APPLICATION BUILDING

Python Code

Date	17 November 2022
Team ID	PNT2022TMID2564
Project Name	Virtual Eye - Life Guard For Swimming Pools To Detect Active Drowning
Maximum Marks	8 Marks

App.py:

```
  # import necessary packages
import cvlib as cv
from cvlib.object_detection import draw_bbox
# import necessary packages
from flask import Flask, render_template, request
import requests
import os
from sys import exit
import cvlib as cv
from cvlib.object_detection import draw_bbox
import cv2
import time
import numpy as np
import math
import argparse
import playsound

import mysql.connector

app = Flask(_name_)

conn=mysql.connector.connect(host="localhost", user="root", password="", database="login")
cursor=conn.cursor()

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

@app.route('/login')
def login(): # put application's code here
    return render_template('login.html')

@app.route('/register')
def register():
```

```
    return render_template('register.html')

@app.route('/home')
def home():
    return render_template('index1.html')


@app.route('/login_validation', methods=['POST'])
def login_validation():
    email=request.form.get('email')
    password=request.form.get('password')

    cursor.execute("""SELECT * FROM `users` WHERE `email` LIKE'{}' AND `password` LIKE
'{}'""".format(email,password))
    users = cursor.fetchall()

    if len(users)>0:
        return render_template('index1.html')
    else:
        return render_template('login.html', prediction_text = "1" )

@app.route('/add_user', methods=['POST'])
def add_user():
    name= request.form.get('name')
    email = request.form.get('email')
    password = request.form.get('password')

    cursor.execute("""INSERT INTO `users`(`id`, `name`, `email`, `password`) VALUES
(NULL,'{}','{}','{}')""".format(name,email,password))
    conn.commit()
    return render_template('login.html', prediction_text = "0")


@app.route('/step2')
def step2():

    print("Begin")

    webcam = cv2.VideoCapture("garden.mp4")
    padding = 20

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


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

        if not status:
            break
    #small_frame = cv2.resize(frame,(0,0),fx = 0.5,fy = 0.5)
    # apply object detection
        bbox, label, conf = cv.detect_common_objects(frame, confidence=0.25, model='yolov3-
tiny')

        print(bbox, label, conf)


        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
            #print("hmov=",hmov)
            if(hmov>threshold or vmov>threshold):
                print(x-t0, 'sif')
                t0 = time.time()
                isDrowning = False

            else:
                print(x-t0, 'selse')
                if((time.time() - t0) > 10):
                    isDrowning = True
```

```
            print('bbox: ', bbox, 'centre:', centre, 'centre0:', centre0)
            print('Is he/she drowning: ', isDrowning)
                    #print('End of the program')


            centre0 = centre
            # draw bounding box over detected objects
            # draw bounding box over detected objects
        out = draw_bbox(frame, bbox, label, conf, write_conf=True)
            # display output
        cv2.imshow("Real-time object detection", out)
        if(isDrowning == True):
            webcam.release()
            cv2.destroyAllWindows()
            return render_template('index1.html', prediction_text = "1")
    # press "Q" to stop
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

# release resources
    webcam.release()
    cv2.destroyAllWindows()
if__name__== '_main_':
            app.run(debug=True)
```