# Rewriting the project to use PostgreSQL for user registration and image upload

# Updated app.py with PostgreSQL support using psycopg2

app\_py\_code = '''from flask import Flask, render\_template, request, redirect, url\_for, flash

import face\_recognition

import cv2

import os

import psycopg2

import numpy as np

import pickle

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

app.secret\_key = 'secret123'

app.config['UPLOAD\_FOLDER'] = 'dataset/known\_faces'

# PostgreSQL connection

def connect\_db():

return psycopg2.connect(

dbname="face\_db",

user="postgres",

password="yourpassword",

host="localhost",

port="5432"

)

@app.route('/')

def home():

return render\_template('index.html')

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

def register():

if request.method == 'POST':

name = request.form['name']

image = request.files['image']

if image:

path = os.path.join(app.config['UPLOAD\_FOLDER'], image.filename)

image.save(path)

face\_img = face\_recognition.load\_image\_file(path)

encoding\_list = face\_recognition.face\_encodings(face\_img)

if not encoding\_list:

flash("No face found in the image!")

return redirect(url\_for('register'))

encoding = pickle.dumps(encoding\_list[0])

conn = connect\_db()

cur = conn.cursor()

cur.execute("INSERT INTO users (name, encoding) VALUES (%s, %s)", (name, encoding))

conn.commit()

cur.close()

conn.close()

flash("User registered successfully!")

return redirect(url\_for('home'))

return render\_template('register.html')

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

def recognize\_face():

conn = connect\_db()

cur = conn.cursor()

cur.execute("SELECT name, encoding FROM users")

rows = cur.fetchall()

known\_encodings = [pickle.loads(row[1].tobytes()) for row in rows]

names = [row[0] for row in rows]

cur.close()

conn.close()

video = cv2.VideoCapture(0)

result = "No face detected"

while True:

ret, frame = video.read()

rgb\_frame = frame[:, :, ::-1]

face\_locations = face\_recognition.face\_locations(rgb\_frame)

encodings = face\_recognition.face\_encodings(rgb\_frame, face\_locations)

for encoding in encodings:

matches = face\_recognition.compare\_faces(known\_encodings, encoding)

if True in matches:

match\_index = matches.index(True)

result = f"Face Recognized: {names[match\_index]}"

break

break

video.release()

return render\_template("result.html", result=result)

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

app.run(debug=True)

'''

# Write updated app.py

with open("/mnt/data/face\_recognition\_project/app.py", "w") as f:

f.write(app\_py\_code)

# Create register.html template

with open("/mnt/data/face\_recognition\_project/templates/register.html", "w") as f:

f.write('''<!DOCTYPE html>

<html>

<head>

<title>User Registration</title>

</head>

<body>

<h1>Register New Face</h1>

{% with messages = get\_flashed\_messages() %}

{% if messages %}

<ul>

{% for msg in messages %}

<li>{{ msg }}</li>

{% endfor %}

</ul>

{% endif %}

{% endwith %}

<form method="POST" enctype="multipart/form-data">

<label>Name:</label><br>

<input type="text" name="name" required><br><br>

<label>Upload Image:</label><br>

<input type="file" name="image" required><br><br>

<button type="submit">Register</button>

</form>

</body>

</html>''')

# Update index.html to include register link

with open("/mnt/data/face\_recognition\_project/templates/index.html", "w") as f:

f.write('''<!DOCTYPE html>

<html>

<head>

<title>Face Recognition</title>

</head>

<body>

<h1>Face Recognition System</h1>

<form action="/recognize" method="post">

<button type="submit">Start Camera</button>

</form>

<br>

<a href="/register">Register New Face</a>

</body>

</html>''')

# Update requirements.txt

with open("/mnt/data/face\_recognition\_project/requirements.txt", "a") as f:

f.write("psycopg2\npickle-mixin\n")

# Zip the updated project

from zipfile import ZipFile

zip\_path = "/mnt/data/face\_recognition\_project\_postgres.zip"

with ZipFile(zip\_path, 'w') as zipf:

for root, dirs, files in os.walk("/mnt/data/face\_recognition\_project"):

for file in files:

full\_path = os.path.join(root, file)

arcname = os.path.relpath(full\_path, "/mnt/data/face\_recognition\_project")

zipf.write(full\_path, arcname)

zip\_path