import cv2

import os

from flask import Flask, request, render\_template

from datetime import date, datetime

import numpy as np

from sklearn.neighbors import KNeighborsClassifier

import pandas as pd

import joblib

import tempfile

import base64

app = Flask(\_\_name\_\_)

face\_detector = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_frontalface\_default.xml')

def datetoday():

    return date.today().strftime("%m\_%d\_%y")

def datetoday2():

    return date.today().strftime("%d-%B-%Y")

def totalreg():

    return len(os.listdir('static/faces'))

def extract\_faces(img):

    gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

    face\_points = face\_detector.detectMultiScale(gray, 1.3, 5)

    return face\_points

def identify\_face(facearray):

    model = joblib.load('static/face\_recognition\_model.pkl')

    return model.predict(facearray)

def train\_model():

    faces = []

    labels = []

    userlist = os.listdir('static/faces')

    for user in userlist:

        for imgname in os.listdir(f'static/faces/{user}'):

            img = cv2.imread(f'static/faces/{user}/{imgname}')

            resized\_face = cv2.resize(img, (50, 50))

            faces.append(resized\_face.ravel())

            labels.append(user)

    faces = np.array(faces)

    knn = KNeighborsClassifier(n\_neighbors=5)

    knn.fit(faces, labels)

    joblib.dump(knn, 'static/face\_recognition\_model.pkl')

def extract\_attendance():

    attendance\_file = f'Attendance/Attendance-{datetoday()}.csv'

    if not os.path.exists(attendance\_file):

        df = pd.DataFrame(columns=['Name', 'Roll', 'Time'])

        df.to\_csv(attendance\_file, index=False)

    else:

        df = pd.read\_csv(attendance\_file)

    names = df['Name']

    rolls = df['Roll']

    times = df['Time']

    l = len(df)

    return names, rolls, times, l

def add\_attendance(name):

    username = name.split('\_')[1]

    userid = name.split('\_')[0]

    current\_time = datetime.now().strftime("%H:%M:%S")

    attendance\_file = f'Attendance/Attendance-{datetoday()}.csv'

    if not os.path.isfile(attendance\_file):

        with open(attendance\_file, 'w') as f:

            f.write('Name,Roll,Time\n')

    with open(attendance\_file, 'a') as f:

        f.write(f'{username},{userid},{current\_time}\n')

def is\_display\_available():

    try:

        cv2.imshow('Test', np.zeros((1, 1), np.uint8))

        cv2.waitKey(1)

        cv2.destroyAllWindows()

        return True

    except cv2.error:

        return False

@app.route('/')

def home():

    names, rolls, times, l = extract\_attendance()

    mess = None

    if 'face\_recognition\_model.pkl' not in os.listdir('static'):

        mess = 'There is no trained model in the static folder. Please add a new face to continue.'

    return render\_template('home.html', names=names, rolls=rolls, times=times, l=l, totalreg=totalreg(), datetoday2=datetoday2(), mess=mess)

@app.route('/start', methods=['GET'])

def start():

    if 'face\_recognition\_model.pkl' not in os.listdir('static'):

        return render\_template('home.html', totalreg=totalreg(), datetoday2=datetoday2(), mess='There is no trained model in the static folder. Please add a new face to continue.')

    cap = cv2.VideoCapture(0)

    ret = True

    while ret:

        ret, frame = cap.read()

        if extract\_faces(frame) != ():

            (x, y, w, h) = extract\_faces(frame)[0]

            cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 20), 2)

            face = cv2.resize(frame[y:y+h, x:x+w], (50, 50))

            identified\_person = identify\_face(face.reshape(1, -1))[0]

            add\_attendance(identified\_person)

            if is\_display\_available():

                cv2.putText(frame, f'{identified\_person}', (30, 30), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255, 0, 20), 2, cv2.LINE\_AA)

                cv2.imshow('Attendance', frame)

                if cv2.waitKey(1) == 27:

                    break

            break

    cap.release()

    if is\_display\_available():

        cv2.destroyAllWindows()

    names, rolls, times, l = extract\_attendance()

    return render\_template('home.html', names=names, rolls=rolls, times=times, l=l, totalreg=totalreg(), datetoday2=datetoday2())

@app.route('/take\_attendance', methods=['POST'])

def take\_attendance():

    if 'face\_recognition\_model.pkl' not in os.listdir('static'):

        return render\_template('home.html', totalreg=totalreg(), datetoday2=datetoday2(), mess='There is no trained model in the static folder. Please add a new face to continue.')

    cap = cv2.VideoCapture(0)

    ret, frame = cap.read()

    identified\_person = None

    with tempfile.NamedTemporaryFile(suffix='.png', delete=False) as temp\_img:

        temp\_img\_path = temp\_img.name

        while ret:

            ret, frame = cap.read()

            faces = extract\_faces(frame)

            for (x, y, w, h) in faces:

                cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 20), 2)

                face = cv2.resize(frame[y:y+h, x:x+w], (50, 50))

                identified\_person = identify\_face(face.reshape(1, -1))[0]

                add\_attendance(identified\_person)

                cv2.putText(frame, f'{identified\_person}', (30, 30), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255, 0, 20), 2, cv2.LINE\_AA)

                cv2.imwrite(temp\_img\_path, frame)

                break

            if cv2.waitKey(1) == 27 or identified\_person is not None:

                break

        cap.release()

        cv2.destroyAllWindows()

        with open(temp\_img\_path, "rb") as image\_file:

            encoded\_image = base64.b64encode(image\_file.read()).decode('utf-8')

        names, rolls, times, l = extract\_attendance()

        return render\_template('home.html', names=names, rolls=rolls, times=times, l=l, totalreg=totalreg(), datetoday2=datetoday2(), encoded\_image=encoded\_image)

@app.route('/add', methods=['GET', 'POST'])

def add():

    if request.method == 'POST':

        newusername = request.form['newuserid']

        newuserid = request.form['newusername']

        userimagefolder = 'static/faces/' + newusername + '\_' + str(newuserid)

        if not os.path.isdir(userimagefolder):

            os.makedirs(userimagefolder)

        cap = cv2.VideoCapture(0)

        i, j = 0, 0

        while 1:

            \_, frame = cap.read()

            faces = extract\_faces(frame)

            for (x, y, w, h) in faces:

                cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 20), 2)

                if j % 10 == 0:

                    name = newusername + '\_' + str(i) + '.jpg'

                    cv2.imwrite(userimagefolder + '/' + name, frame[y:y+h, x:x+w])

                    i += 1

                j += 1

                if i == 50 or j == 500:

                    break

            if i == 50 or j == 500:

                break

            if cv2.waitKey(1) == 27:

                break

        cap.release()

        cv2.destroyAllWindows()

        print('Training Model')

        train\_model()

        names, rolls, times, l = extract\_attendance()

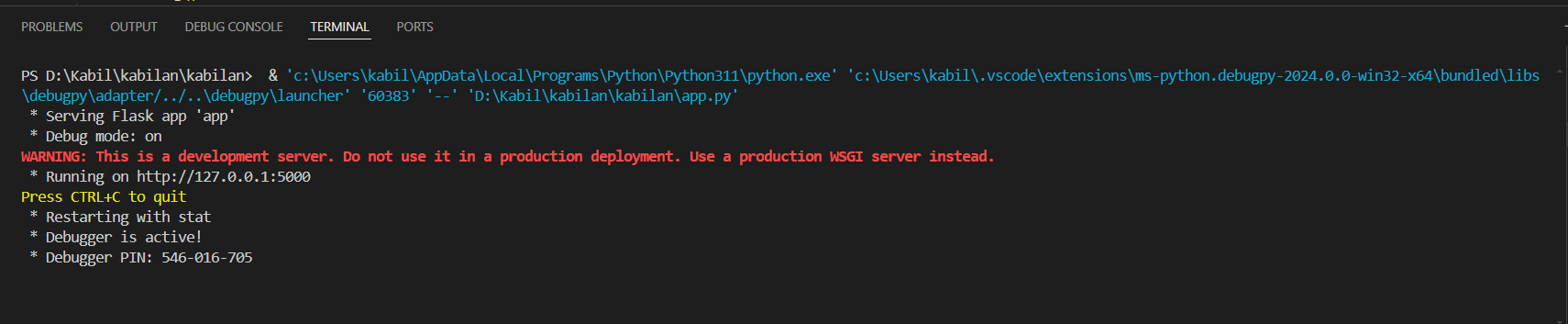
        return render\_template('home.html', names=names, rolls=rolls, times=times, l=l, totalreg=totalreg(), datetoday2=datetoday2())

    return render\_template('home.html', totalreg=totalreg(), datetoday2=datetoday2())

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

**OUTPUT :**

****

