Project Development Phase

Sprint 3

Date	15 November 2022
Team ID	PNT2022TMID41486
Project Name	Virtual Eye - Life Guard for Swimming Pools to Detect Active Drowning
Maximum Marks	4 Marks

```
import re import numpy as np

import os

from flask import Flask, app, request, render_template,
redirect, url_for from 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 concat import
cvlib as cv

from cvlib.object_detection import
draw_bbox import cv2 import time

from playsound import playsound
import 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_qzZ43qaI3UVR77GOI2G
    X', 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('/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], # Setting _id is optional
       'name': x[0],
       'psw':x[2]
```

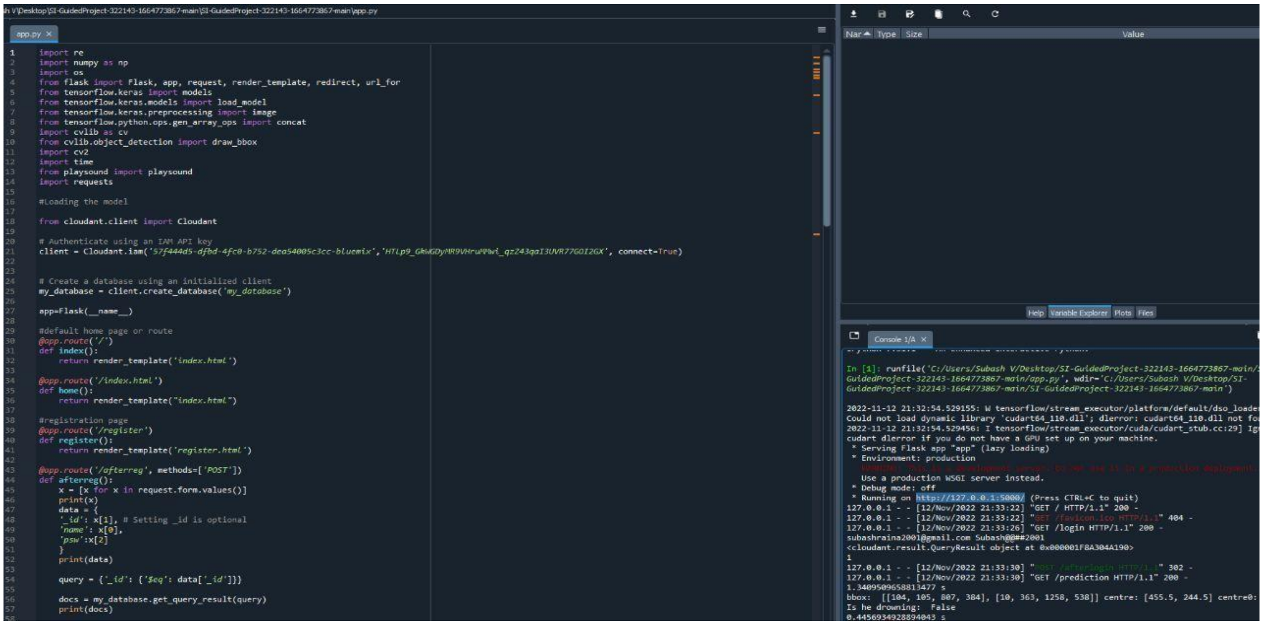
```
         }

         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, 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('/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': {'$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('/logout')
def logout():

 return render_template('logout.html')


@app.route('/prediction'
) def prediction():

 return render_template('prediction.html')


@app.route('/result',methods=["GET","PO
ST"]) 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 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
          webcam.isOpened(): # read
          frame from webcam status,
          frame = webcam.read()
            #print(fram
            e) 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
        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)
```

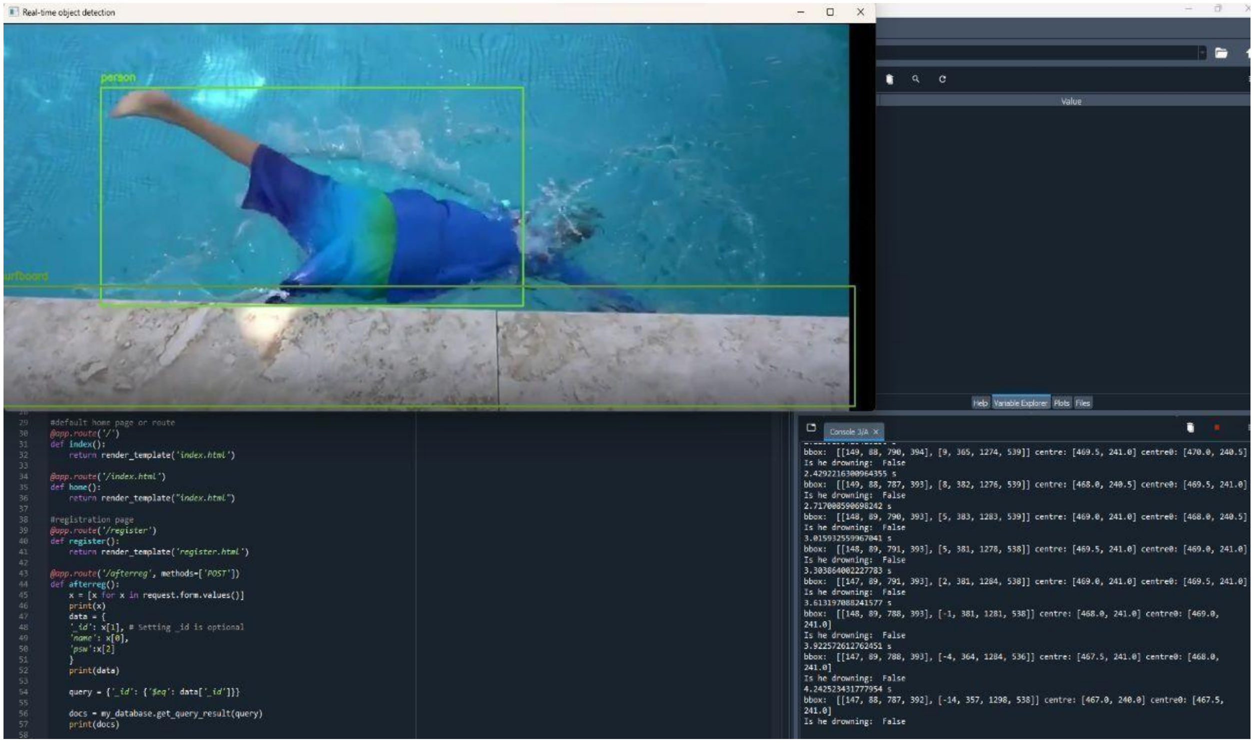
```
            centre0 = centre

            # draw bounding box over detected objects

          #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
            !!! 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
                                                                           is
      drowining")
```

"""" Running our application """ if __name__ == "____main____": app.run(debug=False)


h V\Desktop\SI-GuidedProject-322143-1664773867-main\SI-GuidedProject-322143-1664773867-main\app.py
app.py
Nar
Type
Size
Value
Help
Variable Explorer
Plots
Files
Console 1/A

```python
import re
import numpy as np
import os
from flask import Flask, app, request, render_template, redirect, url_for
from 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 concat
import cvlib as cv
from cvlib.object_detection import draw_bbox
import cv2
import time
from playsound import playsound
import 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_GkWGDyMR9VHruMPwL_qzZ43qaI3UVR77G0I2GX', 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('/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], # 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)
```

```
In [1]: runfile('C:/Users/Subash V/Desktop/SI-GuidedProject-322143-1664773867-main/S
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 21:32:54.529155: W tensorflow/stream_executor/platform/default/dso_loade
Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not fo
2022-11-12 21:32:54.529456: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ig
cudart dlerror if you do not have a GPU set up on your machine.
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   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 /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [12/Nov/2022 21:33:26] "GET /login HTTP/1.1" 200 -
subashraina2001@gmail.com Subash@@##2001
<cloudant.result.QueryResult object at 0x000001F8A304A190>
1
127.0.0.1 - - [12/Nov/2022 21:33:30] "POST /afterlogin HTTP/1.1" 302 -
127.0.0.1 - - [12/Nov/2022 21:33:30] "GET /prediction HTTP/1.1" 200 -
1.3409509658813477 s
bbox:  [[104, 105, 807, 384], [10, 363, 1258, 538]] centre: [455.5, 244.5] centre0:
Is he drowning:  False
0.4456934928894043 s
```



```python
#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 = {
    '_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)
```

```
bbox:  [[149, 88, 790, 394], [9, 365, 1274, 539]] centre: [469.5, 241.0] centre0: [470.0, 240.5]
Is he drowning:  False
2.4292216300964355 s
bbox:  [[149, 88, 787, 393], [8, 382, 1276, 539]] centre: [468.0, 240.5] centre0: [469.5, 241.0]
Is he drowning:  False
2.717008590698242 s
bbox:  [[148, 89, 790, 393], [5, 383, 1283, 539]] centre: [469.0, 241.0] centre0: [468.0, 240.5]
Is he drowning:  False
3.015932559967041 s
bbox:  [[148, 89, 791, 393], [5, 381, 1278, 538]] centre: [469.5, 241.0] centre0: [469.0, 241.0]
Is he drowning:  False
3.303864002227783 s
bbox:  [[147, 89, 791, 393], [2, 381, 1284, 538]] centre: [469.0, 241.0] centre0: [469.5, 241.0]
Is he drowning:  False
3.613197088241577 s
bbox:  [[148, 89, 788, 393], [-1, 381, 1281, 538]] centre: [468.0, 241.0] centre0: [469.0,
241.0]
Is he drowning:  False
3.922572612762451 s
bbox:  [[147, 89, 788, 393], [-4, 364, 1284, 536]] centre: [467.5, 241.0] centre0: [468.0,
241.0]
Is he drowning:  False
4.242523431777954 s
bbox:  [[147, 88, 787, 392], [-14, 357, 1298, 538]] centre: [467.0, 240.0] centre0: [467.5,
241.0]
Is he drowning:  False
```