INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, RANCHI

Sketch Animation

CS PROJECT

NAME : ARYA PRATAP SINGH

SUBJECT : PYTHON PROGRAMMING

SESSION : 2024-2025

INDEX

SERIAL NUMBER.	TOPIC	PAGE NUMBER
1	INTRODUCTION	3
2	PYTHON CONCEPTS USED	4
3	DATA FLOW DIAGRAM	12
4	SOURCE CODE	13
5	OUTPUT	24
6	LEARNING POINTS	26
7	LIMITATIONS	27
8	REFERENCES	28

INTRODUCTION

"If you're sitting in your minivan, playing your computer animated films for your children in the back seat, is it the animation that's entertaining you as you drive and listen? No, it's the storytelling. That's why we put so much importance on the story. No amount of great animation will save a bad story."

– John Lasseter

ANIMATIONS ARE USED TO EXPRESS DIFFERENT STORIES AND EMOTIONS WHICH INSPIRE US. THIS PROJECT USES A STACK OF FRAMES TO CREATE A BASIC SKETCH ANIMATION USING LIVE WEBCAM CAPTURES. IT ALSO DEPICTS THE HARD WORK REQUIRED TO CREATE AN ANIMATION CLIP OF EVEN A MINUTE. DEDICATED TO ALL THE AMAZING ANIMATORS OUT THERE.

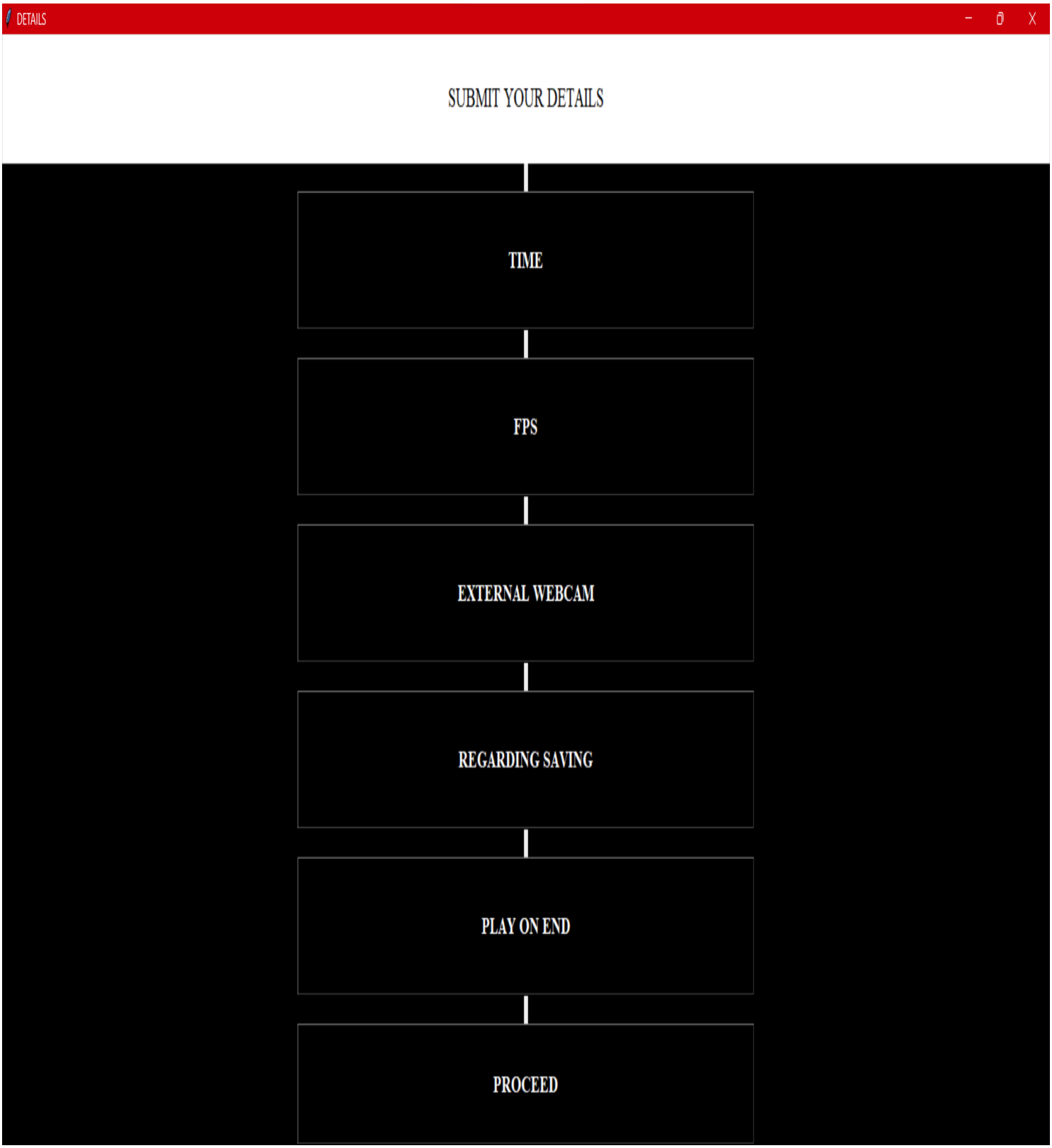
PYTHON MODULES USED :

THIS PROJECT CONSISTS OF INNUMERABLE MODULES WHICH HELPED IN INCREASING THE EFFICACY AND USER EXPERIENCE. VARIOUS CONDITIONAL STATEMENTS ARE USED TO MAINTAIN PROPER FLOW OF DATA AS WELL AS TO IMPROVE THE ERROR HANDLING CAPACITY OF THE PROJECT TO ITS MAXIMUM LEVEL POSSIBLE. ALTHOUGH NO PROJECT IS ERROR FREE , BUT ERROR CAN BE MINIMIZED TO A CERTAIN DEGREE

THE MAJOR MODULES WHICH HELPED IN THE DEVELOPMENT OF THIS PROGRAM ARE AS FOLLOW :

1. **TKINTER :**

   THIS LIBRARY IS USED TO CREATE INTERACTIVE GRAPHICAL USER INTERFACE (GUI). IT CONSISTS OF VARIED VARIETY OF OPTIONS TO CUSTOMIZE THE USER INTERFACE. IT IS BEING USED IN THIS PROJECT TO SERVE AS A LOGIN SYSTEM INTERFACE AND HANDLE USER LOGINS / REGISTERS / LOGIN VERIFICATION OR LOGIN ERROR. THE PROGRAM WON'T CONTINUE IF THE USER DOES NOT LOGIN


DETAILS
SUBMIT YOUR DETAILS
TIME
FPS
EXTERNAL WEBCAM
REGARDING SAVING
PLAY ON END
PROCEED

2. OS :

OS MODULE PROVIDES FUNCTIONALITY TO THE PROGRAM TO INTERACT WITH THE USER'S OPERATING SYSTEM AND DO SOME IMPORTANT BACKEND WORK.

```
import os
try:
    os.system('pip install -q -r requirements.txt')
    ms=0
except:
    ms=1
import cv2
import os.path
```

```
try:
    directory = f"RAW_IMAGES{rad}"
    path = os.path.join(parent_dir, directory)
    os.mkdir(path)
    directory = f"SKETCHED_IMAGES{rad}"
    parent_dir = location
    path = os.path.join(parent_dir, directory)
    os.mkdir(path)

except Exception as error1:
    pass

try:
    directory = f"RUNTIME_LOGS"
    parent_dir = location
    path = os.path.join(parent_dir, directory)
    os.mkdir(path)
except Exception as error2:
    pass
```

3. RANDOM :

THIS MODULE HELPS IN AUTO-SELECTING RANDOM INTEGERS FROM A GIVEN RANGE OF INTEGERS. IT IS BEING USED HERE TO SET THE POSITION OF ENEMIES AFTER GETTING HIT BY THE BULLET AND ON START ALSO.

```
class main_program(multiprocessing.Process):
    def run(self):
        global rad
        start_time = ti.time()
        Id = subprocess.check_output(['systeminfo']).decode('utf-8').split('\n')
        location = os.getcwd()
        parent_dir = location
        rad = (random.random())*100
```

4. TIME :

THIS IS AN EXTREMELY HANDY MODULE AND IS USED HERE TO IMPROVE THE USER EXPERIENCE BY DISPLAYING THINGS ONE BY ONE AFTER A PAUSE OF CERTAIN TIME PERIOD. IT IS ALSO USED TO CALCULATE BACKGROUND TIME ELAPSED IN PLAYING THE TUTORIAL OF THE GAME.

5. DATETIME :

THIS IS A VARIATION OF THE PREVIOUS MODULE. IT IS USED TO GET CURRENT DATE AND TIME ,THE TIME WHEN THE USER RUNS THE CODE. IT IS BEING USED HERE TO GET THE CURRENT TIME AND WISH THE USER “GOOD MORNING” / “GOOD AFTERNOON” AND SO ON ACCORDINGLY.

```
l = datetime.now()
m = "DATE AND TIME" + str(l)
```

6. MULTIPROCESSING :

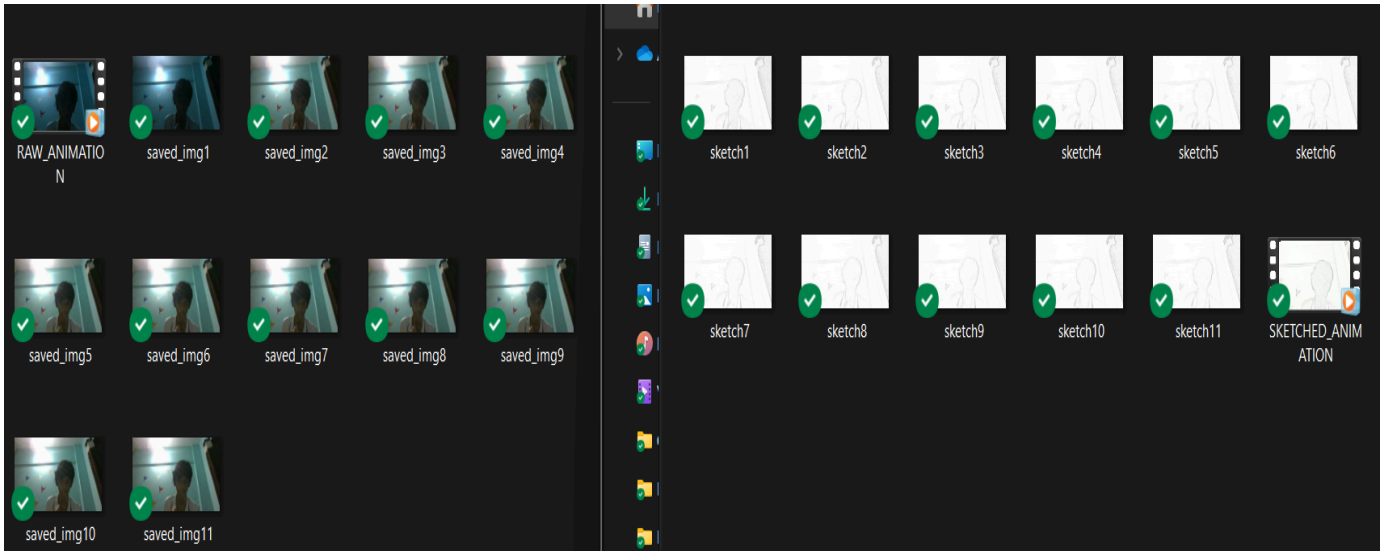
THIS MODULE HELPS IN EXECUTION OF MULTIPLE PROCESSES AT THE SAME TIME WITH MINIMAL MEMORY USAGE. FOR THIS VERY PROJECT THERE ARE TWO PRINCIPAL PROCESSES RUNNING IN THE BACKGROUND, ONE IS ANIMATION FRAMEWORK WHILE OTHER IS VIRTUAL MOUSE.

```
f t==0 and ms==0:
    try:
        class main_program(multiprocessing.Process):
```

```
class virtual_mouse(multiprocessing.Process):

    def run(self):
```

7. OPENCV :

THIS MODULE HELPS IN CAPTURING IMAGES FROM THE WEBCAM AND PROCESSING THEM TO MAKE A SPECIFIC FPS AND TIMED VIDEO.



8. CSV :

THIS MODULE HELPS IN READING AND WRITING COMMA SEPARATED VALUES (CSV) FILES. IT IS BEING USED HERE TO PRINT AND STORE THE GAME HIGH SCORE OF A PARTICULAR USER IN A .csv FILE.

```
Original I  14:03:32
System B  18:20:51
System Manufacturer:      Acer
System Model:          Nitro AN515-45
System Type:          x64-based PC
Processor(s):          1 Processor(s) Installed.
                [01]: AMD64 Family 25 Model 80 Stepping 0 AuthenticAMD ~3301 Mhz
BIOS Vers 02-08-2022
Windows Directory:       C:\WINDOWS
System Directory:       C:\WINDOWS\system32
Boot Device:          \Device\HarddiskVolume1
System Locale:          en-us;English (United States)
Input Locale:          00004009
Time Zon  Kolkata    Mumbai    New Delhi
Total Phy 532 MB
Available 445 MB
Virtual M 796 MB
Virtual M 592 MB
Virtual M 204 MB
Page File Location(s):    C:\pagefile.sys
Domain:              WORKGROUP
Logon Server:          \\ARYA-PC
Hotfix(s):          4 Hotfix(s) Installed.
                [01]: KB5020880
                [02]: KB5012170
                [03]: KB5021255
                [04]: KB5020487
Network Card(s):        4 NIC(s) Installed.
                [01]: Killer E2600 Gigabit Ethernet Controller
                    Connection Name: Ethernet
                    Status:       Media disconnected
                [02]: MediaTek Wi-Fi 6 MT7921 Wireless LAN Card
                    Connection Name: Wi-Fi
                    DHCP Enabled:    Yes
                    DHCP Server:    192.168.0.1
                    IP address(es)
                    [01]: 192.168.0.5
                    [02]: fe80::caa4:7033:4be2:cbb5
```

9. SELF MADE MODULE (MOONSHINE_SUNSHINE) :

HERE A MODULE IS CREATED BY US WHEREIN THE INFORMATION AND ALGORITHM TO SUCCESSFULLY CALCULATE A VIRTUAL MOUSE MOVEMENT AND DIFFERENTIATE IT WITH A CLICK ,IS CALCULATED.

```python
import cv2
import time
import math
import mediapipe as mp

class handDetector():
    def __init__(self, mode=False, maxHands=2,modelComplexity=1, detectionCon=0.5, trackCon=0.5):
        self.mode = mode
        self.maxHands = maxHands
        self.modelComplex = modelComplexity
        self.detectionCon = detectionCon
        self.trackCon = trackCon

        self.mpHands = mp.solutions.hands
        self.hands = self.mpHands.Hands(self.mode, self.maxHands,self.modelComplex,
                                        self.detectionCon, self.trackCon)
        self.mpDraw = mp.solutions.drawing_utils
        self.tipIds = [4, 8, 12, 16, 20]

    def findHands(self, img, draw=True):
        imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        self.results = self.hands.process(imgRGB)

        if self.results.multi_hand_landmarks:
            for handLms in self.results.multi_hand_landmarks:
                if draw:
                    self.mpDraw.draw_landmarks(img, handLms,
                                               self.mpHands.HAND_CONNECTIONS)
```

SOURCE CODE

MAIN.PY

```
'''

TESTED ON :

Processor        AMD Ryzen 5 5600H with Radeon Graphics 3.30 GHz
Installed RAM    8.00 GB (7.36 GB usable)
System type      64-bit operating system, x64-based processor

Edition Windows 11 Home Single Language
Version 21H2
OS build 22000.708
Experience Windows Feature Experience Pack 1000.22000.708.0

Python Version 3.7.9 64-bit

MADE WITH LOVE
BY KARYA (KARTHIK & ARYA)

'''


import multiprocessing
import random

import os
try:
    os.system('pip install -q -r requirements.txt')
    ms=0
except:
    ms=1
import cv2
import os.path
import glob
import csv
import time as ti
from datetime import datetime
import subprocess
import sys
import psutil
from tkinter import *
import numpy as np
import moonshine_sunshine as htm
t=0
try:
    import autopy
except Exception as ImportError:
    print(ImportError)
    t=1

if t==0 and ms==0:
    try:
        class main_program(multiprocessing.Process):
            def run(self):
                global rad
                start_time = ti.time()
                Id = subprocess.check_output(['systeminfo']).decode('utf-8').split('\n')
                location = os.getcwd()
                parent_dir = location
                rad = (random.random())*100
```

```
def delete3():
    screen3.destroy()


def delete4():
    screen4.destroy()

def delete5():
    screen5.destroy()

def delete():
    screen.destroy()

def save_time():
    global save_TIME
    save_TIME = TIME.get()

def save_fps():
    global save_FPS
    save_FPS = FPS.get()

def save_webcam():
    global save_WEBCAM
    save_WEBCAM = WEBCAM.get()

def save_play():
    global save_PLAY
    save_PLAY = PLAY.get()

def save_save():
    global save_SAVE
    save_SAVE = SAVE.get()




def time():


    global screen1
    screen1 = Toplevel(screen)
    screen1.title("DURATION")
    screen1.geometry("600x300")

    global TIME
    global time_entry
    TIME = StringVar()

    Label(screen1, text = "PLEASE ENTER DURATION OF VIDEO",fg='red',font=("Times New Roman", 14,'bold')).pack()
    Label(screen1, text = "").pack()
    Label(screen1, text = "DURATION * ",fg='cyan',font=("Times New Roman", 12,'bold')).pack()

    time_entry = Entry(screen1, textvariable = TIME)
    time_entry.pack()
    Button(screen1, text = "SAVE", width = 20, height = 2, fg='green',font=("Times New Roman", 12,'bold'),command = save_time).pack()
    Label(screen1, text = "").pack()
    Button(screen1, text = "PROCEED", width = 20, height = 2, fg='green',font=("Times New Roman", 12,'bold'),command = delete1).pack()
```

```
def fps():

    global screen2
    screen2 = Toplevel(screen)
    screen2.title("FRAMES PER SECOND")
    screen2.geometry("600x300")

    global FPS
    global fps_entry
    FPS = StringVar()

    Label(screen2, text = "PLEASE ENTER REQUIRED FPS",fg='red',font=("Times New Roman", 14,'bold')).pack()
    Label(screen2, text = "").pack()
    Label(screen2, text = "FPS * ",fg='cyan',font=("Times New Roman", 12,'bold')).pack()

    fps_entry = Entry(screen2, textvariable = FPS)
    fps_entry.pack()
    Button(screen2, text = "SAVE", width = 20, height = 2, fg='green',font=("Times New Roman", 12,'bold'),command = save_fps).pack()
    Label(screen2, text = "").pack()
    Button(screen2, text = "PROCEED", width = 20, height = 2, fg='green',font=("Times New Roman", 12,'bold'),command = delete2).pack()


def webcam():

    global screen3
    screen3 = Toplevel(screen)
    screen3.title("WEBCAM ACCESS")
    screen3.geometry("600x300")

    global WEBCAM
    global webcam_entry
    WEBCAM = StringVar()

    Label(screen3, text = "DO YOU HAVE EXTERNAL WEBCAM?",fg='red',font=("Times New Roman", 14,'bold')).pack()
    Label(screen3, text = "").pack()
    Label(screen3, text = "WEBCAM INFO * ",fg='cyan',font=("Times New Roman", 12,'bold')).pack()

    webcam_entry = Entry(screen3, textvariable = WEBCAM)
    webcam_entry.pack()
    Button(screen3, text = "SAVE", width = 20, height = 2, fg='green',font=("Times New Roman", 12,'bold'),command = save_webcam).pack()
    Label(screen3, text = "").pack()
    Button(screen3, text = "PROCEED", width = 20, height = 2, fg='green',font=("Times New Roman", 12,'bold'),command = delete3).pack()


def save():

    global screen4
    screen4 = Toplevel(screen)
    screen4.title("KEEP OR NOT")
    screen4.geometry("600x300")

    global SAVE
    global save_entry
    SAVE = StringVar()

    Label(screen4, text = "DO YOU WISH TO KEEP CAPTURED IMAGES?",fg='red',font=("Times New Roman", 14,'bold')).pack()
    Label(screen4, text = "").pack()
    Label(screen4, text = "YOUR ANSWER * ",fg='cyan',font=("Times New Roman", 12,'bold')).pack()

    save_entry = Entry(screen4, textvariable = SAVE)
```

```
def playo():

    global screen5
    screen5 = Toplevel(screen)
    screen5.title("PLAY OR NOT")
    screen5.geometry("600x300")

    global PLAY
    global play_entry
    PLAY = StringVar()

    Label(screen5, text = "ENTER 1 TO PLAY ANIMATED VIDEO",fg='red',font=("Times New Roman", 14,'bold')).pack()
    Label(screen5, text = "").pack()
    Label(screen5, text = "YOUR ANSWER * ",fg='cyan',font=("Times New Roman", 12,'bold')).pack()

    play_entry = Entry(screen5, textvariable = PLAY)
    play_entry.pack()
    Button(screen5, text = "SAVE", width = 20, height = 2, fg='green',font=("Times New Roman", 12,'bold'),command = save_play).pack()
    Label(screen5, text = "").pack()
    Button(screen5, text = "PROCEED", width = 20, height = 2, fg='green',font=("Times New Roman", 12,'bold'),command = delete5).pack()


def main_screen():

    global screen
    screen = Tk()
    screen.geometry("600x300")
    screen.title("DETAILS")
    screen.configure(background='black')
    Label(text = "SUBMIT YOUR DETAILS", fg='black',bg='white', width = "600", height = "4", font = ("Times New Roman", 16)).pack()
    Label(text = "").pack()


    Button(text = "TIME", height = "4", width = "60", fg='white',bg='black',font=("Times New Roman", 14,'bold'),command = time).pack()


    Label(text = "").pack()


    Button(text = "FPS",height = "4", width = "60", fg='white',bg='black',font=("Times New Roman", 14,'bold'),command = fps).pack()


    Label(text = "").pack()


    Button(text = "EXTERNAL WEBCAM", height = "4", width = "60", fg='white',bg='black',font=("Times New Roman", 14,'bold'),command = webcam).pack(


    Label(text = "").pack()


    Button(text = "REGARDING SAVING",height = "4", width = "60", fg='white',bg='black',font=("Times New Roman", 14,'bold'),command = save).pack()


    Label(text = "").pack()

    Button(text = "PLAY ON END" ,height = "4", width = "60", fg='white',bg='black',font=("Times New Roman", 14,'bold'),command = playo).pack()


    Label(text = "").pack()
```

```
main_screen()

try:
    frames = int(save_FPS)*int(save_TIME)
except Exception as CriticalError:
    pass

try:
    directory = f"RAW_IMAGES{rad}"
    path = os.path.join(parent_dir, directory)
    os.mkdir(path)
    directory = f"SKETCHED_IMAGES{rad}"
    parent_dir = location
    path = os.path.join(parent_dir, directory)
    os.mkdir(path)

except Exception as error1:
    pass

try:
    directory = f"RUNTIME_LOGS"
    parent_dir = location
    path = os.path.join(parent_dir, directory)
    os.mkdir(path)
except Exception as error2:
    pass

try:
    key = cv2. waitKey(1)
    if save_WEBCAM.lower() == 'no':
        webcam = cv2.VideoCapture(0, apiPreference=cv2.CAP_ANY, params=[
        cv2.CAP_PROP_FRAME_WIDTH, 100000,
        cv2.CAP_PROP_FRAME_HEIGHT, 100000])

        width = int(webcam.get(cv2.CAP_PROP_FRAME_WIDTH))
        height = int(webcam.get(cv2.CAP_PROP_FRAME_HEIGHT))
    else:
        webcam = cv2.VideoCapture(1, apiPreference=cv2.CAP_ANY, params=[
        cv2.CAP_PROP_FRAME_WIDTH, 100000,
        cv2.CAP_PROP_FRAME_HEIGHT, 100000])

        width = int(webcam.get(cv2.CAP_PROP_FRAME_WIDTH))
        height = int(webcam.get(cv2.CAP_PROP_FRAME_HEIGHT))
except Exception as error3:
    pass

ti.sleep(2)


k = 0
try:
    while k<frames+1:
        k += 1
        try:
            check, frame = webcam.read()
            cv2.imshow("Capturing", frame)
            key = cv2.waitKey(1)
            os.chdir(f'{location}/RAW_IMAGES{rad}')
```

```
            img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            img_invert = cv2.bitwise_not(img_gray)
            img_smoothing = cv2.GaussianBlur(img_invert, (21, 21),sigmaX=0, sigmaY=0)
            final = cv2.divide(img_gray, 255 - img_smoothing, scale=255)
            os.chdir(f'{location}/SKETCHED_IMAGES{rad}')
            cv2.imwrite(f'sketch{k}.jpg',final)
            if key == ord('q'):
                webcam.release()
                cv2.destroyAllWindows()
                break

        except(KeyboardInterrupt):
            print("Turning off camera.")
            webcam.release()
            print("Camera off.")
            print("Program ended.")
            cv2.destroyAllWindows()
            break
except Exception as error4:
    pass
os.chdir(f'{parent_dir}/SKETCHED_IMAGES{rad}')
fourcc = cv2.VideoWriter_fourcc(*'mp4v')
video = cv2.VideoWriter('SKETCHED_ANIMATION.avi', fourcc, int(save_FPS), (width,height))


try:

    for i in range(1,frames+1):
        img = cv2.imread(f'sketch{i}.jpg')
        video.write(img)

    cv2.destroyAllWindows()
    video.release()

    os.chdir(f'{parent_dir}/RAW_IMAGES{rad}')
    fourcc = cv2.VideoWriter_fourcc(*'mp4v')
    video = cv2.VideoWriter('RAW_ANIMATION.avi', fourcc, int(save_FPS), (width,height))

    for i in range(1,frames+1):
        img = cv2.imread(f'saved_img{i}.jpg')
        video.write(img)

    cv2.destroyAllWindows()
    video.release()
except Exception as error5:
    pass

try:
    if save_SAVE != 'yes':
        os.chdir(f'{parent_dir}/RAW_IMAGES{rad}')
        for k in range(1,frames+2):
            removing_files = glob.glob(f'saved_img{k}.jpg')
            for i in removing_files:
                os.remove(i)

        os.chdir(f'{parent_dir}/SKETCHED_IMAGES{rad}')
        for k in range(1,frames+2):
            removing_files = glob.glob(f'sketch{k}.jpg')
            for i in removing_files:
                os.remove(i)

except Exception as error6:
```

```
            ret, frame = cap.read()
            if ret == True:

                cv2.imshow('Frame', frame)


                if cv2.waitKey(25) & 0xFF == ord('q'):
                    break


            else:
                break

        cap.release()


        cv2.destroyAllWindows()

    else:
        print("THANK YOU")

except Exception as error7:
    pass


end = ti.time()
elap = end-start_time
elap = "TOTAL ELAPSED TIME" + str(elap)

os.chdir(f"{parent_dir}/RUNTIME_LOGS")
myf = open(f'log{rad}.csv','a+')
csvwriter = csv.writer(myf,delimiter = " ")

p = "PYTHON VERSION" + sys.version
Id.insert(0,"SYSTEM INFO.")
for i in Id:
    if i.isspace():
        Id.remove(i)

l = datetime.now()
m = "DATE AND TIME" + str(l)

csvwriter.writerow(Id)
csvwriter.writerow(p)
csvwriter.writerow(m)
csvwriter.writerow(elap)


p = psutil.Process()
for i in range(20):
    if i==19:
        x = p.io_counters()
csvwriter.writerow("READ AND WRITE SPEED "+ str(x))

cam_info = f"CAMERA RESOLUTION : {width},{height}"

csvwriter.writerow(cam_info)
id1 = os.getpid()
id3 = os.getppid()

csvwriter.writerow(f"process id of main process is : {id1}")
csvwriter.writerow(f"parent process id of main process is : {id3}")
```

```
        try:
            csvwriter.writerow(error1) or csvwriter.writerow(error2) or csvwriter.writerow(error3) or csvwriter.writerow(error4) or csvwriter.writerow(error5) or csvwriter.writerow(error6) or csvwriter.writerow(error7) or csvwriter.writerow(CriticalError) or c
        except:
            csvwriter.writerow("SUCCESSFULLY EXECUTED THE PROGRAM : ()")

        myf.close()

        print("CHECK LOGS FOR MORE INFORMATION")
        print("**IGNORE**")


class virtual_mouse(multiprocessing.Process):

    def run(self):

        try:
            wCam, hCam = 640, 480
            frameR = 100
            smoothening = 7
            pTime = 0
            plocX, plocY = 0, 0
            clocX, clocY = 0, 0

            cap = cv2.VideoCapture(0)
            cap.set(3, wCam)
            cap.set(4, hCam)
            detector = htm.handDetector(maxHands=2)
            wScr, hScr = autopy.screen.size()
            while True:

                success, img = cap.read()
                img = detector.findHands(img)
                lmList, bbox = detector.findPosition(img)
                if len(lmList) != 0:
                    x1, y1 = lmList[8][1:]
                    x2, y2 = lmList[12][1:]

                fingers = detector.fingersUp()

                cv2.rectangle(img, (frameR, frameR), (wCam - frameR, hCam - frameR),
                (255, 0, 255), 2)

                if fingers[1] == 1 and fingers[2] == 0:

                    x3 = np.interp(x1, (frameR, wCam - frameR), (0, wScr))
                    y3 = np.interp(y1, (frameR, hCam - frameR), (0, hScr))

                    clocX = plocX + (x3 - plocX) / smoothening
                    clocY = plocY + (y3 - plocY) / smoothening


                    autopy.mouse.move(wScr - clocX, clocY)
                    cv2.circle(img, (x1, y1), 15, (255, 0, 255), cv2.FILLED)
                    plocX, plocY = clocX, clocY

                if fingers[1] == 1 and fingers[2] == 1:

                    length, img, lineInfo = detector.findDistance(8, 12, img)
```

```
                    if fingers[1] == 1 and fingers[2] == 0:

                        x3 = np.interp(x1, (frameR, wCam - frameR), (0, wScr))
                        y3 = np.interp(y1, (frameR, hCam - frameR), (0, hScr))

                        clocX = plocX + (x3 - plocX) / smoothening
                        clocY = plocY + (y3 - plocY) / smoothening


                        autopy.mouse.move(wScr - clocX, clocY)
                        cv2.circle(img, (x1, y1), 15, (255, 0, 255), cv2.FILLED)
                        plocX, plocY = clocX, clocY

                    if fingers[1] == 1 and fingers[2] == 1:

                        length, img, lineInfo = detector.findDistance(8, 12, img)


                        if length < 30:
                            cv2.circle(img, (lineInfo[4], lineInfo[5]),
                            15, (0, 255, 0), cv2.FILLED)
                            autopy.mouse.click()

                    cTime = ti.time()
                    fps = 1 / (cTime - pTime)
                    pTime = cTime
                    cv2.putText(img, str(int(fps)), (20, 50), cv2.FONT_HERSHEY_PLAIN, 3,
                    (255, 0, 0), 3)

                    cv2.imshow("RC1 CAM", img)
                    cv2.setWindowProperty("RC1 CAM", cv2.WND_PROP_TOPMOST, 1)

                    if cv2.waitKey(1) & 0xFF == ord('q'):
                        break
                cv2.destroyAllWindows()
            except:
                print("unhandled exception in virtual mouse")

        t1 = main_program()
        t2 = virtual_mouse()

        if __name__ == "__main__":
            askng = input("WOULD YOU LIKE TO USE VIRTUAL MOUSE? : ").lower()
            t1.start()
            if askng == 'yes':
                print("WHILE ON CAM VIEW SCREEN , PRESS {q} TO QUIT VIRTUAL MOUSE")
                t2.start()

            try:
                t2.join()
            except:
                print("TRYNNA USE VIRTUAL MOUSE NEXT TIME")
            t1.join()
            exit()

    except:
        print("OPERATION ENDED")

else:
    print("autopy IMPORT ERROR")
```

SUBMODULE.PY

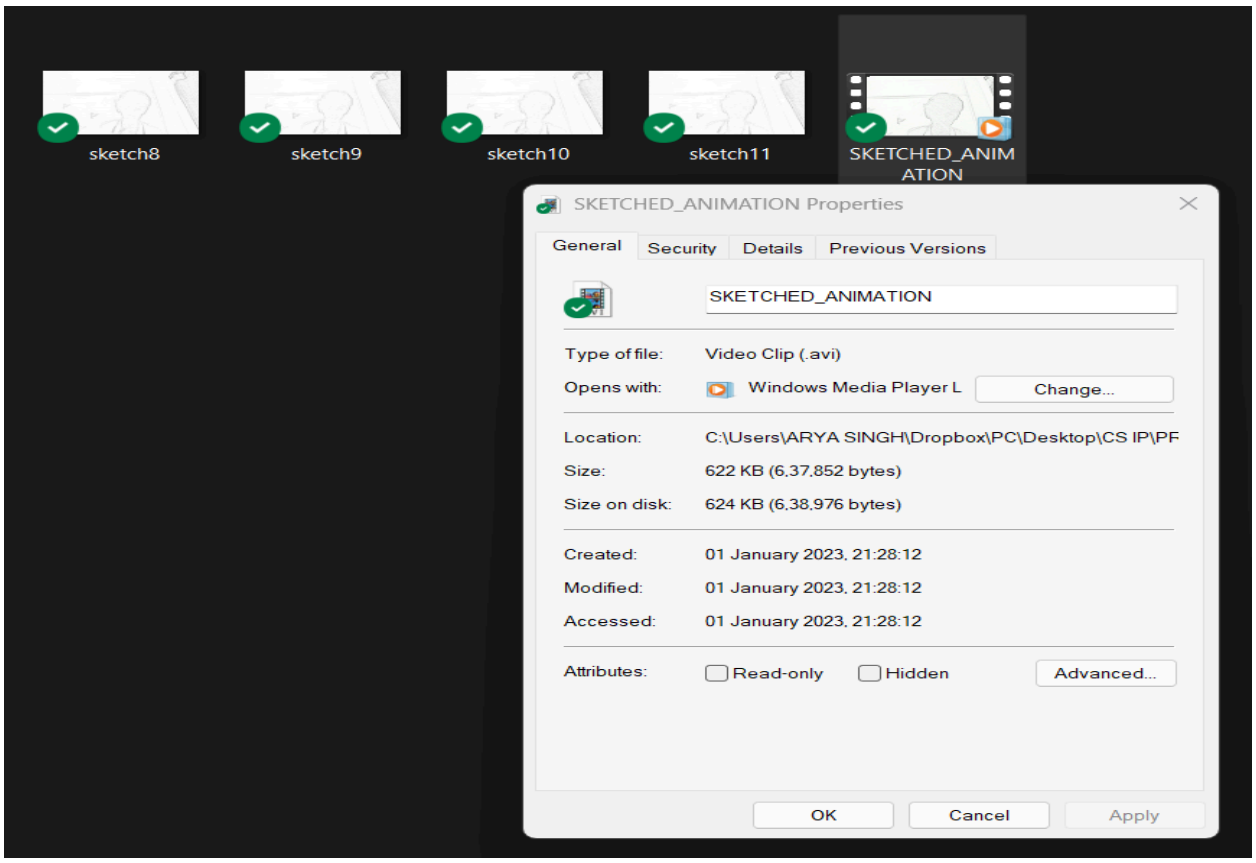
```
import cv2
import time
import math
import mediapipe as mp

class handDetector():
    def __init__(self, mode=False, maxHands=2,modelComplexity=1, detectionCon=0.5, trackCon=0.5):
        self.mode = mode
        self.maxHands = maxHands
        self.modelComplex = modelComplexity
        self.detectionCon = detectionCon
        self.trackCon = trackCon

        self.mpHands = mp.solutions.hands
        self.hands = self.mpHands.Hands(self.mode, self.maxHands,self.modelComplex,
                                        self.detectionCon, self.trackCon)
        self.mpDraw = mp.solutions.drawing_utils
        self.tipIds = [4, 8, 12, 16, 20]

    def findHands(self, img, draw=True):
        imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        self.results = self.hands.process(imgRGB)

        if self.results.multi_hand_landmarks:
            for handLms in self.results.multi_hand_landmarks:
                if draw:
                    self.mpDraw.draw_landmarks(img, handLms,
                                               self.mpHands.HAND_CONNECTIONS)

        return img

    def findPosition(self, img, handNo=0, draw=True):
        xList = []
        yList = []
        bbox = []
        self.lmList = []
        if self.results.multi_hand_landmarks:
            myHand = self.results.multi_hand_landmarks[handNo]
            for id, lm in enumerate(myHand.landmark):

                h, w, c = img.shape
                cx, cy = int(lm.x * w), int(lm.y * h)
                xList.append(cx)
                yList.append(cy)

                self.lmList.append([id, cx, cy])
                if draw:
                    cv2.circle(img, (cx, cy), 5, (255, 0, 255), cv2.FILLED)

            xmin, xmax = min(xList), max(xList)
            ymin, ymax = min(yList), max(yList)
            bbox = xmin, ymin, xmax, ymax

            if draw:
                cv2.rectangle(img, (xmin - 20, ymin - 20), (xmax + 20, ymax + 20),
                              (0, 255, 0), 2)

        return self.lmList, bbox

    def fingersUp(self):
        fingers = []
```

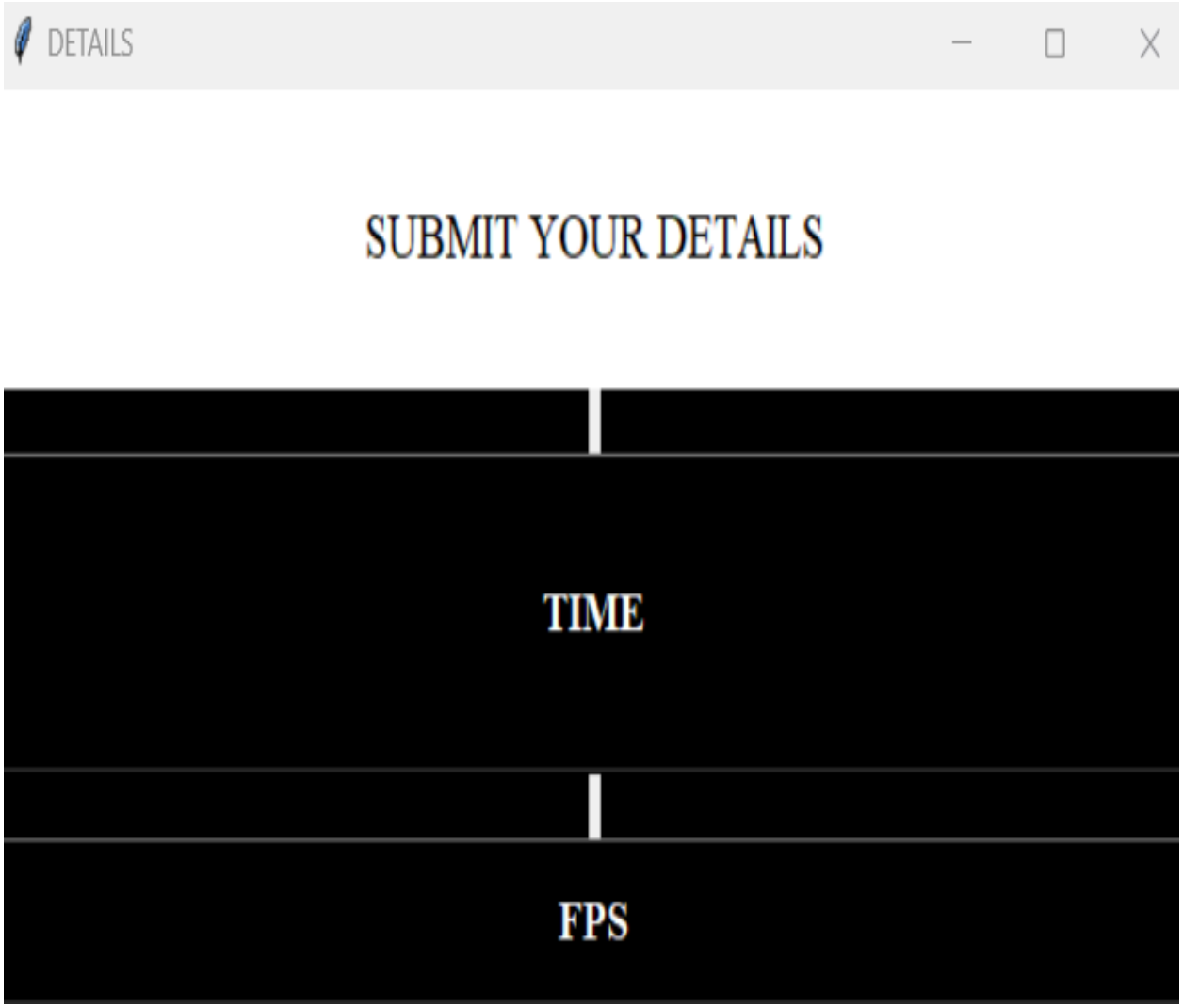
```
        try:
            if self.lmList[self.tipIds[0]][1] > self.lmList[self.tipIds[0]-1][1]:
                fingers.append(1)
            else:
                fingers.append(0)
        except:
            pass

        for id in range(1, 5):

            if self.lmList[self.tipIds[id]][2] < self.lmList[self.tipIds[id] - 2][2]:
                fingers.append(1)
            else:
                fingers.append(0)



        return fingers

    def findDistance(self, p1, p2, img, draw=True,r=15, t=3):
        x1, y1 = self.lmList[p1][1:]
        x2, y2 = self.lmList[p2][1:]
        cx, cy = (x1 + x2) // 2, (y1 + y2) // 2

        if draw:
            cv2.line(img, (x1, y1), (x2, y2), (255, 0, 255), t)
            cv2.circle(img, (x1, y1), r, (255, 0, 255), cv2.FILLED)
            cv2.circle(img, (x2, y2), r, (255, 0, 255), cv2.FILLED)
            cv2.circle(img, (cx, cy), r, (0, 0, 255), cv2.FILLED)
        length = math.hypot(x2 - x1, y2 - y1)

        return length, img, [x1, y1, x2, y2, cx, cy]


def main():
    pTime = 0
    cTime = 0
    cap = cv2.VideoCapture(0)
    detector = handDetector()
    while True:
        success, img = cap.read()
        img = detector.findHands(img)
        lmList, bbox = detector.findPosition(img)


        cTime = time.time()
        fps = 1 / (cTime - pTime)
        pTime = cTime

        cv2.putText(img, str(int(fps)), (10, 70), cv2.FONT_HERSHEY_PLAIN, 3,
                    (255, 0, 255), 3)

        cv2.imshow("Image", img)
        cv2.setWindowProperty("Image", cv2.WND_PROP_TOPMOST, 1)

        cv2.waitKey(1)



if __name__ == "__main__":
    main()
```

SOURCE CODE AND OTHER ASSETS

OUTPUT

```
[notice] A new release of pip available: 22.2.1 -> 22.3.1
[notice] To update, run: C:\Users\ARYA SINGH\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.7_qbz5n2kfra8p0\python.e
xe -m pip install --upgrade pip
WOULD YOU LIKE TO USE VIRTUAL MOUSE? : NO
TRYNNA USE VIRTUAL MOUSE NEXT TIME

[notice] A new release of pip available: 22.2.1 -> 22.3.1
[notice] To update, run: C:\Users\ARYA SINGH\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.7_qbz5n2kfra8p0\python.e
xe -m pip install --upgrade pip
CHECK LOGS FOR MORE INFORMATION
**IGNORE**
[ WARN:1@61.349] global D:\a\opencv-python\opencv-python\opencv\modules\videoio\src\cap_msmf.cpp (539) `anonymous-namespace'::SourceReaderC
B::~SourceReaderCB terminating async callback
OPERATION ENDED
```





BACKEND FILES CREATED

__pycache__	01-01-2023 21:11	File folder
RAW_IMAGES79.2807063129436	01-01-2023 21:28	File folder
RUNTIME_LOGS	01-01-2023 21:28	File folder
SKETCHED_IMAGES79.2807063129436	01-01-2023 21:28	File folder

LEARNING POINTS

THIS PROJECT TEACHES US MANY THINGS , ALL AT ONCE. IT USES A COMPLEX ALGORITHM AND THE FLOW OF CONTROL IS MADE IN SUCH A WAY THAT ALMOST ALL ERRORS CAN BE MINIMIZED. OBVIOUSLY THERE ARE MANY LIBRARIES USED WHICH ULTIMATELY IMPROVES THE VISUAL QUALITY OF THE PROJECT AND UI. KEY LEARNINGS ARE :

I. CSV FILE HANDLING
II. TKINTER INTERACTIVE GRAPHICS
III. TKINTER WIDGETS
IV. FUNCTIONS AND LOOPS
V. IF ELSE STATEMENTS AND TRY EXCEPT STATEMENTS
VI. MEMORY HANDLING
VII. INPUT HANDLING
VIII. LIST , DICTIONARIES AND FORMATTING
IX. MODULE INTEGRATION AND OPTIMISATION

THERE IS NO END TO LEARNING AND THE ABOVE MENTIONED TOPICS WERE THE KEY LEARNINGS OF THE PROJECT. THERE ARE MANY MORE LEARNINGS WHICH CAN ALSO BE TAKEN.

LIMITATIONS

NO PROJECT IS COMPLETE AND THIS PROJECT IS NO DIFFERENT. THERE CAN BE MANY ERRORS THAT CAN OCCUR DURING EXECUTION OF A COMPLEX PROGRAM AND THERE CAN BE MANY CHANGES / CORRECTIONS THAT CAN BE IMPLEMENTED IN A CODE TO IMPROVE THE EFFECTIVENESS AND THE UI OF THE PROGRAM. SOME OF THE IDENTIFIED LIMITATIONS OF THIS PROJECT ARE :

1. MULTIPROCESSING IS MUCH MORE INEFFICIENT AS COMPARED TO RUNNING FILE SEPARATELY. HENCE MULTIPROCESSING NEEDS TO BE OPTIMIZED AGAIN TO IMPROVE THE USER EXPERIENCE.
2. THIS PROGRAM NEEDS VARIOUS LIBRARIES AND ASSETS WHICH ULTIMATELY INCREASES MINIMUM SYSTEM REQUIREMENTS TO RUN THE CODE AS THE RAM AND CPU USAGE INCREASES. THE CODE ALSO INCREASES LOAD ON CPU AND REDUCES USER EXPERIENCE BY INCREASING COMPUTATION TIME.

MOREOVER , THESE LIMITATIONS CAN BE SOLVED BY MAKING SOME UPDATES TO THE CODE AND USING MUCH MORE EFFICIENT MODULES.

REFERENCES

1. TKINTER DOCUMENTATION
2. MULTIPROCESSING DOCUMENTATION
3. OPENCV DOCUMENTATION
4. PYTHON PLAYLIST BY “CODEWITHHARRY” YOUTUBE CHANNEL
5. VIRTUAL MOUSE FILE HANDLING ON STACK OVERFLOW AND GEEKSFORGEEKS
6. SEVERAL ANONYMOUS NEWSPAPER TITLES
7. MEMORY MANAGEMENT BY TIM
8. ALGORITHM AND FLOW DESIGNING ARTICLES

THANK YOU