Final Report on

VIRTUAL VOICE ASSISTANT

Subject: Artificial Intelligence (INT 404)

Submitted to: Mr.Sagar Pande

Section: K18KK

Submitted By:

S.no	Roll No.	Group Member	Reg. ID	Marks
1	38	P.Bhargavi Vinay	11803021	
2	60	M.Jai Surya Teja	11803022	

Git Hub :

https://github.com/vinay1001/Virtual-Assistant-AI-Project

1. Abstract of the project:

This project is “VIRTUAL VOICE ASSISTANT”. The main aim of this project is to create a responsive virtual assistant to perform basic operations based on oral commands of user. It can perform operations based on oral instructions given by the user. It performs operations like opening applications, displaying date and time, browsing information, greetings etc. This project is more effective because of its simple and responsive GUI (Graphical User Interface). As virtual assistance is a playing a vital role in every sector like virtual help in business, health etc. There is a lot of scope for this project in the present generation AI. As everything is getting automated these days people are showing enormous interest towards the applications that are working for them like chatbots, voice assistants , which are easy to work on. Examples of these trending application is Google Assistant, Siri for iPhone, Amazon Alexa, Cortana in windows etc., So, it has lot of future scope to learn and to expand its application.

2. Previous Work and Limitations:

The project's previous work is limited only to very basic operations like common greetings to user, displaying info of a person, and displaying date and time. Previous work didn't contain any GUI for effective interaction to the virtual assistant. It also limited in the sense of diversity of user requirements, it was very specified for only 2 to 3 applications.

The main limitation of the previous work was lack of GUI, which makes difficult for user to interact with the assistant. Another main limitation of previous work was the lack of synchronization between request time and response time of assistant. It didn't contain features to browse information on internet, opening applications on oral commands.

All these limitations were removed during the progress of the project.

3. Implementation:

CODE

```
# Import packages required

import speech_recognition as sr
import playsound  # to play saved mp3 file
from playsound import playsound
from gtts import gTTS  # google text to speech
import os  # to save/open files
import wolframalpha  # to calculate strings into formula
from selenium import webdriver  # to control browser operations
import warnings
import calendar
import random
import wikipedia
import datetime
import threading
import webbrowser

import subprocess

from tkinter import *
import tkinter as tk

import requests
import urllib.request
from bs4 import *
from PIL import Image

# Ignoring warning messages
warnings.filterwarnings('ignore')


# Record audio and return as a string

def recordAudio():
    # to record the audio
    r = sr.Recognizer()  # Added a recogniser object

    # Use the microphone
    with sr.Microphone() as source:
        print('Say Something')
        #answer.set('Say Something...')
        audio = r.listen(source)

    # Use Google speech_reconition
    data = ''
    try:
        data = r.recognize_google(audio)
        update_user(data)
```

```
        print("You said : " + data)
    except sr.UnknownValueError:  # Checks for unknown errors
        print("Google speech recognition could not understand the audio,unknown
error")
    except sr.RequestError as e:
        print('Google speech recognition error is ' + str(e))

    return data


# A function to get the virtual assistance response
def assistantResponse(text):
    print(text)
    if(len(text) <= 123):
        text = text
    else:
        if(len(text) > 123):
            text = text[:123]+ "\n" + text[123:]
        if(len(text) > 246):
            text = text[:246] + "\n" + text[246:]
        if (len(text) > 369):
            text = text[:369] + "\n" + text[369:]
    update(text)

    # Converting the text into speech
    myobj = gTTS(text=text, lang='en', slow=False)  # slow = false makes computer to
re text more slowly

    # To save the audio in a mp3 file

    myobj.save('assistant_response.mp3')

    # Play the file audio saved

    os.system('start assistant_response.mp3')


# A function for wake word

def wakeWord(text):
    wake_words = ['hello vinay', 'hello ruby', 'hi ruby', 'ruby baby', 'ruby ',
'google',
                  'Google']  # list of wake phrases

    text = text.lower()  # converting text to lower case as all my all wake words are
in lowercase

    # Check if the user text is any of the wake words
    for phrase in wake_words:
        if phrase in text:
            return True

    # Executes if wake word not found
    return False


# A function to get the current date
def getDate():
    now = datetime.datetime.now()
    date = datetime.datetime.today()
    weekday = calendar.day_name[date.weekday()]  # E.g : Friday
    monthNum = now.month
    dayNum = now.day
```

```
    # A list of months
    month_names = ['January', 'Febraury', 'March', 'April', 'May', 'April', 'June',
'July', 'August', 'September',
                   'October', 'November', 'December']

    # A list of ordinal numbers
    ordinalNumbers = ['1st', '2nd', '3rd', '4th', '5th', '6th', '7th', '8th', '9th',
'10th', '11th', '12th', '13th',
                      '14th', '15th', '16th', '17th', '18th', '19th', '20th', '21st',
'22nd', '23rd', '24th', '25th',
                      '26th', '27th', '28th', '30th', '31st']

    return 'Today is ' + weekday + ' ' + month_names[monthNum - 1] + ' the ' +
ordinalNumbers[dayNum - 1] + ' . '


def greetings(text):
    # List of greeting inputs
    greetings_input = ['hey', 'hi', 'bonjour', 'namaste', 'hello', 'wassup', 'hello']

    # List of greeting response
    greetings_response = ['howdy', 'whats good', 'hello', 'hey there']

    # If the users input is a greeting - so return a greeting response

    for word in text.split():
        if word.lower() in greetings_input:
            return random.choice(greetings_response) + ' . '

    # No greeting detected
    return ''


# A function to get a peson's first name and last name from the text
def getPerson(text):
    wordList = text.split()  # We are spliting the text to list of words

    for i in range(0, len(wordList)):
        try:
            if wordList[i].lower() == 'who' and wordList[i + 1].lower() == 'is':
                return wordList[i + 2] + ' ' + wordList[i + 3]
        except IndexError:
            return wordList[i + 2]


# To get information from wikipedia other than persons
def getAbout(text):
    wordList = text.split()  # We are spliting the text to list of words

    for i in range(0, len(wordList)):
        try:
            if (wordList[i].lower() == 'what' and wordList[i + 1].lower() == 'is') or
((wordList[i].lower() == 'describe' or wordList[i].lower() == 'explain') and
wordList[i + 1].lower() == 'about'):
                return wordList[i + 2] + ' ' + wordList[i + 3]
        except IndexError:
            return wordList[i + 2]


def close_application(text):
    closing_words = ['see you', 'bye', 'sleep', 'close', 'hasta vista']
    text_words = text.split()
```

```
    for i in text_words:
        if i in closing_words:
            return True


# Function to check and open available desktop applications
def open_app(text):
    WordList = text.split()
    app = ''
    for i in range(0, len(WordList)):
        try:
            if (WordList[i].lower() == "open"):
                app = WordList[i + 1].lower()
                break
        except IndexError:
            app = WordList[i].lower()

    app_list = ['google', 'word', 'presentation', 'excel', 'notepad']
    count = 0
    for i in app_list:
        if app.lower() == i:
            count = 1
            break

    if (count == 0):
        return 0

    if (app == 'notepad'):
        resource = r"C:\Program Files (x86)\Notepad++\notepad++.exe"
        subprocess.Popen(resource)
        return 1
    if (app == 'word'):
        resource = r'C:\Users\lenovo\Desktop\AI.docx'
        os.system(resource)
        return 1
    if (app == 'excel'):
        resource = r'C:\Users\lenovo\Desktop\AI.xlsx'
        os.system(resource)
        return 1
    if (app == 'presentation'):
        resource = r'C:\Users\lenovo\Desktop\AI.pptx'
        os.system(resource)
        return 1




# Initiation of virtual assistance


def main():
    # while True:
    # Record a Audio

    text = recordAudio()
    response = ' '

    # Check for the wake word
    if (wakeWord(text) == True):

        # Check for greetings by the user
        response = response + greetings(text)
```

```
        # Check to see user has anything said about date
        if ('date' in text):
            get_date = getDate()
            response = response + ' ' + get_date

        # User asked time - maybe
        if ('time' in text):
            now = datetime.datetime.now()
            meridiem = ''
            if now.hour >= 12:
                meridiem = 'p.m'  # Post Meridiem (PM)
                hour = now.hour - 12
            else:
                meridiem = 'a.m'  # Ante Meridiem (AM)
                hour = now.hour

            # Convert minute into string
            if now.minute < 10:
                minute = '0' + str(now.minute)
            else:
                minute = str(now.minute)

            response = response + ' ' + 'It is ' + str(hour) + ':' + minute + ' ' +
meridiem + ' .'
        # Check if user said 'who is'
        if ('who is' in text):
            person = getPerson(text)

            wiki = wikipedia.summary(person, sentences=2)
            response = response + ' ' + wiki

        if ('explain about ' in text) or ('what is ' in text) or ('describe about ' in
text):
            about = getAbout(text)
            wiki = wikipedia.summary(about, sentences=3)
            response = response + ' ' + wiki

        if ('open' in text):
            res = open_app(text)
            if (res == 1):
                response = response + " " + " application opened."
            if (res == 0):

                textWords = text.split()
                for i in range(0, len(textWords)):
                    if (textWords[i].lower() == 'open'):
                        website = textWords[i + 1].lower()
                        break
                response = response + " " + website + " opened."
                website = "https://www." + website + ".com/"
                webbrowser.open_new(website)

        if ('search' in text):
            textWords = text.split()
            for i in range(0, len(textWords)):
                if (textWords[i].lower() == 'search'):
                    if (textWords[i + 1].lower() == 'for' or textWords[i + 1].lower()
== 'about'):
                        website = 'https://www.google.com/search?client=firefox-b-
d&q=' + textWords[i + 2].lower()
                        break
                    else:
```

```
                        website = 'https://www.google.com/search?client=firefox-b-
d&q=' + textWords[i + 1].lower()
                        break
            response = response + " " + " These are the search results."
            webbrowser.open_new(website)

        if ('how are you' in text or 'How are you' in text):
            response = response + " " + " I am fine,what can i do for you?"

        # Save the assistant response back using audio and text from response
        if len(response) == 1:
            assistantResponse('Say Something')
            length_of_text = len(response)
            length = (length_of_text / 10)
            timer = threading.Timer(length, main)
            timer.start()
        else:
            assistantResponse(response)

            # Calling main function using timer
            length_of_text = len(response)
            length = (length_of_text / 10)
            timer = threading.Timer(length, main)
            timer.start()

        if (close_application(text) == True):
            assistantResponse("see you later.")
            exit(0)


# main()


top = tk.Tk()
top.title('Home')
top.configure(background='white')

photo = PhotoImage(file=r"C:\Users\lenovo\Desktop\AIPROJECT\dots.png")
photoimage = photo.subsample(1, 1)

photo1 = PhotoImage(file=r"C:\Users\lenovo\Desktop\AIPROJECT\dots.png")
photoimage1 = photo1.subsample(1, 1)


def update(to_update):
    update_text.set(to_update)
    return 0


def update_user(to_update):
    answer.set(to_update)
    return 0


def call():
    main()


global update_text, answer
answer = StringVar()
update_text = StringVar()

ra = 'Say Something....'
```

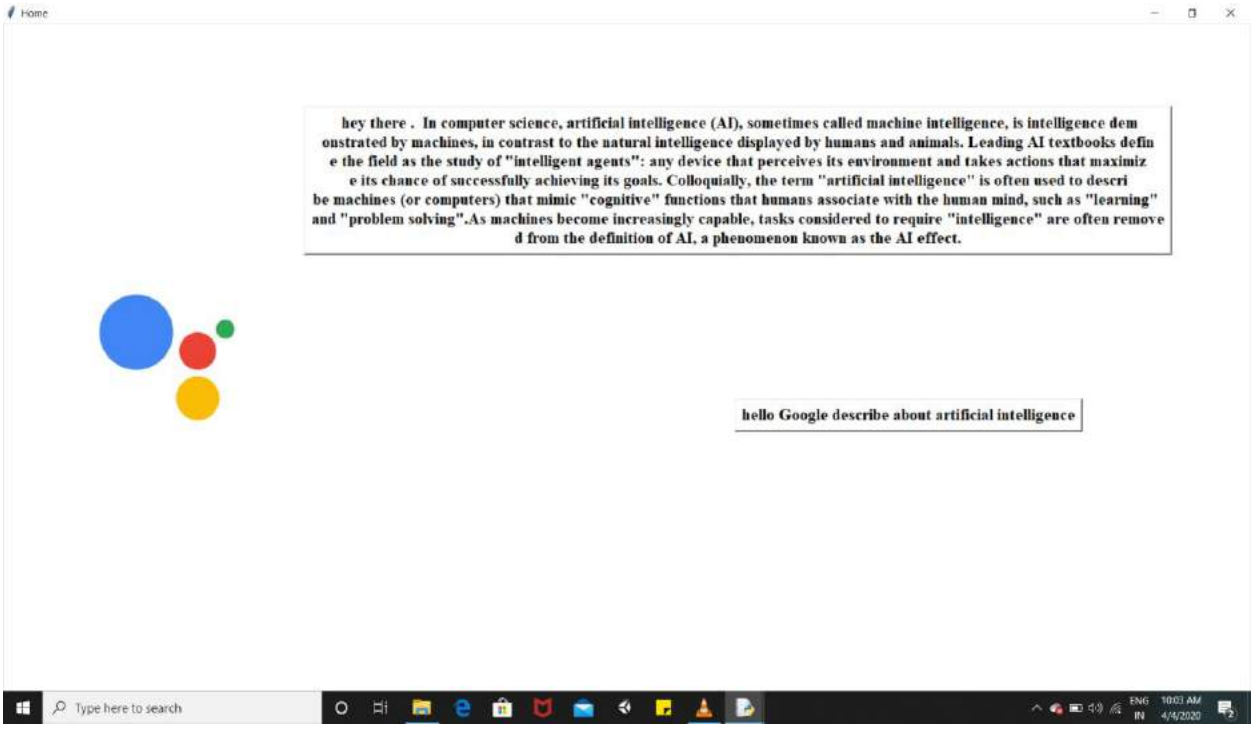
```
Q = Button(top, textvariable=answer, bg='white', font='Times 15 bold').place(x=900,
y=450)
A = Button(top, textvariable=update_text, bg='white', font='Times 15
bold').place(x=370, y=100)

update_text.set(ra)
answer.set(ra)

va = Button(top, bg='white', image=photo, command=lambda: call(), width=200,
height=200, bd=0).place(x=100, y=300)
#display = Button(top,bg = 'white',image = photo1,width = 200,height = 400,bd =
0).place(x = 500,y = 350)

top.mainloop()
```

4. Output:

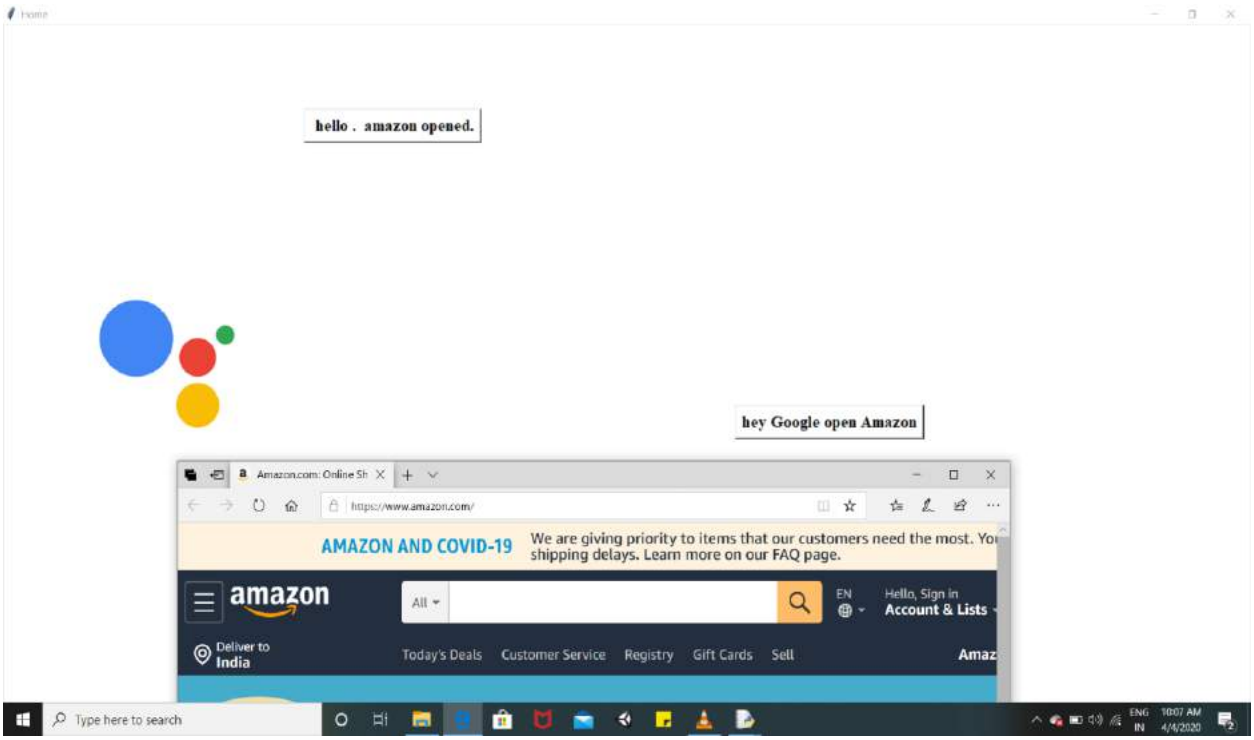
OUTPUT SCREENSHOTS

Note:

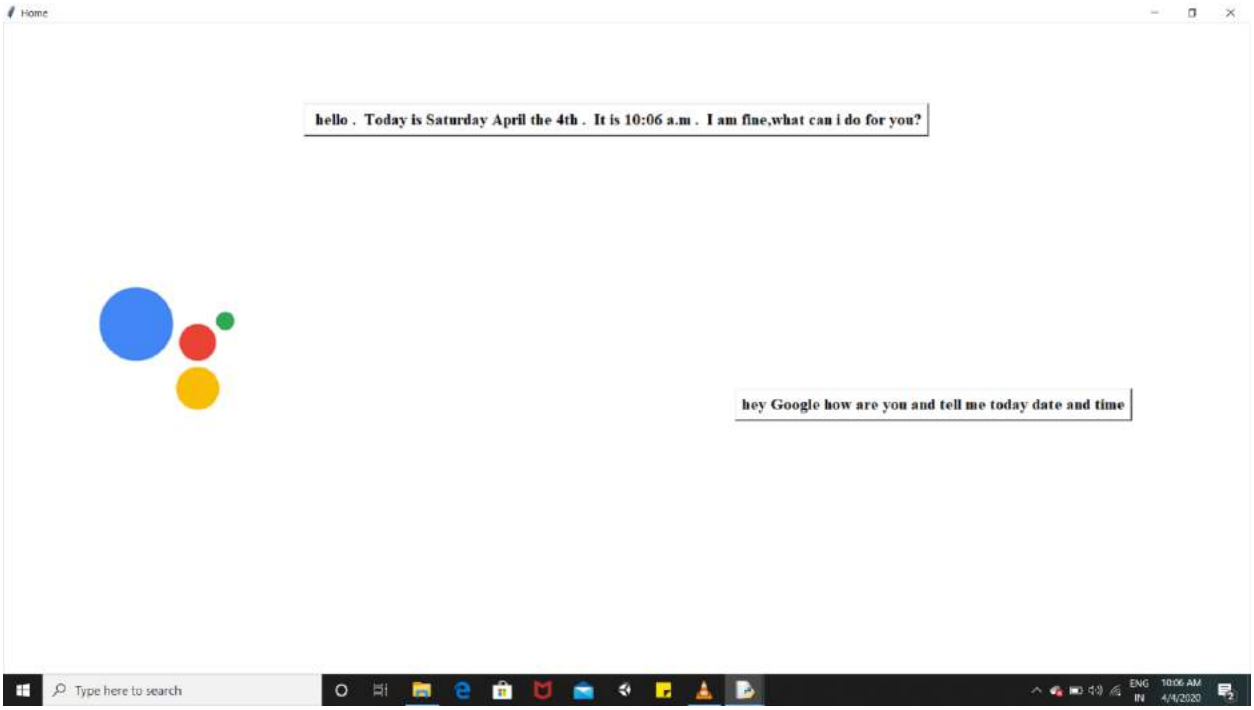
For a voice assistant all the input and outputs are oral. These are the screenshots of those oral input and outputs displayed on interface screen (GUI).



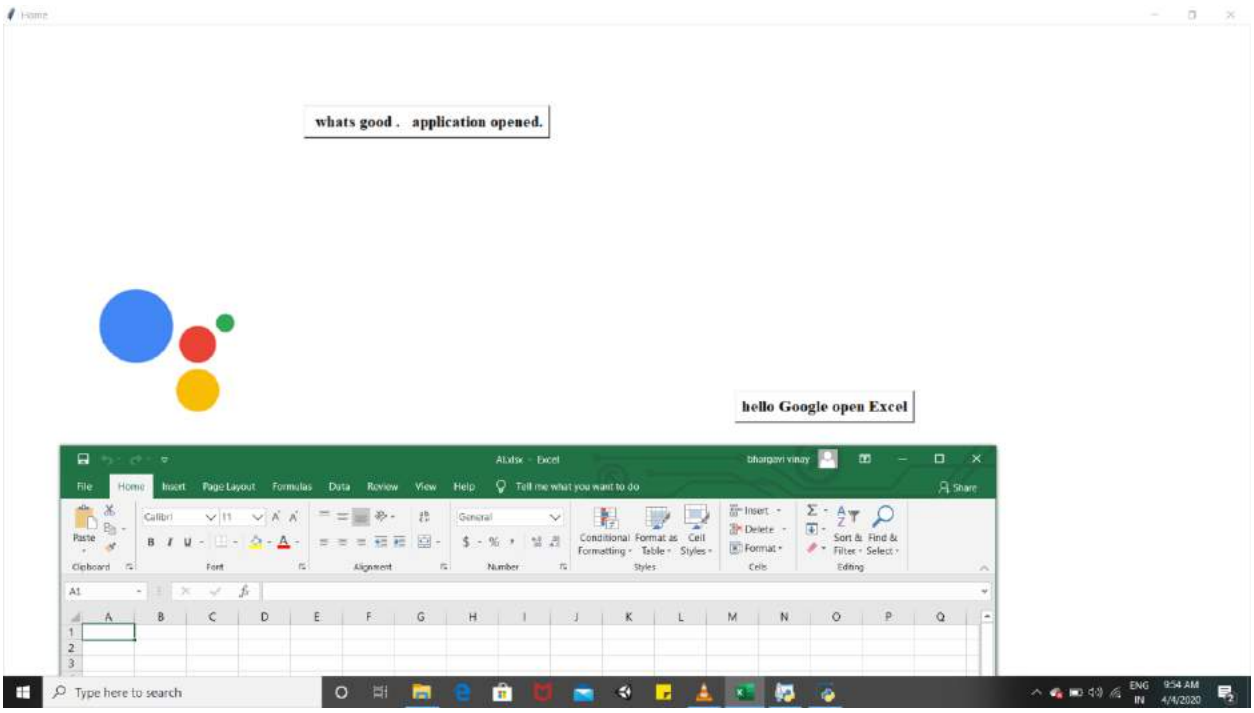
Describing about requested topic



Opening applications on user's oral command



Telling Time and Date



Opening Desktop Applications like word, power point, excel etc.,

Telling Information about persons, places etc.,

Replying for greetings.

5. Libraries Used:

Gtts (Google text to speech) library is used to convert the text to audio. It is useful in production of voice for the assistant according to the user instruction.

speech_recognition Library for performing speech recognition, with support for several engines and APIs, online and offline.

playsound library is used to play the sound of converted text uskking Gtts*. It is a* Pure Python, cross platform, single function module with no dependencies for playing sounds.

Calendar library was used for fetching date and time. Wikipedia library is used for fetching the information of persons, places, topics according to user command.

os, subprocess libraries are used to open the desktop applications on the system like excel, word etc, threading library is used

to create a timer between the user command and virtual assistant response.

warning library was used to warn the system for malfunctioning in the code. Warning messages are typically issued in situations where it is useful to alert the user of some condition in a program, where that condition (normally) doesn't warrant raising an exception and terminating the program.

Tkinter was used to create GUI for the Virtual Assistant. This project also used webbrowser library to search for user requirements.

6. Team Responsibilities:

This project team comprised of two persons. The project has lot of applications to work on. The work is divided among the team members according to the number of applications it is dealing with. The work of team members is divided among the modules included in the project.

Team member 1:

Tokenizing the user command to fetch the suitable word, and to understand the user's requirement. Opening applications on user request by using subprocess and os libraries. Used wikipedia library to fetch the information about a person, palce etc., and to display the text on the virtual assistant section in GUI. Modules were created and divided based on the no.of libraries used.

Team member 2:

Invoking and shutting of the virtual assistant application. Interacting with web using web browser library to collect the required information about the request of user. GUI (Graphical User Interface) for the Virtual

Assistant.Recognising greetings, date, time commands of user writing of code for response of Virtual Assistant.

7. References:

https://pypi.org/

This contains the documentations of various libraries that were used in this project. It was really useful to identify the working of various modules and the ways to install them.

https://www.geeksforgeeks.org/personal-voice-assistant-in-python/

This is a simple personal voice assistant code written in python on geeks for geeks platform. It helped as beacon of start for this project. It helped to understand the project. It contained the names of basic libraries that can be used for this project.

https://searchcustomerexperience.techtarget.com/definition/virtual-assistant-AI-assistant

This link provided every theoretical concept of virtual assistant.