Project Title – Third Eye Assistant

A report submitted to

Techno India University

For the partial fulfilment of

Bachelor of Technology (B. Tech)

Degree in

Computer Science Engineering (CSE)

by

Group Members: -

Amit Dutta| |ID: 181001011010

Subham Kundu| |ID: 171001001045

Adibrata Sinha| |ID: 171001001044

Bratati Banerjee| |ID: 181001011013

Anjan Das| |ID: 171001001019

Supervisor Sir: -

Prof. Sambhunath Biswas

Department of

Computer Science Engineering

EM 4, Sector V, Salt Lake, Kolkata - 700091, West Bengal, India.
Phone- +91 33 2357 6163, FAX - +91 33 2357 1097.
www.technoindiauniversity.ac.in

Date: 2nd July, 2021

Abstract:

There are enormous range of technologies available today, yet there is no technological standard beyond the guide dog or the cane for the visually impaired persons. Therefore, we intend to help the visually impaired persons by developing a technology that can be their assistant in their day to day life. The Third Eye Assistant is a Python based group project which has Artificial Intelligence implementation in it. We have implemented object and face detection, which will help the user know about the objects in his/her surroundings. Apart from that, we have a technology that will help the user to listen to news from top websites and guide to a particular location through map and direction prompts. This is going through a development phase, we have many more ideas to implement, and we will surely try to take them forward as we learn new things in future.

Contents

Introduction:

Technological advancements have created new opportunities for enhancing the quality of life for people with various disabilities. The “Third Eye” stemmed from two distinct routes: one ideological, and the other came from a technical challenge. The ideological route is simple. How is it possible that despite the enormous range of technologies available today, there is still no technological standard beyond the guide dog and the cane for the visually impaired.

We are firm believers in the “humanist technology”. Regarding the technical challenge, there was one key question: How can we provide spatial information to a blind person in a way that is both easy to understand and instantaneous? We are passionate about challenges, and it was while seeking a solution to both these questions that we originated with Third Eye Assistant.

Related Work: ~

The whole project can be divided into **Three** main segments: -

- Text Recognition
- Object Detection
- Voice based assistant

Whole project link(GitHub) - https://github.com/Third-Eye-Assistant/TEA

Text Recognition: ~

Segments	**Description**	**Status**
Text Extraction	Extraction of text from an object or surface as captured on camera.	Completed
Text Summarisation	Text summarization is the process of creating a short, accurate, and fluent summary of a longer text document. It is the process of distilling the most important information from a source text. Automatic text summarization is a common problem in machine learning and natural language processing (NLP). Automatic text summarization methods are greatly needed to address the ever-growing amount of text data available online to both better help discover relevant information and to consume relevant information faster.	Completed

Object Detection: -

Segments	Description	Status
Stationary Object Detection	Detects objects from pictures or videos and shows the type or name of the object from its trained set of data. Some modules that we have used here are **OpenCV**, numpy, and we have also used an architecture named **MobileNetV3**, which is a convolutional neural network architecture to detect objects from pre-fed data.	Completed (Trained data of about **91** objects, still adding more)

Voice Based Assistant: -

Segments	Description	Status
Voice Command Interpreter	This will decode the user's voice to follow his/her commands(specific) and work accordingly. We have used a python module named "**speech_recognition**" as a transcriber.	Completed
Voice Alerts	This will notify the user about the objects around them and also will prompt them any text messages if required/detected. We have used a python module named "**pyttsx3**", which helps in text to speech conversion.	Completed

Objective of Proposed Work: ~

Our goal is to expand blind people's mobility and independence. The main goal of "Third Eye Assistant" is to help blind people by introducing a new technology that makes them able to read the typed text through text recognition and provides the technology to scan any written text and convert it into audio and keep aware of people and current obstacles in their path by providing image recognition and a voice assistant for searching their basic queries.

Proposed Work: ~

We are going to use Python as our programming language. Dividing our project into three parts.

- Text Recognition
- Object Detection
- Voice Based Assistant

OBJECT DETECTION

✔ First, we are going to get images as input from camera module. After getting the input we are going to use OpenCV (Open Source Computer Vision Library) to break the images into 3D Matrix. **OpenCV i**s a library of programming functions mainly aimed at real-time computer vision. Using this library, we have loaded face detection model in a video stream. Specially It helps us to show a proper output with a video frame. By using OpenCV we are going to break the image into Matrix and scaled down the image for fast and accurate performance.

✔ The values in the Matrix are nothing but pixels representation of an image where the first 2 dimensions of the image will be represented as pixel and the last value is mainly the colour. If the image is black and white the value will be from 0-255 or if it is coloured image then it will be in RGB format (0-255, 0-255, 0-255).

✔ We also have to classify certain images by using DNN (Deep Neural Network) Algorithm with the help of an architecture named **MobileNetV3**. This is tuned to mobile phone CPUs through a combination of hardware-aware network architecture search (NAS) complemented by the NetAdapt algorithm and then subsequently improved through novel architecture advances. These automated search algorithms and network design can work together to harness complementary approaches improving the overall state of the art.

TEXT RECOGNITION

- ✔ Firstly, we are taking an image from the webcam, then the image is being processed for getting better outcomes. Image processing includes the checking of valid image, i.e we check if the image is blurry or not. If the image is prominent, we follow few steps: -
  1. **Grey Scale** - converts image into Gray Scale
  2. **GaussianBlur** - it uses Gaussian filter which is low pass filter which removes the high frequency components.
  3. **Canny Edge detection** - Canny Edge Detection is used to detect the edge in an image. It takes a grey scale image and uses multistage algorithm to find the edges.
  4. **Finding the contour area** - Contours are defined as the line joining all the points along the boundary of an image that are having the same intensity, it's very handy for shape analysis.
  5. **Arc length** - According to the contours, we run a loop to calculate the Arclength.
  6. **ApproxPolyDp** - is used to approximate the shape of polygonal curves to the specified precision called approxPolyDP.
  7. And it returns the approximated contour whose shape is the same as the input curve. Approximation of a shape of the contour is widely used in the area of robotics to classify the patterns and analysis of scenes.
  8. Then we get the target value which has length of 4 then sent it to **mapper function**.
  9. In mapper function, we reshape the array into 4 X 2 size.
  10. Then we have created a **np zeros list,** and using argmax (Argument Maximum) and argmin (Argument Minimum) we get all the max and min values.
  11. Then we return all 4 edges.
  12. After Getting approx. 4 values by using **get_prespectiveImage** we convert it into viewable image and return all 4 probable edges.
  13. After getting 4 edges we can get only that image that can be useful for us.
  14. Then we use **image classification** to detect what kind of image it is.
  15. We trained 4 kinds of images (Readable, Road Sign, ID, Posters/Covers) using **tensorflow** python module.
  16. Then we fetch the text from both input image and image 2 (After Image Processing).
  17. Then if it's Readable then we store it and implement some basic logic to get the best outcome form and the image.

✔ If the person wants to summarize the data or want to get only important information, we can add the processed data using **Natural Language Processing** by the help of **Spacy Module. spaCy** is a free, open-source library for NLP in Python. It's written in **Cython** and is designed to build information extraction or natural language understanding systems. It's built for production use and provides a concise and user-friendly API.

VOICE BASED ASSISTANT

✔ We are going to get input from microphone. Here we are using a python module named **speech_recognition**. We can import speech_recognition as sr. This is used to convert speech to text and the one and only class we need is the Recognizer class from the speech_recognition module. Depending upon the underlying API used to convert speech to text, the Recognizer class has following methods: recognize_bing() : Uses Microsoft Bing Speech API.

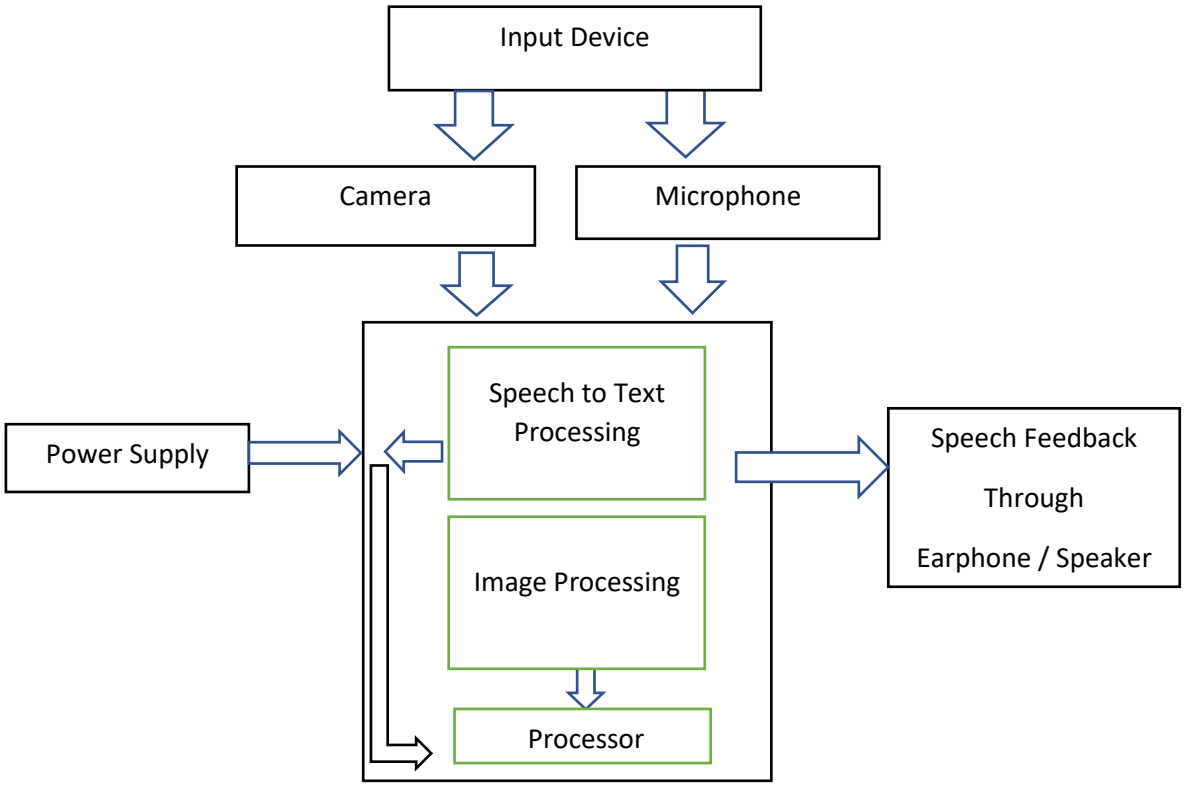
✔ Using python, we are going to create a voice assistant, and in addition, the voice assistant can react to voice commands and give the user relevant information about their inquiry. We have used another python module named **pyttsx3**. This is a **text-to-speech** conversion library in Python. Unlike alternative libraries, it works offline and is compatible with both Python 2 and 3. An application invokes the pyttsx3. init() factory function to get a reference to a pyttsx3.

The modules used in the Voice assistant: -

```
import os
import speech_recognition as sr
import pyttsx3
from tkinter import *
from tkinter import ttk
from tkinter import messagebox
from tkinter import colorchooser
from PIL import Image, ImageTk
from time import sleep
from threading import Thread
```

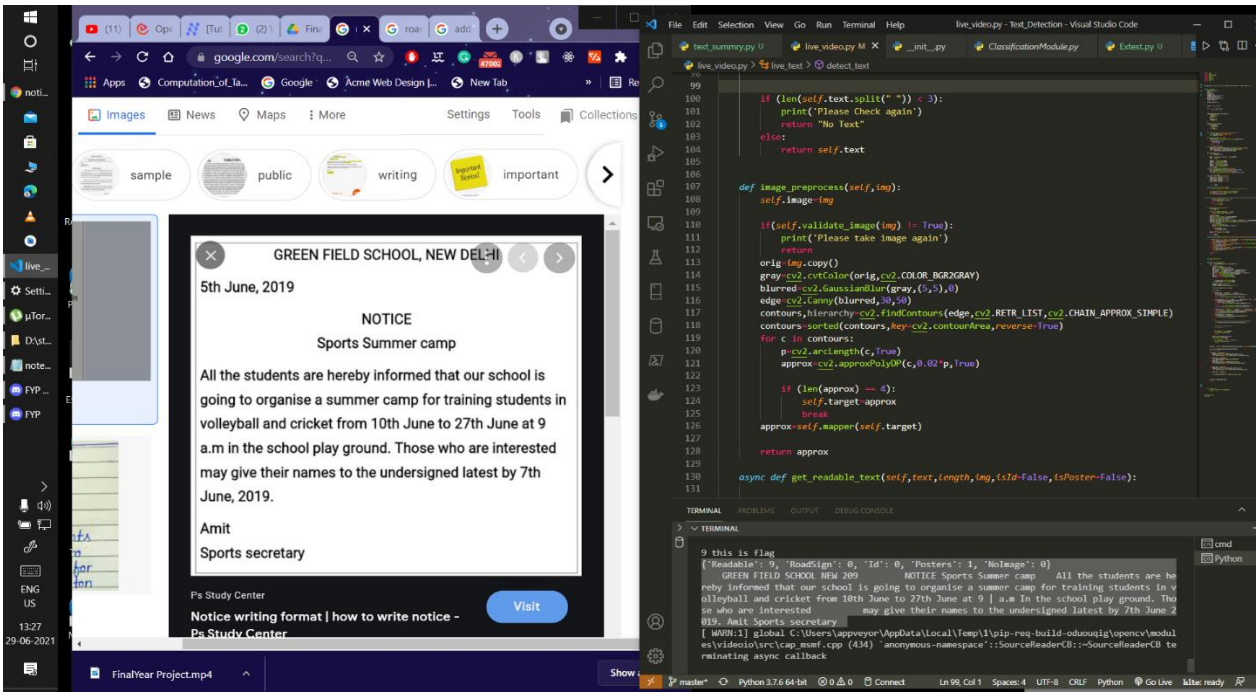
Experimental Details: ~

Block Diagram



Text Recognition:

Live Text Detection: -



Sample Input: -

GREEN FIELD SCHOOL, NEW DELHI

5th June, 2019

NOTICE

Sports Summer camp

All the students are hereby informed that our school is going to organise a summer camp for training students in volleyball and cricket from 10th June to 27th June at 9 a.m in the school play ground. Those who are interested may give their names to the undersigned latest by 7th June, 2019.

Amit

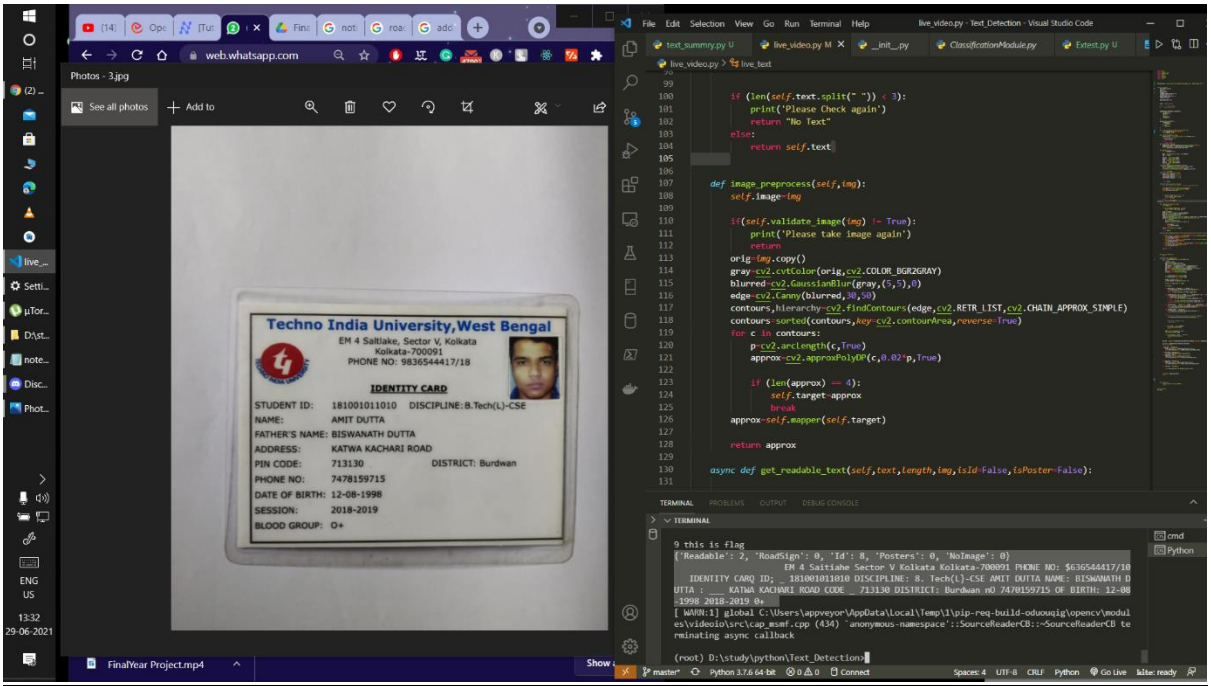
Sports secretary

Sample Output: -

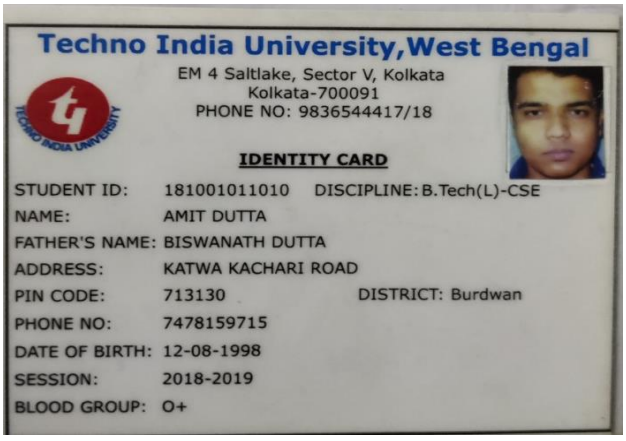
{'Readable': 9, 'RoadSign': 0, 'Id': 0, 'Posters': 1, 'NoImage': 0}

GREEN FIELD SCHOOL NEW 209 NOTICE Sports Summer camp All the students are hereby informed that our school is going to organise a summer camp for training students in volleyball and cricket from 10th June to 27th June at 9 | a.m In the school play ground. Those who are interested may give their names to the undersigned latest by 7th June 2019. Amit Sports secretary

Identity Card Detection: -



Sample Input: -



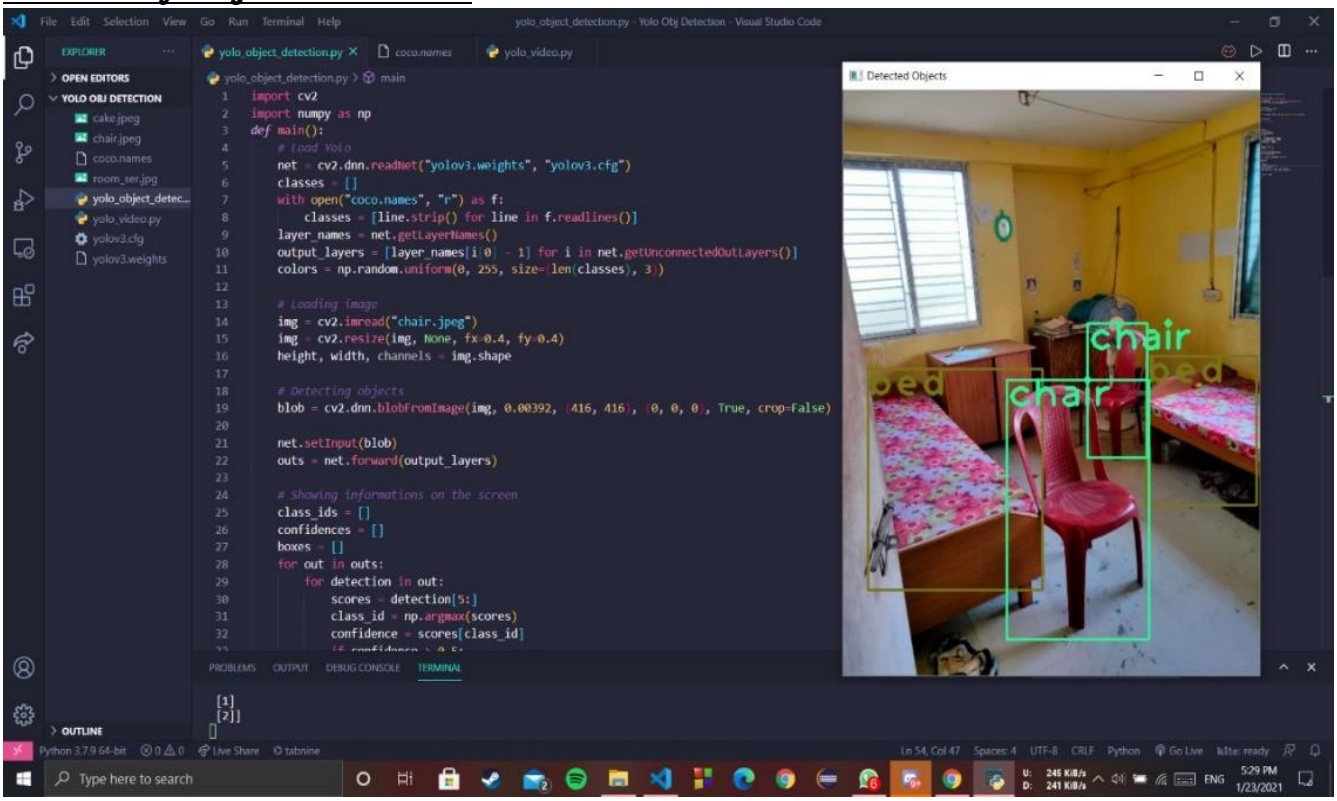
Sample Output: -

{'Readable': 2, 'RoadSign': 0, 'Id': 8, 'Posters': 0, 'NoImage': 0}

EM 4 Saitiahe Sector V Kolkata Kolkata-700091 PHONE NO: $636544417/10 IDENTITY CARQ ID; _ 181001011010 DISCIPLINE: 8. Tech(L}-CSE AMIT DUTTA NAME: BISWANATH DUTTA : _ KATWA KACHARI ROAD CODE _ 713130 DISTRICT: Burdwan nO 7470159715 OF BIRTH: 12-08-1998 2018-2019 0+

Object Detection:

Stationary Object Detection



Some important piece of code under Objection Detection: -

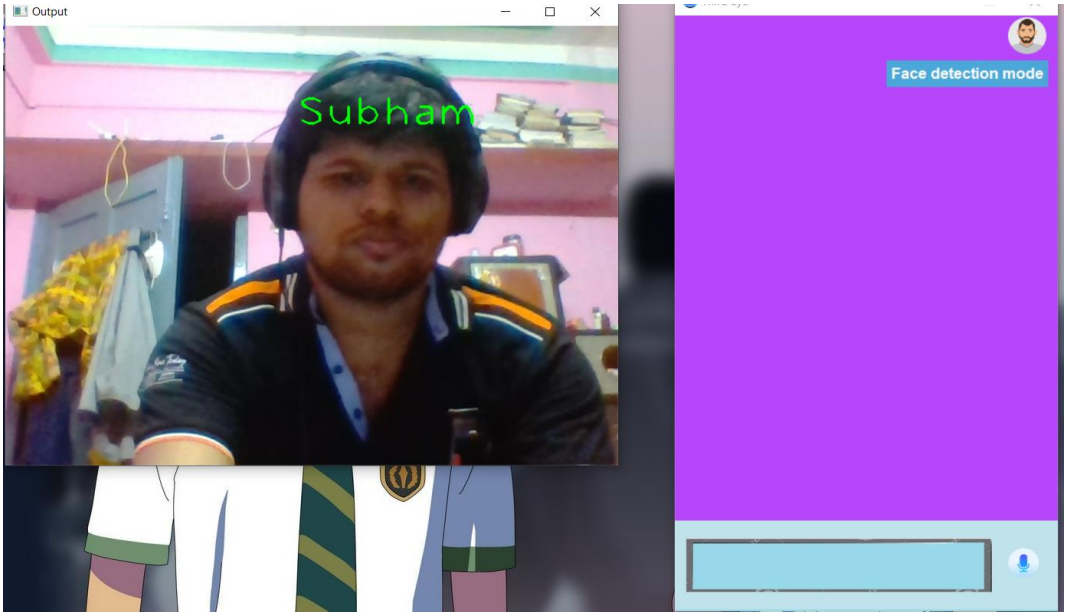
```python
import numpy as np

class ObjectDetection:
    def __init__(self):
        self.thres = 0.45 # Threshold to detect object
        self.nms_threshold = 0.2
        self.cap = cv2.VideoCapture(0)

    def obj_detection_v2(self):
        classNames= []
        classFile = 'Utils/ObjDetection_v2/coco.names'
        with open(classFile,'rt') as f:
            classNames = f.read().rstrip('\n').split('\n')

        configPath = 'Utils/ObjDetection_v2/ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt'
        weightsPath = 'Utils/ObjDetection_v2/frozen_inference_graph.pb'
        net = cv2.dnn_DetectionModel(weightsPath,configPath)
        net.setInputSize(320,320)
        net.setInputScale(1.0/ 127.5)
        net.setInputMean((127.5, 127.5, 127.5))
        net.setInputSwapRB(True)
        output = list()
        while True:
            success,img = self.cap.read()
            classIds, confs, bbox = net.detect(img,confThreshold=self.thres)
            if len(classIds) != 0:
                for classId, confidence,box in zip(classIds.flatten(),confs.flatten(),bbox):
                    recbox = False
                    if recbox == True:
                        cv2.rectangle(img,box,color=(0,255,0),thickness=2)
                    cv2.putText(img,classNames[classId-1].upper(),(box[0]+10,box[1]+30),
                                cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),2)
                    output.append(classNames[classId-1].upper())
                    cv2.putText(img,str(round(confidence*100,2)),(box[0]+200,box[1]+30),
                                cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),2)

            cv2.imshow("Output",img)
            if cv2.waitKey(1) & 0xFF == ord("q"):
                output = set(output)
                return output
```

Face Detection & Recognition:

We have added a Face detection and recognition functionality for the user to help him/her know about the persons who so ever will come near them. This will help them get an assurance if the person they are interacting with is the same person they think they are talking to. Some of the Modules used here are **mediapipe**, **tensorflow**, **keras**, **OpenCV**, **numpy**.

The **MediaPipe** Python framework grants direct access to the core components of the MediaPipe C++ framework such as Timestamp, Packet, and CalculatorGraph, whereas the ready-to-use Python solutions hide the technical details of the framework and simply return the readable model inference results back to the callers. MediaPipe framework sits on top of the pybind11 library. The C++ core framework is exposed in Python via a C++/Python language binding. The content below assumes that the reader already has a basic understanding of the MediaPipe C++ framework.

Keras is a powerful and easy-to-use free open source Python library for developing and evaluating deep learning models. It wraps the efficient numerical computation libraries Theano and TensorFlow and allows you to define and train neural network models in just a few lines of code.



The Python Code for Face Detection & recognition: -

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3' # hiding logs
import tensorflow as tf
import cv2
import mediapipe as mp
import time
import numpy as np
```

```python
def face_model():
    res = []
    cap = cv2.VideoCapture(0)
    pTime = 0
    model = tf.keras.models.load_model('Utils/Face_detection/face_final.h5')

    mp_face_detection = mp.solutions.face_detection
    mp_draw = mp.solutions.drawing_utils
    faceDetection = mp_face_detection.FaceDetection()

    while True:
        success, img = cap.read()

        imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        results = faceDetection.process(imgRGB)

        if results.detections:
            for id, detection in enumerate(results.detections):
                # mp_draw.draw_detection(img, detection)
                bboxC = detection.location_data.relative_bounding_box
                ih, iw, ic = img.shape
                #print(img.shape)
                final_img = cv2.resize(img,(224,224))
                final_img = np.expand_dims(final_img,axis=0) # need 4th dimension
                final_img = final_img/255 # normalizing

                prediction = model.predict(final_img)
                pred_gen = model.predict_generator(final_img)
                pred = np.argmax(prediction[0])
                print(np.round(np.amax(pred_gen[0])*100,2))
                mylist = ["Adi","Subham"]
                bbox = int(bboxC.xmin * iw), int(bboxC.ymin * ih), int(bboxC.width * iw), int(bboxC.height * ih)
                #cv2.rectangle(img, bbox, (255, 0, 255), 2)
                cv2.putText(img, f"{mylist[pred]}", (bbox[0],bbox[1]-25 ), cv2.FONT_HERSHEY_PLAIN, 3, (0, 255, 0), 2)
                res.append(mylist[pred])



        new_list = list(set(res))

        cv2.imshow("Output", img)
        if cv2.waitKey(1) & 0xFF == ord("q"):
            return new_list
```

Voice Based Assistant:

Email Sender:

```python
def email(rec_email=None, text="Hello, It's Third Eye here...", sub='Third Eye'):
        if '@gmail.com' not in rec_email: return
        s = smtplib.SMTP('smtp.gmail.com', 587)
        s.starttls()
        s.login("senderEmail", "senderPassword") # eg, abc@gmail.com (email) and ****(pass)
        message = 'Subject: {}\n\n{}'.format(sub, text)
        s.sendmail("senderEmail", rec_email, message)
        print("Sent")
        s.quit()
```

We have added a functionality to our assistant to send email to any user just by instructing the assistant verbally. We have used a library named “smtplib” for this. The “smtplib” module defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP (Simple Mail Transfer Protocol). An SMTP instance encapsulates an SMTP connection. It has methods that support a full repertoire of SMTP and ESMTP operations.

Math Calculation:

We have added a functionality to our assistant to do basic math operations to help the visually impaired user in day to day life. We have used a library named “math” for this. This module provides access to the mathematical functions defined by the C standard. Though, these functions cannot be used with complex numbers.

```
import math
def basicOperations(text):
        if 'root' in text:
                temp = text.rfind(' ')
                num = int(text[temp+1:])
                return round(math.sqrt(num),2)

        text = text.replace('one','1 ') # Added Special case
        text = text.replace('plus', '+')
        text = text.replace('minus', '-')
        text = text.replace('x', '*')
        text = text.replace('multiplied by', '*')
        text = text.replace('multiply', '*')
        text = text.replace('divided by', '/')
        text = text.replace('to the power', '**')
        text = text.replace('power', '**')
        result = eval(text)
        return round(result,2)

```

Translation:

We have added a functionality to our assistant to help the user by translating any sentence to any language he wants. We have used a library named “difflib” for this. This module provides classes and functions for comparing sequences. difflib.get_close_matches(word, possibilities, n, cutoff) accepts four parameters in which n, cutoff are optional. word is a sequence for which close matches are desired, possibilities is a list of sequences against which to match word.

```
def translate(query):
        try:
                query = query.replace('dictionary', '')
                if 'meaning' in query:
                        ind = query.index('meaning of')
                        word = query[ind+10:].strip().lower()
                elif 'definition' in query:
                        try:
                                ind = query.index('definition of')
                                word = query[ind+13:].strip().lower()
                        except:
                                ind = query.index('definition')
                                word = query[ind+10:].strip().lower()
                else: word = query

                word, result, check = getMeaning(word)
                result = choice(result)

                if check==1:
                        return ["Here's the definition of \"" +word.capitalize()+ '"', result]
                elif check==0:
                        return ["I think you're looking for \"" +word.capitalize()+ '"', "It's definition is,\n" + result]
                else:
                        return [result, '']

        except:
                print("Don't know")
                return ["What?",""]
```

Smart Dictionary Search:

We have added a functionality to our assistant to help the user with meaning of any word that user wants to know in any language. We have

used a library named “difflib” for this. This module provides classes and functions for comparing sequences. difflib.get_close_matches(word, possibilities, n, cutoff) accepts four parameters in which n, cutoff are optional. word is a sequence for which close matches are desired, possibilities is a list of sequences against which to match word.

```
from difflib import get_close_matches
import json
from random import choice

data = json.load(open('extrafiles/dict_data.json', encoding='utf-8'))

def getMeaning(word):
        if word in data:
                return word, data[word], 1
        elif len(get_close_matches(word, data.keys())) > 0:
                word = get_close_matches(word, data.keys())[0]
                return word, data[word], 0
        else:
                return word, ["This word doesn't exists in the dictionary."], -1

```

Play Video from YouTube:

We have added a functionality to our assistant to help the user if he wants play any video from YouTube. We have used a library named “youtube_search” for this. This is a python function for searching for Youtube videos to avoid using their heavily rate-limited API. To avoid using the API, this uses the form on the youtube homepage and scrapes the resulting page.

```

def youtube(query):

        query = query.replace('play',' ')
        query = query.replace('on youtube',' ')
        query = query.replace('youtube',' ')

        results = YoutubeSearch(query,max_results=1).to_dict()

        webbrowser.open('https://www.youtube.com/watch?v=' + results[0]['id'])
        return "Enjoy Sir..."
```

Web-Scrapping: -

Packages Used:

wikipedia, web browser, **requests from bs4** ,**BeautifulSoup**, **urllib.request** ,**os from geopy.geocoders** ,**Nominatim** from **geopy.distance** ,**great_circle**

Search from Wikipedia

Initiate by installing the library first then fetch it to import in the system, later called in to scrape information likewise while needed.

```
def wikiResult(query):
    query = query.replace('wikipedia','')
    query = query.replace('search','')
    if len(query.split())==0: query = "wikipedia"
    try:
        return wikipedia.summary(query, sentences=2)
    except Exception as e:
        return "Desired Result Not Found"
```

Latest News:

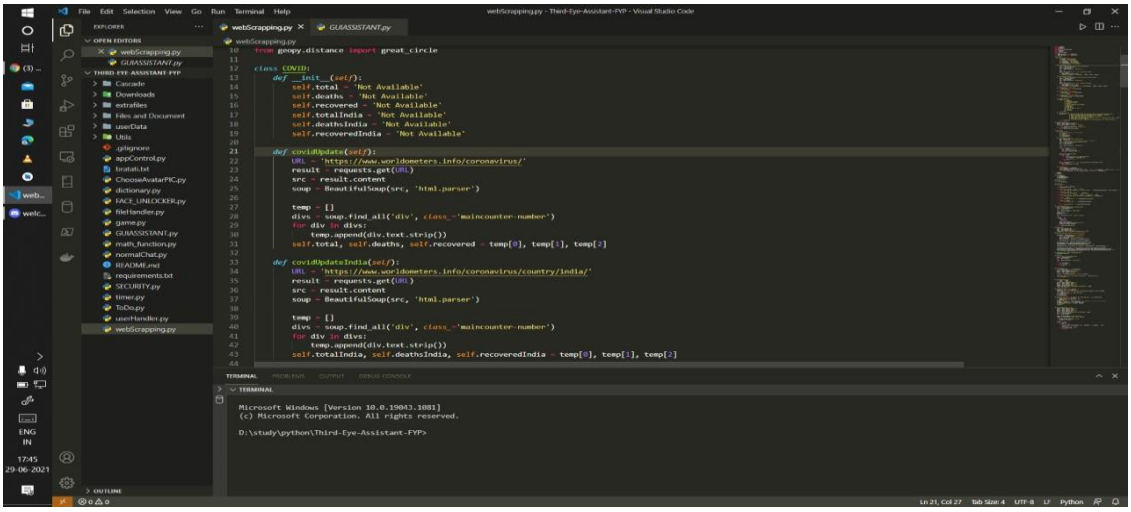
```python
def latestNews(news=5):
    URL = 'https://indianexpress.com/latest-news/'
    result = requests.get(URL)
    src = result.content

    soup = BeautifulSoup(src, 'html.parser')

    headlineLinks = []
    headlines = []

    divs = soup.find_all('div', {'class':'title'})

    count=0
    for div in divs:
        count += 1
        if count>news:
            break
        a_tag = div.find('a')
        headlineLinks.append(a_tag.attrs['href'])
        headlines.append(a_tag.text)

    return headlines,headlineLinks
```

Initiation of data fetching from URL: https://indianexpress.com/latest-news/ (i.e.;) parsed it using BeautifulSoup, which were all searched for the title numberclass in the parsed content loop it in

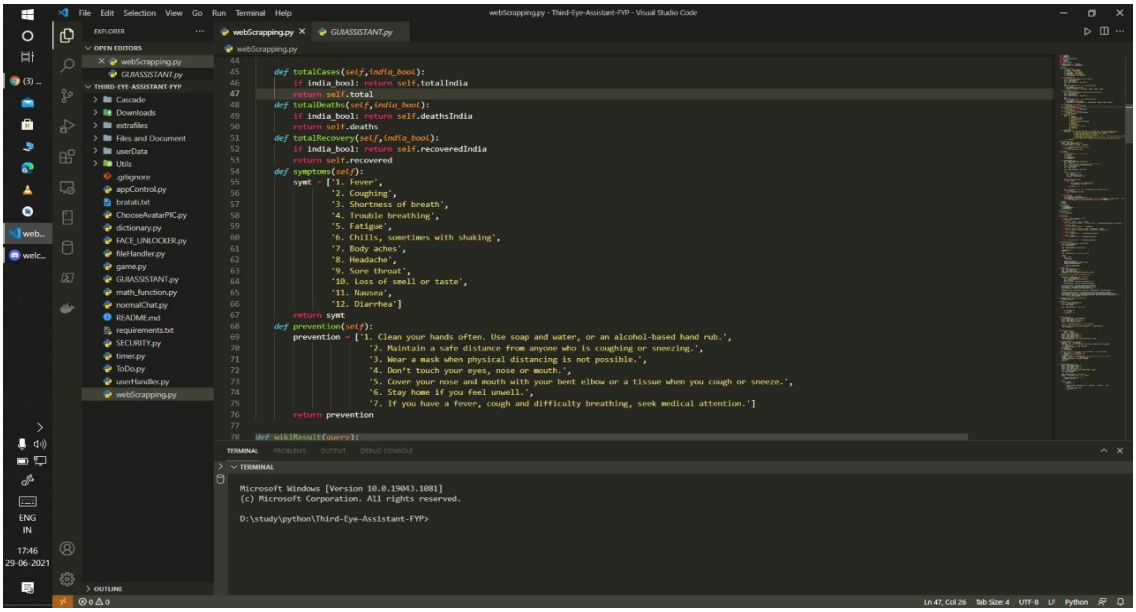
```python
def youtube(query):
    from youtube_search import YoutubeSearch
    query = query.replace('play',' ')
    query = query.replace('on youtube',' ')
    query = query.replace('youtube',' ')

    results = YoutubeSearch(query,max_results=1).to_dict()

    webbrowser.open('https://www.youtube.com/watch?v=' + results[0]['id'])
    return "Enjoy Sir..."


def googleSearch(query):
    if 'image' in query:
        query += "&tbm=isch"
    query = query.replace('images','')
    query = query.replace('image','')
    query = query.replace('search','')
    query = query.replace('show','')
    webbrowser.open("https://www.google.com/search?q=" + query)
    return "Here you go..."
```

to get headlines and headlinelinks then get the tag.attrs with href meanwhile if we want to search about it we can by accessing the package itself.

Covid Tracker:



Importing BeautifulSoup and urllib.request from bs4 in use to fetch data first, then parse it through BeautifulSoup by using find all searched for main counter numberclass in the parsed content then divided all the div which we get and store it all the info of totalDeath, totalRecovery, totalCases same as it is done for any specific country and more help prevention and symptoms have added.



Direction using Map:

```
    return headlines,headlineLinks

def maps(text):
    text = text.replace('maps', '')
    text = text.replace('map', '')
    text = text.replace('google', '')
    openWebsite('https://www.google.com/maps/place/'+text)

def giveDirections(startingPoint, destinationPoint):

    geolocator = Nominatim(user_agent='assistant')
    if 'current' in startingPoint:
        res = requests.get("https://ipinfo.io/")
        data = res.json()
        startinglocation = geolocator.reverse(data['loc'])
    else:
        startinglocation = geolocator.geocode(startingPoint)

    destinationlocation = geolocator.geocode(destinationPoint)
    startingPoint = startinglocation.address.replace(' ', '+')
    destinationPoint = destinationlocation.address.replace(' ', '+')

    openWebsite('https://www.google.co.in/maps/dir/'+startingPoint+'/'+destinationPoint+'/')

    startinglocationCoordinate = (startinglocation.latitude, startinglocation.longitude)
    destinationlocationCoordinate = (destinationlocation.latitude, destinationlocation.longitude)
    total_distance = great_circle(startinglocationCoordinate, destinationlocationCoordinate).km #.mile
    return str(round(total_distance, 2)) + 'KM'
```

Importing Nominatim from geopy.geocoders and great_circle from geopy.distance which gives the starting point and destination point then use of package Nominatim to get a request from "https://ipinfo.io/" then the use of starting location = geolocator.reverse(data['loc'])

else : starting location = geolocator.geocode(starting point)

destination location = geolocator.geocode(destination Point)

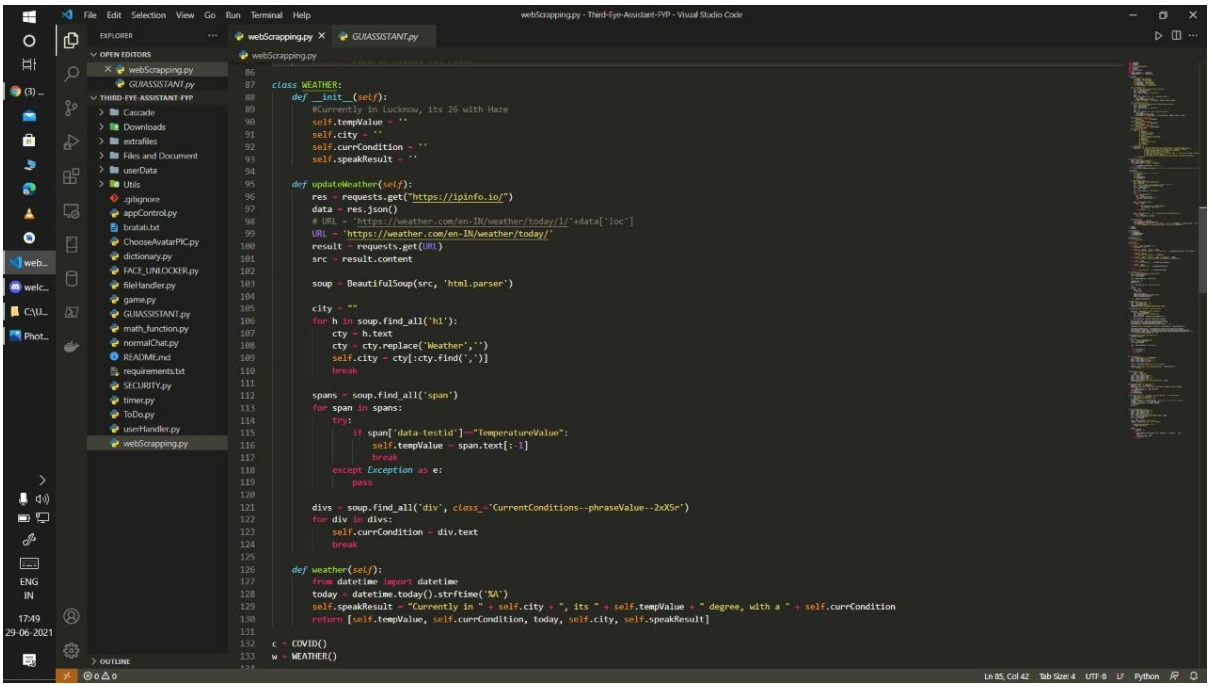
In the starting point replace all the symbols like "+" from starting location and destination location to get starting location Coordinate.

Weather:

After scrapping the data from https://ipinfo.io/ parse it into JSON file to fetch the html form;

```
def dataUpdate():
    c.covidUpdate()
    c.covidUpdateIndia()
    w.updateWeather()

##### WEATHER #####
def weather():
    return w.weather()
```

the URL again from https://weather.com/en-IN/weather/today/ then parsing it by using BeautifulSoup. By using find all searched for h1 and process the text to find span class to store it into spans and to get TemperatureValue in order to process



```
class WEATHER:
    def __init__(self):
        #Currently in Lucknow, its 26 with Haze
        self.tempValue = ''
        self.city = ''
        self.currCondition = ''
        self.speakResult = ''

    def updateWeather(self):
        res = requests.get("https://ipinfo.io/")
        data = res.json()
        # URL = 'https://weather.com/en-IN/weather/today/1/'+data['loc']
        URL = 'https://weather.com/en-IN/weather/today/'
        result = requests.get(URL)
        src = result.content

        soup = BeautifulSoup(src, 'html.parser')

        city = ""
        for h in soup.find_all('h1'):
            cty = h.text
            cty = cty.replace('Weather','')
            self.city = cty[:cty.find(',')]
            break

        spans = soup.find_all('span')
        for span in spans:
            try:
                if span['data-testid']=="TemperatureValue":
                    self.tempValue = span.text[:-1]
                    break
            except Exception as e:
                pass

        divs = soup.find_all('div', class_='CurrentConditions--phraseValue--2xXSr')
        for div in divs:
            self.currCondition = div.text
            break

    def weather(self):
        from datetime import datetime
        today = datetime.today().strftime('%A')
        self.speakResult = "Currently in " + self.city + ", its " + self.tempValue + " degree, with a " + self.currCondition
        return [self.tempValue, self.currCondition, today, self.city, self.speakResult]

c = COVID()
w = WEATHER()
```

the text then again finds all div with CurrentConditions--phraseValue--2xXSr after receiving text from it, initialization of process the text where print all the data which we have collected from these processes.

Timer:

We have added a timer functionality to our assistant which will let the user use a stopwatch/timer to keep any time record that they might need later. The **timer** module provides functions that help manipulate time and use them in your program. There are other "cousin" modules that work more precisely with date and the calendar within your system, but this module is more focused on just manipulating time - working with hours going all the way down to milliseconds. The **sleep** function is available within the time module that contains many other functions that deal with time access and

conversions. And although this module is available on all platforms, not all functions within it might be available or even work the same exact way.

```
from time import sleep
import re
import playsound
from tkinter import *
from threading import Thread

def startTimer(query):
        nums = re.findall(r'[0-9]+', query)
        time = 0
        if "minute" in query and "second" in query:
                time = int(nums[0])*60 + int(nums[1])
        elif "minute" in query:
                time = int(nums[0])*60
        elif "second" in query:
                time = int(nums[0])
        else: return

        print("Timer Started")
        sleep(time)
        Thread(target=timer).start()
        playsound.playsound("extrafiles/audios/Timer.mp3")

```

Operating System & Battery Information:

This provides the basic system information, i.e. about the operating system status and battery status. We have used module named **wmi** and it stands for Windows Management Instrumentation, which is Microsoft's implementation of Web-Based Enterprise Management (WBEM), an industry initiative to provide a Common Information Model (CIM) for pretty much any information about a computer system.

The Python WMI module is a lightweight wrapper on top of the pywin32 extensions, and hides some of the messy plumbing needed to get Python to talk to the WMI API.

```

def systemInfo():
        import wmi
        c = wmi.WMI()
        my_system_1 = c.Win32_LogicalDisk()[0]
        my_system_2 = c.Win32_ComputerSystem()[0]
        info = ["Total Disk Space: " + str(round(int(my_system_1.Size)/(1024**3),2)) + " GB",
                        "Free Disk Space: " + str(round(int(my_system_1.Freespace)/(1024**3),2)) + " GB",
                        "Manufacturer: " + my_system_2.Manufacturer,
                        "Model: " + my_system_2. Model,
                        "Owner: " + my_system_2.PrimaryOwnerName,
                        "Number of Processors: " + str(my_system_2.NumberOfProcessors),
                        "System Type: " + my_system_2.SystemType]
        return info

def batteryInfo():
        # usage = str(psutil.cpu_percent(interval=0.1))
        battery = psutil.sensors_battery()
        pr = str(battery.percent)
        if battery.power_plugged:
                return "Your System is currently on Charging Mode and it's " + pr + "% done."
        return "Your System is currently on " + pr + "% battery life."

```

Tab & File Operations:

from pynput.keyboard import Key, Controller

```
class TabOpt:
        def __init__(self):
                self.keyboard = Controller()

        def switchTab(self):
                self.keyboard.press(Key.ctrl)
                self.keyboard.press(Key.tab)
                self.keyboard.release(Key.tab)
                self.keyboard.release(Key.ctrl)

        def closeTab(self):
                self.keyboard.press(Key.ctrl)
                self.keyboard.press('w')
                self.keyboard.release('w')
                self.keyboard.release(Key.ctrl)

        def newTab(self):
                self.keyboard.press(Key.ctrl)
                self.keyboard.press('n')
                self.keyboard.release('n')
                self.keyboard.release(Key.ctrl)
```

The tab and file operations are added for the user to perform/operate any tabular or file operations like minimizing a particular program/file/folder or create a new file/document etc. We have used a library named "**pynput**" for this. This library allows you to control and monitor input devices. Currently, mouse and keyboard input and monitoring are supported.

File Operations:

```
def createFile(text):
        # change the applocation as per your system path
        appLocation = "C:\\Program Files\\Sublime Text 3\\sublime_text.exe"

        if isContain(text, ["ppt","power point","powerpoint"]):
                file_name = "sample_file.ppt"
                appLocation = "C:\\Program Files (x86)\\Microsoft Office\\root\\Office16\\POWERPNT.exe"

        elif isContain(text, ['excel','spreadsheet']):
                file_name = "sample_file.xsl"
                appLocation = "C:\\Program Files (x86)\\Microsoft Office\\root\\Office16\\EXCEL.EXE"

        elif isContain(text, ['word','document']):
                file_name = "sample_file.docx"
                appLocation = "C:\\Program Files (x86)\\Microsoft Office\\root\\Office16\\WINWORD.EXE"

        elif isContain(text, ["text","simple","normal"]): file_name = "sample_file.txt"
        elif "python" in text: file_name = "sample_file.py"
        elif "css" in text:     file_name = "sample_file.css"
        elif "javascript" in text: file_name = "sample_file.js"
        elif "html" in text: file_name = "sample_file.html"
        elif "c plus plus" in text or "c + +" in text: file_name = "sample_file.cpp"
        elif "java" in text: file_name = "sample_file.java"
        elif "json" in text: file_name = "sample_file.json"
        else: return "Unable to create this type of file"

        file = open(path + file_name, 'w')
        file.close()
        subprocess.Popen([appLocation, path + file_name])
        return "File is created.\nNow you can edit this file"
```

Volume Control:

We have added a functionality to our assistant to help the user to change the volume of the assistant if he wants. We have used a library named "**pynput**" for this. This library allows you to control and monitor input devices. Currently, mouse and keyboard input and monitoring are supported.

```
from pynput.keyboard import Key, Controller
```

```
keyboard = Controller()
def mute():
        for i in range(50):
                keyboard.press(Key.media_volume_down)
                keyboard.release(Key.media_volume_down)

def full():
        for i in range(50):
                keyboard.press(Key.media_volume_up)
                keyboard.release(Key.media_volume_up)


def volumeControl(text):
        if 'full' in text or 'max' in text: full()
        elif 'mute' in text or 'min' in text: mute()
        elif 'incre' in text:
                for i in range(5):
                        keyboard.press(Key.media_volume_up)
                        keyboard.release(Key.media_volume_up)
        elif 'decre' in text:
                for i in range(5):
                        keyboard.press(Key.media_volume_down)
                        keyboard.release(Key.media_volume_down)
```

To Do List:

We have added a functionality to our assistant to help the user by adding daily tasks in his to-do list. We have used libraries named "os" and "datetime" for this. "os" module provides a portable way of using operating system dependent functionality. The "datetime" module supplies classes for manipulating dates and times.

```
import os
file = "userData/toDoList.txt"

def createList():
        f = open(file,"w")
        present = datetime.now()
        dt_format = present.strftime("Date: " + "%d/%m/%Y"+ " Time: " + "%H:%M:%S" + "\n")
        f.write(dt_format)
        f.close()

def toDoList(text):
        if os.path.isfile(file) == False:
                createList()

        f = open(file,"r")
        x = f.read(8)
        f.close()
        y = x[6:]
        yesterday = int(y)
        present = datetime.now()
        today = int(present.strftime("%d"))
        if (today-yesterday) >= 1:
                createList()
        f = open(file,"a")
        dt_format = present.strftime("%H:%M")
        print(dt_format)
        f.write(f"[{dt_format}] : {text}\n")
        f.close()
```

Conclusion: ~

In conclusion, the “Third Eye Assistant” is a technology which can be very useful for the visually impaired persons and this can eventually be a better replacement to a guide dog or a white cane for the visually impaired persons. We have included some basic functionalities which a human might need to do in their day to day life. Apart from that, there are many other functionalities that we can include, but after having proper knowledge about them. We know that there are many scopes of improvements and we would be looking forward to improve those particular segments in order to make it as helpful as possible.