STUDENT MONITORING SYSTEM

App.py

```
from flask import Flask, render_template, request, session, flash
import sys
import mysql.connector

app = Flask(__name__)
app.config['SECRET_KEY'] = 'aaa'
@app.route('/')
def home():
    return render_template('index.html')
@app.route('/AdminLogin')
def AdminLogin():
    return render_template('AdminLogin.html')
@app.route('/Report')
def Report():
    conn = mysql.connector.connect(user='root', password='', host='localhost', database='1studentdb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM reporttb")
    data = cur.fetchall()
    return render_template('Report.html',data=data)


@app.route("/adminlogin", methods=['GET', 'POST'])
def adminlogin():
    error = None
    if request.method == 'POST':
        if request.form['uname'] == 'admin' and request.form['password'] == 'admin':
            return render_template('AdminHome.html')
        else:
            return render_template('index.html', error=error)
@app.route("/AdminHome")
def AdminHome():
    conn = mysql.connector.connect(user='root', password='', host='localhost', database='1studentdb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM regtb")
    data = cur.fetchall()
    return render_template('AdminHome.html', data=data)
@app.route("/AStudentInfo")
def AStudentInfo():
    conn = mysql.connector.connect(user='root', password='', host='localhost', database='1studentdb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM studenttb")
    data = cur.fetchall()
    return render_template('AStudentInfo.html', data=data)
@app.route('/NewStudent')
def NewStudent():
    return render_template('NewStudent.html')
@app.route("/newstudent", methods=['GET', 'POST'])
def newstudent():
    if request.method == 'POST':
        regno = request.form['regno']
        uname = request.form['uname']
        gender = request.form['gender']
        mobile = request.form['mobile']
        email = request.form['email']
        address = request.form['Address']
        depart = request.form['depart']
        conn = mysql.connector.connect(user='root', password='', host='localhost', database='1studentdb')
        cursor = conn.cursor()
        cursor.execute(
            "insert into studenttb values('','" + regno + "','" + uname + "','" + gender + "','" + mobile + "','" + email + "','" +
address + "' ,'" + depart + "')")
        conn.commit()
        conn.close()
        conn = mysql.connector.connect(user='root', password='', host='localhost', database='1studentdb')
        cur = conn.cursor()
        cur.execute("SELECT * FROM studenttb  ")
        data = cur.fetchall()

        flash("Record Saved!")
        return render_template('AStudentInfo.html', data=data)
@app.route("/newreport", methods=['GET', 'POST'])
def newreport():
    if request.method == 'POST':
        regno = request.form['regno']
        Reason = request.form['Reason']
        import datetime
        date = datetime.datetime.now().strftime('%Y-%m-%d')
        time = datetime.datetime.now().strftime('%H:%M:%S')
        conn = mysql.connector.connect(user='root', password='', host='localhost', database='1studentdb')
        cursor = conn.cursor()
        cursor.execute(
            "insert into reporttb values('','" + regno + "','" + date + "','" + time + "','" + Reason + "')")
        conn.commit()
        conn.close()
        conn = mysql.connector.connect(user='root', password='', host='localhost', database='1studentdb')
        cur = conn.cursor()
        cur.execute("SELECT * FROM studenttb  ")
        data = cur.fetchall()
        flash("Record Saved!")
        return render_template('AStudentInfo.html', data=data
@app.route('/FStudentInfo')
def FStudentInfo():
    conn = mysql.connector.connect(user='root', password='', host='localhost', database='1studentdb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM studenttb  ")
    data = cur.fetchall()
    return render_template('FStudentInfo.html', data=data)
def sendmsg(targetno, message):
    import requests
    requests.post(
      "http://smsserver9.creativepoint.in/api.php?username=fantasy&password=596692&to=" + targetno +
"&from=FSSMSS&message=Dear user  your msg is " + message + " Sent By FSMSG
FSSMSS&PEID=1501563800000030506&templateid=1507162882948811640")
@app.route("/Remove")
def Remove():
    id = request.args.get('id')
    conn = mysql.connector.connect(user='root', password='', host='localhost', database='1studentdb')
    cursor = conn.cursor()
    cursor.execute("delete from  studenttb  where id='" + id + "' ")
    conn.commit()
    conn.close()
    conn = mysql.connector.connect(user='root', password='', host='localhost', database='1studentdb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM studenttb ")
    data = cur.fetchall()
    return render_template('FStudentInfo.html', data=data)
@app.route("/Remove1")
def Remove1():
    id = request.args.get('id')
    conn = mysql.connector.connect(user='root', password='', host='localhost', database='1studentdb')
    cursor = conn.cursor()
    cursor.execute("delete from  studenttb  where id='" + id + "' ")
    conn.commit()
    conn.close()
    conn = mysql.connector.connect(user='root', password='', host='localhost', database='1studentdb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM studenttb ")
    data = cur.fetchall()
    return render_template('AStudentInfo.html', data=data)
@app.route('/StudentLogin')
def StudentLogin():
    return render_template('StudentLogin.html')
@app.route("/StudentHome")
def StudentHome():
    regno = session['regno']
    conn = mysql.connector.connect(user='root', password='', host='localhost', database='1studentdb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM studenttb where RegisterNo='" + regno + "' ")
    data = cur.fetchall()
    return render_template('StudentHome.html', data=data)
@app.route("/AdminResult")
def AdminResult():
    conn = mysql.connector.connect(user='root', password='', host='localhost', database='1studentdb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM answertb ")
    data = cur.fetchall()
    return render_template('AdminResult.html', data=data)
@app.route("/Scan")
def Scan():
    import cv2
    from ultralytics import YOLO  # Ensure you have the correct YOLO library installed
    # Load the YOLO model with pre-trained weights
    model = YOLO('runs/detect/personcount/weights/best.pt')

    # Open webcam
    cap = cv2.VideoCapture(0)
    dd1 = 0
    previous_count = 0
    stt = ''
    ddd = 0
    ddd11 = 0
    ddd22 = 0
    while cap.isOpened():
```

```
ret, frame = cap.read()
if not ret:
    break
# Perform object detection with a confidence threshold of 0.1
results = model(frame, conf=0.3)
if results and results[0].boxes:
    # Get the class names
    class_labels = results[0].names
    # Get detected class indices
    detected_classes = results[0].boxes.cls.cpu().numpy().astype(int)
    # Count "person" detections
    person_count = sum(1 for class_index in detected_classes if class_labels[class_index] == "person")
    # Notify if count changes
    dd1 += 1
    print(dd1)
    if dd1 == 30:
        ddd = person_count
    if dd1 > 30:
        if ddd < previous_count:
            print("Person count increased!")
            stt = "increased"
            ddd11 += 1
            if ddd11 > 30:
                session['st'] = "IN"
                cap.release()
                cv2.destroyAllWindows()
                # annotated_frame = results[0].plot()
                # cv2.imwrite("alert.jpg", annotated_frame)
                # sendmail()
                # sendmsg("9087259509", 'Person count increased!'+str(person_count))
                conn = mysql.connector.connect(user='root', password='', host='localhost',
                                               database='1studentdb')
                cur = conn.cursor()
                cur.execute("SELECT RegisterNo FROM studenttb  ")
                data = cur.fetchall()
                return render_template('NewReport.html', data=data)
        elif ddd > previous_count:
            print("Person count decreased!")
            stt = "decreased"
            ddd22 += 1
            if ddd22 > 30:
                session['st'] = "OUT"
                cap.release()
                cv2.destroyAllWindows()
                # annotated_frame = results[0].plot()
                # cv2.imwrite("alert.jpg", annotated_frame)
                # sendmail()
                # sendmsg("9087259509", 'Person count increased!'+str(person_count))
                conn = mysql.connector.connect(user='root', password='', host='localhost',
                                               database='1studentdb')
                cur = conn.cursor()
                cur.execute("SELECT RegisterNo FROM studenttb  ")
                data = cur.fetchall()
                return render_template('NewReport.html', data=data)
    previous_count = person_count
```

```
            # Visualize the results
            annotated_frame = results[0].plot()
            # Display person count on the image
            text = f"Person Count: {person_count}" + stt + "SetCount" + str(ddd)
            cv2.putText(annotated_frame, text, (20, 40), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2,
cv2.LINE_AA)
        else:
            annotated_frame = frame  # Display the original frame if no detections are made
        # Show output
        cv2.imshow("YOLOv8 Head Count", annotated_frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
def sendmsg(targetno, message):
    import requests
    requests.post(

"http://sms.creativepoint.in/api/push.json?apikey=6555c521622c1&route=transsms&sender=FSSMSS&mobileno=" +
targetno + "&text=Dear customer your msg is " + message + "  Sent By FSMSG FSSMSS")
def sendmail():
    import smtplib
    from email.mime.multipart import MIMEMultipart
    from email.mime.text import MIMEText
    from email.mime.base import MIMEBase
    from email import encoders
    fromaddr = "projectmailm@gmail.com"
    toaddr = "sangeeth5535@gmail.com"
    # instance of MIMEMultipart
    msg = MIMEMultipart()

    # storing the senders email address
    msg['From'] = fromaddr
    # storing the receivers email address
    msg['To'] = toaddr
    # storing the subject
    msg['Subject'] = "Alert"
    # string to store the body of the mail
    body = "Crowd  Detection"
    # attach the body with the msg instance
    msg.attach(MIMEText(body, 'plain'))
    # open the file to be sent
    filename = "alert.jpg"
    attachment = open("alert.jpg", "rb")
    # instance of MIMEBase and named as p
    p = MIMEBase('application', 'octet-stream')
    # To change the payload into encoded form
    p.set_payload((attachment).read())
    # encode into base64
    encoders.encode_base64(p)
    p.add_header('Content-Disposition', "attachment; filename= %s" % filename)
    # attach the instance 'p' to instance 'msg'
    msg.attach(p)

    # creates SMTP session
    s = smtplib.SMTP('smtp.gmail.com', 587)
    # start TLS for security
    s.starttls()
    # Authentication
    s.login(fromaddr, "qmgn xecl bkqv musr")
    # Converts the Multipart msg into a string
    text = msg.as_string()
    # sending the mail
    s.sendmail(fromaddr, toaddr, text)
    # terminating the session
    s.quit()
if __name__ == '__main__':
    app.run(debug=True, use_reloader=True)
newcount.py:
import numpy as np
from tkinter import *
import os
from tkinter import filedialog
import cv2
import time, sys
from matplotlib import pyplot as plt
from tkinter import messagebox

def endprogram():
    print("\nProgram terminated!")
    sys.exit()
def testing():
    global testing_screen
    testing_screen = Toplevel(main_screen)
    testing_screen.title("Testing")
    # login_screen.geometry("400x300")
    testing_screen.geometry("600x450+650+150")
    testing_screen.minsize(120, 1)
    testing_screen.maxsize(1604, 881)
    testing_screen.resizable(1, 1)
    testing_screen.configure(bg='cyan')
    # login_screen.title("New Toplevel")
    Label(testing_screen, text='''Upload Image''', disabledforeground="#a3a3a3",
          foreground="#000000", width="300", height="2", bg='cyan', font=("Calibri", 16)).pack()
    Label(testing_screen, text="").pack()
    Label(testing_screen, text="").pack()
    Label(testing_screen, text="").pack()
    Button(testing_screen, text='''Upload Image''', font=(
        'Verdana', 15), height="2", width="30", bg='cyan', command=imgtest).pack()


global affect
def imgtest():
    from ultralytics import YOLO
    import cv2
    import_file_path = filedialog.askopenfilename()

    # image = cv2.imread(import_file_path)
    image = cv2.imread(import_file_path)
    model = YOLO('runs/detect/workhelmet/weights/best.pt')
    # Load the image
    # image = cv2.imread('path/to/your/image.jpg')
    # Perform object detection
    results = model(image, conf=0.2)
    # Get the predicted class labels and their confidence scores
    class_labels = results[0].names
    confidences = results[0].boxes.conf

    if len(confidences) > 0:
        max_confidence_index = confidences.argmax().item()  # Convert tensor to integer
        predicted_class = class_labels[max_confidence_index]
    else:
        predicted_class = "No Detections"

    # Print the predicted class and its confidence score
    print(f"Predicted Class: {predicted_class}")
    print(f"Confidence Score: {confidences[max_confidence_index]}")
    # Optionally, visualize the results
    annotated_frame = results[0].plot()
    # Display the annotated frame
    cv2.imshow("YOLOv8 Inference", annotated_frame)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
def image1():
    import cv2
    from tkinter import filedialog
    from ultralytics import YOLO  # Ensure you have the correct YOLO library installed

    # Open a file dialog to select an image file
    import_file_path = filedialog.askopenfilename()

    # Load the selected image
    image = cv2.imread(import_file_path)

    # Load the YOLO model with pre-trained weights
    model = YOLO('runs/detect/personcount/weights/best.pt')

    # Perform object detection with a confidence threshold of 0.1
    results = model(image, conf=0.1)

    # Extract predictions
    if results and results[0].boxes:
        # Get the class names
        class_labels = results[0].names
        # Get the confidence scores and detected class indices
        confidences = results[0].boxes.conf.cpu().numpy()  # Convert tensor to numpy array
        detected_classes = results[0].boxes.cls.cpu().numpy().astype(int)  # Get class indices
        # Count total detections
        total_detections = len(detected_classes)
        # Count "person" detections
        person_count = sum(1 for class_index in detected_classes if class_labels[class_index] == "person")
        # Find the least confident detection
        min_confidence_index = confidences.argmin()
        predicted_class = class_labels[detected_classes[min_confidence_index]]
        confidence_score = confidences[min_confidence_index]
        # Print results
        print(f"Total Detections: {total_detections}")
        print(f"Person Count: {person_count}")
        print(f"Least Confident Prediction: {predicted_class}")
        print(f"Confidence Score: {confidence_score:.2f}")

        # Visualize the results
        annotated_frame = results[0].plot()
        # Display person count on the image
        text = f"Person Count: {person_count}"
        cv2.putText(annotated_frame, text, (20, 40), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2, cv2.LINE_AA)
    else:
        print("No detections made")
        annotated_frame = image  # Display the original image if no detections are made
    # Display the annotated frame
    cv2.imshow("YOLOv8 Inference", annotated_frame)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
def Camera1():
    import cv2
    from ultralytics import YOLO  # Ensure you have the correct YOLO library installed
    # Load the YOLO model with pre-trained weights
    model = YOLO('runs/detect/personcount/weights/best.pt')
    # Open webcam
    cap = cv2.VideoCapture(0)
    dd1 = 0
    previous_count = 0
    stt=''
    ddd = 0

    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break
        # Perform object detection with a confidence threshold of 0.1
        results = model(frame, conf=0.3)

        if results and results[0].boxes:
            # Get the class names
            class_labels = results[0].names
            # Get detected class indices
            detected_classes = results[0].boxes.cls.cpu().numpy().astype(int)
            # Count "person" detections
            person_count = sum(1 for class_index in detected_classes if class_labels[class_index] == "person")
            # Notify if count changes
            dd1 += 1
            print(dd1)

            if dd1 == 30:
                ddd = person_count
            if dd1 > 30:
                if ddd < previous_count:
                    #annotated_frame = results[0].plot()
                    #cv2.imwrite("alert.jpg", annotated_frame)
                    #sendmail()
                    #sendmsg("9087259509", 'Person count increased!'+str(person_count))
                    print("Person count increased!")
                    stt = "increased"
                elif ddd > previous_count:
                    print("Person count decreased!")
                    stt = "decreased"
                    #annotated_frame = results[0].plot()
                    #cv2.imwrite("alert.jpg", annotated_frame)
                    #sendmail()
                    #sendmsg("9087259509", 'Person count increased!' + str(person_count))
            previous_count = person_count
            # Visualize the results
            annotated_frame = results[0].plot()
            # Display person count on the image
            text = f"Person Count: {person_count}"+stt  + "SetCount"+str(ddd)
            cv2.putText(annotated_frame, text, (20, 40), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2,
cv2.LINE_AA)
        else:
            annotated_frame = frame  # Display the original frame if no detections are made

        # Show output
        cv2.imshow("YOLOv8 Head Count", annotated_frame)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
def sendmsg(targetno,message):
    import requests
    requests.post(

"http://sms.creativepoint.in/api/push.json?apikey=6555c521622c1&route=transsms&sender=FSSMSS&mobileno=" +
targetno + "&text=Dear customer your msg is " + message + "  Sent By FSMSG FSSMSS")
def sendmail():
    import smtplib
    from email.mime.multipart import MIMEMultipart
    from email.mime.text import MIMEText
    from email.mime.base import MIMEBase
    from email import encoders
    fromaddr = "projectmailm@gmail.com"
    toaddr =  "sangeeth5535@gmail.com"
    # instance of MIMEMultipart
    msg = MIMEMultipart()

    # storing the senders email address
    msg['From'] = fromaddr

    # storing the receivers email address
    msg['To'] = toaddr
    # storing the subject
    msg['Subject'] = "Alert"
    # string to store the body of the mail
    body = "Crowd  Detection"
    # attach the body with the msg instance
    msg.attach(MIMEText(body, 'plain'))
    # open the file to be sent
    filename = "alert.jpg"
    attachment = open("alert.jpg", "rb")

    # instance of MIMEBase and named as p
    p = MIMEBase('application', 'octet-stream')
    # To change the payload into encoded form
    p.set_payload((attachment).read())
    # encode into base64
    encoders.encode_base64(p)
    p.add_header('Content-Disposition', "attachment; filename= %s" % filename)

    # attach the instance 'p' to instance 'msg'
    msg.attach(p)
    # creates SMTP session
    s = smtplib.SMTP('smtp.gmail.com', 587)
    # start TLS for security
    s.starttls()
    # Authentication
    s.login(fromaddr, "qmgn xecl bkqv musr")
    # Converts the Multipart msg into a string
    text = msg.as_string()
    # sending the mail
    s.sendmail(fromaddr, toaddr, text)

    # terminating the session
    s.quit()
def main_account_screen():
    global main_screen
    main_screen = Tk()
    width = 600
    height = 600
    screen_width = main_screen.winfo_screenwidth()
    screen_height = main_screen.winfo_screenheight()
    x = (screen_width / 2) - (width / 2)
    y = (screen_height / 2) - (height / 2)
    main_screen.geometry("%dx%d+%d+%d" % (width, height, x, y))
    main_screen.resizable(0, 0)
    # main_screen.geometry("300x250")
    main_screen.configure()
    main_screen.title("   Detection ")
    Label(text="   Detection", width="300", height="5", font=("Calibri", 16)).pack()
    Label(text="").pack()
    Button(text="Image", font=(
        'Verdana', 15), height="2", width="30", command=image1).pack(side=TOP)
    Label(text="").pack()
    Button(text="Camera", font=(
        'Verdana', 15), height="2", width="30", command=Camera1).pack(side=TOP)
    main_screen.mainloop()
main_account_screen()
personcount.py:
import numpy as np
from tkinter import *
import os
from tkinter import filedialog
```

```
import cv2
import time, sys
from matplotlib import pyplot as plt
from tkinter import messagebox
def endprogram():
    print("\nProgram terminated!")
    sys.exit()
def testing():
    global testing_screen
    testing_screen = Toplevel(main_screen)
    testing_screen.title("Testing")
    # login_screen.geometry("400x300")
    testing_screen.geometry("600x450+650+150")
    testing_screen.minsize(120, 1)
    testing_screen.maxsize(1604, 881)
    testing_screen.resizable(1, 1)
    testing_screen.configure(bg='cyan')
    # login_screen.title("New Toplevel")

    Label(testing_screen, text='''Upload Image''', disabledforeground="#a3a3a3",
          foreground="#000000", width="300", height="2", bg='cyan', font=("Calibri", 16)).pack()
    Label(testing_screen, text="").pack()
    Label(testing_screen, text="").pack()
    Label(testing_screen, text="").pack()
    Button(testing_screen, text='''Upload Image''', font=(
        'Verdana', 15), height="2", width="30", bg='cyan', command=imgtest).pack()


global affect
def imgtest():
    from ultralytics import YOLO
    import cv2
    import_file_path = filedialog.askopenfilename()
    # image = cv2.imread(import_file_path)
    image = cv2.imread(import_file_path)
    model = YOLO('runs/detect/workhelmet/weights/best.pt')
    # Load the image
    # image = cv2.imread('path/to/your/image.jpg')
    # Perform object detection
    results = model(image, conf=0.2)
    # Get the predicted class labels and their confidence scores
    class_labels = results[0].names
    confidences = results[0].boxes.conf
    if len(confidences) > 0:
        max_confidence_index = confidences.argmax().item()  # Convert tensor to integer
        predicted_class = class_labels[max_confidence_index]
    else:
        predicted_class = "No Detections"
    # Print the predicted class and its confidence score
    print(f"Predicted Class: {predicted_class}")
    print(f"Confidence Score: {confidences[max_confidence_index]}")
    # Optionally, visualize the results
    annotated_frame = results[0].plot()
    # Display the annotated frame
    cv2.imshow("YOLOv8 Inference", annotated_frame)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
def image1():
    import cv2
    from tkinter import filedialog
    from ultralytics import YOLO  # Ensure you have the correct YOLO library installed
    # Open a file dialog to select an image file
    import_file_path = filedialog.askopenfilename()
    # Load the selected image
    image = cv2.imread(import_file_path)
    # Load the YOLO model with pre-trained weights
    model = YOLO('runs/detect/personcount/weights/best.pt')
    # Perform object detection with a confidence threshold of 0.1
    results = model(image, conf=0.1)
    # Extract predictions
    if results and results[0].boxes:
        # Get the class names
        class_labels = results[0].names
        # Get the confidence scores and detected class indices
        confidences = results[0].boxes.conf.cpu().numpy()  # Convert tensor to numpy array
        detected_classes = results[0].boxes.cls.cpu().numpy().astype(int)  # Get class indices
        # Count total detections
        total_detections = len(detected_classes)
        # Count "person" detections
        person_count = sum(1 for class_index in detected_classes if class_labels[class_index] == "person")
        # Find the least confident detection
        min_confidence_index = confidences.argmin()
        predicted_class = class_labels[detected_classes[min_confidence_index]]
        confidence_score = confidences[min_confidence_index]
        # Print results
        print(f"Total Detections: {total_detections}")
        print(f"Person Count: {person_count}")
        print(f"Least Confident Prediction: {predicted_class}")
        print(f"Confidence Score: {confidence_score:.2f}")
        # Visualize the results
        annotated_frame = results[0].plot()
        # Display person count on the image
        text = f"Person Count: {person_count}"
        cv2.putText(annotated_frame, text, (20, 40), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2, cv2.LINE_AA)


    else:
        print("No detections made")
        annotated_frame = image  # Display the original image if no detections are made
    # Display the annotated frame
    cv2.imshow("YOLOv8 Inference", annotated_frame)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
def Camera1():
    import cv2
    from ultralytics import YOLO  # Ensure you have the correct YOLO library installed
    import pyttsx3
    # Initialize the engine
    engine = pyttsx3.init()
    # Configure voice and speed
```

```
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[1].id)  # Choose female voice
engine.setProperty('rate', 150)  # Set speed
engine.setProperty('volume', 0.9)  # Set volume
# Load the YOLO model with pre-trained weights
model = YOLO('runs/detect/personcount/weights/best.pt')

# Open a video file or webcam (0 for webcam, or provide a file path)
# video_path = "video.mp4"  # Change this to your video file path or use 0 for webcam
cap = cv2.VideoCapture(0)
# Check if the video file is opened
if not cap.isOpened():
    print("Error: Cannot open video file or webcam")
    exit()
# Process each frame
flag = 0
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break  # Stop when video ends
    # Perform object detection
    results = model(frame, conf=0.4)
    # Extract predictions
    if results and results[0].boxes:
        class_labels = results[0].names
        detected_classes = results[0].boxes.cls.cpu().numpy().astype(int)  # Convert tensor to int array
        # Count "person" detections
        person_count = sum(1 for class_index in detected_classes if class_labels[class_index] == "person")
    else:
        person_count = 0  # No detections
    # Visualize the results
    annotated_frame = results[0].plot()
    # Display person count on the video frame
    text = f"Person Count: {person_count}"
    if person_count > 5:
        flag += 1
        if flag == 20:
            flag = 0
            engine.say("Over Crowd")
            # Wait for completion
            engine.runAndWait()

            cv2.imwrite("alert.jpg", annotated_frame)
            sendmail()
            sendmsg("8610283590", "Over Crowd:")

    cv2.putText(annotated_frame, text, (20, 40), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2, cv2.LINE_AA)

    # Show the annotated frame
    cv2.imshow("YOLO11 Video Inference", annotated_frame)

    # Press 'q' to exit the video loop
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
    # Release resources
    cap.release()
    cv2.destroyAllWindows()
def sendmsg(targetno,message):
    import requests
    requests.post(
"http://sms.creativepoint.in/api/push.json?apikey=6555c521622c1&route=transsms&sender=FSSMSS&mobileno=" +
targetno + "&text=Dear customer your msg is " + message + "  Sent By FSMSG FSSMSS")
def sendmail():
    import smtplib
    from email.mime.multipart import MIMEMultipart
    from email.mime.text import MIMEText
    from email.mime.base import MIMEBase
    from email import encoders
    fromaddr = "projectmailm@gmail.com"
    toaddr =  "harishgiri8036@gmail.com"

    # instance of MIMEMultipart
    msg = MIMEMultipart()

    # storing the senders email address
    msg['From'] = fromaddr
    # storing the receivers email address
    msg['To'] = toaddr

    # storing the subject
    msg['Subject'] = "Alert"
    # string to store the body of the mail
    body = "Bus Crowd  Detection"

    # attach the body with the msg instance
    msg.attach(MIMEText(body, 'plain'))

    # open the file to be sent
    filename = "alert.jpg"
    attachment = open("alert.jpg", "rb")

    # instance of MIMEBase and named as p
    p = MIMEBase('application', 'octet-stream')

    # To change the payload into encoded form
    p.set_payload((attachment).read())

    # encode into base64
    encoders.encode_base64(p)

    p.add_header('Content-Disposition', "attachment; filename= %s" % filename)
    # attach the instance 'p' to instance 'msg'
    msg.attach(p)
    # creates SMTP session
    s = smtplib.SMTP('smtp.gmail.com', 587)

    # start TLS for security
    s.starttls()
    # Authentication
    s.login(fromaddr, "qmgn xecl bkqv musr")

    # Converts the Multipart msg into a string
    text = msg.as_string()
    # sending the mail
    s.sendmail(fromaddr, toaddr, text)

    # terminating the session
    s.quit()

def main_account_screen():
    global main_screen
    main_screen = Tk()
    width = 600
    height = 600
    screen_width = main_screen.winfo_screenwidth()
    screen_height = main_screen.winfo_screenheight()
    x = (screen_width / 2) - (width / 2)
    y = (screen_height / 2) - (height / 2)
    main_screen.geometry("%dx%d+%d+%d" % (width, height, x, y))
    main_screen.resizable(0, 0)
    # main_screen.geometry("300x250")
    main_screen.configure()
    main_screen.title("   Detection ")
    Label(text="   Detection", width="300", height="5", font=("Calibri", 16)).pack()
    Label(text="").pack()
    Button(text="Image", font=(
        'Verdana', 15), height="2", width="30", command=image1).pack(side=TOP)
    Label(text="").pack()
    Button(text="Camera", font=(
        'Verdana', 15), height="2", width="30", command=Camera1).pack(side=TOP)

    main_screen.mainloop()

main_account_screen()
```