Project title:

DIGITAL ATTENDANCE SYSTEM

Course name	**:**	**ITC LAB**
Semester	**:**	**FIRST**
Instructor	**:**	**MS ALMEERAH**
University	**:**	**SZABIST**
Submission date	**:**	**29\12\2025**
Group Members	**:**	**RAHEEL-2512407**
		ALI-2512428
		FAIZAN-2512434
		ROHAAN-2512451
Group number	**:**	**11**

Project Overview

The Digital Attendance System is a web-based application developed to record and manage student attendance efficiently. It allows users to enter student details, mark attendance as present or absent, and store records digitally using browser local storage.

Problem Statement

Manual attendance systems are time-consuming, error-prone, and difficult to maintain. This project aims to provide a simple digital solution to record, store, and manage attendance data accurately.

Target Users

The system is designed for teachers, educational institutions, and students for easy attendance management and academic use.

Tech Stack Used

- **HTML** – For structuring the web page
- **CSS** – For styling and responsive design
- **JavaScript (Vanilla JS)** – For handling logic, DOM manipulation, and data management
- **Browser Local Storage** – For storing attendance records on the client side

Project Repository (GitHub)

Public GitHub repository link:

https://github.com/raheel10000/ITC-FINALPROJECT-GROUP11.git

Project UI & Working Explanation

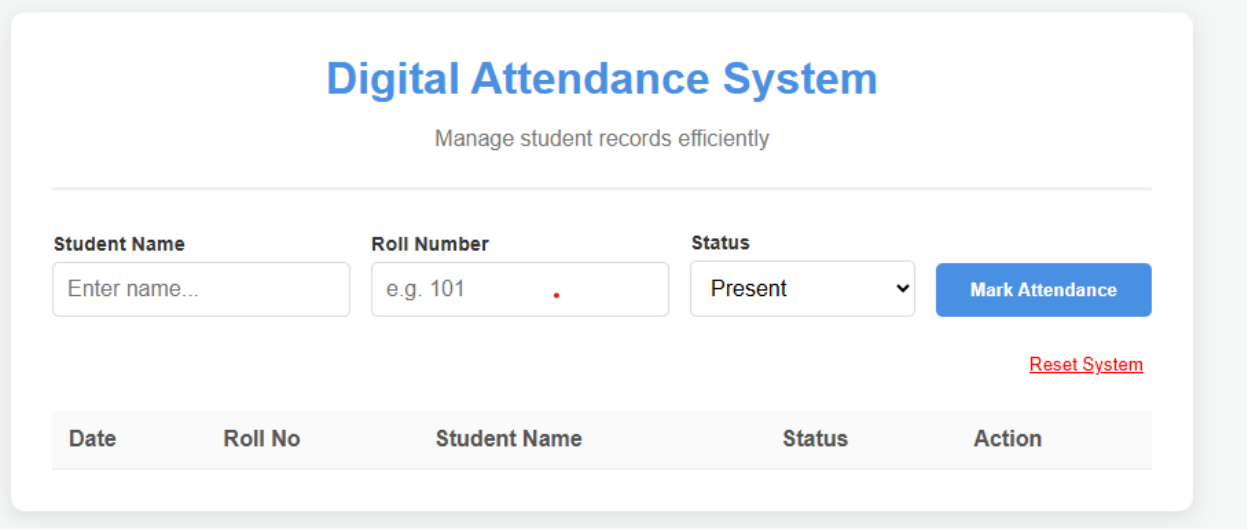
This section explains the user interface and working of the Digital Attendance System in a step-by-step manner.

---

1. Main Screen

(Insert Screenshot: Main Screen UI)

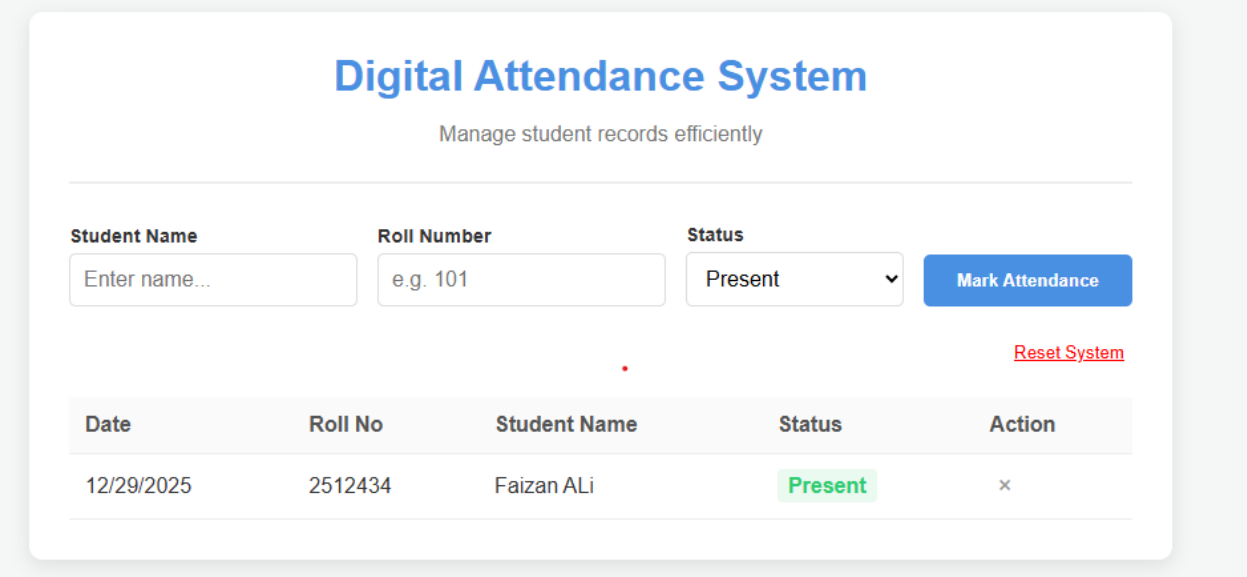
Explanation:
The main screen displays the title *Digital Attendance System* along with input fields for student name, roll number, and attendance status. A table is shown below to display all attendance records. The interface is simple, clean, and easy to use.



---

2. User Input

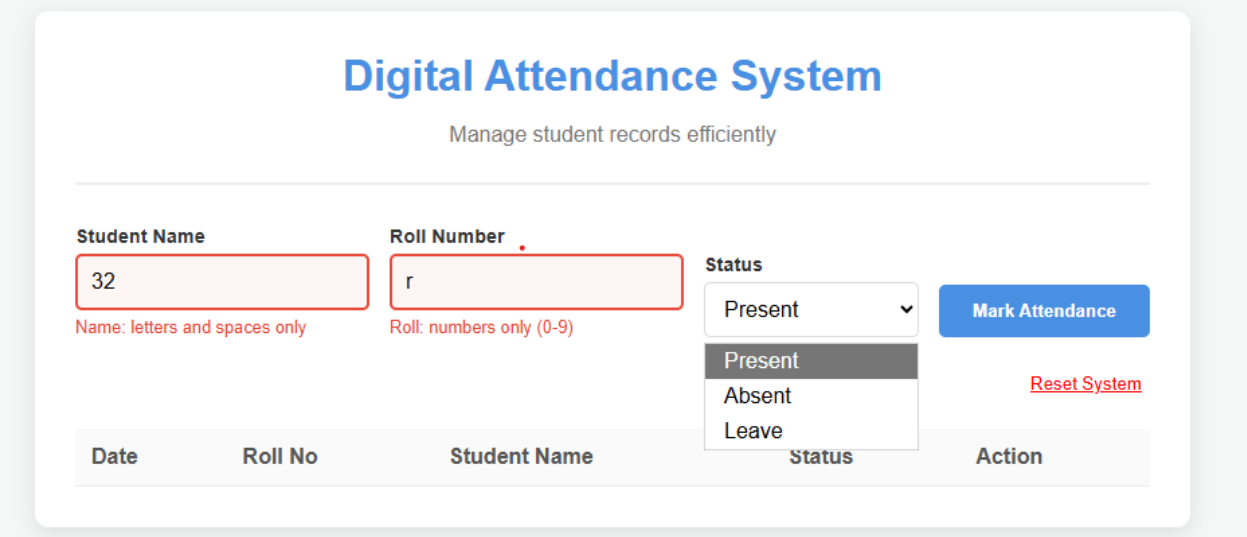
Explanation:
The user enters the student name, roll number, and selects the attendance status (Present or Absent). After filling in the details, the user clicks the **"Mark Attendance"** button to submit the data.



---

3. Processing Logic

Explanation:
When the button is clicked, JavaScript validates the input fields. If the inputs are valid, the system creates a new attendance record with the current date. This data is then stored in the browser's **Local Storage** for persistent access.

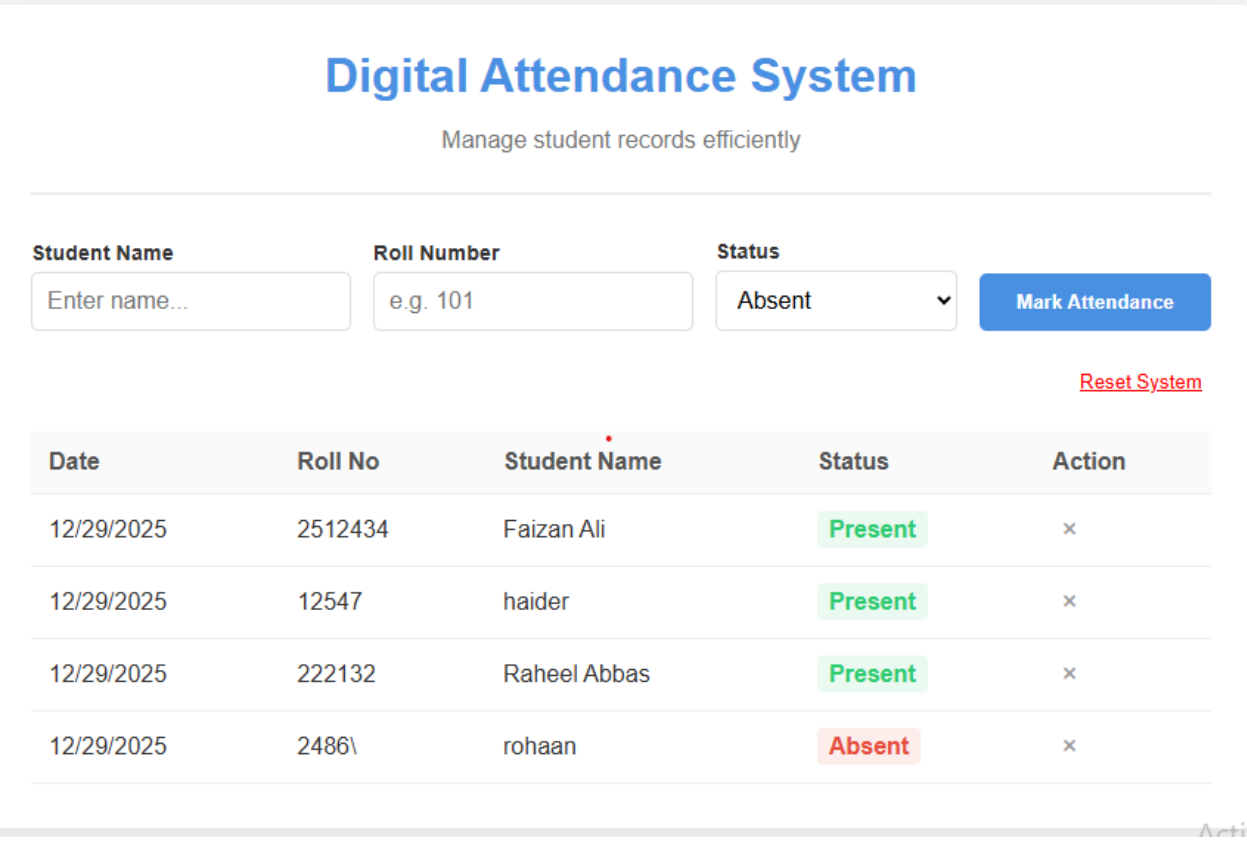


---

4. Output Display

Explanation:
The stored attendance records are displayed in a table format. Each row shows the

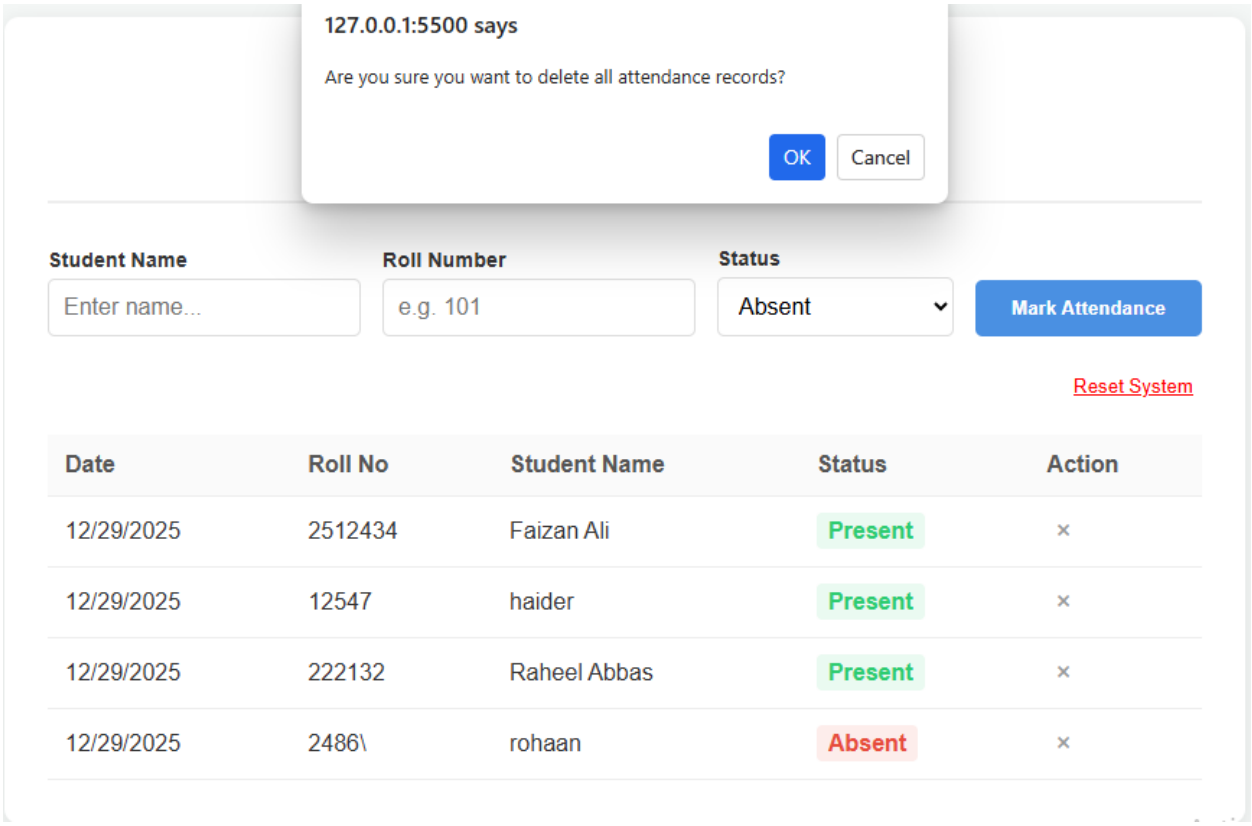
date, roll number, student name, and attendance status. Present and Absent statuses are highlighted using different colors for better visibility.



5. Other Features

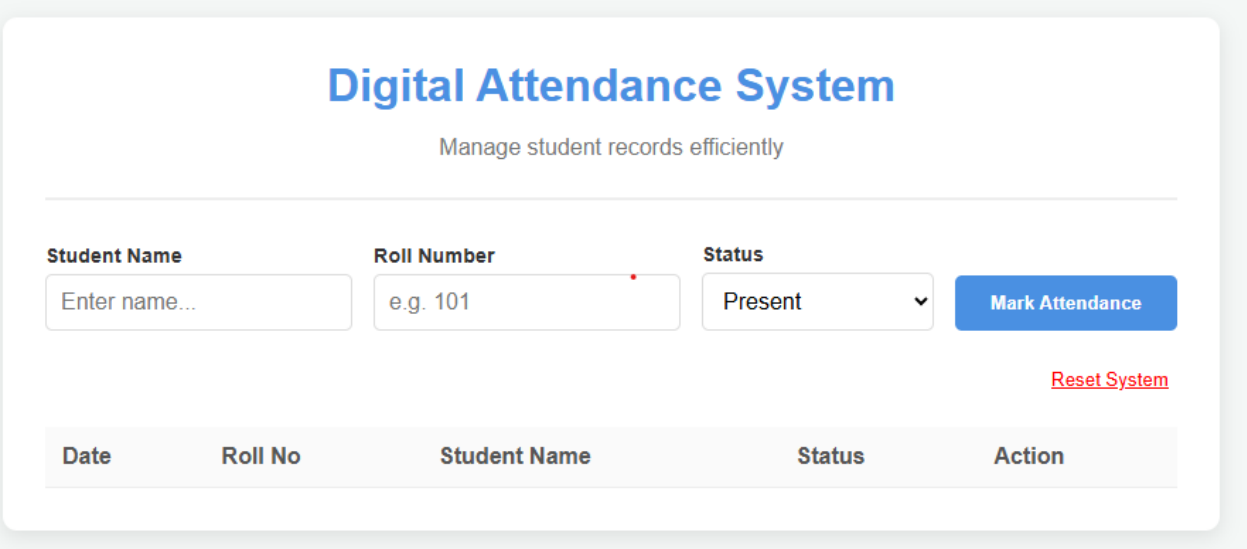
Explanation:

- Each record has a **delete (×)** button to remove individual entries.
- A **Reset System** option allows the user to delete all attendance records at once after confirmation.
- The data remains saved even after refreshing the page due to Local Storage usage.



-

After reset the Table:



Functionalities & Code Explanation

This section explains the main JavaScript functions used in the Digital Attendance System.

---

1. Save Attendance Data

Function Name: `saveData()`

Code Snippet:

```
function saveData() {
    var name = document.getElementById('sName').value.trim();
    var roll = document.getElementById('sRoll').value.trim();
    var status = document.getElementById('sStatus').value;

    if (name === '' || roll === '') {
        alert("Please enter both Name and Roll Number.");
        return;
    }
}
```

Explanation:
This function collects the student name, roll number, and attendance status from the input fields. It checks whether the name and roll number are entered before proceeding further.

---

2. Create Attendance Record

Function Name: `saveData()`

Code Snippet:

```
var newRecord = {
    id: Date.now(),
    date: new Date().toLocaleDateString(),
    name: name,
    roll: roll,
    status: status
};
```

Explanation:
This code creates an object for each attendance entry. It stores a unique ID, the current date, student details, and attendance status.

---

3. Store Data in Local Storage

Function Name: `saveData()`

Code Snippet:

```
var currentData = localStorage.getItem('myAttendanceData');
var list = [];

if(currentData) {
    list = JSON.parse(currentData);
}

list.push(newRecord);
localStorage.setItem('myAttendanceData', JSON.stringify(list));
```

Explanation:
This part retrieves existing attendance data from LocalStorage, adds the new record to the list, and saves the updated list back into LocalStorage.

---

4. Display Attendance Records

Function Name: `showData()`

Code Snippet:

```
function showData() {
    var tableBody = document.getElementById('list-body');
    var currentData = localStorage.getItem('myAttendanceData');
    var list = [];

    if(currentData) {
        list = JSON.parse(currentData);
    }

    tableBody.innerHTML = '';
}
```

Explanation:
This function fetches attendance records from LocalStorage and clears the table body before displaying the updated data.

---

5. Attendance Status Styling

Function Name: `showData()`

Code Snippet:

```
if(item.status === 'Present') {
    badgeClass = 'present';
} else {
    badgeClass = 'absent';
}
```

Explanation:
This logic applies different styles based on attendance status. Present students are shown in green and absent students in red.

---

6. Remove Individual Attendance Record

Function Name: `removeRow(id)`

Code Snippet:

```
function removeRow(id) {
    var currentData = localStorage.getItem('myAttendanceData');
    var list = JSON.parse(currentData);
    var newList = [];

    for(var i = 0; i < list.length; i++) {
        if(list[i].id !== id) {
            newList.push(list[i]);
        }
    }
}
```

Explanation:
This function removes a specific attendance record by comparing the unique ID and creating a new list without the selected record.

---

7. Update LocalStorage After Deletion

Function Name: `removeRow(id)`

Code Snippet:

```
localStorage.setItem('myAttendanceData', JSON.stringify(newList));
showData();
```

Explanation:
After deleting a record, the updated attendance list is saved back into LocalStorage and the table is refreshed.

---

8. Reset Entire Attendance System

Function Name: `resetSystem()`

Code Snippet:

```
function resetSystem() {
```

```
    var check = confirm("Are you sure you want to delete all attendance
records?");
    if(check) {
        localStorage.removeItem('myAttendanceData');
        showData();
    }
}
```

Explanation:
This function clears all attendance records from Local Storage after user confirmation.

---

9. Load Data on Page Load

Function Name: `window.onload`

Code Snippet:

```
window.onload = function() {
    showData();
};
```

Explanation:
This ensures that saved attendance records are automatically displayed when the page is loaded or refreshed.

Complete Source Code

HTML Code

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Digital Attendance System</title>
   <style>
      body {
         font-family: sans-serif;
         background-color: #f4f7f6;
         color: #333;
         margin: 0;
```

```
    padding: 20px;
    display: flex;
    justify-content: center;
}
.main-container {
    width: 100%;
    max-width: 800px;
    background: #ffffff;
    padding: 30px;
    border-radius: 10px;
    box-shadow: 0 4px 15px rgba(0,0,0,0.1);
}
.header-section {
    text-align: center;
    margin-bottom: 30px;
    border-bottom: 2px solid #eee;
    padding-bottom: 10px;
}
h1 { margin: 0; color: #4a90e2; }
p { color: #777; }

.input-area {
```

```
    display: grid;
    grid-template-columns: 1fr 1fr 1fr auto;
    gap: 15px;
    margin-bottom: 25px;
    align-items: end;
}

.field { display: flex; flex-direction: column; }
label { font-size: 0.9rem; margin-bottom: 5px; font-weight: 600; }
input, select {
    padding: 10px;
    border: 1px solid #ddd;
    border-radius: 5px;
    font-size: 1rem;
}
.invalid-field {
    border: 2px solid #e74c3c !important;
    background-color: rgba(231, 76, 60, 0.05);
}

.error-message {
    color: #e74c3c;
```

```
    font-size: 0.8rem;
    margin-top: 5px;
    display: none;
}

.btn-add {
    background-color: #4a90e2;
    color: white;
    border: none;
    padding: 12px 25px;
    border-radius: 5px;
    cursor: pointer;
    font-weight: bold;
}

.btn-add:hover { background-color: #357abd; }

.table-box { overflow-x: auto; }

table {
    width: 100%;
    border-collapse: collapse;
```

```
    margin-top: 10px;
}

th, td {
    text-align: left;
    padding: 12px;
    border-bottom: 1px solid #eee;
}

th { background-color: #fafafa; color: #555; }

.present {
    color: #2ecc71;
    font-weight: bold;
    background: rgba(46, 204, 113, 0.1);
    padding: 4px 8px;
    border-radius: 4px;
}

.absent {
    color: #e74c3c;
    font-weight: bold;
```

```
    background: rgba(231, 76, 60, 0.1);
    padding: 4px 8px;
    border-radius: 4px;
}

.leave {
    color: #f39c12;
    font-weight: bold;
    background: rgba(243, 156, 18, 0.1);
    padding: 4px 8px;
    border-radius: 4px;
}
.invalid-cell {
    color: #e74c3c !important;
    background: rgba(231, 76, 60, 0.1) !important;
    padding: 4px 8px;
    border-radius: 4px;
    font-weight: bold;
    border-left: 3px solid #e74c3c;
}

.remove-btn {
```

```
            background: none;
            border: none;
            color: #999;
            cursor: pointer;
            font-size: 1.2rem;
        }
        .remove-btn:hover { color: #e74c3c; }

        @media (max-width: 600px) {
            .input-area { grid-template-columns: 1fr; }
            .btn-add { width: 100%; }
        }
    </style>
</head>
<body>
    <div class="main-container">
        <div class="header-section">
            <h1>Digital Attendance System</h1>
            <p>Manage student records efficiently</p>
        </div>

        <div class="input-area">
```

```
        <div class="field">
            <label>Student Name</label>
            <input type="text" id="sName" placeholder="Enter name...">
            <div class="error-message" id="nameError">Name: letters and spaces only</div>
        </div>

        <div class="field">
            <label>Roll Number</label>
            <input type="text" id="sRoll" placeholder="e.g. 101">
            <div class="error-message" id="rollError">Roll: numbers only (0-9)</div>
        </div>

        <div class="field">
            <label>Status</label>
            <select id="sStatus">
                <option value="Present">Present</option>
                <option value="Absent">Absent</option>
                <option value="Leave">Leave</option>
            </select>
        </div>
```

```
        <button class="btn-add" onclick="saveData()">Mark Attendance</button>

    </div>


    <div style="margin-bottom: 15px; text-align: right;">

        <button onclick="resetSystem()" style="color: red; background: none; border: none; cursor: pointer; text-decoration: underline;">Reset System</button>

    </div>


    <div class="table-box">

      <table>

        <thead>

          <tr>

            <th>Date</th>

            <th>Roll No</th>

            <th>Student Name</th>

            <th>Status</th>

            <th>Action</th>

          </tr>

        </thead>

        <tbody id="list-body"></tbody>

      </table>
```

```
        </div>
    </div>


    <script>
        window.onload = function() {
            showData();
            setupValidation();
        };
        function setupValidation() {
            document.getElementById('sName').addEventListener('input',
validateName);
            document.getElementById('sRoll').addEventListener('input',
validateRoll);
        }
        function validateName() {
            var nameInput = document.getElementById('sName');
            var nameError = document.getElementById('nameError');
            var nameValue = nameInput.value.trim();


            if (nameValue === '') {
                nameInput.classList.remove('invalid-field');
                nameError.style.display = 'none';
                return true;
```

```
    }
    var isValid = /^[A-Za-z\s.'-]+$/.test(nameValue);

    if (!isValid) {
        nameInput.classList.add('invalid-field');
        nameError.style.display = 'block';
        return false;
    } else {
        nameInput.classList.remove('invalid-field');
        nameError.style.display = 'none';
        return true;
    }
}
function validateRoll() {
    var rollInput = document.getElementById('sRoll');
    var rollError = document.getElementById('rollError');
    var rollValue = rollInput.value.trim();

    if (rollValue === '') {
        rollInput.classList.remove('invalid-field');
        rollError.style.display = 'none';
        return true;
```

```
    }
    var isValid = /^\d+$/.test(rollValue);

    if (!isValid) {
        rollInput.classList.add('invalid-field');
        rollError.style.display = 'block';
        return false;
    } else {
        rollInput.classList.remove('invalid-field');
        rollError.style.display = 'none';
        return true;
    }
}
function validateAll() {
    return validateName() && validateRoll();
}
function saveData() {
    var name = document.getElementById('sName').value.trim();
    var roll = document.getElementById('sRoll').value.trim();
    var status = document.getElementById('sStatus').value;

    if (name === '' || roll === '') {
```

```
    alert("Please enter both Name and Roll Number.");
    return;
}

if (!validateAll()) {
    alert("Please correct the errors before saving.");
    return;
}
var newRecord = {
    id: Date.now(),
    date: new Date().toLocaleDateString(),
    name: name,
    roll: roll,
    status: status,
    isValidName: /^[A-Za-z\s.'-]+$/.test(name),
    isValidRoll: /^\d+$/.test(roll)
};
var currentData = localStorage.getItem('myAttendanceData');
var list = [];
if(currentData) {
    list = JSON.parse(currentData);
}
```

```
    list.push(newRecord);
    localStorage.setItem('myAttendanceData', JSON.stringify(list));

    document.getElementById('sName').value = '';
    document.getElementById('sRoll').value = '';
    document.getElementById('sName').classList.remove('invalid-field');
    document.getElementById('sRoll').classList.remove('invalid-field');
    document.getElementById('nameError').style.display = 'none';
    document.getElementById('rollError').style.display = 'none';
    showData();
}
function showData() {
    var tableBody = document.getElementById('list-body');
    var currentData = localStorage.getItem('myAttendanceData');
    var list = [];

    if(currentData) {
        list = JSON.parse(currentData);
    }

    tableBody.innerHTML = '';
```

```
        for(var i = 0; i < list.length; i++) {
            var row = document.createElement('tr');
            var item = list[i];
            var statusClass = '';

            if(item.status === 'Present') {
                statusClass = 'present';
            } else if(item.status === 'Absent') {
                statusClass = 'absent';
            } else if(item.status === 'Leave') {
                statusClass = 'leave';
            }
            var nameClass = item.isValidName ? '' : 'invalid-cell';
            var rollClass = item.isValidRoll ? '' : 'invalid-cell';

            row.innerHTML = '<td>' + item.date + '</td>' +
                        '<td><span class="' + rollClass + '">' + item.roll +
'</span></td>' +
                        '<td><span class="' + nameClass + '">' + item.name +
'</span></td>' +
                        '<td><span class="' + statusClass + '">' + item.status +
'</span></td>' +
                        '<td><button class="remove-btn" onclick="removeRow(' +
item.id + ')">&times;</button></td>';
```

```
        tableBody.appendChild(row);
    }
}
function removeRow(id) {
    if(!confirm("Delete this record?")) return;

    var currentData = localStorage.getItem('myAttendanceData');
    var list = JSON.parse(currentData);
    var newList = [];

    for(var i = 0; i < list.length; i++) {
        if(list[i].id !== id) {
            newList.push(list[i]);
}
    }
    localStorage.setItem('myAttendanceData', JSON.stringify(newList));
    showData();
}
function resetSystem() {
    var check = confirm("Delete all attendance records?");
    if(check) {
```

```
            localStorage.removeItem('myAttendanceData');
            showData();
        }
    }
  </script>
</body>
</html>
```

Conclusion

Learning Outcomes:

- Learned how to build a fully functional web application using **HTML, CSS, and JavaScript**.
- Gained practical experience in **DOM manipulation**, **event handling**, and **data storage with LocalStorage**.
- Understood how to dynamically update the UI based on user input and data changes.
- Improved skills in organizing code for readability and maintainability.

Challenges:

- Ensuring data persisted correctly in **LocalStorage** across page reloads.
- Handling edge cases such as empty input fields or duplicate records.
- Dynamically updating the table and applying status badges without reloading the page.

Role of JavaScript in Building Logic:

- JavaScript was essential for **validating user input**, **storing/retrieving attendance data**, and **dynamically generating the table**.
- It enabled implementing **core functionalities** like adding, removing, and resetting attendance records.
- Using JavaScript made the system **interactive**, efficient, and fully client-side without requiring a backend database.