Application Building

Build Python Code

Date	17 November 2022
Team ID	PNT2022TMID30291
Project Name	Virtual Eye - Life Guard for Swimming Pools to Detect Active Drowning

Import the libraries:

```
import cvlib as cv
from cvlib.object_detection import draw_bbox
import cv2
import time
import numpy as np
from playsound import playsound
```

```
from flask import Flask, request, render_template, redirect, url_for
#loading the model

from cloudant.client import Cloudant
```

Create a database using an initiated client:

```
#Authenticate using an IAM API key
client = Cloudant.iam('a1885fb9-67af-469a-83e5-f1f78af7b19a-bluemix','dBqnX2HXtY8Dxm6Gd8nWvmiQ5R-f4HaM_seOK96b30zj', connect=True

#Create a database using an initialized client
my_database = client.create_database('my_database')

app=Flask(__name__)
```

Here we will be using a declared constructor to route to the HTML page which we have created earlier:

```
#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')

@app.route('/afterreg', methods=['POST'])
def afterreg():
    x = [x for x in request.form.values()]
    print(x)
    data = {
    'email': x[1], #Setting _id is optional
    'fullname': x[0],
    'password': x[2]
    }
    print(data)

    query = {'email': {'$eq': data['email']}}

    docs = my_database.get_query_result(query)
    print(docs)
```

Configure the registration page:

```
@app.route('/afterreg', methods=['POST'])
def afterreg():
    x = [x for x in request.form.values()]
    print(x)
    data = {
    'email': x[1], #Setting _id is optional
    'fullname': x[0],
    'password': x[2]
    }
    print(data)

    query = {'email': {'$eq': data['email']}}

    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 using your 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['email']
    passw = request.form['password']
    print(user,passw)

    query = {'email': {'$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]['email'] and passw==docs[0][0]['password'])):
            return redirect('/prediction')
        else:
            return render_template('login.html',pred="Invalid user")
```

Logout from web application:

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


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

Open CV Input Capture:

```
webcam = cv2.VideoCapture('sample3.mp4')

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 True:

    # 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)
```

Creating bounding box:

```
# apply object detection
bbox, label, conf = cv.detect_common_objects(frame)
#simplifying for only 1 person

#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 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
        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

out = draw_bbox(frame, bbox, label, conf,isDrowning)
cv2.imwrite('image.jpg',out)

if isDrowning:
    playsound(r'H:\PROJECT FILES\Drowning-Detector\alarm.mp3')
```

Main Function:

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