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

import os

from datetime import datetime

import cv2

import torch

from werkzeug.utils import secure\_filename

from color\_detector import detect\_watch\_color # Make sure this function exists

# Configuration

UPLOAD\_FOLDER = 'static/uploads'

SAVED\_ITEMS\_FILE = 'saved\_items.txt'

app = Flask(\_name\_)

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

# Ensure upload folder exists

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

# Load YOLOv5 model once (at app start)

model = torch.hub.load(

'ultralytics/yolov5',

'custom',

path='C:/lostandfound/yolov5/runs/train/exp/weights/best.pt', # Change if needed

force\_reload=True

)

# Save detected item info to text file

def save\_item(item, color, filename):

try:

with open(SAVED\_ITEMS\_FILE, 'a') as f:

safe\_item = item.strip().lower()

safe\_color = color.strip().lower()

safe\_filename = filename.strip()

f.write(f"{safe\_item},{safe\_color},{safe\_filename}\n")

except Exception as e:

print(f"Error saving item: {e}")

# Search for matching item

def search\_items(item\_name, color):

try:

if os.path.exists(SAVED\_ITEMS\_FILE):

with open(SAVED\_ITEMS\_FILE, 'r') as f:

for line in f:

parts = line.strip().split(',')

if len(parts) != 3:

continue

saved\_item, saved\_color, saved\_filename = parts

if saved\_item.strip().lower() == item\_name and saved\_color.strip().lower() == color:

return saved\_filename

except Exception as e:

print(f"Error searching items: {e}")

return None

@app.route('/')

def index():

return render\_template('index.html', result\_message='')

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

def predict():

image = request.files.get('image')

if not image:

return jsonify({"success": False, "message": "No image uploaded."})

# Save uploaded image

raw\_name = datetime.now().strftime("%Y%m%d%H%M%S") + '\_' + image.filename

filename = secure\_filename(raw\_name).replace(" ", "\_") # Clean filename

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

image.save(filepath)

# Run YOLOv5 detection

results = model(filepath)

boxes = results.pandas().xyxy[0]

if boxes.empty:

return jsonify({"success": False, "message": "No watch detected."})

# Take the first detected object

row = boxes.iloc[0]

xmin, ymin, xmax, ymax = int(row['xmin']), int(row['ymin']), int(row['xmax']), int(row['ymax'])

# Crop detected watch region

img\_cv = cv2.imread(filepath)

cropped\_img = img\_cv[ymin:ymax, xmin:xmax]

# Save cropped image (optional)

cropped\_path = os.path.join('static', 'cropped.jpg')

cv2.imwrite(cropped\_path, cropped\_img)

# Detect color

box = (xmin, ymin, xmax, ymax)

detected\_color = detect\_watch\_color(filepath, box)

detected\_item = "watch"

# Save item

save\_item(detected\_item, detected\_color, filename)

return jsonify({

"success": True,

"message": f"Upload successful! Detected a {detected\_color} {detected\_item}.",

"color": detected\_color,

"image\_url": f"/static/uploads/{filename}"

})

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

def search():

try:

data = request.get\_json()

item\_name = data.get('item\_name', '').strip().lower()

color = data.get('color', '').strip().lower()

if not item\_name or not color:

return jsonify({"result": "Please provide both item name and color.", "image\_url": ""})

matched\_filename = search\_items(item\_name, color)

if matched\_filename:

return jsonify({

"result": f"Found a matching {color} {item\_name}!",

"image\_url": f"/static/uploads/{matched\_filename}"

})

else:

return jsonify({

"result": f"No {color} {item\_name} found.",

"image\_url": ""

})

except Exception as e:

print(f"Error during search: {e}")

return jsonify({"result": "An error occurred while searching.", "image\_url": ""})

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

app.run(debug=True)