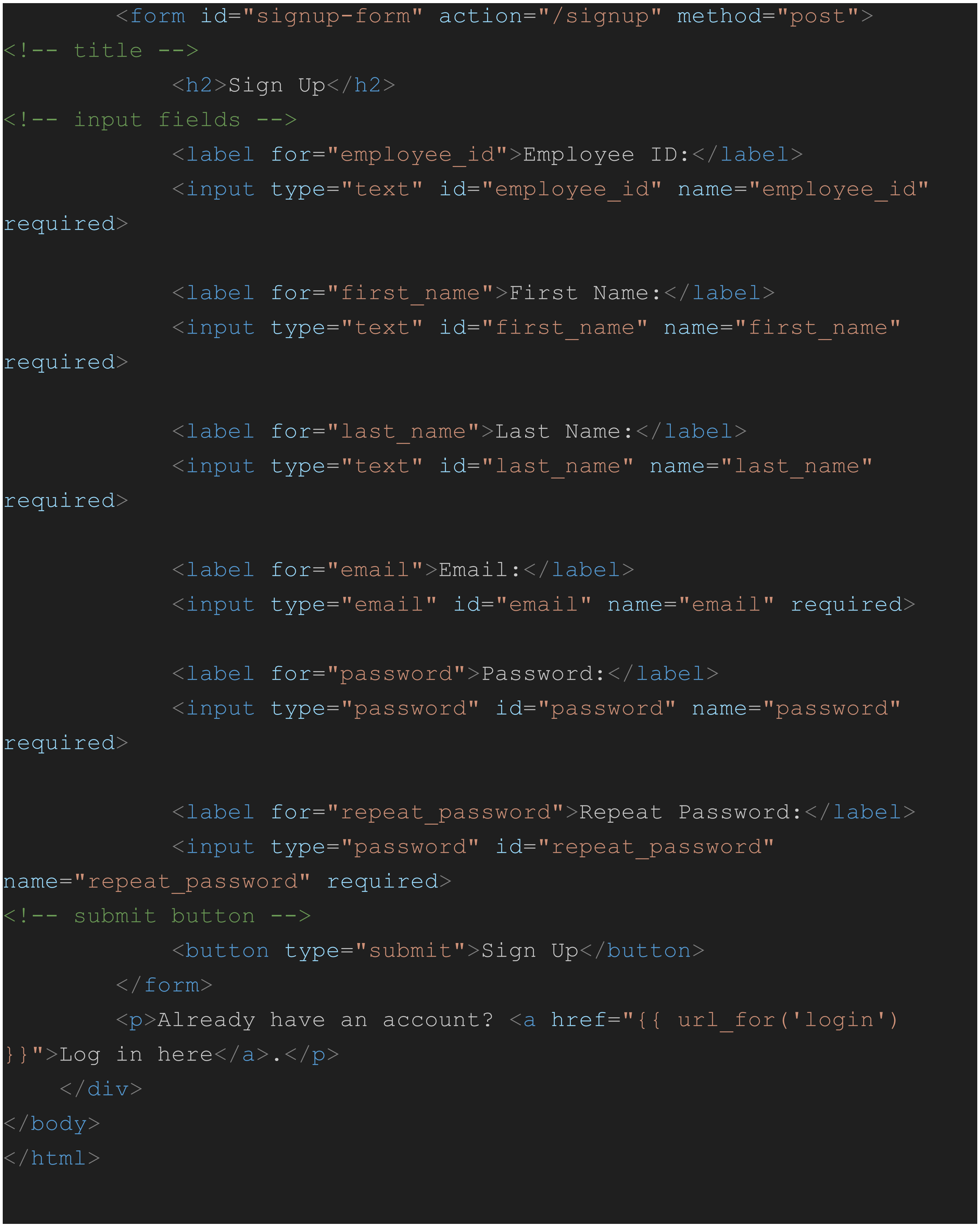
Codes from the Flask App

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)

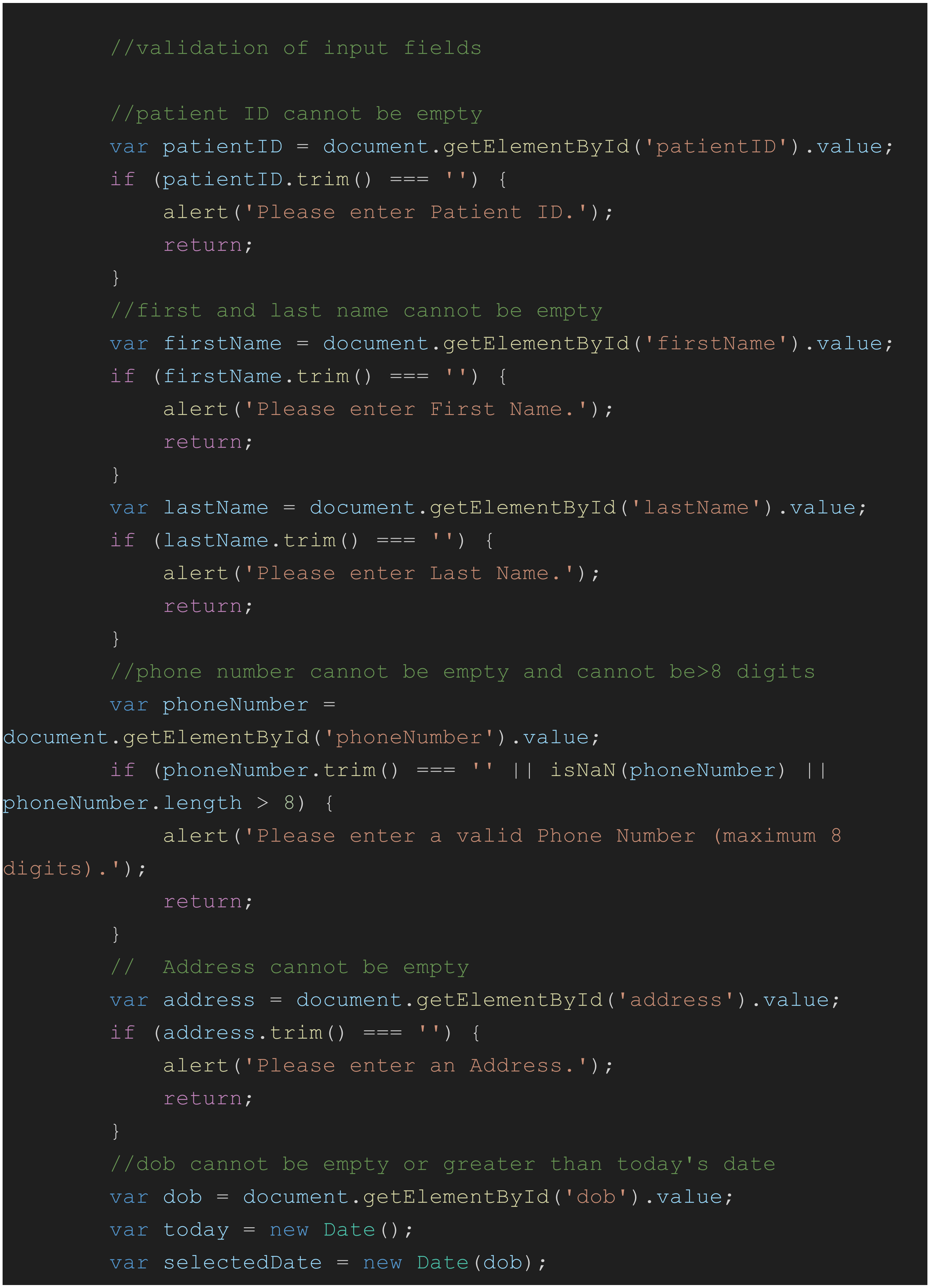
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <link rel="stylesheet" href="{{ url_for('static',
filename='auth_style.css') }}">
    <title>Login</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() {
```



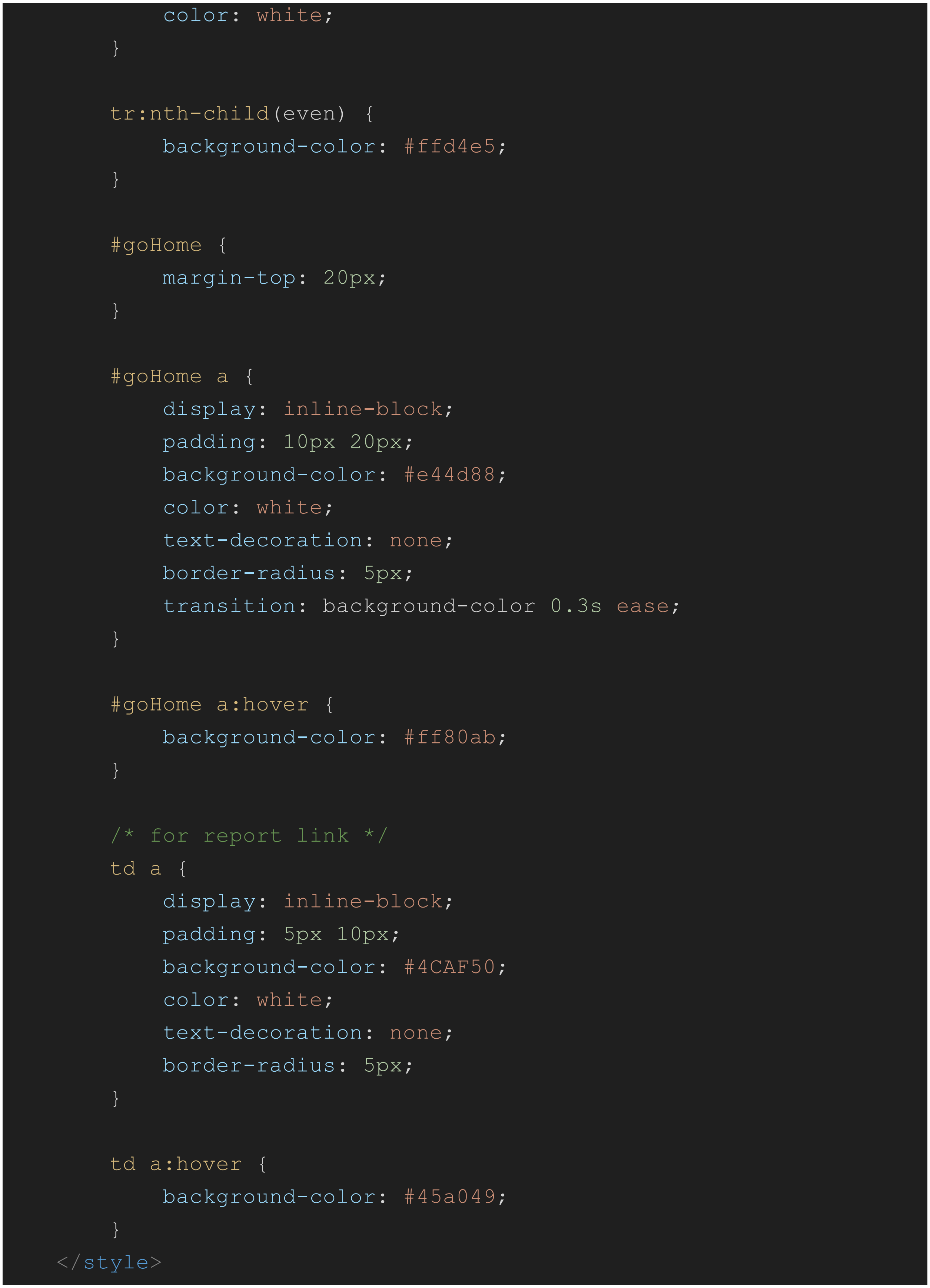
```
        //validation of input fields

        //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>
```

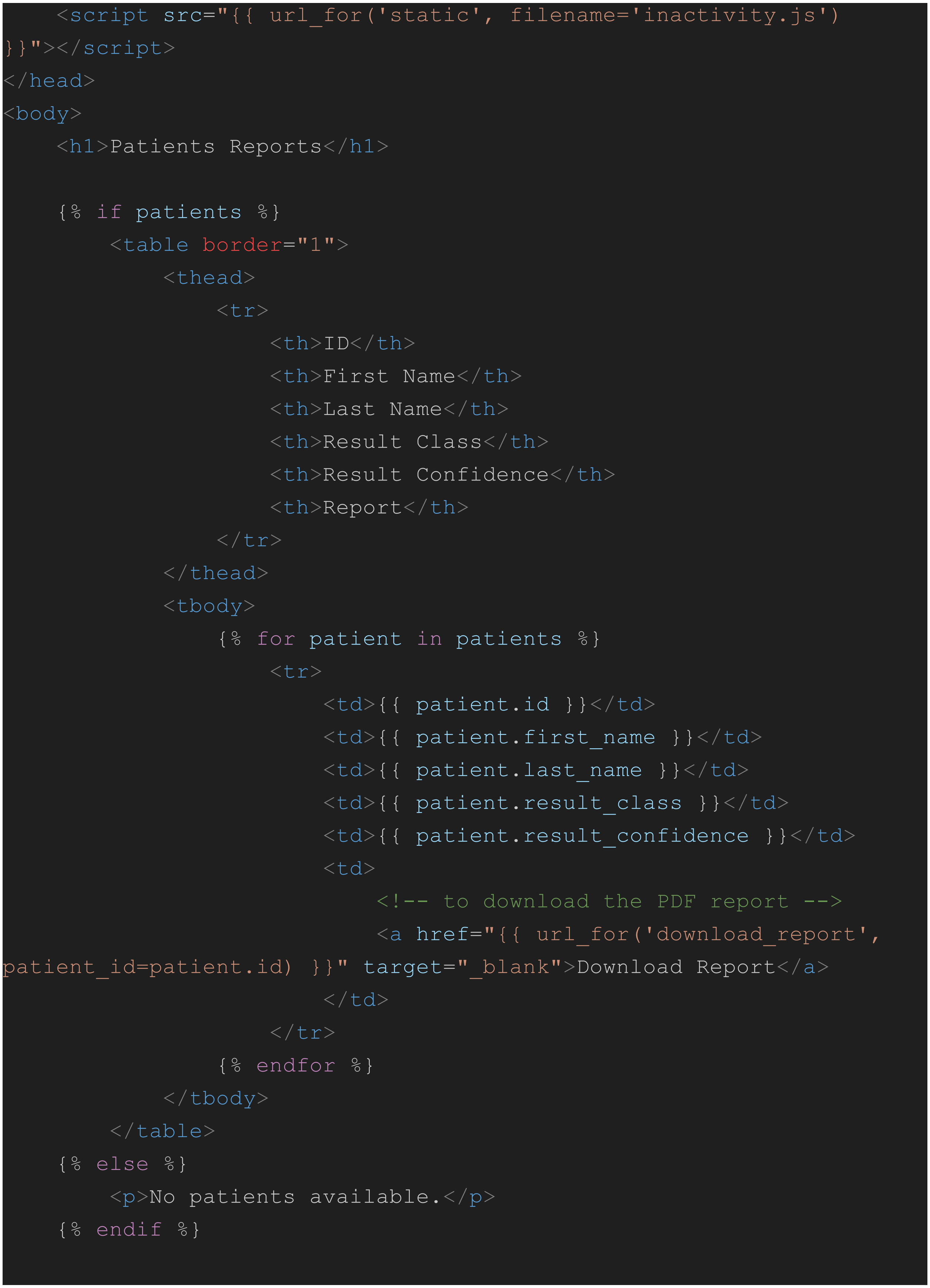
```
</html>
```

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>
```

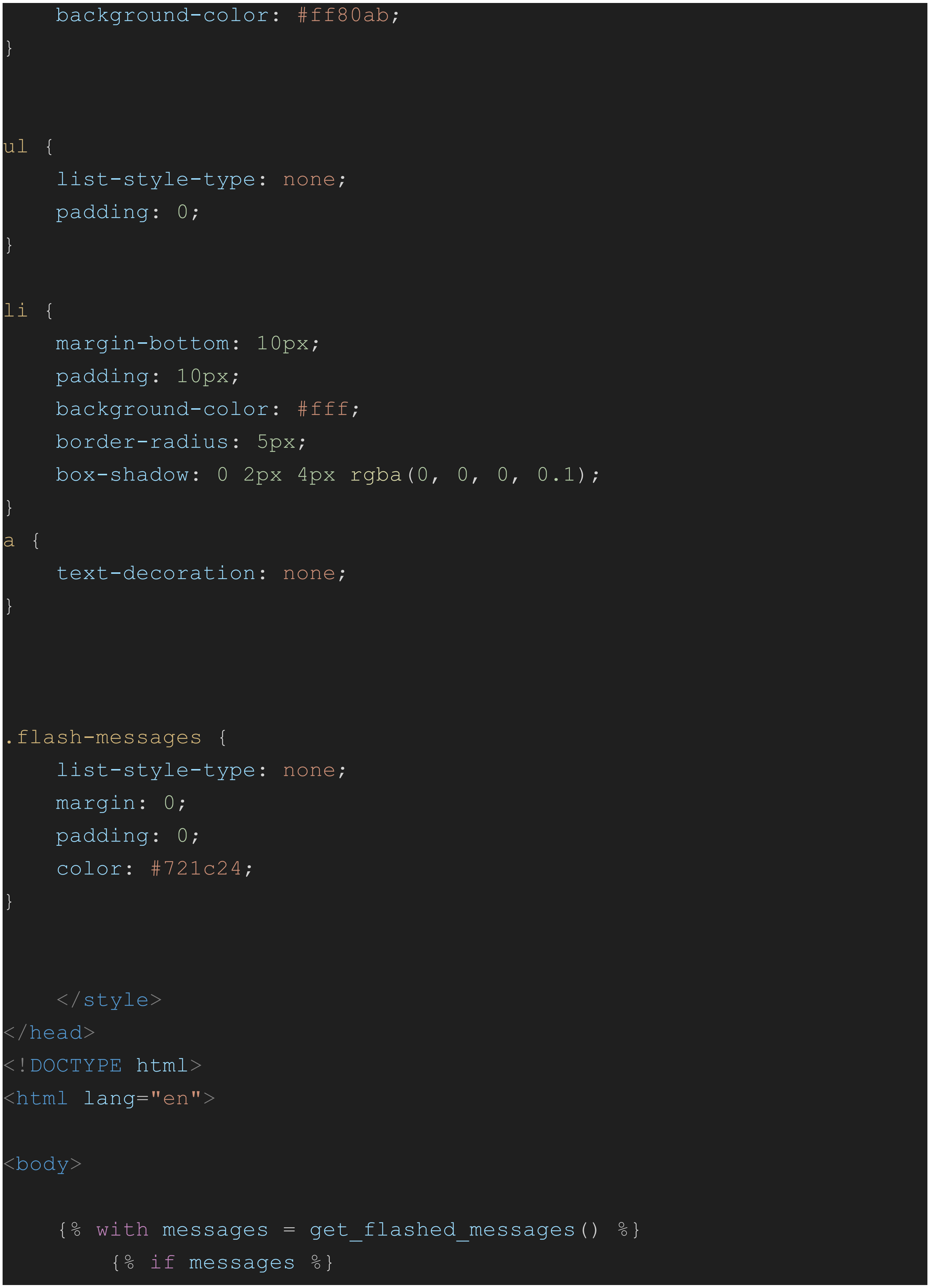
View Insights Page (insights.html)

```
<!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 %}
```

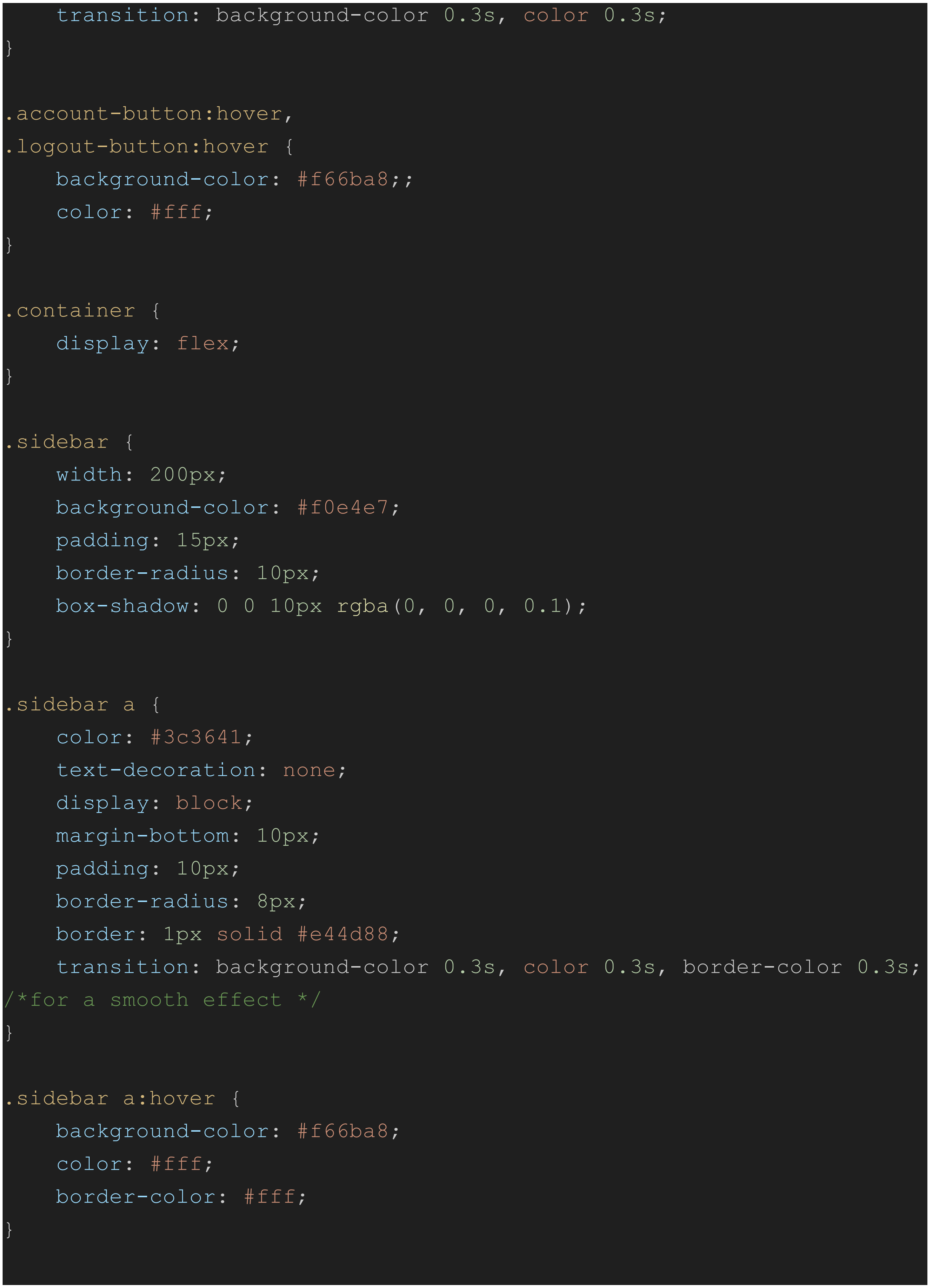
```
            <ul class="flash-messages">
                {% for message in messages %}
                    <li>{{ message }}</li>
                {% endfor %}
        </ul>
        {% endif %}
    {% endwith %}

    <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>
```

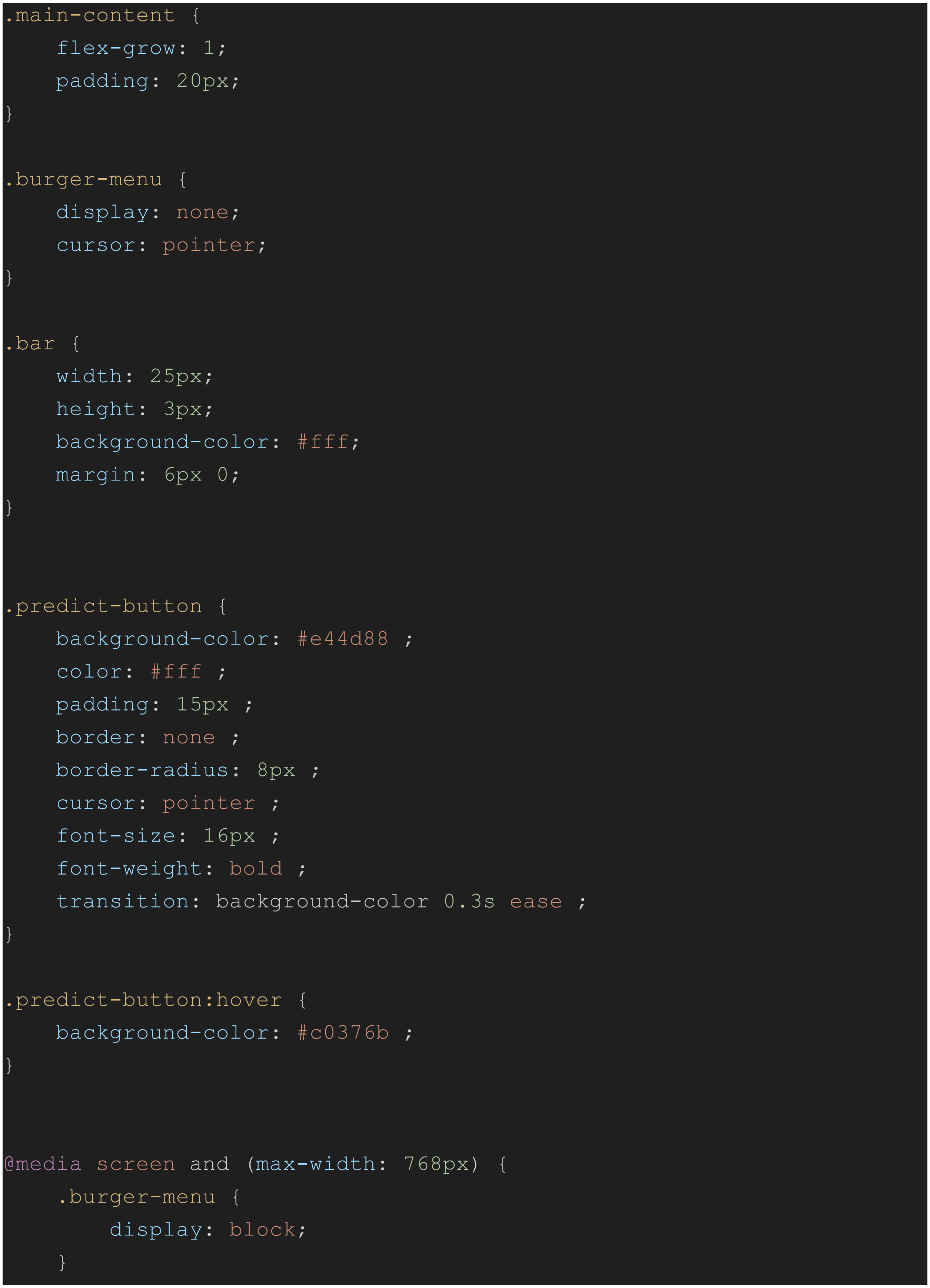
Static Folder - CSS style sheets + JS file to handle inactivity

styles.css

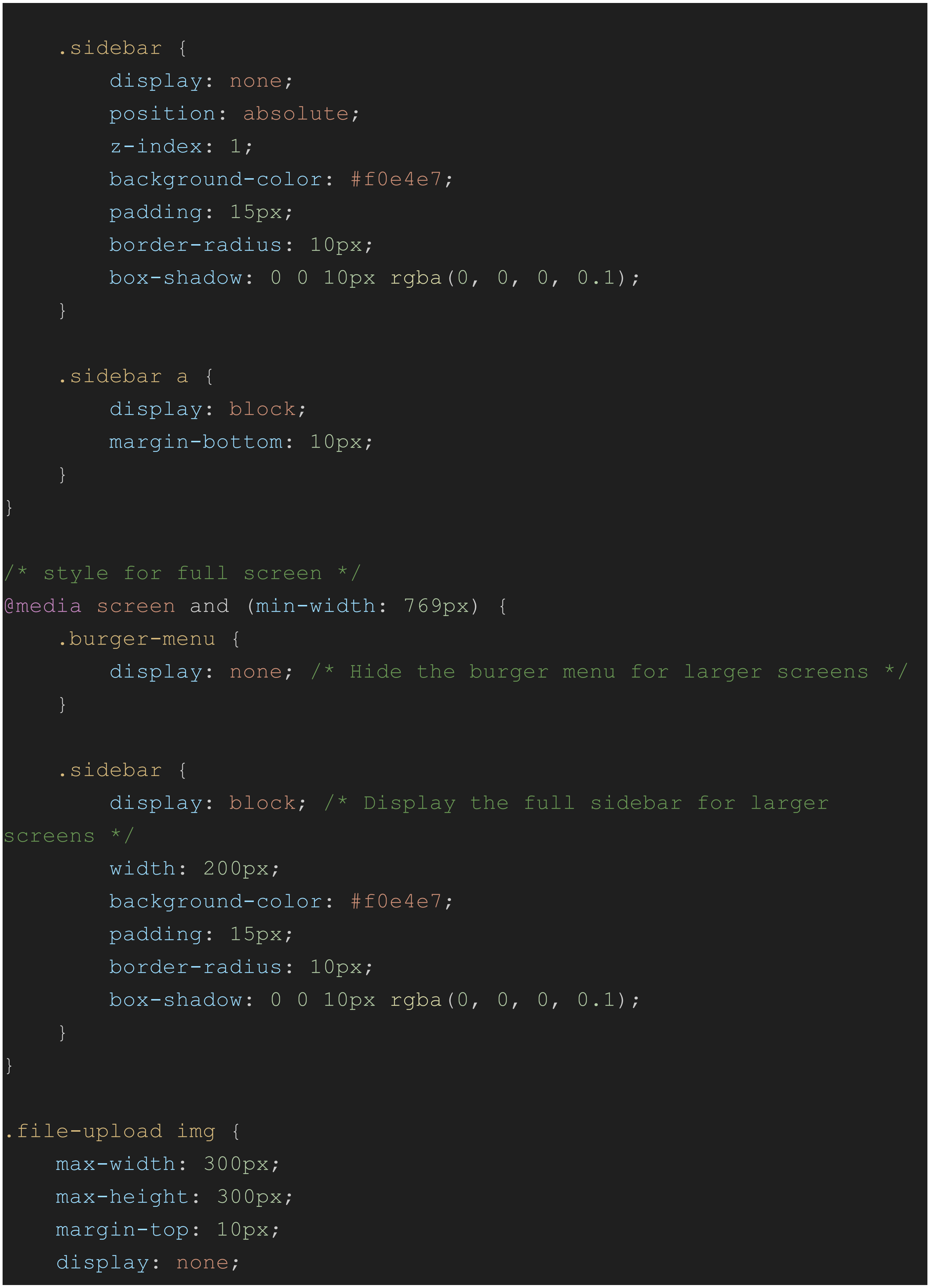
```
body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    margin: 0;
    padding: 0;
}


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;
}
```



```css
.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;
}
```

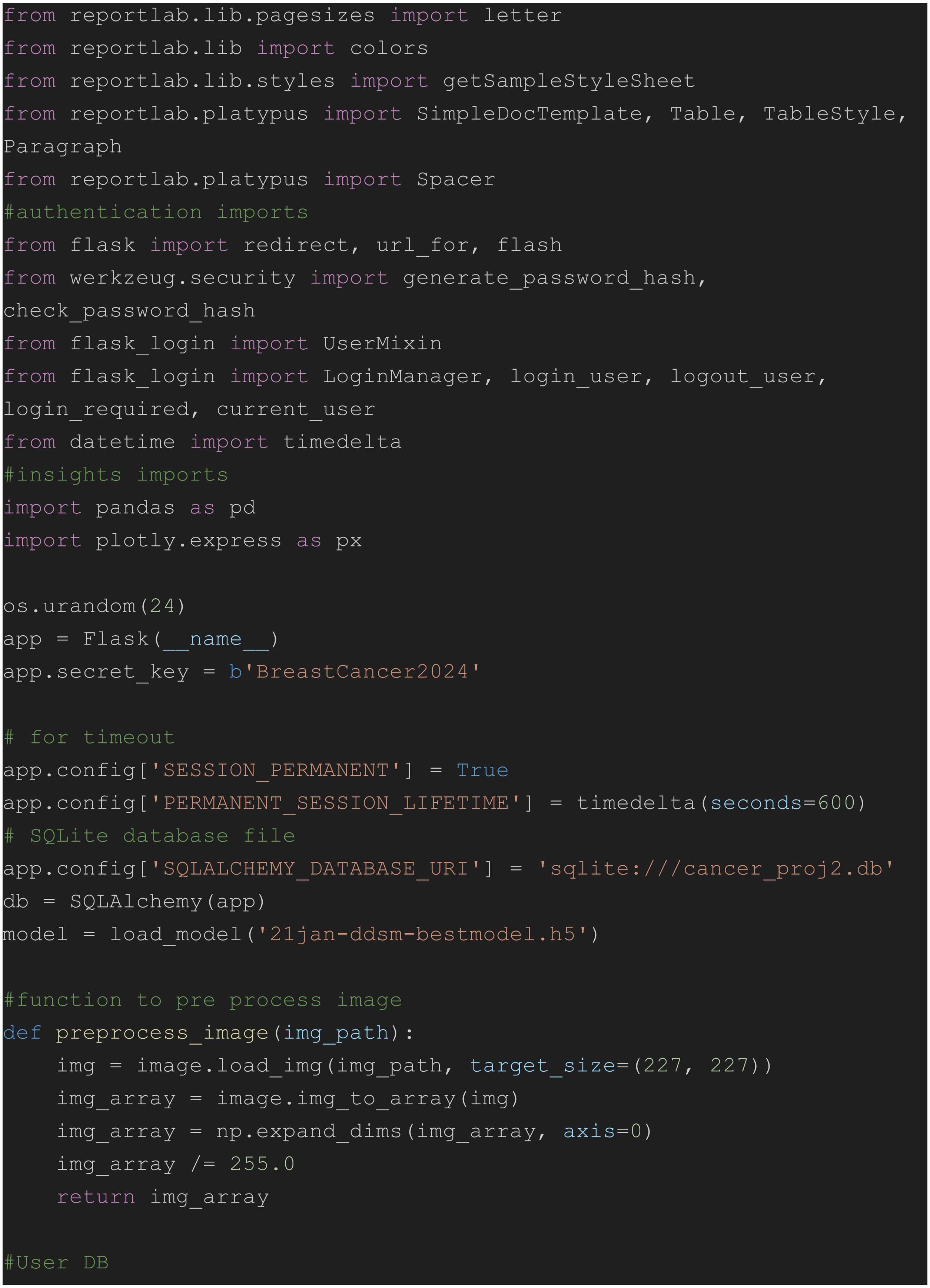
inactivity.js

```
// 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;
```

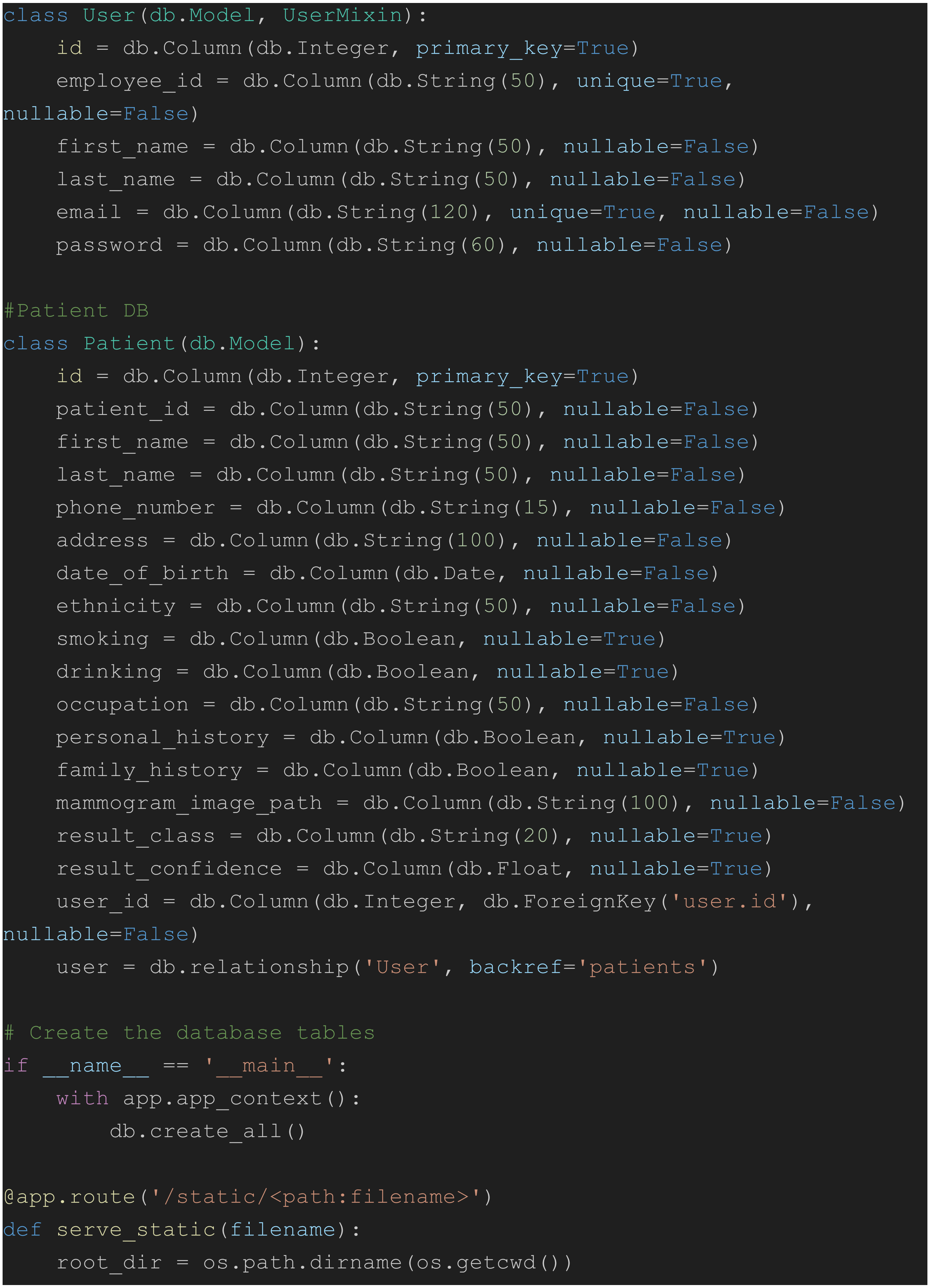
```
// reload page
function reloadPage() {
    location.reload();
}

// 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)

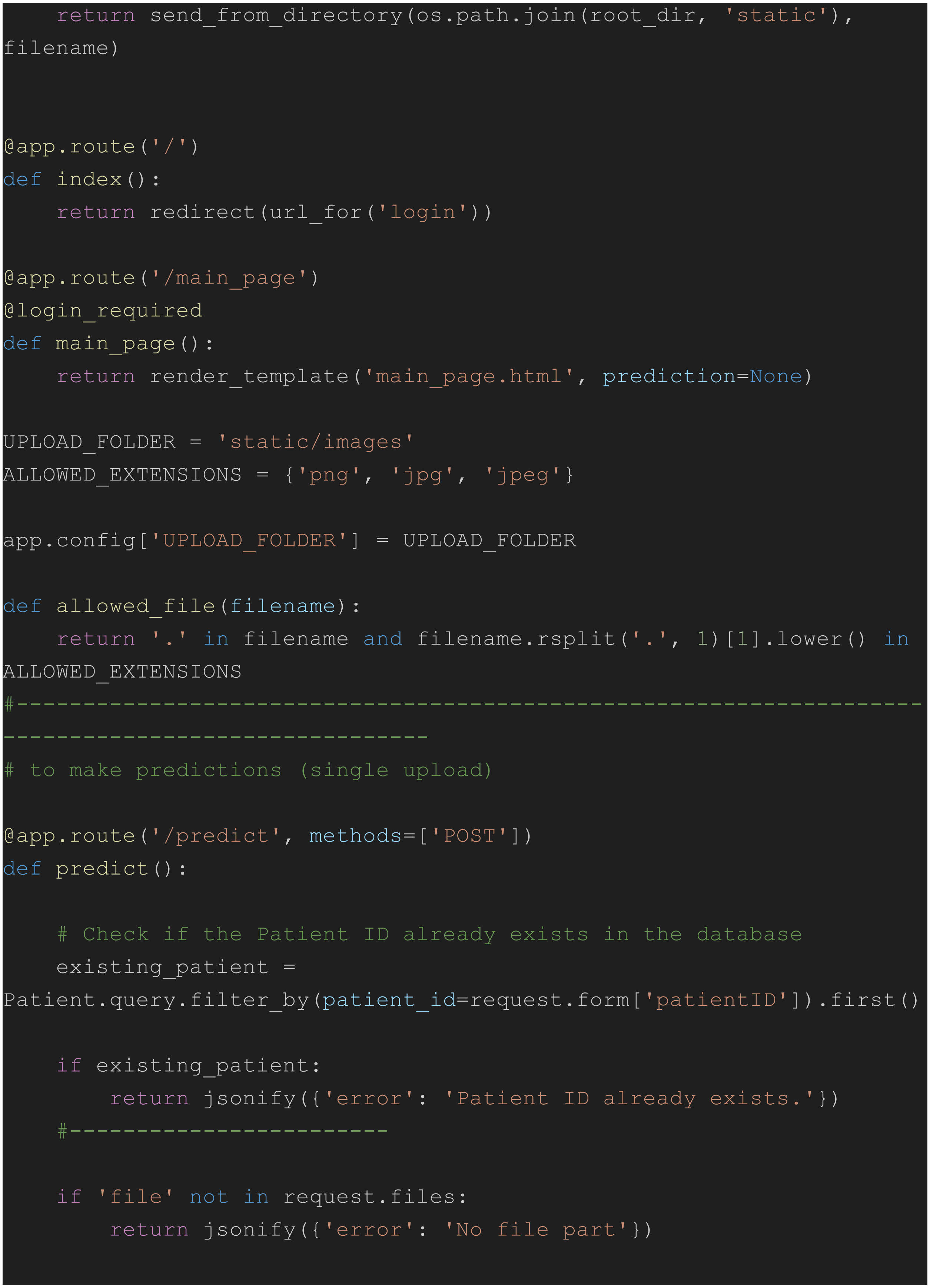
```
from flask import Flask, render_template, request, jsonify,
send_from_directory
from keras.models import load_model
from keras.preprocessing import image
import numpy as np
import os
from flask_sqlalchemy import SQLAlchemy
from werkzeug.utils import secure_filename
from datetime import datetime
from flask import send_file
import csv
from PIL import Image,UnidentifiedImageError
#DB imports
from io import BytesIO
```



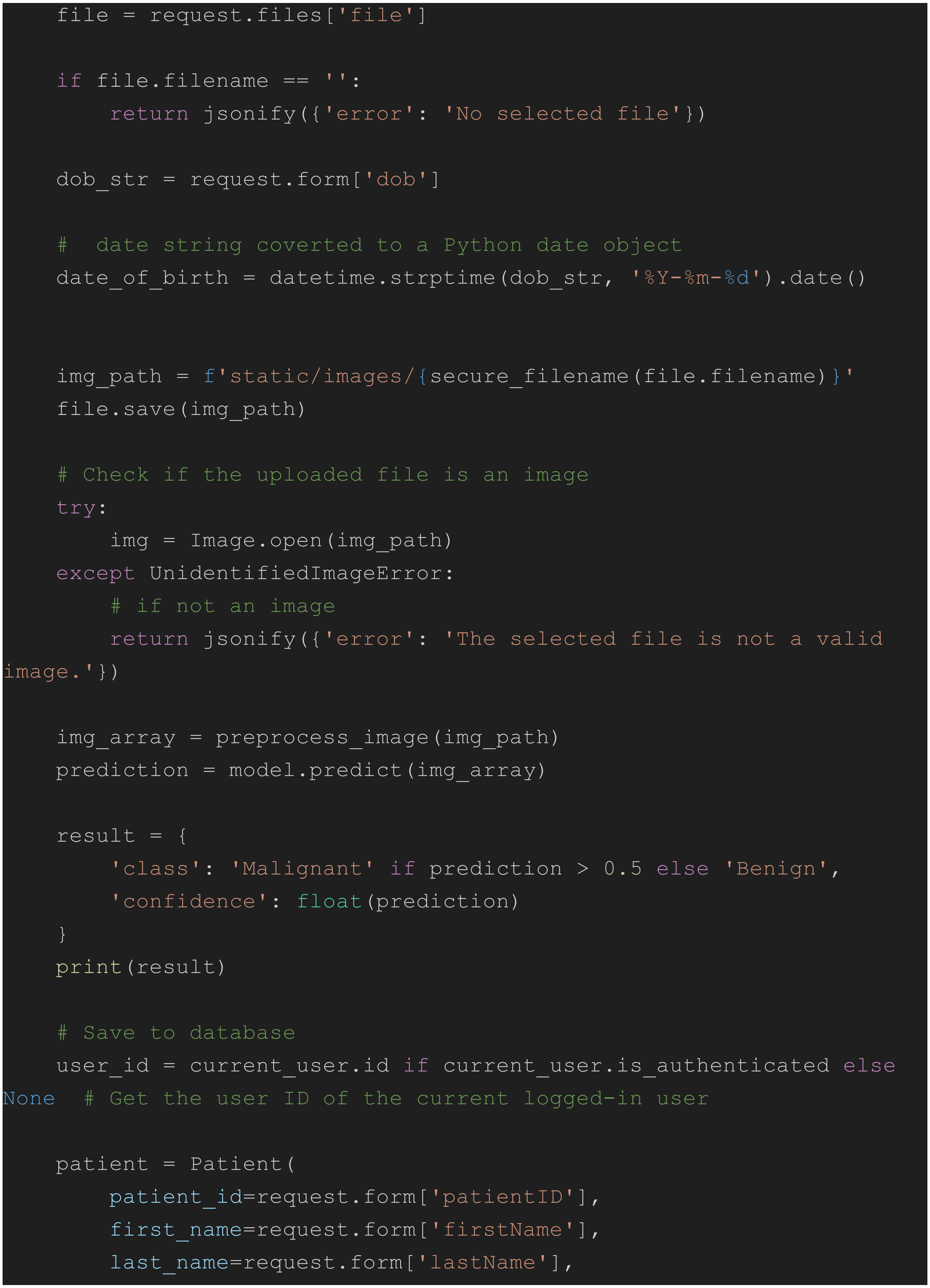
```
from reportlab.lib.pagesizes import letter
from reportlab.lib import colors
from reportlab.lib.styles import getSampleStyleSheet
from reportlab.platypus import SimpleDocTemplate, Table, TableStyle,
Paragraph
from reportlab.platypus import Spacer
#authentication imports
from flask import redirect, 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
```



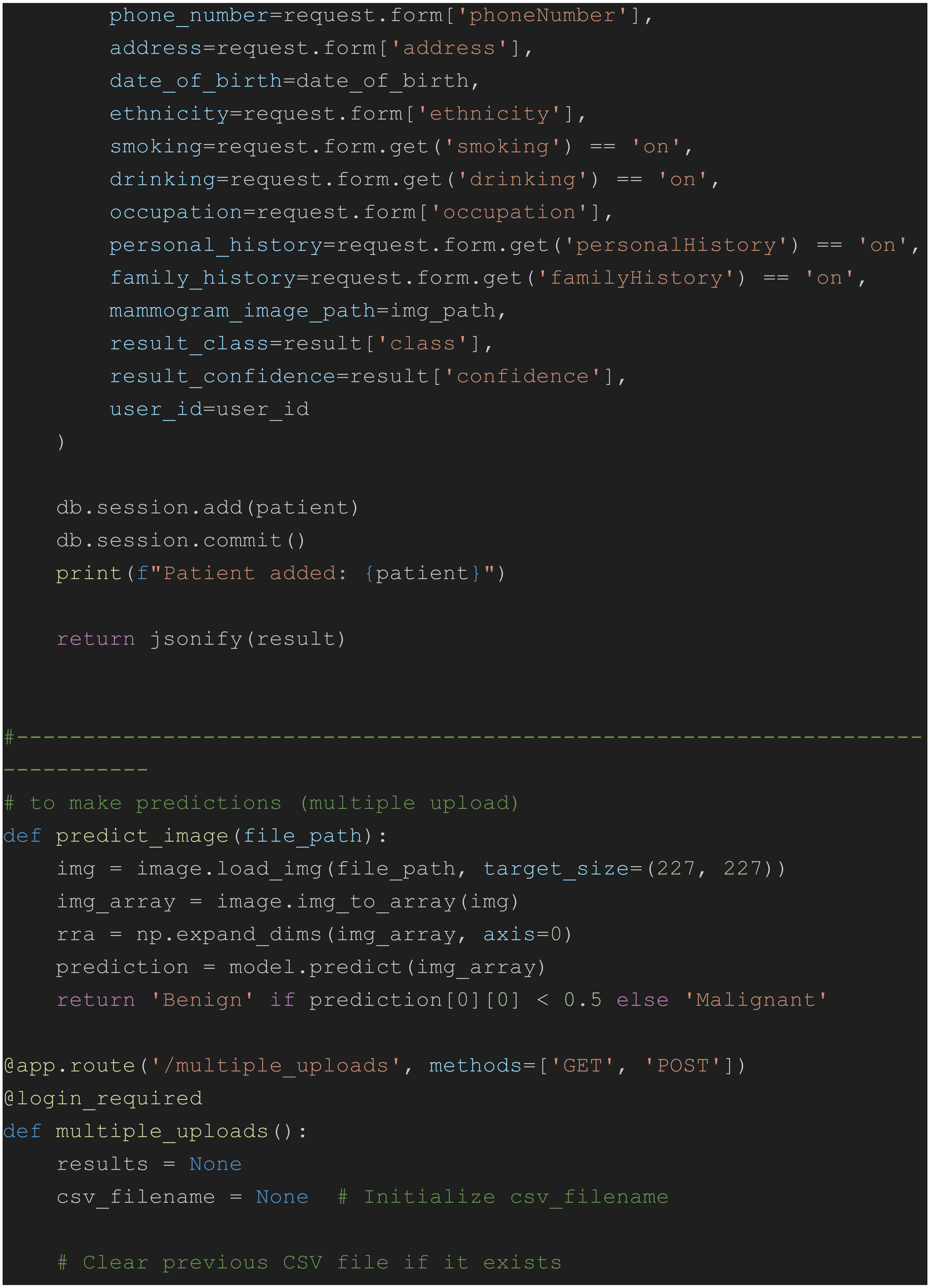
```
class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    employee_id = db.Column(db.String(50), unique=True,
nullable=False)
    first_name = db.Column(db.String(50), nullable=False)
    last_name = db.Column(db.String(50), nullable=False)
    email = db.Column(db.String(120), 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())
```



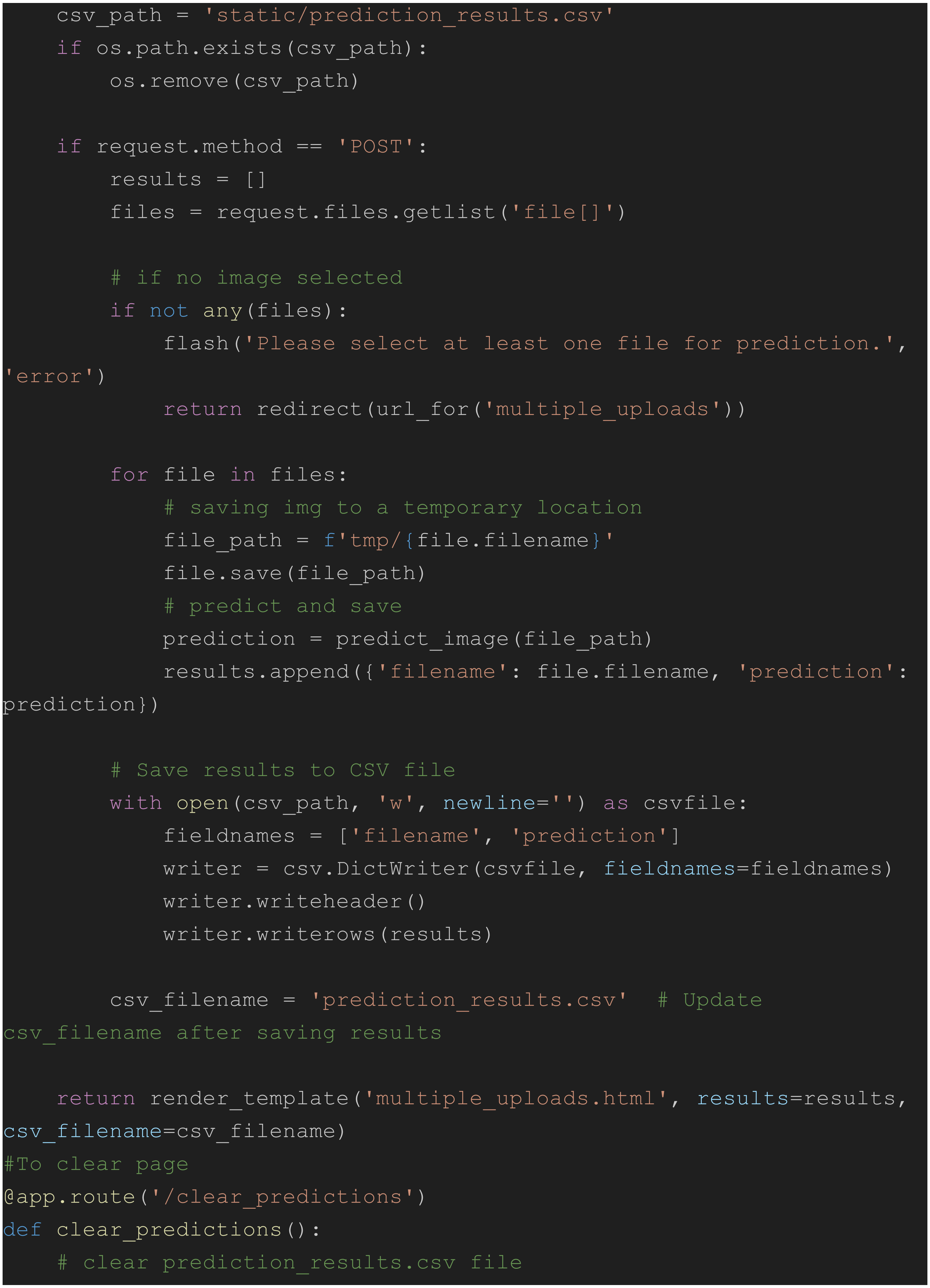
```
    return send_from_directory(os.path.join(root_dir, 'static'),
filename)


@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'})
```



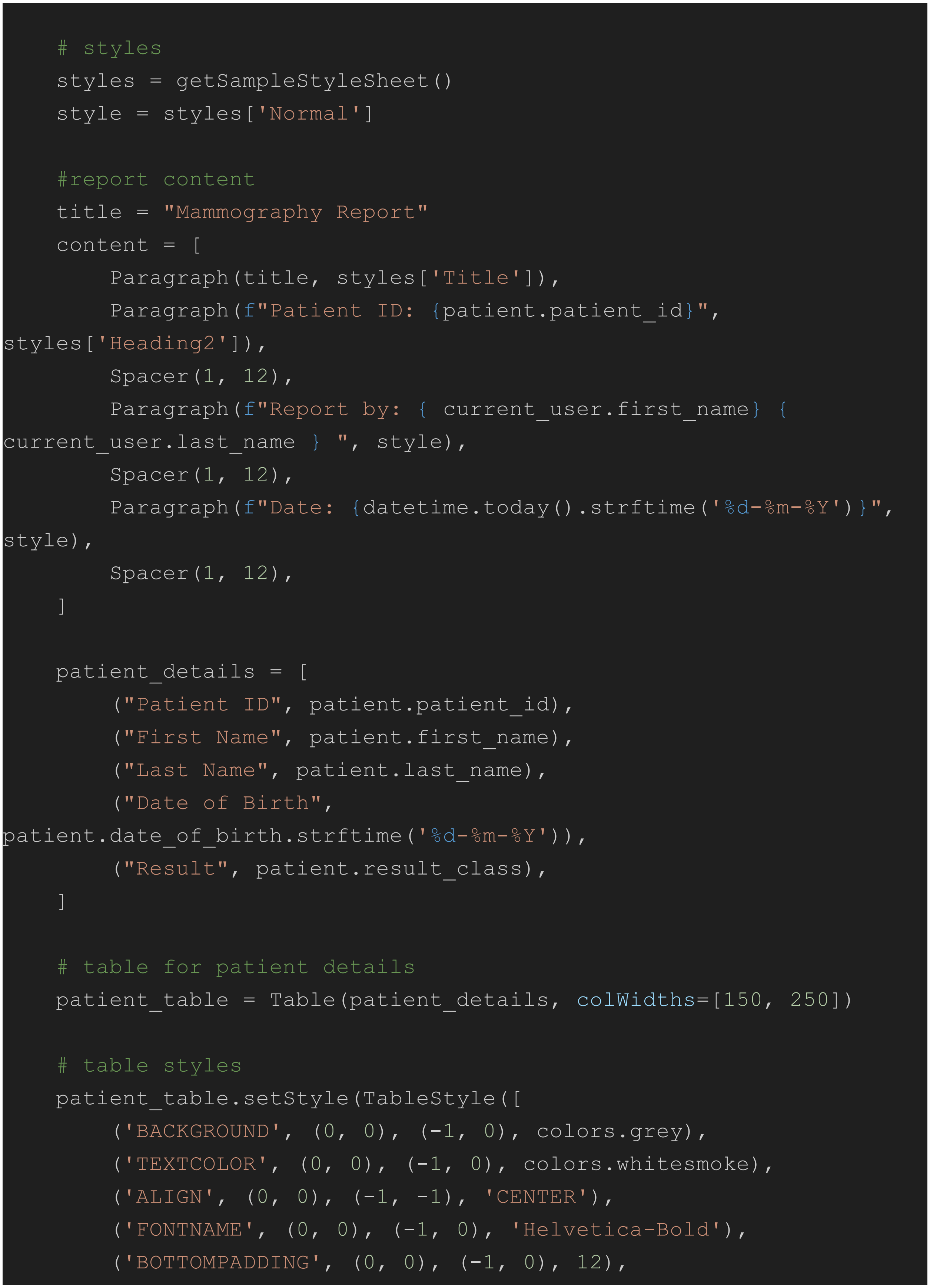
```
    file = request.files['file']

    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'],
```



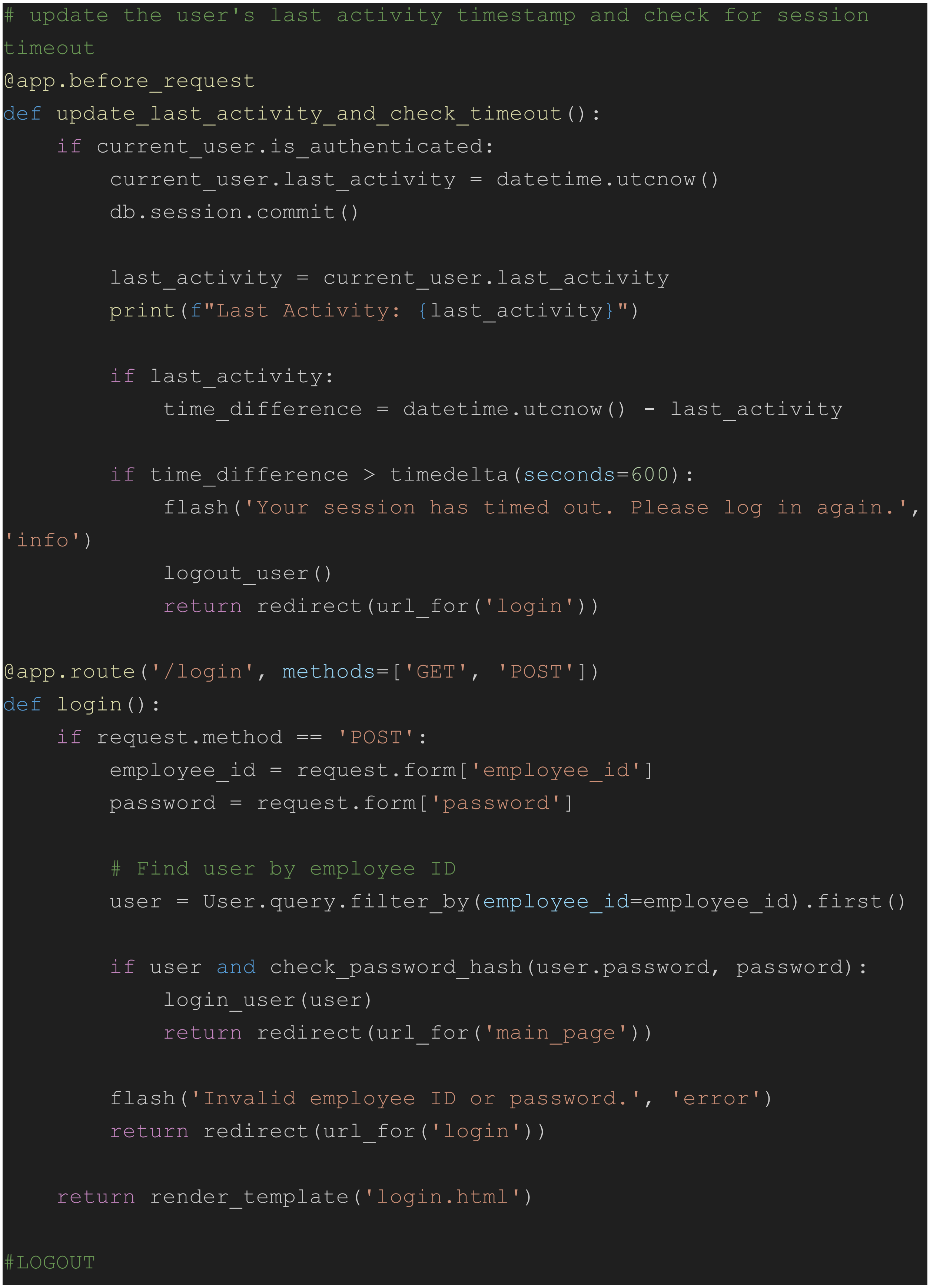
```
        phone_number=request.form['phoneNumber'],
        address=request.form['address'],
        date_of_birth=date_of_birth,
        ethnicity=request.form['ethnicity'],
        smoking=request.form.get('smoking') == 'on',
        drinking=request.form.get('drinking') == 'on',
        occupation=request.form['occupation'],
        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
```



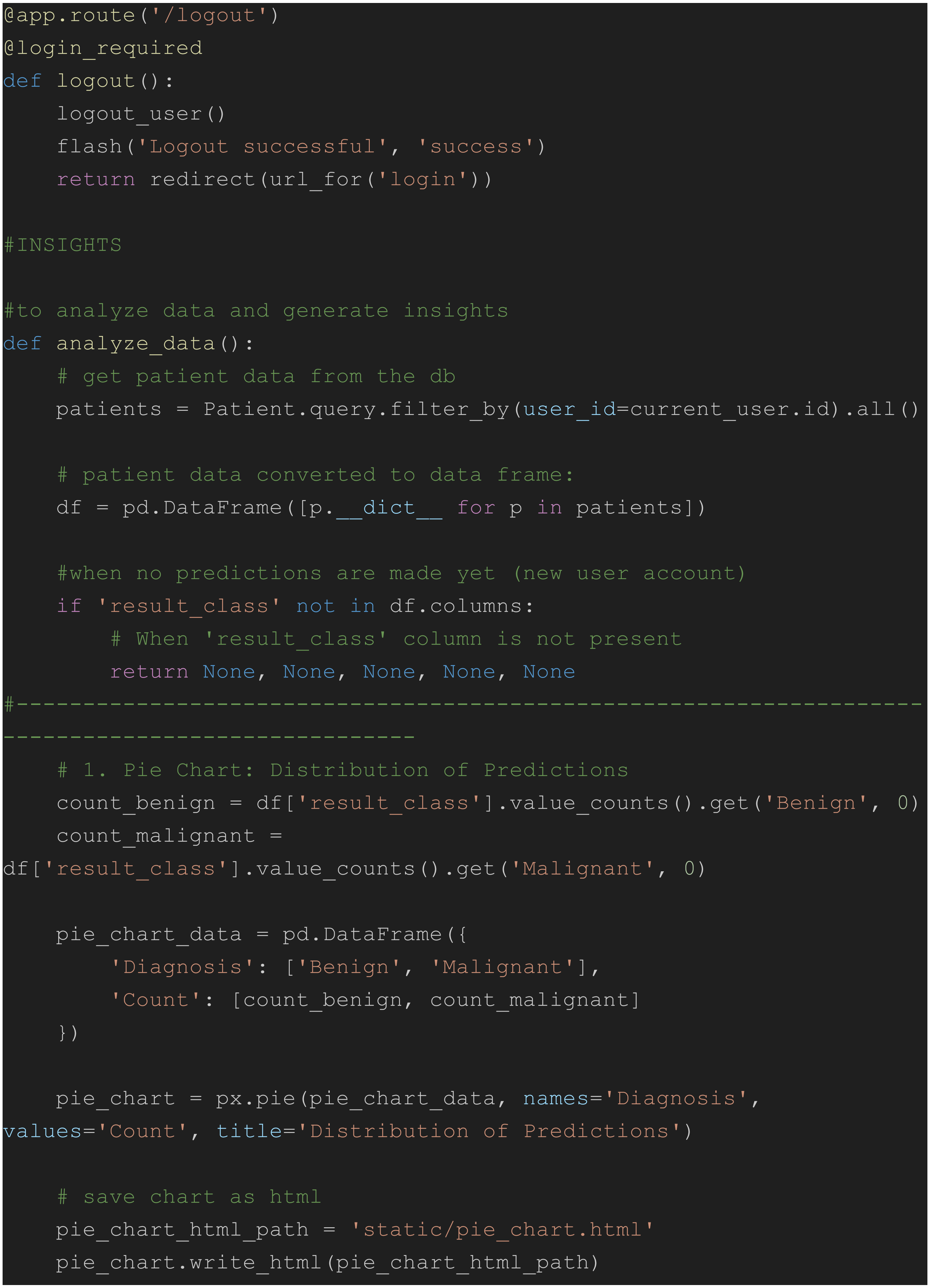
```
    csv_path = 'static/prediction_results.csv'
    if os.path.exists(csv_path):
        os.remove(csv_path)

    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)
```



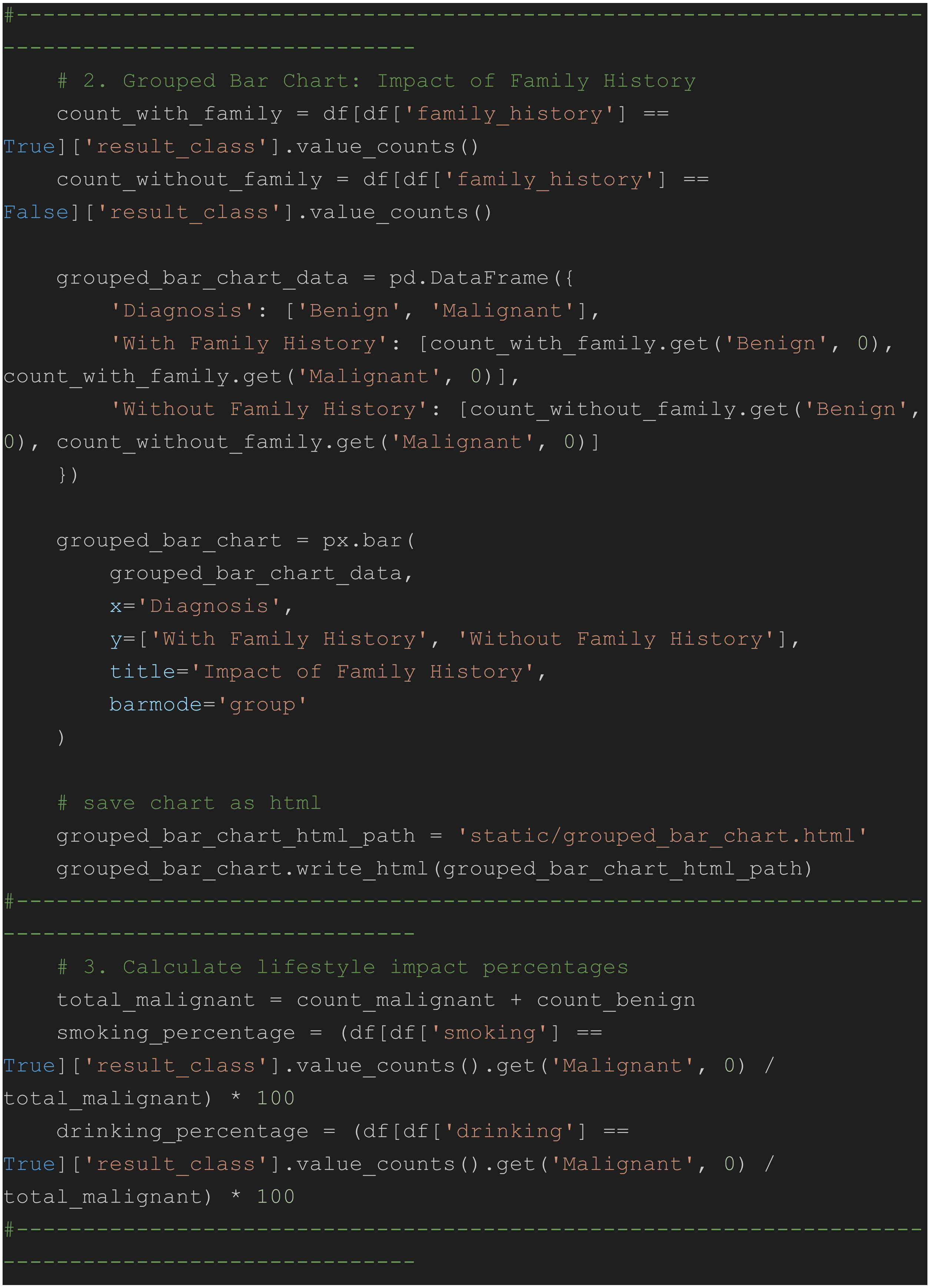
```
    # styles
    styles = getSampleStyleSheet()
    style = styles['Normal']

    #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))
```



```python
# 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)
```