**PROJECT TITLE**

**VIRTUAL EYE** – LIFE GUARD FOR SWIMMING POOLS TO DETECT ACTIVE DROWNING.

**TEAM ID :** PNT2022TMID09949

**Source code :**

**Python :**

from flask import Flask  
from flask import flash, request, redirect, render\_template, url\_for  
from cloudant.client import Cloudant  
from cvlib.object\_detection import draw\_bbox  
  
import time  
import os  
import cvlib as cv  
import cv2  
import time  
import numpy as np  
  
app = Flask(\_\_name\_\_)  
client = Cloudant.iam(  
 'c56fd99d-acbb-4081-8351-ed1e1e82ba02-bluemix',  
 'RKYFESfjUudW-C8Pm-2WaAn3Q9N4Ud49q2PzIbVV4NdU',  
 connect=True)  
db = client['user\_details']  
  
  
@app.after\_request  
def add\_header(r):  
 *"""  
 Add headers to both force latest IE rendering engine or Chrome Frame,  
 and also to cache the rendered page for 10 minutes.  
 """* r.headers["Cache-Control"] = "no-cache, no-store, must-revalidate"  
 r.headers["Pragma"] = "no-cache"  
 r.headers["Expires"] = "0"  
 r.headers['Cache-Control'] = 'public, max-age=0'  
 return r  
  
  
@app.route("/")  
def home():  
 return render\_template('index.html')  
  
  
@app.route("/login", methods=["GET", "POST"])  
def login():  
 if request.method == "POST":  
 x = [x for x in request.form.values()]  
 data = {  
 '\_id': x[0],  
 'psw': x[1]  
 }  
 query = {'\_id': {'$eq': data['\_id']}}  
 docs = db.get\_query\_result(query)  
 if len(docs.all()) == 0:  
 db.create\_document(data)  
 return render\_template('login.html', title="VirtualEye - Login", status='NR')  
 else:  
 if x[0] == docs[0][0]['\_id'] and x[1] == docs[0][0]['psw']:  
 return redirect(url\_for('prediction'))  
 else:  
 return render\_template('login.html', title="VirtualEye - Login", status="Failed")  
 return render\_template('login.html', title="VirtualEye - Login")  
  
  
@app.route("/register", methods=['GET', 'POST'])  
def register():  
 if request.method == "POST":  
 x = [x for x in request.form.values()]  
 data = {  
 '\_id': x[1],  
 'name': x[0],  
 'psw': x[2]  
 }  
 query = {'\_id': {'$eq': data['\_id']}}  
 docs = db.get\_query\_result(query)  
 if len(docs.all()) == 0:  
 db.create\_document(data)  
 return render\_template('register.html', title='VirtualEye - Register', status='Success')  
 else:  
 return render\_template('register.html', title='VirtualEye - Register', status='Failed')  
 return render\_template('register.html', title='VirtualEye - Register')  
  
  
@app.route("/demo", methods=['GET'])  
def demo():  
 return render\_template('base.html', title="VirtualEye - Demo")  
  
  
@app.route("/forgotpassword")  
def forgotpass():  
 return render\_template('base.html', title="VirtualEye")  
  
  
@app.route("/logout")  
def logout():  
 return render\_template('logout.html', title="VirtualEye - Logged out")  
  
  
@app.route('/result')  
def prediction():  
 webcam = cv2.VideoCapture('drowning.mp4')  
  
 if not webcam.isOpened():  
 flash("Could not open webcam")  
 exit()  
  
 t0 = time.time()  
 centre0 = np.zeros(2)  
 isDrowning = False  
  
 while webcam.isOpened():  
 status, frame = webcam.read()  
  
 bbox, label, conf = cv.detect\_common\_objects(frame)  
  
 if len(bbox) > 0:  
 centre = [0, 0]  
 centre = [(bbox[0][0] + bbox[0][2]) / 2, (bbox[0][1] + bbox[0][3]) / 2]  
 hmov = abs(centre[0] - centre0[0])  
 vmov = abs(centre[1] - centre0[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:', centre, ' centre0:', centre0)  
 print('Are they drowning: ', isDrowning)  
  
 centre0 = centre  
  
 out = draw\_bbox(frame, bbox, label, conf)  
  
 cv2.imshow("Real-time object detection", out)  
  
 if isDrowning:  
 os.system("mpg123 -q alarm.mp3")  
 webcam.release()  
 cv2.destroyAllWindows()  
 return render\_template('prediction.html', prediction="Emergency!!! The Person is drowning",  
 title="VirtualEye - Prediction")  
  
 if cv2.waitKey(1) & 0xFF == ord('q'):  
 break  
  
 webcam.release()  
 cv2.destroyAllWindows()  
 return render\_template('prediction.html', title='VirtualEye - Prediction', prediction='Waiting for footage')  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(debug=True)

**\_init\_.py:**

from .object\_detection import detect\_common\_objects

**object detection.py :**

#import necessary packages

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

    url = 'https://github.com/Nico31415/Drowning-Detector/raw/master/yolov3.txt'

    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://github.com/Nico31415/Drowning-Detector/raw/master/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

**utils.py :**

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

**drowndetect.py :**

import cvlib as cv

from cvlib.object\_detection import draw\_bbox

import cv2

import time

import numpy as np

#for PiCamera

#from picamera Import PiCamera

#camera = PiCamera

#camera.start\_preview()

# open webcam

webcam = cv2.VideoCapture(0)

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)

    # press "Q" to stop

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

        break

# release resources

webcam.release()

cv2.destroyAllWindows()

**HTML code :**

**INDEX PAGE :**

<!-- NAVIGATION MENUS -->  
 <!DOCTYPE html>  
<html lang="en">  
 <head>  
 <meta charset="UTF-8" />  
 <meta http-equiv="X-UA-Compatible" content="IE=edge" />  
 <meta name="viewport" content="width=device-width, initial-scale=1.0" />  
 <style>  
 \* {  
 margin: 0;  
 padding: 0;  
 box-sizing: border-box;  
}  
body {  
 font-family: cursive;  
}  
a {  
 text-decoration: none;  
}  
li {  
 list-style: none;  
}   
.navbar {  
 display: flex;  
 align-items: center;  
 justify-content: space-between;  
 padding: 20px;  
 background-color: teal;  
 color: #fff;  
}  
.nav-links a {  
 color: #fff;  
}  
/\* LOGO \*/  
.logo {  
 font-size: 32px;  
}  
/\* NAVBAR MENU \*/  
.menu {  
 display: flex;  
 gap: 1em;  
 font-size: 18px;  
}  
.menu li:hover {  
 background-color: #4c9e9e;  
 border-radius: 5px;  
 transition: 0.3s ease;  
}  
.menu li {  
 padding: 5px 14px;  
}  
.services {  
 position: relative;  
}  
.dropdown {  
 background-color: rgb(1, 139, 139);  
 padding: 1em 0;  
 position: absolute; /\*WITH RESPECT TO PARENT\*/  
 display: none;  
 border-radius: 8px;  
 top: 35px;  
}  
.dropdown li + li {  
 margin-top: 10px;  
}  
.dropdown li {  
 padding: 0.5em 1em;  
 width: 8em;  
 text-align: center;  
}  
.dropdown li:hover {  
 background-color: #4c9e9e;  
}  
.services:hover .dropdown {  
 display: block;  
}  
#example1 {  
 background: url(swim.jpg);  
  
}  
#swim  
{  
 height: 200px;  
 width: 50%;  
}  
 </style>  
 <title>Document</title>  
 </head>  
 <body>  
 <nav class="navbar">  
 <!-- LOGO -->  
 <div class="logo">VIRTUAL EYE</div>  
 <!-- NAVIGATION MENU -->  
 <ul class="nav-links">  
 <!-- USING CHECKBOX HACK -->  
  
 <div class="menu">  
 <li><a href="/">Home</a></li>  
 <li><a href="/">About</a></li>  
 <li class="services">  
 <a href="/">Prediction</a>  
  
 </li>  
 <li><a href="/register">Register</a></li>  
 <li><a href="/login">Login</a></li>  
 </div>  
 </ul>  
 </nav>  
 <div class="swim">  
 <img style="height:670px; width:1520px"src="/static/swim.jpg">  
 <a href="/result"><button style = "position:absolute; right:60px; bottom:45px; height:40px; width:500px; color:cyan; background:black;">TRY THIS PROJECT IN DEMO VERSION (CLICK HERE)</button></a>  
 </div>  
  
 </body>  
</html>

**LOGIN PAGE :**

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>login</title>  
 <style>  
 \* {box-sizing: border-box}  
  
/\* Add padding to containers \*/  
.container {  
 padding: 16px;  
}  
  
/\* Full-width input fields \*/  
input[type=text], input[type=password] {  
 width: 100%;  
 padding: 15px;  
 margin: 5px 0 22px 0;  
 display: inline-block;  
 border: 1px solid;  
 background: #f1f1f1;  
}  
  
input[type=text]:focus, input[type=password]:focus {  
 background-color: #ddd;  
 outline: none;  
}  
  
/\* Overwrite default styles of hr \*/  
hr {  
 border: 1px solid #f1f1f1;  
 margin-bottom: 25px;  
}  
  
/\* Set a style for the submit/register button \*/  
.registerbtn {  
 background-color: #04AA6D;  
 color: baby blue;  
 padding: 16px 20px;  
 margin: 8px 0;  
 border: none;  
 cursor: pointer;  
 width: 100%;  
 opacity: 0.9;  
}  
  
.registerbtn:hover {  
 opacity:1;  
}  
  
/\* Add a blue text color to links \*/  
a {  
 color: aqua  
;  
}  
  
/\* Set a grey background color and center the text of the "sign in" section \*/  
.signin {  
 background-color: #89cfef;  
 text-align: center;  
}  
p {  
 background-image: url('img\_girl.jpg');  
}  
 </style>  
</head>  
<body>  
<form>  
 <div class="container">  
 <h1>LOGIN</h1>  
 <p>Login with your credentials.</p>  
 <hr>  
  
 <label for="email"><b>Email</b></label>  
 <input type="text" placeholder="Enter Email" name="email" id="email" required>  
  
 <label for="psw"><b>Password</b></label>  
 <input type="password" placeholder="Enter Password" name="psw" id="psw" required>  
  
  
 <p><a href="/index" class="registerbtn">LOGIN</a>.</p>  
 < p style="background-image: url('img\_girl.jpg');">  
 </div>  
  
  
</form>  
</body>  
</html>

**LOGOUT PAGE :**

<!DOCTYPE html>  
<html lang="en">  
  
<head>  
 <meta charset="UTF-8">  
 <meta http-equiv="X-UA-Compatible" content="IE=edge">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Virtual Eye - Home</title>  
 <script src="https://cdn.tailwindcss.com?plugins=forms,typography,aspect-ratio,line-clamp"></script>  
</head>  
  
<body>  
 <header>  
 <nav class="flex items-center justify-between flex-wrap bg-gray-700 p-6">  
 <div class="flex items-center flex-shrink-0 text-white mr-6 basis-4/5">  
 <span class="font-semibold text-xl tracking-tight">Virtual Eye</span>  
 </div>  
 <div class="w-full block flex-grow lg:flex lg:items-center lg:w-auto">  
 <div class="text-sm lg:flex-grow font-semibold">  
 <a href="index.html" class="block mt-4 lg:inline-block lg:mt-0 text-gray-400 hover:text-white mr-4">  
 Home  
 </a>  
 <a href="register.html"  
 class="block mt-4 lg:inline-block lg:mt-0 text-gray-400 hover:text-white mr-4">  
 Register  
 </a>  
 <a href="login.html" class="block mt-4 lg:inline-block lg:mt-0 text-gray-400 hover:text-white mr-4">  
 Login  
 </a>  
 <a href="demo.html" class="block mt-4 lg:inline-block lg:mt-0 text-gray-400 hover:text-white mr-4">  
 Demo  
 </a>  
 </div>  
 </div>  
 </nav>  
 </header>  
  
 <div class="flex-col items-center">  
 <p class="font-semibold text-2xl m-8" style="text-align: center">Successfully Logged Out!</p>  
 <p class="font-semibold text-l mb-4 text-green-500" style="text-align: center">Login for more information</p>  
 <button  
 class="block m-auto bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded-full">Login</button>  
 </div>  
  
</body>  
  
</html>

**PREDICTION PAGE :**

<!DOCTYPE html>  
<html lang="en">  
  
<head>  
 <meta charset="UTF-8">  
 <meta http-equiv="X-UA-Compatible" content="IE=edge">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Virtual Eye - Home</title>  
 <script src="https://cdn.tailwindcss.com?plugins=forms,typography,aspect-ratio,line-clamp"></script>  
</head>  
  
<body>  
 <header>  
 <nav class="flex items-center justify-between flex-wrap bg-gray-700 p-6">  
 <div class="flex items-center flex-shrink-0 text-white mr-6 basis-4/5">  
 <span class="font-semibold text-xl tracking-tight">Virtual Eye</span>  
 </div>  
 <div class="w-full block flex-grow lg:flex lg:items-center lg:w-auto">  
 <div class="text-sm lg:flex-grow font-semibold">  
 <a href="index.html" class="block mt-4 lg:inline-block lg:mt-0 text-gray-400 hover:text-white mr-4">  
 Home  
 </a>  
 <a href="logout.html"  
 class="block mt-4 lg:inline-block lg:mt-0 text-gray-400 hover:text-white mr-4">  
 Logout  
 </a>  
 </div>  
 </div>  
 </nav>  
 </header>  
  
 <p class="font-bold text-3xl px-8 pt-8 underline" style="text-align: center">VirtualEye - Life Guard for Swimming  
 Pools to Detect Active Drowning</p>  
 <div class="grid grid-cols-2">  
 <div class="flex-col bg-white shadow-md rounded px-8 pt-10 pb-8 m-8 mr-4">  
 <p class="m-auto text-xl font-semibold mb-4">Problem</p>  
 <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. To overcome this conflict, a  
 meticulous system is to be implemented along the swimming pools to save human life.  
 </p>  
  
 </div>  
 <div class="flex-col bg-white shadow-md rounded px-8 pt-10 pb-8 m-8 ml-4">  
 <img class="h-56 m-auto" src="static\_images/background.jpg">  
 <button  
 class="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded-full block m-auto mt-4 mb-4">Click  
 me! For a demo</button>  
 <p style="text-align: center">{{ Prediction }}</p>  
 </div>  
 </div>  
  
</body>  
  
</html>

**REGISTER PAGE :**

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title> Register </title>  
 <style>  
 \* {box-sizing: border-box}  
  
  
.container {  
 padding: 16px;  
}  
  
  
input[type=text], input[type=password] {  
 width: 100%;  
 padding: 15px;  
 margin: 5px 0 22px 0;  
 display: inline-block;  
 border: none;  
 background: #f1f1f1;  
}  
  
input[type=text]:focus, input[type=password]:focus {  
 background-color: #ddd;  
 outline: none;  
}  
  
hr {  
 border: 1px solid #f1f1f1;  
 margin-bottom: 25px;  
}  
  
.registerbtn {  
 background-color: #04AA6D;  
 color: white;  
 padding: 16px 20px;  
 margin: 8px 0;  
 border: none;  
 cursor: pointer;  
 width: 100%;  
 opacity: 0.9;  
}  
  
.registerbtn:hover {  
 opacity:1;  
}  
  
  
a {  
 color: red;  
}  
  
.signin {  
 background-color: #f1f1f1;  
 text-align: center;  
}  
  
 </style>  
</head>  
<body>  
<form>  
 <div class="container">  
 <h1>Register</h1>  
 <p>Please fill in this form to create an account.</p>  
 <hr>  
  
 <label for="email"><b>Email</b></label>  
 <input type="text" placeholder="Enter Email" name="email" id="email" required>  
  
 <label for="psw"><b>Password</b></label>  
 <input type="password" placeholder="Enter Password" name="psw" id="psw" required>  
  
 <label for="psw-repeat"><b>Repeat Password</b></label>  
 <input type="password" placeholder="Reenter Password" name="psw-repeat" id="psw-repeat" required>  
 <hr>  
  
  
 <p><a href="/index" class="registerbtn">Register</a>.</p>  
 </div>  
  
 <div class="container signin">  
 <p>Already a member? <a href="/login">Sign in</a>.</p>  
 </div>  
</form>  
</body>  
</html>

**RESULT PAGE :**

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
</head>  
<body>  
<h1>result</h1>  
</body>  
</html>