**HYDRA – THE AI ASSISTANT**

**A MINI PROJECT REPORT**

**Submitted by**

**ALLEN JOSHUA L 142220104008**

**HARISH B L 142220104039**

**HARSHAVARDHAN S V 142220104040**

*in a partial fulfilment for the award of the degree*

***of***

**BACHELOR OF ENGINEERING**

***in***

**COMPUTER SCIENCE AND ENGINEERING**

A picture containing diagram

Description automatically generated

**SRM VALLIAMMAI ENGINEERING COLLEGE**

**(AN AUTONOMOUS INSTITUTION)**

**SRM NAGAR, KATTANKULATHUR,**

**ANNA UNIVERSITY: CHENNAI 600 025**

**APRIL 2023**

**BONAFIDE CERTIFICATE**

Certified this project report **“HYDRA – THE AI ASSISTANT USING PYTHON”, is the bonafide work of “ALLEN JOSHUA L (142220104008), HARISH B L (142220104039), HARSHAVARDHAN S V (142220104040)” who carried out the work under my Supervision.**

|  |  |
| --- | --- |
| **SIGNATURE**  **Dr. B. VANATHI,M.E.,Ph.D.,**  **PROFESSOR & HEAD**  Department of CSE,  SRM Valliammai Engineering  College,  Kattankulathur-603 203. | **SIGNATURE**  **Dr. G. KUMARESAN, M.E., MBA., Ph.D.,**  **ASSISTANT PROFESSOR**  Department of CSE.  SRM Valliammai Engineering  College,  Kattankulathur-603 203. |

Submitted for the university examination held on\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ at SRM Valliammai Engineering College, Kattankulathur.

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

We sincerely express our gratitude in depth to our esteemed Founder Chairman & Chancellor **Dr. T. R. Paarivendhar, Thiru. Ravi Pachamoothoo** Chairman

SRM Group & Pro-Chancellor (Admin), **Mrs. R. Padmapriya Ravi** Vice Chairman, **Ms. R. Harini** Correspondent, **SRM VALLIAMMAI ENGINEERING COLLEGE** for the patronage on our welfare rooted in the academic year.

We express our sincere gratitude to our respected **Director Dr. B. Chidhambararajan, M.E., Ph.D.,** for his constant encouragement, which has been our motivation to strive towards excellence.

We thank our sincere **Principal Dr. M. Murugan, M.E., Ph.D.,** for his constant encouragement, which has been our motivation to strive towards excellence.

We extend our hand of thanks to our Head of the Department,   
**Dr. B. Vanathi, M.E., Ph.D.,** Professor for her unstinted support.

We are thankful to our project coordinator, **Dr., G. Kumaresan, M.E., MBA., Ph.D.,** for providing us with the essential facilities.

We also like to thank all Teaching and non-teaching staff members of our department, for their support during the course of the project. We finally thank our friends and family for their support during the course of the project.

**ABSTRACT**

Hydra AI Voice Assistant Mini Project focuses on developing a basic AI voice assistant using Python. The project aims to create a conversational assistant that understands and responds to user voice commands. It utilizes libraries such as Speech Recognition and pyttsx3 to enable speech-to-text and text-to-speech conversion. The project includes an example command for greeting the user based on the current time and provides a framework for adding additional functionalities like Date and Time, weather information, web searches using Wikipedia, making remainder list, takes screenshots, cracks non-funny jokes, open websites. By combining speech recognition, text-to-speech conversion, and command implementation, an interactive AI voice assistant is created. The project offers practical experience in developing voice-based AI assistants, exploring AI techniques, and integrating external libraries. It serves as a foundation for building more advanced voice assistants and encourages further customization and exploration in the field of artificial intelligence.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER**  **NO** | **TITLE** | **PAGE**  **NO** |
|  | **ABSTRACT** | iv |
|  | **LIST OF FIGURES** | vii |
| **1.** | **INTRODUCTION** | 2 |
|  | 1.1 PROLOGUE | 2 |
|  | 1.2 EXISTING SYSTEM | 3 |
|  | 1.3 PROPOSED SYSTEM | 3 |
|  | 1.4 LITERATURE SURVEY | 4 |
|  | 1.5 ALGORITHM STUDY | 8 |
| **2.** | **REQUIREMENT SPECIFICATION** | 9 |
|  | 2.1 HARDWARE SPECIFICATION | 10 |
|  | 2.2 SOFTWARE SPECIFICATION | 11 |
| **3.** | **SYSTEM DESIGN** | 13 |
|  | 3.1 SYSTEM ARCHITECTURE | 14 |
| **4.** | **IMPLEMENTATION** | 16 |
|  | 4.1 MODULES | 17 |
|  | 4.1.1 Divider | 17 |
|  | 4.1.2 Encrypter | 17 |
|  | 4.1.3 Decrypter | 18 |
|  | 4.1.4 Assembler | 19 |
| **5.** | **CONCLUSION AND FUTURE WORK** | 20 |
|  | 5.1 CONCLUSION | 21 |
|  | 5.2 FUTURE WORK | 21 |
| **6.** | **APPENDICES** | 22 |
|  | 6.1 APPENDIX-1 | 23 |
|  | 6.2 APPENDIX-2 | 36 |
| **7.** | **REFERENCES**  7.1REFERENCES | 41  42 |

**LIST OF FIGURES**

|  |  |  |  |
| --- | --- | --- | --- |
| **SI.NO** | **FIG.NO** | **TITLE** | **PAGE.NO** |
| 1. | 1.1 | ALGORITHM STUDY | 8 |
| 2. | 3.1 | ARCHITECTURE DIAGRAM | 14 |
| 5. | A1 | HOME SCREEN | 36 |
| 6. | A2 | CHOOSE FILE SCREEN | 37 |
| 7. | A3 | DOWNLOAD KEY SCREEN | 38 |
| 8. | A4 | RESTORE FILE SCREEN | 39 |
| 9. | A5 | ORIGINAL FILE DOWNLOAD SCREEN | 40 |

**CHAPTER -1**

**INTRODUCTION**

##### **1.1 PROLOGUE**

In an increasingly connected world, voice assistants have become an integral part of our daily lives. They provide us with information, help us manage tasks, and control our smart devices with just a simple voice command. The AI Voice Assistant Mini Project aims to explore the fascinating world of artificial intelligence and create a basic voice assistant using Python.

This project takes us on a journey to understand the underlying technologies behind voice assistants. We delve into speech recognition algorithms, text-to-speech conversion techniques, and command implementation. By building our own AI voice assistant, we gain hands-on experience in harnessing the power of AI to create intelligent and interactive applications.

Through this project, we not only learn how to capture and convert speech into text but also discover the art of transforming text into lifelike speech. We embark on the task of implementing various commands, empowering our assistant to respond intelligently to user requests and perform a range of tasks.

As we develop our AI voice assistant, we unlock the potential for customization and expansion. We can integrate additional APIs, enhance its capabilities with new commands, and create a truly personalized and powerful voice assistant.

##### **1.2 EXISTING SYSTEM**

##### Amazon Alexa: Alexa, developed by Amazon, is a widely used voice assistant integrated into devices like Echo speakers and other smart home devices. It supports a range of functionalities such as voice-controlled smart home automation, music playback, weather updates, and integration with third-party services through skills.

##### **1.3 PROPOSED SYSTEM**

##### The project aims to create a conversational assistant that understands and responds to user voice commands. It utilizes libraries such as Speech Recognition and pyttsx3 to enable speech-to-text and text-to-speech conversion. The project includes an example command for greeting the user based on the current time and provides a framework for adding additional functionalities like Date and Time, weather information, web searches using Wikipedia, making remainder list, takes screenshots, cracks non-funny jokes, open websites.

**1.5 ALGORITHM STUDY**

* Start
* Initialize the required libraries and modules for speech recognition, text-to-speech conversion, and command implementation.
* Set up the environment, including installing necessary libraries such as Speech Recognition, pyttsx3, pyaudio, wikipedia, and date and time.
* Define a function to capture user speech using the microphone and convert it to text.
* Define a function to convert text to speech using the pyttsx3 library.
* Implement various commands based on user inputs, such as greetings, Date and Time, weather information, web searches using Wikipedia, making remainder list, takes screenshots, cracks non-funny jokes, open websites.
* Create a continuous loop to listen for user commands.
* Within the loop, capture user speech and convert it to text.
* Process the user command by matching it with the implemented commands and execute the corresponding functionality.
* If the command is not recognized or understood, provide an appropriate response.
* Repeat steps 8-10 until the program is terminated or interrupted.
* End

**CHAPTER-2**

**REQUIREMENT**

**SPECIFICATION**

**2.1 HARDWARE SPECIFICATION**

The hardware requirements for the AI Voice Assistant Mini Project are relatively minimal. Here are the recommended specifications:

* Computer: Any modern computer or laptop should suffice. It should have enough processing power to run Python and the required libraries smoothly.
* Microphone: A functional microphone is essential for capturing user speech. It can be an external USB microphone, a built-in microphone on a laptop, or a headset with a microphone.
* Speakers or Headphones: To hear the assistant's responses, you will need speakers or headphones connected to your computer. Ensure they are in working condition.
* Internet Connection: An internet connection is required for accessing online services, such as speech recognition APIs or fetching information from the web.

**2.2 SOFTWARE SPECIFICATION**

The software requirements for the AI Voice Assistant include the following:

* Python: Ensure that you have Python installed on your system. The project was developed using Python, so having a compatible version of Python is necessary.
* Integrated Development Environment (IDE): While you can use any text editor or IDE of your choice, Visual Studio Code (VS Code) is a popular and highly recommended option. It provides a rich set of features, including code editing, debugging, and integration with various Python extensions.
* Python Libraries: The project utilizes several Python libraries, which can be installed using the pip package manager. Here are the key libraries required:
* Speech Recognition: Used for converting speech to text.
* pyttsx3: Used for converting text to speech.
* Pyaudio: Required for audio input and output.
* Wikipedia: Used for fetching information from Wikipedia.
* datetime: Used for retrieving the current date and time.
* You can install these libraries by running the following commands in your command prompt or terminal:
* pip install SpeechRecognition
* pip install pyttsx3
* pip install pyaudio
* pip install wikipedia

**CHAPTER-3**

**SYSTEM DESIGN**

**3.1 SYSTEM ARCHITECTURE**

In this diagrammatic representation, the User Interface consists of a microphone for capturing user speech input and speakers/headphones for outputting the assistant's responses. The speech input is sent to the Speech Recognition Module, which converts the audio input into recognized text. The recognized text is then passed to the Command Processing Module, where the user command is matched with implemented commands and corresponding functionality is executed. The Command Processing Module may utilize external APIs or libraries for specific commands, such as fetching information from Wikipedia. The response text from the Command Processing Module is then passed to the Text-to-Speech Module, which converts it into synthesized speech. The generated speech is outputted to the speakers/headphones for the user to hear. The loop continues, allowing the user to provide continuous speech input and receive responses from the assistant.

The system architecture for the AI Voice Assistant is as follows:

User Interface:

* Microphone: Captures user speech input.
* Speakers/Headphones: Outputs the assistant's responses in the form of speech.

Speech Recognition Module:

* Receives audio input from the microphone.
* Converts the audio into text using the SpeechRecognition library.
* Provides the recognized text as input to the Command Processing Module.

Command Processing Module:

* Receives the recognized text from the Speech Recognition Module.
* Matches the user command with implemented commands.
* Executes the corresponding functionality based on the recognized command.
* Utilizes external APIs or libraries for specific commands (e.g., Wikipedia API for fetching information).

Text-to-Speech Module:

* Receives the assistant's response or information to be conveyed.
* Converts the text into speech using the pyttsx3 library.
* Outputs the speech to the speakers or headphones for the user to hear.

Continuous Loop:

* Listens for user commands continuously.
* Triggers the Speech Recognition Module to capture user speech input.
* Passes the recognized text to the Command Processing Module.
* Receives the assistant's response from the Command Processing Module.
* Sends the response to the Text-to-Speech Module for conversion into speech.
* Outputs the speech response to the speakers or headphones.

**CHAPTER-4**

**IMPLEMENTATION**

**4.1 MODULES**

**4.1.1 USER INTERFACE MODULE**

The User Interface phase represents the hardware components used for interaction with the AI voice assistant. This includes a microphone for capturing user speech input and speakers or headphones for outputting the assistant's responses. The microphone allows the user to speak commands, while the speakers or headphones deliver the assistant's spoken responses, providing a seamless conversational experience.

**4.1.2 SPEECH RECOGNITION MODULE**

The Speech Recognition Module is responsible for converting the user's spoken input into text form. It receives audio input from the microphone and uses the SpeechRecognition library to process and convert the audio into recognized text. The module employs various algorithms and techniques, such as speech-to-text algorithms and machine learning models, to accurately transcribe the user's speech.

**4.1.3 COMMAND PROCESSING MODULE**

The Command Processing Module takes the recognized text from the Speech Recognition Module as input and processes it to determine the user's intent and execute the appropriate command. This module contains the logic and implementation for various commands, such as greetings, weather information retrieval, web searches, or other functionalities desired for the AI voice assistant. It matches the user's input with implemented commands, performs any necessary data processing or retrieval, and triggers the corresponding functionality.

**4.1.4 TEXT-TO-SPEECH MODULE**

The Text-to-Speech Module converts the assistant's responses or information into synthesized speech. It takes the response text generated by the Command Processing Module and uses the pyttsx3 library to convert it into speech. The module handles the conversion by utilizing speech synthesis algorithms and techniques to generate human-like speech output.

**4.1.5 CONTINUOUS LOOP**

The Continuous Loop phase enables the AI voice assistant to continuously listen for user commands, process them, and generate appropriate responses. The loop starts by capturing user speech input through the microphone using the Speech Recognition Module. The recognized text is then passed to the Command Processing Module for command identification and execution. The module generates a response text, which is subsequently converted into synthesized speech by the Text-to-Speech Module. The generated speech is outputted through the speakers or headphones, allowing the user to hear the assistant's response. The loop continues to listen for further user commands, creating an interactive and continuous conversational experience.

To implement the AI Voice Assistant Mini Project, you can follow these steps:

1] Set up the Environment:

* Install Python on your system.
* Install necessary libraries using pip: SpeechRecognition, pyttsx3, pyaudio, wikipedia, and datetime.

2] Import the Required Libraries:

* import speech\_recognition as sr
* import pyttsx3
* import datetime
* import wikipedia

3] Initialize the Text-to-Speech Engine:

* engine = pyttsx3.init()

4] Define the Speech Recognition Function:

def recognize\_speech():

r = sr.Recognizer()

with sr.Microphone() as source:

print("Listening...")

audio = r.listen(source)

try:

command = r.recognize\_google(audio)

print("You said:", command)

return command

except sr.UnknownValueError:

print("Sorry, I couldn't understand that.")

return ""

except sr.RequestError as e:

print("Sorry, I couldn't reach the speech recognition service.")

return ""

5] Define the Text-to-Speech Function:

def speak(text):

engine.say(text)

engine.runAndWait()

6] Implement the Greeting Function:

def greet():

hour = datetime.datetime.now().hour

if 0 <= hour < 12:

speak("Good morning!")

elif 12 <= hour < 18:

speak("Good afternoon!")

else:

speak("Good evening!")

7] Implement the Main Loop:

while True:

command = recognize\_speech()

if "hello" in command.lower():

greet()

elif "search" in command.lower():

search\_query = command.lower().replace("search", "").strip()

search\_result = wikipedia.summary(search\_query, sentences=2)

speak(search\_result)

else:

speak("Sorry, I didn't understand that.")

8] Run the Program:

Save the file with a .py extension (e.g., voice\_assistant.py) and run it using Python. The program will continuously listen for user commands and respond accordingly.

You can expand the functionality of the AI voice assistant by adding more commands and implementing additional features like weather information, music playback, or integration with external APIs.

**CHAPTER-5**

**CONCLUSION AND FUTURE WORK**

**5.1 CONCLUSION**

In conclusion, the AI Voice Assistant offers a practical and hands-on exploration of building a basic AI voice assistant using Python. The project incorporates speech recognition, text-to-speech conversion, and command processing to create an interactive conversational experience for users. By implementing functionalities like greetings, web searches, and information retrieval, the project demonstrates the potential of voice-enabled artificial intelligence in enhancing human-computer interaction.

Throughout the project, we have gained valuable experience in working with libraries such as SpeechRecognition and pyttsx3, understanding the fundamentals of speech recognition and synthesis, and implementing command-based functionalities. The project serves as a foundation for further customization and expansion, allowing you to explore additional features, integrate external APIs, and enhance the assistant's capabilities.

Developing an AI voice assistant not only enables practical applications but also fosters an understanding of artificial intelligence techniques and their integration into real-world scenarios. This project provides a stepping stone for future endeavors in the field, encouraging you to explore advanced concepts, natural language processing, machine learning, and voice interface design.

**5.2 FUTURE WORK**

**Contextual Conversations:**

Enable the assistant to maintain context and engage in more meaningful conversations. Develop mechanisms to remember previous interactions, handle follow-up questions, and provide coherent responses based on the context of the conversation.

**Personalization:**

Introduce personalization features to tailor the assistant's responses and functionalities to individual users. Implement user profiles, preferences, and user-specific commands to create a more personalized experience.

**Voice Recognition:**

Incorporate speaker recognition capabilities to identify and authenticate users based on their unique voice patterns. This can enable personalized responses and user-specific commands, enhancing the overall user experience.

**Integration with External Services and APIs:**

Extend the assistant's functionality by integrating it with various external services and APIs. For example, integrate with weather APIs to provide real-time weather updates, connect with music streaming platforms for music playback, or integrate with home automation systems for controlling smart devices.

**Multi-Language Support:**

Enhance the assistant's language capabilities to support multiple languages. Implement language detection mechanisms and enable seamless switching between different languages based on user preferences.

**Error Handling and Robustness:**

Improve error handling mechanisms to handle scenarios where user input is not recognized or understood. Implement error messages, suggestions, or fallback strategies to provide a better user experience.

**Continuous Learning:**

Implement mechanisms for the assistant to learn and adapt over time. Use techniques like machine learning, reinforcement learning, or user feedback to continuously improve the assistant's performance and accuracy.

**Integration with Mobile Devices or Smart Speakers:**

Explore integration possibilities with mobile devices or smart speakers to make the assistant accessible on various platforms and devices.

**CHAPTER-6**

**APPENDICES**

**6.1 APPENDIX-1**

**SOURCE CODE**

import pyttsx3 #pip install pyttsx3

import datetime #module

import speech\_recognition as sr

import wikipedia

import smtplib

import webbrowser as wb

import os #inbuilt

import pyautogui

import psutil #pip install psutil

import pyjokes # pip install pyjokes

import requests, json #inbuilt

engine = pyttsx3.init()

engine.setProperty('rate', 190)

voices = engine.getProperty('voices')

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

engine.setProperty('volume', 1)

#change voice

def voice\_change(v):

x = int(v)

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

speak("done sir")

#speak function

def speak(audio):

engine.say(audio)

engine.runAndWait()

#time function

def time():

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

speak("The current time is")

speak(Time)

#date function

def date():

year = int(datetime.datetime.now().year)

month = int(datetime.datetime.now().month)

date = int(datetime.datetime.now().day)

speak("The current date is")

speak(date)

speak(month)

speak(year)

def checktime(tt):

hour = datetime.datetime.now().hour

if ("morning" in tt):

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

speak("Good morning sir")

else:

if (hour >= 12 and hour < 18):

speak("it's Good afternoon sir")

elif (hour >= 18 and hour < 24):

speak("it's Good Evening sir")

else:

speak("it's Goodnight sir")

elif ("afternoon" in tt):

if (hour >= 12 and hour < 18):

speak("it's Good afternoon sir")

else:

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

speak("Good morning sir")

elif (hour >= 18 and hour < 24):

speak("it's Good Evening sir")

else:

speak("it's Goodnight sir")

else:

speak("it's night sir!")

#welcome function

def wishme():

speak("Welcome Back")

hour = datetime.datetime.now().hour

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

speak("Good Morning sir!")

elif (hour >= 12 and hour < 18):

speak("Good afternoon sir")

elif (hour >= 18 and hour < 24):

speak("Good Evening sir")

else:

speak("Goodnight sir")

speak("Hydra at your service, Please tell me how can i help you?")

def wishme\_end():

speak("signing off")

hour = datetime.datetime.now().hour

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

speak("Good Morning")

elif (hour >= 12 and hour < 18):

speak("Good afternoon")

elif (hour >= 18 and hour < 24):

speak("Good Evening")

else:

speak("Goodnight.. Sweet dreams")

quit()

#command by user function

def takeCommand():

r = sr.Recognizer()

with sr.Microphone() as source:

print("Listing...")

r.pause\_threshold = 0.5

audio = r.listen(source)

try:

print("Recognizing...")

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

#speak(query)

#print(query)

except Exception as e:

print(e)

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

return "None"

return query

#sending email function

def sendEmail(to, content):

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

server.ehlo()

server.starttls()

server.login("ajplayzsing07@gmail.com", "Joshua@2507")

server.sendmail("allen25072002@gmail.com", to, content)

server.close()

#screenshot function

def screenshot():

img = pyautogui.screenshot()

img.save(

"C:\\Users\\Hydra-AI-using-python3-\\screenshots\\ss.png"

)

#battery and cpu usage

def cpu():

usage = str(psutil.cpu\_percent())

speak('CPU usage is at ' + usage)

print('CPU usage is at ' + usage)

battery = psutil.sensors\_battery()

speak("Battery is at")

speak(battery.percent)

print("battery is at:" + str(battery.percent))

#joke function

def jokes():

j = pyjokes.get\_joke()

print(j)

speak(j)

#weather condition

def weather():

api\_key = "YOUR-API\_KEY" #generate your own api key from open weather

base\_url = "http://api.openweathermap.org/data/2.5/weather?"

speak("tell me which city")

city\_name = takeCommand()

complete\_url = base\_url + "appid=" + api\_key + "&q=" + city\_name

response = requests.get(complete\_url)

x = response.json()

if x["cod"] != "404":

y = x["main"]

current\_temperature = y["temp"]

current\_pressure = y["pressure"]

current\_humidiy = y["humidity"]

z = x["weather"]

weather\_description = z[0]["description"]

r = ("in " + city\_name + " Temperature is " +

str(int(current\_temperature - 273.15)) + " degree celsius " +

", atmospheric pressure " + str(current\_pressure) + " hpa unit" +

", humidity is " + str(current\_humidiy) + " percent"

" and " + str(weather\_description))

print(r)

speak(r)

else:

speak(" City Not Found ")

def personal():

speak(

"I am Hydra, version 1.0, I am an AI assistent, I am developed by H A H Technologies from CSE1"

)

speak("Now i hope you know me")

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

wishme()

while (True):

query = takeCommand().lower()

#time

if ('time' in query):

time()

#date

elif ('date' in query):

date()

#personal info

elif ("tell me about yourself" in query):

personal()

elif ("about you" in query):

personal()

elif ("who are you" in query):

personal()

elif ("yourself" in query):

personal()

elif ("developer" in query or "tell me about your developer" in query

or "father" in query or "who develop you" in query

or "developer" in query):

res = open("about.txt", 'r')

speak("here is the details: " + res.read())

#searching on wikipedia

elif ('wikipedia' in query or 'what' in query or 'who' in query

or 'when' in query or 'where' in query):

speak("searching...")

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

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

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

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

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

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

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

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

print(query)

print(result)

speak(result)

#sending email

elif ("send email" in query):

try:

speak("What is the message for the email")

content = takeCommand()

to = 'allen25072002@gmail.com'

sendEmail(to, content)

speak("Email has sent")

except Exception as e:

print(e)

speak(

"Unable to send email check the address of the recipient")

elif ("search on google" in query or "open website" in query):

speak("What should i search or open?")

chromepath = 'C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe %s'

search = takeCommand().lower()

wb.get(chromepath).open\_new\_tab(search + '.com')

#sysytem logout/ shut down etc

elif ("logout" in query):

os.system("shutdown -1")

elif ("restart" in query):

os.system("shutdown /r /t 1")

elif ("shut down" in query):

os.system("shutdown /r /t 1")

#play songs

elif ("play songs" in query):

speak("Playing...")

songs\_dir = "C:\\Music"

songs = os.listdir(songs\_dir)

os.startfile(os.path.join(songs\_dir, songs[1]))

quit()

#reminder function

elif ("create a reminder list" in query or "reminder" in query):

speak("What is the reminder?")

data = takeCommand()

speak("You said to remember that" + data)

reminder\_file = open("data.txt", 'a')

reminder\_file.write('\n')

reminder\_file.write(data)

reminder\_file.close()

#reading reminder list

elif ("do you know anything" in query or "remember" in query):

reminder\_file = open("data.txt", 'r')

speak("You said me to remember that: " + reminder\_file.read())

#screenshot

elif ("screenshot" in query):

screenshot()

speak("Done!")

#cpu and battery usage

elif ("cpu and battery" in query or "battery" in query

or "cpu" in query):

cpu()

#jokes

elif ("tell me a joke" in query or "joke" in query):

jokes()

#weather

elif ("weather" in query or "temperature" in query):

weather()

#Hydra features

elif ("tell me your powers" in query or "help" in query

or "features" in query):

features = ''' i can help to do lot many things like..

i can tell you the current time and date,

i can tell you the current weather,

i can tell you battery and cpu usage,

i can create the reminder list,

i can take screenshots,

i can send email to your boss or family or your friend,

i can shut down or logout or hibernate your system,

i can tell you non funny jokes,

i can open any website,

i can search the thing on wikipedia,

i can change my voice from male to female and vice-versa

And yes one more thing, My boss is working on this system to add more features...,

tell me what can i do for you??

'''

print(features)

speak(features)

elif ("hii" in query or "hello" in query or "goodmorning" in query

or "goodafternoon" in query or "goodnight" in query

or "morning" in query or "noon" in query or "night" in query):

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

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

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

if ("morning" in query or "night" in query or "goodnight" in query

or "afternoon" in query or "noon" in query):

checktime(query)

else:

speak("what can i do for you")

#changing voice

elif ("voice" in query):

speak("for female say female and, for male say male")

q = takeCommand()

if ("female" in q):

voice\_change(1)

elif ("male" in q):

voice\_change(0)

elif ("male" in query or "female" in query):

if ("female" in query):

voice\_change(1)

elif ("male" in query):

voice\_change(0)

#exit function

elif ('i am done' in query or 'bye bye Hydra' in query

or 'go offline Hydra' in query or 'bye' in query

or 'nothing' in query):

wishme\_end()

**6.2 APPENDIX-2**

**OUTPUT**

**CHAPTER-7**

**REFERENCES**

**7.1 REFERENCES**

[1] Vivek Sharma; Abhishek Chauhan;Harsh Saxena;Shubham Mishra; Secure File Storage on Cloud using Hybrid Cryptography “ 2021 5th International Conference on Information Systems and Computer Networks (ISCON).

[2] Putta Bharathi;Gayathri Annam;Jaya Bindu Kandi;Vamsi KrishnaDuggana; Anjali T. “Secure File Storage using Hybrid Cryptography” 2021 6th International Conference on Communication and Electronics Systems (ICCES).

[3] [Punam V. Maitri](https://ieeexplore.ieee.org/author/37085468226);[Aruna Verma](https://ieeexplore.ieee.org/author/37085831568) “ Secure file storage in cloud computing using hybrid cryptography algorithm” [2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)](https://ieeexplore.ieee.org/xpl/conhome/7561562/proceeding)

[4] Vyom Verma; Pulkit Kumar; Raj Kumar Verma; Shristi Priya “A Novel Approach for Security in Cloud Data Storage Using AES-DES Hybrid Cryptography”

[5] Surya Nepal; Carsten Friedrich; Leakha Henry; Shiping Chen “A Secure Storage Service in the Hybrid Cloud” 2011 Fourth IEEE International Conference on Utility and Cloud Computing.

[6] [H Sai Charan Reddy](https://ieeexplore.ieee.org/author/37089329267);[Vajjala Veera Karthik](https://ieeexplore.ieee.org/author/37089328992);[Dedeepya V](https://ieeexplore.ieee.org/author/37089329442);[A Pavan](https://ieeexplore.ieee.org/author/37089328671);[Sarasvathi V](https://ieeexplore.ieee.org/author/37089329471) “Data Storage on Cloud using Split-Merge and Hybrid Cryptographic Techniques” 2022 International Conference for Advancement in Technology (ICONAT)

[7] Osama Ahmed Khashan “Secure Outsourcing and Sharing of Cloud Data Using a User-Side Encrypted File System”.

[8] [Guilherme Sperb Machado](https://ieeexplore.ieee.org/author/37088481959);[Thomas Bocek](https://ieeexplore.ieee.org/author/37295319100);[Burkhard Stiller](https://ieeexplore.ieee.org/author/37274556000) “PiCsMu: A system to aggregate multiple heterogeneous Cloud Services' storage” 2014 IEEE Network Operations and Management Symposium (NOMS).

[9] [Sanjeev Kumar](https://ieeexplore.ieee.org/author/37087139644);[Garima Karnani](https://ieeexplore.ieee.org/author/37088879795);[Madhu Sharma Gaur](https://ieeexplore.ieee.org/author/37087139336);[Anju Mishra](https://ieeexplore.ieee.org/author/37088878453) “Cloud Security using Hybrid Cryptography Algorithms” 2021 2nd International Conference on Intelligent Engineering and Management (ICIEM).

[10] [Osama Fouad Abdel Wahab](https://ieeexplore.ieee.org/author/37088761977);[Ashraf A. M. Khalaf](https://ieeexplore.ieee.org/author/37445008300);[Aziza I. Hussein](https://ieeexplore.ieee.org/author/37276922200);[Hesham F. A. Hamed](https://ieeexplore.ieee.org/author/37340443100) “Hiding Data Using Efficient Combination of RSA Cryptography, and Compression Steganography Techniques”.

[11] [Jun Ren](https://ieeexplore.ieee.org/author/37086142276);[Zhiqiang Yao](https://ieeexplore.ieee.org/author/37398725400);[Jinbo Xiong](https://ieeexplore.ieee.org/author/37076768400);[Yuanyuan Zhang](https://ieeexplore.ieee.org/author/37086144756);[Ayong Ye](https://ieeexplore.ieee.org/author/37064562100) “[A Secure Data Deduplication Scheme Based on Differential Privacy](https://ieeexplore.ieee.org/document/7823884/)” 2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS).

[12] [V. P. Lalitha](https://ieeexplore.ieee.org/author/37085785097);[M. Y. Sagar](https://ieeexplore.ieee.org/author/37089577827);[S. Sharanappa](https://ieeexplore.ieee.org/author/37086405734);[Shredar Hanji](https://ieeexplore.ieee.org/author/37086404739);[R Swarup](https://ieeexplore.ieee.org/author/37086406422) Data security in cloud 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)

[13] [C. Radhakrishnan](https://ieeexplore.ieee.org/author/37088801113);[K. Karthick](https://ieeexplore.ieee.org/author/37088802906);[R. Asokan](https://ieeexplore.ieee.org/author/37392701700) “Fragmentation Based Hybridized Encryption Scheme for Cloud Environment” 2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE).