```
import
time



          import cv2
          import numpy as np
          from cloudant.client import Cloudant
          from flask import Flask, request, render_template, redirect, url_for
          from playsound import playsound


          import cvlib as cv
          from cvlib.object_detection import draw_bbox


          # Loading the model


          # Authenticate using an IAM API key
          client = Cloudant.iam('8780b82a-5a3b-4da0-a180-a0e1516479f9-bluemix',
          'TzYs8u0Q5eoj204gDo2eOEDAuGRhj0fG_9rlZr5SsJUH',
                                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", "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()


        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)
        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)
        # 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',)





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