from flask import Flask, request, render\_template, jsonify  
import cv2  
import face\_recognition  
import os  
import base64  
from datetime import datetime  
import numpy as np  
import re  
import shutil  
  
app = Flask(\_\_name\_\_)  
  
# Mật khẩu quản trị viên  
ADMIN\_PASSWORD = "N7@tnl"  
  
# Đường dẫn lưu trữ  
face\_path = "VFD01/library/Pic\_face\_ID"  
card\_path = "VFD01/library/Pic\_the\_sv"  
history\_path = "library/lichsuquet.csv"  
  
# Khởi tạo các danh sách dữ liệu  
face\_images = []  
card\_images = []  
class\_names = []  
mssvs = []# Kiểm tra và tạo thư mục nếu chưa tồn tại  
if not os.path.exists(face\_path):  
 os.makedirs(face\_path)  
 print(f"Thư mục '{face\_path}' đã được tạo.")  
  
  
# Tải hình ảnh khuôn mặt từ thư mục  
face\_list = [f for f in os.listdir(face\_path) if f.lower().endswith(('.png', '.jpg', '.jpeg'))]  
  
for cl in face\_list:  
 cur\_img = cv2.imread(f"{face\_path}/{cl}")  
 if cur\_img is not None:  
 parts = os.path.splitext(cl)[0].rsplit('\_', 1)  
 if len(parts) == 2:  
 name, mssv = parts  
 face\_images.append(cur\_img)  
 class\_names.append(name)  
 mssvs.append(mssv)  
 else:  
 print(f"Bỏ qua tệp không đúng định dạng: {cl}")  
  
# Hàm mã hóa hình ảnh khuôn mặt  
def encode\_images(images):  
 encode\_list = []  
 for img in images:  
 img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)  
 encodings = face\_recognition.face\_encodings(img)  
 if encodings:  
 encode\_list.append(encodings[0])  
 else:  
 print("Không tìm thấy khuôn mặt trong hình ảnh.")  
 return encode\_list  
  
# Mã hóa khuôn mặt từ cơ sở dữ liệu  
encode\_list\_known = encode\_images(face\_images)  
  
# Tạo file lịch sử nếu chưa tồn tại  
if not os.path.exists(history\_path):  
 os.makedirs(os.path.dirname(history\_path), exist\_ok=True)  
 with open(history\_path, "w") as f:  
 f.write("name,mssv,timestamp,method\n")  
  
# Hàm giải mã hình ảnh từ base64  
def decode\_base64\_image(data\_url):  
 try:  
 image\_data = re.sub('^data:image/.+;base64,', '', data\_url)  
 image\_data = base64.b64decode(image\_data)  
 nparr = np.frombuffer(image\_data, np.uint8)  
 img = cv2.imdecode(nparr, cv2.IMREAD\_COLOR)  
 return img  
 except Exception as e:  
 print(f"Error decoding base64 image: {e}")  
 return None  
  
# Route chính  
@app.route('/')  
def home():  
 return render\_template('index.html')  
  
# Route nhận diện khuôn mặt  
@app.route('/recognize\_faces', methods=['POST'])  
def recognize\_faces():  
 data = request.get\_json()  
 image\_data = data.get('image')  
 frame = decode\_base64\_image(image\_data)  
  
 if frame is None:  
 return jsonify({"status": "failed", "message": "Error decoding image."})  
  
 framS = cv2.resize(frame, (0, 0), None, fx=0.5, fy=0.5)  
 framS = cv2.cvtColor(framS, cv2.COLOR\_BGR2RGB)  
  
 face\_cur\_frame = face\_recognition.face\_locations(framS, model='hog')  
 encode\_cur\_frame = face\_recognition.face\_encodings(framS)  
  
 if len(encode\_cur\_frame) != 1:  
 return jsonify({"status": "failed", "message": "Multiple or no faces detected!"})  
  
 encode\_face = encode\_cur\_frame[0]  
 face\_loc = face\_cur\_frame[0]  
  
 matches = face\_recognition.compare\_faces(encode\_list\_known, encode\_face, tolerance=0.4)  
 face\_dis = face\_recognition.face\_distance(encode\_list\_known, encode\_face)  
  
 if True in matches:  
 match\_index = np.argmin(face\_dis)  
 recognized\_name = class\_names[match\_index].upper()  
 recognized\_mssv = mssvs[match\_index]  
  
 # Ghi lịch sử nhận diện  
 with open(history\_path, "a") as f:  
 now = datetime.now()  
 dtstring = now.strftime("%Y-%m-%d %H:%M:%S")  
 f.write(f"{recognized\_name},{recognized\_mssv},{dtstring},face\n")  
  
 return jsonify({"status": "success", "message": f"Recognized: {recognized\_name}, MSSV: {recognized\_mssv}"})  
 else:  
 return jsonify({"status": "failed", "message": "Face not recognized!"})  
  
# Chạy ứng dụng Flask  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(debug=True)

**Giải thích đoạn code:**

Đây là một ứng dụng nhận diện khuôn mặt sử dụng Flask (framework Python) và thư viện xử lý hình ảnh như OpenCV và face\_recognition. Dưới đây là giải thích từng phần của đoạn mã:

**1. Import các thư viện cần thiết**

from flask import Flask, request, render\_template, jsonify

import cv2

import face\_recognition

import os

import base64

from datetime import datetime

import numpy as np

import re

import shutil

* **Flask**: Tạo và quản lý ứng dụng web.
* **OpenCV (cv2)**: Xử lý hình ảnh.
* **face\_recognition**: Thư viện nhận diện khuôn mặt.
* **os**: Quản lý tệp và thư mục.
* **base64**: Xử lý ảnh được mã hóa dưới dạng Base64.
* **datetime**: Lấy ngày giờ hiện tại.
* **numpy (np)**: Xử lý mảng số học.
* **re**: Thư viện hỗ trợ biểu thức chính quy (regex).
* **shutil**: Quản lý tệp, thư mục nâng cao.

**2. Khởi tạo Flask app và các biến toàn cục**

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

# Mật khẩu quản trị viên

ADMIN\_PASSWORD = "N7@tnl"

# Đường dẫn lưu trữ

face\_path = "VFD01/library/Pic\_face\_ID"

card\_path = "VFD01/library/Pic\_the\_sv"

history\_path = "library/lichsuquet.csv"

* **app = Flask(\_\_name\_\_)**: Tạo ứng dụng Flask.
* **ADMIN\_PASSWORD**: Lưu mật khẩu quản trị viên.
* **face\_path, card\_path, history\_path**: Định nghĩa các đường dẫn lưu trữ:
  + **face\_path**: Thư mục chứa ảnh khuôn mặt.
  + **card\_path**: Thư mục chứa ảnh thẻ sinh viên.
  + **history\_path**: File CSV lưu lịch sử nhận diện.

**3. Tạo và kiểm tra thư mục**

if not os.path.exists(face\_path):

os.makedirs(face\_path)

print(f"Thư mục '{face\_path}' đã được tạo.")

* Nếu thư mục chứa ảnh khuôn mặt chưa tồn tại, chương trình sẽ tự tạo thư mục.

**4. Tải dữ liệu ảnh và thông tin từ thư mục**

face\_list = [f for f in os.listdir(face\_path) if f.lower().endswith(('.png', '.jpg', '.jpeg'))]

for cl in face\_list:

cur\_img = cv2.imread(f"{face\_path}/{cl}")

if cur\_img is not None:

parts = os.path.splitext(cl)[0].rsplit('\_', 1)

if len(parts) == 2:

name, mssv = parts

face\_images.append(cur\_img)

class\_names.append(name)

mssvs.append(mssv)

else:

print(f"Bỏ qua tệp không đúng định dạng: {cl}")

* **face\_list**: Lấy danh sách các tệp hình ảnh trong thư mục face\_path.
* **Định dạng tệp yêu cầu**: .png, .jpg, .jpeg.
* **cv2.imread**: Đọc ảnh từ thư mục.
* **Tách thông tin từ tên tệp**:
  + Tên tệp phải có định dạng Tên\_MSSV.jpg:
    - name: Tên sinh viên.
    - mssv: Mã số sinh viên.
  + Nếu không đúng định dạng, tệp bị bỏ qua.

**5. Mã hóa khuôn mặt**

def encode\_images(images):

encode\_list = []

for img in images:

img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

encodings = face\_recognition.face\_encodings(img)

if encodings:

encode\_list.append(encodings[0])

else:

print("Không tìm thấy khuôn mặt trong hình ảnh.")

return encode\_list

encode\_list\_known = encode\_images(face\_images)

* **Hàm encode\_images**:
  + Chuyển ảnh sang không gian màu RGB (cv2.cvtColor).
  + **face\_recognition.face\_encodings**: Mã hóa khuôn mặt thành vector đặc trưng.
  + Nếu không tìm thấy khuôn mặt trong ảnh, in thông báo.
* **encode\_list\_known**:
  + Danh sách mã hóa tất cả khuôn mặt đã lưu trong thư mục.

**6. Tạo file lịch sử**

if not os.path.exists(history\_path):

os.makedirs(os.path.dirname(history\_path), exist\_ok=True)

with open(history\_path, "w") as f:

f.write("name,mssv,timestamp,method\n")

* Nếu file lịch sử lichsuquet.csv chưa tồn tại:
  + Tạo file mới.
  + Thêm dòng tiêu đề: name, mssv, timestamp, method.

**7. Hàm giải mã hình ảnh Base64**

def decode\_base64\_image(data\_url):

try:

image\_data = re.sub('^data:image/.+;base64,', '', data\_url)

image\_data = base64.b64decode(image\_data)

nparr = np.frombuffer(image\_data, np.uint8)

img = cv2.imdecode(nparr, cv2.IMREAD\_COLOR)

return img

except Exception as e:

print(f"Error decoding base64 image: {e}")

return None

* **data\_url**: Chuỗi Base64 chứa hình ảnh.
* **re.sub**: Xóa phần đầu data:image/...;base64, khỏi chuỗi.
* **base64.b64decode**: Giải mã Base64 thành byte.
* **cv2.imdecode**: Chuyển byte thành ảnh OpenCV.

**8. Route chính (/)**

@app.route('/')

def home():

return render\_template('index.html')

* Trả về giao diện chính (index.html) khi truy cập root URL (/).

**9. Nhận diện khuôn mặt**

@app.route('/recognize\_faces', methods=['POST'])

def recognize\_faces():

data = request.get\_json()

image\_data = data.get('image')

frame = decode\_base64\_image(image\_data)

if frame is None:

return jsonify({"status": "failed", "message": "Error decoding image."})

framS = cv2.resize(frame, (0, 0), None, fx=0.5, fy=0.5)

framS = cv2.cvtColor(framS, cv2.COLOR\_BGR2RGB)

face\_cur\_frame = face\_recognition.face\_locations(framS, model='hog')

encode\_cur\_frame = face\_recognition.face\_encodings(framS)

if len(encode\_cur\_frame) != 1:

return jsonify({"status": "failed", "message": "Multiple or no faces detected!"})

encode\_face = encode\_cur\_frame[0]

matches = face\_recognition.compare\_faces(encode\_list\_known, encode\_face, tolerance=0.4)

face\_dis = face\_recognition.face\_distance(encode\_list\_known, encode\_face)

if True in matches:

match\_index = np.argmin(face\_dis)

recognized\_name = class\_names[match\_index].upper()

recognized\_mssv = mssvs[match\_index]

with open(history\_path, "a") as f:

now = datetime.now()

dtstring = now.strftime("%Y-%m-%d %H:%M:%S")

f.write(f"{recognized\_name},{recognized\_mssv},{dtstring},face\n")

return jsonify({"status": "success", "message": f"Recognized: {recognized\_name}, MSSV: {recognized\_mssv}"})

else:

return jsonify({"status": "failed", "message": "Face not recognized!"})

* **Giải thích logic:**
  + Lấy dữ liệu ảnh từ yêu cầu POST (Base64).
  + Giải mã ảnh và chuẩn hóa kích thước.
  + Tìm vị trí khuôn mặt (face\_locations) và mã hóa nó (face\_encodings).
  + So sánh mã hóa với danh sách khuôn mặt đã biết:
    - **compare\_faces**: So khớp khuôn mặt.
    - **face\_distance**: Tính khoảng cách (độ chính xác).
  + Nếu tìm thấy:
    - Lưu kết quả vào file lịch sử.
    - Trả về thông báo nhận diện thành công.
  + Nếu không:
    - Trả về thông báo thất bại.

**10. Chạy ứng dụng**

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

app.run(debug=True)

* Khởi chạy ứng dụng Flask ở chế độ debug (giúp dễ dàng phát hiện lỗi).