VirtualEye-Life Guard for Swimming Pools to Detect Active Drowning

TEAM ID : PNT2022TMID44696

TEAM MEMBERS : ELANGOVAN D (732519104006)

MONAL PRATHAP S (732519104018)

TAMIZHARASAN M (732519104029)

KARTHIK S (732519104010)

DEPARTMENT : COMPUTER SCIENCE & ENGINEERING

COLLEGE NAME : SHREE VENKATESWARA HI-TECH ENGINEERING COLLEGE,GOBI.

Project Development PhaseSprint-3

APPLICTION BUILDING

App.py

```
import re
import numpy as np
import os
from flask import Flask, app, request, render_template, redirect,
url_forfrom 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
concatimport cvlib as cv
from cvlib.object_detection import
draw_bboximport cv2
import time
from playsound import
playsoundimport requests

#Loading the model

from cloudant.client import Cloudant# Authenticate using an IAM API key
client = Cloudant.iam(e6697307-8c7b-4a1a-bbbb-4402a47bccd4-bluemix',
tRqrfR8394n1mxhJ8LQfzhPrGByfCS7gC5ezUTZkQRK0', 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('/registe
r')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, pleaselogin using your details")
  else:
    return render_template('register.html', pred="You are already a
member,please login using your details")

#login page
@app.route('/logi
n')def login():
  return render_template('login.html')

@app.route('/afterlogin',methods=['POS
T'])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('/logou
t')def logout():
    return render_template('logout.html')

@app.route('/predictio
n')def prediction():
    return render_template('prediction.html')

@app.route('/result',methods=["GET","P
OST"])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 stillfor
    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
    variableshmov = 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 movedx=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.mp
        3') webcam.release()
        cv2.destroyAllWindo
        ws()
        #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 isdrowining")

""" Running our application """
if __name_== "_main_":
    app.run(debug=False)
```

OUTPUT