APPLICATION BUILDING

Python Code

Date	17 November 2022
Team ID	PNT2022TMID18501
Project Name	VirtualEye - Life Guard For Swimming Pools To Detect Active Drowning
Maximum Marks	8 Marks

App.py

```
import cv2
import os
import numpy as np
from pathlib import Path
import cvlib as cv
import time
from cv2 import threshold
from cvlib.object_detection import draw_bbox
# from matplotlib.patches import draw_bbox

from flask import Flask , request, render_template , redirect , url_for

from playsound import playsound
# from utils import download_file

from cloudant.client import Cloudant

ACCOUNT_NAME, API_KEY="bd84549c-d8e0-47c4-9fac-c68107bcf136-
bluemix","M2omO01qPVjfoQ0tmEoHfmWIJiYVYIu2JpT9w0puZ1h0"
client=Cloudant.iam(ACCOUNT_NAME, API_KEY, connect=True)

my_database=client.create_database('my_database')
app=Flask(__name__)

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

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

@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],
        '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)
        return render_template('register.html',message='Registration
Successful, Please login using your details')
    else:
        return render_template('register.html',message="You are alredy a
member, please login using your details")
    return "nothing"


@app.route('/login')
def login():
    return render_template('login.html',message="")

@app.route('/afterlogin',methods=['POST'])
def afterlogin():
    x=[x for x in request.form.values()]
    user =x[0]
    passw=x[1]
    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):
```

```
        print("login")
        return render_template('login.html',message="The user is not found")
    else:
        print("holaaaaaaaaaa")
        if((user==docs[0][0]['_id'] and passw==docs[0][0]['psw'])):
            return redirect(url_for('prediction'))
        else:
            print('Invalid User')
            # flash("invalid")
            return render_template('login.html',message="invalid credentials")
    return "nothing"

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

# class dotdict(dict):
#     """dot.notation access to dictionary attributes"""
#     __getattr__ = dict.get
#     __setattr__ = dict.__setitem__
#     __delattr__ = dict.__delitem__

@app.route('/prediction')
def prediction():
    return render_template('prediction.html',prediction="Checking for
drowning")

def draww(frame,bbox,conf):
    for i in range(len(bbox)):
        print(conf)
        start_point = (bbox[i][0], bbox[i][1])
        end_point = (bbox[i][2], bbox[i][3])
        color = (255, 0, 0)
        thickness = 2
        frame = cv2.rectangle(frame, start_point, end_point, color, thickness)
    return frame

@app.route('/result',methods=['GET',"POST"])
def res():
    webcam =cv2.VideoCapture('drowninga.mp4')

    if not webcam.isOpened():
        print("Could Not Open Webcam")
        exit()
    t0=time.time()
    center0=np.zeros(2)
    isDrowning=False
```

```
while webcam.isOpened():
    status,frame=webcam.read()
    bbox,label,conf=cv.detect_common_objects(frame)
    print("seeeeeeee")
    print("-----------------------------------------------")
    print(bbox)
    print("-----------------------------------------------")
    if(len(bbox)>0):

        bbox0=bbox[0]

        center =[0,0]

        center=[(bbox0[0]+bbox0[2])/2,(bbox0[1]+bbox0[3])/2]

        hmov=abs(center[0]-center0[0])
        vmov= abs(center[1]-center0[1])

        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('bbox: ',bbox,'center:',center, 'center0:',center0 )
        print('Is he drowning: ',isDrowning)

        center0 =center

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

        # print(bbbox.x0)
        # out=draw_bbox(frame,bbbox,label,conf)
        # out=draw_bbox(bbox,frame)

        # frame=draww(frame,bbox,conf)
        # out=frame
        out= draw_bbox(frame, bbox, label, conf)
        cv2.imshow("Real-Time objects detection",out)
    else:
        out=frame
        cv2.imshow("Real-Time objects detection",out)
```

```
        # cv2.imshow("Real-Time objects detection",frame)
        if(isDrowning==True):
            #audio =os.path.dirname(__file__)+"/s.wav"
            #playsound(audio)
            playsound("C:\\Users\\SAI\\Downloads\\IBM-Project-2094-1658428458-
main\\IBM-Project-2094-1658428458-main\\Project development phase\\sprint
2/a.mp3")
            webcam.release()
            cv2.destroyAllWindows()
            # return "nothing"
            return render_template('prediction.html',prediction="Emergency !!!
The Person is drowning")

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

    webcam.release()
    cv2.destroyAllWindows()
    return render_template('prediction.html',prediction="Checking for
drowning")

if __name__ =='main':
    app.run(debug=True)
```