VIRTUAL ASSISTANT USING NLP

PROJECT REPORT

Submit by

DIVAKARAN M

KRISHNAKUMAR R

In partial fulfillment of the award of the degree

Of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

UNIVERSITY COLLEGE OF ENGINEERING TINDIVANAM
MELPAKKAM – 604001

ANNA UNIVERSITY: CHENNAI 600 025

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"VIRTUAL ASSISTANT USING NLP"** is the bonafide work of **"DIVAKARAN M(422420205011), KRISHNAKUMAR R(422420205023) ",** who carried out the project work under my supervision.

Staff Signature **HOD Signature**

Submitted for the University Examination held on _____

INTERNAL EXAMINAR **EXTERNALEXAMINER**

ACKNOWLEDGEMENT

We would like to thank our revered Dean, **Dr.P. Thamizhazhagan, M.E., Ph.D.,** for providing infrastructure facilities and wholehearted encouragement for completing our project successfully.

For mostly, we pay our grateful acknowledgement and extend our sincere gratitude to **Dr. S. Milton Ganesh, M.E., Ph.D.,** Assistant Professor and Head, Department of Information Technology, University College of Engineering Tindivanam, for extending the facilities of the department towards our project and for his unstinting support.

We express our sincere thanks to our supervisor r **Dr. S. Jacophine Susmi, M.E.,** Assistant Professor, Department of Information Technology, University College of Engineering Tindivanam, for guiding us for every phase of the Mini project. We appreciate his thoroughness, tolerance and ability to share his knowledge with us.

We thank her for being easily approachable and quite thoughtful. We owe his harnessing our potential and bringing out the best in us. Without her immense support through every step of the way, we could never have it to this extent. We thank all our teaching and non-teaching faculty members, our class advisor, and also our fellow friends helping us in providing valuable suggestions and timely ideas for the successful completion of the Mini project.

Last but not least, we extend our thanks to our family members, who have been a great source of inspiration and strength to us during the course of this Mini project work. We sincerely thank all of them.

ABSTRACT

This project report presents the development of a virtual assistant using Natural Language Processing (NLP) techniques and implemented in Python. The virtual assistant aims to provide intelligent and interactive conversational capabilities to assist users in various tasks. The report covers the design and implementation of the NLP model, including text preprocessing, entity recognition, intent classification, and dialogue management. Additionally, it discusses the integration of external APIs for enhanced functionality. This project utilizes various NLP libraries and tools available in Python, including NLTK, spaCy, and Hugging Face Transformers, to handle tasks such as text preprocessing, entity recognition, and sentiment analysis. The virtual assistant employs a combination of rule-based and machine learning approaches to enable effective interaction with users. The evaluation of the virtual assistant's performance and user satisfaction is also presented. Overall, this project demonstrates the effectiveness of NLP and Python in building a sophisticated virtual assistant for real-world applications. The primary objective of this virtual assistant is to provide users with a sophisticated and interactive conversational experience, enabling them to effortlessly accomplish various tasks and obtain information.

DIVAKARAN M

KRISHNAKUMAR R

TABLE OF CONTENT

CHAPTER 1
INTRODUCTION

1.1 Introduction:

In recent years, the advancements in NLP and machine learning have revolutionized the way humans interact with technology. The Virtual Assistant using Natural Language Processing (NLP) are cutting-edge solution that leverages the power of NLP techniques to build an intelligent and interactive virtual assistant. This project aims to create a user-friendly interface that can understand and respond to natural language commands and queries. By incorporating NLP algorithms, such as text classification, named entity recognition, and sentiment analysis, the virtual assistant can provide personalized assistance, answer questions, perform tasks, and even engage in meaningful conversations. Through the implementation of this project, users can experience the convenience of a virtual assistant that adapts to their needs, making their daily lives more efficient and enjoyable.

1.2 PROBLEM STATEMENT:

This project aims to address the challenge of developing an intelligent and responsive virtual assistant that can understand and interpret natural language commands and queries, providing personalized assistance and performing tasks effectively.

1.3 Scope:

This Virtual Assistant have wide range of applications. It can enhance personal productivity by assisting individuals in managing tasks, schedules, and reminders while providing personalized recommendations for improved efficiency. As a reliable source of information, the virtual assistant can retrieve data, answer queries, and offer real-time updates on various topics, making it useful for research purposes. Additionally, it can facilitate language translation, enabling users to communicate and understand different languages seamlessly. In customer support systems, virtual assistants can provide personalized and automated assistance, resolving common issues and guiding customers through troubleshooting processes. Healthcare applications involve the virtual assistant addressing patient queries, offering basic medical advice, and aiding in medication reminders. Lastly, the virtual assistant can analyze user preferences and historical data to deliver personalized recommendations for products, services, and entertainment options.

1.4. FEASIBILITY STUDY

Feasibility study can help you determine whether or not you should proceed with your project. It is essential to evaluate cost and benefit. It is essential to evaluate cost and benefit of the proposed system. Three aspects of feasibility study are,

1.4.1 Technical feasibility

It includes finding out technologies for the project, both hardware and software. For virtual assistant, user must have microphone to convey their message and a speaker to listen when system speaks. These are very cheap now a days and everyone generally possess them. Besides, system needs internet connection. While using, make sure you have a steady internet connection. It is also not an issue in this era where almost every home or office has Wi-Fi.

1.4.2 Economic feasibility

Here, we find the total cost and benefit of the proposed system over current system. For this project, the main cost is documentation cost. User also, would have to pay for microphone and speakers. Again, they are cheap and available. As far as maintenance is concerned, it won't cost too much.

1.4.3 Operational feasibility

It is the ease and simplicity of operation of proposed system. System does not require any special skill set for users to operate it. In fact, it is designed to be used by almost everyone. Kids who still don't know to write can read out problems for system and get answers.

1.4.4 Cultural feasibility:

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

CHAPTER 2

SYSTEM DESIGN

2.1 LITERATURE SURVEY

1.Nivedita Singh (2021) et al. proposed a voice assistant using python speech to text (STT) module and had performed some api calls and system calls which has led to developing a voice assistant using python which allows the user to run any type of command through voice without interaction of keyboard. This can also run-on hybrid platforms. Therefore, this paper lacks in some parts like the system calls that aren't much supported.

2.Abeed Sayyed (2021) et al. presented a paper on Desktop Assistant AI using python with IOT features and also used Artificial Intelligence (AI) features along with a SQLite DB with the use of Python. This Project has a Database connection and a query framework but lacks API call and System calls features.

3.Philipp Sprengholz (2021) et al. has proposed Ok Google: Using virtual assistants for data collection in psychological and behavioural research which is a survey mate that they have developed which is an extension of the Google Assistant that was used to check the reliability and validity of data collected by this test. Possible answers and synonyms are defined for every different type of questions so, it can be used to analyse the behaviour of an individual. As it is a psychological and behavioural research assistant.

4.Rahul Kumar (2020) et al. has proposed Power Efficient Smart Home with voice assistant by which we can say that a Voice Assistant is one of the important part of the Smart home which is becoming one of the major things in the current world as it can operate the Home Appliances just with voice which also increase the home security because of the smart locks but it requires a reliable internet connection.

5.Benedict D. C (2020) et al. proposed Consumer decisions with artificially intelligent voice assistants that will have stronger psychological reactions to the system's look on human like behaviours. The assistant has an IoT (Internet of Things) features. It can also order stuffs which the user want but there are some cons in this paper. Voice assistant relies on the speaker's ability to represent the decision alternatives to catch up in voice dialogues and another main disadvantage is that, it lacks system calls.

2.2 SYSTEM ARCHITECTURE

This Architecture discusses the key components involved in building a virtual assistant. The user interface serves as the platform for user interaction, while the speech-to-text component converts voice input into text. Text preprocessing techniques, such as tokenization and stemming, are applied to clean and prepare user input. The natural language understanding component extracts meaning from user queries using techniques like intent classification and entity recognition. Dialog management handles conversation flow and context. Natural language generation generates human-like responses based on the assistant's understanding. Access to knowledge bases or external data sources is crucial for providing accurate information. Machine learning models, trained using libraries like TensorFlow or PyTorch, can be employed for tasks like sentiment analysis or intent classification. Integration with external services, such as weather or news APIs, may be necessary for additional functionality.

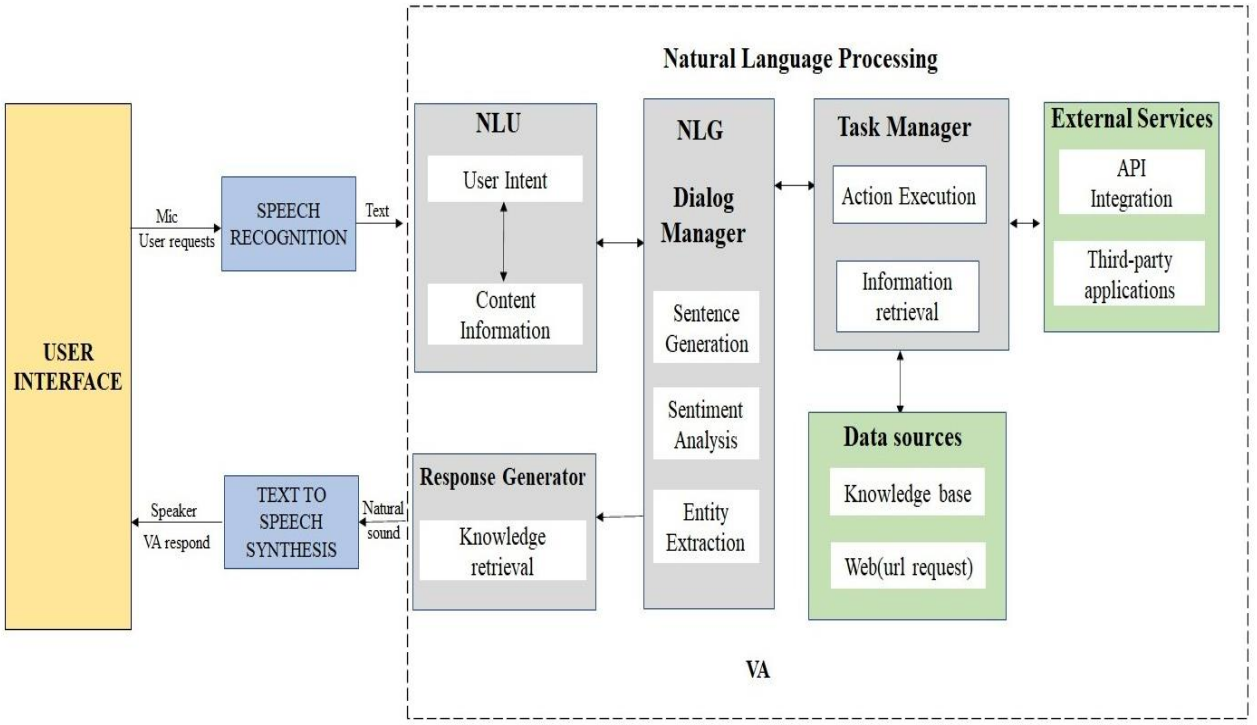

Figure 2.3 System Architecture

2.3 DATA FLOW DIAGRAM

This Data Flow diagram describes the functioning of a voice assistant model. User commands are captured through a microphone and passed through speech recognition to convert spoken language into a machine-readable format. Natural language processing (NLP) is then applied to analyze the input and understand the interaction between computers and human language. The VA determines whether the input is a question or an action. For actions, the voice assistant performs the requested action and provides acknowledgment through a synthesized voice. For questions, the assistant searches a dialog box or knowledge base and responds using a synthesized voice. The voice assistant utilizes the Google Text-to-Speech API to comprehend user input and generate appropriate responses based on predefined conditions.

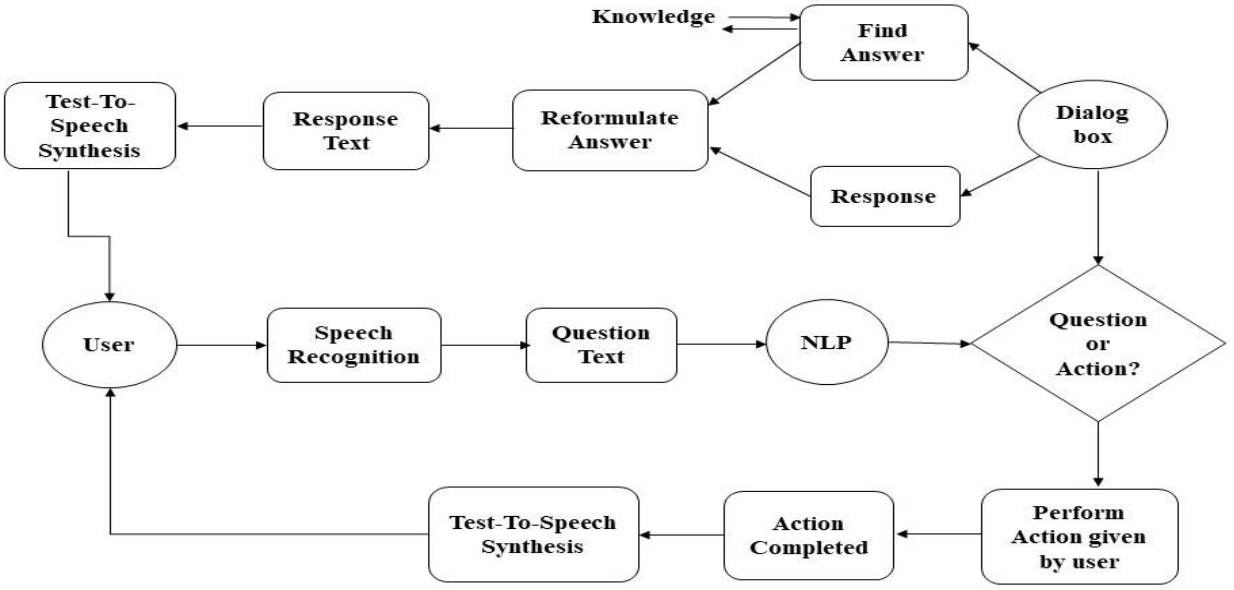

Figure 2.4 Data flow diagram

2.4 UML DIAGRAM

UML diagrams, short for Unified Modeling Language diagrams, are graphical representations used in software development to visualize and communicate system designs. They provide a standardized way to depict different aspects of a system, such as its structure, behavior, and interactions. UML diagrams include various types like class diagrams, use case diagrams, sequence diagrams, and more, each serving different purposes to aid in understanding and documenting software systems. Overall, UML diagrams help developers and stakeholders to analyze, design, and communicate about complex systems in a clear and visual manner.

2.4.1 Use case diagram

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

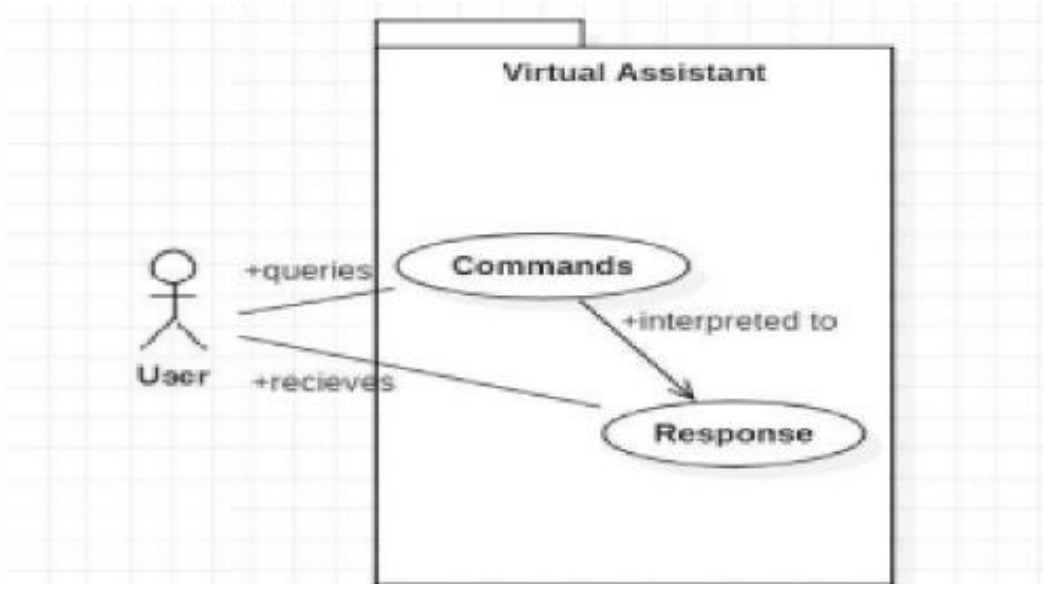

Figure 2.6 Use Case Diagram

2.4.2 Class diagram

Class diagram shows the class user has 2 attributes command that it sends in audio and the response it receives which is also audio. It performs function to listen the user command. Interpret it and then reply or sends back response accordingly. Question class has the command in string form as it is interpreted by interpret class. It sends it to general or about or search function based on its identification. The task class also has interpreted command in string format. It has various functions like reminder, note, mimic, research and reader.

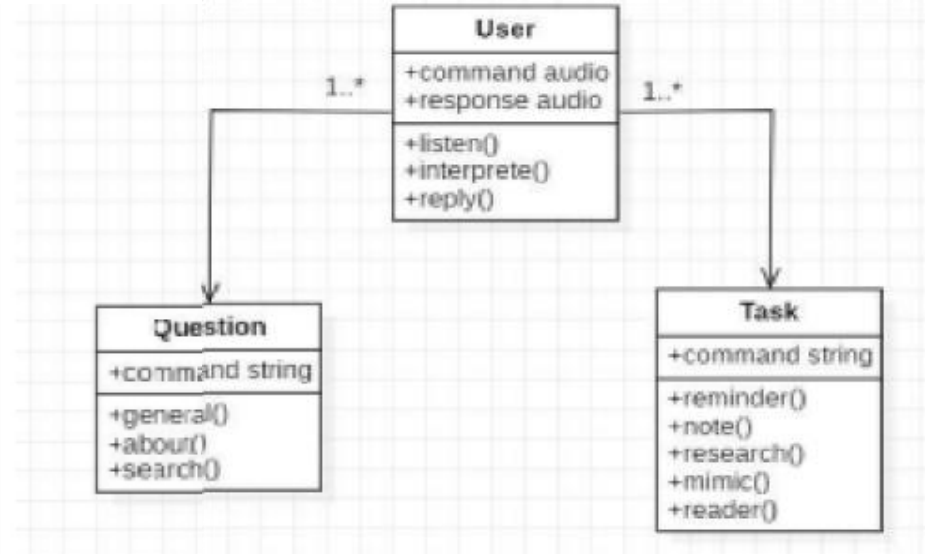

Figure 2.7 Class Diagram

2.4.3 Component diagram

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

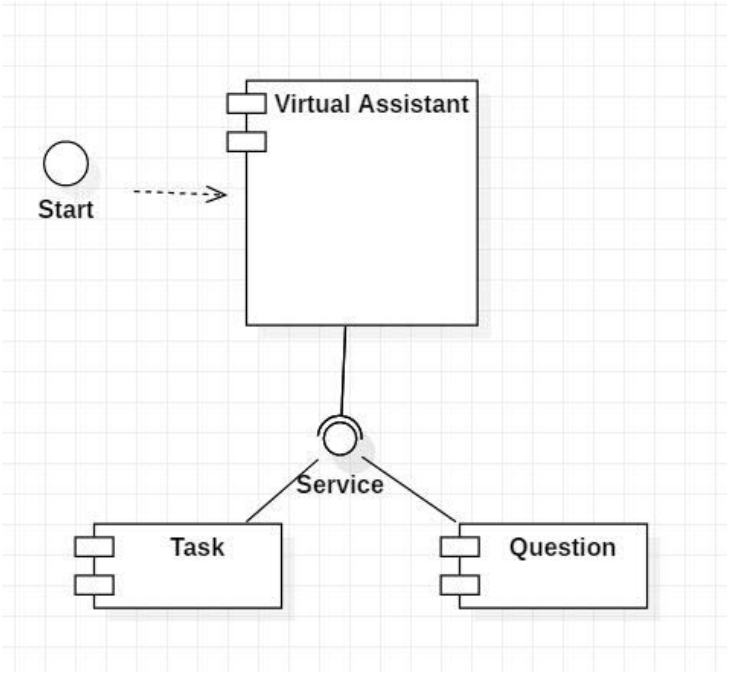

Figure 2.8 Component Diagram

2.4.4 Sequence diagram

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

The below 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.

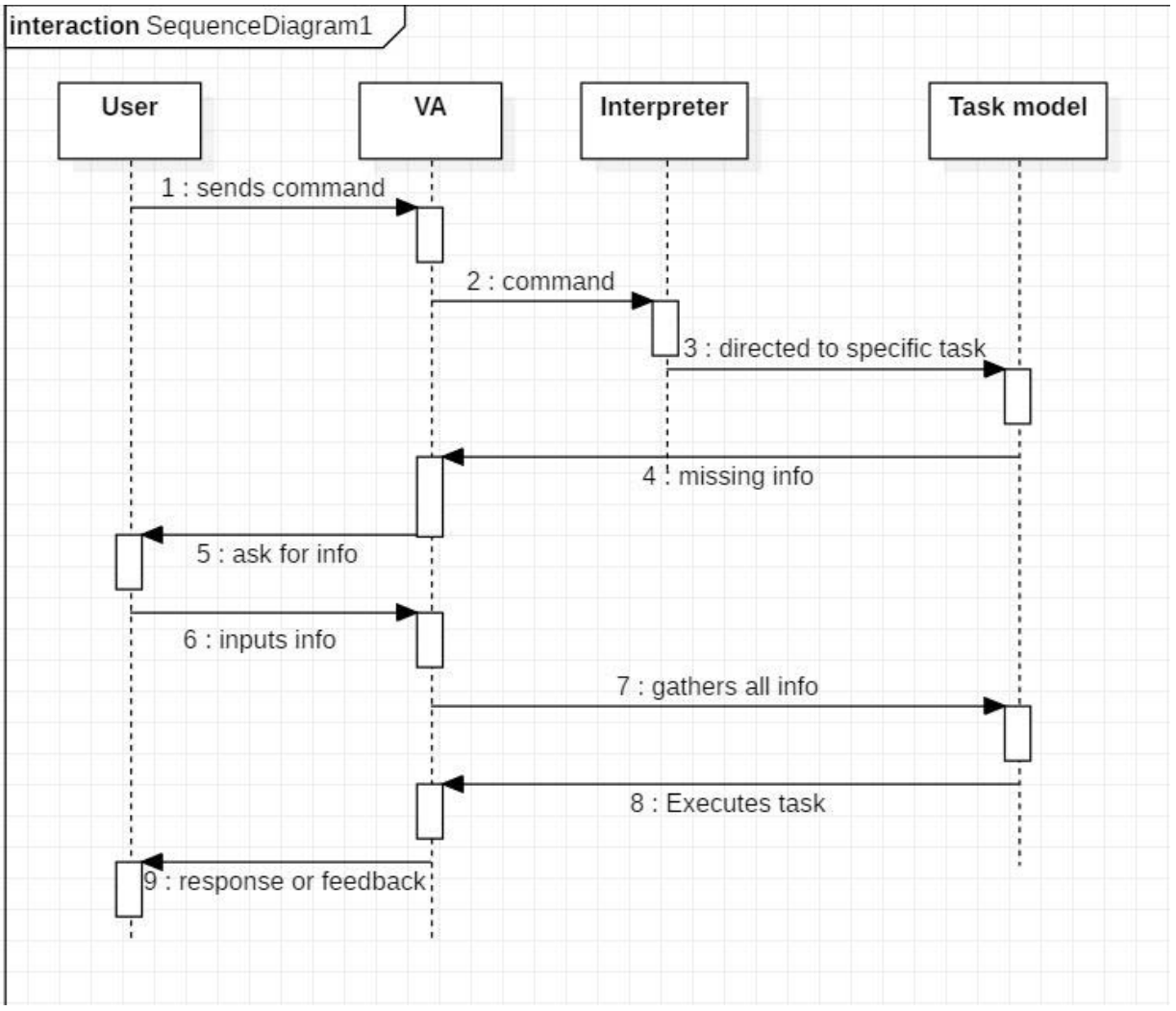

Figure 2.9 Sequence Diagram

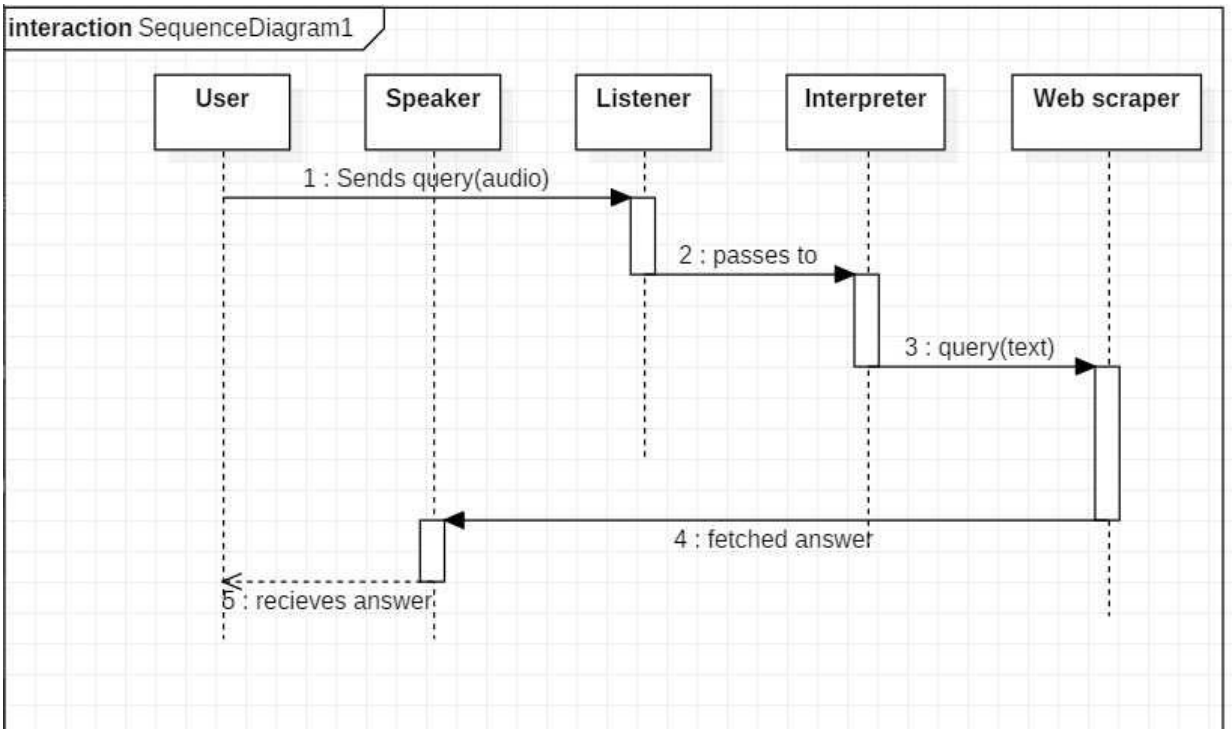

Figure 2.10 Sequence Diagram (Answering the user)

2.4.5 Activity diagram

An activity diagram for a virtual assistant using NLP would visually represent the flow of activities involved in processing user input, performing natural language processing tasks, and generating appropriate responses, facilitating efficient communication between the user and the virtual assistant.

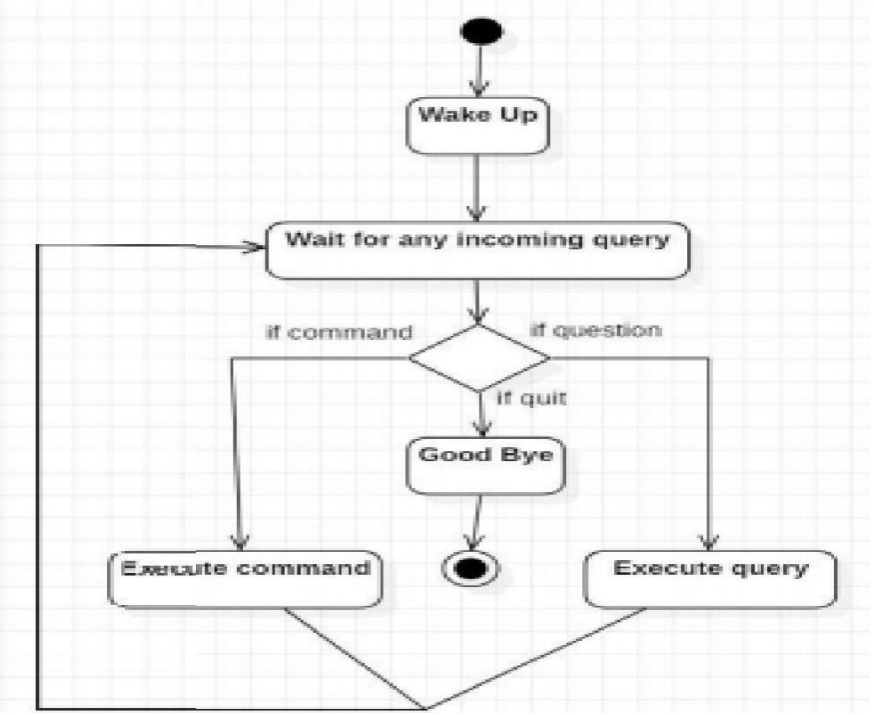

Figure 2.11 Activity Diagram

CHAPTER 3
MODULE

3.1 MODULE

There are various modules used in this project

1. Speech Recognition
2. Natural Language Understanding (NLU)
3. Dialogue Management
4. Natural Language Generation (NLG)

3.2 MODULE DESCRIPTION

3.2.1 Speech Recognition

This module is responsible for converting spoken language into written text. It utilizes techniques such as acoustic modeling and language modeling to accurately transcribe spoken words into textual representations.

3.2.2 Natural Language Understanding (NLU)

The NLU module focuses on interpreting and extracting meaning from user queries or commands. It analyzes the text to identify important components like intents (the user's intention or action) and entities (specific information or parameters). NLU employs techniques like tokenization, part-of-speech tagging, named entity recognition, and sentiment analysis.

3.2.3 Dialogue Management

This module handles the conversation flow and maintains context and state information. It determines the appropriate responses based on the user's input, the current dialogue history, and the system's capabilities. Dialogue management can involve techniques such as rule-based systems, finite-state machines, or more advanced approaches like reinforcement learning or deep learning-based models.

3.2.4 Natural Language Generation (NLG)

The NLG module generates human-like responses in natural language. It takes the structured or processed data from the dialogue management module and transforms it into coherent and contextually appropriate text. NLG can employ techniques such as template-based generation, rule-based systems, or more advanced methods like neural network-based models to generate natural language responses.

Some other Modules used;

3.2.5 Intent Classification

This module focuses on classifying the user's intent or purpose behind a query or command. It helps in understanding what action the user wants the virtual assistant to perform. Intent classification can be achieved using machine learning algorithms like support vector machines, naive Bayes, or deep learning models like recurrent neural networks (RNNs) or transformers.

3.2.6 Entity Recognition

This module identifies and extracts specific information or parameters (entities) from the user's input. For example, in a weather-related query, entity recognition would identify and extract the location, date, and time mentioned by the user. Entity recognition techniques include rule-based matching, statistical models, or more advanced approaches like named entity recognition using conditional random fields (CRFs) or deep learning models.

3.2.7 Knowledge Base/Database Integration

This module enables the virtual assistant to access and retrieve information from a knowledge base or database. It allows the assistant to provide relevant and accurate information to the user based on their queries.

3.2.8 Speech Synthesis

Also known as Text-to-Speech (TTS), this module converts textual responses into spoken words. It enables the virtual assistant to provide voice-based output to the user.

CHAPTER 4

CODING AND SYSTEM TESTING

4.1 CODING

The coding of a virtual assistant involves implementing various components and functionalities to enable natural language processing (NLP) capabilities. It typically includes modules for speech recognition, text-to-speech conversion, intent recognition, entity extraction, and dialog management. The code integrates APIs and libraries such as speech recognition libraries, NLP frameworks, and external services like Wikipedia, weather APIs, and email services. It utilizes machine learning algorithms for training and improving the assistant's language understanding. The code handles user input, processes it using NLP techniques, retrieves relevant information from external sources, and generates appropriate responses. It combines programming concepts, data processing, and NLP techniques to create an interactive virtual assistant.

4.2 DEVELOPING METHODOLOGIES

Developing methodologies for a virtual assistant using NLP involves analyzing requirements, collecting relevant data, preprocessing it, selecting and training NLP models, implementing intent recognition and response generation, integrating external services, designing a user interface, testing, and deploying the assistant. The process includes understanding user needs, cleaning and preparing data, choosing appropriate NLP models, training them with the data, recognizing user intents, generating suitable responses, and integrating with external APIs. User interface design, testing, and maintenance are also crucial. Documentation is important for future improvements. Overall, these steps ensure the systematic development of an efficient virtual assistant powered by NLP.

4.3 SYSTEM TESTING

System testing for a virtual assistant using NLP involves validating the overall functionality, performance, and reliability of the system. It includes various tests to ensure the assistant's accuracy, responsiveness, and user experience. Key testing areas include intent recognition, entity extraction, response generation, natural language understanding, and integration with external services. Test cases are designed to cover different user scenarios, edge cases, and error handling. The tests assess the assistant's ability to understand and respond accurately, handle user queries effectively, and provide relevant

and context-aware information. System testing ensures that the virtual assistant meets the desired quality standards and delivers a seamless user experience. The testing are,

4.3.1 Unit testing

Unit testing focuses on testing individual components and functions to ensure their correctness and proper functionality. It involves testing specific modules such as intent recognition, entity extraction, natural language processing, and response generation in isolation. Unit tests are designed to verify that each component performs as expected, handles different inputs correctly, and produces the desired outputs. Test cases are created to cover various scenarios, edge cases, and potential errors. The goal of unit testing is to identify and fix any bugs or issues in the code at an early stage, ensuring the reliability and accuracy of the virtual assistant's core functionality.

4.3.2 Functional testing

Functional testing focuses on testing the system as a whole to ensure that it meets the specified functional requirements. It involves testing the end-to-end functionality of the virtual assistant, simulating user interactions and verifying that the system behaves as expected. Functional tests cover various aspects such as intent recognition, entity extraction, dialog flow, context handling, and response generation. Test cases are designed to validate the correct interpretation of user queries, accurate extraction of relevant information, appropriate generation of responses, and smooth handling of conversation flows. The goal of functional testing is to ensure that the virtual assistant provides the intended functionality and delivers a satisfactory user experience.

4.3.2.1 Performance testing

Performance testing involves evaluating its performance in terms of responsiveness, scalability, and resource utilization under different workloads. The objective is to identify potential performance bottlenecks, measure system response times, and assess its ability to handle concurrent user interactions. Performance tests include stress testing, where the system is subjected to high loads to determine its limits, and load testing, which measures the system's response under expected user loads. Through performance testing, the virtual assistant's efficiency, speed, and ability to handle peak usage scenarios are assessed, ensuring that it can consistently deliver optimal performance and meet user expectations.

4.3.2.2 Stress testing

Stress testing involves subjecting the system to high levels of load and stress beyond its normal capacity to assess its performance and stability under extreme conditions. It helps

identify the system's breaking points, evaluate its robustness, and determine if it can handle unexpected spikes in user interactions without crashing or degrading performance. The goal is to ensure the virtual assistant can handle heavy workloads and maintain functionality under stressful scenarios.

4.3.3 Integration testing

Integration testing in the context of a virtual assistant involves testing the interaction and collaboration between different modules, components, and external services. It ensures that all the integrated parts of the system work together seamlessly and produce the expected results. The focus is on verifying the communication, data flow, and functionality between various components, such as speech recognition, intent recognition, dialog management, and external APIs. Integration testing involves both positive and negative test scenarios, evaluating the integration points, handling of data transitions, and verifying the proper functioning of the integrated system. It helps identify and address any issues or inconsistencies that may arise during the interaction between different parts of the virtual assistant.

4.3.4 Validation testing

Validation testing focuses on verifying that the system meets the specified requirements and fulfills the user's needs. It involves evaluating the overall performance, accuracy, and effectiveness of the virtual assistant in understanding and responding to user queries and commands. The testing process includes assessing the correctness of the generated responses, ensuring that the virtual assistant provides relevant and meaningful information, and validating the system's ability to handle various user inputs and scenarios. Validation testing also involves conducting user acceptance testing (UAT) to gather feedback from real users and validate the virtual assistant's usability and user satisfaction. The goal is to ensure that the virtual assistant meets the intended purpose and delivers a satisfactory user experience.

4.3.5 Output testing

Output testing ensures that the virtual assistant's responses and outputs are accurate, relevant, and meet the specified requirements. It involves verifying if the generated outputs align with the expected outputs and evaluating the system's ability to handle various input scenarios. By conducting output testing, any discrepancies, errors, or inconsistencies in the virtual assistant's responses can be identified and addressed, ensuring reliable and high-quality outputs for users.

CHAPTER 5

CONCLUSION

The integration of Natural Language Processing (NLP) in virtual assistant technology has proven to be a game-changer. NLP enables virtual assistants to understand and respond to human language, enhancing their usability and effectiveness. Through advanced algorithms and machine learning, virtual assistants can accurately interpret user queries, extract relevant information, and provide meaningful responses. This technology has revolutionized the way we interact with digital systems, offering personalized and efficient assistance across various domains. The future potential of NLP-based virtual assistants is vast, with applications ranging from customer service to healthcare and education. As NLP continues to evolve, virtual assistants will become even more intelligent, versatile, and indispensable in our everyday lives.

5.1 APPENDIX

```
import requests

from functions.online_ops import find_my_ip, get_latest_news, get_random_advice, get_random_joke, get_trending_movies, get_weather_report, play_on_youtube, search_on_google, search_on_wikipedia, send_email, send_whatsapp_message

import pyttsx3

import speech_recognition as sr

from decouple import config

from datetime import datetime

from functions.os_ops import open_calculator, open_camera, open_cmd, open_notepad, open_discord

from random import choice

from utils import opening_text

from pprint import pprint
```

```
USERNAME = config('USER')

BOTNAME = config('BOTNAME')

engine = pyttsx3.init('sapi5')

# Set Rate

engine.setProperty('rate', 190)

# Set Volume

engine.setProperty('volume', 1.0)

# Set Voice (Female)

voices = engine.getProperty('voices')

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

# Text to Speech Conversion

def speak(text):

"""Used to speak whatever text is passed to it"""

engine.say(text)

engine.runAndWait()

# Greet the user

def greet_user():

"""Greets the user according to the time"""

hour = datetime.now().hour

if (hour >= 6) and (hour < 12):

speak(f"Good Morning {USERNAME}")

elif (hour >= 12) and (hour < 16):
```

```
speak(f"Good afternoon {USERNAME}")

elif (hour >= 16) and (hour < 19):

speak(f"Good Evening {USERNAME}")

speak(f"I am {BOTNAME}. How may I assist you?")

# Takes Input from User

def take_user_input():

"""Takes user input, recognizes it using Speech Recognition module and converts it into text"""

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')

if not 'exit' in query or 'stop' in query:

speak(choice(opening_text))

else:

hour = datetime.now().hour

if hour >= 21 and hour < 6:

speak("Good night sir, take care!")

else:
```

```
speak('Have a good day sir!')
exit()
except Exception:
speak('Sorry, I could not understand. Could you please say that again?')
query = 'None'
return query
if __name__ == '__main__':
greet_user()
while True:
query = take_user_input().lower()
if 'open notepad' in query:
open_notepad()
elif 'open discord' in query:
open_discord()
elif 'open command prompt' in query or 'open cmd' in query:
open_cmd()
elif 'open camera' in query:
open_camera()
elif 'open calculator' in query:
open_calculator()
elif 'ip address' in query:
ip_address = find_my_ip()
```

```
speak(f'Your IP Address is {ip_address}.\n For your convenience, I am printing it on the screen sir.')

print(f'Your IP Address is {ip_address}')

elif 'wikipedia' in query:

speak('What do you want to search on Wikipedia, sir?')

search_query = take_user_input().lower()

results = search_on_wikipedia(search_query)

speak(f"According to Wikipedia, {results}")

speak("For your convenience, I am printing it on the screen sir.")

print(results)

elif 'youtube' in query:

speak('What do you want to play on Youtube, sir?')

video = take_user_input().lower()

play_on_youtube(video)

elif 'search on google' in query:

speak('What do you want to search on Google, sir?')

query = take_user_input().lower()

search_on_google(query)

elif "send whatsapp message" in query:

speak(

On what number should I send the message sir? Please enter in the console: ')

number = input("Enter the number: ")

speak("What is the message sir?")
```

```
message = take_user_input().lower()

send_whatsapp_message(number, message)

speak("I've sent the message sir.")

elif "send an email" in query:

speak("On what email address do I send sir? Please enter in the console: ")

receiver_address = input("Enter email address: ")

speak("What should be the subject sir?")

subject = take_user_input().capitalize()

speak("What is the message sir?")

message = take_user_input().capitalize()

if send_email(receiver_address, subject, message):

speak("I've sent the email sir.")

else:

speak("Something went wrong while I was sending the mail. Please check the error logs sir.")

elif 'joke' in query:

speak(f"Hope you like this one sir")

joke = get_random_joke()

speak(joke)

speak("For your convenience, I am printing it on the screen sir.")

pprint(joke)

elif "advice" in query:

speak(f"Here's an advice for you, sir")
```

```
advice = get_random_advice()
speak(advice)
speak("For your convenience, I am printing it on the screen sir.")
pprint(advice)
elif "trending movies" in query:
speak(f"Some of the trending movies are: {get_trending_movies()}")
speak("For your convenience, I am printing it on the screen sir.")
print(*get_trending_movies(), sep='\n')
elif 'news' in query:
speak(f"I'm reading out the latest news headlines, sir")
speak(get_latest_news())
speak("For your convenience, I am printing it on the screen sir.")
print(*get_latest_news(), sep='\n')
elif 'weather' in query:
ip_address = find_my_ip()
city = requests.get(f"https://ipapi.co/{ip_address}/city/").text
speak(f"Getting weather report for your city {city}")
weather, temperature, feels_like = get_weather_report(city)
```

5.1.2 OUTPUT:

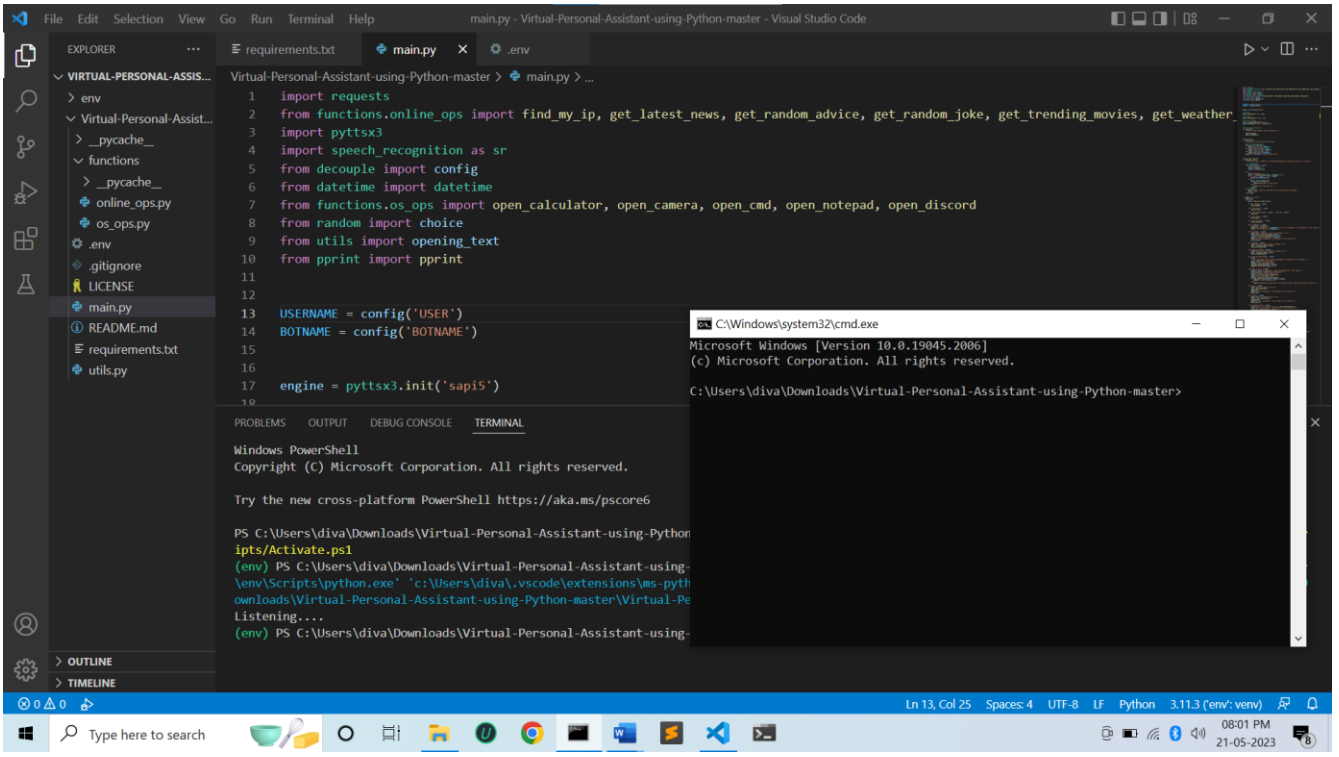

Figure 5.1 Open Cmd

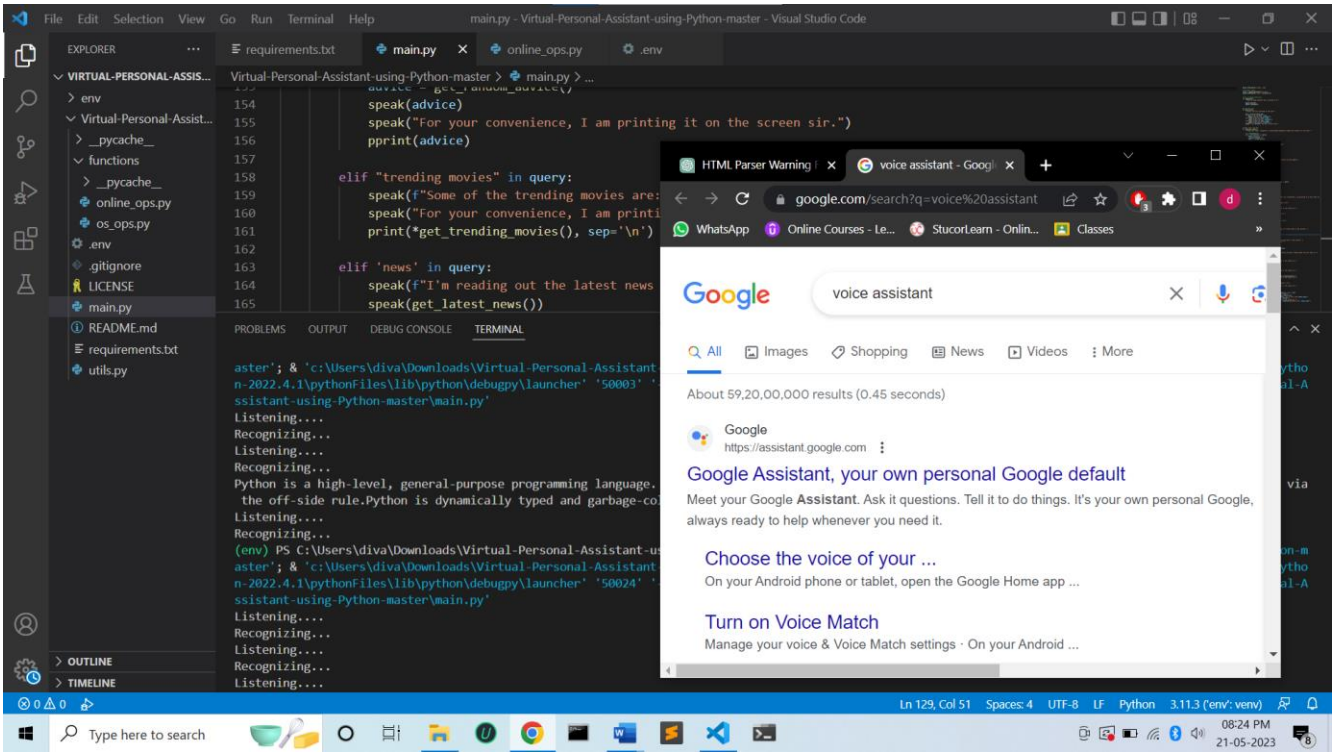

Figure 5.2 Searching Google

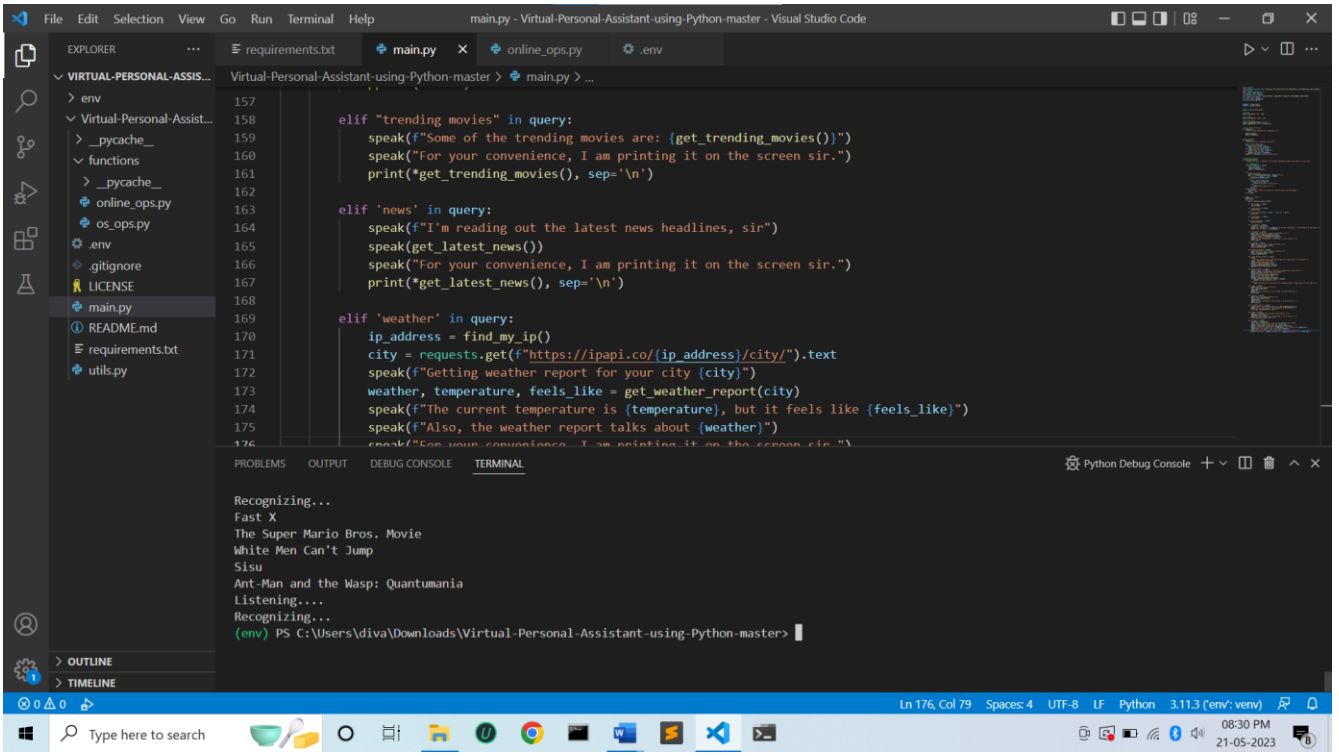

Figure 5.3 Getting Movies Updates

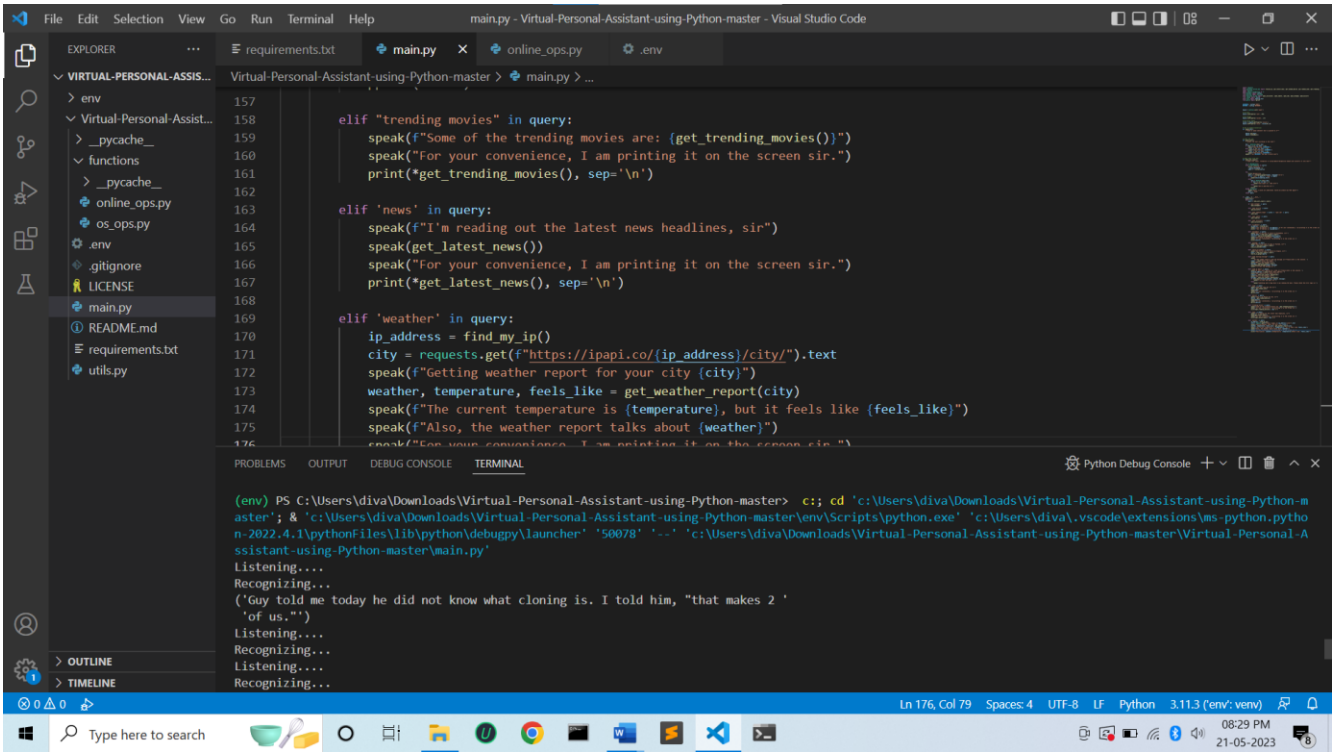

Figure 5.4 Getting Random Jokes

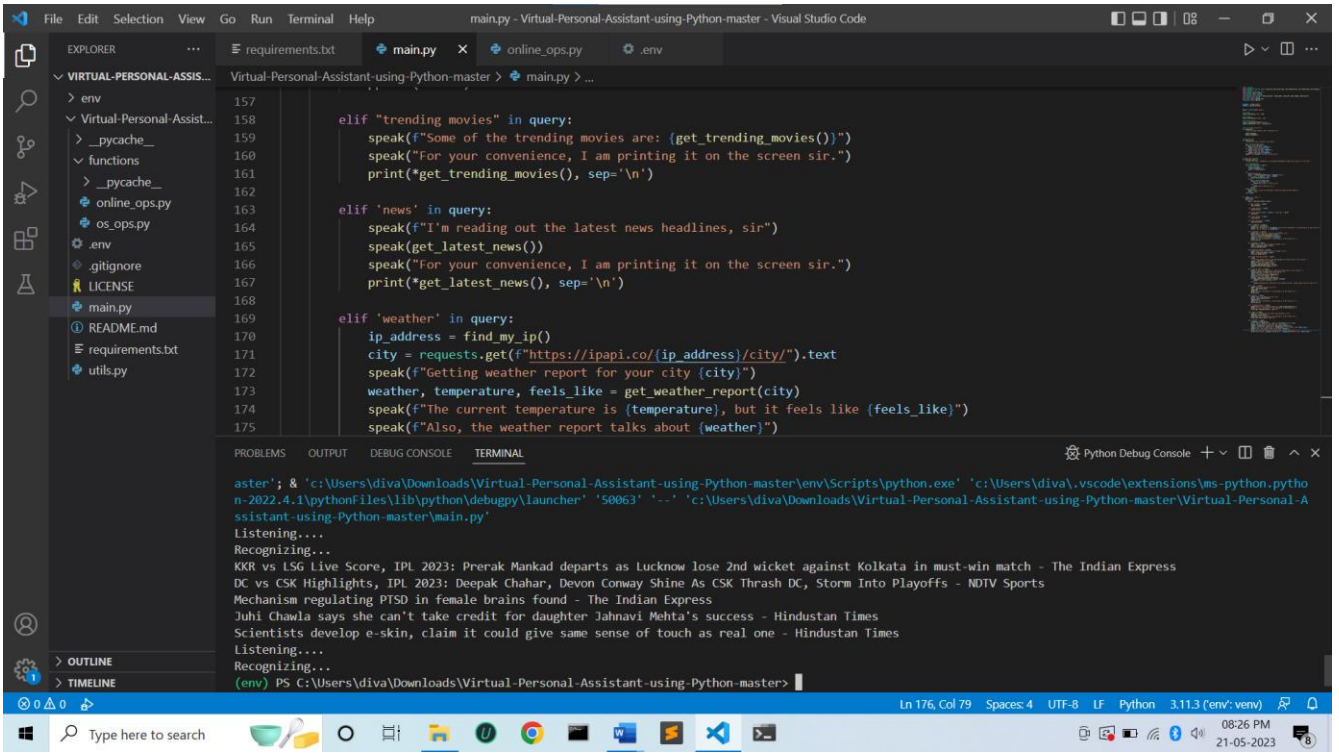

Figure 5.5 Asking News Headlines

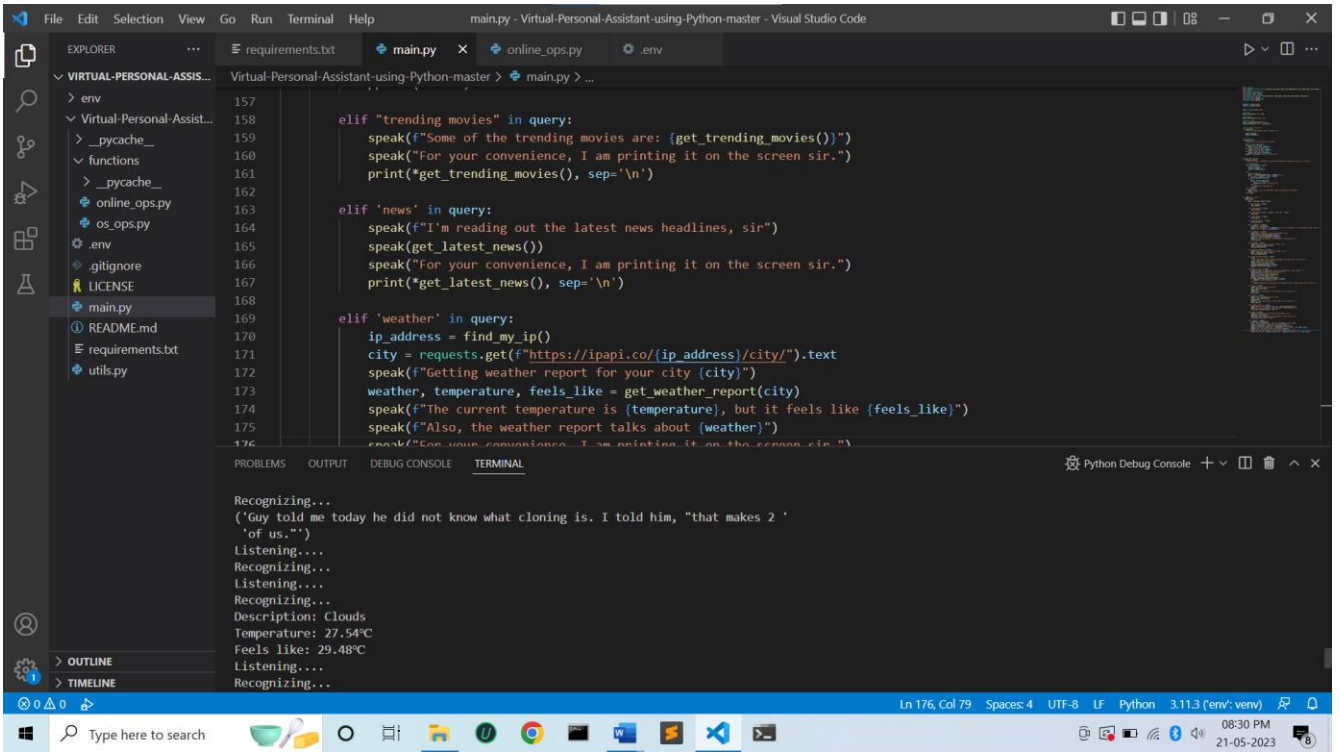

Figure 5.6 Asking Weather details

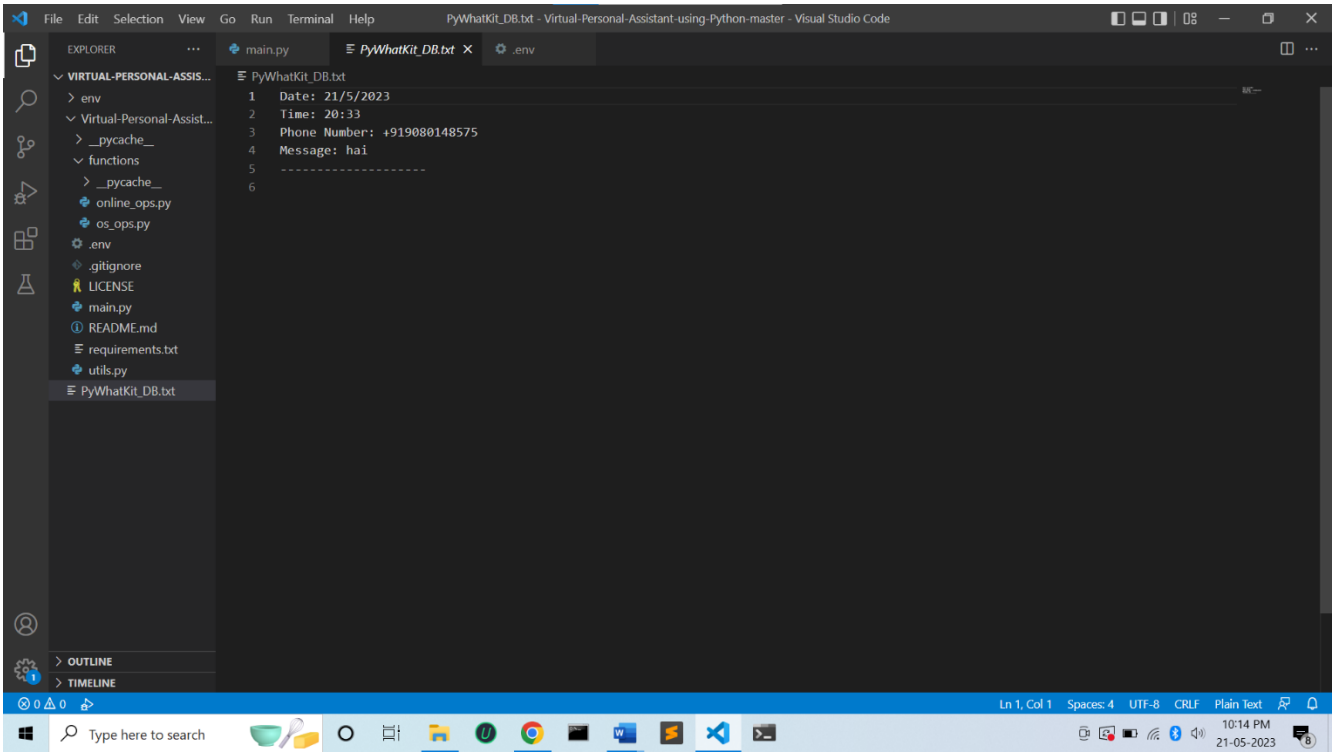

Figure 5.7 PyWhatkit image

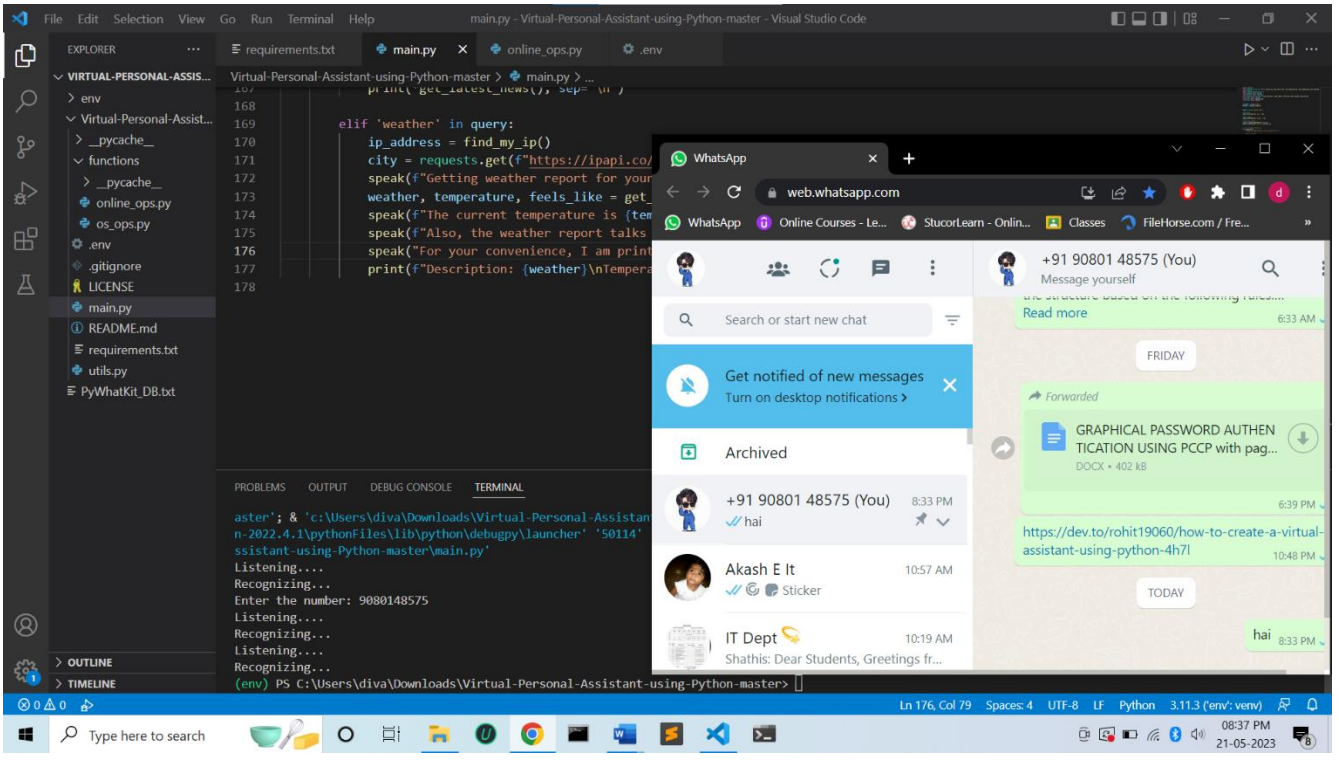

Figure 5.8 Send WhatsApp msg

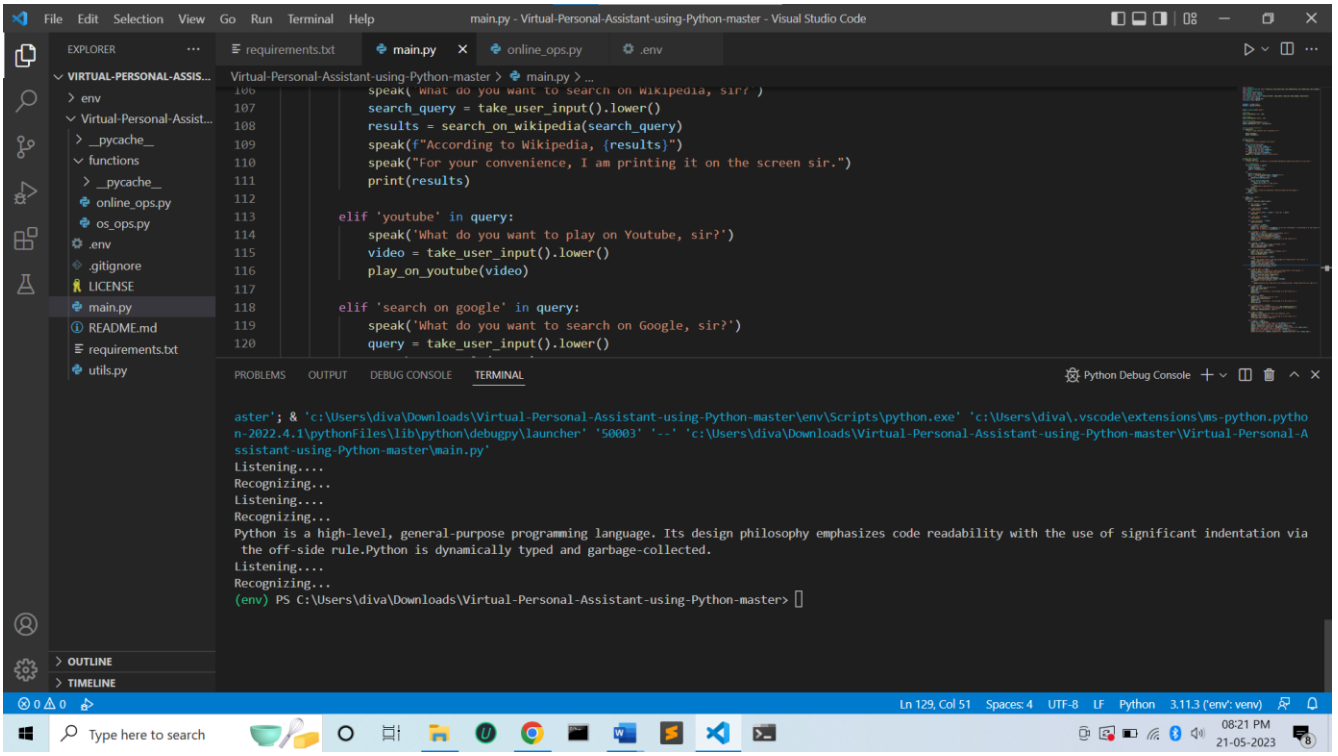

Figure 5.9 Answering from Wikipedia

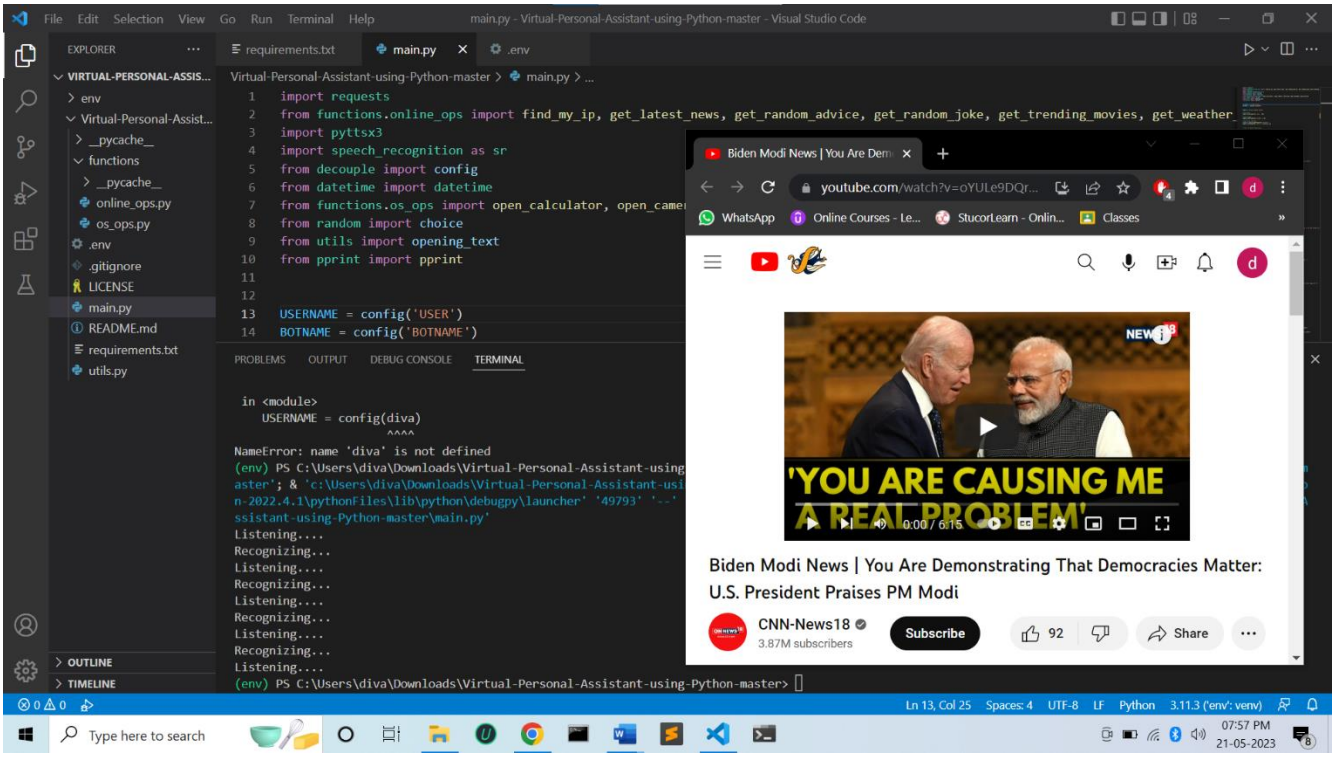

Figure 5.10 Open Youtube Video

REFERENCES

1. Title: "Siri: A Virtual Assistant for iPhone" Authors: Kittlaus, Dag; Cheyer, Adam; Winarsky, Tom Conference: International Conference on Intelligent User Interfaces (IUI) Year: 2012
2. Title: "Alexa: A Virtual Assistant for Amazon Echo" Authors: Ram, Ashwin; Kulkarni, Rakesh; Bergman, Orran; et al. Conference: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) Year: 2017
3. Title: "Google Assistant: A Virtual Assistant for Google Products" Authors: Li, Yan; Chen, Shuyang; Niu, Liang; et al. Conference: International Conference on Asian Language Processing (IALP) Year: 2017
4. Title: "Cortana: A Virtual Assistant for Windows" Authors: Gao, Qi; Zhang, Yongdong; Xu, Lei; et al. Conference: International Conference on Multimodal Interaction (ICMI) Year: 2015
5. Title: "IBM Watson: Building a Cognitive Virtual Assistant" Authors: Ferrucci, David A. Journal: AI Magazine Year: 2012
6. Title: "Mycroft: An Open Source Virtual Assistant" Authors: Wittenburg, Luke; Penkov, Dmitry; Chetty, Marshini; et al. Conference: International Conference on Intelligent User Interfaces Companion (IUI) Year: 2018
7. Title: "Nuance: Developing Conversational Virtual Assistants" Authors: Abella, Anthony; Catto, Jeff; Naudin, Gregory; et al. Conference: International Conference on Text, Speech, and Dialogue (TSD) Year: 2017
8. Title: "Viv: A Global Virtual Assistant for Everything" Authors: Cheyer, Adam; Orkin, Jeffrey Mark; Ibanez, Juan Conference: AAAI Conference on Artificial Intelligence (AAAI) Year: 2015
9. Title: "Bixby: A Virtual Assistant for Samsung" Authors: Lee, Yongkuk; Lee, Kyubyong; Ko, Yonghwan Conference: International Conference on Asian Language Processing (IALP) Year: 2018.