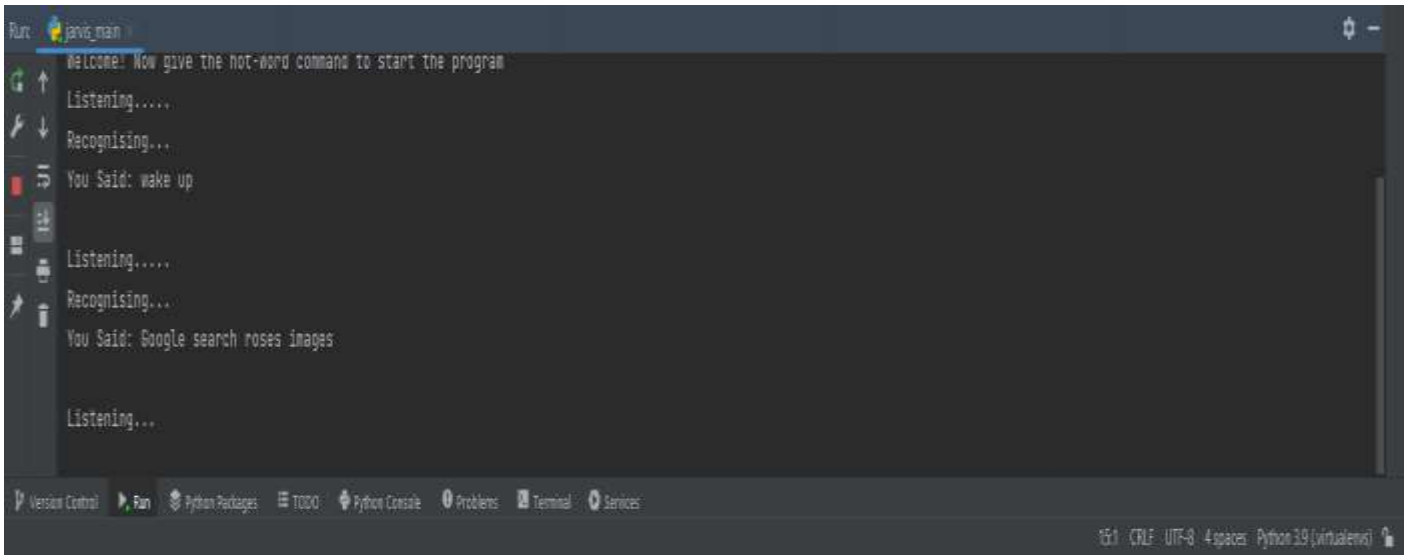

Figure 7.2 Input for Google search

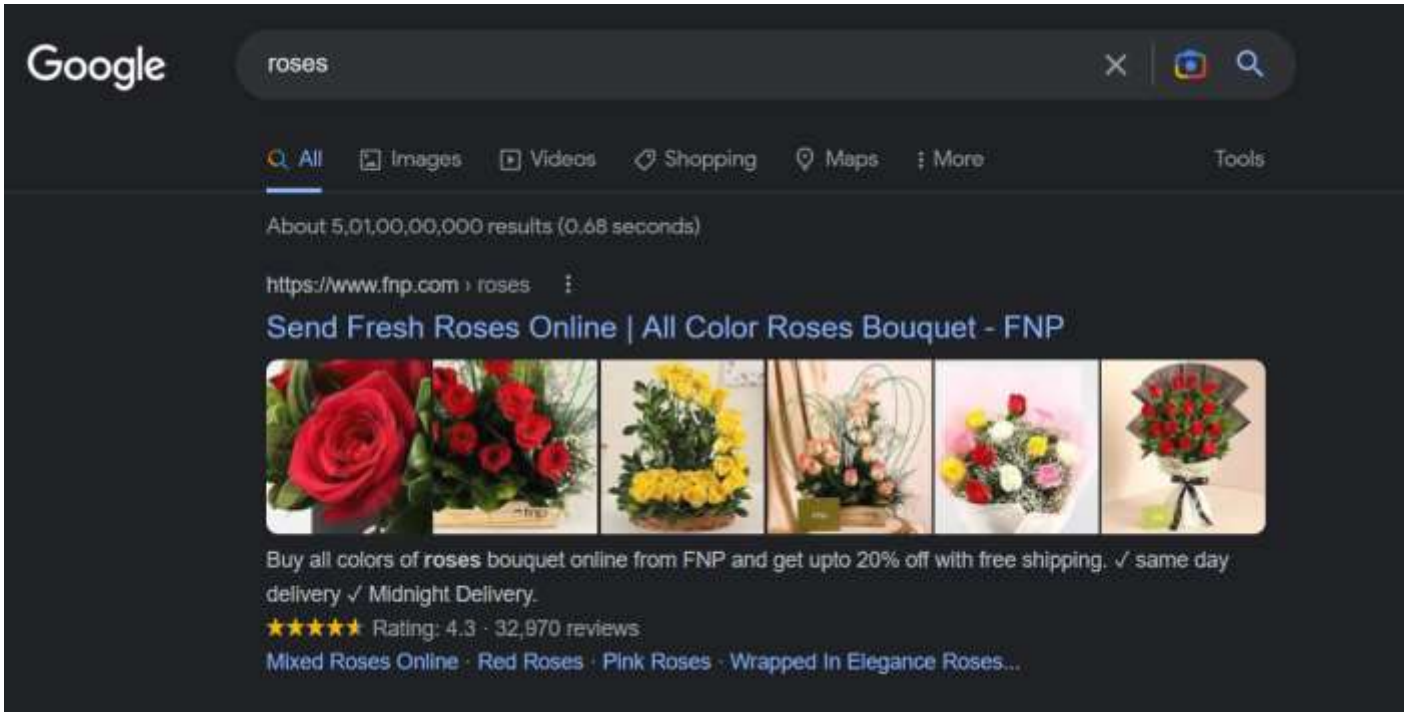

Figure 7.3 Output for Google search

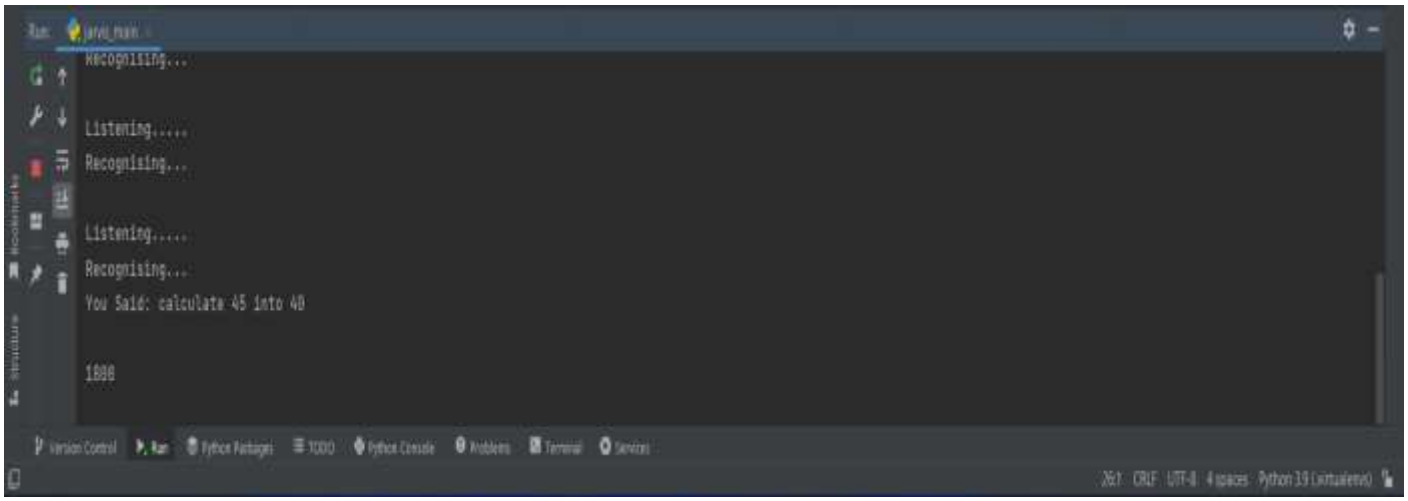

Figure 7.4 Input & Output for Calculator

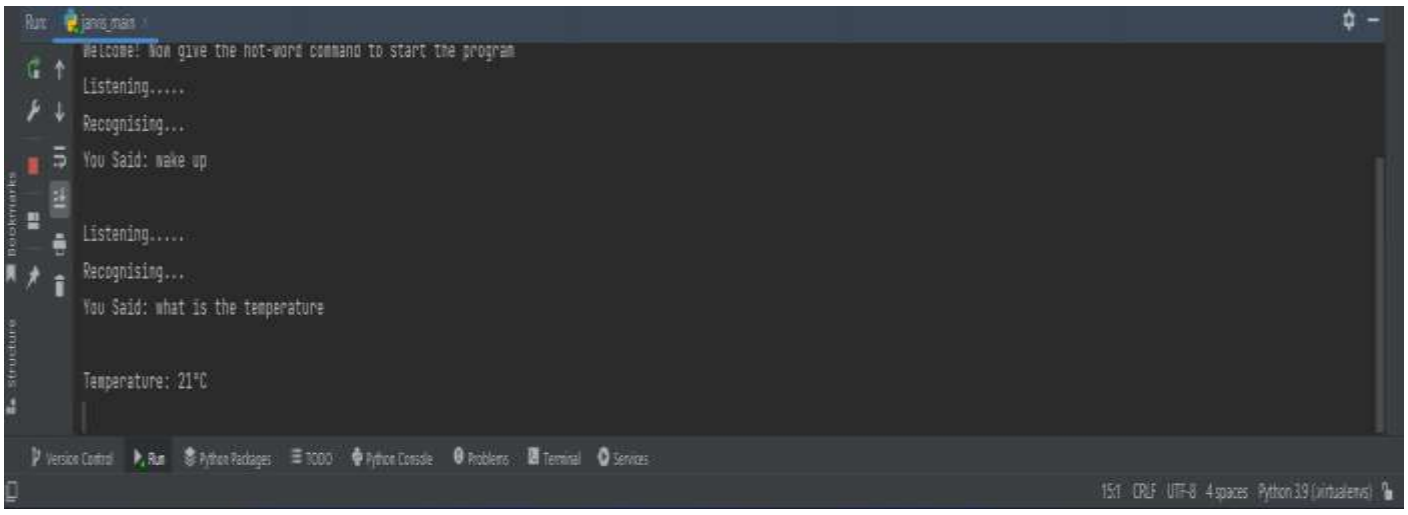

Figure 7.5 Input & Output for Temperature

Figure 7.6 Input for YouTube search

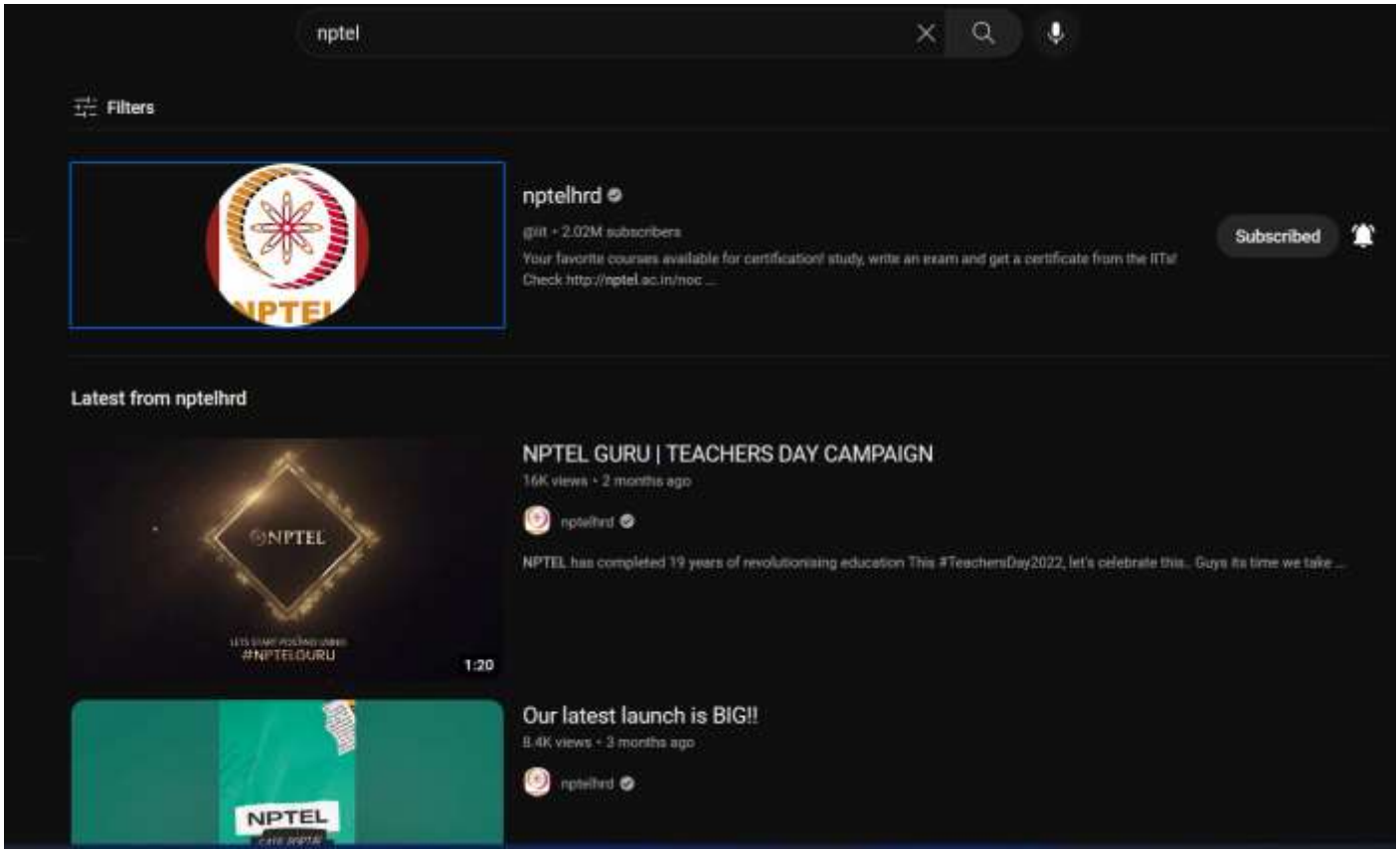

Figure 7.7 Output for YouTube search

Figure 7.8 Input to play music on YouTube

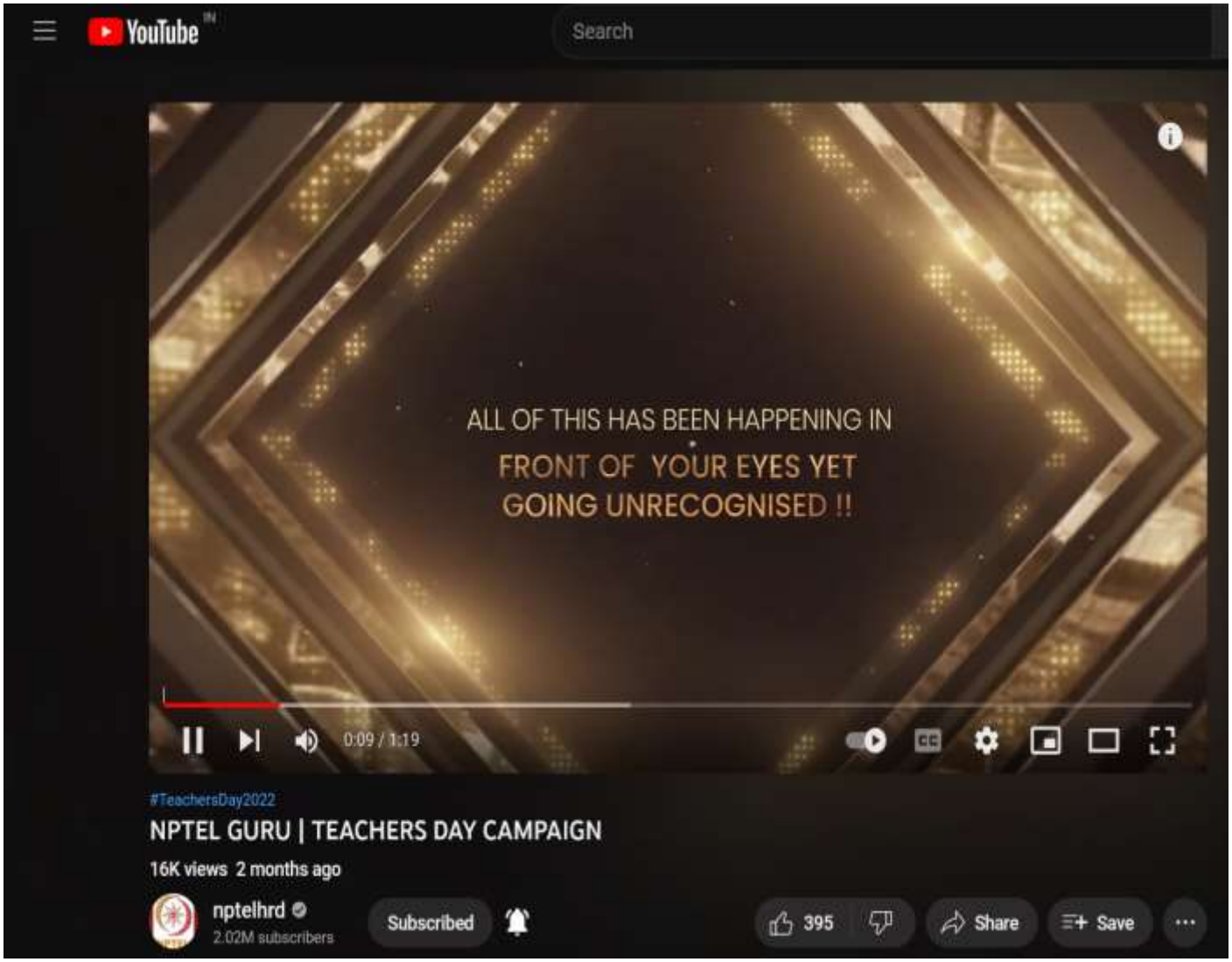

Figure 7.9 Output to play music on YouTube

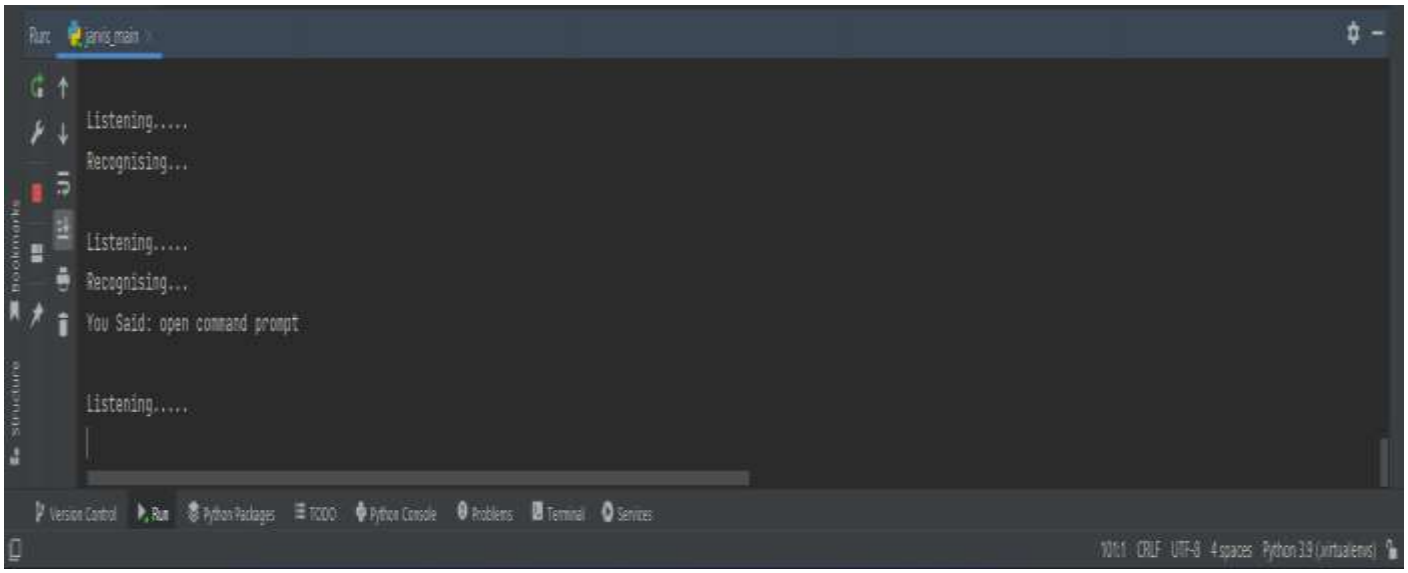

Figure 7.10 Input to open cmd

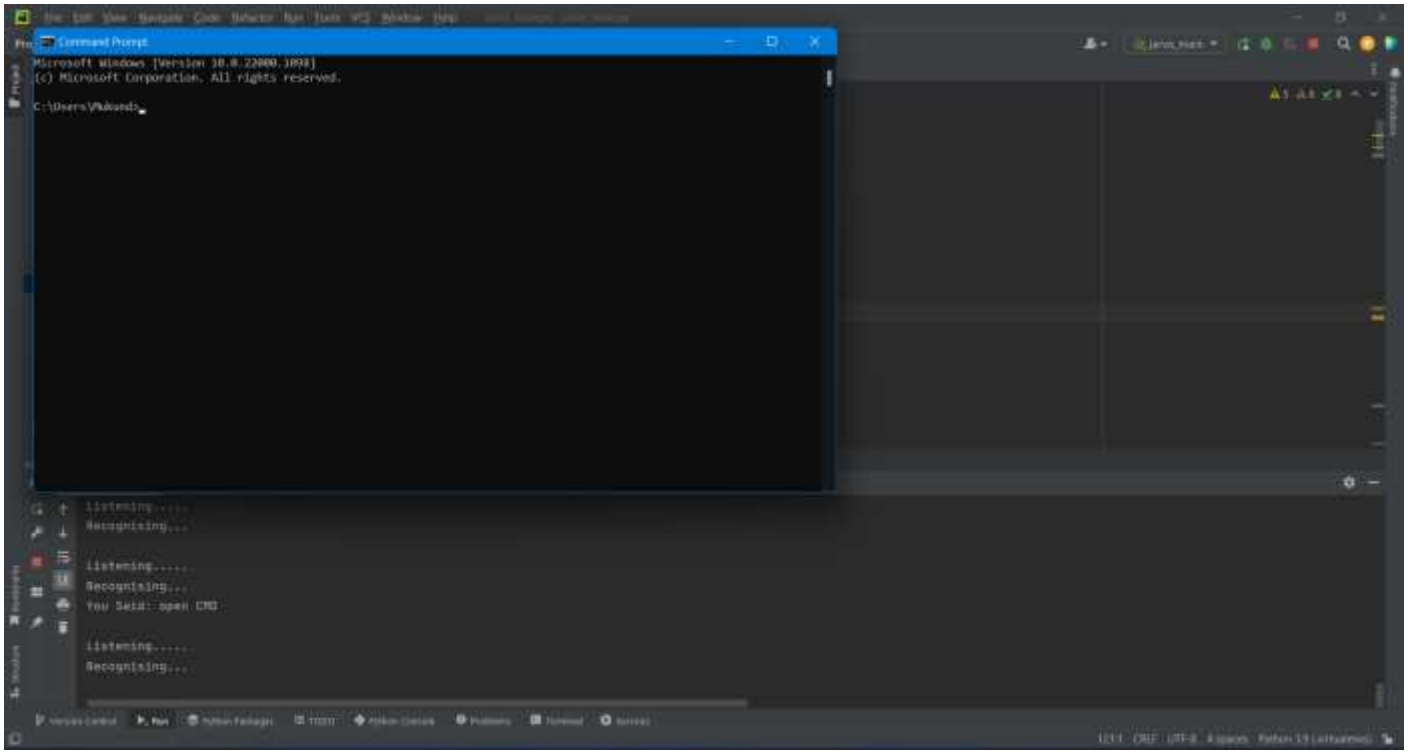

Figure 7.11 Output to open cmd

Figure 7.12 Input and output for Wikipedia search

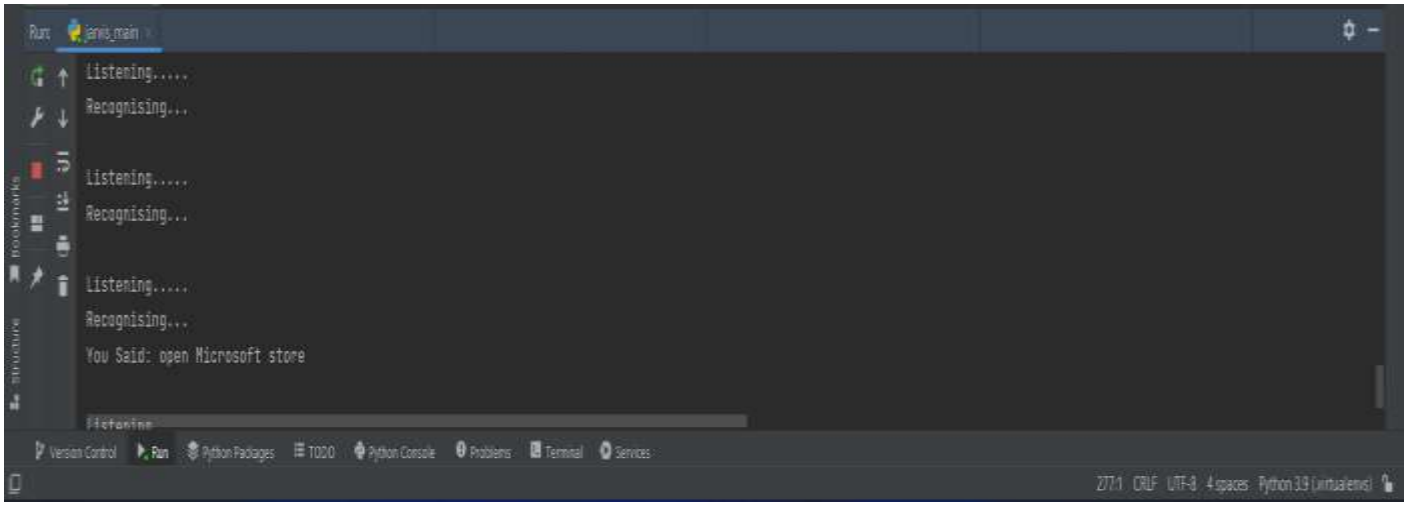

Figure 7.13 Input to open Microsoft Store

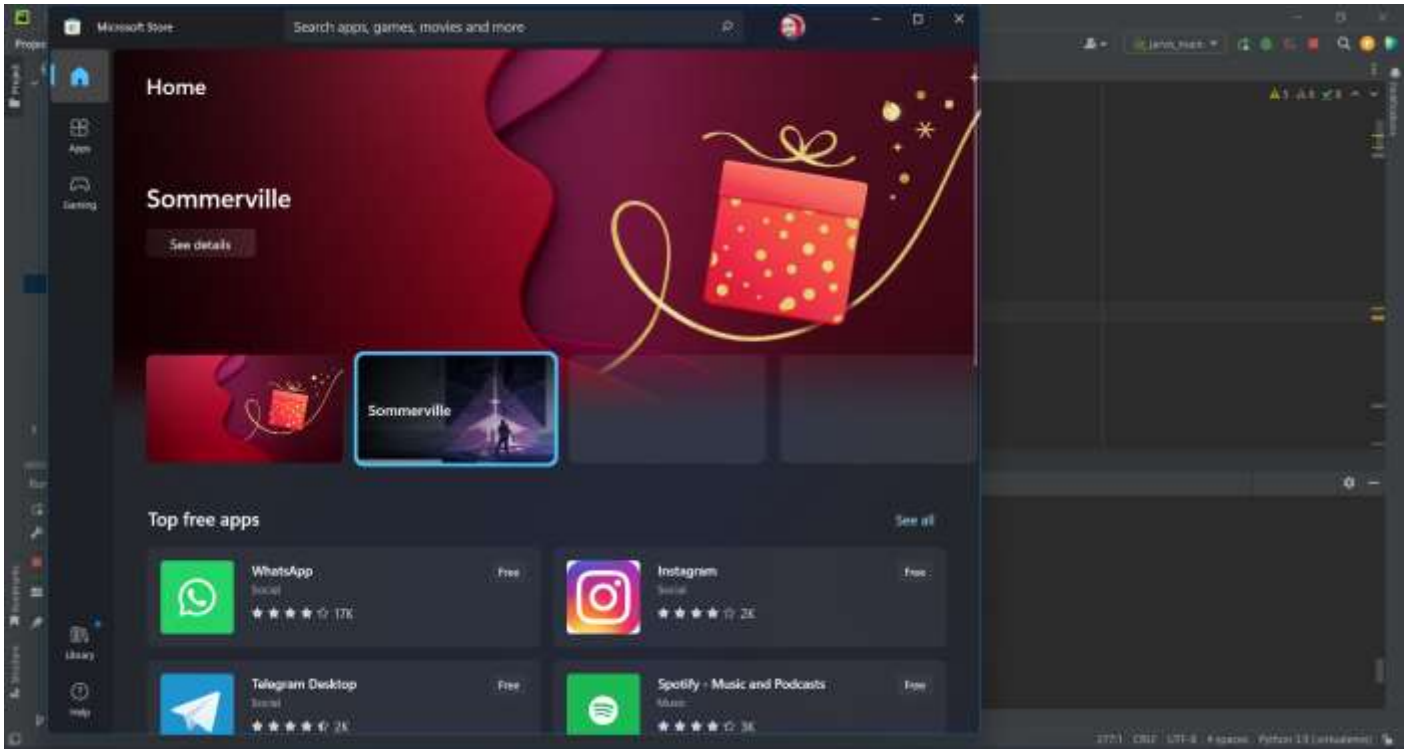

Figure 7.14 Output to open Microsoft Store

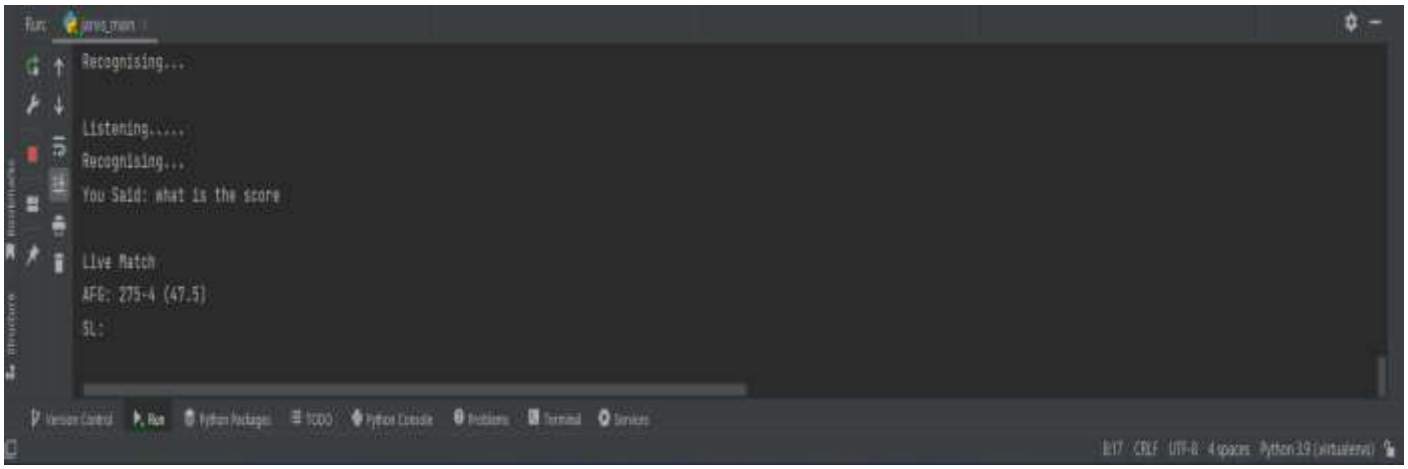

Figure 7.15 Input & Output for Cricket Scorecard

CHAPTER 8: SYSTEM TESTING

The system testing is done on fully integrated system to check whether the requirements are matching or not. The system testing for JARVIS desktop assistant focuses on the following four parameters:

8.1. FUNCTIONALITY

In this we check the functionality of the system whether the system performs the task which it was intended to do. To check the functionality each function was checked and run, if it can execute the required task correctly then the system passes in that functionality test. For example, to check whether JARVIS can search on Google or not, as we can see in the **Figure 8.1**, user said “Open Google", then Jarvis asked, "What should I search on Google?" then user said, "What is Python", Jarvis open Google and searched for the required input.

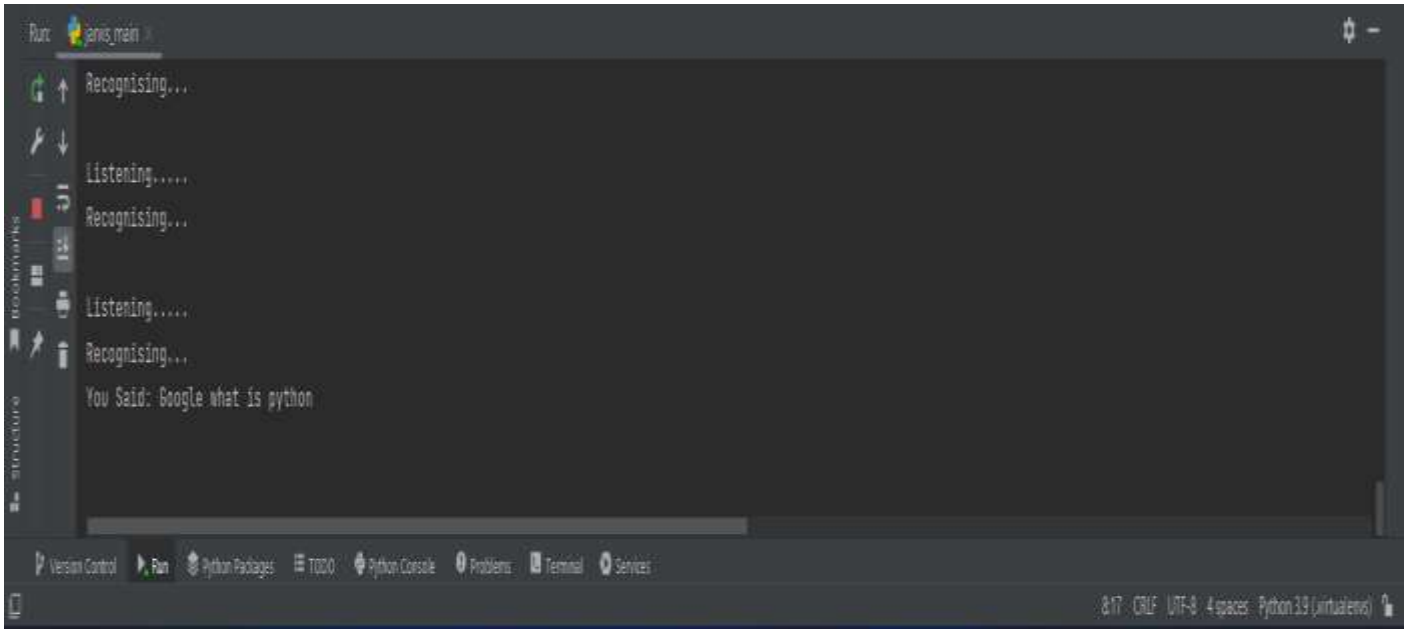

Figure 8.1 Input through voice commands

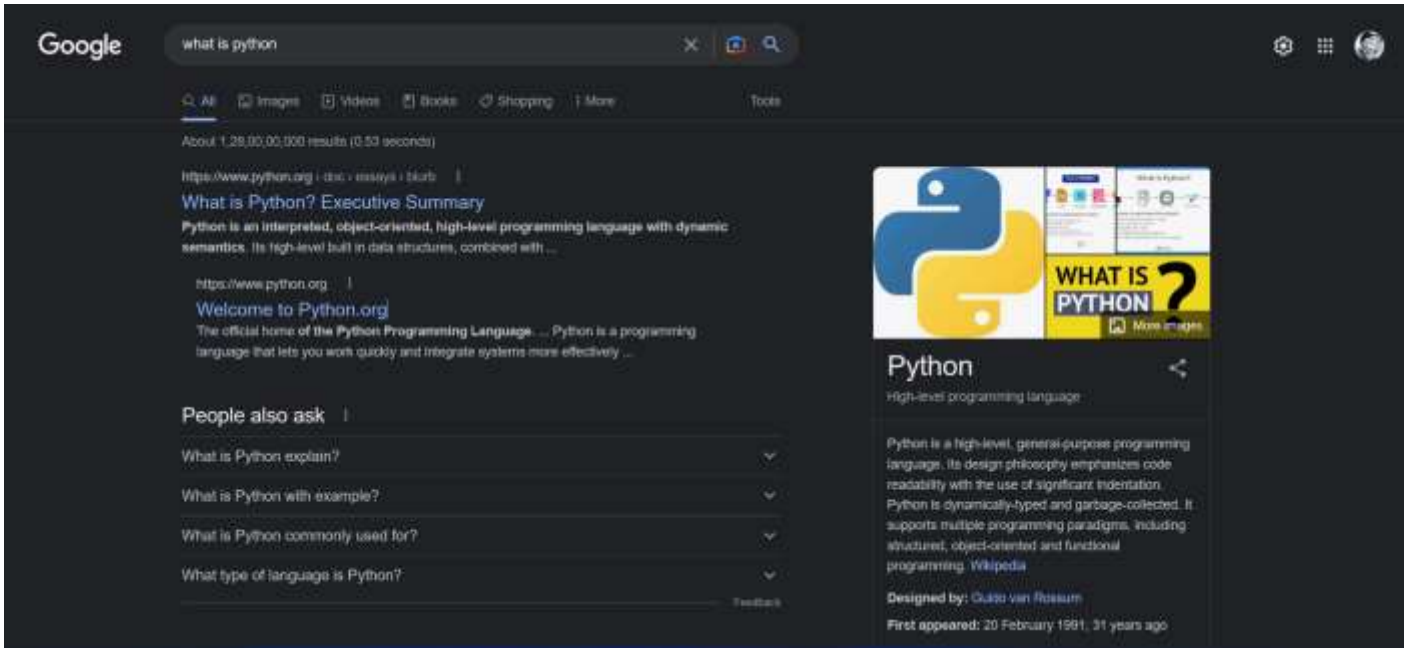

Figure 8.2 Output

8.2. USABILITY

Usability of a system is checked by measuring the easiness of the software and how user friendly it is for the user to use, how its responses to each query that is being asked by the user. It makes it easier to complete any task as it automatically does it by using the essential module or libraries of Python, in a conversational interaction way. Hence any user when instruct any task to it, they feel like giving task to a human assistant because of the **conversational interaction** for giving input and getting the desired output in the form of task done.

The desktop assistant is **reactive** which means it know human language very well and understand the context that is provided by the user and gives response in the same way, i.e., human understandable language, English. So, user finds its reaction in an informed and smart way.

The main application of it can be its **multitasking** ability. It can ask for continuous instruction one after other until the user "QUIT" it. It asks for the instruction and listen the

response that is given by user without needing any trigger phase and then only executes the task.

8.3. SECURITY

The security testing mainly focuses on vulnerabilities and risks. As JARVIS is a local desktop application, hence there is no risk of data breaching through remote access. The software is dedicated to a specific system so when the user logs in, it will be activated.

8.4. STABILITY

Stability of a system depends upon the output of the system, if the output is bounded and specific to the bounded input then the system is said to be stable. If the system works on all the poles of functionality, then it is stable.

CHAPTER 9: CONCLUSION

In this project **"JARVIS – AI AUTOMATED VIRTUAL ASSISTANT"** we have discussed about Jarvis voice assistance for windows using python. Jarvis voice assistance makes life easier to humans. As like Google assistance and Cortana we make Jarvis voice assistance to be available to all the windows version, we use Artificial intelligence technology for this project, Jarvis voice assistance be able to do all the tasks like other assistance including some special functions like restarting the devices, locking the device, sleeping the device for some time and shut down the device with our voice input. We can expect this Jarvis Voice Assistant to be permanent.

It works on voice command and gives responses to the user supported question/query being asked or the voice command spoken by the user like opening any task and performing any operations. It is greeting the user in a specific way then user feels liberal to interact with the virtual assistant the virtual assistant should also eliminate any unnecessary manual work of the user. The entire system works on the verbal voice input.

CHAPTER 10: LIMITATIONS & FUTURE WORK

10.1. LIMITATIONS

- Background voice can interfere.
- Misinterpretation because of accents and may cause inaccurate results.
- JARVIS cannot be called externally anytime like other traditional assistants like Google Assistant can be called just by saying, "Ok Google!"

10.2. SCOPE FOR FUTURE WORK

- Make JARVIS to learn more on its own and develop a new skill in it.
- JARVIS android app can also be developed.
- Make more Jarvis voice terminals. Voice commands can be encrypted to maintain security.

REFERENCES

[1] www.stackoverflow.com

[2] www.pythonprogramming.net

[3] www.freecodecamp.org

[4] www.codewithharry.com

[5] www.google.co.in

[6] Python Programming - Kiran Gurbani

[7] Learning Python - Mark Lutz

[8] CodeWithHarry – YouTube Channel

[9] Pratham Taneja – YouTube Channel

[10] freeCodeCamp.org – YouTube Channel

[11] Avi Upadhyay – YouTube Channel

[12] Kaushik Shresth – YouTube Channel

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

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