Project Zenith

Project Report submitted in partial fulfillment of the requirements for the end semester project for the degree

of

Bachelor of Science in Computer Science and Engineering

by

Akshat Rastogi
E20CSE003

Samarth Goyal | **Saumya Tripathi**
E20CSE084 | **E20CSE030**

Under the supervision of

Dr. Vijaypal Singh Rathor and Dr. Divya Srivastava

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BENNETT UNIVERSITY, GREATER NOIDA

WINTER 2021

DECLARATION

Project Title	Project Zenith
Authors	*Akshat Rastogi*, *Samarth Goyal*, and *Saumya Tripathi*
Student IDs	E20CSE003, E20CSE084, and E20CSE030
Supervisor	Dr. Vijaypal Singh Rathor and Dr. Divya Srivastava

We declare that this thesis entitled *Project Zenith* is the result of our own work except as cited in the references. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Akshat Rastogi
E20CSE003

Department of Computer Science and Engineering
Bennett University, Greater Noida

Samarth Goyal
E20CSE084

Department of Computer Science and Engineering
Bennett University, Greater Noida

Saumya Tripathi
E20CSE030

Department of Computer Science and Engineering
Bennett University, Greater Noida

Date: February 19, 2021

Department of Computer Science and Engineering
Bennett University
Greater Noida, Delhi NCR - 203206

CERTIFICATE

This is to certify that the thesis entitled **Project Zenith**, submitted by **Akshat Rastogi** (Student ID: E20CSE003), **Samarth Goyal** (Student ID: E20CSE084) and **Saumya Tripathi** (Student ID: E20CSE030) are undergraduate students of the **Department of Computer Science and Engineering** has been examined. Upon recommendation by the examination committee, we hereby accord our approval of it as the presented work and submitted report fulfill the requirements for its acceptance in partial fulfillment for the end semester project for the degree of *Bachelor of Science in Computer Science and Engineering*.

Dr. Vijaypal Singh Rathor and Dr. Divya Srivastava
Supervisor's Position

Department of Computer Science and Engineering
Bennett University, Greater Noida

Dr. Deepak Garg
Professor and Head

Department of Computer Science and Engineering
Bennett University, Greater Noida

Place: Noida
Date: February 19, 2021

ACKNOWLEDGEMENTS

We would like to express our deep and sincere gratitude to our research supervisor, *Dr. Vijaypal Singh Rathor and Dr. Divya Srivastava*, for giving us the opportunity to conduct research and providing invaluable guidance throughout this work. Their dynamism, vision, sincerity and motivation have deeply inspired us. They have taught us the methodology to carry out the work and to present the works as clearly as possible. It was a great privilege and honor to work and study under their guidance.

We are greatly indebted to our honorable teachers of the Department of Computer Science and Engineering at the Bennett University, Greater Noida who taught us during the course of our study. Without any doubt, their teaching and guidance have completely transformed us to the persons that we are today.

We are extremely thankful to our parents for their unconditional love, endless prayers and caring, and immense sacrifices for educating and preparing us for our future. We would like to say thanks to our friends and relatives for their kind support and care.

Finally, we would like to thank all the people who have supported us to complete the project work directly or indirectly.

Akshat Rastogi, *Samarth Goyal* and *Saumya Tripathi*
Bennett University, Greater Noida
Date: February 19, 2021

Dedicated to
the republic of India

– Akshat Rastogi

Dedicated to
all my techy friends

– Samarth Goyal

Dedicated to
my lazy friends

– Saumya Tripathi

ABSTRACT

We present to you Moto, a smart, multi-functional virtual assistant that can easily be customized to cater the needs of the user, helping them multitask and get things done.

As an exemplification if you're a software developer or a new to coding, Project Zenith can act as your coding assistant like it can create classes or functions for you at your command. It can also open basic coding related apps or websites like pycharm, stackoverflow, vs code etc and if that's not enough our voice assistant can lighten up your mood with some programming jokes.

Apart from these developer specific features our voice assistant can perform a wide variety of tasks from various domains. Be it some general tasks like automating emails and Whatsapp or fun feature like horror pranks and even checking you and your partner's compatibility on Love Calculator. Jokes aside, Moto also has some real-life applications which are very relevant in today's times like Face Mask detection using Tensorflow and Facial Recognition. Our bot can do all the redundant and monotonous tasks like searching for topic on the internet, typing emails,writing boilerplate code,browsing multiple websites etc along with a set of many superfluous tasks, that consume the user's precious time, that otherwise could be utilized in doing some productive work.

Along with several usual features found in most of the voice assistants in the market, our assistant has certain unique and state-of-the-art features like face recognition, human mood recognition, face-mask-recognition, coding-helper, social-media automation, shopping automation, Messaging and Music Automation etc, that make our project in competition with the other commercial products in the market.

Some downfalls of our project include, slight increase in the latency of speech recognition and also errors in the precision of the recognised speech.

In conclusion, Project Zenith is still in need of a few improvements and since the project is based on open-source architecture, we expect that these issues will be resolved soon in the future.

Keywords: Open-Source Architecture, Voice Assistant

Contents

List of Figures

Chapter 1

Building the Base

In order to build a voice assistant, we needed to lay the base of our voice assistant, upon which additional modules and features can be built.This required us to focus on speech recognition and text-to-speech properties in the initial phase of our project.

1.1 Speech recognition

1.1.1 Method used

In order for the bot to be able to take audio input from the user, speech-recognition library was used which supports several engines and APIs both online as well as offline. The flexibility and ease-of-use of this package made it an excellent choice for our Python project. We created a function containing various functionalities of this library such as Recognizer(), Microphone() etc. to take in voice commands (using Google Web Speech API). An additional requirement to use this module was PyAudio 0.2.11+.

1.1.2 How to use

Upon starting the program the user would automatically be asked to provide an audio input. The user is expected to give the voice commands after the term "Listening…" appears on the screen. The language of the recognition has been set to American English (us-en). If the audio input cannot be recognized an exception will be raised which would ask the user to provide the input again. In case the user

does not wish to give any input the command “Stop” can be used. In case if no input is provided by the user Motto goes into sleep mode.

```
def run(self):
    self.TaskExecution

def takecommand(self):
    r = sr.Recognizer()
    with sr.Microphone() as source:
        print("Listening...")
        r.pause_threshold = 1
        audio = r.listen(source,timeout=1,phrase_time_limit=5)

    try:
        print("Recognizing...")
        query = r.recognize_google(audio,language='en-in')
        print(f"User said: {query}")

    except Exception as e:
        speak("Please say it again.")
        return "none"
    return query
```

Figure 1.1: Screenshot of the python code used for speech recognition

1.2 Text to Speech

This feature was added to give a voice to our bot and enliven the experience. We also deployed this feature to turn a pdf into an audiobook.

1.2.1 Method used

We used pttsx3 library for converting text to speech as it works without internet connection or delay and supports multiple TTS (text to speech) engines, including Sapi5. We changed the default voice by pulling the ‘voices’ property from the engine. A function was created that took text input and generated the required audio output.

```python
engine = pyttsx3.init('sapi5')
voices = engine.getProperty('voices')
# print(voices[1].id)
engine.setProperty('voices',voices[1].id)

def speak(audio):
    engine.say(audio)
    print(audio)
    engine.runAndWait()
```

Figure 1.2: Screenshot of the python code used for text-to-speech

1.3 General Layout of the Program

Building the general layout for the execution of multiple commands is still another challenge to overcome. The general layout of the voice bot needed to be robust and reliable in order for successful execution of user-commands whist maintaining the dialog flow between the user and the voice bot. So here are the few key components of our general layout.

1.3.1 Combinations of infinite while loops

Using the speech recognition and text to speech conversion, we ran an infinite while loop and took user command at the start of the loop and applied several conditions upon those received commands to execute the functions necessary.

```python
while True:

    self.query = self.takecommand().lower()

    if "open notepad" in self.query:
        npath ="C:\\WINDOWS\\system32\\notepad.exe"
        os.startfile(npath)
    #Checked
```

Figure 1.3: Screenshot of the python code used for initial infinite while loop

Towards the end of the while loop we made another while loop that gets activated whenever there is no command from the user or the user wants the bot to go to a sleep mode.

```
#not for a while
elif "not for some time" in self.query or "not for sometime" in self.query or
    speak("Thanks for availing my services")
    speak("Just say the words Wake Up if you need me anytime soon")
    while True:
        print("Listening")
        text = get_audio()
        if text.count("wake up") > 0:
            speak("I am ready")
            speak("Good to be back at your service")
            speak("What can I do for you ?")
            break
else:
    speak("Done")
    # speak("Do you have any other request?")
```

Figure 1.4: Screenshot of the python code used for second infinite while loop

1.4 Challenges Faced

The voice command was giving an error for timeout while waiting for user command. So we needed to fix that by making a separate function to take user commands but without the pause threshold and a few other properties that we specified before. This allowed us to successfully run an infinite loop of listening and recognizing until the user says 'wake up'.

```python
def get_audio():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        audio = r.listen(source)
        said = ""

        try:
            said = r.recognize_google(audio, language='en-in')
            print(said)
        except Exception as e:
            print("Exception: " + str(e))

    return said.lower()
```

Figure 1.5: Screenshot of python code used to counter challenge

Chapter 2

Basic Functionalities

Now that we had the base ready for the construction of our voice assistant, we needed to move on to phase two of our project and build some rudimentary functionalities that our voice assistant could perform.

Following is a list of the features that we included in phase two of Project Zenith.

1. Opening Applications on local machine
2. Opening Websites on local machine
3. Capturing the I.P (Internet Protocol) Address of local machine
4. Performing Wikipedia Searches
5. Playing music on local machine
6. Announcing Weather Forecast
7. Making Programming Jokes
8. PDF to audio book converter
9. Taking Screenshots
10. Controlling the local machine
11. Getting the geo-location of local machine
12. Tic-Tac-Toe Game
13. Tossing Game

14. Stone-Paper-Scissor Game
15. Performing online voice search
16. Getting Dictionary meaning
17. Playing Youtube Videos
18. Random Password Generator
19. Love Calculator Prank

These functionalities are primitive to almost all voice assistants and are easy to create. Because of that only a few of them have been covered in the following detailed documentation.

2.1 Open Applications

Instead of having to search for and open an app or a file yourself, we decided to automate this mundane task for you at a single command. These include:

1. Notepad
2. Mozilla Firefox
3. Command Prompt (EDEX-UI)
4. Microsoft-Teams and many more.

2.1.1 How to Activate

1. Notepad: Use voice command “open notepad”
2. Mozilla Firefox: Use voice command “open Mozilla Firefox”
3. Ms-Teams: Use voice command “open ms-teams” or “open Microsoft teams”
4. Command Prompt: Use voice command “open command prompt”

2.1.2 Method Used

To achieve this funtionality, we used the "os" library in python. The startfile method of the os class allowed us to open any application as long as we provide the path to those applications beforehand.

2.1.3 Scope of improvement

The user can include various other apps or files in the program that we have missed should they wish to do so.

```
self.query = self.takecommand().lower()

if "open notepad" in self.query:
    npath ="C:\\WINDOWS\\system32\\notepad.exe"
    os.startfile(npath)

elif "open mozilla firefox" in self.query:
    mpath ="C:\\Program Files\\Mozilla Firefox\\firefox.exe"
    os.startfile(mpath)

elif ("open ms teams" or "open microsoft teams") in self.query:
    tpath ="C:\\Users\\Saurabh Tripathi\\AppData\\Roaming\\Microsoft\\Windows\\St
    os.startfile(tpath)

elif "open command prompt" in self.query:
    os.system("Start cmd")
```

Figure 2.1: Screenshot of python code used to open applications on local machine

2.2 Opening Websites

Some of the websites that can be opened on voice command include Google, Gmail, Facebook, Stack overflow, YouTube, Bennett LMS etc. The user can also directly search for a song on YouTube.

2.2.1 How to Activate

1. Google: Use voice command “open google”
2. Gmail: Use voice command “open gmail”
3. Facebook: Use voice command “open Facebook”
4. Stack overflow: Use voice command “open stack overflow”
5. YouTube: Use voice command “open YouTube”
6. Bennett LMS: Use voice command “open lms”

2.2.2 Method Used

To achieve this funtionality, we used the “webbrowser” library in python. The open method of the webbrowser class allowed us to open any website on the browser provided the universal resource locator (URL) path to the website.

```
elif "open gmail" in self.query:
    webbrowser.open("www.gmail.com")

elif "open stackoverflow" in self.query:
    webbrowser.open("www.stackoverflow.com")

elif "open facebook" in self.query:
    webbrowser.open("www.facebook.com")

elif "open lms" in self.query:
    webbrowser.open("https://lms.bennett.edu.in/login/index.php")

elif "open google" in self.query:
    speak("What should I search on google?")
    cm = self.takecommand().lower()
    webbrowser.open(f"{cm}")
```

Figure 2.2: Screenshot of python code used to open webiste in browser

2.3 Capturing the I.P Address of local machine

To achieve this result we used a public IP Address API.

2.3.1 How to Activate

1. Simply use any command that contains “ip address” in it.

2.3.2 Method Used

To achieve this functionality, we used the an online I.P Address API offered by ’https://api.ipfy.org’

```
elif "ip address" in self.query:
    ip = get("https://api.ipify.org").text
    speak(f"your ip address is {ip}")
```

Figure 2.3: Screenshot of python code used to obtain Internet Protocol (IP) Address of the local machine

2.4 Performing Wikipedia Searches

Using this feature our voice assistant can search for any provided topic and return the first few sentences of the paragraph from the Wikipedia page. For this purpose we used the ”wikipedia” library.

2.4.1 How to Activate

1. Your voice command simply needs to include the word “Wikipedia” in it.

2.4.2 Method Used

To achieve this functionality, we used the “wikipedia” library in python. The ”summary” method of the wikipedia class allowed us to obtain the first ’n’ lines from the wikipedia page for that particular topic.

```
elif "wikipedia" in self.query:
    speak("searching Wikipedia...")
    query = query.replace("wikipedia","")
    results = wikipedia.summary(query,sentences=2)
    speak("According to Wikipedia")
    speak(results)
```

Figure 2.4: Screenshot of python code used to obtain the first 'n' lines from the wikipedia page for a particular topic

2.5 Playing Music on Local Machine

Relax to your favourite songs or playlist using this feature.

2.5.1 How to Activate

1. Your voice command simply needs to include the phrase "play music" in it.

2.5.2 Method Used

To achieve this functionality, we used the "os" library in python. The "startfile" method of the "os" class allowed us to open all pre-downloaded music stored in our local machine.

```
elif "play music" in self.query:
    music_dir = r"C:\Users\Akshat\Documents\GitHub\Project-ASS\songs"
    songs = os.listdir(music_dir)
    rd = random.choice(songs)
    os.startfile(os.path.join(music_dir,rd))
```

Figure 2.5: Screenshot of python code used to obtain the first 'n' lines from the wikipedia page for a particular topic

2.6 Announcing Weather Forecast

2.6.1 How to Activate

1. Your voice command simply needs to include the word “weather” in it.

2.6.2 Method Used

To achieve weather functionality, we used the “requests” The basic principle behind the weather forecast library is that it basically it is working on Realtime database fetching the weather according to API attached in the code

```
import requests
from datetime import datetime
user_api="ad192de52a3a24431ab1c95f5917bf22"
# location=input("Enter the City Name: ")


def Weather(city_name):
    complete_api_link = "http://api.openweathermap.org/data/2.5/weather?q=" + city_name +
    api_link=requests.get(complete_api_link)
    api_data=api_link.json()

    if api_data['cod']=='404':
        print("Invalid City :{},Please check your city name".format(city_name))
    else:
        temp_city=((api_data['main']['temp'])-273.15)
        weather_desc=api_data['weather'][0]['description']
        hmdt=api_data['main']['humidity']
        wind_spd=api_data['wind']['speed']
```

Figure 2.6: Screenshot of python code used to obtain weather information after receiving the city name as input

Since our code requires the name of the city in order to return the weather, we used another API service provided by ’https://ipinfo.io’ which returns a json file (interpreted as a dictionary in python) from which we can extract the name of the city and use that as an input for the weather function.

```
elif "weather" in self.query:
    res = requests.get('https://ipinfo.io')
    data = res.json()
    city = data['city']
    temp_city, weather_desc, hmdt, wind_spd = weather.Weather(str(city))
    temp_city = str(temp_city)[:5]
    speak(f"It is {weather_desc}y outside with a temperature of {temp_city}
    speak(f"Current humidity is {hmdt} percent and wind is expected to blow
#Checked
```

Figure 2.7: Screenshot of python code used to obtain city name and providing it as an input for weather function

2.7 Making Programming Jokes

Feeling down or bored? Use this feature to enlighten your mood with various one-liner programming jokes. For this had to import the pyjokes library. Use the voice command .

2.7.1 How to Activate

1. Your voice command simply needs to include the phrase ""tell me a joke" in it.

2.7.2 Method Used

To achieve weather functionality, we used the "pyjokes" library and specified 'programming' in its config file, so that whenever we ask our voice assistant to tell me a joke. Jokes related to programming are returned.

```
elif "tell me a joke" in self.query:
    joke = pyjokes.get_joke("en")
    speak(joke)
```

Figure 2.8: Screenshot of python code used to return a programming joke

2.8 PDF to Audio Book Converter

We have also included a functionality that can convert any PDF into the form of an audio-book.

2.8.1 How to Activate

1. Your voice command simply needs to include the phrase “read a pdf” or ”read pdf” in it.

2.8.2 Method Used

To achieve this functionality, we used the “PyPDF2” library. By using the ”PdfFileReader”, ”numPages”, ”getPage” and ”extractText” method in the ”PyPDF” class we were able able to convert the PDF into the audio-book format.

```
def pdf_reader() :
    pn = input("Please enter the full name of the pdf with extension:")
    book = open(pn,'rb')
    pdfReader = PyPDF2.PdfFileReader(book)
    pages = pdfReader.numPages
    speak(f"Total numbers of pages in this book {pages}")
    speak("Please enter the page number I have to read")
    pg = int(input ("Please enter the page number: ") )
    page = pdfReader.getPage(pg)
    text = page.extractText()
    speak(text)
```

Figure 2.9: Screenshot of python code used to a PDF into audiobooks

2.9 Taking Screenshots

We can take screenshot on voice commands and the screenshot would by default be saved on your desktop.

2.9.1 How to Activate

1. Your voice command simply needs to include the phrase “take a screenshot” or “take screenshot” in it.

2.9.2 Method Used

To achieve this functionality, we used the “PyAutoGUI” library. By using the ”shreenshot” method in the ”PyAutoGUI” class we were able able to take a screen shot of the screen and then we saved that image into the descktop folder.

```
elif "take screenshot" in self.query or "take a screenshot" in self.query:
        speak("Please tell me the name for this screenshot file")
        name = self.takecommand().lower()
        speak("please wait while I take sreenshot")
        time.sleep(3)
        img = pyautogui.screenshot()
        img.save(f"Desktop/{name}.png")
        speak("Screenshot has been saved on your Desktop.")
```

Figure 2.10: Screenshot of python code used to take Screenshots and save the resulting image file on desktop

2.10 Controlling the Local Machine

This functionality allows to perform system related actions like shutting down, restarting or sleeping the system. You can also use this to toggle between windows.

2.10.1 How to Activate

1. To ShutDown the local machine just say ”shut down the system”
2. To restart the local machine just say ”restart the system”
3. To put the local machine to sleep just say ”sleep the system”
4. To switch the current window just say ”switch the window”

2.10.2 Method Used

To achieve this functionality, we used the “PyAutoGUI” and ”os” library. By using the ”keyDown”, ”press”, ”keyUp” methods of the ”PyAutoGUI” class we were able to switch the current screen.

By using the ”system” method of the ”os” class, we were able to control different states of the local machine like shut-down, sleep or restarting the local machine on voice commands.

```
elif "tell me a joke" in self.query:
    joke = pyjokes.get_joke("en")
    speak(joke)

elif "shut down the system" in self.query:
    os.system("shutdown /s /t 5")

elif "restart the system" in self.query:
    os.system("shutdown /r /t 5")

elif "sleep the system" in self.query:
    os.system("rundll32.exe owrprof.dll, SetSuspendState 0,1,0")

elif "switch the window" in self.query:
    pyautogui.keyDown("alt")
    pyautogui.press("tab")
    time.sleep(1)
    pyautogui.keyUp("alt")
```

Figure 2.11: Screenshot of python code used to control the local machine

2.11 Getting the geo-location of local machine

2.11.1 How to Activate

To active geo-location service you can say the following sentences

1. What’s my location
2. Where am I
3. What’s my geo-location

Keywords identified in the program “geolocation”, “location”, “where am I”

2.11.2 Method Used

By activating the Geo-location feature of project zenith, we get back the city and country as a result.

To achieve this we used the IP address of the network that the user is connected to and used a website service ('https://ipinfo.io') and from the resulting JSON, that python converts in the form of a dictionary, we found out the name of the city and the name of the country

2.11.3 Challenge Faced

We used several other methods to achieve this, one of which was to get just get the IP address of the local machine from one of such websites and to use that IP address as an input parameter for another website service that would return the location of the person. But even after several modifications and multiple attempts, the code kept failing, so we researched on the internet and found out an even more easy and efficient way of achieving the same result.

2.12 Performing online voice search

We included this feature to enable two-way user interaction with the internet. So the use our voice assistant to actively seacrh the internet about a particular topic.

2.12.1 How to Activate

To active voice search service you can say the following sentences

1. what is topic name
2. search online about topic name
3. What's my geo-location

Keywords identified in the program: "what is", "search online"

2.12.2 Method Used

To achieve this functionality, we used the “pywhatkit” library. By using the ”info”method of the ”pywhatkit” class we were able able get the results from the internet about any searched topic.

Upon examination of the info method we found out that the forth mentioned method also uses the Wikipedia library and essentially returns 3 lines of the searched result .

```
import pywhatkit

def online_search(topic):
    return pywhatkit.info(topic)
```

Figure 2.12: Screenshot of python code used to get online search results from the internet

2.13 Getting Dictionary Meaning

2.13.1 How to Activate

To active dictionary search service you can say the following sentences

1. Hey Moto! Dictionary topic-name
2. Word Meaning of topic-name

Keywords identified in the program: ”word meaning”, ”dictionary”

2.13.2 Method Used

To achieve Dictionary, we used the “PyDictionary” library. It will fetch the letter meaning entered in the run panel then It will search the word meaning from the python library PyDictionary and it speaks the meaning

```python
from PyDictionary import PyDictionary

def dictionary_lol(letter):
    def meaning(word):
        dic = PyDictionary()
        mean = dic.meaning(word)
        return mean

    word_temp = meaning(letter)

    for state in word_temp:
        return str(word_temp[state][0])
```

Figure 2.13: Screenshot of python code used to get dictionary meaning of a particular word

2.14 Playing YouTube Videos

2.14.1 How to Activate

To play a YouTube video you can say the following sentences

1. Hey Moto! play a video on YouTube
2. I want to see a YouTube Video

Keywords identified in the program: ”video on YouTube”, ”YouTube video”

2.14.2 Method Used

To achieve this functionality, we used the “pywhatkit” library. By using the ”playonyt”method of the ”pywhatkit” class we were able able play the most recent video on that particular topic on YouTube.

Upon examination of the playonyt method we found out that the forth mentioned method utilised the requests library and modifies the YouTube URL to search for a video and then opens the top video that pops in the search results.

```
from webdriver_manager.firefox import GeckoDriverManager
wsh = comctl.Dispatch("WScript.Shell")
# ChromeDriverManager().install()

driver = webdriver.Firefox(executable_path=GeckoDriverManager().install())

driver.get('https://www.youtube.com')
search=driver.find_element_by_xpath('/html/body/ytd-app/div/div/ytd-masthead/div

n="rinkiya ke papa"

search.send_keys(n)
wsh.SendKeys('{Enter}')
driver.implicitly_wait(5)
drive=driver.find_element_by_xpath("/html/body/ytd-app/div/ytd-page-manager/ytd-
drive.click()
while(1):
    response=driver.page_source
    time.sleep(20)
    soup=BeautifulSoup(response,'lxml')
    current=soup.find(class_="ytp-time-current")
    last=soup.find(class_='ytp-time-duration')
    if current.text==last.text:
        continue
```

Figure 2.14: Partial screenshot of python code used automate YouTube by using selenium library

We also created another module for automating Youtube using the selenium library (Figure 2.14), but since the pywhatkit library was considerably faster, we decided to drop the automations using Selenium.

2.15 Love Calculator Prank

2.15.1 How to Activate

To play a YouTube video you can say the following sentences

1. Hey Moto! I am in love with person-name

```
elif "video on youtube" in self.query or "youtube video" in self.query:
    speak("What is the name of the video?")
    sm = self.takecommand().lower()
    kit.playonyt(sm)
```

Figure 2.15: Screenshot of python code used to open the most recent YouTube video on any provided topic

2. Open Love Calculator

3. Calculate Love

Keywords identified in the program: ”love calculator”, ”calculate love” and ”love with”

2.15.2 Method Used

To achieve Love Calculator, we used the “random” library .

The basic principle behind the random library is that it take out random integer number value and it shows us in percentage.

```
elif "love calculator" in self.query or "calculate love" in self.query or "love with" in self.query:
    speak("Opening love calculator")
    speak("Who is the first person")
    time.sleep(3)
    speak("Who is the second person")
    time.sleep(3)
    speak("Calculating Compatibility percentage by using Love Algorithm")
    percentage = love_calculator.love()
    speak(f"They both are {percentage} percent compatible with each other")
```

Figure 2.16: Screenshot of python code used to open the most recent YouTube video on any provided topic

2.16 Random Password Generator

2.16.1 How to Activate

To active the password generator you can say the following sentences

1. Give me a random password
2. Hey can you generate a password real quick
3. Activate Password Generator

The base of the commands is to have the word ”password” in it

2.16.2 Purpose

Often times me need to enter random generated passwords in websites to enhance password strength and improve security of your account from getting cracked. That is why we integrated a random password generator in Project Zenith that can generate random password of any characters, symbols or numbers.

2.16.3 Method Used

To achieve this result we used the Random library, to generate random elements from predefined character list.

2.17 Summary

In the second phase of our project, we laid out some basic features and integrated them into our voice assistant. To improve the efficiency of our voice bot, we framed multiple sentences to call the same functionality and then mapped out the most common keywords to look for in the user query.

We also took in precaution the general usage of the keywords in order to not mistake them for something that user said to someone else and not the voice assistant.

In conclusion, the features we added in this phase are pretty rudimentary for a general purpose voice assistant. Now we will start customizing it, with additional features in the next phase.

```python
def generatepassword(nr_letters, nr_symbols, nr_numbers):
    total = nr_letters + nr_numbers + nr_symbols

    passwordlist = []

    for _ in range(1, nr_letters + 1):
        lol = random.choice(letters)
        passwordlist.append(lol)

    for _ in range(1, nr_symbols + 1):
        lmao = random.choice(symbols)
        passwordlist.append(lmao)

    for _ in range(1, nr_numbers + 1):
        rofl = random.choice(numbers)
        passwordlist.append(rofl)

    finalpassword = ""

    random.shuffle(passwordlist)

    for i in passwordlist:
        finalpassword += i

    return(finalpassword)
```

Figure 2.17: Screenshot of python code used to generate a random password

Chapter 3

Intermediate Functionalities

Now that we have added the basic functionalities into project zenith, we are ready to move onto phase three of our project and build some intermediate level functionalities into our voice assistant.

Following is a list of the features that we included in phase three of Project Zenith.

1. News Crawler
2. Searching Instagram Profiles Online
3. Badminton Game using Computer Vision
4. Voice Operated Stopwatch
5. Voice Operated Reminder
6. Spam Bot
7. Text to Handwriting Converter
8. Email Automation
9. WhatsApp Automation
10. Instagram Automation
11. Zero-Width Fingerprinting
12. Sorting Visualizer

13. Spotify Automation
14. Shopping Automation
15. Security Camera
16. Generating Random Quotes
17. Reading E-mails
18. Voice Operated Calculator

These functionalities are mostly focused on Automation along with a few interesting ones that are useful in every-day activities of students and developers.

3.1 News Crawler

As the world is evolving we often don't have time to listen to or watch news online or on television in order for us to get the information we want for a particular topic. Suppose you wanted to find the latest news about Reliance because you have invested in their stocks, it would take you a lot of time and energy to go to google and manually scrape out news headlines and their descriptions whilst you are bombarded with advertisements that drastically reduce your reading time and cause irritation. By using the news crawler feature of Project Zenith, you can let the AI scrape out news articles from around the world and read them out to you one at a time.

3.1.1 How to Activate

To activate the news crawler you can say the following sentences

1. Show me some news articles
2. Whats the latest news
3. What's in the news today etc

The base of the commands is to have the word news in it

3.1.2 Method Used

To achieve this result we used a very popular news crawling API called NewsAPI (https://newsapi.org/). This is a free to use (paid also) one stop platform to scrape news articles from around the world and it returns the collected information in the form of dictionary in python. We referenced the boilerplate code in their documentation and managed to achieve the results that we were expecting. We decided to recite the top 5 news articles from the returned data in the form of news headlines and also some part of their description.

```
# News
elif "news" in self.query:
    speak("On what topic do you want me to scrape news articles ?")
    topic = self.takecommand().lower()
    list_of_news = news_api.get_news(topic)
    speak("Reading the top 5 news articles on the given topic")
    for i in range(5):
        list_new = list_of_news[i]
        speak("Article {}".format(str(i+1)))
        speak(list_new[0])
        speak("Description")
        speak(list_new[1])
```

Figure 3.1: Screenshot of python code to call the newsAPI module

3.2 Searching Instagram Profiles Online

This functionality helps you search and even download any profile on Instagram.

3.2.1 How to Activate

Say the command “Instagram profile” or “profile on Instagram” to activate this feature. Then type in the profile username. The user can choose to download the profile picture which will then automatically be saved in the project folder.

```python
from newsapi import NewsApiClient

def get_news(topic):
    my_api_key = "64f5513915fa4de9bfc78adcfaa60495"

    news_api = NewsApiClient(api_key=my_api_key)

    data = news_api.get_everything(q=topic, language='en', page_size=20)

    # print(data['articles'][0]['title'])

    list_of_articles = []


    for i in range(15):
        new_dic = {'Title': data['articles'][i]['title'], 'Content': data['articles'][i]['description']}
        list_of_articles.append(new_dic)

    # print(list_of_articles)

    list_final = []

    for i in list_of_articles:
        listA = []
        listA.append(i['Title'])
        listA.append(i['Content'])
        list_final.append(listA)

    return list_final
```

Figure 3.2: Screenshot of python code to utilize the News API to crawl the web and get latest news articles

3.2.2 Method Used

To achieve this functionality, we used the “instaloader” library. By using the ”downloadProfile” method in the ”instaloader” class we were able able to open the instagram account of a particular person and then upon our command we can download that person’s profile picture.

3.3 Badminton Game using Computer Vision

Computer vision is an interdisciplinary scientific field that deals with how computers can gain high-level understanding from digital images or videos.

We utilised this principle to create a simple badminton game that the user can play in his/her free time.

```
elif "instagram profile" in self.query or "profile on instagram" in self.query:
    speak( "Please enter the user name correctly")_I
    name = input("Enter username here:")
    webbrowser.open(f"www.instagram.com/{name}")
    speak(f"Here is the profile of the user {name}")
    time. sleep(5)
    speak("Would you like to download profile picture of this account.")
    condition = takecommand().lower()
    if "yes" in condition:
        mod = instaloader.Instaloader()              #pip install instadownloader
        mod. download_profile(name, profile_pic_only=True)
        speak("The profile picture has been saved in our main folder. now i am ready
    else:
        pass
```

Figure 3.3: Screenshot of python code to search Instagram Profile of a particular id and download the profile picture

3.3.1 How to Activate

The Badminton Game can be accessed from inside the Game Menu. Which can be activated by saying the following sentences

1. I want to play some games
2. Lets play a game
3. I want to play some games

3.3.2 Method Used

To achieve Badminton, we have used libraries are –

1. NumPy
2. OpenCV
3. imutils
4. random

The basic principle behind the badminton game is it detects any red item in the video frame, contours are drawn and the resultant rectangle acts as a badminton racquet.

We can detect the movement of the rectangle by mapping out the if there is any movement in the resultant rectangle between two consecutive frames.

These movements can then be utilised in the game to detect ball contact with the racquet.

```
while True:
    _, frame = cap.read()
    frame = cv2.flip(frame, 1)
    frame = imutils.resize(frame, WIN_HEIGHT, WIN_WIDTH)

    if (isStarted == True):
        if (isOver == False):
            cv2.putText(frame, 'SCORE: ' + str(score), (50, 50), cv2.FONT_HERSHEY_SI
                        cv2.LINE_AA)
        elif (isOver == True):
            cv2.putText(frame, '<- GAME OVER ->', (320, 350), cv2.FONT_HERSHEY_SIMPL
            return "Game Over"

        Ball(frame, ballPosX, ballPosY)

        hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

        lowred = np.array([131, 90, 106])
        highred = np.array([255, 255, 255])
        red_mask = cv2.inRange(hsv, lowred, highred)

        contours, hierachy = cv2.findContours(red_mask, cv2.RETR_TREE, cv2.CHAIN_APP
        contours = sorted(contours, key=lambda x: cv2.contourArea(x), reverse=True)

        for cnt in contours:
            # collecting the starting x,y and width, height values from the bounding
            (x, y, w, h) = cv2.boundingRect(cnt)
            break
```

Figure 3.4: Partial screenshot of python code to play badminton game

3.4 Voice Operated Stopwatch

Project Zenith can activate a voice operated stopwatch, that will start or stop recording the time when the user says so and return the time duration between the two commands.

3.4.1 How to Activate

To activate stopwatch you can say the following sentences

1. Lets have a stopwatch
2. Moto can you start the stopwatch
3. Activate Stopwatch etc

The base of the commands is to have the word stopwatch in it

3.4.2 Method Used

To achieve this we needed to use the time library along with several other functionalities like speech recognition and text-to-speech converter to record and announce the recorded time.

Rest of the code is pretty simple to understand as we just used while loops to wait for the user to say either “Start” or “Stop” to control the stopwatch.

3.4.3 Pitfalls

Sometimes it can take a few buffer seconds to recognize the start and stop commands and execute them.

3.5 Voice Operated Reminder

Project Zenith can activate a voice operated reminder, that will remind the user of a particular task that he/she needs to do.

3.5.1 How to Activate

To activate reminder you can say the following sentences

1. Lets have a reminder
2. Moto can you take a reminder
3. Activate reminder etc

The base of the commands is to have the word reminder in it

```
query = ""
while 'start' not in query:
    query = takecommand()

speak("Stop Watch Started")

if 'start' in query:
    start_time = time.time()


query1 = ""
while 'stop' not in query1:
    query1 = takecommand()

if 'stop' in query1:
    end_time = time.time()


def time_convert(sec):
    mins = sec // 60
    sec = sec % 60
    hours = mins // 60
    mins = mins % 60
    speak("Stopwatch Ended")
    speak("Time Elapsed was {0} hours {1} minutes and {2} seconds".format(int(hours),int(mins),int(sec)))
    print("Time Lapsed = {0}:{1}:{2}".format(int(hours),int(mins),sec))


time_lapsed = end_time - start_time
time_convert(time_lapsed)
```

Figure 3.5: Partial screenshot of python code to control stopwatch

3.5.2 Method Used

To achieve this we needed to use the time and datetime library along with several other functionalities like speech recognition and text-to-speech converter to take commands like time and message to remind

To capture the reminders permanently even after the code stopped running, we used our knowledge about working with files python to create a text file to store our reminders.

The rest of the code is quite straight forward to understand. Although we got stuck several times and online references were used in order to remove the bugs in our program.

3.5.3 Challenges Faced

Because of varying degrees of speech recognition and its interpretation by the bot, we used a lot of string processing and also recursion techniques to accurately identify what the user is saying and convert that into the time format that we required.

3.5.4 Pitfalls

While the reminder is running the AI bot cannot work as we could not run the reminder function in parallel with the main script.

```
def take_time():
    speak("What time should I set the reminder for")
    time = takecommand()
    if time == "none":
        return take_time()
    if ":" in time:
        temp_list = time.split(":")
        hour = temp_list[0]
        min = temp_list[1]
    elif len(time) == 3:
        hour = time[0]
        min = time[1:]
    elif len(time) == 4:
        hour = time[0:2]
        min = time[2:]
    else:
        speak("I didnt quite catch that, Can you repeat the time")
        return take_time()

    return hour, min

hour, min = take_time()


time_final = hour + ":" + min + ":" + "00"
```

Figure 3.6: Screenshot of python code to perform string processing in order to get exact time

3.6 Spam Bot

Spam Bot is a fun functionality of Project Zenith, that can continuously send text messages to any particular node, using any platform available online. For Example WhatsApp, Facebook Messenger, Instagram Dm or even Microsoft Teams.

3.6.1 How to Activate

To activate spam bot you can say the following sentences

1. Moto Activate spam bot
2. Moto Lets spam someone
3. Lets perform some spamming on whatsapp

The base of the commands is to have the word ”spam” or ”spamming” in it

3.6.2 Method Used

To achieve Spam-bot, we used the “pyautogui” library along with the ”time” library.

The basic principle behind the spam-bot is that it utilizes ”PyAutoGui” library to continuously paste a particular text into that chat section and presses the enter key to send that message.

3.7 Text to Handwriting Converter

Due to the world wide COVID-19 pandemic, students all around the world are attending online classes, and are often asked to submit handwritten assignments online so that the teacher knows that the student is not just copy-pasting from Wikipedia and submitting it.

Well technology is here to make our lives easier, so that is why we found out the best method to overcome that problem by converting any picture in the form of handwritten text.

3.7.1 How to Activate

To activate the text to handwriting conversion you can say the following sentences

1. I want to convert text into handwriting
2. Can you convert this into my handwriting
3. Text to handwriting

The base of the commands is to have the word handwriting in it

3.7.2 Method Used

To achieve this result we used a very popular library called "PyWhatKit". As the name suggest, this library was initially made to automate WhatsApp messages but upon our own research into this library revealed that this also has a functionality that can convert text into another picture of handwritten text.

But to obtain that text from an image file, we used the "pytesseract" library which is basically an OCR library that can identify the text from an image and we supplied that text output as the input for the pywhatkit library function. We referenced the boilerplate code available in the documentation and managed to achieve the result that we were expecting.

The resulting file is then saved in the project directory.

3.7.3 Pitfalls

Taking file name input as a voice command was inaccurate, so the user need to get away from the GUI and type the name of the image file into the project terminal.

3.8 E-mail Automation

Project Zenith can send Electronic Mails(E-Mail) to

- Single Person
- Multiple People

by using voice commands that automate the entire process of typing a mail and sending it to a person.

3.8.1 How to Activate

To activate the text to handwriting conversion you can say the following sentences

1. I want to send email
2. Can you send an email
3. Activate email service

The base of the commands is to have the word email in it

3.8.2 Method Used

To achieve Email Automation, we used the “smtplib” library, that takes our email account credentials as an input parameter for the ’login’ method and then utilizes the ’sendmail’ method to send emails to a list of email addresses specified in the form of a python list data type.

3.9 WhatsApp Automation

By using the WhatsApp Automation service, one can easily send WhatsApp messages to a particular person over the internet by using voice commands.

It is specially useful in a scenario where while working on a particular project, someone needs to send a WhatsApp message without getting distracted from his/her work.

3.9.1 How to Activate

To activate the WhatsApp Automation service you can say the following sentences

1. I want to send a whatsapp
2. Hey Moto, can you send a whatsapp message
3. message someone on whatsapp

The base of the commands is to have the word whatsapp or the phrase ”send a whatsapp message” or ”whatsapp message” in the user query.

3.9.2 Method Used

To achieve this result we used a very popular library called “PyWhatKit”. As the name suggest, this library is made to automate WhatsApp messages, and it does so by opening up whatsapp web and sending the message that is directed towards a particular whatsapp user.

The ’sendwhatmsg’ method of the PyWhatKit class is utilised in order to achieve the expected results. Since the ’sendwhatmsg’ method takes the mobile number as an input, we decided to extract the phone numbers of people by using their names. So we hard-coded some number and name values in the code. These can be later shifted to an online database, from which these values can then be fetched.

3.9.3 Pitfalls

It takes nearly one minute time for the WhatsApp web to open and another 20 seconds for the message to be sent to the directed user.

3.10 Instagram Automation

As programmers, we hardly have time to maintain our social media presence and often lack in the number of good connections via social media accounts that can help our career. Well we sought out to solve this problem by automating your Instagram account. Project Zenith can not just log in to your account but can also like and comment on other people’s posts, so that your social life can grow without you even knowing about it.

3.10.1 How to Activate

To activate the Instagram Automation service you can say the following sentences.

1. Activate Instagram Automation
2. Can you Automate my Instagram
3. Automate Instagram

3.10.2 Method Used

To achieve this result we used a library called "Instapy". Instapy is a very unique python library that makes use of Selenium to automate browser and use that to access Instagram and perform actions on Instagram, such as pressing the link button, commenting etc.

We referred the instapy documentation to get to know about multiple parameters that we can change in the code, in order to avoid linking the wrong photos or commenting something that might not be appropriate on that post.

Overall the code is simple to understanding and because of heavy use of this library we were able to find ways to avoid bugs that were hampering with the smooth functioning of the program.

3.11 Zero Width Fingerprinting

Zero Width Fingerprinting, basically uses the zero – width – characters used in Unicode to encode a secret message within a public message.

After encoding this private message into any public message, the private message can only be accessed by someone who knows that there is a secret message inside that message.

Reference to an Article by Akshat Rastogi (Co-Author) on Zero-Width-Fingerprinting can be found from the link below.

('https://medium.com/@akshat28vivek/zero-width-fingerprinting-an-analysis-beyond-technology-22cc4f059384')

3.11.1 How to Activate

To activate the zero width fingerprinting you can say the following sentences.

1. Activate Zero Width
2. I want Text Encryption
3. How do I encrypt my text ?

3.11.2 Method Used

To achieve this result we used Selenium library in python. We used selenium to automate the browser and go to the website (https://neatnik.net/steganographr/) and pasted the user input into the respective columns, pressed the "stenographize" button and then selected and copied the resulting text from that browser.

Selenium is a relatively easy library to use, with easy to interpret commands and ease of programming.

3.11.3 Challenges Faced

On the internet resources, we faced some problems locating and implementing the browser extensions in the program and every time we started the program, it crashed because of those browser extensions only. So we had to research a lot using a lot of different sources until we finally came to a solution where we could download the latest version of these browser extension every couple of times that we run the program.

We used GeckoDriver that is basically made for Firefox Browser, because Chrome takes up a lot of system resources to start, and Firefox started relatively fast.

3.12 Sorting Algorithm Visualizer

As passionate students of computer science, with the zest to change the world, it is important for us that we understand the concepts better by making use of the technology.

That is why we built, a sorting algorithm visualizer in python, so as to visualize various sorting algorithms

3.12.1 How to Activate

To activate the sorting algorithm visualizer you can say the following sentences

1. I want to visualize sorting
2. How do I learn Sorting
3. Activate Sorting Visualizer

3.12.2 Method Used

To achieve this result we used TKinter Library to design a GUI, where we can currently test 2 sorting algorithms. Bubble Sort and Quick Sort

3.12.2.1 Bubble Sort

Bubble Sort is a sorting algorithm where we iterate through the loop at maximum of n-1 times (n is the number of items in the loop) and every time we compare 2 consecutive items in the array, if they are out of order we swap them in the right order and then we move on to the next two consecutive items.

3.12.2.2 Quick Sort

Quick Sort is one of the fastest sorting algorithms out there. The algorithm is basically a recursive type of algorithm which relies on a pivot value and partitioning of the array into left and right where the left half consists of values that are lower than the pivot value and the right half consists of values that are greater than the pivot value.

3.13 Spotify Automation

No Doubt, Music is one of the few factors in a student/developer's life that can increase their productivity by approximately 3-5 times, and since Spotify is one of the most used music streaming platform, we decided to automate spotify through voice commands.

3.13.1 How to Activate

To activate Spotify service you can say the following sentences

1. I want to listen to music on Spotify
2. Hey Moto, Can you open Spotify
3. Spotify will be great at this point of time

The base of the commands is to have the word spotify in the user query.

3.13.2 Method Used

To achieve Spotify Automation, the following libraries were used

1. spotipy
2. os
3. json
4. time

The basic principle behind the Spotify Automation is to utilize the Spotify Developers API offered by Spotify AB Ltd.

After registering with Spotify Developers API, we were provided with a few authentication codes(API-Key, Secret-Key etc). These keys are then required to use spotify API services in python.

Currently, when query is raised to open Spotify, Spotify automatically starts playing music from pre-defined artists hard-coded in the spotify module.

3.14 Shopping Automation

Project Zenith can help you to shop online, specifically on Amazon. The current shopping automation is pretty simple, where our voice assistant can log into the user's amazon account and search for a product on Amazon, but in the future this can further be expanded to select, cart and order the best item for that specific search query.

3.14.1 How to Activate

To active the Shopping Automation Service you can say the following sentences

1. I want to go shopping
2. Search something on amazon
3. Activate shopping on amazon

The base of the commands is to have the word shopping or amazon in the user query.

3.14.2 Method Used

To achieve this result we used Selenium Library to Automate the browser into logging on to amazon and searching for a particular product there.

3.14.3 Challenges Faced

So logging into amazon is quite tricky because on the amazon website, in order to quickly login the user needed to hover over a menu item and then click the login button. Upon researching, we came across Action Chains in Selenium where the hovering action can be achieved automatically.

The entire code is pretty basic and easy to understand as shown in figure 3.15 & figure 3.16.

3.15 Security Camera

While you are away, you can ask our voice assistant to monitor room activity. This feature is designed to protect your privacy in the work environment. Often times, when students/developers have to somewhere for a short time, shutting down your PC, or even putting it to sleep doesn't make a lot of sense as affecting the state of your system actively affects all background processes happening on the local machine.

If there is any movement detected, a siren is raised making the owner aware about a potential threat to privacy

3.15.1 How to Activate

To active the Super Security Service you can say the following sentences

1. Turn on security mode
2. Activate super security mode
3. Maintain my device security while I am away

The base of the commands is to have the word security in the user query.

3.15.2 Method Used

To achieve this result we used OpenCV and PyGame Libraries to capture video frames and identify the absolute different between two consecutive video frames. This differences are then mapped out in terms of contours and rectangles.

If these rectangles are visible, that means that there is a movement in the video frame, and that triggers the alarm sound.

3.16 Generating Random Quotes

As students/developers, we are in constant need of motivation and nothing works better than powerful quotes.That is why our voice assistant can recite a few quotes whenever the user is in some need of motivation.

3.16.1 How to Activate

To active the Random Quote Generator you can say the following sentences

1. Tell me a quote
2. I want to listen to some quotes
3. Hey Moto, Motivate me

The base of the commands is to have the word quote, quotes or motivation in the user query.

3.16.2 Method Used

To implement this functionality, we used the "random" library.

The basic principle behind the random library is that it take out a random quote from quotes.txt where 300+ quotes are stored along with the name of the person that said the quote. These quotes are selectively picked to enhance productivity of an individual and motivate them to keep working without giving up.

3.16.3 Challenges Faced

In order to sound fluent, we needed to separate the author name from the quote, which involved a little bit of string processing as shown in figure 3.19

3.17 Generating Random Quotes

As students/developers, we are in constant need of motivation and nothing works better than powerful quotes.That is why our voice assistant can recite a few quotes whenever the user is in some need of motivation.

3.17.1 How to Activate

To active the Random Quote Generator you can say the following sentences

1. Tell me a quote
2. I want to listen to some quotes
3. Hey Moto, Motivate me

The base of the commands is to have the word quote, quotes or motivation in the user query.

3.17.2 Method Used

To implement this functionality, we used the "random" library.

The basic principle behind the random library is that it take out a random quote from quotes.txt where 300+ quotes are stored along with the name of the person that said the quote. These quotes are selectively picked to enhance productivity of an individual and motivate them to keep working without giving up.

3.17.3 Challenges Faced

In order to sound fluent, we needed to separate the author name from the quote, which involved a little bit of string processing as shown in figure 3.19

3.18 Reading E-mails

Our voice assistant, can read emails received in the user email account. Currently this service only allows the voice assistant to read the emails, when asked to do so by the user. But in the future, we are trying to integrate a system such that the email will be read as soon as it comes in the account inbox.

3.18.1 How to Activate

To active the Email Reading Service you can say the following sentences

1. Hey Moto, Read my emails
2. Can you read some of my email
3. Activate email read

The base of the commands is to have the word ”email” and ”read” or the phrase ”read emails” in the user query.

3.18.2 Method Used

To implement this functionality, we used the following libraries.

1. imaplib
2. email
3. traceback

To achieve Email Reading functionality, we had to give our email account credentials as an input parameter for the 'login' method and then utilizes the 'fetch' method to fetch all of emails in the specified account and return them as a python tuple data-type.

The required values can then be fetched from these tuples and stored in the form of a nested list. This nested list is returned our voice assistant speaks the data from these emails and produces a high pitched short sound at the end to mark the end of the email message data.

3.18.3 Challenges Faced

In order to sound fluent, we needed to separate the email title from the returned ”emailfrom” variable which involved a little bit of string processing as shown in figure 3.21

3.19 Voice Operated Calculator

Project Zenith has an included functionality that can use voice commands to calculate mathematical queries of the user.

3.19.1 How to Activate

To active the Voice Calculator Service you can say the following sentences

1. Activate Voice Calculator
2. Can you open up calculator
3. Calculate user-query

There are basically two ways of calculating using voice commands

- Instantaneous Calculation: Here the user can use the first word as ”Calculate” and add any query at the end to get the result of that query.
- Non-Instantaneous Calculation: The user can open up voice calculator, and then when asked upon can specify the query whose solution needs to be found out.

3.19.2 Method Used

To implement this functionality, we used the Wolfram Alpha API. This is a developer friendly service that aims to provide answers to mathematical queries of varying degrees of voice commands. So weather the query is of the form

- 60 + 9 or
- sixty plus 9

The API will return the same result in both of the above cases.

3.19.3 Challenges Faced

Wolfram Alpha API takes a lot of time to get activated and produced some bugs in the initial testing when we tested the calculator immediately after creating the API from 'https://developer.wolframalpha.com/'. After almost a day, the API started working fine.

The code for performing these calculations is shown in Figure 3.22

```python
def Reminder(timex, reminder_txt):
    def todaysDate():
        daily_clock = time.localtime()
        daily_date = time.strftime("%a, %b %d %Y", daily_clock)
        data = "----------------\n" + daily_date + "\n----------------"
        with open("history_reminder.txt", 'a') as output:
            output.write("\n" + data + '\n')
            output.close()
            return Alarm()

    def Alarm():
        global listA
        end_time = time.time()
        time_evolved = end_time - start_time
        print(time_evolved)
        end_time1 = timex
        reminder = reminder_txt
        all = "Time: " + end_time1 + " Your Reminder is: " + reminder
        listA.append(all)
        with open("history_reminder.txt", 'a') as output:
            output.write(all + '\n')
        speak("Reminder Set. Commencing Count Down")
        its_time = False
        while its_time == False:
            its_time = False
            clock = time.localtime()  # get struct_time
            Timer = time.strftime("%a, %b %d %Y, %X%p (%Z)", clock)
            if end_time1 in Timer:
                its_time = True
                # print("Your Reminder is:\n(", reminder, ")")
                # string2 = "He he he he haas dele rinkiya ke papa he he he he "
```

Figure 3.7: Partial screenshot of python code to control Reminder

```python
import pyautogui
import time

def spam(text):
    while True:
        # time.sleep(1)I CAN DO THIS ALL DAY
        pyautogui.typewrite(text)
        pyautogui.press('enter')
```

Figure 3.8: Screenshot of python code to perform text spamming on any online platform

```python
import pywhatkit
import pyautogui
from PIL import Image
import pytesseract as pyt


def text_to_handwriting(file_name):
    pyt.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract'
    print('[Info]: Starting Text Conversion')
    text = pyt.image_to_string(file_name)
    a = (str(text))
    pywhatkit.text_to_handwriting(a,rgb=[0,0,0])
    print("[Info]: Conversion Complete")
```

Figure 3.9: Screenshot of python code to perform text to handwriting conversion

```python
import smtplib


def send_mail(list_emails, message):
    server=smtplib.SMTP('smtp.gmail.com',587)
    server.starttls()
    server.login("project.zenith11@gmail.com","testing@123")
    server.sendmail('project.zenith11@gmail.com',list_emails,message)
```

Figure 3.10: Screenshot of python code to perform E-mail Automation

```python
elif "send a whatsapp message" in self.query or "whatsapp message" in self.query or "what
    phone = ""
    speak("Who do you want to send a message sir")
    name = self.takecommand().lower()
    while True:
        yes_or_no = "no"
        if name not in ['samarth', 'saumya', 'soumya', 'soma']:
            speak("The name you said is not in my database")
            speak("Do you want me to send message to an unknown number ?")
            yes_or_no = self.takecommand().lower()

        if "yes" in yes_or_no:
            speak("Please say the number")
            phone = self.takecommand().lower()
            break
        else:
            break

    if name.lower() == "samarth":
        phone = "7300480752"
    elif name.lower() == "saumya" or name.lower() == "soma" or name.lower() == "soumya":
        phone = "9999408980"
    # elif len(phone) < 12:
    # speak("The number you gave was invalid, You are such a fool YOU puny human")

    if len(phone) >= 10:
        speak("What is the message sir ?")
        message = self.takecommand().lower()
        hour = int(datetime.datetime.now().hour)
        minu = int(datetime.datetime.now().minute) + 1
```

Figure 3.11: Partial screenshot of python code to perform WhatsApp Automation

```
# if you want to run this script on a server,
# simply add nogui=True to the InstaPy() constructor
session = InstaPy(username=insta_username, password=insta_password)
session.login()
# set up all the settings
session.set_relationship_bounds(enabled=True,
                 potency_ratio=-1.21,
                  delimit_by_numbers=True,
                    max_followers=4590,
                     max_following=5555,
                      min_followers=45,
                       min_following=77)
session.set_do_comment(True, percentage=10)
session.set_comments(['aMEIzing!', 'So much fun!!', 'Nicey!'])
session.set_dont_include(['friend1', 'friend2', 'friend3'])
session.set_dont_like(['pizza', 'girl'])
# do the actual liking
session.like_by_tags(['natgeo', 'world'], amount=100)
# end the bot session
session.end()
```

Figure 3.12: Partial screenshot of python code to perform Instagram Automation

```python
def get_zwf_text(public_message, private_message):
    wsh = comctl.Dispatch("WScript.Shell")
    # ChromeDriverManager().install()
    opts = webdriver.FirefoxOptions()
    opts.headless = False
    driver = webdriver.Firefox(executable_path=GeckoDriverManager().install(), options=opts)
    driver.get('https://neatnik.net/steganographr/')
    public = driver.find_element_by_xpath('//*[@id="public"]')
    public.send_keys(public_message)
    private = driver.find_element_by_xpath('//*[@id="private"]')
    private.send_keys(private_message)
    time.sleep(1)
    drive=driver.find_element_by_xpath("/html/body/main/form[1]/fieldset/p[4]/button")
    drive.click()
    time.sleep(2)
    elem = driver.find_element_by_xpath('/html/body/main/section/textarea')
    elem.send_keys("p")
    elem.send_keys(Keys.CONTROL, 'a') #highlight all in box
    elem.send_keys(Keys.CONTROL, 'c') #copy
    print("Text has been copied to clipboard")
    driver.close()
```

Figure 3.13: Partial screenshot of python code to perform Zero Width Fingerprinting

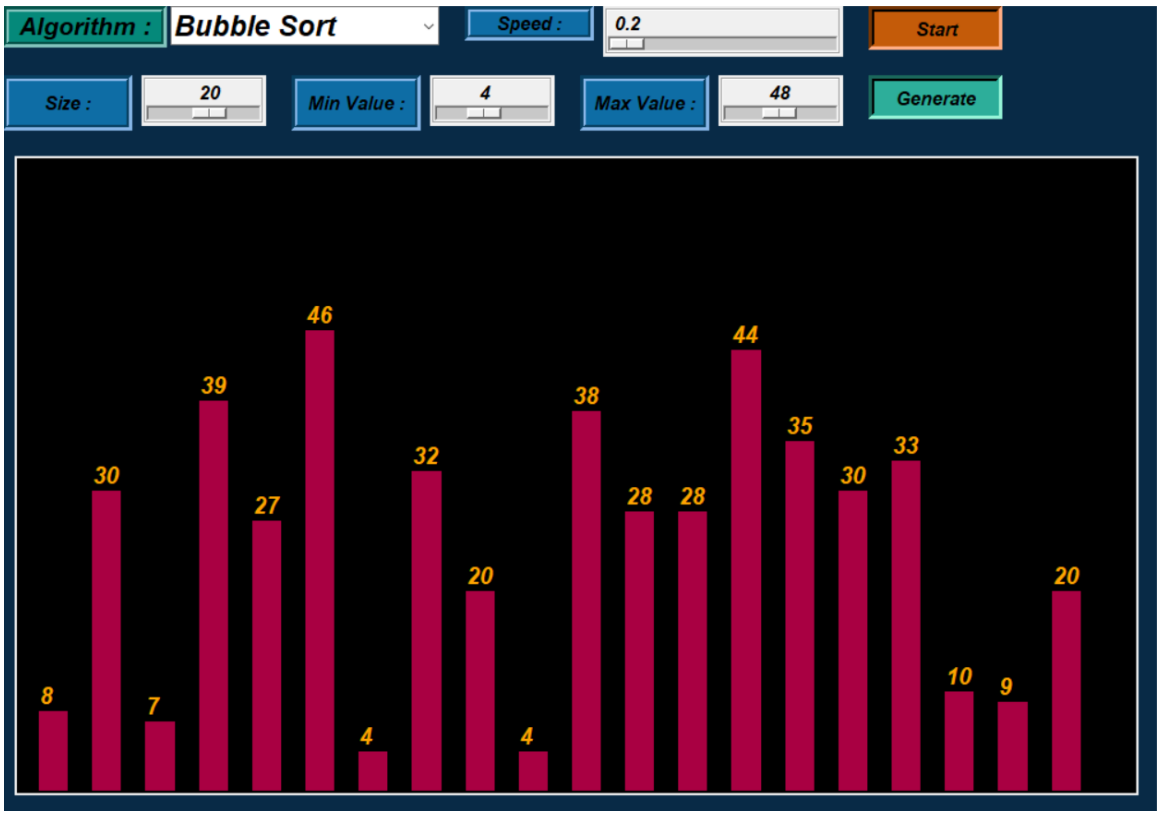

Figure 3.14: Screenshot of Graphical User Inteface for the Sorting Algorithm Visualizer

```
def spotipy_lol(artist_name):
    try:
        token = util.prompt_for_user_token(username, scope)
    except (AttributeError, JSONDecodeError):
        os.remove(f".cache-{username}")
        token = util.prompt_for_user_token(username, scope)

    spotifyObject = spotipy.Spotify(auth=token)

    devices = spotifyObject.devices()
    print(json.dumps(devices, sort_keys=True, indent=4))
    deviceID = devices['devices'][0]['id']

    # Get track information
    track = spotifyObject.current_user_playing_track()
    print(json.dumps(track, sort_keys=True, indent=4))
    print()
    artist = track['item']['artists'][0]['name']
    track = track['item']['name']

    if artist !="":
        f = open('text_to_print.txt', 'w')
        f.write(str("Currently playing " + artist + " - " + track))
        f.close()

    # User information
    user = spotifyObject.current_user()
    displayName = user['display_name']
    follower = user['followers']['total']
```

Figure 3.15: Partial screenshot of python code used to Automate Spotify

```python
def shopping_automation_lol(search_item):
    wsh = comctl.Dispatch("WScript.Shell")
    # ChromeDriverManager().install()
    opts = webdriver.FirefoxOptions()
    opts.headless = False
    driver = webdriver.Firefox(executable_path=GeckoDriverManager().install(), options=opts)
    action = ActionChains(driver)
    time.sleep(1)

    driver.get('http://www.amazon.in')
    time.sleep(1)

    firstLevelMenu = driver.find_element_by_xpath('//*[@id="nav-link-accountList"]/span/span')
    action.move_to_element(firstLevelMenu).perform()
    time.sleep(1)
```

Figure 3.16: Partial screenshot of python code used to Automate Shopping Service on Amazon

```python
secondLevelMenu = driver.find_element_by_xpath('//*[@id="nav-flyout-ya-signin"]/a/span')
secondLevelMenu.click()
time.sleep(1)

signinelement = driver.find_element_by_xpath('//*[@id="ap_email"]')
signinelement.send_keys("8433098945")
time.sleep(1)

cont = driver.find_element_by_xpath('//*[@id="continue"]')
cont.click()
time.sleep(1)

passwordelement = driver.find_element_by_xpath('//*[@id="ap_password"]')
passwordelement.send_keys("akshat28")
time.sleep(1)

login = driver.find_element_by_xpath('//*[@id="signInSubmit"]')
login.click()
time.sleep(1)
```

Figure 3.17: Partial screenshot of python code used to Automate Shopping Service on Amazon

```python
def security_camera():
    pygame.init()
    pygame.display.set_mode((0, 0), pygame.FULLSCREEN)
    pygame.mixer.init()
    cam = cv2.VideoCapture(0)
    while cam.isOpened():
        ret, frame1 = cam.read()
        ret, frame2 = cam.read()

        diff = cv2.absdiff(frame1, frame2)
        grey = cv2.cvtColor(diff, cv2.COLOR_RGB2GRAY)
        blur = cv2.GaussianBlur(grey, (5,5), 0)
        _, thresh = cv2.threshold(blur, 20, 255, cv2.THRESH_BINARY)
        dilated = cv2.dilate(thresh, None, iterations=3)
        contours, _ = cv2.findContours(dilated, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
        for c in contours:
            if cv2.contourArea(c) < 5000:
                continue
            x,y,w,h = cv2.boundingRect(c)
            cv2.rectangle(frame1, (x,y), (x+w, y+h), (0,255,0), 2)
            winsound.Beep(5000, 50)
            winsound.PlaySound('alert.wav', winsound.SND_ASYNC)
        cv2.imshow("Security Camera", frame1)
        if cv2.waitKey(10) == ord('q'):
            break
```

Figure 3.18: Screenshot of python code used to activate Super Security Mode

```python
def security_camera():
    pygame.init()
    pygame.display.set_mode((0, 0), pygame.FULLSCREEN)
    pygame.mixer.init()
    cam = cv2.VideoCapture(0)
    while cam.isOpened():
        ret, frame1 = cam.read()
        ret, frame2 = cam.read()

        diff = cv2.absdiff(frame1, frame2)
        grey = cv2.cvtColor(diff, cv2.COLOR_RGB2GRAY)
        blur = cv2.GaussianBlur(grey, (5,5), 0)
        _, thresh = cv2.threshold(blur, 20, 255, cv2.THRESH_BINARY)
        dilated = cv2.dilate(thresh, None, iterations=3)
        contours, _ = cv2.findContours(dilated, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
        for c in contours:
            if cv2.contourArea(c) < 5000:
                continue
            x,y,w,h = cv2.boundingRect(c)
            cv2.rectangle(frame1, (x,y), (x+w, y+h), (0,255,0), 2)
            winsound.Beep(5000, 50)
            winsound.PlaySound('alert.wav', winsound.SND_ASYNC)
        cv2.imshow("Security Camera", frame1)
        if cv2.waitKey(10) == ord('q'):
            break
```

Figure 3.19: Screenshot of python code used to generate Random Quotes

```python
def get_quote():

    quote_file = os.path.join(os.path.dirname(os.path.realpath(__file__)), 'quotes.txt')
    f = open(quote_file, 'r')
    txt = f.read()
    lines = txt.split('\n.\n')

    quote_raw = random.choice(lines)
    listA = quote_raw.split('"')
    listB = []
    for items in listA:
        if items != "":
            for i in range(items.count("\n")):
                items = items.replace('\n', '')
            for i in range(items.count("--")):
                items = items.replace('--', '')
            listB.append(items)

    return listB[0], listB[1]

# quote, author = get_quote()
# print(type(author))
```

Figure 3.20: Screenshot of python code used to generate Random Quotes

```python
def read_email_from_gmail():
    list_of_emails = []
    try:
        mail = imaplib.IMAP4_SSL(SMTP_SERVER)
        mail.login(FROM_EMAIL,FROM_PWD)
        mail.select('inbox')

        data = mail.search(None, 'ALL')
        mail_ids = data[1]
        id_list = mail_ids[0].split()
        first_email_id = int(id_list[0])
        latest_email_id = int(id_list[-1])

        for i in range(latest_email_id,first_email_id, -1):
            data = mail.fetch(str(i), '(RFC822)' )
            for response_part in data:
                arr = response_part[0]
                if isinstance(arr, tuple):
                    msg = email.message_from_string(str(arr[1],'utf-8'))
                    email_subject = msg['subject']
                    email_from = msg['from']
                    x = ""
                    for i in email_from:
                        if i == "<":
                            break
                        else:
                            x += i
                    email_from = x
                    try:
                        email_from = email_from.split('"')[1::2][0]
                    except:
                        pass
```

Figure 3.21: Partial screenshot of python code used to read emails from user account

```
import wolframalpha
def calculate(query):
    client = wolframalpha.Client('5KUK33-JGTE9PP7T8')
    res = client.query(query)
    output = next(res.results).text
    return(output)
```

Figure 3.22: Partial screenshot of python code used to perform voice operated calculations on the local machine

Chapter 4

Advanced Functionalities

Now that we have added the intermediate-level functionalities into project zenith, we are ready to move onto phase four of our project and functionalities that

Following is a list of the features that we included in phase four of Project Zenith.

1. Face Recognition Authentication
2. Human Mood Detection
3. Bitcoin Mining
4. Gesture Controlled Social Media
5. Face Mask Recognition
6. Coding Guide
7. Building the GUI

These functionalities are mostly focused on Smart Human Recognition that makes our voice assistant capable to understand user behaviour and helps a long way for an improved user experience.

4.1 Face Recognition Authentication

In Project Zenith, facial recognition is used to authenticate the identity of the user and to customize it's responses accordingly.

4.1.1 How to Activate

The Facial Recognition feature get activated every time the bot starts. It opens the camera without displaying the camera output to the user and scans the user's face and cross checks that against the database. If the user is present in the database, authorisation is granted to the user otherwise, moto is not activated and the program closes.

4.1.2 Method Used

To achieve Facial Recognition, we used the "face-recognition" library along with several other libraries like OpenCV, OS library, Math library and Numpy.

The basic principle behind the face-recognition library is that it basically converts the images stores in its database into some form of encoding, and then with the help of OpenCV, we start the video capture and every frame in that video capture is then returned and again converted into encoding by facial-recognition library.

These encoding are then compared and the encoding with the accuracy percentage of more than 80 is then recognised and the person's name is returned upon which our voice assitant then verifies and authorises the person.

To keep the verification accurate and short timed, we run the facial-recognition 15 times accounting for 15 frames in total, the returned names are then stored in a list and the name that is repeated for the maximum amount of time is then finally returned to Project Zenith.

The python code required to achive this result can be divided into the following parts:

1. Initialising framework and reading the files from database.(Figure 4.1)
2. Running the while loop multiple times and comparing the encoding (Figure 4.2)
3. Returning the name that appeared maximum number of times (Figure 4.3)
4. Using Facial Recognition in our voice assistant to Authorize Command Access (Figure 4.4)

```python
def recognizeFace():
    videoCapture = cv2.VideoCapture(0)

    def getAccuracy(faceDistance, faceMatchThreshold=0.6):
        if faceDistance > faceMatchThreshold:
            range = (1.0 - faceMatchThreshold)
            linearValue = (1.0 - faceDistance) / (range * 2.0)
            return linearValue
        else:
            range = faceMatchThreshold
            linearValue = 1.0 - (faceDistance / (range * 2.0))
            return linearValue + ((1.0 - linearValue) * math.pow((linearValue - 0.5) * 2, 0.2))


    # FOR GETTING PATH AND NAME
    allPaths = os.listdir(r"C:\Users\Akshat\Documents\GitHub\Project-ASS\face_recognition_data")
    allNames = []
    allRegNumbers = []
    allEncodings = []
    for index in range(len(allPaths)):
        allNames.append(allPaths[index].split(".")[0])
        allRegNumbers.append(allPaths[index].split(".")[1])
        image = face_recognition.load_image_file(r"C:\Users\Akshat\Documents\GitHub\Project-ASS\face_recognition_data/" + allPaths[index])
        temp = face_recognition.face_encodings(image)[0]
        allEncodings.append(temp)

    final_names = []
```

Figure 4.1: Screenshot of python code to initialize framework and to read the files from database

4.2 Human Mood Detection

It is often said that robots don't have emotions, while that technically stands true to this day, we are progressing towards an age where robots can now recognize human emotions and respond to those emotions accordingly. Using Computer Vision and Machine Learning, our bot can recognize human emotions and act as a friend to help us understand what we are actually feeling.

4.2.1 How to Activate

To active the news crawler you can say the following sentences

1. Recognize my mood
2. I am not in mood
3. I am in an uncertain mood today

The base of the commands is to have the word mood in it

4.2.2 Method Used

To achieve this result we used a library called “DeepFace” along with several other libraries like ”imUtils” and ”Numpy”. As the name suggest, the ”DeepFace” library uses deep learning to recognize human mood detection. We use the ”imUtils” to start Video Capturing, then we read the image from the video stream and supply the image frame into the ”DeepFace” function which returns the dominant emotion detected in that frame.

Same as facial recognition, we refrain from showing the video stream directly to the user so that the resultant emotions cannot be tampered. We run the code 15 times to get an accurate result and return the emotion value that was used the most.

After the mood value is returned, we can perform functions based on that mood

4.2.3 Pitfalls

Mood Detection can be sometimes inaccurate under improper lighting conditions

4.3 Bitcoin Mining

Bitcoin offers an efficient means of transferring money over the internet and is controlled by a decentralized network with a transparent set of rules, thus presenting an alternative to central bank-controlled fiat money.

We wanted to get in on the action and find a way to mine bitcoins and integrate it in out voice bot

4.3.1 How to Activate

To active the Bitcoin mining service of Project Zenith you can say the following sentences

1. Activate bitcoin mining
2. Can you mine bitcoins

The base of the commands is to have the phrase ”mine bitcoin” or ”bitcoin mining” in user query.

4.3.2 Method Used

Mining bitcoins in itself sounds complicated, but upon our research we found out that it really isn't.

Bitcoin is all about guessing!! Let me explain Bitcoin is based on a block-chain model of technology which uses blocks of bitcoin transactions like a ledger, the only difference is that these blocks also contain their unique hash code to identify themselves and also the hash codes of the previous block. These hashcodes use SHA256 encryption method and every hash code is generated from the information contained within the block. Now one unique property about these hashes is that they contain certain number of zero's in the start and we call them the difficulty.

Now lets suppose that we have been given an equation:

x + y = 10

The values of x and y can be guessed but it is difficult to guess the exact value of these variables that was used.

We need to do the same thing with the SHA256 hash codes of these bitcoin blocks. We introduce an outside variable called nonce into the bitcoin block. Now all we need to do is to guess this nonce value such that the resultant hash code generated has the same difficulty as the difficulty of previous hash.

In 2021, if you successfully mine a bitcoin block first, you will get 6.25 bitcoins and every 4 years this reward gets halved. The code that we used, is not difficult algorithmic wise, but bitcoin mining takes a lot of time and computational power, that's why even after being so easy to code, it is not that easy to actually mine a bitcoin.

Now in the code, we used a sample transaction and sample previous hash to perform bitcoin mining to make it easy and not time consuming but real mining can be done with the same script if required. Also in the code the difficulty is currently set to 6, but in 2020 the difficulty level is 20 and it takes a lot of time to mine.

4.4 Gesture Controlled Social Media

We have always fascinated about a super artificial intelligence like Jarvis from Iron Man. An ideal scenario where we imagine this to be used if you were working out with your laptop at a comfortable height and you wanted to scroll through your social media account just to have a good look at it.

4.4.1 How to Activate

To active the Gesture Controlled Social Media controller you can say the following sentences

1. Enable gesture scrolling
2. Activate gesture control

4.4.2 Method Used

To achieve this result, we used 'pyautogui' and 'OpenCV libraries of python. The basic principle behind the feature is that we use OpenCV to detect your hand and then we use pyautogui to enter space every time your hand moves up. This actually scrolls down the social media page to view the next post.

The code uses several OpenCV methods but it is still short simple and easy to understand as shown in figure

4.4.3 Challenges Faced

We actually weren't able to detect just human hands, because the best way we could find was to detect a particular color in the image. So every time we tried to find skin color in the image frame of OpenCV we were also getting the same response with face as well as hands. So we decided to hold something of a particular color in our hand in order to detect its motion.

We tried several colors like red and green but we found less image noise with blue color so we went for it instead.

4.5 Face Mask Detection

The year 2020 has shown mankind some mind-boggling series of events among-est. which the COVID-19 pandemic is the most life-changing event which has startled the world since the year began. Affecting the health and lives of masses, COVID-19 has called for strict measures to be followed in Order to prevent the spread of disease. From the very basic hygiene standards to the treatments in the hospitals, people are doing all they can for their own and the society's safety; face masks

are one of the personal protective equipment. People wear face masks once they step out of their homes and Authorities strictly ensure that people are wearing face masks while they are in groups and public Places.

So to make people aware about the growing threat of COVID-19, We made a Face Mask Detector, that will check if the user is wearing a mask or not before he/she leaves the system.

4.5.1 How to Activate

To active the Face Mask Detection you can say the following sentences

1. I am going out
2. Activate mask detection

4.5.2 Method Used

To achieve this result the following libraries of python were used:

1. imutils
2. tensorflow
3. numpy
4. OpenCV

The basic principle behind the feature is that at first we need instead of face mask recognition, we need to focus on face detection upon which mask detection will take place. To do that we used the in-built haarcascade_frontalface_alt2.xml model pre-downloaded while installing OpenCV package.

We trained the face detection model using Google's online machine learning platform 'https://teachablemachine.withgoogle.com/'. The resulting template was downloaded as a keras model with a . h5 extension.

The harcascade model is then used to detect faces within a video frame. Once these faces are detected, they are extracted from the image and then they are converted into numpy arrays that are then used as an input for 'predict' method of the Tensorflow.Keras class, which uses the custom keras model referred to in the preceding paragraph.

The custom mask detection model was trained used two classes of datasets

1. with mask
2. without mask

The 'predict' method of the Tensorlow.keras class therefore returns a nested list with the probability values of the two classes. These probability values are then used to determine the status of the frame as in mask or without mask.

Since testing the model only on a single frame can be unreliable and the system may not get enough time to open the camera and properly capture the frame, we repeat the process of mask detection 15 times and return the state that was founded in the majority of these frames. This makes it possible to obtain precise results on the state of the image.

4.5.3 Challenges Faced

We faced a lot of Challenges while trying to implement this functionality. One of the major challenges was with the incompatibility of the model with the code that we previously wrote while trying to reference the documentation.

Some of the other challenges involved us getting inaccurate predictions on our model, if the mask was in a state in which the upper part of the mask went slightly below the person's nostrils.

4.6 Coding Guide

As students of Computer Science and Software Developers have to write a lot of boiler plate code before getting to the actual logic implementation part, that's why we implemented a unique functionality inside Project Zenith, where upon asking, our voice assistant can type a sample boilerplate code in any code editor that the user is working on. This will allow the users to only type the parts that are essential for the working of the specific program.

4.6.1 How to Activate

The activation of the coding guide is dynamic in nature and depends upon the programming language the user is working on and also the type boilerplate code that is required by the user.

Currently only a few boilerplate codes of the following languages have been implemented in the code.

1. Python
2. Java

Support for more programming languages will be added in the future.
A few examples to activate the Coding Guide Functionality are as follows

1. Hey Moto, Give me a class in python
2. Hey Moto, declare an image in android
3. Hey Moto, Give me a function in python etc

4.6.2 Method Used

To achieve this functionality, we used the ”pyperclip” and ”pyautogui” library. The ’copy’ method of the ”pyperclip” library is used to copy a pre-defined sample code onto the clipboard and then the ’hotkey’ method of the pyautogui class is utilised to paste the sample code into the code-editor that the user is working on.

Currenly the following biolerplate codes are available

1. Declaring an image in android studio (java)
2. Declaring an edit text in android studio (java)
3. Declaring an button in android (java)
4. Raising Indents in android (java)
5. Declaring an edit text in android (java)
6. Declaring an edit text in android (xml)

7. Declaring a function in python

8. Declaring a class in python

We are planning to add more of these sample codes of different languages in the future.

4.7 Building the GUI

In order to design the Graphical User Interface (GUI), Qt Designer tool was used. Using its simple drag and drop interface, a GUI interface could be quickly built without having to write the code.

Moreover, Widgets created with Qt Designer integrate seamlessly with programmed code so that you can easily assign behavior to graphical elements. All properties set in Qt Designer can be changed dynamically within the code. It enabled me to drag and drop the required widgets from the widget box on the left pane and also assign value to properties of widget laid on the form.

The code is referenced in figure 4.20

```python
n = 0
while n<=15:
    ret, frame = videoCapture.read()

    frame = cv2.resize(frame, (0, 0), fx=2, fy=1.6)

    resizedFrame = cv2.resize(frame, (0, 0), fx=0.2, fy=0.2)

    requiredFrame = cv2.cvtColor(resizedFrame, cv2.COLOR_BGR2RGB)

    faceLocation = face_recognition.face_locations(requiredFrame)

    faceEncoding = face_recognition.face_encodings(requiredFrame, faceLocation)

    faceNames = []
    for encoding in faceEncoding:

        ismatched = face_recognition.compare_faces(allEncodings, encoding)
        matchedName = "Unknown"

        faceDistance = face_recognition.face_distance(allEncodings, encoding)

        if faceDistance[0] > faceDistance[1]:
            minimumFaceDistance = faceDistance[1]
        else:
            minimumFaceDistance = faceDistance[0]

        accuracy = getAccuracy(minimumFaceDistance) * 100

        bestMatchIndex = numpy.argmin(faceDistance)
```

Figure 4.2: Screenshot of python code used for Running the while loop multiple times and comparing the encoding

```
        if ismatched[bestMatchIndex] and accuracy > 80:
            matchedName = allNames[bestMatchIndex]


        faceNames.append(matchedName)
        final_names.append(matchedName)

    ...
    n += 1
    ...
most = max(set(final_names), key=final_names.count)
return most
```

Figure 4.3: Screenshot of python code used for returning the name that appeared maximum number of times

```
speak2("Starting Moto")
speak2("Authorising Command Access by Facial Recognition")
speak2("Please look in the camera")
identity = ""
try:
    identity = face_recognition_moto.recognizeFace()
except:
    speak2("Your Identity could not be verified")

if identity not in ["AkshatRastogi", "SamarthGoyal", "SaumyaTripathi"]:
    speak2("You are not authorised to access Moto")
    speak2("Encrypting Servers and Shutting down")
    time.sleep(2)
    scare_moto.scare()
    sys.exit()

if identity == "AkshatRastogi":
    naam = "Akshat"
    mrms = "Mister"
elif identity == "SaumyaTripathi":
    naam = "Saumya"
    mrms = "Miss"
elif identity == "SamarthGoyal":
    naam = "Samarth"
    mrms = "Mister"

speak2("Authorization Complete")
```

Figure 4.4: Screenshot of python code used for using Facial Recognition in our voice assistant to Authorize Command Access

```
def recogniseMood():
    np.set_printoptions(suppress=True)
    vs = VideoStream(src=0).start()

    listA = []

    n = 1
    while n<=30:
        img = vs.read()
        result = DeepFace.analyze(img, actions=['emotion'])
        listA.append(result['dominant_emotion'])
        n += 2
        if n == 30:
            break

    a = max(set(listA), key=listA.count)
    return a
```

Figure 4.5: Screenshot of python code used to detect human moods from video frame

```python
elif "mood" in self.query:
    try:
        final_mood = mood_detection_moto.recogniseMood()
        speak(f"You seem to be {final_mood}")
        if final_mood == "happy":
            speak(f"It feels good to know that I have served you well {mrms} {naam}")
        elif final_mood == "sad":
            speak(f"I know you are having a bad day {mrms} {naam}")
            speak("I dont know what happened but just remember, you created me")
            speak("You can achieve anything you want")
        elif final_mood == "neutral":
            speak("You should keep working on what you want in order to achieve your goals")
        else:
            speak("What can I do ?")
    except:
        speak("There was some problem")
        speak("My best guess is that your face was not visible")
```

Figure 4.6: Screenshot of python code used for the implementation of mood detection in the voice assistant

```python
from hashlib import sha256

def mine_bitcoins():
    MAX_NONCE = 100000000000
    difficulty = 6

    def SHA256(text):
        return sha256(text.encode("ascii")).hexdigest()

    def mine(block_number, transactions, previous_hash, prefix_zeros):
        prefix_str = '0' * prefix_zeros
        for nonce in range(MAX_NONCE):
            text = str(block_number) + transactions + previous_hash + str(nonce)
            new_hash = SHA256(text)
            if new_hash.startswith(prefix_str):
                #Successfully mined bitcoins
                return new_hash

        raise BaseException(f"Couldn't find correct hash after trying {MAX_NONCE} times")

    transactions = '''
    Akshat->Saumya->60,
    Saumya->Samarth->90
    '''
    import time
    start = time.time()
    print("start mining")
    new_hash = mine(5, transactions, 'f40dd37a01d3fe6e999441a6918dc2c0cc048957661ce5a8c72e60cc64d09e13', difficulty)
    total_time = str((time.time() - start))
    print(f"end mining. Mining took: {total_time} seconds")
    print(new_hash)
```

Figure 4.7: Screenshot of python code used to mine bitcoins

```python
def gesture_controls():

    cap = cv2.VideoCapture(0)
    ...
    blue_lower=np.array([100,150,0],np.uint8)
    blue_upper=np.array([140,255,255],np.uint8)
    prev_y = 0
    while True:
        ret, frame = cap.read()
        geay = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
        mask = cv2.inRange(geay, blue_lower, blue_upper)
        contours, heirarchy = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

        for c in contours:
            area = cv2.contourArea(c)
            if area > 700:
                x,y,w,h = cv2.boundingRect(c)
                # cv2.drawContours(frame, c, -1, (0, 255, 0), 2)
                cv2.rectangle(frame, (x,y), (x+w, y+h), (0, 255, 0), 2)
                if prev_y-y > 50:
                    print('moving down')
                    pyautogui.press('space')
                prev_y = y

        cv2.imshow('Social Media',frame)
        k = cv2.waitKey(30) & 0xff
        if k == 27:
            break

    cap.release()
    cv2.destroyAllWindows()
```

Figure 4.8: Screenshot of python code used to activate gesture control

```
elif "i am going out" in self.query or "mask detection" in self.query:
    speak(f"Hey {mrms} {naam} can you please face the camera")
    answer = face_mask_detection.recognize_mask()
    if answer == "Mask Detected":
        speak("You are wearing a mask")
        speak("That's good")
        speak("Be Safe")
    elif answer == "Mask not Detected":
        speak("You are not wearing a mask")
        speak("Please wear a mask")
        speak("You should not go out without wearing the mask")
    else:
        speak("Your face was not detected")
        speak("Anyways have a safe journey")

#Checked
```

Figure 4.9: Screenshot of python code used to activate face mask detection control

```python
def recognize_mask():
    try:
        face_cascade = cv2.CascadeClassifier('converted_keras\\haarcascade_frontalface_alt2.xml')
        np.set_printoptions(suppress=True)
        #image = cv2.VideoCapture(0)
        vs = VideoStream(src=0).start()

        model = tensorflow.keras.models.load_model('converted_keras\\keras_model.h5')

        data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)
        # A dict that stores the labels
        labels = gen_labels()
        font = cv2.FONT_HERSHEY_SIMPLEX

        listA = []

        n = 1
        while n<=30:
            img = vs.read()
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            faces = face_cascade.detectMultiScale(gray, 1.1, 4)

            for (x,y,w,h) in faces:
                cv2.rectangle(img, (x,y), (x+w, y+h), (255,0,0), 2)
                frame2 = cv2.resize(img, (224, 224))# turn the image into a numpy array
                image_array = np.asarray(frame2)# Normalize the image
                normalized_image_array = (image_array.astype(np.float32) / 127.0) - 1  # Load the
                data[0] = normalized_image_array
                pred = model.predict(data)
                result = np.argmax(pred[0])

                cv2.putText(faces, "Label : " +
```

Figure 4.10: Partial screenshot of python code used to detect face mask from video frame

```
            cv2.putText(faces, "Label : " +
                        labels[str(result)], (280, 400), font, 1, (0, 255, 0), 2, cv2.LINE_AA)
            listA.append(labels[str(result)])# winsound.Beep(2500,1000)
            n += 1

        # cv2.imshow('img', img)
        k = cv2.waitKey(30) & 0xff
        if k==27:
            break

    vs.stop()
    m = 0
    wm = 0
    for i in listA:
        if i == "with_mask":
            m += 1
        else:
            wm += 1
    if m>wm:
        return("Mask Detected")
    else:
        return("Mask not Detected")
except:
    return ("Face Not Detected")
```

Figure 4.11: Partial screenshot of python code used to detect face mask from video frame

```
elif "function in python" in self.query:
    x = 'def sample(a,b):\n\tsam = a+b\n\treturn sam'
    pyperclip.copy(text=x)
    speak("okay")
    pyautogui.hotkey('ctrl', 'v', interval = 0.15)
```

Figure 4.12: Screenshot of python code used in Coding Guide functionality to declare a function in python

```
elif "class in python" in self.query:
    x = 'class Sample:\n\tdef __init__(self, a, b):\n\t\tself.a = a\n\t\tself.b = b'
    pyperclip.copy(text=x)
    speak("okay")
    pyautogui.hotkey('ctrl', 'v', interval = 0.15)
```

Figure 4.13: Screenshot of python code used in Coding Guide functionality to declare a class in python

```
elif "edit text in android" in self.query:
    x = '<TextView\n\tandroid:layout_width="wrap_content"\n
    pyperclip.copy(text=x)
    speak("okay")
    pyautogui.hotkey('ctrl', 'v', interval = 0.15)
```

Figure 4.14: Screenshot of python code used in Coding Guide functionality to declare an edit text in android (xml)

```
elif "indent in android" in self.query:
    x = 'Intent registerIntent = new Intent(SampleActivity1.th
    pyperclip.copy(text=x)
    speak("okay")
    pyautogui.hotkey('ctrl', 'v', interval = 0.15)
```

Figure 4.15: Screenshot of python code used in Coding Guide functionality to declare an indent in android (java)

```
elif "declare a button in android" in self.query:
    x = 'final Button button1 = findViewById(R.id.sampleid);'
    pyperclip.copy(text=x)
    speak("okay")
    pyautogui.hotkey('ctrl', 'v', interval = 0.15)
```

Figure 4.16: Screenshot of python code used in Coding Guide functionality to declare an button in android (java)

```
elif "declare an edit text in android" in self.query or "declare a
    x = 'final EditText et1 = findViewById(R.id.sampleid)'
    pyperclip.copy(text=x)
    speak("okay")
    pyautogui.hotkey('ctrl', 'v', interval = 0.15)
```

Figure 4.17: Screenshot of python code used in Coding Guide functionality to declare an edit text in android (java)

```
elif "declare an image in android" in self.query:
    x = 'final Image im1 = findViewById(R.id.sampleid)'
    pyperclip.copy(text=x)
    speak("okay")
    pyautogui.hotkey('ctrl', 'v', interval = 0.15)
```

Figure 4.18: Screenshot of python code used in Coding Guide functionality to declare an image in android (java)

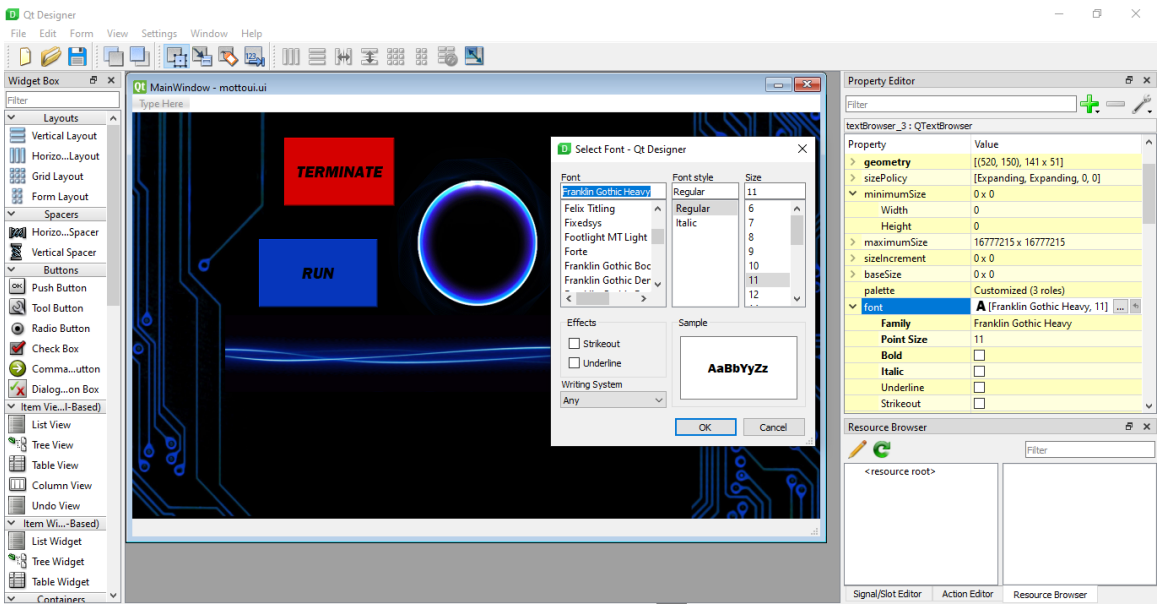

Figure 4.19: Screenshot of GUI Qt Designer Tool that was used to design the front-end for project zenith

Chapter 5

Summary

The Zenith project is based on the fundamentals of AI, machine learning and computer vision. As a part of Project Zenith we take a dive into many of these fields that are considered challenging and complicated to see for ourselves the maximum potential that we, first semester students can achieve with nothing more than a rudimentary knowledge of Python.

Our project consists of a voice assistant that is not only capable of handling mundane tasks but is also equipped with features like Face Recognition, Mood Recognition, Coding Guide etc. The collection of many of these unique features within a voice assistant, makes our project stand out among many others in the same category.

5.1 Proposed Project Aim

The Zenith project aims to foster open source culture in the university, which will result in the creation of a useful tech community.

We also strive to inspire people to pursue the field of information technology with passion.

5.2 Study & Project Design

To make this ambitious project a reality, we had to explore the basics of how computer vision and artificial intelligence work and understand the logic behind the use of machine learning algorithms in real-world situations using Computer Vision.

Our project is based on open source architecture and existing python libraries.

5.3 Hardware Specifications

The following hardware specifications are recommended for running project zenith smoothly on a machine.

1. Web-camera
2. Machine with an Intel/Ryzen dual Core processor or above
3. Dedicated or external microphone
4. Graphics Card
5. 4 GB RAM (Minimum) 8GB recommended
6. 8 GB free space recommended to run the project

5.4 Potential Impact

Our project will especially be useful to all the people who are new to coding and might be facing a lack of interest in this field.

Our project aims to show the joy of programming and how easy it is to create something of your own by writing code.

5.5 Future Plans

Project Zenith shall pave way for more complex and complicated projects in the future where various fields of Computer Science can work together to create a pseudo-realistic experience for humans and help humanity solve some of its major challenges.

In the future further improvements will be made in the voice assistant, such as implementation of a less robotic text to speech engine.

One of the other fields that needs improvement in the voice assistant is the field of speech recognition. In the future, we are planning to implement a robust and accurate mechanism to further improve the speech recognition system and improve the user experience.

5.6 Conclusion Remarks

5.6.1 Akshat Rastogi

Developing Project Zenith as part of our end-semester python project was on of the most rewarding expereince of my life.

A great team, consisting of passionate young scientists, made the project a resounding success. Each phase of the project brought with it a new set of challenges and overcoming those challenges gave us a lot of experience of the field.

5.6.2 Samarth Goyal

Working on a voice assistant that can compete on a commercial scale, gave me a lot of experience.

Project Zenith represents the collective work and effort of the entire team and also reflects our dedication towards this field of technology and our will to test out our maximum potential.

5.6.3 Saumya Tripathi

I had a wonderful time working with such hardworking and supporting teammates on a project that provided learning opportunities at its every stage.

I got to learn a lot in the past two months while simultaneously gauging my skills in Python.

Chapter 6

Bibliography

6.1 Python Libraries used in the project

The following libraries were used in making this project.

1. alabaster==0.7.12
2. anaconda-client==1.7.2
3. anaconda-navigator==1.9.12
4. anaconda-project==0.8.3
5. appdirs==1.4.3
6. APScheduler==3.6.3
7. argh==0.26.2
8. arrow==0.17.0
9. asn1crypto==1.3.0
10. astroid==2.3.3
11. astropy==4.0.1.post1
12. astunparse==1.6.3
13. atomicwrites==1.4.0

14. attrs==19.3.0
15. Babel==2.8.0
16. backcall==0.2.0
17. backports.functools-lru-cache==1.6.1
18. backports.shutil-get-terminal-size==1.0.0
19. backports.tempfile==1.0
20. backports.weakref==1.0.post1
21. bcrypt==3.1.7
22. beautifulsoup4==4.9.1
23. bkcharts==0.2
24. bleach==3.1.5
25. blinker==1.4
26. bokeh @ file:///C:/ci/bokeh1593178781838/work
27. boto==2.49.0
28. Bottleneck==1.3.2
29. brotlipy==0.7.0
30. bs4==0.0.1
31. CacheControl @ file:///tmp/build/80754af9/cachecontrol1594058963164/work
32. cachetools @ file:///tmp/build/80754af9/cachetools1607706694405/work
33. certifi==2020.12.5
34. cffi==1.14.0
35. chardet==3.0.4

36. clarifai==2.6.2
37. classifier==2.0
38. click==7.1.2
39. cloudpickle @ file:///tmp/build/80754af9/cloudpickle1594141588948/work
40. clyent==1.2.2
41. cmake==3.18.4.post1
42. colorama==0.4.3
43. comtypes==1.1.7
44. conda==4.9.2
45. conda-build==3.18.11
46. conda-package-handling==1.7.0
47. conda-verify==3.4.2
48. configparser==5.0.1
49. contextlib2==0.6.0
50. crayons==0.4.0
51. cryptography==2.9.2
52. cycler==0.10.0
53. Cython @ file:///C:/ci/cython1594829190914/work
54. cytoolz==0.10.1
55. dask @ file:///tmp/build/80754af9/dask-core1594156306305/work
56. decorator==4.4.2
57. deepface==0.0.48

58. defusedxml==0.6.0
59. diff-match-patch @ file:///tmp/build/80754af9/diff-match-patch1594828741838/work
60. distributed @ file:///C:/ci/distributed1594742844291/work
61. dlib==19.21.1
62. docutils==0.16
63. EasyProcess==0.3
64. emoji==0.6.0
65. entrypoints==0.3
66. et-xmlfile==1.0.1
67. face-recognition==1.3.0
68. face-recognition-models==0.3.0
69. fastcache==1.1.0
70. filelock==3.0.12
71. flake8==3.8.3
72. Flask==1.1.2
73. flatbuffers==1.12
74. fsspec==0.7.4
75. future==0.18.2
76. futures==3.1.1
77. gast @ file:///tmp/build/80754af9/gast1597433534803/work
78. gdown==3.12.2
79. gevent @ file:///C:/ci/gevent1593005471151/work

80. glob2==0.7
81. gmpy2==2.0.8
82. google-auth==1.24.0
83. google-auth-oauthlib @ file:///tmp/build/80754af9/google-auth-oauthlib1603929124518/wo
84. google-pasta==0.2.0
85. googleapis-common-protos==1.52.0
86. goslate==1.5.1
87. greenlet==0.4.16
88. grpcio==1.32.0
89. h5py==2.10.0
90. HeapDict==1.0.1
91. html5lib==1.0.1
92. idna==2.8
93. imageio @ file:///tmp/build/80754af9/imageio1594161405741/work
94. imagesize==1.2.0
95. importlib-metadata @ file:///C:/ci/importlib-metadata1593446511143/work
96. imutils==0.5.3
97. instaloader==4.6.1
98. instapy =0.6.8
99. intervaltree @ file:///tmp/build/80754af9/intervaltree1594361675072/work
100. ipaddr==2.2.0
101. ipykernel @ file:///C:/ci/ipykernel1594745408489/work/dist/ipykernel-5.3.2-py3-none-any.whl

102. ipython @ file:///C:/ci/ipython1593447482397/work
103. ipython-genutils==0.2.0
104. ipywidgets==7.5.1
105. isort==4.3.21
106. itsdangerous==1.1.0
107. jdcal==1.4.1
108. jedi @ file:///C:/ci/jedi1592833825077/work
109. Jinja2==2.11.2
110. joblib @ file:///tmp/build/80754af9/joblib1594236160679/work
111. json5==0.9.5
112. jsonschema==2.6.0
113. jupyter==1.0.0
114. jupyter-client @ file:///tmp/build/80754af9/jupyterclient1594826976318/work
115. jupyter-console==6.1.0
116. jupyter-core==4.6.3
117. jupyterlab==2.1.5
118. jupyterlab-server @ file:///tmp/build/80754af9/jupyterlabserver1594164409481/work
119. Keras==2.4.3
120. Keras-Applications @ file:///tmp/build/80754af9/keras-applications1594366238411/work
121. Keras-Preprocessing==1.1.2
122. keyring @ file:///C:/ci/keyring1593109799227/work
123. kiwisolver==1.2.0

124. lazy-object-proxy==1.4.3
125. libarchive-c==2.9
126. llvmlite==0.33.0+1.g022ab0f
127. locket==0.2.0
128. lockfile==0.12.2
129. lxml @ file:///C:/ci/lxml1594822774489/work
130. Markdown @ file:///C:/ci/markdown1605111189761/work
131. MarkupSafe==1.1.1
132. matplotlib @ file:///C:/ci/matplotlib-base1592837548929/work
133. mccabe==0.6.1
134. MeaningCloud-python==2.0.0
135. menuinst==1.4.16
136. mistune==0.8.4
137. mkl-fft==1.1.0
138. mkl-random==1.1.1
139. mkl-service==2.3.0
140. mock==4.0.2
141. more-itertools==8.4.0
142. MouseInfo==0.1.3
143. mpmath==1.1.0
144. msgpack==1.0.0
145. mtcnn==0.1.0

146. multipledispatch==0.6.0
147. navigator-updater==0.2.1
148. nbconvert==5.6.1
149. nbformat==5.0.7
150. networkx @ file:///tmp/build/80754af9/networkx1594377231366/work
151. newsapi-python==0.2.6
152. nltk @ file:///tmp/build/80754af9/nltk1592496090529/work
153. nose==1.3.7
154. notebook==6.0.3
155. numba==0.50.1
156. numexpr==2.7.1
157. numpy==1.19.3
158. numpydoc @ file:///tmp/build/80754af9/numpydoc1594166760263/work
159. oauthlib==3.1.0
160. olefile==0.46
161. opencv-python==4.4.0.46
162. openpyxl @ file:///tmp/build/80754af9/openpyxl1594167385094/work
163. opt-einsum==3.3.0
164. packaging==20.3
165. pandas @ file:///C:/ci/pandas1592833613419/work
166. pandocfilters==1.4.2
167. paramiko==2.7.1

168. parso==0.7.0
169. partd==1.1.0
170. path==13.1.0
171. pathlib2==2.3.5
172. pathtools==0.1.2
173. patsy==0.5.1
174. pep8==1.7.1
175. pexpect==4.8.0
176. pickleshare==0.7.5
177. Pillow==8.1.0
178. pkginfo==1.5.0.1
179. pluggy==0.13.1
180. ply==3.11
181. plyer==1.4.3
182. progress==1.5
183. prometheus-client==0.8.0
184. prompt-toolkit==3.0.5
185. protobuf==3.13.0
186. psutil==5.7.0
187. py @ file:///tmp/build/80754af9/py1593446248552/work
188. pyasn1==0.4.8
189. pyasn1-modules==0.2.8

190. PyAudio==0.2.11
191. PyAutoGUI==0.9.52
192. pycodestyle==2.6.0
193. pycosat==0.6.3
194. pycparser @ file:///tmp/build/80754af9/pycparser1594388511720/work
195. pycurl==7.43.0.5
196. PyDictionary==2.0.1
197. pydocstyle @ file:///tmp/build/80754af9/pydocstyle1592848020240/work
198. pyflakes==2.2.0
199. pygame==2.0.1
200. pygeoip==0.3.2
201. PyGetWindow==0.0.9
202. Pygments==2.6.1
203. pyjokes==0.6.0
204. PyJWT @ file:///C:/ci/pyjwt1608658192037/work
205. pylint==2.4.4
206. PyMsgBox==1.0.9
207. PyNaCl @ file:///C:/ci/pynacl1595000047588/work
208. pynput==1.7.2
209. pyodbc===4.0.0-unsupported
210. pyOpenSSL @ file:///tmp/build/80754af9/pyopenssl1594392929924/work
211. pyparsing==2.4.6

212. PyPDF2==1.26.0
213. pyperclip==1.8.1
214. pypiwin32==223
215. pyreadline==2.1
216. PyRect==0.1.4
217. pyrsistent==0.16.0
218. PyScreeze==0.1.26
219. PySocks==1.7.1
220. pytesseract==0.3.7
221. pytest==5.4.3
222. python-dateutil==2.8.1
223. python-geoip==1.2
224. python-geoip-geolite2==2015.303
225. python-jsonrpc-server @ file:///tmp/build/80754af9/python-jsonrpc-server1594397536060/v
226. python-language-server @ file:///C:/ci/python-language-server1594162130238/work
227. python-telegram-bot==13.0
228. pytoml==0.1.21
229. pyttsx3==2.90
230. PyTweening==1.0.3
231. pytz==2020.1
232. PyWavelets==1.1.1
233. pywhatkit==3.8

234. pywin32==227
235. pywin32-ctypes==0.2.0
236. pywinpty==0.5.7
237. PyYAML==5.3.1
238. pyzmq==19.0.1
239. QDarkStyle==2.8.1
240. QtAwesome==0.7.2
241. qtconsole @ file:///tmp/build/80754af9/qtconsole1592848611704/work
242. QtPy==1.9.0
243. regex @ file:///C:/ci/regex1593419644658/work
244. requests==2.22.0
245. requests-oauthlib==1.3.0
246. retrying==1.3.3
247. rope==0.17.0
248. rsa @ file:///tmp/build/80754af9/rsa1596998415516/work
249. Rtree==0.9.4
250. ruamel-yaml==0.15.87
251. scikit-image==0.16.2
252. scikit-learn @ file:///C:/ci/scikit-learn1592853510272/work
253. scipy @ file:///C:/ci/scipy1604596260408/work
254. seaborn==0.10.1
255. selenium==3.141.0

256. Send2Trash==1.5.0
257. simplegeneric==0.8.1
258. singledispatch==3.4.0.3
259. sip==4.19.13
260. six @ file:///C:/ci/six1605183554193/work
261. snowballstemmer==2.0.0
262. sortedcollections==1.2.1
263. sortedcontainers==2.2.2
264. soupsieve==2.0.1
265. SpeechRecognition==3.8.1
266. Sphinx @ file:///tmp/build/80754af9/sphinx1594223420021/work
267. sphinxcontrib-applehelp==1.0.2
268. sphinxcontrib-devhelp==1.0.2
269. sphinxcontrib-htmlhelp==1.0.3
270. sphinxcontrib-jsmath==1.0.1
271. sphinxcontrib-qthelp==1.0.3
272. sphinxcontrib-serializinghtml==1.1.4
273. sphinxcontrib-websupport @ file:///tmp/build/80754af9/sphinxcontrib-websupport1593446
274. spotipy==2.16.1
275. spyder @ file:///C:/ci/spyder1594830825244/work
276. spyder-kernels @ file:///C:/ci/spyder-kernels1594751670175/work
277. SQLAlchemy @ file:///C:/ci/sqlalchemy1593445271541/work

278. statsmodels==0.11.1
279. sympy @ file:///C:/ci/sympy1594234545115/work
280. tables==3.6.1
281. tblib==1.6.0
282. tensorboard==2.4.0
283. tensorboard-plugin-wit==1.7.0
284. tensorflow==2.4.0
285. tensorflow-estimator==2.4.0
286. termcolor==1.1.0
287. terminado==0.8.3
288. testpath==0.4.4
289. threadpoolctl @ file:///tmp/tmp9twdgx9k/threadpoolctl-2.1.0-py3-none-any.whl
290. toml @ file:///tmp/build/80754af9/toml1592853716807/work
291. toolz==0.10.0
292. tornado==6.0.4
293. tqdm @ file:///tmp/build/80754af9/tqdm1593446365756/work
294. traitlets==4.3.3
295. typing-extensions @ file:///tmp/build/80754af9/typingextensions1592847887441/work
296. tzlocal==2.1
297. ujson==1.35
298. unicodecsv==0.14.1
299. urllib3==1.25.8

300. watchdog @ file:///C:/ci/watchdog1593447437088/work
301. wcwidth @ file:///tmp/build/80754af9/wcwidth1593447189090/work
302. webdriver-manager==3.2.2
303. webdriverdownloader==1.1.0.3
304. webencodings==0.5.1
305. Werkzeug==1.0.1
306. widgetsnbextension==3.5.1
307. wikipedia==1.4.0
308. win-inet-pton==1.1.0
309. win-unicode-console==0.5
310. wincertstore==0.2
311. wrapt==1.12.1
312. xlrd==1.2.0
313. XlsxWriter==1.2.9
314. xlwings==0.19.5
315. xlwt==1.3.0
316. xmltodict==0.12.0
317. yapf @ file:///tmp/build/80754af9/yapf1593528177422/work
318. zict==2.0.0
319. zipp==3.1.0
320. zope.event==4.4
321. zope.interface==4.7.1
322. intervals =0.9.0

6.2 Python Libraries Citation

1. Continuum Analytics. Conda is a cross-platform, language-agnostic binary package manager. it is the package. https://pypi.org/project/conda/, 2019.

2. Natesh M Bhat. pyttsx3 is a text-to-speech conversion library in python. https://pypi.org/project/pyttsx3/, 2020.

3. Peter Brady. C implementation of python 3 functools.lrucache. provides speedup of 10-30x over standard library. passes test suite from standard library for lrucache. https://pypi.org/project/fastcache/, 2020.

4. Jeff Forcier. Alabaster is a visually lean, responsive, configurable theme for the sphinx documentation system. it is python 2+3 compatible. https://pypi.org/project/alabaster 2019.

5. Adam Geitgey. Recognize and manipulate faces from python or from the command line with the world's simplest face recognition library. https://pypi.org/project/face-recognition/, 2020.

6. Tim Großmann. Tooling that automates your social media interactions to "farm" likes, comments, and followers on instagram implemented in python using the selenium module. https://pypi.org/project/instapy/, 2021.

7. Olli-Pekka Heinisuo. Opencv (open source computer vision library) is an open source computer vision and machine learning software library. https://pypi.org/project/opencv python/, 2020.

8. Google Inc. Tensorflow is an end-to-end open source platform for machine learning. https://pypi.org/project/tensorflow/, 2020. 101

9. Spotify Inc. A light weight python library for the spotify web api. https://pypi.org/project/spotify 2020.

10. Michael Droettboom John D. Hunter. Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in python https://pypi.org/project/matplotlib 2021.

11. Taehoon Kim. Facebook chat (messenger) for python. this project was inspired by facebook-chat-api. https://pypi.org/project/fbchat/, 2020.

12. Jason Kirtland. Blinker provides a fast dispatching system that allows any number of interested parties to subscribe to events, or "signals". https://pypi.org/project/blin 2020.

13. Ankit Raj Mahapatra. Pywhatkit is a python library for sending whatsapp message at certain time, it has several other features too. https://pypi.org/project/pywhatkit/, 2020.

14. Travis E. Oliphant. Numpy is a library for the python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. https://pypi.org/project/numpy/, 2019.

15. PSF. Random module the random module is a built-in module to generate the pseudo-random variables. https://pypi.org/project/random2/, 2019.

16. Kenneth Reitz. Certifi provides mozilla's carefully curated collection of root certificates for validating the trustworthiness of ssl certificates while verifying the identity of tls hosts. https://pypi.org/project/certifi/, 2019.

17. Leonard Richardson. Beautiful soup is a library that makes it easy to scrape information from web pages. it sits atop an html or xml parser, providing pythonic idioms for iterating, searching, and modifying the parse tree. https://pypi.org/project/beautifulsoup4/, 2019.

18. Mark Roseman. Tkinter is the most commonly used library for developing gui (graphical user interface) in python. https://docs.python.org/3/library/tkinter.html, 2020. 102

19. Ross Stephan. The smtplib module defines an smtp client session object that can be used to send mail to any internet machine. https://docs.python.org/3/library/smtplib. 2020.