Student Registration System

1.0 Overview

The Student Registration System is a web-based application that facilitates the management of student information. Users can register new students, view existing records, edit details, and delete entries as needed. The system employs local storage for data persistence, ensuring that registered student data is retained across sessions. It features a responsive design to accommodate various device screen sizes.

1.1 Features

Student Registration: Users can input student details through a user-friendly form.

View Registered Students: Displays all registered students in a dynamic table format.

Edit and Delete Records: Allows users to modify or remove student records directly from the table.

Data Persistence: Utilizes local storage to store and retrieve student data.

Responsive Design: Adjusts layout and functionality for optimal viewing on different devices.

1.2 File Structure

index.html: Defines the structure and content of the web page.

styles.css: Contains styles for layout, colours, and responsiveness.

script.js: Handles form submissions, manages student records, and interacts with local storage for data persistence.

1.3 Example Code Snippets

index.html

The HTML structure defines a student registration form and a table for displaying registered students. It includes sections for input fields and dynamically populated table rows for each student record.

```
<!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="./styles.css">
</head>
<body>
    <!-- Header Section -->
    <header>
        <h1>Student Registration System</h1>
        <p>The Student Registration System allows users to
register, view, edit, and delete student details.</p>
```

```html
    </header>

    <!-- Main Section for Student Registration Form -->
    <section>
        <!-- Student Registration Form -->
        <form id="studentRegistrationForm">
            <!-- Input fields for student details -->
            <label for="studentName">Student Name</label>
            <input type="text" id="studentName"
name="studentName" placeholder="Enter Student Name"
required>

            <label for="studentID">Student ID:</label>
            <input type="number" id="studentID"
name="studentID" placeholder="Enter Student ID" required>

            <label for="class">Class:</label>
            <input type="text" id="class" name="class"
placeholder="Enter Student's Class" required>

            <label for="emailID">Email ID:</label>
            <input type="email" id="emailID" name="emailID"
placeholder="Enter Email ID" required>

            <label for="contactNo">Contact No:</label>
            <input type="number" id="contactNo"
name="contactNo" placeholder="Enter Contact No" required>

            <label for="address">Address:</label>
            <textarea id="address" name="address"
placeholder="Enter Address" required></textarea>

            <!-- Submit button -->
            <button type="submit">Register</button>
        </form>
    </section>

    <!-- Section for Displaying Registered Students -->
    <section class="table-wrapper">
        <!-- Heading for Registered Students -->
        <h2>Registered Students</h2>

        <!-- Table to display student records -->
        <table id="studentTable">
            <thead>
                <!-- Table header row -->
                <tr>
                    <th>Student Name</th>
                    <th>Student ID</th>
                    <th>Class</th>
                    <th>Email ID</th>
                    <th>Contact No</th>
                    <th>Address</th>
                    <th>Edit/Delete</th>
                </tr>
            </thead>
            <tbody>
                <!-- Records will be added dynamically using
JavaScript -->
            </tbody>
        </table>
    </section>

    <!-- JavaScript file for functionality -->
    <script src="./script.js"></script>
</body>
</html>
```

styles.css

CSS styles ensure a visually appealing and responsive layout. It provides styling for the header, form inputs, buttons, tables, and media queries for different screen sizes. The design emphasizes a dark theme with contrasting colours for readability and user interaction.

```css
/* Resetting default margin, padding, and box-sizing */
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

/* Body styles */
body {
    font-family: Arial, sans-serif;
    background: linear-gradient(to right, #2c3e50, #4ca1af);
/* Dark gradient background */
    color: white; /* Text color */
}

/* Header styles */
header {
    text-align: center;
    padding: 20px;
    background-color: #34495e; /* Dark blue-gray background
*/
    color: white; /* Text color */
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1); /* Box shadow
for header */
}

header h1 {
    margin: 0;
    font-size: 2.5em; /* Header title font size */
}

header p {
    margin: 10px 0 0; /* Top margin for paragraph */
    font-size: 1.2em; /* Paragraph font size */
```

```css
}

/* Section styles */
section {
    margin: 30px 20px 30px 20px; /* Margin around section */
    padding: 20px; /* Padding inside section */
    background-color: #2c3e50; /* Dark blue-gray background
*/
    border-radius: 8px; /* Rounded corners */
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1); /* Box shadow
for section */
    color: white; /* Text color */
}

/* Form styles */
form {
    display: grid; /* Grid layout for form */
    grid-template-columns: 1fr 2fr; /* Two columns */
    gap: 10px; /* Gap between grid items */
}

form label {
    align-self: center; /* Center-align labels vertically */
    color: #ecf0f1; /* Light gray text color */
}

form input, textarea {
    padding: 10px; /* Padding for inputs and textarea */
    font-size: 16px; /* Font size */
    border: 1px solid #34495e; /* Dark blue-gray border */
    border-radius: 4px; /* Rounded corners */
    background-color: #3b4b58; /* Slightly lighter dark gray
background */
    color: white; /* Text color */
}

textarea::placeholder,
form input::placeholder {
    color: #bdc3c7; /* Light gray placeholder text color */
}

form button {
    padding: 10px; /* Padding for button */
    font-size: 16px; /* Font size */
    color: white; /* Text color */
    background-color: #1abc9c; /* Green background */
    border: none; /* No border */
    border-radius: 4px; /* Rounded corners */
    cursor: pointer; /* Pointer cursor */
    transition: background-color 0.3s; /* Smooth background
color transition */
    grid-column: 1 / span 2; /* Button spans two columns */
}

form button:hover {
    background-color: #16a085; /* Darker green background on
hover */
}

/* Table wrapper styles */
.table-wrapper {
    overflow-y: auto; /* Vertical scrollbar when content
overflows */
    margin-top: 20px; /* Top margin */
    border-radius: 8px; /* Rounded corners */
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1); /* Box shadow
*/
}

/* Table styles */
table {
    width: 100%; /* Full width */
    border-collapse: collapse; /* Collapse border */
    background-color: #2c3e50; /* Dark blue-gray background
*/
}
```

```css
table,
th,
td {
    border: 1px solid #ccc; /* Light gray border */
}

th,
td {
    padding: 10px; /* Padding for cells */
    text-align: left; /* Left-align text */
}

th {
    background-color: #34495e; /* Dark blue-gray background
for header */
    color: white; /* Text color */
}

/* Heading styles */
h2 {
    text-align: center; /* Center-align heading */
}

/* Button styles */
button {
    cursor: pointer; /* Pointer cursor */
    border: none; /* No border */
    padding: 5px 10px; /* Padding for button */
    border-radius: 4px; /* Rounded corners */
    transition: background-color 0.3s, box-shadow 0.3s; /*
Smooth transitions */
}

button.edit {
    background-color: #3498db; /* Blue background for edit
button */
    color: white; /* Text color */
```

```css
}

button.edit:hover {
    background-color: #2980b9; /* Darker blue background on hover */
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.2); /* Light box shadow on hover */
}

button.delete {
    background-color: #e74c3c; /* Red background for delete button */
    color: white; /* Text color */
}

button.delete:hover {
    background-color: #c0392b; /* Darker red background on hover */
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.2); /* Light box shadow on hover */
}

/* Media query for smaller screens */
@media (max-width: 600px) {
    form {
        grid-template-columns: 1fr; /* Single column layout */
    }

    form label,
    form input,
    form textarea {
        width: 100%; /* Full width for labels, inputs, and textarea */
    }

    form button {
        width: 100%; /* Full width for button */
        margin-top: 10px; /* Top margin */
    }

    .table-wrapper {
        overflow-x: auto; /* Horizontal scrollbar when
content overflows */
    }

    th,
    td {
        white-space: nowrap; /* Prevent text wrapping */
        padding: 8px 16px; /* Padding for cells */
    }
}
```

script.js

JavaScript handles form submissions, manages student records dynamically, and interacts with local storage for data persistence. Functions include adding, editing, and deleting student records, ensuring data integrity and responsiveness based on the number of records displayed.

```
// Retrieve form and table elements from the DOM
const form =
document.getElementById('studentRegistrationForm');
const studentTable =
document.getElementById('studentTable').getElementsByTagName
('tbody')[0];

// Event listener for form submission
form.addEventListener('submit', handleFormSubmit);

// Load existing student records from local storage on page
load
```

```
loadFromLocalStorage();

// Function to handle form submission
function handleFormSubmit(event) {
    event.preventDefault(); // Prevent default form
submission behavior

    // Retrieve form input values
    const studentName =
document.getElementById('studentName').value.trim();
    const studentID =
document.getElementById('studentID').value.trim();
    const className =
document.getElementById('class').value.trim();
    const emailID =
document.getElementById('emailID').value.trim();
    const contactNo =
document.getElementById('contactNo').value.trim();
    const address =
document.getElementById('address').value.trim();

    // Validate input fields
    if (!studentName || !studentID || !className || !emailID
|| !contactNo || !address) {
        alert('Please fill in all fields'); // Alert user if
any field is empty
        return;
    }

    // Add student record to the table
    addStudentRecord({ studentName, studentID, className,
emailID, contactNo, address });

    // Reset form inputs after submission
    form.reset();
}

// Function to add a new student record to the table
```

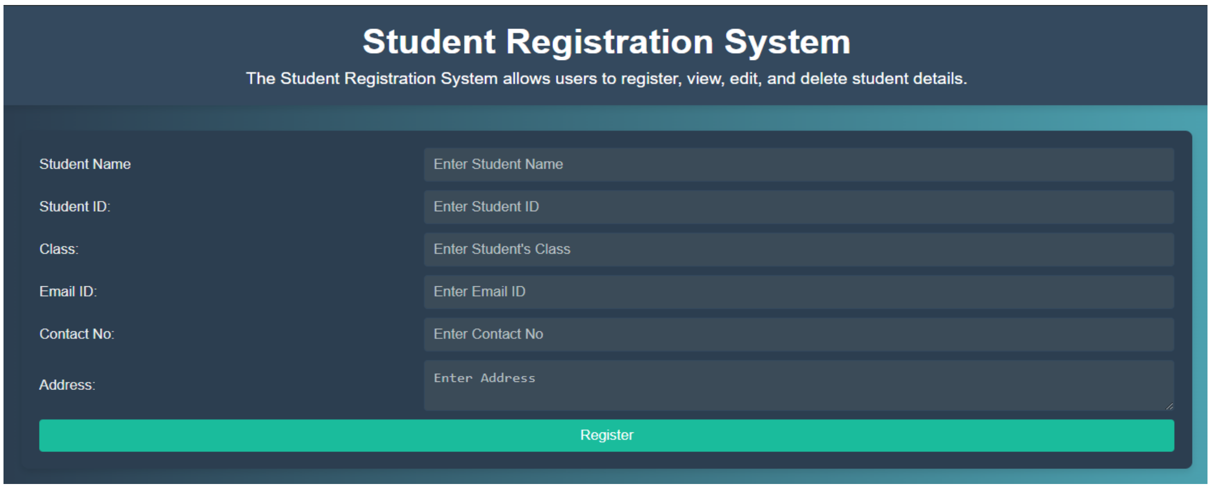
```
function addStudentRecord({ studentName, studentID, className, emailID, contactNo, address }) {
    const row = studentTable.insertRow(); // Insert new row in the table body
    row.dataset.id = studentID; // Set data attribute with student ID

    // Populate row cells with student data
    row.innerHTML = `
        <td>${studentName}</td>
        <td>${studentID}</td>
        <td>${className}</td>
        <td>${emailID}</td>
        <td>${contactNo}</td>
        <td>${address}</td>
        <td>
            <button class="edit">Edit</button>
            <button class="delete">Delete</button>
        </td>
    `;

    // Add event listeners to edit and delete buttons in the new row
    const editButton = row.querySelector('.edit');
    const deleteButton = row.querySelector('.delete');

    editButton.addEventListener('click', () => editStudentRecord(row));
    deleteButton.addEventListener('click', () => deleteStudentRecord(row));

    saveToLocalStorage(); // Save updated student records to local storage
    adjustTableWrapperHeight(); // Adjust table wrapper height based on row count
}
```

```
// Function to populate form fields for editing a student
record
function editStudentRecord(row) {
    // Populate form fields with data from the selected row
for editing
    document.getElementById('studentName').value =
row.cells[0].innerText;
    document.getElementById('studentID').value =
row.cells[1].innerText;
    document.getElementById('class').value =
row.cells[2].innerText;
    document.getElementById('emailID').value =
row.cells[3].innerText;
    document.getElementById('contactNo').value =
row.cells[4].innerText;
    document.getElementById('address').value =
row.cells[5].innerText;

    deleteStudentRecord(row); // Delete the selected row
after editing
}

// Function to delete a student record from the table
function deleteStudentRecord(row) {
    row.remove(); // Remove the selected row from the table
    saveToLocalStorage(); // Save updated student records to
local storage
    adjustTableWrapperHeight(); // Adjust table wrapper
height based on row count
}

// Function to save student records to local storage
function saveToLocalStorage() {
    const students = []; // Initialize array to store
student records

    // Loop through table rows and populate students array
with student objects
```

```
    for (let i = 0; i < studentTable.rows.length; i++) {
        const row = studentTable.rows[i];
        const student = {
            studentName: row.cells[0].innerText,
            studentID: row.cells[1].innerText,
            className: row.cells[2].innerText,
            emailID: row.cells[3].innerText,
            contactNo: row.cells[4].innerText,
            address: row.cells[5].innerText,
        };
        students.push(student); // Push each student object
to the array
    }

    // Store students array in local storage as a JSON
string
    localStorage.setItem('students',
JSON.stringify(students));
}

// Function to load existing student records from local
storage
function loadFromLocalStorage() {
    const students =
JSON.parse(localStorage.getItem('students')) || []; //
Retrieve students array from local storage or initialize
empty array
    students.forEach(addStudentRecord); // Add each student
record to the table on page load
}

// Function to adjust table wrapper height based on the
number of rows
function adjustTableWrapperHeight() {
    const tableWrapper = document.querySelector('.table-
wrapper'); // Select table wrapper element
    console.log(tableWrapper.clientHeight); // Log current
height (for debugging)
```

```
    // Adjust table wrapper height and overflow based on the number of rows
    if (studentTable.rows.length > 5) {
        tableWrapper.style.maxHeight = '70vh'; // Set max height with vertical scrollbar
        tableWrapper.style.overflowY = 'scroll'; // Enable vertical scrolling
    } else {
        tableWrapper.style.maxHeight = 'auto'; // Auto height without scrollbar
        tableWrapper.style.overflowY = 'auto'; // Disable scrolling
    }
}
```

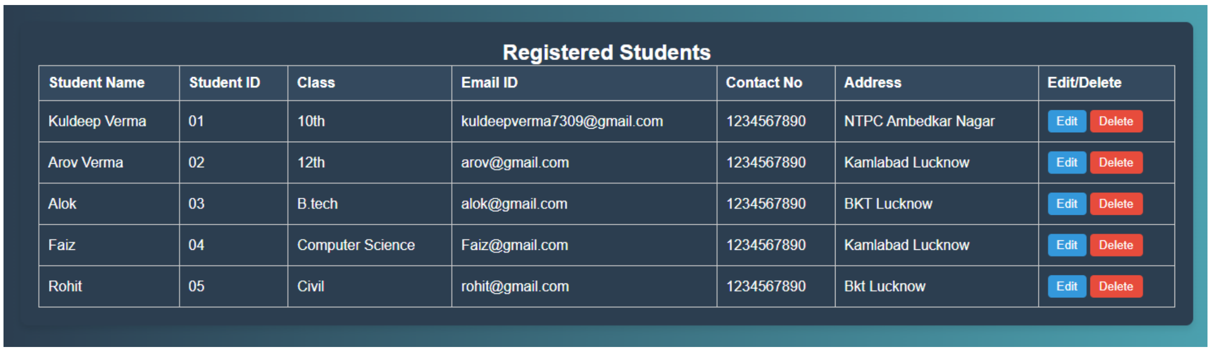
1.4 Output



Student Registration System

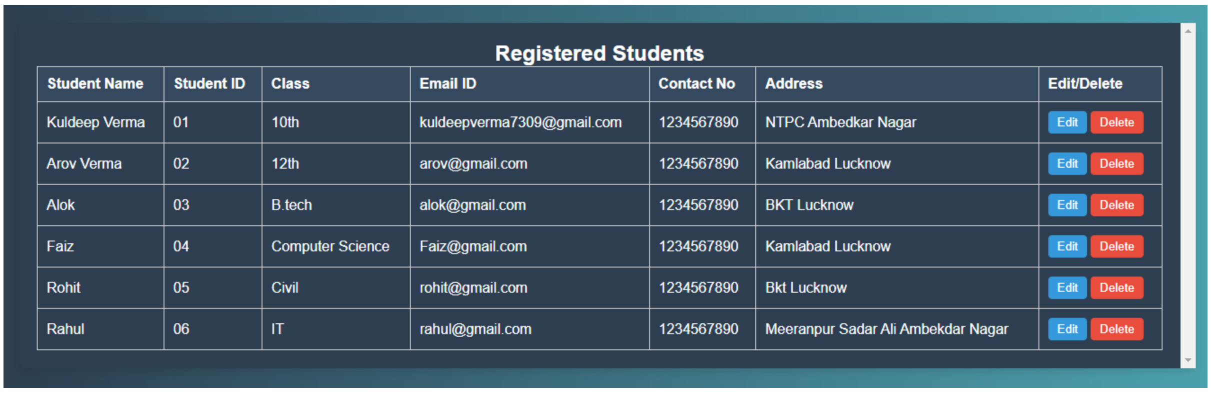
The Student Registration System allows users to register, view, edit, and delete student details.

Student Name	Enter Student Name
Student ID:	Enter Student ID
Class:	Enter Student's Class
Email ID:	Enter Email ID
Contact No:	Enter Contact No
Address:	Enter Address



Registered Students

Student Name	Student ID	Class	Email ID	Contact No	Address	Edit/Delete
Kuldeep Verma	01	10th	kuldeepverma7309@gmail.com	1234567890	NTPC Ambedkar Nagar	Edit Delete
Arov Verma	02	12th	arov@gmail.com	1234567890	Kamlabad Lucknow	Edit Delete
Alok	03	B.tech	alok@gmail.com	1234567890	BKT Lucknow	Edit Delete
Faiz	04	Computer Science	Faiz@gmail.com	1234567890	Kamlabad Lucknow	Edit Delete
Rohit	05	Civil	rohit@gmail.com	1234567890	Bkt Lucknow	Edit Delete



Registered Students

Student Name	Student ID	Class	Email ID	Contact No	Address	Edit/Delete
Kuldeep Verma	01	10th	kuldeepverma7309@gmail.com	1234567890	NTPC Ambedkar Nagar	Edit Delete
Arov Verma	02	12th	arov@gmail.com	1234567890	Kamlabad Lucknow	Edit Delete
Alok	03	B.tech	alok@gmail.com	1234567890	BKT Lucknow	Edit Delete
Faiz	04	Computer Science	Faiz@gmail.com	1234567890	Kamlabad Lucknow	Edit Delete
Rohit	05	Civil	rohit@gmail.com	1234567890	Bkt Lucknow	Edit Delete
Rahul	06	IT	rahul@gmail.com	1234567890	Meeranpur Sadar Ali Ambekdar Nagar	Edit Delete

1.5 GitHub Link

https://github.com/kuldeepverma7309/Student-Registration-System