

Project Development
Phase
Sprint 3 Source code

Date	13 November 2022
Team ID	PNT2022TMID13390
Project Name	Virtual Eye - Life Guard For Swimming Pools To Detect Active Drowning
Maximum Marks	8 Marks

```
import re
import numpy
as npimport 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('57f444d5-dfbd-4fc0-b752-dea54005c3cc-
```

```
bluemix','HTLp9_GkWGDyMR9VHruMMwi_qzZ43qaI3UVR77GOI2  
GX', 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('/regi
ster') 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(qu
ery)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")

#login page
@app.route('/lo
```

```

gin')def
login():
    return render_template('login.html')

@app.route('/afterlogin',methods=[
'POST'])def afterlogin():
    user =
    request.form['_id']
    passwd =
    request.form['psw']
    print(user,passwd)

    query = {'_id': {'$eq': user}}

    docs =
    my_database.get_query_result(qu
    ery)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
        passwd==docs[0][0]['psw'])): return
        redirect(url_for('prediction'))
        else:
            print('Invalid User')

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

```

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

```
@app.route('/result',methods=["GET",  
"POST"])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.
        mp3')
        webcam.release()
        cv2.destroyAllWindows()
        #return
    render_template('prediction.html', prediction="Emergency !!!
    ThePerson is drowning")
    #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 isdrowning")

""" Running our
application """ if ____
name__=="_main_":
    app.run(debug=False)

```

The screenshot shows a code editor with a Python file named `app.py` and a console window displaying the application's output.

Code Editor (app.py):

```

1 import re
2 import numpy as np
3 import os
4 from flask import Flask, app, request, render_template, redirect, url_for
5 from tensorflow.keras.models import load_model
6 from tensorflow.keras.preprocessing import image
7 from tensorflow.python.ops.gen_array_ops import concat
8 import cvlib as cv
9 from cvlib.object_detection import draw_bbox
10
11 import cv2
12 import time
13 from playsound import playsound
14 import requests
15
16 #loading the model
17
18 from cloudant.client import Cloudant
19
20 # Authenticate using an IAM API key
21 client = Cloudant.IAM('57f444ds-dfbd-4fcb-b752-dea54003c3cc-blumix', 'http://gh402yW9W9wUwPwL_q243qz13UwR7N0L2GX', connect=True)
22
23 # Create a database using an initialized client
24 my_database = client.create_database('my_database')
25
26 app = Flask(__name__)
27
28 #default home page on route
29 @app.route('/')
30 def index():
31     return render_template("index.html")
32
33 @app.route('/index.html')
34 def home():
35     return render_template("index.html")
36
37 #registration page
38 @app.route('/register')
39 def register():
40     return render_template("register.html")
41
42 @app.route('/offerreg', methods=['POST'])
43 def offerreg():
44     x = [x for x in request.form.values()]
45     print(x)
46     data = {
47         '_id': x[1], # Setting _id is optional
48         'name': x[0],
49         'psw': x[2]
50     }
51     print(data)
52
53 query = {'_id': ('$eq': data['_id'])}
54
55 docs = my_database.get_query_result(query)
56 print(docs)
57

```

Console Output:

```

In [1]: runfile('C:/Users/Subash V/Desktop/SI-GuidedProject-322143-1664773867-main/GuidedProject-322143-1664773867-main/app.py', wdir='C:/Users/Subash V/Desktop/SI-GuidedProject-322143-1664773867-main/SI-GuidedProject-322143-1664773867-main')

2022-11-12 21:32:54.529455: W tensorflow/stream_executor/platform/default/dso_loader.cc:29] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not found
2022-11-12 21:32:54.529456: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignoring cudart dlerror if you do not have a GPU set up on your machine.
* Serving Flask app "app" (lazy loading)
* Environment: production
   Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [12/Nov/2022 21:33:22] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [12/Nov/2022 21:33:22] "GET / HTTP/1.1" 404 -
127.0.0.1 - - [12/Nov/2022 21:33:26] "GET /login HTTP/1.1" 200 -
subashrains2001@gmail.com Subash@ms2001
<Cloudant.result.QueryResult object at 0x000001f8a304a190>
1
127.0.0.1 - - [12/Nov/2022 21:33:30] "GET /login HTTP/1.1" 302 -
127.0.0.1 - - [12/Nov/2022 21:33:30] "GET /prediction HTTP/1.1" 200 -
1.340908658813277 s
bbox: [[104, 105, 807, 304], [10, 363, 1258, 538]] centre: [455.5, 244.5] centre0:
Is he drowning: False
0.4456934922592943 s

```


