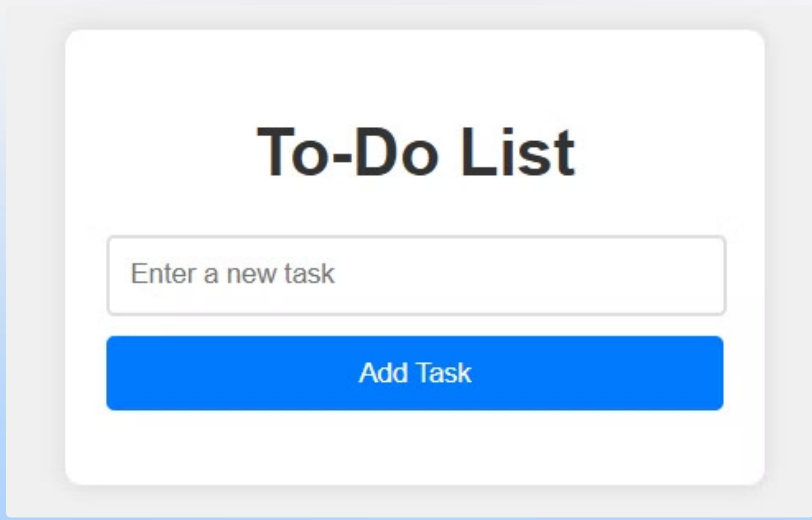
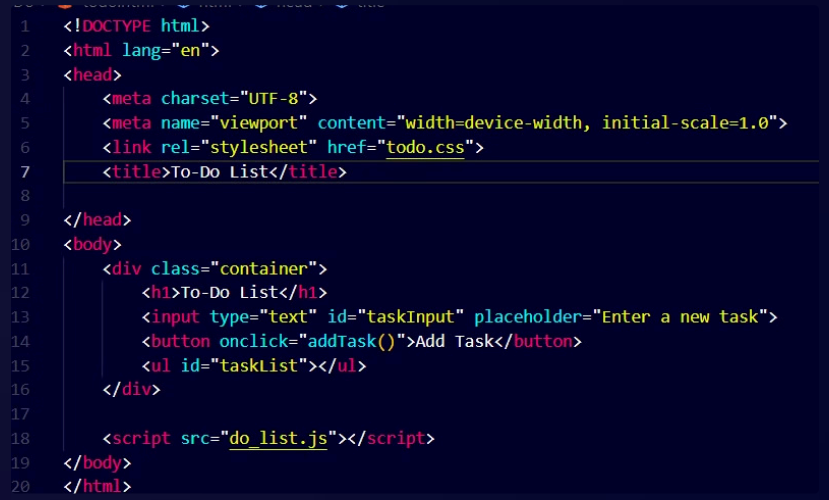
Introduction:

- Name: SHIVAM GUPTA
- University Roll No: 2300290120232
- Branch: Computer Science
- Year: 2
- Section: D

To-Do List Application

This application is a simple and effective tool for managing tasks. Its user-friendly interface and essential features make it a practical solution for individuals seeking a straightforward method to organize their daily activities.





```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="todo.css">
    <title>To-Do List</title>

</head>
<body>
    <div class="container">
        <h1>To-Do List</h1>
        <input type="text" id="taskInput" placeholder="Enter a new task">
        <button onclick="addTask()">Add Task</button>
        <ul id="taskList"></ul>
    </div>

    <script src="do_list.js"></script>
</body>
</html>
```

Introduction to the Project

Goal

The primary goal of this project was to develop a to-do list application that is user-friendly and efficient, helping users stay organized and productive.

Target Audience

The application targets individuals who need a simple yet effective tool to manage their daily tasks, from students and professionals to busy individuals.

Technology Stack

The application leverages the core web technologies: HTML for structure, CSS for styling, and JavaScript for functionality and dynamic behavior.

Key Features

1 Add Tasks

Users can easily add new tasks to their list by entering a description and pressing the "Add Task" button.

2 Remove Tasks

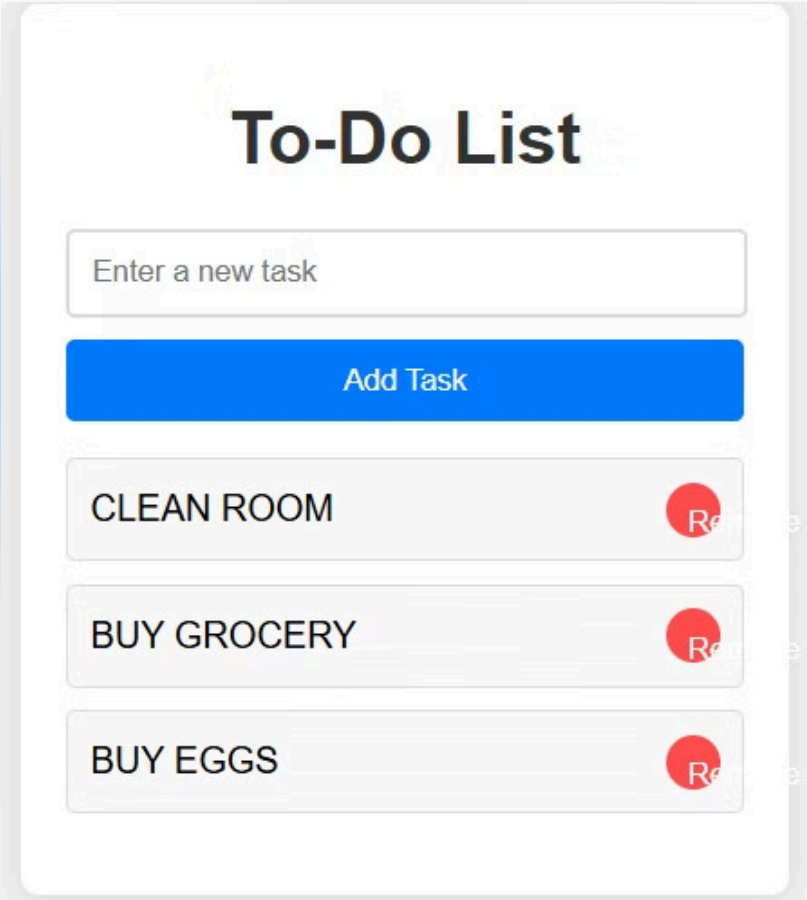
Completed tasks can be removed from the list by clicking a designated "Remove Task" button, effectively decluttering the list and promoting focus.

3 Responsive Design

The application is designed to adapt seamlessly to different screen sizes, ensuring a comfortable and efficient experience on both desktop and mobile devices.

4 Persistent State

Using JavaScript's localStorage API, the application retains user data even after browser sessions are closed, ensuring that task lists persist and are accessible for future use.



Add Tasks

1

Input Field

Users can enter a task description in a designated input field, allowing them to define the tasks they need to complete.

2

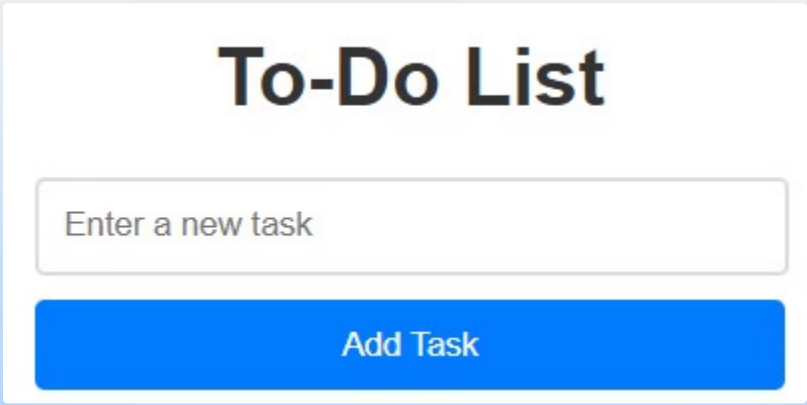
Add Task Button

Clicking the "Add Task" button triggers the addition of the entered task to the list, expanding the user's task inventory.

3

Task List Update

The newly added task appears in the task list, providing a visual confirmation of the task's inclusion and allowing users to track their progress.



Remove Tasks

Task List

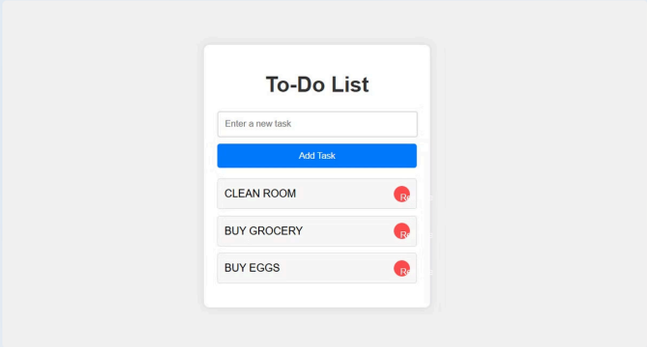
Users can view their tasks, each displayed individually with a corresponding "Remove Task" button.

Remove Task Button

Clicking the "Remove Task" button next to a specific task removes that task from the list, allowing users to manage their tasks effectively.

List Update

After removing a task, the list updates, reflecting the change and presenting an accurate representation of the remaining tasks.



Responsive Design

Device Detection

The application dynamically detects the user's device type, whether it's a desktop or a mobile device.

Optimized Viewing

The application ensures that content remains easily readable and accessible regardless of the device size, ensuring a smooth and consistent experience.

1

2

3

Layout Adjustment

Based on the device type, the application adjusts its layout, resizing elements and adapting the user interface to optimize the user experience.

Persistent State

LocalStorage API	The browser's localStorage API is used to store user data, allowing the application to retain task lists even after the browser is closed.
Data Persistence	Upon reopening the application, the previously saved tasks are automatically retrieved from localStorage and displayed, ensuring data consistency and a seamless user experience.
User Data	The application utilizes localStorage to store and retrieve user-specific information, such as task descriptions and completion status.

```css
body {
    font-family: Arial, sans-serif;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background-color: #f0f0f0;
}
.container {
    background: white;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0,0,0,0.1);
    width: 300px;
}
h1 {
    color: #333;
    text-align: center;
}
input[type="text"] {
    width: calc(100% - 22px);
    padding: 10px;
    border: 2px solid #ddd;
    border-radius: 4px;
    margin-bottom: 10px;
}
button {
    width: 100%;
    padding: 10px;
    background-color: #007bff;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}
```

```
function addTask() {
    const taskInput = document.getElementById('taskInput');
    const task = taskInput.value.trim();
    if (task) {
        tasks.push(task);
        taskInput.value = '';
        renderTasks();
    }
}

function removeTask(index) {
    tasks.splice(index, 1);
    renderTasks();
}

function renderTasks() {
    const taskList = document.getElementById('taskList');
    taskList.innerHTML = '';
    tasks.forEach((task, index) => {
        const li = document.createElement('li');
        li.textContent = task;
        const button = document.createElement('button');
        button.textContent = 'Remove';
        button.onclick = () => removeTask(index);
        li.appendChild(button);
        taskList.appendChild(li);
    });
}
```

Conclusion and Next Steps

Project Completion

The to-do list application has been successfully developed and implemented, providing users with a functional tool for task management.

Future Enhancements

Future enhancements could include the addition of features such as task prioritization, deadlines, and notifications to further enhance the application's functionality.

User Feedback

User feedback will be collected to identify areas for improvement and guide future development efforts, ensuring that the application meets the needs of its users.