APPLICATION BUILDING

BUILD PYTHON CODE

Team ID	TEAM ID : PNT2022TMID07728
Project Name	Virtual Eye - Life Guard for Swimming Pools To Detect Active Drowning
Maximum Marks	4Marks

Import the libraries

```
#import necessary packages

import cv2

import os

import numpy as np

from .utils import download_file

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 #Leading the modelfrom

cloudant.client import Cloudant
```

Create a database using an initiated client

```
from cloudant.client import Cloudant

Authenticate using an IAM API key

client Cloudant, Lam('2eb40045-0886-450d-9d24-82ee7ebb2810-bluemix', 'Udovun TPOT 8h5ZtEq(17xk1g1KeYLmpusCROECONTAR', connectatrue)
```

#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')Gopp.route('/ofterreg',

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'ix[2]

print(data)

query('id': ('Seq': 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 usingyour details") else: return render_template('register.html', pred-"You are already a member, please login using your details")

Configure the login page

#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': {'Seq': 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')

For logout from web application:

@app.route('/Logout')def

logout():

return render_template('Logout.html')

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

res():

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

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

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

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

centre@= np.zeros(2)

isDrowning = False

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

#loop through frames while

webcam.isOpened():

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

Creating bounding box:

bbox, label, conf= cv.detect_common_objects (frame)

#simplifying for only 1 person

#s = (len(bbox), 2)

if(len(bbox)>0):

bboxe= 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 [(bbox®[0]+bbox@[2])/2, (bbox@[1]+bbox@[3])/2]

#make vertical and horizontal movement variables

hmov abs (centre[0]-centre0[0])

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

#this threshold is for checking how much the centre has movedx-

time.time()

threshold 10

if(hmov threshold or veov?threshold):

print(x-te, 's')

to time.time()

isDrowning False

Kelse:

```
print(x-te, ")

if((time.time()- to)> 10): isDrowning True

sprint('bounding box:", bbox, label: label, 'confidence: conf[e], 'centres, centre) #print (bbox,
label,conf, centre) print('bbox, bbox, 'centre:', centre, 'centred:', centree) print('Is hedrowning:,
isDrowning)

centree centre

draw bounding box over detected objects

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

#print("Seconds since last epoch:, time.time()-to)

# display output

cv2.imshow("Real-time object detection", out)

if(isDrowning = True): playsound('alarm.mp3")

webcam.release()

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',)
```

Main function:

```
if__name_=="_main_":
    app.run(debug=True)
```