

Project Development

Phase

Sprint 3 Source code

```
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('/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
        '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
    passw==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('/predi
ction')def
prediction():
    return render_template('prediction.html')

@app.route('/result',methods=["GE
T","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 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(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 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 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 is drowning")

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

```


h:\Desktop\SI-GuidedProject-322143-1664773867-main\SI-GuidedProject-322143-1664773867-main\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 import models
6 from tensorflow.keras.models import load_model
7 from tensorflow.keras.preprocessing import image
8 from tensorflow.python.ops.gen_array_ops import concat
9 import cvlib as cv
10 from cvlib.object_detection import draw_bbox
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('5f7444d5-dfbd-4fc0-b752-dca54005c3cc-bluewin', 'HTLp9_Gh6GyW9V9rUwWt_gz243pI3UR77G0I2GX', connect=True)
22
23
24 # Create a database using an initialized client
25 my_database = client.create_database('my_database')
26
27 app=Flask(__name__)
28
29 #default home page or route
30 @app.route('/')
31 def index():
32     return render_template("index.html")
33
34 @app.route("/index.html")
35 def home():
36     return render_template("index.html")
37
38 #registration page
39 @app.route("/register")
40 def register():
41     return render_template("register.html")
42
43 @app.route("/afterreg", methods=['POST'])
44 def afterreg():
45     x = [x for x in request.form.values()]
46     print(x)
47     data = {
48         'id': x[1], # Setting _id is optional
49         'name': x[0],
50         'paw':x[2]
51     }
52     print(data)
53     query = {'_id': ['$eq': data['_id']]}
54     docs = my_database.get_query_result(query)
55     print(docs)
```

Var Type Size Value

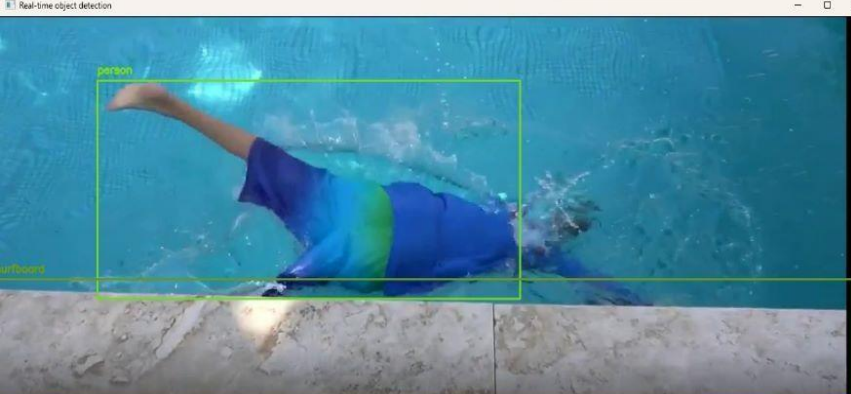
Help Variable Explorer Files F10

Console [X] X

```
h:\Desktop\SI-GuidedProject-322143-1664773867-main\SI-GuidedProject-322143-1664773867-main\app.py
In [4]: 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 21:32:54.529155: W tensorflow/stream_executor/platform/default/dso_loader
Could not load dynamic library 'cudart64_118.dll'; dlerror: cudart64_118.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
  Use a production WSGI server instead.
  * Debug mode: off
  * Running on http://0.0.0.0:5000 (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 0x000001f6a304a190>
1
127.0.0.1 - - [12/Nov/2022 21:33:30] "GET /prediction HTTP/1.1" 302 -
127.0.0.1 - - [12/Nov/2022 21:33:30] "GET /prediction HTTP/1.1" 200 -
1.540950655815477 s
bbox: [[184, 185, 807, 384], [10, 363, 1258, 538]] centre: [455.5, 244.5] centre0:
Is the drawing: false
0.4445034022894043 s
```

Real-time object detection



```
29 #default home page or route
30 @app.route("/")
31 def index():
32     return render_template("index.html")
33
34 @app.route("/index.html")
35 def home():
36     return render_template("index.html")
37
38 #registration page
39 @app.route("/register")
40 def register():
41     return render_template("register.html")
42
43 @app.route("/afterreg", methods=['POST'])
44 def afterreg():
45     x = [x for x in request.form.values()]
46     print(x)
47     data = {
48         'id': x[1], # setting_id is optional
49         'name': x[0],
50         'pos': x[2]
51     }
52     print(data)
53
54     query = {'_id': {'$eq': data['_id']}}
55     docs = my_database.get_query_result(query)
56     print(docs)
57
58
```

Console Jupyter Web

```
-----
bboxes: [[149, 88, 790, 394], [9, 365, 1274, 539]] centre: [469.5, 241.0] centre: [470.0, 240.5]
is he drowning: False
2.429221838095455 s
bboxes: [[149, 88, 787, 393], [8, 382, 1276, 539]] centre: [468.0, 240.5] centre: [469.5, 241.0]
is he drowning: False
2.717805966060440 s
bboxes: [[149, 89, 790, 393], [5, 383, 1283, 539]] centre: [469.0, 241.0] centre: [468.0, 240.5]
is he drowning: False
3.01591255989944 s
bboxes: [[149, 89, 791, 393], [5, 381, 1278, 538]] centre: [469.5, 241.0] centre: [469.0, 241.0]
is he drowning: False
3.203064061223793 s
bboxes: [[147, 89, 791, 393], [2, 381, 1284, 538]] centre: [469.0, 241.0] centre: [469.5, 241.0]
is he drowning: False
3.613197802248197 s
bboxes: [[140, 89, 788, 393], [-1, 381, 1281, 538]] centre: [468.0, 241.0] centre: [469.0, 241.0]
is he drowning: False
3.922572611764551 s
bboxes: [[147, 89, 788, 393], [-4, 364, 1284, 536]] centre: [467.5, 241.0] centre: [468.0, 241.0]
is he drowning: False
4.242523431777954 s
bboxes: [[147, 88, 787, 392], [-14, 357, 1290, 538]] centre: [467.0, 240.0] centre: [467.5, 241.0]
is he drowning: False
```

