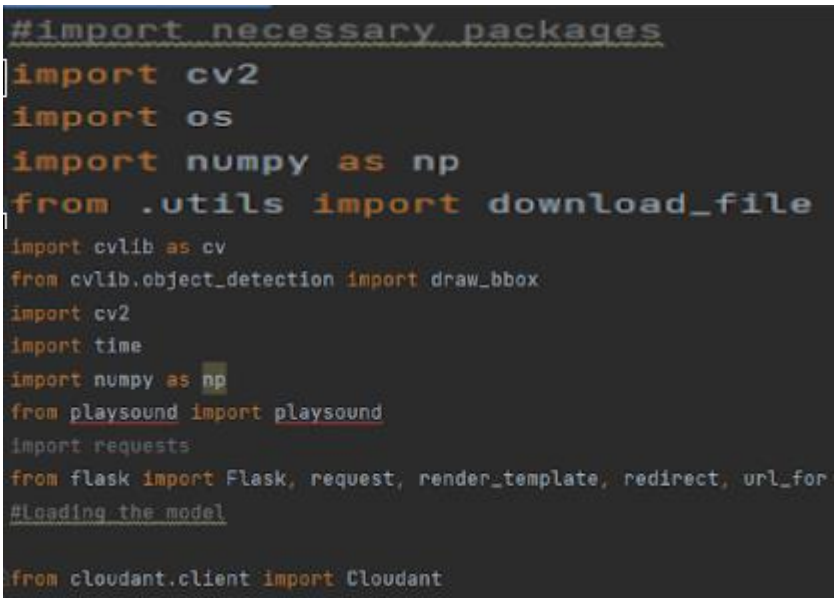
Date	30 OCTOBER 2022
Team ID	PNT2022TMID13803
Project Name	Virtual eye – lifeguard for swimming pool to Detect Active Drowning

Build Python Code

Import the libraries



```
#import necessary packages
import cv2
import os
import numpy as np
from .utils import download_file
import cvlib as cv
from cvlib.object_detection import draw_bbox
import cv2
import time
import numpy as np
from playsound import playsound
import requests
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.

An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (__name__) as argument.

```
app=Flask(__name__)
```

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

In the above example, '/' URL is bound with the index function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered.

```
#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')
```

Configure the registration page

Based on user input into the registration form we stored it on data dictionary then we can validate the data using _id parameter with user input that we can store it on query variable then we can validate by passing the query variable into the my_database.get_user_result() method.Then we can check the docs length by using len(docs.all()) function.If the length of docs is 0 then user will register successfully on the platform and user data will store on the database.Otherwise its shows

the message as user already registered please login and use our web application for DR prediction.

```
@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")
```

Configure the login page

Based on user input into the login form we stored user id and password into the (user,passw) variables. Then we can validate the credentials using _id parameter with user input that we can store it on a query variable then we can validate by passing the query variable into the my_database.get_user_result() method. Then we can check the docs length by using len(docs.all()) function.If the length of doc is 0 then it means username is not found.Otherwise it validates the data that is stored on the database and checks the username & password. If it's matched then the user will be able to login and use our web application for DR prediction.Otherwise the user needs to provide correct credentials.

```
#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')
```

For logout from web application.

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

Create res() function for drowning detection

Open CV Input Capture: We get ready with video input needed to process the drowning detection.In OpenCV, video is captured using VideoCapture() Function. It takes one parameter – the source of the video. It can either be from the External Camera (Front or Rear Camera) or any previous recorded Video. Provide 0 for Front Camera and 1 for Rear Camera. If you want to use pre-recorded video, Provide the path of the source video.

```
@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 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()
```

Creating bounding box

If the drowning person is identified, The bounding box is created .

```
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])
```

Draw bbox function is used to draw bounding boxes, find the labels of the object detected and predict if the person is drowning or not by calculating its centroid .

```
        #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)

    #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()
```

Now we check if a person is drowning then an emergency alarm is triggered using playsound method and alert will be showcased on UI.If the person is not drowning then it will check till the video end and opencv window will terminate.

```
        # 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',)
```

Main Function:

```
""" Running our application """
if __name__ == "__main__":
    app.run(debug=True)
```