DOM Assignment: Student Registration System:

Student Registration System is designed to manage student records such as Student name, student ID, email ID, and Contact No. It provides functionalities for editing and deleting records also.  
This assignment consists of

* HTML
* CSS
* JavaScript

HTML

The HTML code creates a basic structure for the Student Registration System which consists a header, navigation bar, registration form, and a section to display enrolled students.

Head Section:  
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Student Registration System</title>

    <link rel="stylesheet" href="style.css">

</head>

* It contains meta tags for character encoding and viewport settings.
* Style.css to link the css page to html file.
* Title of the page is Student Registration System.

Body section:  
Header Section:

<header class="header-title">

        <h1>STUDENT   REGISTRATION   SYSTEM</h1>

        <p>Student registration system is designed to manage student records such as Student name, Student ID, Email ID, and Contact No. It provides functionalities for editing and deleting records</p>

</header>

* It displays the main title of the of the application and a brief description summarizing the system's functionalities

Navigation Bar:

 <nav class="navbar">

        <ul>

            <li><a id="formLink">REGISTRATION FORM</a></li>

            <li><a id="listLink">ENROLLED STUDENTS</a></li>

        </ul>

    </nav>

* Navigation bar consists of two links

1. Registration form that contains form details
2. Enrolled Students data.

Main content:

<main>

        <section class="container">

            <!-- Registration Form -->

            <section id="formSection" class="form-container">

                <h1>REGISTRATION FORM</h1>

                <label for="name">Student Name:</label>

                <input type="text" id="name" placeholder="Enter student name">

                <label for="id">Student ID:</label>

                <input type="text" id="id" placeholder="Enter student ID">

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

                <input type="email" id="email" placeholder="Enter student email">

                <label for="contact">Contact Number:</label>

                <input type="text" id="contact" placeholder="Enter contact number">

                <button id="addBtn">Add Student</button>

            </section>

            <!-- Enrolled Students -->

            <section id="listSection" class="table-container" style="display: none;">

                <h1>Enrolled Students</h1>

                <section class="scroll-container">

                    <table>

                        <thead>

                            <tr>

                                <th>Name</th>

                                <th>Student ID</th>

                                <th>Email</th>

                                <th>Contact</th>

                                <th>Actions</th>

                            </tr>

                        </thead>

                        <tbody id="studentTable">

                            <!-- Dynamic rows will be added here -->

                        </tbody>

                    </table>

                </section>

            </section>

        </section>

    </main>

* Registration Form: Allows users to input student details (name, ID, email, contact).
* Enrolled Students List: Displays the list of enrolled students in a table format.(it is hidden)

Script:

<script src="script.js"></script>

* To link JavaScript file to the HTML file

CSS CODE:

* I wanted my web page to be in dark colors so I used mostly the shades of black . it gives a gazy look to the web page.
* I used poppins as the fonts which attracted me a lot.
* Header : Title was at center.
* Navbar:justify content :space evenly
* For inputs buttons I used dark grey colors.
* Dynamic scroll bars are also added.
* I also added hovering properties to the nav elements and button properties.
* For enrolled students the edit and delete buttons are opposite in colors which gives pleasant look.
* I also added media queries for ipad and normal phones.

JavaScript Code:

Selectors:

document.addEventListener("DOMContentLoaded", () => {

    const nameInput = document.querySelector("#name");

    const idInput = document.querySelector("#id");

    const emailInput = document.querySelector("#email");

    const contactInput = document.querySelector("#contact");

    const addBtn = document.querySelector("#addBtn");

    const studentTable = document.querySelector("#studentTable");

    const formSection = document.querySelector("#formSection");

    const listSection = document.querySelector("#listSection");

    const formLink = document.querySelector("#formLink");

    const listLink = document.querySelector("#listLink");

* Selects various HTML elements by their IDs or class names for later use in the script:
  + nameInput, idInput, emailInput, contactInput: Input fields for student details.
  + addBtn: Button to add a new student.
  + studentTable: The table body where student data will be displayed.
  + formSection, listSection: Sections for the form and student list, respectively.
  + formLink, listLink: Navigation links to switch views between the form and list sections.

 let students = JSON.parse(localStorage.getItem("students")) || [];

* Retrieves student data from localStorage if it exists.
* Parses the JSON string into a JavaScript array. If no data exists, initializes an empty array.

Collecting and validation Of inputs:

 const validateInputs = () => {

        const name = nameInput.value.trim();

        const id = idInput.value.trim();

        const email = emailInput.value.trim();

        const contact = contactInput.value.trim();

        const nameValid = /^[A-Za-z\s]+$/.test(name);

        const idValid = /^[0-9]+$/.test(id);

        const emailValid = /^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(email);

        const contactValid = /^[0-9]+$/.test(contact);

        return {

            nameValid,

            idValid,

            emailValid,

            contactValid,

            name,

            id,

            email,

            contact

        };

    };

* .value: Fetches the current value of the input field.
* .trim(): Removes any whitespace at the beginning and end of the string.
* I used regular expressions (regex) for validation and .test() returns true if it matches the pattern or return false otherwise.

Adding new student and checking validation:

addBtn.addEventListener("click", () => {

        const { nameValid, idValid, emailValid, contactValid, name, id, email, contact } = validateInputs();

        // Check if all fields are filled and valid

        if (!name || !id || !email || !contact) {

            alert("Please fill in all fields.");

            return;

        }

        // Validate inputs

        if (!nameValid) {

            alert("Student name must only contain letters and spaces.");

            return;

        }

        if (!idValid) {

            alert("Student ID must be a number.");

            return;

        }

        if (!emailValid) {

            alert("Please enter a valid email address.");

            return;

        }

        if (!contactValid) {

            alert("Contact number must be a number.");

            return;

        }

        // Add student if all validations pass

        students.push({ name, id, email, contact });

        localStorage.setItem("students", JSON.stringify(students));

        alert("Student added successfully!");

        nameInput.value = "";

        idInput.value = "";

        emailInput.value = "";

        contactInput.value = "";

        renderTable();

    });

* Adds an event listener to the "Add" button that runs whenever it is clicked.
* Validates inputs and displays error messages for invalid fields.
* Prevents further actions if any field is invalid.
* Adds the validated student details to the students array.
* Saves the updated array to localStorage.
* Clears the form fields for new input.
* Calls renderTable to update the displayed table.

Rendering the table:

   const renderTable = () => {

        studentTable.innerHTML = "";

        students.forEach((student, index) => {

            const row = document.createElement("tr");

            row.innerHTML = `

                <td>${student.name}</td>

                <td>${student.id}</td>

                <td>${student.email}</td>

                <td>${student.contact}</td>

                <td>

                    <button class="edit-btn" onclick="editStudent(${index})">Edit</button>

                    <button class="delete-btn" onclick="deleteStudent(${index})">Delete</button>

                </td>

            `;

            studentTable.appendChild(row);

        });

    };

* Renders the table for new data.
* The inner Html is used for dynamic entry of students details.
* Two buttons also created foe edit and delete purpose. And then append happens.

Edit Student:

 window.editStudent = (index) => {

        const student = students[index];

        const newName = prompt("Edit Name:", student.name);

        const newId = prompt("Edit Student ID:", student.id);

        const newEmail = prompt("Edit Email:", student.email);

        const newContact = prompt("Edit Contact Number:", student.contact);

        if (newName && newId && newEmail && newContact) {

            students[index] = { name: newName, id: newId, email: newEmail, contact: newContact };

            localStorage.setItem("students", JSON.stringify(students));

            renderTable();

        }

    };

* Prompt method is used for the edit purpose.
* Updates the student data and saves it to localStorage. Rerenders the table with updated data.

Delete student:

window.deleteStudent = (index) => {

        if (confirm("Are you sure you want to delete this student?")) {

            students.splice(index, 1);

            localStorage.setItem("students", JSON.stringify(students));

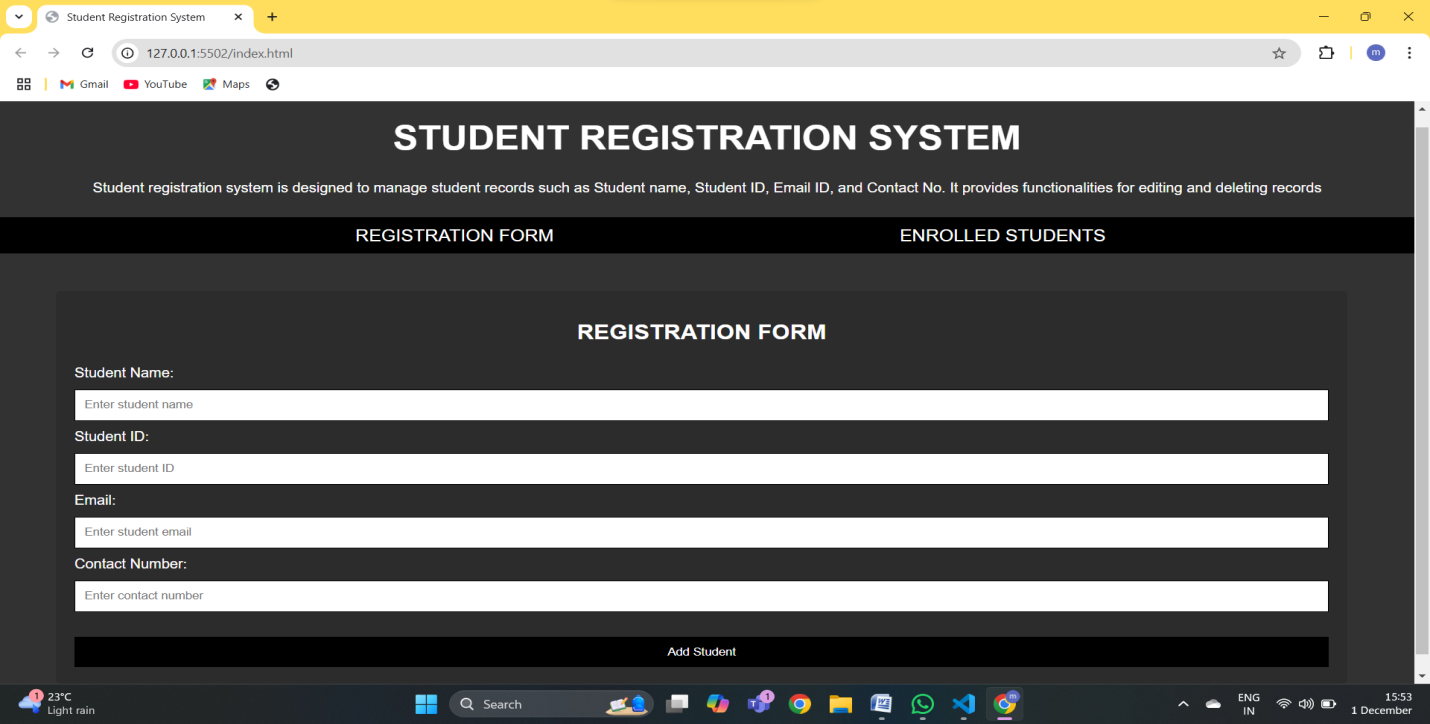
            renderTable();

        }

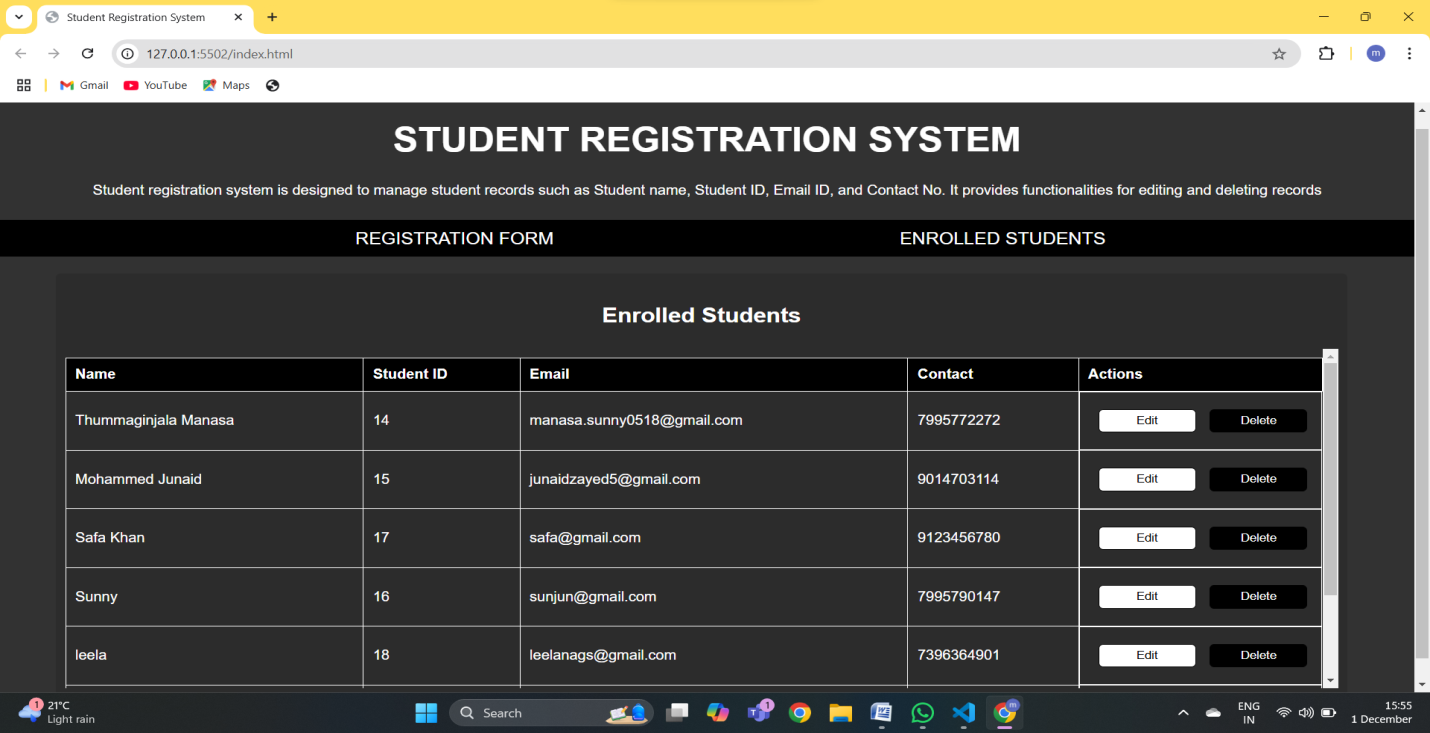
    };

* Confirms if the user wants to delete a student.
* Removes the student from the students array and updates localStorage.
* Calls renderTable to reflect the change.

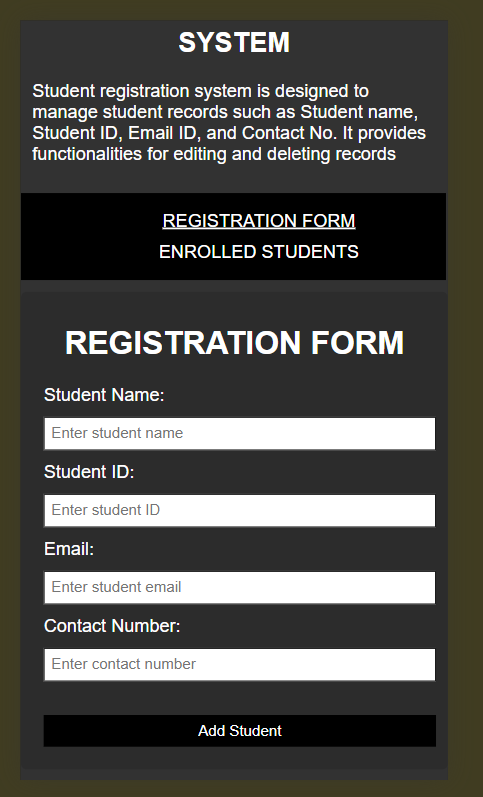
Output :



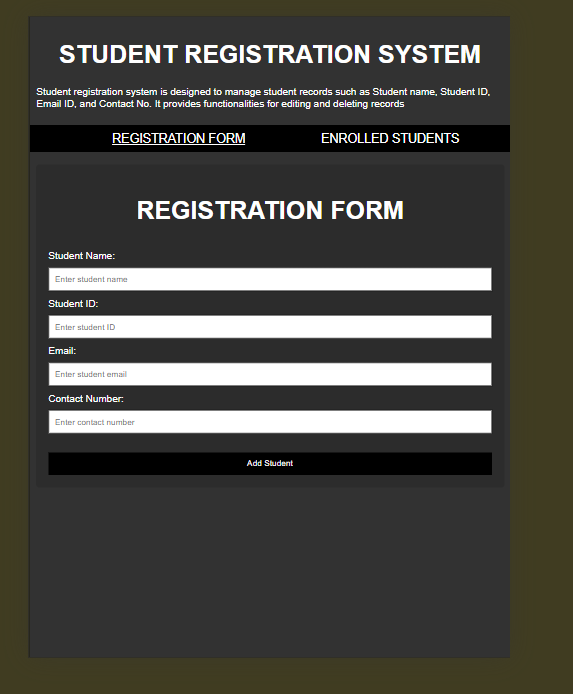
HOME PAGE



ENROLLED STUDENTS PAGE



MOBILE VIEW



IPAD VIEW

GIT HUB LINK:  
<https://github.com/MohammedJunaidKhan18/Student-Registration-System>