7 November 2020

# PROJECT REPORT
# ON

# F.R.I.D.A.Y.
# (Personal Assistant)


PREPARED BY:

**Rutvij Pritesh Shah** & **Patel Sahil Manishkumar**
17ECUOG026 17ECUOT013
EC67 EC55

UNDER THE GUIDANCE OF:

**Prof. Hetal B. Shah** & **Prof. Sohilkumar A. Dabhi**

DHARMSINH DESAI UNIVERSITY
FACULTY OF TECHNOLOGY
DEPARTMENT OF ELECTRONICS & COMMUNICATION

# CERTIFICATE

This is to certify that the project on **F.R.I.D.A.Y.** and term work carried out in the subject of Term Project is bona fide work of **PATEL SAHIL MANISHKUMAR** of B. Tech. **semester 7** in the branch of **Electronics & Communication**, during the academic year **2020-21**.

Prof. Hetal B. Shah
Project Guide, EC Dept.

Dr. Nikhil Kothari
HOD, EC Dept.

# CERTIFICATE

This is to certify that the project on **F.R.I.D.A.Y.** and term work carried out in the subject of Term Project is bona fide work of **RUTVIJ PRITESH SHAH** of B. Tech. **semester 7** in the branch of **Electronics & Communication**, during the academic year **2020-21**.

Prof. Sohilkumar A. Dabhi
Project Guide, EC Dept.

Dr. Nikhil Kothari
HOD, EC Dept.

# ACKNOWLEDGEMENT

Firstly, we offer our sincere thanks to our guides **Prof. Hetal B. Shah**, Associate Professor, Electronics and Communication Department, Dharmsinh Desai University and **Prof. Sohilkumar A. Dabhi**, Associate Professor, Electronics and Communication Department, Dharmsinh Desai University for their invaluable support, guidance and advice throughout the duration of this Project.

We express my deep and sincere sense of gratitude to **Dr. Nikhil J. Kothari**, HOD and Professor, EC Department, Dharmsinh Desai University, who has given us invaluable support and opportunities to learn and develop technical skills. He has been an unending source of inspiration to me. We are thankful to all the faculties members and university staff for their consistent support and guidance

# ABSTRACT

The purpose of this project is to make a Smart Personal Assistant which can help us do various tasks while we are busy doing something else. We named our Personal Assistant "F.R.I.D.A.Y." based on the fictional A.I. developed by billionaire genius Tony Stark. We were inspired by the smart and sassy A.I. and aimed to make our own version of it. We constructed F.R.I.D.A.Y.'s code from scratch in Python language and used various open-source libraries to add various functionalities to F.R.I.D.A.Y.

F.R.I.D.A.Y. has speech recognition capability and thus can make our life easier and more fun while doing work.

Currently, F.R.I.D.A.Y. can be used as a desktop application in Windows, Linux or Apple operating systems. We further aim to deploy it on a Raspberry Pi, so that it becomes portable and has even more versatile uses.

# TABLE OF CONTENTS:

# 1. Introduction

## 1.1 Premise:

- F.R.I.A.Y is a smart, intelligent and sassy personal assistant to help you with your daily tasks and have some fun along the way.
- Responding to voice commands, she has a ton of features customizable to the user and very simple to use.
- She takes various commands and implements them. She can Search Wikipedia for you, send email, do a chrome search. Can help you shut down or restart with voice commands, take a screen shot, crack jokes, update you about the weather and the health of your PC.
- Along with home-automation hardware, she can help you turn off/on lights in your house. She can help you take notes, etc.
- We will be developing it in our laptop and future extension project we will convert it in fully function smart speaker.

- **Language used for programming:** Python 3.7
- **Editor:** Microsoft Visual Studio Code

## 1.2 List of Features of Personal Assistant:

- Text to Speech
- Unique Greeting
- Tell Date and Time in various formats
- Accurate Speech Recognition
- Translate feature (12 languages available)
- GUI Application
- Wikipedia Search
- Google Search
- YouTube Search
- Location Search
- Latest News Update
- Weather Update
- Jokes for Programmers
- Custom Email Sending
- Can answer any Random Question
- Calculator
- Open PC Application
- Play offline Songs
- CPU and Battery Status Update
- Take screenshot
- Take Notes
- Shutdown PC
- Customizable by the end user
- Remember various preferences of the user

# 2. ALGORITHM OF VARIOUS COMPONENTS

## 2.1 Working Mechanism & Code flow:

```
#Basic algorithm of our code:
GUI_infinite_loop()
{
        if ( 'x' button press)
        {
                #Individual functions for various features:

                buttonfunc1()
                {       function 1    }

                buttonfunc2()
                {       function 2    }
                .
                .
                .
                buttonfunc29()
                {       function 29   }
        }

        else if(TTS input)
        {
                # Main TTS function loop:

                TTS_infinite_loop()
                {
                        if()
                        {       do this         }

                        else if()
                        {       do that         }
                        .
                        .
                        .
                        else
                        {       default msg   }
                }
        }

}
```

## 2.2 Text to Speech (TTS):

**Library Used:** pyttsx3

**Basic Code:**

```
import pyttsx3                  # Library
engine = pyttsx3.init()         # One time initialization
def text_to_speech(audio):      # function to pass a string to convert into speech.
    engine.say(audio)           #Speech text
    engine.runAndWait()         # Flush the say() queue and play the audio

# passing string through function:
text_to_speech("Hello I am your Personal Assistant. What can I do for you?")

# Setting properties
engine.setProperty('rate', 175)             # Speed percent (can go over 100)
engine.setProperty('volume', 1)             # Volume 0-1
voices = engine.getProperty('voices')       #Taking list of voices
```

**Explanation:**

- This feature helps the assistant to interact with us by converting the output text to speech and emitting the sound through speakers of our PC.
- We can change the speech rate, the gender of the assistant, the type of voice being heard.
- We have used pyttsx3 library for this conversion instead of gTTS as pyttsx3 works completely offline and converts TTS using the voices stored on the PC. Whereas gTTS works completely online and thus gives considerably more delay than pyttsx3.

**For information on pyttsx3:** Refer Pg. No 34

## 2.3 Date and time:

**Library Used:** datetime

**Basic Code:**

```
import datetime                                    # Date & Time library
def time_12f():                                    # 12hr time function
   Time12f = datetime.datetime.now().strftime("%I:%M:%S")
   AM_PM = datetime.datetime.now().strftime("%p")
   text_to_speech("Current time is:")              # Function Call
   text_to_speech(Time12f)                         # Function Call
   text_to_speech(AM_PM)                           # Function Call
def time_24f():                                    # 24hr time function
   Time24f = datetime.datetime.now().strftime("%H:%M:%S")
  text_to_speech("Current time is:")               # Function Call
   text_to_speech(Time24f)                         # Function Call
def date():                                        # Getting current date
   Year = int(datetime.datetime.now().year)        # typecasting in to integer value
   Month = int(datetime.datetime.now().month)      # typecasting in to integer value
   Date = int(datetime.datetime.now().day)         # typecasting in to integer value
   Day= datetime.datetime.now().strftime("%A")
```

**Explanation:**

- This feature is useful for knowing the current date and time. We have made various customized functions to help the assistant output time or/and date in various different formats according to the user's need.
- F.R.I.D.A.Y can tell you the time in 12 hour and 24-hour format. The day of the week, week number, date in various formats, etc.
- Time, date, day, hour, min, sec, week values are extracted from the datetime library in string or integer formats.
- Then the values are passed along to the text to speech function to be sent to us.

**For information on library:** Refer Pg. No 34

## 2.4 Customized Greeting:

**Library Used:** datetime

**Code:**

```
def greeting():
    hour = datetime.datetime.now().hour
    if hour >= 6 and hour<12:
        text_to_speech("Good morning!")          # passing string through function
        time_12f()
    elif hour >=12 and hour<18:
        text_to_speech("Good afternoon!")        # passing string through function
        time_12f()
    elif hour>=18 and hour<24:
        text_to_speech("Good evening!")          # passing string through function
        time_12f()
    else:
        text_to_speech("Good night!")            # passing string through function
        time_12f()
```

**Explanation:**

- This feature helps F.R.I.D.A.Y greet us according to the time of the day. As this time is extracted from the computer time, even when the time zones change it can still tell us the correct time.

**For information on library:** Refer Pg. No 34

## 2.5 Speech Recognition:

**Library Used:** speech_recognition

**Code:**

```
def SpeechCommand():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        r.adjust_for_ambient_noise(source)          # listen for 1 second to calibrate
the energy threshold for ambient noise levels
        print("\n.....Listening.....\n")
        # r.pause_threshold=1                  # Not required when using adjust_for
_ambient_noise_function()
        audio=r.listen(source)

    # recognize speech using Google Speech Recognition
    try:
        query = r.recognize_google(audio, language='en-in')
        print(query)

    except sr.UnknownValueError:
        print("Google Speech Recognition could not understand audio")
        #text_to_speech(" Pardon, Please repeat again...")
        query="recognition error"

    except sr.RequestError as e:
        print("Could not request results from Google Speech Recognition service;
{0}".format(e))
        query="server error"

    return(query)
```

**Explanation:**

- This function enables F.R.I.D.A.Y to recognize what we speak and convert it into machine readable format. For this, the microphone of our PC is used, for which PyAudio library is required.
- Before listening, it calibrates the background noise level for 1 second to get a clearer recording of what we say.
- The recorded sound is converted to string by using Google Speech recognition library. Internet connection is required.

**For information on library:** Refer Pg. No 35

## 2.6 Google Translate:

**Library Used:** googletrans

**12 Languages Available:**

- English
- Hindi
- Gujrati
- French
- German
- Italian
- Spanish
- Polish
- Russian
- Japanese
- Chinese
- Korean

**Snippets of Code:**

```
# Select input language:
text_to_speech("Select input language:")
tr_in_data = SpeechCommand().lower()

# Select output language:
text_to_speech("Select output language:")
tr_out_data = SpeechCommand().lower()

# Conversion:
translator = Translator()
t= translator.translate(text = text, src = tr_in_data, dest = tr_out_data)
writeToLog(f"Translated: {t.text},{t.src},{t.dest}")
tts = gTTS(text = t.text, lang=tr_in_data,slow=False)
```

**Explanation:**

- This is the flagship feature of F.R.I.D.A.Y. Using it we can translate any of the given 12 languages into any of the remaining 11 languages.
- It uses Google Translate API and thus gives high accuracy.
- Output is provided in both text and speech formats.

**For information on library:** Refer Pg. No 39

## 2.7 Wikipedia Search:

**Library Used:** wikipedia

**Code:**

```
def wiki():
    query = "Defined to avoid error!"
    text_to_speech(" What should I search for?")
    query=SpeechCommand()
    text_to_speech("Searching.....")
    #query = query.replace("wikipedia","")
    r = query
    result = wikipedia.summary(query,sentences=2)
    writeToLog(result)
    text_to_speech(result)
    text_to_speech("Do you want a full wikipedia content? Yes or No")
    query=SpeechCommand().lower()
    if 'yes' in query:
        out = wikipedia.page(r).content
        writeToLog(out)
    elif 'no' in query:
        text_to_speech("Quiting wikipedia")
```

**Explanation:**

- This feature enables F.R.I.D.A.Y. to search anything on Wikipedia.
- An option of receiving the answer in summary form or long answer form is also provided by us.
- This uses the wikipedia library API to search for information on wikipedia and then copy information in string format from wikipedia and provide it to text to speech function.

**For information on library:** Refer Pg. No 35

## 2.8 Email Functionality:

**Library Used:** smtplib

**Code:**

```
def send_email():
    try:
        text_to_speech("What should i say?")
        content = SpeechCommand()
        receiver_id = '17ecuot013@ddu.ac.in'
        server = smtplib.SMTP('smtp.gmail.com',587)
        server.ehlo()                                   # check connection
        server.starttls()                               # Puts the connection to the SMTP
 server into TLS mode.
        server.login('17ecuog026@ddu.ac.in','Iwontforget11')
        server.sendmail('17ecuog026@ddu.ac.in',receiver_id,content)
        server.close()
        text_to_speech("Email sent successfully to Sahil...")
    except Exception as e:
        writeToLog(e)
        text_to_speech("Unable to send email. Please retry...")
```

**Explanation:**

- This feature enables F.R.I.D.A.Y. to send email to anyone from our email address.
- This feature uses the smtplib library to open secure TLS connection on Gmail and then login using our credentials and then send email to the given mail id.

**For information on library:** Refer Pg. No 35

## 2.9 Google Search:

**Library Used:** webbrowser

**Code:**

```
def chrome():
    text_to_speech("What should i search ?")
    googlechromepath = "C:/Program Files (x86)/Google/Chrome/Application/chrome.exe %s"
    search = SpeechCommand().lower()
    wb.get(googlechromepath).open_new_tab(search)
```

**Explanation:**

- This feature enables F.R.I.D.A.Y. to send anything on Google.com in the Google chrome application.
- Google gives the vastest results for any search and thus we can obtain whatever we need with ease.

**For information on library:** Refer Pg. No 36

## 2.10 PC Shutdown:

**Library Used:** os

**Code:**

```
# Logout:
def f_logout():
    text_to_speech(" Do you really want to Logout ??")
    query = SpeechCommand().lower()
    if 'yes' in query:
        os.system("logout -1")
    else:
        text_to_speech("Okay. Will not Logout.")
# SHutdown:
def f_shutdown():
    text_to_speech(" Do you really want to ShutDown ??")
    query = SpeechCommand().lower()
    if 'yes' in query:
        os.system("shutdown /s /t 1")
    else:
        text_to_speech("Okay. Will not ShutDown.")
# Restart:
def f_restart():
    text_to_speech(" Do you really want to Restart ??")
    query = SpeechCommand().lower()
    if 'yes' in query:
        os.system("shutdown /r /t 1")
    else:
        text_to_speech("Okay. Will not Restart")
```

**Explanation:**

- This feature enables F.R.I.D.A.Y. to Shut down, Restart or Log out from our PC.
- This feature uses the os library to give commands to the computer and it immediately responds.
- Thus, we have inserted a layer of protection where the user has to confirm the request of turning our PC off.

**For information on library:** Refer Pg. No 36

## 2.11 Offline Songs:

**Library Used:** os

**Code:**

```
# Play Songs:
def songs():
    songs_dir = 'E:\\MusicPA'
    songs = os.listdir(songs_dir)
    os.startfile(os.path.join(songs_dir,songs[0]))
```

**Explanation:**

- This feature enables F.R.I.D.A.Y. to play songs stored on our computer.
- This feature uses the os library to make a list of all the songs present in the mentioned directory and then start playing the files from the beginning using the default application for mp3 files.
- Here, we have to mention the path of the directory beforehand.

**For information on library:** Refer Pg. No 36

## 2.12 Take Notes:

**Library Used:** None.

**Code:**

```
# Take Notes:
def notes():
    text_to_speech("What should i remember?")
    data = SpeechCommand()
    remember = open('data.txt', 'w',encoding='utf-8')
    remember.write(data)
    remember.close()

# Extract Note:
def extract():
    text_to_speech("Here is the extracted data:")
    extract = open('data.txt', 'r',encoding='utf-8')
    text_to_speech(extract.read())
```

**Explanation:**

- This feature enables F.R.I.D.A.Y. to take notes, save them to a file.
- Then whenever it extracts the information and tells it back to us whenever we require it.
- Python has an in-built function which helps us read from and write to files.
- We have used a txt file for storage.

**For information on library:** N/A

## 2.13 PC Status:

**Library Used:** psutil

**Code:**

```
def cpu():
    usage = str(psutil.cpu_percent())
    text_to_speech("CPU is at " + usage + "percent")
    battery = psutil.sensors_battery()
    text_to_speech("Battery is at :" )
    text_to_speech(battery.percent)
    text_to_speech("percent")
```

**Explanation:**

- This feature enables F.R.I.D.A.Y. to get CPU usage information and the battery information among other things.
- This feature uses the psutil library to get information from the processor.
- Various other information about the PC is also available which we'll add in the future upgrades.

**For information on library:** Refer Pg. No 37

## 2.14 Jokes:

**Library Used:** pyjokes

**Code:**

```
def jokes():
    text_to_speech(pyjokes.get_joke())
```

**Explanation:**

- This feature enables F.R.I.D.A.Y. to tell us jokes related to programmers.
- This feature uses the pyjokes library to import jokes from online storage.
- Hence this feature can be used to lighten our mood.

**For information on library:** Refer Pg. No 38

## 2.15 YouTube Search:

**Library Used:** webbrowser

**Code:**

```
def youtube():
    text_to_speech("What should I search?")
    search_youtube = SpeechCommand().lower()
    text_to_speech("Here we go to YouTube")
    wb.open('https://www.youtube.com/results?search_query='+search_youtube)
    quit()
```

**Explanation:**

- This feature enables F.R.I.D.A.Y. to search YouTube and show us the results on the default web browser of our PC.
- This feature uses the webbrowser library to open YouTube and search in it and show us the results.

**For information on library:** Refer Pg. No 36

## 2.16 Screenshot:

**Library Used:** pyautogui

**Code:**

```
def screenshot():
    img = pyautogui.screenshot()
    img.save("C:\\Users\\Rutvij Shah\\Desktop\\PA-Python\\Personal-
Assistant\\SS.png")
    text_to_speech("Screenshot successfully taken")
```

**Explanation:**

- This feature enables F.R.I.D.A.Y. to take screenshots.
- This feature uses the pyautogui library to take a full screen screenshot and save it at the pre mentioned destination.
- In future, we'll add an upgrade to take partial screen screenshots.

**For information on library:** Refer Pg. No 37

## 2.17 PC Applications:

**Library Used:** os

**Code:**

```
# WPS office:
def wps():
    text_to_speech("Opening WPS Office.....")
    wps_office = r'C:/Users/patel/AppData/Local/Kingsoft/WPS Office/ksolaunch.exe'
    os.startfile(wps_office)

# Adobe Acrobat:
def adobe():
    text_to_speech("Opening Acrobat Reader.....")
    acrobat_read = r'C:/Program Files (x86)/Adobe/Acrobat Reader DC/Reader/AcroRd32.exe'
    os.startfile(acrobat_read)

# Snipping Tool:
def sniptool():
    text_to_speech("Opening Snipping Tool.....")
    snipping_tool = r'C:/WINDOWS/system32/SnippingTool.exe'
    os.startfile(snipping_tool)
```

**Explanation:**

- This feature enables F.R.I.D.A.Y. to open pre-installed applications on our PC.
- This feature uses the os library to open the application.
- Here, we have to mention the path of all the applications beforehand.

**For information on library:** Refer Pg. No 36

## 2.18 Latest News:

**Library Used:** json

**Categories Available:**

- India
- World

**Subcategories Available:**

- Miscellaneous Headlines
- Technology
- Sports
- Business
- Entertainment
- Money
- Politics
- Health
- Lifestyle
- Cricket
- Football

**Snippet of the Code:**

```
if 'India' in h_data:
        try:
        jsonObj = urlopen("https://newsapi.org/v2/topheadlines?country=in&so
rtBy=popularity&apiKey=7071cb067cff46369f40d7b91e09a7df")
        newsdata = json.load(jsonObj)
       i=1
       text_to_speech("Here are some top headlines from India.")
       writeToLog("**********HEADLINES**********")

       for item in newsdata['articles'] & i<6:
       writeToLog(str(i)+'. '+item['title']+'\n')
       writeToLog(item['description']+'\n')
       text_to_speech(item['title'])
       i = i + 1
       except Exception as e:
       writeToLog(str(e))
```

**Explanation:**

- This feature enables F.R.I.D.A.Y. to search the web for latest news and give us the results.
- This feature uses the json library to parse data from the internet and give it to us.
- Here, we can choose from many different topics to get specific news according to our interest.

**For information on library:** Refer Pg. No 38

## 2.19 Random Question:

**Library Used:** wolframealpha

**Code:**

```
def question():
    text_to_speech("Ask me a random question!")
    query = SpeechCommand().lower()
    client = wolframalpha.Client('******-********')
    res = client.query(query)
    answer = next(res.results).text
    writeToLog('The Answer is :'+answer)
    text_to_speech('The Answer is : '+answer)
```

**Explanation:**

- This feature enables F.R.I.D.A.Y. to answer to any random question.
- This feature uses the wolframalpha library API to help answer any question.
- Here, we have to have a wolframalpha client number beforehand.

**For information on library:** Refer Pg. No 40

## 2.20 Location:

**Library Used:** webbrowser

**Code:**

```
def findloc():
    text_to_speech(" What should I locate on the map?")
    query=SpeechCommand()
    try:
        #query = query.replace("find","")
        location = query
        text_to_speech("User asked to locate"+location)
        wb.open_new_tab("https://www.google.com/maps/place/"+location)
    except Exception as e:
        writeToLog(str(e))
```

**Explanation:**

- This feature enables F.R.I.D.A.Y. to search any place and give us its location on Google Maps.
- This feature uses the webbrowser library to open Google Maps on our browser and then search in it.

**For information on library:** Refer Pg. No 36

## 2.21 Calculator:

**Library Used:** wolframealpha

**Code:**

```
# Calculate:
def calculate():
    text_to_speech(" What should I calculate?")
    query=SpeechCommand()
    client = wolframalpha.Client(wolframalpha_app_id)
    res = client.query(query)
    answer = next(res.results).text
    writeToLog('The Answer is :'+answer)
    text_to_speech('The Answer is : '+answer)
```

**Explanation:**

- This feature enables F.R.I.D.A.Y. to answer to any calculative question.
- This feature uses the wolframalpha library API to search the answer on its online interface and give us the calculated answer.
- Here, we have to have a wolframalpha client number beforehand.

**For information on library:** Refer Pg. No 40

## 2.22 Weather:

**Library Used:** wolframealpha

**Code:**

```
def weather():
    text_to_speech(" Weather of which location?")
    query=SpeechCommand()
    client = wolframalpha.Client(wolframalpha_app_id)
    res = client.query(query)
    answer = next(res.results).text
    writeToLog('The Answer is :'+answer)
    text_to_speech('The Answer is : '+answer)
```

**Explanation:**

- This feature enables F.R.I.D.A.Y. to provide us the latest weather of any location.
- This feature uses the wolframalpha library API to search the location and then extract information regarding to its weather.
- Here, we have to have a wolframalpha client number beforehand.

**For information on library:** Refer Pg. No 40

## 2.23 GUI Application Layer:

**Library Used:**

**Snippets of the Code:**

```
# Main Window
root = Tk()
root.title("F.R.I.D.A.Y")

# Defining fonts:
headfont = font.Font(family='Helvetica', name='HEADINGfont', size=10, weig
ht='bold',slant='italic')

# Frames:
main_frame = Frame(root,relief=GROOVE,bd=1)

# Labels:
pc_label = Label(pc_frame, text="PC Options",fg="Red",font=headfont)

# Buttons:
logout_button = Button(pc_frame, text="Log out",activebackground = "Cyan",f
g="Blue",command=f_logout,bd=0)

# Outout LOG screen:
log = Text(root, state='disabled',height=10,width=45,bg="black",fg="red",wr
ap=WORD)

# Grid Placement:
apps_label.grid(row=6,column=0,pady=4,padx=2,sticky="nsew")
wps_button.grid(row=7, column=0,pady=2,padx=2,sticky="nsew")

root.mainloop()
```

**Explanation:**

- This feature enables F.R.I.D.A.Y. to have a GUI.
- We used the in-built Tkinter library to build this Graphical User Interface for ease of access.
- Using this F.R.I.D.A.Y. behaves and looks like an application rather than a standalone python code.

**For information on library:** Refer Pg. No 41

## 2.24 Applications and Future Work:

### Applications:

- We can use this in the background while our PC is on.
- Whenever we need to do any task, we can call upon F.R.I.D.A.Y to help us do it.
- We have already discussed its many features and after a few upgrades to smooth over the glitches and add some vital new features it can behave as a very efficient personal assistant.

### Future Work:

- Store the data of the user and also the usage logs in an Excel worksheet from where it can be systematically stored and accessed whenever required.
- Enable F.R.I.D.A.Y to store information of more than one user and customize itself based on the user which is interacting with it.
- Make an exe application which can be easily installed across various PCs and be distributed and used with ease.
- We further aim to deploy it on a Raspberry Pi, so that it becomes portable and has even more versatile uses.

# 3. List of Python Libraries Used in the Project:

## 3.1 pyttsx3:

- **pyttsx3** is a text-to-speech conversion library in Python. Unlike alternative libraries, it works offline, and is compatible with both Python 2 and 3.
- Works without internet connection or delay.
- Supports multiple TTS engines, including Sapi5, nsss, and espeak. These engines are already present in your OS and are not required to be downloaded.

❖ **Features:**

- Fully offline text to speech conversion
- Control speed/rate of speech
- Choose among different voices installed in your system
- Simple, powerful, & intuitive API
- Save the speech audio as a file

## 3.2 datetime:

- Encapsulation of date/time values.
- DateTime objects represent instants in time and provide interfaces for controlling its representation without affecting the absolute value of the object.
- DateTime objects may be created from a wide variety of string or numeric data, or may be computed from other DateTime objects. DateTimes support the ability to convert their representations to many major time zones, as well as the ability to create a DateTime object in the context of a given time zone.
- DateTime objects may be converted to integer, long, or float numbers of days since January 1, 1901, using the standard int, long, and float functions.
- The date component consists of year, month, and day values.
- The time component consists of hour, minute, and second values separated by colons.

## 3.3 speech_recognition:

- It is a library to convert audio to string in a particular language.
- PyAudio Library is used to record speech through microphone. And then the audio file sent to Google Speech Converter function.
- Google Speech Converter function converts the audio into string of a particular language.
- There are other Speech Recognizers available, but Google's translation is generally the most accurate, and it offers option of many languages.
- Using it we have enabled out P.A to respond to basic queries about current time and date.

## 3.4 wikipedia:

- Wikipedia is a multilingual online encyclopedia created and maintained as an open collaboration project[4] by a community of volunteer editors using a wiki-based editing system. It is the largest and most popular general reference work on the World Wide Web
- **Wikipedia** is also a Python library that makes it easy to access and parse data from Wikipedia (encyclopedia).
- Search Wikipedia, get article summaries, get data like links and images from a page, and more. Wikipedia wraps the MediaWiki API so you can focus on using Wikipedia data, not getting it.

## 3.5 smtplib:

- smtplib — **SMTP protocol client**
- The smtplib module defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP or ESMTP listener daemon.
- An SMTP instance encapsulates an SMTP connection. It has methods that support a full repertoire of SMTP and ESMTP operations.

- **SMTP Objects used by us:**

1. SMTP.login(user, password, *, initial_response_ok=True)
2. SMTP.starttls(keyfile=None, certfile=None, context=None)
3. SMTP.sendmail(from_addr, to_addrs, msg, mail_options=(), rcpt_options=())
4. SMTP.quit()

## 3.6 webbrowser:

- **webbrowser — Convenient Web-browser controller**
- The webbrowser module provides a high-level interface to allow **displaying Web-based documents to users.**
- The script webbrowser can be used as a command-line interface for the module. It **accepts a URL as the argument.**
- The webbrowser module can be used to launch a browser in a platform-independent manner or open a particular browser and search the required information.
- To open a page in a specific browser(e.g. Chrome), we used the **webbrowser.get()** function to specify Google Chrome.

## 3.7 os:

- **os — Miscellaneous operating system interfaces**
- OS is a python library which provides a portable way of using operating system dependent functionality.
- Command used: **os.system()**
- Execute the command (a string) in a subshell. This is implemented by calling the Standard C function system(), and has the same limitations. If command generates any output, it will be sent to the interpreter standard output stream.

Usage:

**1) Define path of directory**

**2) os.listdir(path='.')**

- Return a list containing the names of the entries in the directory given by path. Thus, a list of files present in the current folder is made by us.

**3) os.startfile(path[, operation])**

- Start a file with its associated application.
- When operation is not specified or 'open', this acts like double-clicking the file in Windows Explorer, or giving the file name as an argument to the start command from the interactive command shell: the file is opened with whatever application (if any) its extension is associated. startfile() returns as soon as the associated application is launched.
- Here, an application depending on extension of the file is launched.

## 3.8 pyautogui:

- PyAutoGUI is a cross-platform GUI automation Python module for human beings. Used to programmatically control the mouse & keyboard.
- The three major operating systems (Windows, macOS, and Linux) each have different ways to programmatically control the mouse and keyboard. This can often involve confusing, obscure, and deeply technical details. The job of PyAutoGUI is to hide all of this complexity behind a simple API.

- Example Uses of pyautogui:
  1. Keyboard and Mouse Control
  2. Display Message Boxes
  3. Screenshot Functions
  4. locate where an image is on the screen:

## 3.9 psutil:

- **psutil (process and system utilities)** is a cross-platform library for retrieving information on running processes and system utilization (CPU, memory, disks, network, sensors) in Python. It is useful mainly for system monitoring, profiling and limiting process resources and management of running processes.

- Usage:

1. CPU & Process management
2. Memory & Disks
3. Network
4. Sensors
5. Other system info
6. Windows services, etc.

## 3.10 pyjokes:

- One-line jokes for programmers (jokes as a service)

**How to use?**

- Import the pyjokes module in a Python file and use the get_joke function to easily drop a random joke into your application:

**Functions:**

- **pyjokes.get_joke() :**Returns a random joke from the given category in the given language.
- **pyjokes.get_jokes() :**Returns a list of jokes from the given category in the given language.

**Parameters:**

- 1) language: 'en', 'de', 'es', 'gl', 'eus'
- 2) category: 'neutral', 'chuck', 'all'

## 3.11 json:

- json — JSON encoder and decoder library
- JSON (JavaScript Object Notation), specified by RFC 7159 (which obsoletes RFC 4627) and by ECMA-404, is a lightweight data interchange format inspired by JavaScript object literal syntax (although it is not a strict subset of JavaScript 1 ).
- json exposes an API familiar to users of the standard library marshal and pickle modules.

**Uses:**

- Encoding basic Python object hierarchies
- Compact encoding
- Decoding JSON
- Pretty printing
- Specializing JSON object decoding

## 3.12 requests:

- Requests allows you to send HTTP/1.1 requests extremely easily. There's no need to manually add query strings to your URLs, or to form-encode your *PUT* & *POST* data.
- Requests is ready for the demands of building robust and reliable HTTP–speak applications, for the needs of today.

- Uses:
  - International Domains and URLs + Keep-Alive & Connection Pooling
  - Sessions with Cookie Persistence + Browser-style SSL Verification
  - Basic & Digest Authentication + Familiar Cookies
  - Automatic Decompression of Content + Automatic Content Decoding
  - Automatic Connection Pooling + Unicode Response Bodies
  - Multi-part File Uploads + SOCKS Proxy Support
  - Connection Timeouts + Streaming Downloads
  - Automatic honoring of Chunked HTTP Requests

## 3.13 googletrans:

- Googletrans is a free and unlimited python library that implemented **Google Translate API**. This uses the Google Translate Ajax API to make calls to such methods as detect and translate.

**Features:**

- Fast and reliable - it uses the same servers that translate.google.com uses
- Auto language detection
- Bulk translations
- Customizable service URL
- HTTP/2 support

## 3.14 gTTS:

**gTTS** (*Google Text-to-Speech*), a Python library and CLI tool to interface with Google Translate's text-to-speech API. We can write spoken mp3 data to a file, a file-like object (byte string) for further audio manipulation, or stdout. Or simply pre-generate Google Translate TTS request URLs to feed to an external program

**Features**

- Customizable speech-specific sentence tokenizer that allows for unlimited lengths of text to be read, all while keeping proper intonation, abbreviations, decimals and more;
- Customizable text pre-processors which can, for example, provide pronunciation corrections;
- Automatic retrieval of supported languages.

## 3.15 playsound:

- Pure Python, cross platform, single function module with no dependencies for playing sounds.
- It requires one argument - the path to the file with the sound you'd like to play. This may be a local file, or a URL.
- There's an optional second argument, block, which is set to True by default. Setting it to False makes the function run asynchronously

## 3.16 wolframalpha:

- **Python Client built against the WolframAlpha v2.0 API.**
- We can Compute expert-level answers using Wolfram's breakthrough algorithms, knowledgebase and AI technology.

**Usage:**

- Basic usage is pretty simple. Create the client with your App ID (request from Wolfram Alpha):
- Then, you can send queries, which return Result objects
- Result objects have pods (a Pod is an answer group from WolframAlpha)

All objects returned are dictionary subclasses, so to find out which attributes Wolfram|Alpha has supplied, simply invoke .keys() on the object.

## 3.17 Tkinter:

- Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit, and is Python's de facto standard GUI. The name Tkinter comes from Tk interface.

  **To create a Tkinter app:**
  1. Importing the module – tkinter
  2. Create the main window (container)
  3. Add any number of widgets to the main window
  4. Apply the event Trigger on the widgets.

- Tkinter is implemented as a Python wrapper around a complete **Tcl interpreter** embedded in the Python interpreter. Tkinter calls are translated into Tcl commands which are fed to this embedded interpreter, thus making it possible to mix Python and Tcl in a single application.
- There are several popular GUI library alternatives available, such as **wxPython, PyQt, PySide, Pygame, Pyglet, and PyGTK.**

## 4. Video Links:

| Topic | Video Link |
|---|---|
| Text to Speech | https://drive.google.com/file/d/1PuKZ2yZ-1wcWOsFImn0YPjxhaQNMFGV0/view?usp=sharing |
| Unique Greeting | https://drive.google.com/file/d/1Bv5wBOnTEVM26IsxnhfYwpj5LM_H0SpW/view?usp=sharing |
| Tell Date and Time | https://drive.google.com/file/d/1m5If1yG9Vi997yd0tPIpcy-L_ltVReOr/view?usp=sharing |
| Accurate Speech Recognition | https://drive.google.com/file/d/1wrZAPdCAKnc5ZWbIyI1TQtcXX2bxfQ2V/view?usp=sharing |
| Translate feature | https://drive.google.com/file/d/1Q4BPQdnItzoIx1AakMnFsTuGBXwoXOS9/view?usp=sharing |
| GUI Application | https://drive.google.com/file/d/1n7reBUIyDfPEfKKEcZssLcZjiPQFjfVM/view?usp=sharing |
| Wikipedia Search | https://drive.google.com/file/d/1y5Yq_MeWDmIjS55l6ZH6iQZEl0uhUxFF/view?usp=sharing |
| Google Search | https://drive.google.com/file/d/1vaALAsmHTIHidNFxAbnUUD010IjdmXlT/view?usp=sharing |
| YouTube Search | https://drive.google.com/file/d/1npoqYAeXxLZ-r4XJ04U0H2dGDDewFrOP/view?usp=sharing |
| Location Search | https://drive.google.com/file/d/1-zh2jCLj94UcZASBkHCUZp5VNO3zOYBE/view?usp=sharing |
| Latest News Update | https://drive.google.com/file/d/1FTbscGyWix6Jrr5u9ccse4UwbnvkAUME/view?usp=sharing |
| Jokes for Programmers | https://drive.google.com/file/d/11s89KaQSxQvzxS63pqXhGK-7POFTKFAc/view?usp=sharing |
| Custom Email Sending | https://drive.google.com/file/d/13OaNaiN762Ap4XtQK8sqdQMmAB_peIy3/view?usp=sharing |
| Random Question | https://drive.google.com/file/d/1t3bebx6-H8rR27c1sj7CekxeSuLURyla/view?usp=sharing |
| Open PC Application | https://drive.google.com/file/d/1HA1dYCx7c1h-HpK7VuhVBZagTBOhcL9W/view?usp=sharing |
| Play offline Songs | https://drive.google.com/file/d/1IWvxN8LydWAdfhKCuQ49ZRra8vM7octI/view?usp=sharing |
| CPU and Battery Status Update | https://drive.google.com/file/d/1hfgXRWVe-rko0F0H7bUlUZRoWFcXOTdm/view?usp=sharing |

| | |
|---|---|
| Take screenshot | https://drive.google.com/file/d/1CkK1ESuVBQp8SFsTCvUxLGjiD8S8EHtv/view?usp=sharing |
| Take Notes | https://drive.google.com/file/d/1R9A960_FSSJmOPslstLsPAbEnh4Doexl/view?usp=sharing |
| Shutdown PC | https://drive.google.com/file/d/1xAN4h3DVis5Gw9celR5anPU8LEAvkjOz/view?usp=sharing |
| All Videos | https://drive.google.com/drive/folders/1waeXobB9-kQPyTVn-KvimIFodJd8dp1T?usp=sharing |

## 5. Conclusion:

In this term project, we had to develop something using only software. Thus, using Python language we made a Personal Assistant named F.R.I.D.A.Y. We also included many features and made it almost production ready. Our flagship feature is the Translate feature, where we can translate to and from, from up to 12 languages. We also made a GUI layer for it to provide ease of access and better readability. We tested it in various ways and also uploaded videos for demonstration. Thus, we conclude that we have made a decent Personal Assistant with a vast array of features in the time provided to us.

## 6. References:


- https://pypi.org/project/gTTS/
- https://pypi.org/project/googletrans/
- https://github.com/nateshmbhat/pyttsx3
- https://docs.python.org/3/library/smtplib.html
- https://docs.python.org/3/library/os.html