Project Development Phase Sprint-3

|  |  |
| --- | --- |
| Date | 12 November 2022 |
| Team ID | PNT2022TMID40673 |
| Project Name | Virtual Eye - Life Guard for Swimming Pools toDetect Active Drowning |
| Maximum Marks | 4 Marks |

import re

import numpy as np import os

from flask import Flask, app, request, render\_template, redirect, url\_for from tensorflow.keras import models

from tensorflow.keras.models import load\_model from tensorflow.keras.preprocessing import image

from tensorflow.python.ops.gen\_array\_ops import concat import cvlib as cv

from cvlib.object\_detection import draw\_bbox import cv2

import time

from playsound import playsound import requests

#Loading the model

from cloudant.client import Cloudant # Authenticate using an IAM API key

client = Cloudant.iam('57f444d5-dfbd-4fc0-b752-dea54005c3cc- bluemix','HTLp9\_GkWGDyMR9VHruMMwi\_qzZ43qaI3UVR77GOI2GX', connect=True)

# Create a database using an initialized client my\_database = client.create\_database('my\_database')

app=Flask( name )

#default home page or route @app.route('/')

def index():

return render\_template('index.html')

@app.route('/index.html') def home():

return render\_template("index.html")

#registration page @app.route('/register') def register():

return render\_template('register.html')

@app.route('/afterreg', methods=['POST']) def afterreg():

x = [x for x in request.form.values()] print(x)

data = {

'\_id': x[1], # Setting \_id is optional 'name': x[0],

'psw':x[2]

}

print(data)

query = {'\_id': {'$eq': data['\_id']}}

docs = my\_database.get\_query\_result(query) print(docs)

print(len(docs.all()))

if(len(docs.all())==0):

url = my\_database.create\_document(data) #response = requests.get(url)

return render\_template('register.html', pred="Registration Successful, please login using your details")

else:

return render\_template('register.html', pred="You are already a member, please login using your details")

#login page @app.route('/login') def login():

return render\_template('login.html')

@app.route('/afterlogin',methods=['POST']) def afterlogin():

user = request.form['\_id'] passw = request.form['psw'] print(user,passw)

query = {'\_id': {'$eq': user}}

docs = my\_database.get\_query\_result(query) print(docs)

print(len(docs.all()))

if(len(docs.all())==0):

return render\_template('login.html', pred="The username is not found.") else:

if((user==docs[0][0]['\_id'] and passw==docs[0][0]['psw'])): return redirect(url\_for('prediction'))

else:

print('Invalid User')

@app.route('/logout') def logout():

return render\_template('logout.html')

@app.route('/prediction') def prediction():

return render\_template('prediction.html')

@app.route('/result',methods=["GET","POST"]) def res():

webcam = cv2.VideoCapture('drowning.mp4')

if not webcam.isOpened(): print("Could not open webcam") exit()

t0 = time.time() #gives time in seconds after 1970

#variable dcount stands for how many seconds the person has been standing still for

centre0 = np.zeros(2) isDrowning = False

#this loop happens approximately every 1 second, so if a person doesn't move, #or moves very little for 10seconds, we can say they are drowning

#loop through frames while webcam.isOpened():

# read frame from webcam status, frame = webcam.read() #print(frame)

if not status:

print("Could not read frame") exit()

# apply object detection

bbox, label, conf = cv.detect\_common\_objects(frame)

#simplifying for only 1 person #print('bbox',bbox) #print('label',label) #print('conf',conf)

#s = (len(bbox), 2)

if(len(bbox)>0): bbox0 = bbox[0]

#centre = np.zeros(s) centre = [0,0]

#for i in range(0, len(bbox)):

#centre[i] =[(bbox[i][0]+bbox[i][2])/2,(bbox[i][1]+bbox[i][3])/2 ] centre =[(bbox0[0]+bbox0[2])/2,(bbox0[1]+bbox0[3])/2 ]

#make vertical and horizontal movement variables hmov = abs(centre[0]-centre0[0])

vmov = abs(centre[1]-centre0[1])

#there is still need to tweek the threshold

#this threshold is for checking how much the centre has moved x=time.time()

threshold = 10

if(hmov>threshold or vmov>threshold): print(x-t0, 's')

t0 = time.time() isDrowning = False

else:

print(x-t0, 's') if((time.time() - t0) > 10):

isDrowning = True

#print('bounding box: ', bbox, 'label: ' label ,'confidence: ' conf[0], 'centre: ', centre)

#print(bbox,label ,conf, centre)

print('bbox: ', bbox, 'centre:', centre, 'centre0:', centre0) print('Is he drowning: ', isDrowning)

centre0 = centre

# draw bounding box over detected objects #print('came here')

out = draw\_bbox(frame, bbox, label, conf,colors=None,write\_conf=isDrowning) #print('Seconds since last epoch: ', time.time()-t0)

# display output

cv2.imshow("Real-time object detection", out) if(isDrowning == True):

playsound('alarm.mp3') webcam.release() cv2.destroyAllWindows()

#return render\_template('prediction.html',prediction="Emergency !!! The Person is drowining")

#return render\_template('base.html')

# press "Q" to stop

if cv2.waitKey(1) & 0xFF == ord('q'): break

# release resources webcam.release() cv2.destroyAllWindows()

return render\_template('prediction.html',prediction="Emergency !!! The Person is drowining")

""" Running our application """ if name == " main ":

app.run(debug=False)



