**Codes from the Flask App**

**Templates (HTML files)**

Signup Page(singup.html)

<!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">

<link rel="stylesheet" href="{{ url\_for('static', filename='auth\_style.css') }}">

<title>Sign Up</title>

</head>

<body>

<!-- this block checks for any feedback Flash messages from the server -->

{% with messages = get\_flashed\_messages() %}

{% if messages %}

<div class="flash-messages">

{% for message in messages %}

{% if 'success' in message %}

<div class="flash-message success">{{ message }}</div>

{% else %}

<div class="flash-message">{{ message }}</div>

{% endif %}

{% endfor %}

</div>

{% endif %}

{% endwith %}

<div class= "container">

<form id="signup-form" action="/signup" method="post">

<!-- title -->

<h2>Sign Up</h2>

<!-- input fields -->

<label for="employee\_id">Employee ID:</label>

<input type="text" id="employee\_id" name="employee\_id" required>

<label for="first\_name">First Name:</label>

<input type="text" id="first\_name" name="first\_name" required>

<label for="last\_name">Last Name:</label>

<input type="text" id="last\_name" name="last\_name" required>

<label for="email">Email:</label>

<input type="email" id="email" name="email" required>

<label for="password">Password:</label>

<input type="password" id="password" name="password" required>

<label for="repeat\_password">Repeat Password:</label>

<input type="password" id="repeat\_password" name="repeat\_password" required>

<!-- submit button -->

<button type="submit">Sign Up</button>

</form>

<p>Already have an account? <a href="{{ url\_for('login') }}">Log in here</a>.</p>

</div>

</body>

</html>

Login Page(login.html)

<!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">

<link rel="stylesheet" href="{{ url\_for('static', filename='auth\_style.css') }}">

<title>Login</title>

</head>

<body>

<!-- this block checks for any feedback Flash messages from the server -->

{% with messages = get\_flashed\_messages() %}

{% if messages %}

<div class="flash-messages">

{% for message in messages %}

{% if 'success' in message %}

<div class="flash-message success">{{ message }}</div>

{% else %}

<div class="flash-message">{{ message }}</div>

{% endif %}

{% endfor %}

</div>

{% endif %}

{% endwith %}

<div class="container">

<form id="login-form" action="/login" method="post">

<!-- title -->

<h2>Login</h2>

<!-- input fields -->

<label for="employee\_id">Employee ID:</label>

<input type="text" id="employee\_id" name="employee\_id" required>

<label for="password">Password:</label>

<input type="password" id="password" name="password" required>

<button type="submit">Login</button>

</form>

<p>Don't have an account? <a href="{{ url\_for('signup') }}">Signup here</a>.</p>

</div>

</body>

</html>

Main Prediction Page(main\_page.html)

<!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">

<link rel="stylesheet" href="{{ url\_for('static', filename='styles.css') }}">

<title>Breast Cancer Detection</title>

<script src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script>

<script src="{{ url\_for('static', filename='inactivity.js') }}"></script>

<script>

function toggleMenu() {

var sidebar = document.getElementById("sidebar");

sidebar.style.display = (sidebar.style.display === "none" || sidebar.style.display === "") ? "block" : "none";

}

</script>

</head>

<body>

<header>

<div class="burger-menu" onclick="toggleMenu()">

<div class="bar"></div>

<div class="bar"></div>

<div class="bar"></div>

</div>

<h1>Breast Cancer Detection</h1>

<div class="header-buttons">

<a href="{{ url\_for('logout') }}" class="logout-button">Logout</a>

</div>

</header>

<div class="container">

<div class="sidebar" id="sidebar">

<a href="{{ url\_for('insights') }}">View Insights</a>

<a href="{{ url\_for('patients') }}">Patients Reports</a>

<a href="{{ url\_for('multiple\_uploads') }}">Multiple Uploads</a>

</div>

<div class="main-content">

<h2>Patient Details</h2>

<form id="patient-form">

<div class="input-group">

<label for="patientID">Patient ID:</label>

<input type="text" id="patientID" name="patientID" required>

</div>

<div class="input-group">

<label for="firstName">First Name:</label>

<input type="text" id="firstName" name="firstName" required>

</div>

<div class="input-group">

<label for="lastName">Last Name:</label>

<input type="text" id="lastName" name="lastName" required>

</div>

<div class="input-group">

<label for="phoneNumber">Phone Number:</label>

<input type="tel" id="phoneNumber" name="phoneNumber" required>

</div>

<div class="input-group">

<label for="address">Address:</label>

<input type="text" id="address" name="address" required>

</div>

<div class="input-group">

<label for="dob">Date of Birth:</label>

<input type="date" id="dob" name="dob" required>

</div>

<div class="input-group">

<label for="ethnicity">Ethnicity:</label>

<input type="text" id="ethnicity" name="ethnicity" required>

</div>

<div class="input-group">

<label for="smoking">Smoking:</label>

<input type="checkbox" id="smoking" name="smoking">

</div>

<div class="input-group">

<label for="drinking">Drinking Alcohol:</label>

<input type="checkbox" id="drinking" name="drinking">

</div>

<div class="input-group">

<label for="occupation">Occupation:</label>

<input type="text" id="occupation" name="occupation" required>

</div>

<div class="input-group">

<label for="personalHistory">Personal History of Breast Cancer:</label>

<input type="checkbox" id="personalHistory" name="personalHistory">

</div>

<div class="input-group">

<label for="familyHistory">Family History of Breast Cancer:</label>

<input type="checkbox" id="familyHistory" name="familyHistory">

</div>

<div class="input-group">

<label for="mammogramImage">Upload Mammogram Image:</label>

<div class="file-upload">

<input type="file" id="mammogramImage" name="file" accept="image/\*" onchange="previewImage(this)" required>

<img id="imagePreview" alt="Mammogram Preview" style="max-width: 100%; margin-top: 10px; display: none;">

</div>

</div>

<button type="button" class="predict-button" onclick="submitForm()">Predict</button>

</form>

<div id="result-container" style="display: none">

<h2>Result</h2>

<p id="predictionResult">

{% if prediction %}

{{ prediction.class }} ({{ prediction.confidence \* 100 }}% confidence)

{% endif %}

</p>

</div>

</div>

</div>

<!-- preview img, file upload and form submission -->

<script>

function previewImage(input) {

var imagePreview = document.getElementById("imagePreview");

var fileInput = input;

if (fileInput.files && fileInput.files[0]) {

var reader = new FileReader();

reader.onload = function (e) {

imagePreview.src = e.target.result;

imagePreview.style.display = "block";

};

reader.readAsDataURL(fileInput.files[0]);

}

}

function submitForm() {

//validation of input fields

//patient ID cannot be empty

var patientID = document.getElementById('patientID').value;

if (patientID.trim() === '') {

alert('Please enter Patient ID.');

return;

}

//first and last name cannot be empty

var firstName = document.getElementById('firstName').value;

if (firstName.trim() === '') {

alert('Please enter First Name.');

return;

}

var lastName = document.getElementById('lastName').value;

if (lastName.trim() === '') {

alert('Please enter Last Name.');

return;

}

//phone number cannot be empty and cannot be>8 digits

var phoneNumber = document.getElementById('phoneNumber').value;

if (phoneNumber.trim() === '' || isNaN(phoneNumber) || phoneNumber.length > 8) {

alert('Please enter a valid Phone Number (maximum 8 digits).');

return;

}

// Address cannot be empty

var address = document.getElementById('address').value;

if (address.trim() === '') {

alert('Please enter an Address.');

return;

}

//dob cannot be empty or greater than today's date

var dob = document.getElementById('dob').value;

var today = new Date();

var selectedDate = new Date(dob);

if (selectedDate > today) {

alert('Please enter a valid Date of Birth.');

return;

}

if (dob.trim() === '') {

alert('Please enter Date of Birth.');

return;

}

// Ethnicity cannot be empty

var ethnicity = document.getElementById('ethnicity').value;

if (ethnicity.trim() === '') {

alert('Please enter Ethnicity.');

return;

}

// Occupation cannot be empty

var occupation = document.getElementById('occupation').value;

if (occupation.trim() === '') {

alert('Please enter Occupation.');

return;

}

//must select image

var fileInput = document.getElementById('mammogramImage');

if (!fileInput.files || fileInput.files.length === 0) {

alert('Please select an image for prediction.');

return;

}

var formData = new FormData();

// Append patient details to FormData

formData.append('patientID', document.getElementById('patientID').value);

formData.append('firstName', document.getElementById('firstName').value);

formData.append('lastName', document.getElementById('lastName').value);

formData.append('phoneNumber', document.getElementById('phoneNumber').value);

formData.append('address', document.getElementById('address').value);

formData.append('dob', document.getElementById('dob').value);

formData.append('ethnicity', document.getElementById('ethnicity').value);

formData.append('smoking', document.getElementById('smoking').checked ? 'on' : 'off');

formData.append('drinking', document.getElementById('drinking').checked ? 'on' : 'off');

formData.append('occupation', document.getElementById('occupation').value);

formData.append('personalHistory', document.getElementById('personalHistory').checked ? 'on' : 'off');

formData.append('familyHistory', document.getElementById('familyHistory').checked ? 'on' : 'off');

formData.append('file', fileInput.files[0]);

// Display privacy reminder -using sweet alert library

const privacyReminder = `

As a dedicated healthcare professional, entrusted with the well-being of patients,

it is imperative to uphold the highest ethical standards and safeguard patient privacy.

Ensure that patient data is handled with utmost confidentiality and used solely for

diagnostic and treatment purposes. By clicking 'OK,' you affirm your commitment to

maintaining the ethical integrity of medical practice.

`;

Swal.fire({

title: 'Privacy Reminder',

html: privacyReminder,

icon: 'info',

confirmButtonText: 'OK',

});

//submit form

fetch('/predict', {

method: 'POST',

body: formData

})

.then(response => {

if (!response.ok) {

throw new Error('Network response was not ok');

}

const contentType = response.headers.get('content-type');

if (contentType && contentType.includes('application/json')) {

return response.json();

} else {

throw new Error('Response is not in JSON format');

}

})

.then(data => {

if ('error' in data) {

// Display the error message

alert(data.error);

} else {

// successful response

console.log('Response:', data); //for debugging

document.getElementById('result-container').style.display = 'block';

// Round the confidence value to 2 decimal places

var confidence = (data.confidence \* 100).toFixed(2);

document.getElementById('predictionResult').innerText = 'Result: ' + data.class + ' (' + confidence + '% confidence)';

}

})

.catch(error => {

console.error('Error:', error);

});

}

</script>

</body>

</html>

Patient Reports Page(patients.html)

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Patients</title>

<style>

body {

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

background-color: #fff5f8;

margin: 20px;

}

h1 {

color: #e44d88;

}

table {

width: 100%;

border-collapse: collapse;

margin-top: 20px;

}

th, td {

padding: 12px;

text-align: left;

border: 1px solid #ff80ab;

}

th {

background-color: #e44d88;

color: white;

}

tr:nth-child(even) {

background-color: #ffd4e5;

}

#goHome {

margin-top: 20px;

}

#goHome a {

display: inline-block;

padding: 10px 20px;

background-color: #e44d88;

color: white;

text-decoration: none;

border-radius: 5px;

transition: background-color 0.3s ease;

}

#goHome a:hover {

background-color: #ff80ab;

}

/\* for report link \*/

td a {

display: inline-block;

padding: 5px 10px;

background-color: #4CAF50;

color: white;

text-decoration: none;

border-radius: 5px;

}

td a:hover {

background-color: #45a049;

}

</style>

<script src="{{ url\_for('static', filename='inactivity.js') }}"></script>

</head>

<body>

<h1>Patients Reports</h1>

{% if patients %}

<table border="1">

<thead>

<tr>

<th>ID</th>

<th>First Name</th>

<th>Last Name</th>

<th>Result Class</th>

<th>Result Confidence</th>

<th>Report</th>

</tr>

</thead>

<tbody>

{% for patient in patients %}

<tr>

<td>{{ patient.id }}</td>

<td>{{ patient.first\_name }}</td>

<td>{{ patient.last\_name }}</td>

<td>{{ patient.result\_class }}</td>

<td>{{ patient.result\_confidence }}</td>

<td>

<!-- to download the PDF report -->

<a href="{{ url\_for('download\_report', patient\_id=patient.id) }}" target="\_blank">Download Report</a>

</td>

</tr>

{% endfor %}

</tbody>

</table>

{% else %}

<p>No patients available.</p>

{% endif %}

<div id="goHome">

<a href="{{ url\_for('main\_page') }}">Back to Home</a>

</div>

</body>

</html>

View Insights Page (insights.html)

<!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">

<link rel="stylesheet" href="{{ url\_for('static', filename='insights\_style.css') }}">

<title>View Insights</title>

<script src="{{ url\_for('static', filename='inactivity.js') }}"></script>

</head>

<body>

<div class="insights-container">

<a href="{{ url\_for('main\_page') }}" class="go-home-btn">Back to Home</a>

<div class="insights-section">

<h2>Distribution of Predictions</h2>

<div class="chart-container">

<iframe src="{{ pie\_chart\_path }}" width="100%" height="400px"></iframe>

</div>

</div>

<div class="insights-section">

<h2>Impact of Family History of Breast Cancer</h2>

<div class="chart-container">

<iframe src="{{ grouped\_bar\_chart\_path }}" width="100%" height="400px"></iframe>

</div>

</div>

<div class="insights-section">

<h2>Lifestyle Impact</h2>

<div class="percentage-info">

<p>{{ smoking\_percentage|round(1) }}% of patients with a malignant diagnosis (cancer) are smokers</p>

<p>{{ drinking\_percentage|round(1) }}% of patients with a malignant diagnosis (cancer) drink alcohol</p>

<p>The occupation with the most malignant cases (cancer) is {{ occupation\_with\_most\_malignant }}</p>

</div>

</div>

</div>

</body>

</html>

Multiple Predictions Page (multiple\_uploads.html)

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Mammogram Prediction</title>

<link rel="stylesheet" href="{{ url\_for('static', filename='styles.css') }}">

<script src="{{ url\_for('static', filename='inactivity.js') }}"></script>

<style>

body {

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

background-color: #fff5f8;

margin: 20px;

}

h1 {

color: #e44d88;

}

h2 {

color: #e44d88;

}

form {

margin-bottom: 20px;

}

input[type="file"] {

margin-bottom: 10px;

padding: 10px;

border: 1px solid #ccc;

border-radius: 5px;

}

button {

background-color: #e44d88;

color: #fff;

padding: 10px 20px;

border: none;

cursor: pointer;

font-size: 16px;

margin-right: 10px;

border-radius: 5px;

}

button:hover {

background-color: #ff80ab;

}

ul {

list-style-type: none;

padding: 0;

}

li {

margin-bottom: 10px;

padding: 10px;

background-color: #fff;

border-radius: 5px;

box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

}

a {

text-decoration: none;

}

.flash-messages {

list-style-type: none;

margin: 0;

padding: 0;

color: #721c24;

}

</style>

</head>

<!DOCTYPE html>

<html lang="en">

<body>

{% with messages = get\_flashed\_messages() %}

{% if messages %}

<ul class="flash-messages">

{% for message in messages %}

<li>{{ message }}</li>

{% endfor %}

</ul>

{% endif %}

{% endwith %}

<h1>Multiple Predictions</h1>

<form action="/multiple\_uploads" method="post" enctype="multipart/form-data">

<input type="file" name="file[]" accept=".jpg, .jpeg, .png" multiple>

<button type="submit">Predict</button>

</form>

{% if results %}

<h2>Prediction Results:</h2>

<ul>

{% for result in results %}

<li>{{ result.filename }}: {{ result.prediction }}</li>

{% endfor %}

</ul>

<a href="{{ url\_for('clear\_predictions') }}">

<button>Clear Predictions</button>

</a>

{% if csv\_filename %}

<a href="{{ url\_for('download\_results', filename=csv\_filename) }}" download>

<button>Download Results</button>

</a>

{% endif %}

{% endif %}

<a href="{{ url\_for('main\_page') }}">

<button>Back to Home</button>

</a>

</body>

</html>

**Static Folder - CSS style sheets + JS file to handle inactivity**

styles.css

body {

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

margin: 0;

padding: 0;

}

header {

display: flex;

justify-content: space-between; /\* buttons to the right \*/

background-color: #e44d88;

color: #fff;

padding: 15px;

align-items: center;

}

.header-buttons {

display: flex;

}

.account-button,

.logout-button {

background-color: #fff;

color: #e44d88;

border: none;

padding: 8px 15px;

margin-left: 10px;

border-radius: 4px;

cursor: pointer;

text-decoration: none;

transition: background-color 0.3s, color 0.3s;

}

.account-button:hover,

.logout-button:hover {

background-color: #f66ba8;;

color: #fff;

}

.container {

display: flex;

}

.sidebar {

width: 200px;

background-color: #f0e4e7;

padding: 15px;

border-radius: 10px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

.sidebar a {

color: #3c3641;

text-decoration: none;

display: block;

margin-bottom: 10px;

padding: 10px;

border-radius: 8px;

border: 1px solid #e44d88;

transition: background-color 0.3s, color 0.3s, border-color 0.3s; /\*for a smooth effect \*/

}

.sidebar a:hover {

background-color: #f66ba8;

color: #fff;

border-color: #fff;

}

.main-content {

flex-grow: 1;

padding: 20px;

}

.burger-menu {

display: none;

cursor: pointer;

}

.bar {

width: 25px;

height: 3px;

background-color: #fff;

margin: 6px 0;

}

.predict-button {

background-color: #e44d88 ;

color: #fff ;

padding: 15px ;

border: none ;

border-radius: 8px ;

cursor: pointer ;

font-size: 16px ;

font-weight: bold ;

transition: background-color 0.3s ease ;

}

.predict-button:hover {

background-color: #c0376b ;

}

@media screen and (max-width: 768px) {

.burger-menu {

display: block;

}

.sidebar {

display: none;

position: absolute;

z-index: 1;

background-color: #f0e4e7;

padding: 15px;

border-radius: 10px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

.sidebar a {

display: block;

margin-bottom: 10px;

}

}

/\* style for full screen \*/

@media screen and (min-width: 769px) {

.burger-menu {

display: none; /\* Hide the burger menu for larger screens \*/

}

.sidebar {

display: block; /\* Display the full sidebar for larger screens \*/

width: 200px;

background-color: #f0e4e7;

padding: 15px;

border-radius: 10px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

}

.file-upload img {

max-width: 300px;

max-height: 300px;

margin-top: 10px;

display: none;

}

.input-group {

margin-bottom: 16px;

}

.input-group label {

display: block;

margin-bottom: 8px;

font-weight: bold;

color: #3c3641;

}

.input-group input[type="text"],

.input-group input[type="tel"],

.input-group input[type="date"],

.input-group input[type="checkbox"],

.input-group input[type="file"] {

width: 100%;

padding: 8px;

margin-bottom: 8px;

border: 1px solid #ddd;

border-radius: 4px;

}

auth\_styles.css

body {

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

margin: 0;

display: flex;

flex-direction: column;

align-items: center;

min-height: 100vh;

background-color: #f9f9f9;

}

.flash-messages {

text-align: center;

margin-top: 20px;

position: absolute;

width: 80%;

z-index: 999; /\* z-index to ensure it's above other elements \*/

}

.flash-message {

padding: 10px;

margin-bottom: 10px;

border: 1px solid #ccc;

border-radius: 5px;

background-color: #f8d7da;

color: #721c24;

}

.flash-message.success {

background-color: #d4edda;

color: #155724;

}

.container {

background-color: #fff;

padding: 20px;

border-radius: 8px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

width: 300px;

margin-top: 150px;

margin-bottom: 50px;

}

form {

display: flex;

flex-direction: column;

}

label {

margin-bottom: 8px;

font-weight: bold;

color: #555;

}

input {

padding: 8px;

margin-bottom: 16px;

border: 1px solid #ddd;

border-radius: 4px;

}

button {

background-color: #e44d88;

color: #fff;

padding: 10px;

border: none;

border-radius: 4px;

cursor: pointer;

}

button:hover {

background-color: #c0376b;

}

a {

color: #e44d88;

text-decoration: none;

}

a:hover {

text-decoration: underline;

}

@keyframes fadeIn {

from {

opacity: 0;

}

to {

opacity: 1;

}

}

h2 {

text-align: center;

}

p {

text-align: center;

margin-top: 10px;

}

insights\_style.css

body {

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

margin: 0;

padding: 0;

background-color: #fff5f8;

}

.insights-container {

max-width: 800px;

margin: 20px auto;

padding: 20px;

background-color: #f0e4e7;

border-radius: 10px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

.insights-section {

margin-bottom: 20px;

}

.chart-container {

border: 1px solid #ccc;

border-radius: 8px;

overflow: hidden;

margin-top: 10px;

}

.chart-container img {

width: 100%;

display: block;

}

.percentage-info, .occupation-info {

background-color: #fff;

border: 1px solid #ccc;

border-radius: 8px;

padding: 10px;

margin-top: 10px;

}

.percentage-info p, .occupation-info p {

margin: 0;

padding: 5px 0;

}

/\* ---------------------------------- \*/

.insights-container {

background-color: #f0e4e7;

color: #4b2e39;

}

.insights-section {

background-color: #ffffff;

padding: 20px;

border-radius: 8px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

}

.chart-container, .percentage-info, .occupation-info {

background-color: #ffffff;

border: 1px solid #ccc;

border-radius: 8px;

margin-top: 10px;

padding: 15px;

}

h2 {

color: #d34c63;

}

.go-home-btn {

position: absolute;

top: 10px;

left: 10px;

text-decoration: none;

padding: 10px;

background-color: #e44d88 !important;

color: #fff !important;

border-radius: 5px;

font-weight: bold;

}

.go-home-btn:hover {

background-color: #ff80ab !important;

}

inactivity.js

// note: since automatic redirection is not working after session timeout in @app.before\_request, im using this script to simulate an automatic redirection by refreshing after 10 minutes of inactivity

// timeout after 10 mins

var inactivityTimeout = 600000;

// reload page

function reloadPage() {

location.reload();

}

// Reset the inactivity timer if user active

function resetInactivityTimer() {

clearTimeout(inactivityTimer);

inactivityTimer = setTimeout(reloadPage, inactivityTimeout);

}

//initial inactivity timer

var inactivityTimer = setTimeout(reloadPage, inactivityTimeout);

//event listeners to reset the timer

document.addEventListener('mousemove', resetInactivityTimer);

document.addEventListener('keydown', resetInactivityTimer);

document.addEventListener('scroll', resetInactivityTimer);

**app.py (Flask Code)**

from flask import Flask, render\_template, request, jsonify, send\_from\_directory

from keras.models import load\_model

from keras.preprocessing import image

import numpy as np

import os

from flask\_sqlalchemy import SQLAlchemy

from werkzeug.utils import secure\_filename

from datetime import datetime

from flask import send\_file

import csv

from PIL import Image,UnidentifiedImageError

#DB imports

from io import BytesIO

from reportlab.lib.pagesizes import letter

from reportlab.lib import colors

from reportlab.lib.styles import getSampleStyleSheet

from reportlab.platypus import SimpleDocTemplate, Table, TableStyle, Paragraph

from reportlab.platypus import Spacer

#authentication imports

from flask import redirect, url\_for, flash

from werkzeug.security import generate\_password\_hash, check\_password\_hash

from flask\_login import UserMixin

from flask\_login import LoginManager, login\_user, logout\_user, login\_required, current\_user

from datetime import timedelta

#insights imports

import pandas as pd

import plotly.express as px

os.urandom(24)

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

app.secret\_key = b'BreastCancer2024'

# for timeout

app.config['SESSION\_PERMANENT'] = True

app.config['PERMANENT\_SESSION\_LIFETIME'] = timedelta(seconds=600)

# SQLite database file

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///cancer\_proj2.db'

db = SQLAlchemy(app)

model = load\_model('21jan-ddsm-bestmodel.h5')

#function to pre process image

def preprocess\_image(img\_path):

img = image.load\_img(img\_path, target\_size=(227, 227))

img\_array = image.img\_to\_array(img)

img\_array = np.expand\_dims(img\_array, axis=0)

img\_array /= 255.0

return img\_array

#User DB

class User(db.Model, UserMixin):

id = db.Column(db.Integer, primary\_key=True)

employee\_id = db.Column(db.String(50), unique=True, nullable=False)

first\_name = db.Column(db.String(50), nullable=False)

last\_name = db.Column(db.String(50), nullable=False)

email = db.Column(db.String(120), unique=True, nullable=False)

password = db.Column(db.String(60), nullable=False)

#Patient DB

class Patient(db.Model):

id = db.Column(db.Integer, primary\_key=True)

patient\_id = db.Column(db.String(50), nullable=False)

first\_name = db.Column(db.String(50), nullable=False)

last\_name = db.Column(db.String(50), nullable=False)

phone\_number = db.Column(db.String(15), nullable=False)

address = db.Column(db.String(100), nullable=False)

date\_of\_birth = db.Column(db.Date, nullable=False)

ethnicity = db.Column(db.String(50), nullable=False)

smoking = db.Column(db.Boolean, nullable=True)

drinking = db.Column(db.Boolean, nullable=True)

occupation = db.Column(db.String(50), nullable=False)

personal\_history = db.Column(db.Boolean, nullable=True)

family\_history = db.Column(db.Boolean, nullable=True)

mammogram\_image\_path = db.Column(db.String(100), nullable=False)

result\_class = db.Column(db.String(20), nullable=True)

result\_confidence = db.Column(db.Float, nullable=True)

user\_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)

user = db.relationship('User', backref='patients')

# Create the database tables

if \_\_name\_\_ == '\_\_main\_\_':

with app.app\_context():

db.create\_all()

@app.route('/static/<path:filename>')

def serve\_static(filename):

root\_dir = os.path.dirname(os.getcwd())

return send\_from\_directory(os.path.join(root\_dir, 'static'), filename)

@app.route('/')

def index():

return redirect(url\_for('login'))

@app.route('/main\_page')

@login\_required

def main\_page():

return render\_template('main\_page.html', prediction=None)

UPLOAD\_FOLDER = 'static/images'

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

app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER

def allowed\_file(filename):

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

#----------------------------------------------------------------------------------------------------

# to make predictions (single upload)

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

def predict():

# Check if the Patient ID already exists in the database

existing\_patient = Patient.query.filter\_by(patient\_id=request.form['patientID']).first()

if existing\_patient:

return jsonify({'error': 'Patient ID already exists.'})

#------------------------

if 'file' not in request.files:

return jsonify({'error': 'No file part'})

file = request.files['file']

if file.filename == '':

return jsonify({'error': 'No selected file'})

dob\_str = request.form['dob']

# date string coverted to a Python date object

date\_of\_birth = datetime.strptime(dob\_str, '%Y-%m-%d').date()

img\_path = f'static/images/{secure\_filename(file.filename)}'

file.save(img\_path)

# Check if the uploaded file is an image

try:

img = Image.open(img\_path)

except UnidentifiedImageError:

# if not an image

return jsonify({'error': 'The selected file is not a valid image.'})

img\_array = preprocess\_image(img\_path)

prediction = model.predict(img\_array)

result = {

'class': 'Malignant' if prediction > 0.5 else 'Benign',

'confidence': float(prediction)

}

print(result)

# Save to database

user\_id = current\_user.id if current\_user.is\_authenticated else None # Get the user ID of the current logged-in user

patient = Patient(

patient\_id=request.form['patientID'],

first\_name=request.form['firstName'],

last\_name=request.form['lastName'],

phone\_number=request.form['phoneNumber'],

address=request.form['address'],

date\_of\_birth=date\_of\_birth,

ethnicity=request.form['ethnicity'],

smoking=request.form.get('smoking') == 'on',

drinking=request.form.get('drinking') == 'on',

occupation=request.form['occupation'],

personal\_history=request.form.get('personalHistory') == 'on',

family\_history=request.form.get('familyHistory') == 'on',

mammogram\_image\_path=img\_path,

result\_class=result['class'],

result\_confidence=result['confidence'],

user\_id=user\_id

)

db.session.add(patient)

db.session.commit()

print(f"Patient added: {patient}")

return jsonify(result)

#-------------------------------------------------------------------------------

# to make predictions (multiple upload)

def predict\_image(file\_path):

img = image.load\_img(file\_path, target\_size=(227, 227))

img\_array = image.img\_to\_array(img)

rra = np.expand\_dims(img\_array, axis=0)

prediction = model.predict(img\_array)

return 'Benign' if prediction[0][0] < 0.5 else 'Malignant'

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

@login\_required

def multiple\_uploads():

results = None

csv\_filename = None # Initialize csv\_filename

# Clear previous CSV file if it exists

csv\_path = 'static/prediction\_results.csv'

if os.path.exists(csv\_path):

os.remove(csv\_path)

if request.method == 'POST':

results = []

files = request.files.getlist('file[]')

# if no image selected

if not any(files):

flash('Please select at least one file for prediction.', 'error')

return redirect(url\_for('multiple\_uploads'))

for file in files:

# saving img to a temporary location

file\_path = f'tmp/{file.filename}'

file.save(file\_path)

# predict and save

prediction = predict\_image(file\_path)

results.append({'filename': file.filename, 'prediction': prediction})

# Save results to CSV file

with open(csv\_path, 'w', newline='') as csvfile:

fieldnames = ['filename', 'prediction']

writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

writer.writeheader()

writer.writerows(results)

csv\_filename = 'prediction\_results.csv' # Update csv\_filename after saving results

return render\_template('multiple\_uploads.html', results=results, csv\_filename=csv\_filename)

#To clear page

@app.route('/clear\_predictions')

def clear\_predictions():

# clear prediction\_results.csv file

csv\_path = 'static/prediction\_results.csv'

if os.path.exists(csv\_path):

os.remove(csv\_path)

# Redirect back to the multiple\_uploads

return redirect(url\_for('multiple\_uploads'))

#route to handle download

@app.route('/download\_results/<filename>')

def download\_results(filename):

return send\_from\_directory('static', filename, as\_attachment=True)

#-------------------------------------------------------------------------------

#REPORTS

@app.route('/patients')

@login\_required

def patients():

# query to get only the patients associated with the current user

patients = Patient.query.filter\_by(user\_id=current\_user.id).all()

return render\_template('patients.html', patients=patients)

@app.route('/download\_report/<int:patient\_id>', methods=['GET'])

@login\_required

def download\_report(patient\_id):

# get patient details from the database based on patient\_id

patient = Patient.query.get(patient\_id)

#creating pdf report using the reportlab library

buffer = BytesIO()

# Create a PDF document

pdf\_title = f"Mammography Report - Patient ID {patient.patient\_id}" #set title of pdf

pdf = SimpleDocTemplate(buffer, pagesize=letter, title=pdf\_title)

# styles

styles = getSampleStyleSheet()

style = styles['Normal']

#report content

title = "Mammography Report"

content = [

Paragraph(title, styles['Title']),

Paragraph(f"Patient ID: {patient.patient\_id}", styles['Heading2']),

Spacer(1, 12),

Paragraph(f"Report by: { current\_user.first\_name} { current\_user.last\_name } ", style),

Spacer(1, 12),

Paragraph(f"Date: {datetime.today().strftime('%d-%m-%Y')}", style),

Spacer(1, 12),

]

patient\_details = [

("Patient ID", patient.patient\_id),

("First Name", patient.first\_name),

("Last Name", patient.last\_name),

("Date of Birth", patient.date\_of\_birth.strftime('%d-%m-%Y')),

("Result", patient.result\_class),

]

# table for patient details

patient\_table = Table(patient\_details, colWidths=[150, 250])

# table styles

patient\_table.setStyle(TableStyle([

('BACKGROUND', (0, 0), (-1, 0), colors.grey),

('TEXTCOLOR', (0, 0), (-1, 0), colors.whitesmoke),

('ALIGN', (0, 0), (-1, -1), 'CENTER'),

('FONTNAME', (0, 0), (-1, 0), 'Helvetica-Bold'),

('BOTTOMPADDING', (0, 0), (-1, 0), 12),

('BACKGROUND', (0, 1), (-1, -1), colors.beige),

]))

content.append(patient\_table)

# Build the PDF document

pdf.build(content)

buffer.seek(0)

return send\_file(buffer, as\_attachment=True, download\_name=f'report\_patient\_{patient.id}.pdf', mimetype='application/pdf')

#SIGNUP

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

def signup():

if request.method == 'POST':

employee\_id = request.form['employee\_id']

first\_name = request.form['first\_name']

last\_name = request.form['last\_name']

email = request.form['email']

password = request.form['password']

repeat\_password = request.form['repeat\_password']

# to check if employee ID or email already exists

existing\_user\_id = User.query.filter\_by(employee\_id=employee\_id).first()

existing\_user\_email = User.query.filter\_by(email=email).first()

if existing\_user\_id:

flash('Employee ID is already in use.', 'danger')

return redirect(url\_for('signup'))

if existing\_user\_email:

flash('Email address is already in use.', 'danger')

return redirect(url\_for('signup'))

# to check if passwords match

if password != repeat\_password:

flash('Passwords do not match', 'error')

return redirect(url\_for('signup'))

# Hash password

hashed\_password = generate\_password\_hash(password)

# Create a new user

new\_user = User(

employee\_id=employee\_id,

first\_name=first\_name,

last\_name=last\_name,

email=email,

password=hashed\_password

)

# Add new user to the db

db.session.add(new\_user)

db.session.commit()

flash('Account created successfully! Please log in.', 'success')

return redirect(url\_for('login'))

return render\_template('signup.html')

#LOGIN

login\_manager = LoginManager(app)

login\_manager.login\_view = 'login'

#SESSION TIMEOUT

#load user

@login\_manager.user\_loader

def load\_user(user\_id):

return User.query.get(int(user\_id))

# update the user's last activity timestamp and check for session timeout

@app.before\_request

def update\_last\_activity\_and\_check\_timeout():

if current\_user.is\_authenticated:

current\_user.last\_activity = datetime.utcnow()

db.session.commit()

last\_activity = current\_user.last\_activity

print(f"Last Activity: {last\_activity}")

if last\_activity:

time\_difference = datetime.utcnow() - last\_activity

if time\_difference > timedelta(seconds=600):

flash('Your session has timed out. Please log in again.', 'info')

logout\_user()

return redirect(url\_for('login'))

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

def login():

if request.method == 'POST':

employee\_id = request.form['employee\_id']

password = request.form['password']

# Find user by employee ID

user = User.query.filter\_by(employee\_id=employee\_id).first()

if user and check\_password\_hash(user.password, password):

login\_user(user)

return redirect(url\_for('main\_page'))

flash('Invalid employee ID or password.', 'error')

return redirect(url\_for('login'))

return render\_template('login.html')

#LOGOUT

@app.route('/logout')

@login\_required

def logout():

logout\_user()

flash('Logout successful', 'success')

return redirect(url\_for('login'))

#INSIGHTS

#to analyze data and generate insights

def analyze\_data():

# get patient data from the db

patients = Patient.query.filter\_by(user\_id=current\_user.id).all()

# patient data converted to data frame:

df = pd.DataFrame([p.\_\_dict\_\_ for p in patients])

#when no predictions are made yet (new user account)

if 'result\_class' not in df.columns:

# When 'result\_class' column is not present

return None, None, None, None, None

#---------------------------------------------------------------------------------------------------

# 1. Pie Chart: Distribution of Predictions

count\_benign = df['result\_class'].value\_counts().get('Benign', 0)

count\_malignant = df['result\_class'].value\_counts().get('Malignant', 0)

pie\_chart\_data = pd.DataFrame({

'Diagnosis': ['Benign', 'Malignant'],

'Count': [count\_benign, count\_malignant]

})

pie\_chart = px.pie(pie\_chart\_data, names='Diagnosis', values='Count', title='Distribution of Predictions')

# save chart as html

pie\_chart\_html\_path = 'static/pie\_chart.html'

pie\_chart.write\_html(pie\_chart\_html\_path)

#---------------------------------------------------------------------------------------------------

# 2. Grouped Bar Chart: Impact of Family History

count\_with\_family = df[df['family\_history'] == True]['result\_class'].value\_counts()

count\_without\_family = df[df['family\_history'] == False]['result\_class'].value\_counts()

grouped\_bar\_chart\_data = pd.DataFrame({

'Diagnosis': ['Benign', 'Malignant'],

'With Family History': [count\_with\_family.get('Benign', 0), count\_with\_family.get('Malignant', 0)],

'Without Family History': [count\_without\_family.get('Benign', 0), count\_without\_family.get('Malignant', 0)]

})

grouped\_bar\_chart = px.bar(

grouped\_bar\_chart\_data,

x='Diagnosis',

y=['With Family History', 'Without Family History'],

title='Impact of Family History',

barmode='group'

)

# save chart as html

grouped\_bar\_chart\_html\_path = 'static/grouped\_bar\_chart.html'

grouped\_bar\_chart.write\_html(grouped\_bar\_chart\_html\_path)

#---------------------------------------------------------------------------------------------------

# 3. Calculate lifestyle impact percentages

total\_malignant = count\_malignant + count\_benign

smoking\_percentage = (df[df['smoking'] == True]['result\_class'].value\_counts().get('Malignant', 0) / total\_malignant) \* 100

drinking\_percentage = (df[df['drinking'] == True]['result\_class'].value\_counts().get('Malignant', 0) / total\_malignant) \* 100

#---------------------------------------------------------------------------------------------------

# 4. Find occupation with the most malignant cases

malignant\_occupations = df[df['result\_class'] == 'Malignant']['occupation']

# to check any rows with 'Malignant' result\_class

occupation\_with\_most\_malignant = malignant\_occupations.mode().iloc[0] if not malignant\_occupations.empty else None

return smoking\_percentage, drinking\_percentage, occupation\_with\_most\_malignant, pie\_chart\_html\_path, grouped\_bar\_chart\_html\_path

@app.route('/insights')

@login\_required

def insights():

# Analyze the data

smoking\_percentage, drinking\_percentage, occupation\_with\_most\_malignant, pie\_chart\_path, grouped\_bar\_chart\_path = analyze\_data()

return render\_template(

'insights.html',

smoking\_percentage=smoking\_percentage,

drinking\_percentage=drinking\_percentage,

occupation\_with\_most\_malignant=occupation\_with\_most\_malignant,

pie\_chart\_path=pie\_chart\_path,

grouped\_bar\_chart\_path=grouped\_bar\_chart\_path

)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)