FLASK CODE:

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

import os

from werkzeug.utils import secure\_filename

from PIL import Image

import torch

from torchvision import transforms, models

from torchvision.models import EfficientNet\_B0\_Weights

# Suppress future warnings globally (optional)

import warnings

warnings.filterwarnings("ignore", category=FutureWarning, module="torch")

# Initialize the Flask app

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

# Set up the upload folder and allowed extensions

UPLOAD\_FOLDER = 'static/uploads'

ALLOWED\_EXTENSIONS = {'png', 'jpg', 'jpeg', 'gif'}

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

# Ensure the upload folder exists

os.makedirs(UPLOAD\_FOLDER, exist\_ok=True)

# Load the trained EfficientNetB0 model

device = torch.device('cuda' if torch.cuda.is\_available() else 'cpu')

model = models.efficientnet\_b0(weights=None) # Use weights=None explicitly

num\_classes = 4 # Replace with the number of classes in your dataset

model.classifier[1] = torch.nn.Linear(model.classifier[1].in\_features, num\_classes)

# Load the trained weights

model.load\_state\_dict(torch.load(r'C:\Users\91984\Desktop\Project\static\models\efficientnet\_best\_model.pth', map\_location=device, weights\_only=False))

model = model.to(device)

model.eval()

# Define image transformations

transform = transforms.Compose([

transforms.Resize((128, 128)),

transforms.ToTensor(),

transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])

])

# Function to check allowed file extensions

def allowed\_file(filename):

return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED\_EXTENSIONS

# Prediction function

def predict\_eye\_disease(image\_path):

try:

img = Image.open(image\_path).convert('RGB')

img\_tensor = transform(img).unsqueeze(0).to(device)

with torch.no\_grad():

outputs = model(img\_tensor)

\_, predicted\_class = torch.max(outputs, 1)

class\_names = ['CNV', 'DME', 'DRUSEN', 'NORMAL'] # Replace with your class names

return class\_names[predicted\_class.item()]

except Exception as e:

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

return "Error in prediction"

# Flask routes

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

def index():

prediction = None

image\_path = None

if request.method == 'POST':

# Check if a file was uploaded

if 'file' not in request.files:

return redirect(request.url)

file = request.files['file']

if file and allowed\_file(file.filename):

# Save the uploaded file

filename = secure\_filename(file.filename)

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

file.save(image\_path)

# Make a prediction

prediction = predict\_eye\_disease(image\_path)

return render\_template('index.html', prediction=prediction, image=os.path.basename(image\_path) if image\_path else None)

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

app.run(debug=True)