import os

import numpy as np

import cv2

import base64

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

from tensorflow.keras.models import load\_model

from tensorflow.keras.applications.mobilenet\_v2 import preprocess\_input

import tensorflow as tf

# Suppress TensorFlow logging

tf.get\_logger().setLevel('ERROR')

# Initialize Flask app

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

# Directory to save uploaded files

UPLOAD\_FOLDER = 'static/uploads'

app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER

# Create upload directory if not exists

if not os.path.exists(UPLOAD\_FOLDER):

os.makedirs(UPLOAD\_FOLDER)

# Load model

try:

model = load\_model("Blood Cell.h5")

print("Model loaded successfully.")

except Exception as e:

print(f"Error loading model: {e}")

exit()

# Class labels

class\_labels = ['eosinophil', 'lymphocyte', 'monocyte', 'neutrophil']

# Prediction function

def predict\_image\_class(image\_path):

try:

img = cv2.imread(image\_path)

if img is None:

raise FileNotFoundError("Image could not be loaded.")

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

img\_resized = cv2.resize(img\_rgb, (224, 224))

img\_preprocessed = preprocess\_input(img\_resized.reshape((1, 224, 224, 3)))

predictions = model.predict(img\_preprocessed)

predicted\_class\_idx = np.argmax(predictions, axis=1)[0]

predicted\_class\_label = class\_labels[predicted\_class\_idx]

return predicted\_class\_label, img\_rgb

except Exception as e:

print(f"Prediction error: {e}")

return "Prediction Error", None

# Upload route

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

def upload\_file():

if request.method == "POST":

if "file" not in request.files:

return redirect(request.url)

file = request.files["file"]

if file.filename == "":

return redirect(request.url)

if file:

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

file.save(file\_path)

predicted\_class\_label, img\_rgb = predict\_image\_class(file\_path)

if img\_rgb is None:

return render\_template("home.html", error\_message="Image processing failed.")

\_, img\_encoded = cv2.imencode('.png', cv2.cvtColor(img\_rgb, cv2.COLOR\_RGB2BGR))

img\_base64 = base64.b64encode(img\_encoded.tobytes()).decode('utf-8')

return render\_template("result.html",

class\_label=predicted\_class\_label,

img\_data=img\_base64)

return render\_template("home.html")

# Run the app

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

🖼️ home.html (in templates folder)

html

Copy

Edit

<!DOCTYPE html>

<html>

<head>

<title>Blood Cell Classifier</title>

</head>

<body>

<h1>Upload a Blood Cell Image</h1>

{% if error\_message %}

<p style="color:red;">{{ error\_message }}</p>

{% endif %}

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

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

<input type="submit" value="Upload & Predict">

</form>

</body>

</html>