Sprint 4 - code

Date	18 November 2022
Team ID	PNT2022TMID07728
Project Name	VirtualEye - Lifeguard for swimming pools to detect active drowning

app.py

```
from cloudant.client import Cloudant
import numpy as np
import os
from flask import Flask, app,request,render_template
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
from tensorflow.keras.applications.inception_v3 import preprocess_input
import cvlib as cv
from cvlib.object_detection import draw_bbox
import cv2
import time
import numpy as np
from playsound import playsound
#import requests
from flask import Flask, request, render_template, redirect, url_for
#Loading the model


# Authenticate using an IAM API key
client =
Cloudant.iam('06e7c9cd-cbb3-4b56-a40a-e669cf5b0906-bluemix','VPbZAA_fmWRYpJdz4kowa
ZwERWNd4vqCSvOzVI5DXmNn', connect=True)

# Create a database using an initialized client
my_database = client['database1']


app = Flask(_name_)

@app.route("/")
```

```
def index():
    return render_template("./login.html")

@app.route("/about")
def about():
    return render_template("./about.html")

@app.route("/demo")
def demo():
    return render_template("./demo.html")

@app.route("/logout")
def logout():
    return render_template("./logout.html")

@app.route("/register")
def register():
    return render_template("./register.html")

@app.route("/result")
def res():
    webcam = cv2.VideoCapture('drowning7.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




    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()

    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

    #s = (len(bbox), 2)
    print(bbox)
    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 = 30
            if(hmov>threshold or vmov>threshold):
                print(x-t0, 's')
                t0 = time.time()
                isDrowning = False
```

```
            else:

                print(x-t0, 's')
                if((time.time() - t0) > 5):
                    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

        out = draw_bbox(frame, bbox, label, conf,isDrowning)

        #print('Seconds since last epoch: ', time.time()-t0)

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

            playsound('alarm.mp3')

        # press "Q" to stop
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

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



@app.route('/afterreg', methods=['GET'])
def afterreg():
    username = request.args.get('uname')
    password = request.args.get('password')
    print(list(request.form.values()))
```

```
    data = {
    'uname': username,
    'password': password
    }
    print(data)
    query = {'uname': {'$eq': data['uname']}}
    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('login.html', pred="Registration Successful, please login using your
details")
    else:
        return render_template('login.html', pred="You are already a member, please login using
your details")


@app.route('/afterlogin',methods=['GET'])
def afterlogin():
    user = request.args.get('uname')
    passw = request.args.get('password')
    print(user + passw)
    query = {'uname': {'$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]['uname'] and passw==docs[0][0]['password'])):
            return render_template('about.html')
        else:
            return render_template('login.html', pred="incorrect password, please try again.")




if__name__== '_main_':
   app.run()
```