**Student Management System Overview**

1. **Purpose:**

The system allows users to add, edit, and delete student details such as name, ID, email, contact number, and course. All data is stored persistently in the browser using **localStorage**.

1. **Core Features:**

**• Add Student**: Users can add student details by filling form.

**• Edit Student:** Users can edit student details by selecting an edit option.

**• Delete Student**: Users can delete student details by clicking on delete.

**• Data Storage**: All data stored in local storage so data show after the page

refreshed.

1. **Validation:**

**• Student Name:** Accepts only letters and space.

**• Student ID:** Accepts only numeric values.

**• Email:** Accepts only valid email format (e.g.,

sandeepjobs03@gmail.com).

**• Contact Number: Accepts only numeric values.**

**• Course: Accepts any non-empty value.**

1. **Flow of Operations:**

**• Add Details:**

• When the user clicks the "Add Student" button, the system validates

the form fields.

• If any field is empty or contains invalid data (like an invalid email or

non-numeric student ID), the system alerts the user and prevents the

student from being added.

• If the form is valid, a new student object is created and added to the

list stored in localStorage.

**• Edit Student:**

• When the user clicks the "Edit" button next to a student's details, the

student's data is pre-populated in the input fields..

• The user can modify the details, and once they save, the changes are

updated in localStorage.

• The previous student entry is deleted from the list before updating it

With new data.

**• Delete Student:**

• When the user clicks the "Delete" button next to a student's

record, that student's entry is removed from localStorage.

• The user can modify the details, and once they save, the

changes are updated in localStorage.

• The list is then re-rendered to reflect the updated data.

**5. Data Heandling:**

• **localStorage** is used to store the student data. The system

retrieves the data from **localStorage** whenever the page is

loaded, ensuring that all previously added students are

displayed.

• **Array Operations** are used to manage the list of students:

• push() to add a new student..

• splice() to remove a student.

• The data is stored in JSON format, using JSON.stringify()

when saving and JSON.parse() when retrieving.

**6. User Interface:**

• The system provides a user-friendly interface where users can see a

list of all students and interact with their data through **Add**, **Edit**,

and **Delete** buttons.

• The form is designed to be responsive and guides the user

with appropriate field validations.

**7. Error Handling:**

• The system provides feedback to the user for invalid inputs via alert

messages.

• It also handles empty or incorrect data gracefully by preventing

1. **Summary of Key Concepts**
2. **localStorage:** Used for storing data across browser sessions.
3. **Form Validation**: Ensures correct data types and values are entered by the user.
4. **CRUD Operations:**

• **Create**: Add new student records.

• **Read**: Display stored student records.

• **Update**: Edit existing student records.

• **Delete**: Remove student records.

1. **Data Flow**: User input is validated, processed, stored in localStorage, and

displayed dynamically on the UI.

1. **User Interaction**: The system allows users to interact with the data through

buttons to add, edit, and delete student details.

**Conclusion**

This project is a simple but effective application for managing student data. It highlights fundamental web development concepts, including DOM manipulation, localStorage, form validation, and CRUD operations**.**

Note: This App is not Fully Responsive, please open in large devices for better experience.

**GitHub Link:** [**https://github.com/SandyBhai03/Internshala-**](https://github.com/SandyBhai03/Internshala-)

**Assignments/tree/main/Assignment-Course4/DOM%20JS**

