PYTHON INNOVATIVE PROJECT

AIM: DIGITALIZING THE IMAGE FORM INTO HTML FORM BY CONVERTING THE IMAGE INTO TEXT FORMAT AND CONVERTING IT TO HTML FORM USING PHP AND LASTLY CONVERTING THE TEXT IN AUDIO FORMAT.

ROLLNO:

1. 18BCE141(TULSI PALAN)
2. 18BCE175(YASH PATEL)
3. 18BCE165 (OM PATEL)

UNDER THE GUIDANCE OF:

-PROF.AJAY PATEL

Step included :

1. SELECT THE IMAGE USING GUI
2. SCAN THE IMAGE
3. SAVE THE SCANNED OUTPUT
4. RECOGNIZE TEXT
5. SAVE THE TEXT IN FILE
6. CONVERT IT INTO HTML DOC USING PHP
7. CONVERTING TEXT TO AUDIO

NOTE: The python file is run in command prompt since we are using GUI to select the image it can not be run in any platform such as anaconda or pycharm.

LIBRARIES WE USED:

- SCIKIT-IMAGE
- NUMPY
- OPENCV
- IMUTILS
- TKINTER
- TESSERACT
- GTTS

1.SELECTING THE IMAGE

Here we used the GUI library called TKINTER to select the image when we run the code the dialog box will be open to choose the image from system

CODE:

```
#initiate tkinker and hide window
main_win = tkinter.Tk()
main_win.withdraw()

main_win.overrideredirect(True)
main_win.geometry('0x0+0+0')

main_win.deiconify()
main_win.lift()
main_win.focus_force()

#open file selector
main_win.sourceFile = filedialog.askopenfilename(filetypes = (("Image
Files",("*.jpg","*.png","*.jpeg")),("All Files","*")),parent=main_win, initialdir=
"/",
title='Please select a image file')

#close window after selection
main_win.destroy()

img_path = main_win.sourceFile
```

2.SCANNING THE IMAGE

It includes of three step:

1.Edge Detection

-The edge is detected in the image

2.Finding Contours

- he contours of the image is find and the binding box is drawn.

3.Perspective Transform

-The scanned image is obtained.

CODE:

```
print("\n----- Image Scanning Started -----\n")

# show the original image and the edge detected image
print("STEP 1: Edge Detection")
cv2.imshow("Image", image)
cv2.imshow("Edged", edged)
#cv2.imwrite('./scanned_output/edged.jpg',edged)
cv2.waitKey(0)
cv2.destroyAllWindows()

# find the contours in the edged image, keeping only the
# largest ones, and initialize the screen contour
cnts = cv2.findContours(edged.copy(), cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
cnts = imutils.grab_contours(cnts)
cnts = sorted(cnts, key = cv2.contourArea, reverse = True)[:5]

# loop over the contours
for c in cnts:
    # approximate the contour
    peri = cv2.arcLength(c, True)
    approx = cv2.approxPolyDP(c, 0.02 * peri, True)

    # if our approximated contour has four points, then we
    # can assume that we have found our screen
    if len(approx) == 4:
        screenCnt = approx
        break

# show the contour (outline) of the piece of paper
print("STEP 2: Find contours of paper")
cv2.drawContours(image, [screenCnt], -1, (0, 255, 0), 2)
cv2.imshow("Outline", image)
#cv2.imwrite('./scanned_output/outline.jpg',image)
cv2.waitKey(0)
cv2.destroyAllWindows()

# apply the four point transform to obtain a top-down
# view of the original image
warped = four_point_transform(orig, screenCnt.reshape(4,2) * ratio)

# convert the warped image to grayscale, then threshold it
# to give it that 'black and white' paper effect
warped = cv2.cvtColor(warped, cv2.COLOR_BGR2GRAY)
T = threshold_local(warped, 11, offset = 10, method = "gaussian")
warped = (warped > T).astype("uint8") * 255


# show the original and scanned images
```

```
print("STEP 3: Apply perspective transform")
cv2.imshow("Original", imutils.resize(orig, height = 650))
cv2.imshow("Scanned", imutils.resize(warped, height = 650))
cv2.waitKey(0)
cv2.destroyAllWindows()
```

3.SAVING THE SCANNED IMAGE

The image is stored is in system using OpenCV Library

-Using Function cv2.imwrite()

CODE:

```
#Write the Scanned Output into an image file
print("\n------ Saving the Scanned Image -------")
cv2.imwrite('./scanned_output/output.jpg',warped)


x = input("\n----- Do You Want to Recognize Text from the Scanned Image??  [y/n] -
----- ")

if(x=='Y' or x=='y'):
    print("\n----- Start recognize text from image -----")
    text = get_string(img_path)
    print(text)
```

4.REGONIZE TEXT FROM IMAGE

The module(library) Opencv and Pytesseract is used to regonize the text from image using following the library function:

- To read the image -using opencv

  -cv2.imread()

- Converting it no greyscale

  -cv2.cvtColor()

- Removing noise from image

  -cv2.dilate()

  -cv2.erode()

- Applying threshold to get image in black and white

  -cv2.adaptiveThreshold()

- Recognizing the text from black and white image using tesseract library

  -pytesseract.image_to_string(img)

```
# Path of working folder on Disk
src_path = "./"
```

```
def get_string(img_path):
    # Read image with opencv
    img = cv2.imread(img_path)

    # Convert to gray
    img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

    # Apply dilation and erosion to remove some noise
    kernel = np.ones((1, 1), np.uint8)
    img = cv2.dilate(img, kernel, iterations=1)
    img = cv2.erode(img, kernel, iterations=1)

    # Apply threshold to get image with only black and white
    img = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
cv2.THRESH_BINARY, 31, 2)

    pytesseract.pytesseract.tesseract_cmd = r"C:\Program Files (x86)\Tesseract-
OCR\tesseract.exe"
    # Recognize text with tesseract for python
    result = pytesseract.image_to_string(img)

    return(result)
```

5. SAVE THE TEXT IN FILE

- Saving the text in file and printing it using file handling

-Here in code we used encoding=”utf-8” as all character may not be in Unicode ,so masking it to unicode character.

- The file will be opened using
  -os.startfile()

CODE:

```
with open("OCR Text.txt", "w+", encoding="utf-8") as f:
    f.write(text)
    f.close()

#change the path of the OCR Text.txt file to open it.
os.startfile("OCR Text.txt")
print("------ Done -------")
```

Thus, using this libraries we scanned and recognize the text from image and save it in file.

6. CONVERTING IT INTO HTML FORM.

We used php to convert the string of file in to html , using various php function we converted text File into html form.

```
<html>
<head>
<link src = "style.css">
```

```
</head>
<body>
<?php
$myfile = fopen("OCR Text.txt", "r") or die("Unable to open file!");
$a = array();
   for ($i=0;;$i++)
{
   $temp=fgets($myfile);
   if ($temp == FALSE){
     break;
  }
  else {
  array_push($a,$temp);
  }
}? >
< center >
  < font
size = 40
px > <?php
echo $a[0]? > < / font > < / center >
                              < div


class ="main" style="padding-left:100px" >

< ?php
for ($j=1;$j < $i;$j++){
    echo "$a[$j]";
echo '<input type="text">';
echo "<br>";
}
fclose($myfile);
? >
< / div >
< / body >
< / html >
```

7.CONVERTING THE TEXT INTO AUDIO

The library gTTS is used to convert the text into audio.

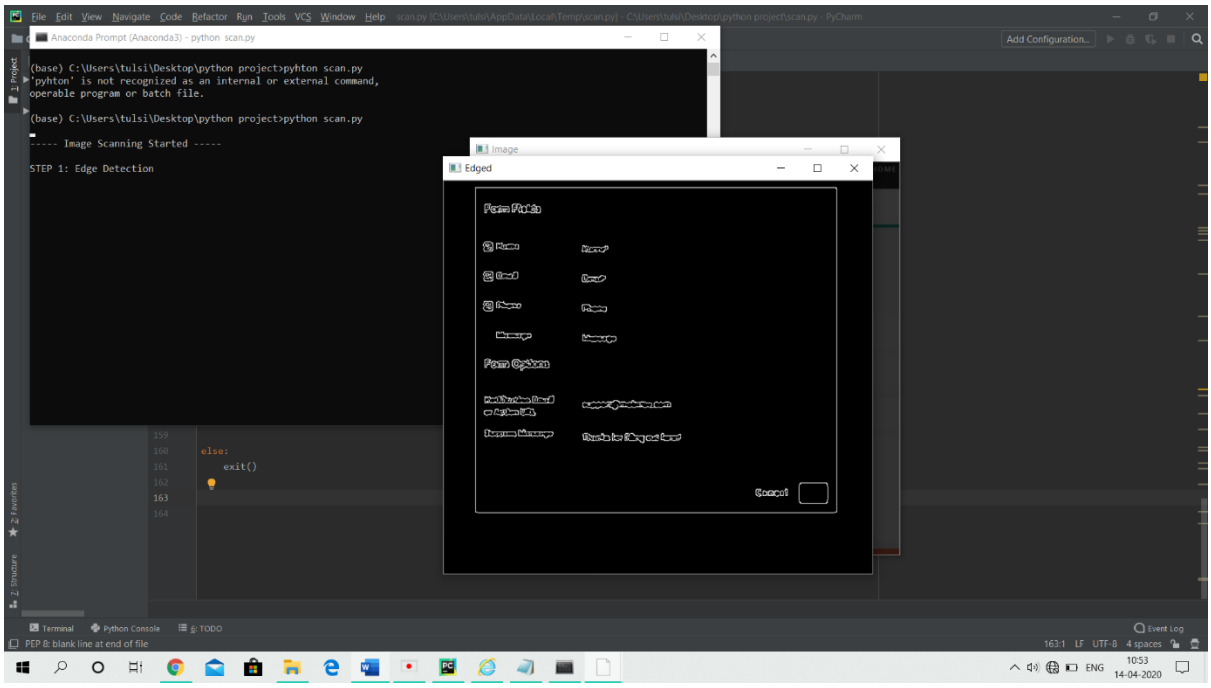
CODE:

```
fh = open("OCR Text.txt", "r")
myText = fh.read().replace("\n", " ")

# Language we want to use
language = 'en'

output = gTTS(text=myText, lang=language, slow=False)

output.save("output.mp3")
fh.close()

# Play the converted file
os.system("start output.mp3")
```

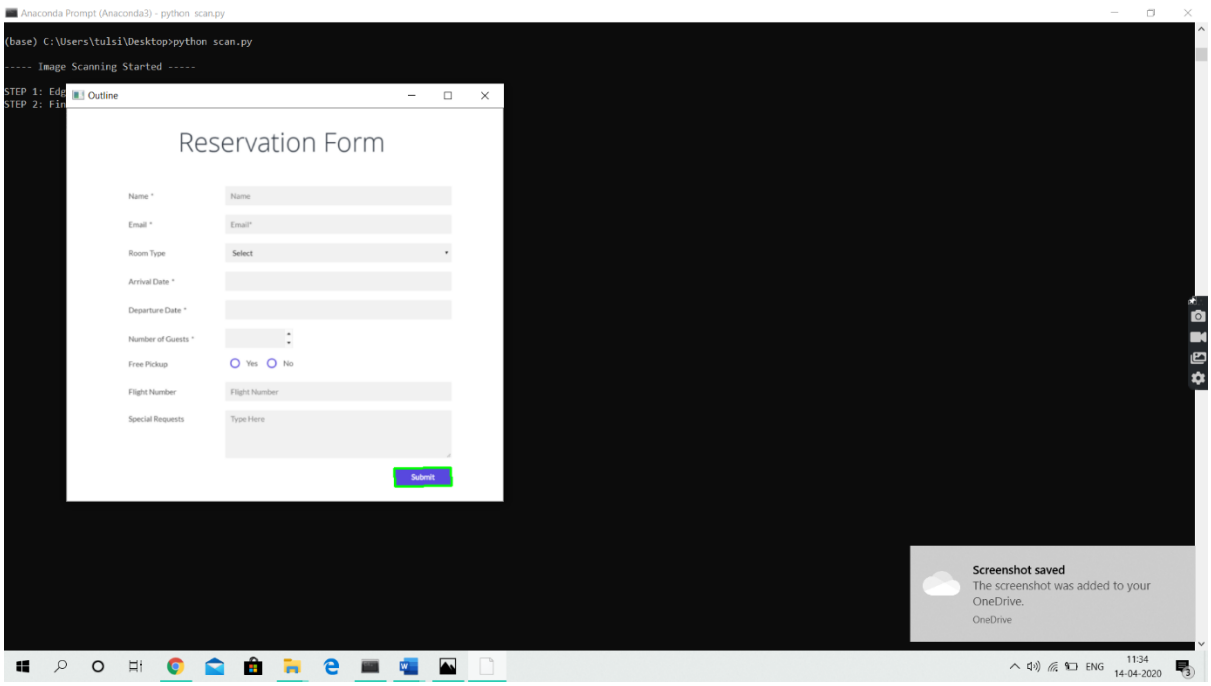
AFTER RUNNING THE CODE:

OUTPUT:

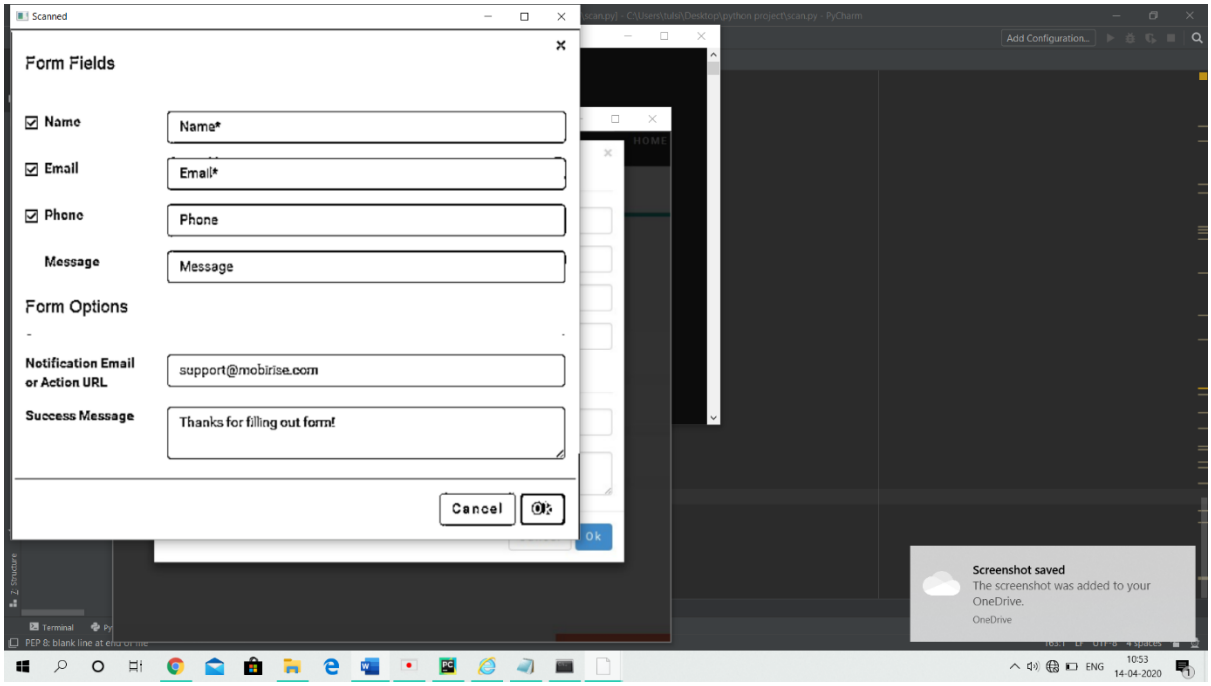
Step-1: Edge Detection



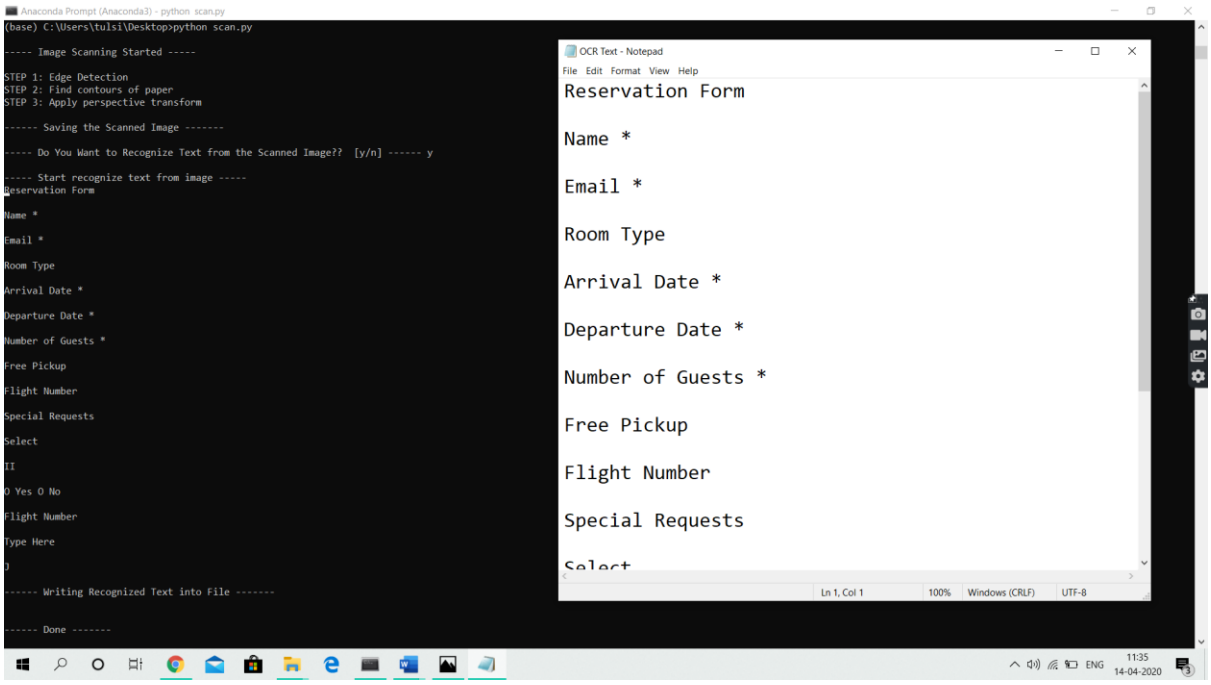
Step-2:Binding the boundaries and finding contours



Step-3: scanned image



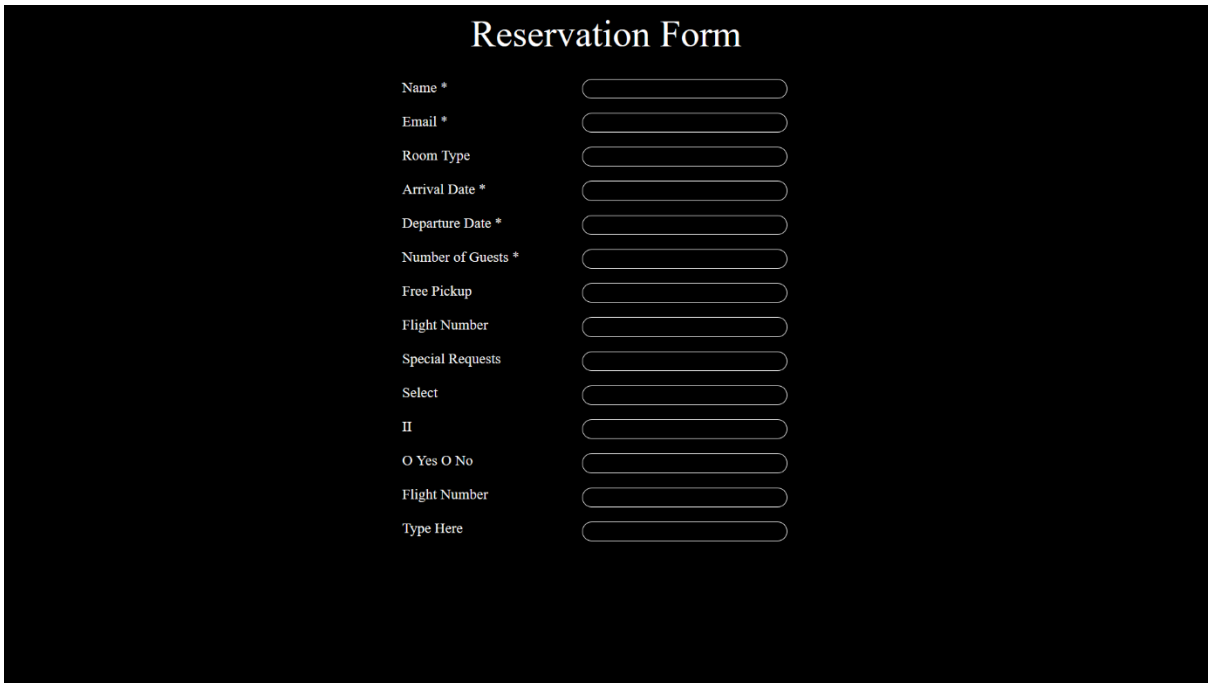
Step-4: printing the text



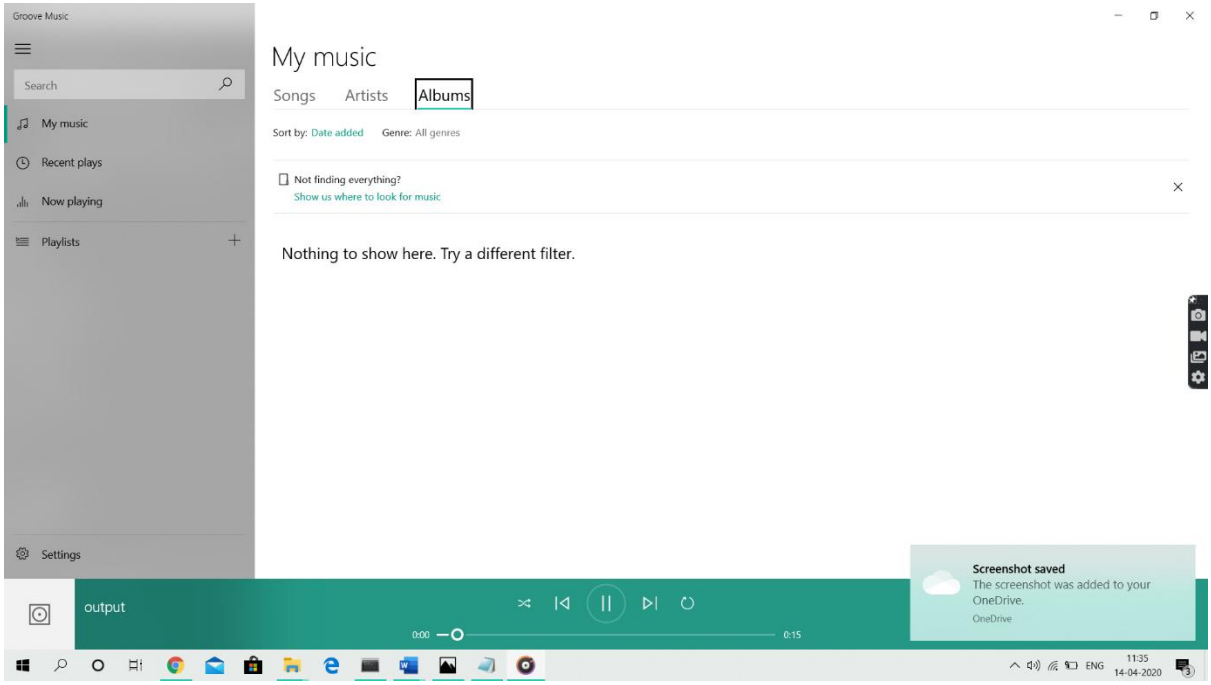
Step-5: the OCR text.txt contain the scanned text.

OCR Text.txt

Step-6: Converting it into HTML form



Step-7:Converting the text file into audio format.



output.mp3