**Face Recognition for Patient Record(Coding)**

# JavaScript

let menu=document.querySelector(‘.menu-btu’);

let navbar=document.querySelctor(‘.navbar’)

menu.onclick=()=>{

menu.classList.toggle(‘fa-times’);

navbar.cassList.toggle(‘active’);

}

Window.onscroll=()=>{

Menu.classList.remove(‘fa-times’);

Navbar.classList.remove(‘active’);

}

# Html

@import url('<https://fonts.googleapis.com/css2?family=Poppins:ital,wght@0,100;0,400;0,600;0,700;1,300&display=swap>');

:root{

    --violet:#d056f1;

    --black:#444;

    --light-color:#777;

    --box-shadow:.5rem .5rem 0 rgba(22,160,133,.2);

    --box-shadow:.4rem .4rem 0 rgba(0,0,0,.2);

    --border:.2rem solid var(--violet);

}

\*{

    font-family: 'Poppins', sans-serif;

    margin:0;

    padding: 0;

    box-sizing: border-box;

    outline: none;

    border: none;

    text-transform: capitalize;

    transition: all .2s ease-out;

    text-decoration: none;

}

/\*svg img h3 ,p\*/

section{

    padding:2rem 9%;

}

/\*bed,available category starts\*/

section:nth-child(even){

    background: #f5f5f5;

}

/\*our services starts\*/

.heading{

    text-align: center;

    padding-bottom: 2rem;

    text-shadow: var(--text-shadow);

    text-transform: uppercase;

    color: var(--black);

    font-size: 5rem;

    letter-spacing: .4rem;

}

.heading span{

    text-transform: uppercase;

    color: var(--violet);

}

/\*our services ends\*/

/\*bed,available category ends\*/

.btn{

    display: inline-block;

    margin-top: 1rem;

    padding: .5rem;

    padding-left: 1rem;

    border:var(--border);

    border-radius: .5rem;

    box-shadow: var(--box-shadow);

    color:var(--violet);

    cursor:pointer;

    font-size: 1.7rem;

}

.btn span{

    padding: .7rem 1rem;

    border-radius: .5rem;

    background: var(--violet); /\*contact us button\*/

    color:#fff;

    margin-left: .5rem;

}

.btn:hover{

    background-color: var(--violet);

    color: #fff;

}

.btn:hover span{

    color: var(--violet);

    background-color: #fff;

    margin-left: 1rem;

}

/\*ends\*/

html{

    font-size: 62.5%;

    overflow-x: hidden;

    scroll-padding-top: 7rem;

    scroll-behavior: smooth;

}

.header{

    padding:.2rem 9%;

    position: fixed;

    top:0;left:0;right: 0;

    z-index: 1000;

    box-shadow: 0.5rem 1.5rem rgba(0,0,0,.2);

    display: flex;

    align-items: center;

    justify-content: space-between;

    background: #fff;

}

.header .logo{

    font-size: 3.5rem;

    color:var(--black)

}

.header .logo i{

    color:var(--violet);

}

.header .navbar a{

    font-size: 1.7rem;

    color:var(--light-color);

    margin-left: 2rem;

}

.header .navbar a:hover{

    color: var(--violet);

}

.menu-btn{

    font-size: 2.5rem;

    border-radius: .5rem;

    background: #eee;

    color:var(--violet);

    padding :1rem 1.5rem;

    cursor: pointer;

    display: none;

}

/\* img display 1\*/

.home{

    display: flex;

    align-items:center; /\*stsy...,h3,p,contactus in center css code\*/

    flex-wrap: wrap;

    gap:1.5rem;

    padding-top: 10rem;

}

.home .image{

    flex:1 1 45rem;

}

.home .image img{

    width:100%;

}

.home .content{

    flex:1 1 45rem;

}

.home .content h3{

    font-size: 4.5rem;

    color:var(--black);

    line-height:1.8;

    text-shadow: var(--text-shadow);

}

.home .content p{

    font-size: 1.7rem;

    color:var(--light-color);

    line-height:1.8;

    padding: 1rem 0;

}

/\* bed,avaliable category css coding starts\*/

.icons-container{

    display: grid;

    gap: 2rem;

    grid-template-columns: repeat(auto-fit,minmax(20rem,1fr));

    padding-top: 1rem;

    padding-bottom: 10rem;

}

.icons-container .icons{

    border:var(--border);

    box-shadow: var(--box-shadow);

    border-radius: .3rem;

    text-align: center;

    padding: 2.5rem;

}

.icons-container .icons i{

    font-size: 3.5rem;

    color:var(--violet);

    padding-bottom: .7rem;

}

.icons-container .icons h3{

    font-size: 4.5rem;

    color:var(--black);

    padding-bottom: .5rem 0;

    text-shadow: var(--text-shadow);

}

.icons-container .icons p{

    font-size: 1.7rem;

    color:var(--light-color);

}

/\* bed,avaliable category css coding ends\*/

/\*our services start\*/

.services .box-container{

    display:grid;

    grid-template-columns: repeat(auto-fit,minmax(27rem,1fr));

    gap:2rem;

}

.services .box-container .box{

    background: #fff;

    border-radius: 0.9rem;

    box-shadow: var(--box-shadow);

    border: var(--border);

    padding: 2.5rem;

}

.services .box-container .box i{

    color: var(--violet);

    font-size: 3rem;

    padding-bottom: .4rem;

}

.services .box-container .box h3{

    color: var(--black);

    font-size: 1.9rem;

    padding: 1rem 0;

}

.services .box-container .box p{

    color: var(--light-color);

    font-size: 1.1rem;

    line-height: 2;

}

/\*our services end\*/

/\*about section start about us\*/

.about .row{

    display:flex;

    align-items:center ;

    flex-wrap: wrap;

    gap: 2rem;

}

.about .row .image{

    flex:1 1 45rem;

}

.about .row .image img{

    width:100%;

}

.about .row .content{

    flex:1 1 45rem;

}

.about .row .content h3{

    color: var(--black);

    text-shadow: var(--text-shadow);

    font-size: 3rem;

    line-height: 1.8;

}

.about .row .content p{

    color: var(--light-color);

    padding: 1rem 0;

    font-size: 1.5rem;

    line-height: 1.8;

}

/\*about section end about us\*/

/\*doctors section starts\*/

.doctors .box-container {

    display:grid;

    grid-template-columns: repeat(auto-fit, minmax(27rem,1fr));

    gap:2rem;

}

.doctors .box-container .box{

    text-align: center;

    background: #fff;

    border-radius: .5rem;

    border:var(--border);

    box-shadow: var(--box-shadow);

    padding: 2rem;

}

.doctors .box-container .box img{

    height: 20rem;

    border: var(--border);

    border-radius: .5rem;

    margin-top: 1rem;

    margin-bottom: 1rem;

}

.doctors .box-container .box h3{

    color:var(--black);

    font-size: 2.5rem;

}

.doctors .box-container .box span{

    color:var(--violet);

    font-size: 1.5rem;

}

.doctors .box-container .box .share{

    padding-top:2rem;

}

.doctors .box-container .box .share a{

    height: 5rem;

    width: 5rem;

    line-height: 4.5rem;

    font-size: 2rem;

    color:var(--violet);

    border-radius: .5rem;

    border: var(--border);

    margin:.3rem;

}

.doctors .box-container .box .share a:hover{

    background: var(--violet);

    color: #fff;

    box-shadow: var(--box-shadow);

}

/\*doctors section end \*/

/\*img display end\*/

/\*review start\*/

.review .box-container{

    display: grid;

    grid-template-columns: repeat(auto-fit,minmax(27rem,1fr));

    gap: 2rem;

}

.review .box-container .box{

    border: var(--border);

    box-shadow: var(--box-shadow);

    border-radius: .5rem;

    padding: 2.5rem;

    background: #fff;

    text-align: center;

    position: relative;

    overflow: hidden;

    z-index: 0;

}

.review .box-container .box img{

    height: 10rem;

    width: 10rem;

    border-radius: 50%;

    object-fit: cover;

    border: .5rem solid #fff;

}

.review .box-container .box h3{

    color: #fff;

    font-size: 2.2rem;

    padding: .5rem 0;

}

.review .box-container .box .stars i{

    color: #fff;

    font-size: 1.5rem;

}

.review .box-container .box .text{

    color: var(--light-color);

    line-height: 1.8;

    font-size: 1.6rem;

    padding-top: 4rem;

}

.review .box-container .box::before{

    content: '';

    position: absolute;

    top:-4rem; left: 50%;

    transform: translateX(-50%);

    background: var(--violet);

    border-bottom-left-radius: 50%;

    border-bottom-right-radius: 50%;

    height: 25rem;

    width: 120%;

    z-index: -1;

}

/\*review ends\*/

/\*footer section start\*/

.footer .box-container{

    display: grid;

    grid-template-columns: repeat(auto-fit, minmax(22rem,1fr));

    gap: 2rem;

}

.footer .box-container .box h3{

    font-size: 2.5rem;

    color:var(--black);

    padding: 1rem 0;

}

.footer .box-container .box a{

    display: block;

    font-size: 1.5rem;

    color:var(--light-color);

    padding: 1rem 0;

}

.footer .box-container .box a i{

    padding-right: .5rem;

    color: var(--violet);

}

.footer .box-container .box a:hover i{

    padding-right: 2rem;

}

.footer .credit{

    padding:1rem;

    padding-top: 2rem;

    margin-top: 2rem;

    text-align: center;

    font-size: 2rem;

    color: var(--light-color);

    border-top: .1rem solid rgba(0,0, 0,.1);

}

.footer .credit span{

    color:var(--violet)

}

/\*footer section ends\*/

/\* media queries \*/

@media(max-width:991px){

    html{

        font-size: 55%;

    }

    .header{

        padding:2rem;

    }

}

@media(max-width:768px) {

    .menu-btn{

        display: initial;

    }

    .header .navbar{

        position:absolute;

        top: 115%;right: 2rem;

        border-radius: .5rem;

        box-shadow: var(--box-shadow);

        width:30rem;

        border:var(--border);

        background: #fff;

        transform: scale(0);

        opacity: 0;

        transform-origin: top right;

        transition: none;

    }

    .header .navbar .active{

        transform: scale(0);

        opacity: 1;

        transition: .2s ease-out;

    }

    .header .navbar a{

        font-size: 2rem;

        display: block;

        margin: 2.5rem;

    }

}

@media(max-width:450px){

    html{

        font-size: 50%;

    }

dy{

    background-color: url(image.png);

    width: 100px;

    height: 100vh;

    font-family: sans-serif;

}

.signup{

    width: 360px;

    height: 800px;

    margin: auto;

    background: #fff;

    border-radius: 3px;

}

h1{

    text-align: center;

}

form{

    width: 300px;

    margin-left: 20px;

}

form label{

    display: flex;

    margin-top: 20px;

    font-size: 18px;

}

form input{

    width: 100%;

    padding: 7px;

    border: none;

    border: 1px solid gray;

    border-radius: 6px;

    outline: none;

}

input[type="Submit"]{

    width: 320px;

    height: 35px;

    margin-top: 20px;

    border: none;

    background-color: #ff7200;

    color: #fff;

    font-size: 18px;

    cursor: pointer;

}

}

# Python

from flask import Flask, render\_template, request, redirect, url\_for,jsonify

import cv2

import os

import json

import numpy as np

import csv

import pandas as pd

from PIL import Image, ImageFont, ImageDraw

import datetime

from flask\_cors import CORS

import time

import sqlite3

app = Flask(\_\_name\_\_)

CORS(app)

ALLOWED\_EXTENSIONS = {'png', 'jpg', 'jpeg'}

REGISTER\_PATH="./Register/"

TEMP\_PATH="./Capturing\_Images"

upload\_path='./upload\_face'

upload\_path2='./upload\_face2'

app.config['SEND\_FILE\_MAX\_AGE\_DEFAULT']=1

def allowed\_file(filename):

    return '.' in filename and \

           filename.rsplit('.', 1)[1].lower() in ALLOWED\_EXTENSIONS

conn = sqlite3.connect('db.sqlite3')

c = conn.cursor()

#alter\_query = '''ALTER TABLE users

                 #ADD COLUMN ph TEXT'''

#c.execute(alter\_query)

c.execute('''CREATE TABLE IF NOT EXISTS srm (

                name TEXT,

                Id TEXT,

                Gender TEXT,

                phonenumber TEXT,

                address TEXT,

                Email TEXT

            )''')

###c.execute('''(DESC users)''')

conn.commit()

conn.close()

@app.route('/patientreg', methods=['GET', 'POST'])

def upload\_register():

    if request.method == 'POST':

        if 'file' not in request.files:

            return json.dumps({"status": "Error", "msg": "Image cannot be empty "})

    return "Registration successful!"

def is\_number(s):

    try:

        float(s)

        return True

    except ValueError:

        pass

    try:

        import unicodedata

        unicodedata.numeric(s)

        return True

    except (TypeError, ValueError):

        pass

    return False

@app.route('/')

def index():

    return render\_template('frontpage.html')

@app.route('/login', methods=['GET', 'POST'])

def login():

    email = request.form.get('email')

    password = request.form.get('password')

    print(email.lower())

    print(password)

    if (email.lower() == '[srm@gmail.com](mailto:srm@gmail.com)' and password == 'srm'):

        return json.dumps ({"status": "true","message": "User Registered Succesfully"})

    else:

        return json.dumps ({"status": "false","message": "User Not Registered"})

@app.route('/take\_image', methods=['GET', 'POST'])

def TakeImages():

    name = request.form.get('name')

    print(name)

    Id = request.form.get('Id')

    Gender = request.form.get('gender')

    phonenumber = request.form.get('phonenumber')

    address = request.form.get('address')

    Email = request.form.get('email')

    print(Id)

    print(Gender)

    print(phonenumber)

    print(address)

    print(Email)

    conn = sqlite3.connect('db.sqlite3')

    c = conn.cursor()

        #c.execute("INSERT INTO users (username, email, password) VALUES (?, ?, ?)", (username, email, password))

    c.execute("INSERT INTO srm (name, Id, Gender, phonenumber, address, Email) VALUES (?, ?, ?, ?, ?, ?)",

              (name, Id, Gender, phonenumber, address, Email))

    conn.commit()

    conn.close()

    #Id='4123'

    #name='ghhwewewe'

    age='34'

    #gender='m'

    if(is\_number(Id) and name.isalpha()):

        cam = cv2.VideoCapture(0)

        harcascadePath = "haarcascade\_frontalface\_default.xml"

        detector=cv2.CascadeClassifier(harcascadePath)

        sampleNum=0

        print("inside if cond")

        while(True):

            ret, img = cam.read()

            gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

            faces = detector.detectMultiScale(gray, 1.3, 5)

            for (x,y,w,h) in faces:

                cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)

                #incrementing sample number

                sampleNum=sampleNum+1

                #saving the captured face in the dataset folder TrainingImage

                cv2.imwrite("Capturing\_Images\ "+name +"."+Id +'.'+ str(sampleNum) + ".jpg", gray[y:y+h,x:x+w])

                #display the frame

                cv2.imshow('frame',img)

            #wait for 100 miliseconds

            if cv2.waitKey(100) & 0xFF == ord('q'):

                break

            # break if the sample number is morethan 100

            elif sampleNum>60:

                break

        cam.release()

        cv2.destroyAllWindows()

        res = "Images Saved for ID : " + Id +" Name : "+ name +"Age :" + age + "Gender:" +Gender

        row = [Id , name,age,Gender]

        with open('patient\_List\patient\_List.csv','a+') as csvFile:

            writer = csv.writer(csvFile)

            writer.writerow(row)

        csvFile.close()

        #message.configure(text= res)

    else:

        if(is\_number(Id)):

            res = "Enter Alphabetical Name"

            message.configure(text= res)

        if(name.isalpha()):

            res = "Enter Numeric Id"

            message.configure(text= res)

    recognizer = cv2.face\_LBPHFaceRecognizer.create()#recognizer = cv2.face.LBPHFaceRecognizer\_create()#$cv2.createLBPHFaceRecognizer()

    harcascadePath = "haarcascade\_frontalface\_default.xml"

    detector =cv2.CascadeClassifier(harcascadePath)

    faces,Id = getImagesAndLabels("Capturing\_Images")

    recognizer.train(faces, np.array(Id))

    recognizer.save("Models\Trainner.yml")

   # return json.dumps ({"status": "true","message": "User Registered Succesfully"})

    #res = "Image Trained"#+",".join(str(f) for f in Id)

    #message.configure(text= res)

def getImagesAndLabels(path):

    #get the path of all the files in the folder

    imagePaths=[os.path.join(path,f) for f in os.listdir(path)]

    #print(imagePaths)

    #print(imagePaths)

    #create empth face list

    faces=[]

    #create empty ID list

    Ids=[]

    #now looping through all the image paths and loading the Ids and the images

    for imagePath in imagePaths:

        #loading the image and converting it to gray scale

        pilImage=Image.open(imagePath).convert('L')

        #Now we are converting the PIL image into numpy array

        imageNp=np.array(pilImage,'uint8')

        #getting the Id from the image

        Id=int(os.path.split(imagePath)[-1].split(".")[1])

        # extract the face from the training image sample

        faces.append(imageNp)

        Ids.append(Id)

    return faces,Ids

# ... (previous code)

@app.route('/track\_images', methods=['GET', 'POST'])

#@app.route('/track\_images')

def TrackImages():

    recognizer = cv2.face.LBPHFaceRecognizer\_create()#cv2.createLBPHFaceRecognizer()

    recognizer.read("Models\Trainner.yml")

    harcascadePath = "haarcascade\_frontalface\_default.xml"

    faceCascade = cv2.CascadeClassifier(harcascadePath);

    df=pd.read\_csv("patient\_List\patient\_List.csv")

    cam = cv2.VideoCapture(0)

    font = cv2.FONT\_HERSHEY\_SIMPLEX

    col\_names =  ['Id','Name','Date','Time','Location']

    attendance = pd.DataFrame(columns = col\_names)

    while True:

        ret, im =cam.read()

        gray=cv2.cvtColor(im,cv2.COLOR\_BGR2GRAY)

        faces=faceCascade.detectMultiScale(gray, 1.2,5)

        for(x,y,w,h) in faces:

            cv2.rectangle(im,(x,y),(x+w,y+h),(225,0,0),2)

            Id, conf = recognizer.predict(gray[y:y+h,x:x+w])

            print(conf)

            if(conf < 50):

                Location="College"

                ts = time.time()

                date = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d')

                timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')

                aa=df.loc[df['Id'] == Id]['Name'].values

                tt=str(Id)+"-"+aa+"-"+"Student"

                name=""

                phonenumber=""

                gender=""

                address=""

                email=""

                attendance.loc[len(attendance)] = [Id,aa,date,timeStamp,Location]

                #break

                con = sqlite3.connect('db.sqlite3')

                #Id =4123

                #completion = False

                with con:

                    cur = con.cursor()

                    cur.execute("SELECT \* FROM srm WHERE Id = ?",(Id,))

                    #cur.execute("SELECT \* FROM srm WHERE Id == :Id", {"Id": str(Id)})

                    rows = cur.fetchone()

                    #for row in rows:

                    if rows:

                        print(rows)

                        name = rows[0]

                        gender