Project Development Phase
Sprint - 3
Application Building

DATE	9 NOV 2022
TEAM ID	PNT2022TMID49543
PROJECT NAME	Virtual Eye - LifeGuard For Swimming Pools To Detect
MAXIMUM MARKS	8 MARKS

Building Html Pages

Index.html:

```
<!-- <!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
<style>
    ul {
      list-style-type: none;
      margin: 0;
      padding: 0;
      overflow: hidden;
    }

    li {
      float: left;
    }

    li a {
      display: block;
      padding: 8px;
      background-color: #dddddd;
    }
    </style>
</head>
<body>
<h1>Virtual EYE</h1>
```

```
<ul>
    <li><a href="index.html">Home</a></li>
    <li><a href="login.html">Login</a></li>
    <li><a href="register.html">Register</a></li>
    <li><a href="demo.html">Demo</a></li>
  </ul>
</body>
</html> -->


<!DOCTYPE html>
<html >

<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial- scale=1">
<title>Virtual Eye</title>
<link href='https://fonts.googleapis.com/css?family=Pacifico'
rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo'
rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300'
rel='stylesheet' type='text/css'>

<link href='https://fonts.googleapis.com/css?family=Open+Sans+Conde
nsed:300' rel='stylesheet' type='text/css'>
<!-- <link rel="stylesheet" href="{{ url_for('static',
filename='css/style.css') }}"> -->

<link href='https://fonts.googleapis.com/css?family=Merriweather'
rel='stylesheet'>
<link href='https://fonts.googleapis.com/css?family=Josefin Sans'
rel='stylesheet'>
<link href='https://fonts.googleapis.com/css?family=Montserrat'
rel='stylesheet'>


<style>
.header {
    top:0; margin:0px; left: 0px; right: 0px;
position: fixed;
background-color: #28272c; color: white;
box-shadow: 0px 8px 4px grey; overflow: hidden;
```

```
padding-left:20px;
font-family: 'Josefin Sans'; font-size: 2vw;
width: 100%; height:8%;
text-align: center;
}
tabb{
        box-sizing: border-box;
    }
    /* Set additional styling options for the columns*/
    .column {
    float: left;
    width: 50%;
    }

    .row:after {
    content: "";
    display: table;
    clear: both;
    }
.topnav {

overflow: hidden; background-color: #333;
}

.topnav-right a { float: left; color: #f2f2f2;
text-align: center; padding: 14px 16px; text-decoration: none;
font-size: 18px;
}

.topnav-right a:hover { background-color: #ddd; color: black;
}

.topnav-right a.active { background-color: #565961; color: white;
}

.topnav-right { float: right;
padding-right:100px;
}

.login{
margin-top:-70px;
}
body {
```

```
background-color:#ffffff; background-repeat: no-repeat;
background-size:cover; background-position: 0px 0px;
}
.login{
margin-top:100px;
}
form {border: 3px solid #f1f1f1; margin-left:400px;margin-right:400px;}

input[type=text],
input[type=email],input[type=number],input[type=password] { width:
100%;
padding: 12px 20px; display: inline-block; margin-bottom:18px; border:
1px solid #ccc; box-sizing: border-box;
}

button {
background-color: #28272c; color: white;
padding: 14px 20px; margin-bottom:8px; border: none; cursor: pointer;
width: 100%;
}

button:hover { opacity: 0.8;

}

.cancelbtn { width: auto;
padding: 10px 18px; background-color: #f44336;
}

.imgcontainer { text-align: center;
margin: 0px 0 0px 0;
padding-top: 0px;
}
.textt{
    text-align: center;
    font-size: 40px;
    text-decoration: underline;
    text-decoration-color: yellow

}
section {
  display: flex;
```

```
  flex-wrap: wrap;
}

section .col {
  flex: 1 1 auto;
}

section .line-break {
  flex-basis: 100%;
  width: 0px;
  height: 0px;
  overflow: hidden;
}
.column {
  float: left;
  width: 50%;
  padding: 10px;
  height: 300px; /* Should be removed. Only for demonstration */
}

/* Clear floats after the columns */
.row:after {
  content: "";
  display: table;
  clear: both;
}
img.avatar {
    width:30%;
/* border-radius: 50%; */
}
.tabb{
    align-items: center;
}

.container { padding: 16px;
}

span.psw { float: right;
padding-top: 16px;

}
section {
  width: 100%;
```

```
}

article {
  position: relative;
  top: 50%;
  left: 50%;
  padding: 1rem;
  text-align: justify;
  transform: translate(-50%, -50%);
}

h1 {
  font-size: 1.75rem;
  margin: 0 0 0.75rem 0;
  text-align: center;
}

/* Pattern styles */
.left-half {

  float: left;
  width: 50%;
}
.right-half {

  float: left;
  width: 50%;
}
.vertical {
            border-left: 1px solid #808080;
            width: 8px;
            border-right:1px solid #808080;;
            height: 230px;
            position:absolute;
            left: 51%;
        }
/* Change styles for span and cancel button on extra small screens
*/
@media screen and (max-width: 300px) { span.psw {
display: block;

float: none;
}
```

```
.cancelbtn { width: 100%;
}
}

</style>
</head>

<body style="font-family:Montserrat;">

<div class="header">
<div
style="width:50%;float:left;font-size:2vw;text-align:left;color:white;
padding-top:1%">Virtual Eye</div>
<div class="topnav-right" >

<a href="index.html">Home</a>
<a href="login.html">Login</a>
<a  href="register.html">Register</a>

</div>
</div>
<div id="login" class="login">
    <div class="imgcontainer">
        <img width=100% src="{{url_for('static',
filename='pics/drowningkid.jpg')}}" alt="Avatar" >
        </div>
        <div class="textt">
           <p class="text-decoration-underline">ABOUT PROJECT</p>

        </div>
        <section class="container">
            <div class="left-half">
              <article>
                <h1>Problem:</h1>
                <p> Swimming is one of the best exercises that helps
people to reduce
                    stress in this urban lifestyle. Swimming pools are
found larger in number in hotels,
                    and weekend tourist spots and barely people have them
in their house backyard.
                    Beginners, especially, often feel it difficult to
breathe underwater which causes
```

```
                    breathing trouble which in turn causes a drowning
accident. Worldwide,
                    drowning produces a higher rate of mortality without
causing injury to children.
                    Children under six of their age are found to be
suffering the highest drowning mortality rates worldwide.
                    Such kinds of deaths account for the third cause of
unplanned death globally,
                    with about 1.2  million cases yearly.
                      </p>
                </article>
              </div>
              <div class="vertical"></div>
              <div class="right-half">
                <article>
                  <h1>Solution:</h1>
                  <p>To overcome this conflict, a meticulous system is to
be implemented along the swimming pools
                    to save human life.  By studying body movement
patterns and connecting cameras to artificial
                    intelligence (AI) systems we can devise an
underwater pool safety system that reduces
                    the risk of drowning.  Usually, such systems can be
developed by installing more than
                    16 cameras underwater and ceiling and analyzing the
video feeds to detect any anomalies.
                    but  AS a POC we make use of one camera that
streams the video underwater and analyses
                    the position of swimmers to assess the probability
of drowning, if it is higher than an
                    alerts will be generated to attract lifeguards'
attention.

                  </p>
                </article>
              </div>
            </section>
</body>
</html>
```

login.html

```
<!-- <!DOCTYPE html>
<html>
```

```
<head>
<title>Page Title</title>
<style>
    ul {
      list-style-type: none;
      margin: 0;
      padding: 0;
      overflow: hidden;
    }

    li {
      float: left;
    }

    li a {
      display: block;
      padding: 8px;
      background-color: #dddddd;
    }
    </style>
</head>
<body>
<h1>Virtual EYE</h1>
<ul>
    <li><a href="index">Home</a></li>
    <li><a href="login">Login</a></li>
    <li><a href="register">Register</a></li>
  </ul>
  {% block content %}

    <form action ="http://localhost:5000/afterlogin"method="post" >

        <input type="mail" name="email"
        placeholder="Enter EmailId"
        value="{{ request.form['email'] }}"></input>

        <br>
        <input type="password" name="password"
        placeholder="Enter your password"
        value="{{ request.form['password'] }}"></input>
        <br>
        <h1>{{message}}</h1>
        <button type="submit">Submit</button>
```

```
    </form>
{% endblock %}
</body>
</html>
 -->

 <!DOCTYPE html>
 <html >

 <head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial- scale=1">
 <title>Virtual Eye</title>
 <link href='https://fonts.googleapis.com/css?family=Pacifico'
rel='stylesheet' type='text/css'>
 <link href='https://fonts.googleapis.com/css?family=Arimo'
rel='stylesheet' type='text/css'>
 <link href='https://fonts.googleapis.com/css?family=Hind:300'
rel='stylesheet' type='text/css'>
 <link href='https://fonts.googleapis.com/css?family=Open+Sans+Conde
nsed:300' rel='stylesheet' type='text/css'>
 <!-- <link rel="stylesheet" href="{{ url_for('static',
filename='css/style.css') }}"> -->

 <link href='https://fonts.googleapis.com/css?family=Merriweather'
rel='stylesheet'>
 <link href='https://fonts.googleapis.com/css?family=Josefin Sans'
rel='stylesheet'>
 <link href='https://fonts.googleapis.com/css?family=Montserrat'
rel='stylesheet'>




 <style>
 .header {

     top:0; margin:0px; left: 0px; right: 0px;
 position: fixed;
 background-color: #28272c; color: white;
 box-shadow: 0px 8px 4px grey; overflow: hidden;
 padding-left:20px;
 font-family: 'Josefin Sans'; font-size: 2vw;
```

```
 width: 100%; height:8%;
 text-align: center;

 }
 .topnav {



 top:0; margin:0px; left: 0px; right: 0px;
 position: fixed;
 background-color: #28272c; color: white;
 box-shadow: 0px 8px 4px grey; overflow: hidden;
 padding-left:20px;
 font-family: 'Josefin Sans'; font-size: 2vw;
 width: 100%; height:8%;
 text-align: center;

 overflow: hidden; background-color: #333;
 }

 .topnav-right a {

 float: left; color: #f2f2f2;
 text-align: center; padding: 14px 16px; text-decoration: none;
font-size: 18px;
 }

 .topnav-right a:hover { background-color: #ddd; color: black;
 }

 .topnav-right a.active { background-color: #565961; color: white;
 }

 .topnav-right { float: right;
 padding-right:100px;
 }

 .login{
 margin-top:-70px;
 }
 body {
```

```
 background-color:#ffffff; background-repeat: no-repeat;
background-size:cover; background-position: 0px 0px;
 }

 .login{
 margin-top:100px;
 }
 form {border: 3px solid #f1f1f1;
margin-left:400px;margin-right:400px;}

 input[type=text],
input[type=email],input[type=number],input[type=password] { width:
100%;
 padding: 12px 20px; display: inline-block; margin-bottom:18px; border:
1px solid #ccc; box-sizing: border-box;
 }

 button {
 background-color: #28272c; color: white;
 padding: 14px 20px; margin-bottom:8px; border: none; cursor: pointer;
width: 100%;
 font-weight:bold;
 }

 button:hover { opacity: 0.8;
 }

 .cancelbtn { width: auto;

 padding: 10px 18px; background-color: #f44336;
 }

 .imgcontainer { text-align: center;
 margin: 24px 0 12px 0;
 }

 img.avatar { width: 30%;
 border-radius: 50%;
 }

 .container { padding: 16px;
 }
```

```
 span.psw { float: right;
 padding-top: 16px;

 }

 /* Change styles for span and cancel button on extra small screens
 */
 @media screen and (max-width: 300px) { span.psw {
 display: block; float: none;
 }
 .cancelbtn { width: 100%;

 }
 }

 </style>
 </head>

 <body style="font-family:Montserrat;">

 <div class="header">
 <div
style="width:50%;float:left;font-size:2vw;text-align:left;color:white;
padding-top:1%">Virtual Eye</div>
 <div class="topnav-right" style="padding-top:0.5%;">

 <a href="index.html">Home</a>
     <a href="login.html">Login</a>
     <a  href="register.html">Register</a>

 </div>
 </div>
 <div id="login" class="login">


 <form action="{{url_for('afterlogin')}}" method="post">
 <div class="imgcontainer">
 <img src="{{url_for('static', filename='pics/avatar.jpg')}}"
alt="Avatar" class="avatar">
 </div>

 <div class="container">
```

```
 <input type="email" placeholder="Enter registered email ID"
name="email" value=><br>


 <input type="password" placeholder="Enter Password" name="password"
value=>
 <h1></h1>
 <button type="submit">Login</button><br>

 </div>

 </form>

 </div>


 </body>
 </html>
```

Prediction.html

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
<style>
    ul {
      list-style-type: none;
      margin: 0;
      padding: 0;
      overflow: hidden;
    }

    li {
      float: left;
    }

    li a {
      display: block;
      padding: 8px;
      background-color: #dddddd;
    }
```

```
    </style>
</head>
<body>

<h1>Virtual EYE</h1>
<ul>
    <li><a href="index.html">Home</a></li>
    <li><a href="login.html">Login</a></li>
    <li><a href="register.html">Register</a></li>
  </ul>
  {% block content %}
    <h4> {{prediction}}</h4>
    <form action ="http://localhost:5000/result"method="get" >
        <button type="submit">Predict</button>
    </form>
{% endblock %}
</body>
</html>
```

Register.html

```
<!-- <!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
<style>
    ul {
      list-style-type: none;
      margin: 0;
      padding: 0;
      overflow: hidden;
    }

    li {
      float: left;
    }

    li a {
      display: block;
      padding: 8px;
      background-color: #dddddd;
    }
    </style>
</head>
```

```
<body>
<h1>Virtual EYE</h1>
<ul>
    <li><a href="index">Home</a></li>
    <li><a href="login">Login</a></li>
    <li><a href="demo">Demo</a></li>
  </ul>
  {% block content %}

    <form action ="http://localhost:5000/afterreg"method="post" >

        <br>
        <input type="text" name="name"
               placeholder="Enter name"
               value="{{ request.form['name'] }}"></input>
        <br>
        <input type="mail" name="email"
        placeholder="Enter EmailId"
        value="{{ request.form['email'] }}"></input>

        <br>
        <input type="password" name="password"
        placeholder="Enter your password"
        value="{{ request.form['password'] }}"></input>
        <br>
        <h1>{{message}}</h1>
        <button type="submit">Submit</button>
    </form>
{% endblock %}
</body>
</html>
 -->


 <!DOCTYPE html>
<html >

<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial- scale=1">
<title>Virtual Eye</title>
<link href='https://fonts.googleapis.com/css?family=Pacifico'
rel='stylesheet' type='text/css'>
```

```
<link href='https://fonts.googleapis.com/css?family=Arimo'
rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300'
rel='stylesheet' type='text/css'>

<link href='https://fonts.googleapis.com/css?family=Open+Sans+Conde
nsed:300' rel='stylesheet' type='text/css'>
<!-- <link rel="stylesheet" href="{{ url_for('static',
filename='css/style.css') }}"> -->

<link href='https://fonts.googleapis.com/css?family=Merriweather'
rel='stylesheet'>
<link href='https://fonts.googleapis.com/css?family=Josefin Sans'
rel='stylesheet'>
<link href='https://fonts.googleapis.com/css?family=Montserrat'
rel='stylesheet'>


<style>
.header {
top:0; margin:0px; left: 0px; right: 0px;
position: fixed;
background-color: #28272c; color: white;
box-shadow: 0px 8px 4px grey; overflow: hidden;
padding-left:20px;
font-family: 'Josefin Sans'; font-size: 2vw;
width: 100%; height:8%;
text-align: center;
}

.topnav {

overflow: hidden; background-color: #333;
}

.topnav-right a { float: left; color: #f2f2f2;
text-align: center; padding: 14px 16px; text-decoration: none;
font-size: 18px;
}

.topnav-right a:hover { background-color: #ddd; color: black;
}
```

```
.topnav-right a.active { background-color: #565961; color: white;
}

.topnav-right { float: right;
padding-right:100px;
}

.login{
margin-top:-70px;
}
body {

background-color:#ffffff; background-repeat: no-repeat;
background-size:cover; background-position: 0px 0px;
}
.login{
margin-top:100px;
}
form {border: 3px solid #f1f1f1; margin-left:400px;margin-right:400px;}

input[type=text],
input[type=email],input[type=number],input[type=password] { width:
100%;
padding: 12px 20px; display: inline-block; margin-bottom:18px; border:
1px solid #ccc; box-sizing: border-box;
}

button {
background-color: #28272c; color: white;
padding: 14px 20px; margin-bottom:8px; border: none; cursor: pointer;
width: 100%;
}

button:hover { opacity: 0.8;

}

.cancelbtn { width: auto;
padding: 10px 18px; background-color: #f44336;
}

.imgcontainer { text-align: center;
margin: 24px 0 12px 0;
```

```
}

img.avatar { width: 30%;
border-radius: 50%;
}

.container { padding: 16px;
}

span.psw { float: right;
padding-top: 16px;

}

/* Change styles for span and cancel button on extra small screens
*/
@media screen and (max-width: 300px) { span.psw {
display: block;

float: none;
}
.cancelbtn { width: 100%;
}
}

</style>
</head>

<body style="font-family:Montserrat;">

<div class="header">
<div
style="width:50%;float:left;font-size:2vw;text-align:left;color:white;
padding-top:1%">Virtual Eye</div>
<div class="topnav-right" >

<a href="index.html">Home</a>
<a href="login.html">Login</a>
<a  href="register.html">Register</a>

</div>
</div>
<div id="login" class="login">
```

```
<form action="{{url_for('afterreg')}}" method="post">
<div class="imgcontainer">
<img src="{{url_for('static', filename='pics/avatar.jpg')}}"
alt="Avatar" class="avatar">
</div>


<div class="container">
<input type="text" placeholder="Enter Name" name="name" value= ><br>
<input type="email" placeholder="Enter Email ID" name="_id"  value=
><br>
<input type="password" placeholder="Enter Password" name="psw" value= >
<h1></h1>
<button type="submit">Register</button><br>

</div>
<div class="container" style="background-color:#f1f1f1">
<div class="psw">Already have an account?   <a href="{{
url_for('login') }}">Login</a></div >
</div>
</form>

</div>


</body>
</html>
```

Python Code:

init

```
from object_detection import detect_common_objects
```

utils

```
import requests
import progressbar as pb
import os
```

```
def download_file(url, file_name, dest_dir):
    if not os.path.exists(dest_dir):
        os.makedirs(dest_dir)

    full_path_to_file = dest_dir + os.path.sep + file_name

    if os.path.exists(dest_dir + os.path.sep + file_name):
        return full_path_to_file

    print("Downloading " + file_name + " from " + url)

    try:
        r = requests.get(url, allow_redirects=True, stream=True)
    except:
        print("Could not establish connection. Download failed")
        return None

    file_size = int(r.headers['Content-Length'])
    chunk_size = 1024
    num_bars = round(file_size / chunk_size)

    bar = pb.ProgressBar(maxval=num_bars).start()

    if r.status_code != requests.codes.ok:
        print("Error occurred while downloading file")
        return None

    count = 0

    with open(full_path_to_file, 'wb') as file:
        for chunk in r.iter_content(chunk_size=chunk_size):
            file.write(chunk)
            bar.update(count)
            count += 1

    return full_path_to_file
```

object_detection

```
import cv2
import os
```

```
import numpy as np
from utils import download_file

initialize = True
net = None
dest_dir = os.path.expanduser(
    '~') + os.path.sep + '.cvlib' + os.path.sep + 'object_detection' +
os.path.sep + 'yolo' + os.path.sep + 'yolov3'
classes = None
# colors are BGR instead of RGB in python
COLORS = [0, 0, 255], [255, 0, 0]


def populate_class_labels():
    # we are using a pre existent classifier which is more reliable and
more efficient than one
    # we could make using only a laptop
    # The classifier should be downloaded automatically when you run
this script
    class_file_name = 'yolov3_classes.txt'
    class_file_abs_path = dest_dir + os.path.sep + class_file_name

    if not os.path.exists(class_file_abs_path):
        download_file(url=url, file_name=class_file_name,
dest_dir=dest_dir)
    f = open(class_file_abs_path, 'r')
    classes = [line.strip() for line in f.readlines()]

    return classes


def get_output_layers(net):
    # the number of output layers in a neural network is the number of
possible
    # things the network can detect, such as a person, a dog, a tie, a
phone...
    layer_names = net.getLayerNames()

    output_layers = [layer_names[i[0] - 1] for i in
net.getUnconnectedOutLayers()]

    return output_layers
```

```
def draw_bbox(img, bbox, labels, confidence, Drowning,
write_conf=False):
    global COLORS
    global classes

    if classes is None:
        classes = populate_class_labels()

    for i, label in enumerate(labels):

        # if the person is drowning, the box will be drawn red instead
of blue
        if label == 'person' and Drowning:
            color = COLORS[0]
            label = 'DROWNING'
        else:
            color = COLORS[1]

        if write_conf:
            label += ' ' + str(format(confidence[i] * 100, '.2f')) +
'%'

        # you only need to points (the opposite corners) to draw a
rectangle. These points
        # are stored in the variable bbox
        cv2.rectangle(img, (bbox[i][0], bbox[i][1]), (bbox[i][2],
bbox[i][3]), color, 2)

        cv2.putText(img, label, (bbox[i][0], bbox[i][1] - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)

    return img


def detect_common_objects(image, confidence=0.5, nms_thresh=0.3):
    Height, Width = image.shape[:2]
    scale = 0.00392

    global classes
    global dest_dir
```

```
    # all the weights and the neural network algorithm are already
preconfigured
    # as we are using YOLO

    # this part of the script just downloads the YOLO files
    config_file_name = 'yolov3.cfg'
    config_file_abs_path = dest_dir + os.path.sep + config_file_name

    weights_file_name = 'yolov3.weights'
    weights_file_abs_path = dest_dir + os.path.sep + weights_file_name

    url =
'https://raw.githubusercontent.com/Reema1234ag/Drowning-Risk-Analysis/m
aster/yolov3.cfg'

    if not os.path.exists(config_file_abs_path):
        download_file(url=url, file_name=config_file_name,
dest_dir=dest_dir)

    url = 'https://pjreddie.com/media/files/yolov3.weights'

    if not os.path.exists(weights_file_abs_path):
        download_file(url=url, file_name=weights_file_name,
dest_dir=dest_dir)

    global initialize
    global net

    if initialize:
        classes = populate_class_labels()
        net = cv2.dnn.readNet(weights_file_abs_path,
config_file_abs_path)
        initialize = False

    blob = cv2.dnn.blobFromImage(image, scale, (416, 416), (0, 0, 0),
True, crop=False)

    net.setInput(blob)

    outs = net.forward(get_output_layers(net))

    class_ids = []
    confidences = []
```

```
    boxes = []

    for out in outs:
        for detection in out:
            scores = detection[5:]
            class_id = np.argmax(scores)
            max_conf = scores[class_id]
            if max_conf > confidence:
                center_x = int(detection[0] * Width)
                center_y = int(detection[1] * Height)
                w = int(detection[2] * Width)
                h = int(detection[3] * Height)
                x = center_x - w / 2
                y = center_y - h / 2
                class_ids.append(class_id)
                confidences.append(float(max_conf))
                boxes.append([x, y, w, h])

    indices = cv2.dnn.NMSBoxes(boxes, confidences, confidence,
nms_thresh)

    bbox = []
    label = []
    conf = []

    for i in indices:
        i = i[0]
        box = boxes[i]
        x = box[0]
        y = box[1]
        w = box[2]
        h = box[3]
        bbox.append([round(x), round(y), round(x + w), round(y + h)])
        label.append(str(classes[class_ids[i]]))
        conf.append(confidences[i])

    return bbox, label, conf
```

App.py

```
import cv2
import os
```

```
import numpy as np
from pathlib import Path
import cvlib as cv
import time
from cv2 import threshold
from cvlib.object_detection import draw_bbox
# from matplotlib.patches import draw_bbox

from flask import Flask , request, render_template , redirect , url_for

from playsound import alarm
# from utils import download_file

from cloudant.client import Cloudant

ACCOUNT_NAME,
API_KEY="33752a8cf8e04c5395279e7f558e0dd6","tFuhxJx262906XTTQZtS7SHvFtj
LKoFdxEpehJlUw1hg"
client=Cloudant.iam(ACCOUNT_NAME, API_KEY, connect=True)

my_database=client.create_database('my_database')
app=Flask(__name__)

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

@app.route('/index')
def home():
    return render_template('index.html')

@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],
        '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)
        return render_template('register.html',message='Registration
Successful, Please login using your details')
    else:
        return render_template('register.html',message="You are alredy
a member, please login using your details")
    return "nothing"


@app.route('/login')
def login():
    return render_template('login.html',message="")

@app.route('/afterlogin',methods=['POST'])
def afterlogin():
    x=[x for x in request.form.values()]
    user =x[0]
    passw=x[1]
    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):
        print("login")
        return render_template('login.html',message="The user is not
found")
    else:
```

```
        print("holaaaaaaaaaa")
        if((user==docs[0][0]['_id'] and passw==docs[0][0]['psw'])):
            return redirect(url_for('prediction'))
        else:
            print('Invalid User')
            # flash("invalid")
            return render_template('login.html',message="invalid
credentials")
    return "nothing"

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

# class dotdict(dict):
#     """dot.notation access to dictionary attributes"""
#     __getattr__ = dict.get
#     __setattr__ = dict.__setitem__
#     __delattr__ = dict.__delitem__

@app.route('/prediction')
def prediction():
    return render_template('prediction.html',prediction="Checking for
drowning")

def draww(frame,bbox,conf):
    for i in range(len(bbox)):
        print(conf)
        start_point = (bbox[i][0], bbox[i][1])
        end_point = (bbox[i][2], bbox[i][3])
        color = (255, 0, 0)
        thickness = 2
        frame = cv2.rectangle(frame, start_point, end_point, color,
thickness)
    return frame

@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()
center0=np.zeros(2)
isDrowning=False

while webcam.isOpened():
    status,frame=webcam.read()
    bbox,label,conf=cv.detect_common_objects(frame)
    print("seeeeeeee")
    print("-----------------------------------------------")
    print(bbox)
    print("-----------------------------------------------")
    if(len(bbox)>0):

        bbox0=bbox[0]

        center =[0,0]

        center=[(bbox0[0]+bbox0[2])/2,(bbox0[1]+bbox0[3])/2]

        hmov=abs(center[0]-center0[0])
        vmov= abs(center[1]-center0[1])

        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('bbox: ',bbox,'center:',center, 'center0:',center0 )
        print('Is he drowning: ',isDrowning)

        center0 =center

        # out=draw_bbox(frame,bbox,label,conf,isDrowning)

        # print(bbbox.x0)
        # out=draw_bbox(frame,bbbox,label,conf)
```

```
            # out=draw_bbox(bbox,frame)

            # frame=draww(frame,bbox,conf)
            # out=frame
            out= draw_bbox(frame, bbox, label, conf)
            cv2.imshow("Real-Time objects detection",out)
        else:
            out=frame
            cv2.imshow("Real-Time objects detection",out)
        # cv2.imshow("Real-Time objects detection",frame)
        if(isDrowning==True):
            audio =os.path.dirname(__file__)+"/sound.wav"
            alarm(audio)
            # playsound('alarm.mp3')
            webcam.release()
            cv2.destroyAllWindows()
            # return "nothing"
            return
render_template('prediction.html',prediction="Emergency !!! The Person
is drowning")

        if cv2.waitKey(1) & 0XFF == ord('q'):
            break

    webcam.release()
    cv2.destroyAllWindows()
    return render_template('prediction.html',prediction="Checking for
drowning")

if __name__ =='main':
    app.run(debug=True)
```