```
import base64
import os
import random
import datetime
from weakref import ref
import FaceRecognition
import IrisRecognition
import numpy as np
import pandas as pd
import systemcheck
from multiprocessing import shared_memory
from flask import Flask, render_template, url_for, redirect, jsonify,
Response, abort, session, request, send_file
from werkzeug.utils import secure_filename
import sqlite3
import shutil
from mailSend import send_mail



conn = sqlite3.connect('data.db')
print ("Opened database successfully")


conn.execute('''CREATE TABLE IF NOT EXISTS USERS
         (ID INTEGER PRIMARY KEY AUTOINCREMENT,
         USERNAME           TEXT UNIQUE,
         PASSWORD           TEXT,
         NAME               TEXT,
         EMAIL              TEXT,
         CONTACT            TEXT,
         TYPE               TEXT);''')

conn.execute('''CREATE TABLE IF NOT EXISTS DETAILS
         (ID INTEGER PRIMARY KEY AUTOINCREMENT,
         NAME       TEXT,
         AGE        TEXT,
         GENDER     TEXT,
         AADHAR     TEXT,
         M_DATE     TEXT,
```

```
        PARENT     TEXT,
        P_CONTACT  TEXT,
        P_ADDRESS  TEXT,
        P_REMARK   TEXT,
        F_DATE     TEXT,
        FOUNDER    TEXT,
        F_CONTACT  TEXT,
        F_ADDRESS  TEXT,
        F_REMARK   TEXT,
        STATUS     TEXT);''')


conn.close()

shape = (480, 640, 3)
app = Flask(__name__)

UPLOAD_FOLDER = os.path.join(os.path.dirname(os.path.realpath(__file__)),
'faces/')
IRIS_FOLDER = os.path.join(os.path.dirname(os.path.realpath(__file__)),
'iris/')
TEMP_FOLDER = os.path.join(os.path.dirname(os.path.realpath(__file__)),
'temp/')

app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
app.config['IRIS_FOLDER'] = IRIS_FOLDER
app.config['TEMP_FOLDER'] = TEMP_FOLDER


outputFrame = None
number = random.randint(1000000, 9999999)


def html_return(msg, redirect_to = "/", delay = 5):
    return f"""
                <html>
                    <head>
                        <title>Missing Child </title>
                        <meta http-equiv="refresh"
content="{delay};URL='{redirect_to}'" />
```

```
                    </head>
                    <body>
                        <h2> {msg}</h2>
                        <p>This page will refresh automatically.</p>
                    </body>
                </html>

                """


@app.route('/', methods=['get', 'post'])
def login_page():
    if request.method == 'POST':
        username, password = request.form['username'],
request.form['password']
        if username == "niltech" and password == "Niltech@12345":
            session['user'] = username+"_Admin"
            print("Admin Login 1")
            return render_template('index.html', user=(session['user']))
        else:
            try:
                conn = sqlite3.connect('data.db')
                print ("Opened database successfully 1")
                cursor = conn.execute(f"SELECT USERNAME, PASSWORD, TYPE,
NAME from USERS")
                for row in cursor:
                    if  row[0] == username  and row[1] == password:
                        if len(row[2]) > 1:
                            session['user'] = row[3]+"_Admin"
                            print("Police Login")
                        else:
                            session['user'] = row[3]
                            print("User Login")
                        conn.close()
                        return render_template('index.html',
user=(session['user']))
                return render_template('login-page.html')
            except Exception as e:
                print("DB Error 1: ", e)
    elif 'user' in session.keys():
```

```
        if "_Admin" in session['user']:
            print("Police Login 1")
        else:
            print("User Login 1")
        return render_template('index.html', user=(session['user']))
    else:
        return render_template('login-page.html')


@app.route('/logout/')
def logout():
    session.clear()
    return redirect(url_for('login_page'))


@app.route('/add_missing/', methods=['get', 'post'])
def add_missing():
    if 'user' in session.keys():
        if request.method == 'POST':

            if 'file' not in request.files:
                return redirect(request.url)
            files = request.files.getlist('file')
            print("Files:",files)
            if files[0].filename == '':
                return redirect(request.url)


            if 'irisfiles' in request.files:
                print("Got Iris Images")
            iris_files = request.files.getlist('irisfiles')
            print("Iris Files:",iris_files)

            name = request.form['name']

            #Save Face Files
            if files:
                for i in range(10):
                    try:
```

```
                        os.mkdir(os.path.join(app.config['UPLOAD_FOLDER'],
name))
                        print("Folder Created (FACE):", name)
                        break
                    except Exception as e:
                            print("Folder Already Exists:", e)
                            name = request.form['name']+str(i)

                for file in files:
                    filename = secure_filename(file.filename)
                    file.save(os.path.join(app.config['UPLOAD_FOLDER'],
name, filename))

            #Save Iris Files
            if iris_files:
                for i in range(10):
                    try:
                        os.mkdir(os.path.join(app.config['IRIS_FOLDER'],
name))
                        print("Folder Created (IRIS):", name)
                        break
                    except Exception as e:
                            print("Folder Already Exists:", e)
                            name = request.form['name']+str(i)

                for file in iris_files:
                    try:
                        filename = secure_filename(file.filename)
                        file.save(os.path.join(app.config['IRIS_FOLDER'],
name, filename))
                    except:
                        print("IRIS file..")




            age = request.form['age']
            gender  = request.form['gender']
            aadhar  = request.form['aadhar']
```

```
            mdate  = request.form['mdate']
            parent  = request.form['parent']
            pcontact  = request.form['pcontact']
            paddress  = request.form['paddress']
            premark  = request.form['premark']

            fdate  = request.form['fdate']
            fname  = request.form['fname']
            fcontact  = request.form['fcontact']
            faddress  = request.form['faddress']
            fremark  = request.form['fremark']

            if files:

FaceRecognition.add_face(os.path.join(app.config['UPLOAD_FOLDER'], name),
name=name)
                print("Face Training Completed")

            if iris_files:

IrisRecognition.add_iris(os.path.join(app.config['IRIS_FOLDER'], name),
name=name)
                print("Iris Training Completed")

            print("Updating Database")
            conn = sqlite3.connect('data.db')
            conn.execute(f"INSERT INTO DETAILS (NAME, AGE, GENDER, AADHAR,
M_DATE, PARENT, P_CONTACT, P_ADDRESS, P_REMARK, F_DATE, FOUNDER,
F_CONTACT, F_ADDRESS, F_REMARK, STATUS ) VALUES \
            ('{name}','{age}', '{gender}', '{aadhar}', '{mdate}',
'{parent}', '{pcontact}', '{paddress}', '{premark}', '{fdate}', '{fname}',
'{fcontact}', '{faddress}', '{fremark}', 'Missing' )")
            conn.commit()
            conn.close()

            return redirect(request.url)
        return render_template('add_missing.html', user=(session['user']))
    else:
        return redirect(url_for('login_page'))
```

```
@app.route('/update_info/', methods=['get', 'post'])
def update_info():
    if 'user' in session.keys():
        if "_Admin" not in session['user']:
            return html_return("Only Admin / Police can Add or Update
Users")
        if request.method == 'POST':

            name = request.form['name']

            print("Updating Database")
            conn = sqlite3.connect('data.db')


            cursor = conn.execute(f"SELECT ID, AGE, GENDER, AADHAR,
STATUS, M_DATE, PARENT, P_CONTACT, P_ADDRESS, P_REMARK, F_DATE, FOUNDER,
F_CONTACT, F_ADDRESS, F_REMARK FROM DETAILS WHERE NAME='{name}' ")
            data_temp = []
            for row in cursor:
                data_temp = row

            if len(data_temp) > 0:
                print("User Found in Database")
                conn.execute(f"DELETE FROM DETAILS WHERE
ID='{data_temp[0]}'")
            else:
                print("User Not Found in Database")
                return html_return(f"{name} User Details not found in
Database. Check for Correct Name.")



            age = request.form['age']
            if len(age) < 1:
                age = data_temp[1]
            try:
                gender  = request.form['gender']
                if len(gender) < 1:
```

```
            gender = data_temp[2]
        except:
            gender = data_temp[2]

        aadhar  = request.form['aadhar']
        if len(aadhar) < 1:
            aadhar = data_temp[3]

        status = request.form['status']
        if len(status) < 1:
            status = data_temp[4]

        mdate  = request.form['mdate']
        if len(mdate) < 1:
            mdate = data_temp[5]

        parent  = request.form['parent']
        if len(parent) < 1:
            parent = data_temp[6]

        pcontact  = request.form['pcontact']
        if len(pcontact) < 1:
            pcontact = data_temp[7]

        paddress  = request.form['paddress']
        if len(paddress) < 1:
            paddress = data_temp[8]

        premark  = request.form['premark']
        if len(premark) < 1:
            premark = data_temp[9]

        fdate  = request.form['fdate']
        if len(fdate) < 1:
            fdate = data_temp[10]

        fname  = request.form['fname']
        if len(fname) < 1:
            fname = data_temp[11]
```

```
            fcontact  = request.form['fcontact']
            if len(fcontact) < 1:
                fcontact = data_temp[12]

            faddress  = request.form['faddress']
            if len(faddress) < 1:
                faddress = data_temp[13]

            fremark  = request.form['fremark']
            if len(fremark) < 1:
                fremark = data_temp[14]



            conn.execute(f"INSERT INTO DETAILS (NAME, AGE, GENDER, AADHAR,
M_DATE, PARENT, P_CONTACT, P_ADDRESS, P_REMARK, F_DATE, FOUNDER,
F_CONTACT, F_ADDRESS, F_REMARK, STATUS ) VALUES \
            ('{name}','{age}', '{gender}', '{aadhar}', '{mdate}',
'{parent}', '{pcontact}', '{paddress}', '{premark}', '{fdate}', '{fname}',
'{fcontact}', '{faddress}', '{fremark}', '{status}' )")
            conn.commit()
            conn.close()
            print("Updated User Details in Database")

            message = f"Following Missing(s) profile Updated:\n {name}"
            send_mail(message)

            return html_return(f"Updated {name} User Details in Database")
        return render_template('update_info.html', user=(session['user']))
    else:
        return redirect(url_for('login_page'))



@app.route('/add_user/', methods=['get', 'post'])
def add_admin():
    if "_Admin" not in session['user']:
            return html_return("Only Admin / Police can Add or Update
Users")
    if 'user' in session.keys():
```

```
        if "_Admin" not in session['user']:
            return html_return("Only Admin / Police can Add or Update
Users")
        if request.method == 'POST':
            print("Got User Enroll details")
            username = request.form['username']
            password = request.form['password']
            name = request.form['name']
            mail = request.form['mail']
            contact = request.form['phone']
            Type = request.form['Type']

            print("Updating Database", end = " ")
            try:
                conn = sqlite3.connect('data.db')
                conn.execute(f"INSERT INTO USERS (USERNAME, PASSWORD,
NAME, EMAIL, CONTACT, TYPE) VALUES ('{username}', '{password}', '{name}',
'{mail}', '{contact}', '{Type}' )")
                conn.commit()
                conn.close()
                print("| User Added Successfully")

                message = f"Following Missing(s) profile Added:\n {name}"
                send_mail(message)

            except Exception as e:
                print("Failed. ERROR:", e)
            return redirect(url_for('login_page'))
        return render_template('add_user.html', user=(session['user']))
    else:
        return redirect(url_for('login_page'))



@app.route('/update_admin/' , methods=['GET', 'POST'])
def update_admin():
    if 'user' in session.keys():
        if "_Admin" not in session['user']:
            return html_return("Only Admin / Police can Add or Update
Users")
```

```
        print("RM", request.method)
        if request.method == 'POST':
            print("Got Admin Update details")
            userid = request.form['username1']
            print("Got userid")
            password = request.form['password1']
            print("Got userid")
            if password == "DEL":
                if userid != "niltech":
                    try:
                        conn = sqlite3.connect('data.db')
                        conn.execute(f"DELETE from USERS where USERNAME =
'{userid}';")
                        conn.commit()
                        conn.close()

                        return html_return("Successfully Deleted Admin
User: "+str(userid), delay = 3)
                    except Exception as e:
                        return html_return("Deletion failed for Admin
User: "+str(userid)+". Reason: "+str(e))
                else:
                    return html_return("Cannot Delete Master Default Admin
User: "+str(userid))
            else:

                try:
                    conn = sqlite3.connect('data.db')
                    conn.execute(f"UPDATE USERS set PASSWORD =
'{password}' where USERNAME = '{userid}';")
                    conn.commit()
                    conn.close()
                    return html_return("Password Updated for Admin:
"+str(userid) +" if exists.")
                except Exception as e:
                        return html_return("Password Update failed for
Admin User: "+str(userid)+". Reason: "+str(e))
    else:
        return redirect(url_for('login_page'))
```

```
@app.route('/all_missing/')
def all_missing():
    if 'user' in session.keys():
        conn = sqlite3.connect('data.db')
        cursor = conn.execute("SELECT NAME, AGE, GENDER, M_DATE, STATUS
from DETAILS")
        users_list = []
        for row in cursor:
            users_list.append(row)
        conn.close()
        return render_template('all_missing.html', user=(session['user']),
users_list=users_list)
    else:
        return redirect(url_for('login_page'))



@app.route('/profile/<name>')
def profile(name):
    if 'user' in session.keys():
        conn = sqlite3.connect('data.db')
        cursor = conn.execute(f"SELECT NAME, AGE, GENDER, M_DATE, STATUS
from DETAILS where NAME = '{name}'")

        cursor = conn.execute(f"SELECT ID, AGE, GENDER, AADHAR, STATUS,
M_DATE, PARENT, P_CONTACT, P_ADDRESS, P_REMARK, F_DATE, FOUNDER,
F_CONTACT, F_ADDRESS, F_REMARK FROM DETAILS WHERE NAME='{name}' ")
        user_details = []
        for row in cursor:
            user_details = row
        conn.close()

        img_list = os.listdir(UPLOAD_FOLDER + '/' + name)
        try:
            with open(UPLOAD_FOLDER + '/' + name + '/' + img_list[0],
'rb') as (image):
                image = base64.b64encode(image.read()).decode('utf-8')
        except:
```

```
            with open("dummy-profile-pic.png", 'rb') as (image):
                image = base64.b64encode(image.read()).decode('utf-8')



        return render_template('profile.html', user=(session['user']),
user_details=user_details, image=image, name=name)
    else:
        return redirect(url_for('login_page'))



@app.route('/delete_user/<name>')
def delete_user(name):
    if 'user' in session.keys():
        suc = 1

        try:
            conn = sqlite3.connect('data.db')
            conn.execute(f"DELETE from DETAILS where NAME = '{name}';")
            conn.commit()
            conn.close()

            message = f"Following Missing(s) profile Deleted:\n {name}"
            send_mail(message)

        except Exception as e:
            print("Unable to delete User from Database. Reason:", e)
            suc = 0

        try:
            FaceRecognition.remove_face(name)
        except Exception as e:
            print("Unable to delete User from Face Model. Reason:", e)
            suc = 0


        try:
            IrisRecognition.remove_iris(name)
        except Exception as e:
```

```
            print("Unable to delete User from Iris Model. Reason:", e)
            suc = 0

        try:
            shutil.rmtree(os.path.join(app.config['UPLOAD_FOLDER'], name))
        except Exception as e:
            print("Unable to delete Face Images. Reason:", e)
            suc = 0


        try:
            shutil.rmtree(os.path.join(app.config['IRIS_FOLDER'], name))
        except Exception as e:
            print("Unable to delete Iris Images. Reason:", e)
            suc = 0

        if suc == 0:
            return html_return("Deletion Completed with some Issues")

        return html_return("User Deletion Completed")

    else:
        return redirect(url_for('login_page'))


@app.route('/searchname/', methods=['get', 'post'])
def searchname():
    if 'user' in session.keys():
        if request.method == 'POST':
            name = request.form['name']
            name = name.strip()
            if len(name) < 1:
                return html_return("Kindly Enter Some Name to Search")

            users_list = []
            conn = sqlite3.connect('data.db')

            cursor = conn.execute(f"SELECT NAME, AGE, GENDER, M_DATE,
STATUS from DETAILS WHERE NAME LIKE '%{name}%'")
            for row in cursor:
```

```
                users_list.append(row)

            cursor = conn.execute(f"SELECT NAME, AGE, GENDER, M_DATE,
STATUS from DETAILS WHERE PARENT LIKE '%{name}%'")
            for row in cursor:
                users_list.append(row)

            cursor = conn.execute(f"SELECT NAME, AGE, GENDER, M_DATE,
STATUS from DETAILS WHERE FOUNDER LIKE '%{name}%'")
            for row in cursor:
                users_list.append(row)


            conn.close()
            if len(users_list) > 0:

                message = "Following Missing(s) were found while Name
Search:\n"
                for nm in users_list:
                    message += f"{nm} \n"
                send_mail(message)

                return render_template('all_missing.html',
user=(session['user']), users_list=users_list)
            else:
                return html_return("Sorry... No Matching Name Found.",
redirect_to="/add_missing/")

        return render_template('index.html', user=(session['user']))
    else:
        return redirect(url_for('login_page'))


@app.route('/searchaddress/', methods=['get', 'post'])
def searchaddress():
    if 'user' in session.keys():
        if request.method == 'POST':
            address = request.form['address']
            address = address.strip()
            if len(address) < 1:
```

```
                return html_return("Kindly Enter Some Address to Search")

            users_list = []
            conn = sqlite3.connect('data.db')
            cursor = conn.execute(f"SELECT NAME, AGE, GENDER, M_DATE,
STATUS from DETAILS WHERE P_ADDRESS LIKE '%{address}%'")
            for row in cursor:
                users_list.append(row)

            cursor = conn.execute(f"SELECT NAME, AGE, GENDER, M_DATE,
STATUS from DETAILS WHERE F_ADDRESS LIKE '%{address}%'")
            for row in cursor:
                users_list.append(row)

            conn.close()
            if len(users_list) > 0:

                message = "Following Missing(s) were found while Address
Search:\n"
                for nm in users_list:
                    message += f"{nm} \n"
                send_mail(message)

                return render_template('all_missing.html',
user=(session['user']), users_list=users_list)
            else:
                return html_return("Sorry... No Matching Address Found.",
redirect_to="/add_missing/")

        return render_template('index.html', user=(session['user']))
    else:
        return redirect(url_for('login_page'))


@app.route('/searchface/', methods=['get', 'post'])
def searchface():
    if 'user' in session.keys():
        if request.method == 'POST':
            if 'file' not in request.files:
                return redirect(request.url)
```

```
            shutil.rmtree(os.path.join(app.config['TEMP_FOLDER']))
            try:
                os.mkdir("temp")
            except:
                pass

            print("Searching Faces")
            files = request.files.getlist('file')
            print("Files:",files)
            if files[0].filename == '':
                return redirect(request.url)

            #Save Face Files
            foldername = str(datetime.datetime.now())[:-7].replace(" ",
"").replace("-","").replace(":","")
            if files:

                os.mkdir(os.path.join(app.config['TEMP_FOLDER'],
foldername))
                print("Folder Created:", foldername)

                for file in files:
                    filename = secure_filename(file.filename)
                    file.save(os.path.join(app.config['TEMP_FOLDER'],
foldername, filename))


            recognised =
FaceRecognition.checkface_folder(os.path.join(app.config['TEMP_FOLDER'],
foldername))
            print("Face Search Completed")
            print(recognised)
            users_list = []

            conn = sqlite3.connect('data.db')
            for name in set(recognised):
                cursor = conn.execute(f"SELECT NAME, AGE, GENDER, M_DATE,
STATUS from DETAILS WHERE NAME LIKE '%{name}%' ")
                for row in cursor:
```

```
                    users_list.append(row)
            conn.close()
            if len(users_list) > 0:

                message = "Following Missing(s) were found while Face
Search:\n"
                for nm in users_list:
                    message += f"{nm} \n"
                send_mail(message)

                return render_template('all_missing.html',
user=(session['user']), users_list=users_list)
            else:
                return html_return("Sorry... No Matching Face Found.",
redirect_to="/add_missing/")

        return render_template('index.html', user=(session['user']))
    else:
        return redirect(url_for('login_page'))


@app.route('/searchiris/', methods=['get', 'post'])
def searchiris():
    if 'user' in session.keys():
        if request.method == 'POST':
            if 'file' not in request.files:
                return redirect(request.url)

            shutil.rmtree(os.path.join(app.config['TEMP_FOLDER']))
            try:
                os.mkdir("temp")
            except:
                pass

            print("Searching Iris")
            files = request.files.getlist('file')
            print("Files:",files)
            if files[0].filename == '':
                return redirect(request.url)
```

```
            #Save Face Files
            foldername = str(datetime.datetime.now())[:-7].replace(" ",
"").replace("-","").replace(":","")
            if files:

                os.mkdir(os.path.join(app.config['TEMP_FOLDER'],
foldername))
                print("Folder Created:", foldername)

                for file in files:
                    filename = secure_filename(file.filename)
                    file.save(os.path.join(app.config['TEMP_FOLDER'],
foldername, filename))


            recognised =
IrisRecognition.checkiris_folder(os.path.join(app.config['TEMP_FOLDER'],
foldername))
            print("Iris Search Completed")
            print(recognised)
            users_list = []

            conn = sqlite3.connect('data.db')
            for name in set(recognised):
                cursor = conn.execute(f"SELECT NAME, AGE, GENDER, M_DATE,
STATUS from DETAILS WHERE NAME LIKE '%{name}%' ")
                for row in cursor:
                    users_list.append(row)
            conn.close()
            if len(users_list) > 0:

                message = "Following Missing(s) were found while Iris
Search:\n"
                for nm in users_list:
                    message += f"{nm} \n"
                send_mail(message)

                return render_template('all_missing.html',
user=(session['user']), users_list=users_list)
            else:
```

```
                return html_return("Sorry... No Matching Iris Found.",
redirect_to="/add_missing/")

        return render_template('index.html', user=(session['user']))
    else:
        return redirect(url_for('login_page'))



@app.errorhandler(404)
def nice(_):
    return render_template('error_404.html')



app.secret_key = 'q12q3q4e5g5htrh@werwer15454'

if __name__ == '__main__':
    app.run(host='0.0.0.0', port= 5000, debug = True)#80)
# global outputFrame ## Warning: Unused global
```