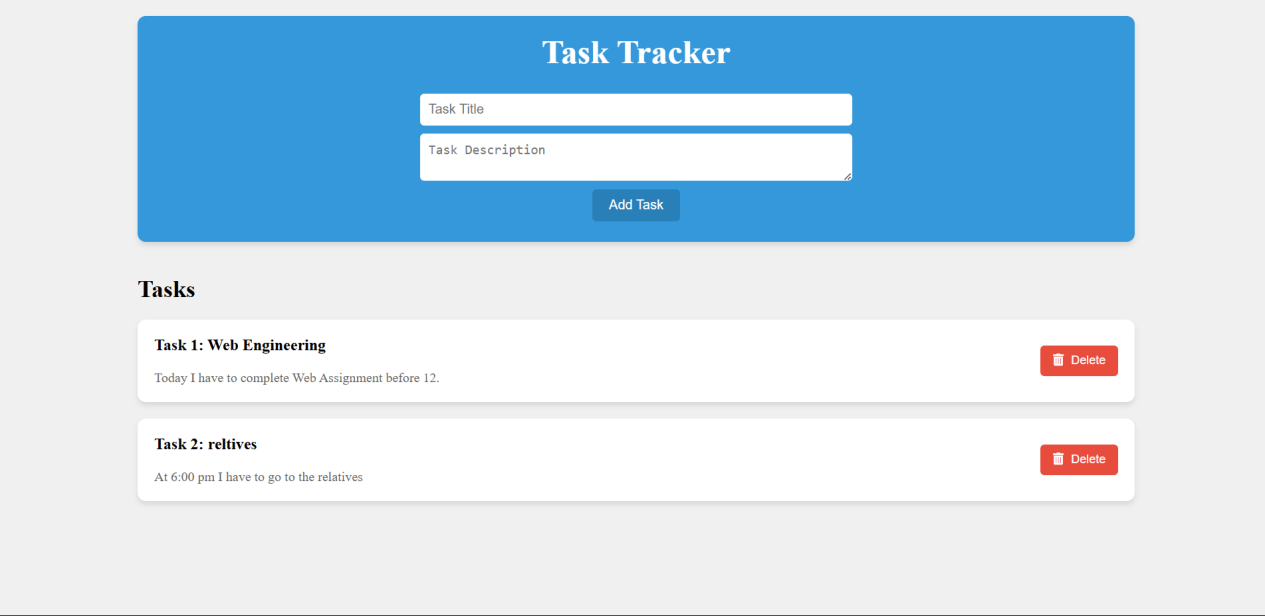
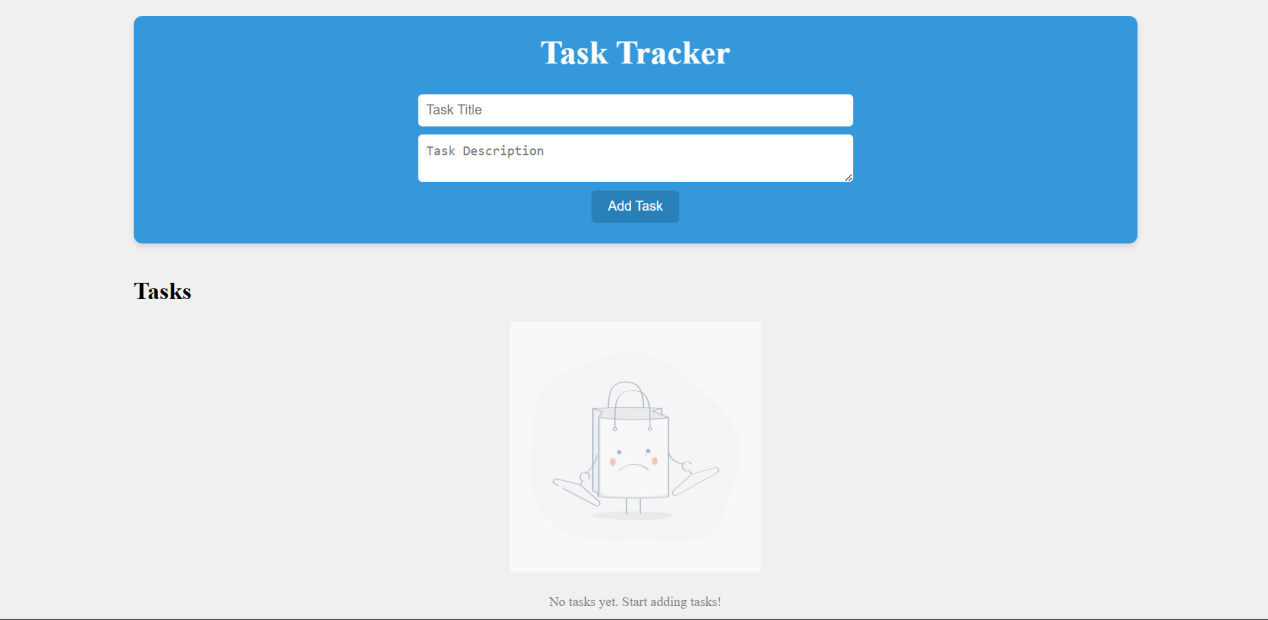
## Todo List Application: Enhanced Functionalities

### Overview

The Task Todo List application has been designed to help users manage their tasks effectively. With several key functionalities, it provides a user-friendly experience while ensuring tasks are organized and easily accessible. Below are the enhanced features and their implementations:



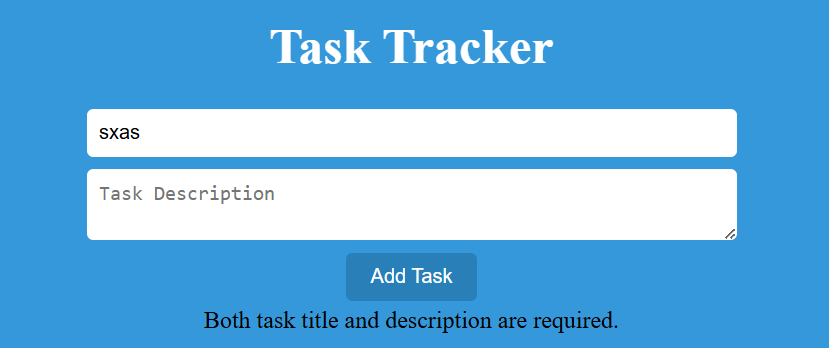
### Key Functionalities

**Empty Text Field Validation**

* 1. **Feature**: If either the task title or description field is empty when the user attempts to add a task, an immediate error message is displayed.
  2. **Implementation**: The addTask() function checks the values of both input fields. If any field is empty, it sets an error message that prompts the user to fill in the required fields.

if (titleValue === '' || descriptionValue === '') { errorMessageContainer.textContent = 'Both

task title and description are required.'; return;}



**Dynamic Button Color Change**

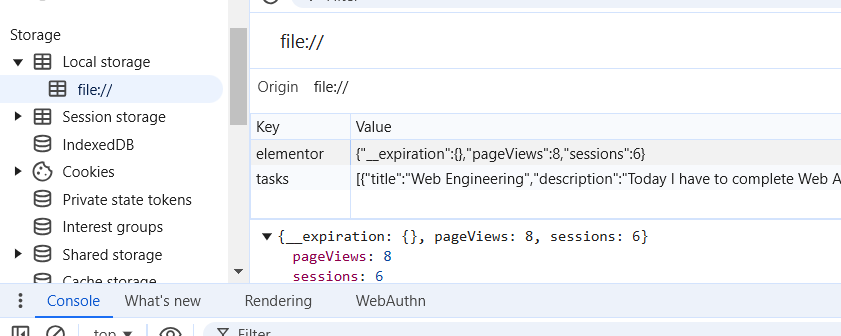
* 1. **Feature**: When both text fields contain valid input, the "Add Task" button changes color to indicate readiness to add a task.
  2. **Implementation**: The updateButtonState() function is called whenever the user types in the input fields, checking if both fields are filled. If they are, an active class is added to the button, changing its color to green.

if (titleValue !== '' && descriptionValue !== '') { addTaskButton.classList.add('active'); // Change button color to green} else { addTaskButton.classList.remove('active'); // Revert button color}

**Storage and Resetting Button Color**

* 1. **Feature**: Upon clicking the "Add Task" button, the task is stored in local storage, and the button color resets to its original state.
  2. **Implementation**: After successfully adding a task, the addTask() function saves the task into the tasks array and updates local storage. It then resets the input fields and reverts the button color to blue.

localStorage.setItem('tasks', JSON.stringify(tasks));taskTitle.value = '';taskDescription.value = '';updateButtonState(); // Reset button color



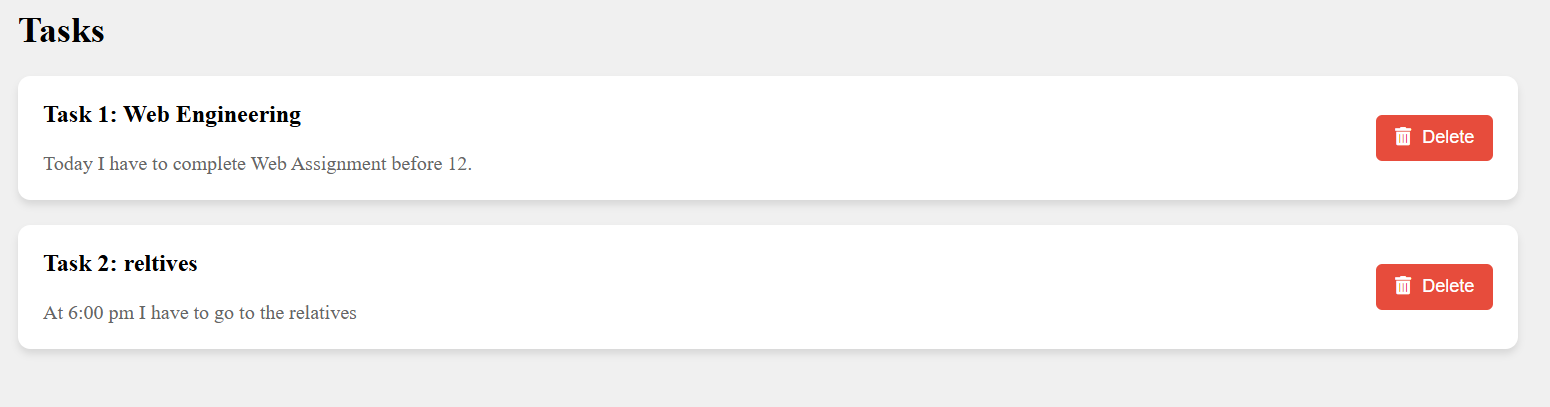
**Dynamic Task Numbering**

* 1. **Feature**: Each task is dynamically numbered (e.g., "Task 1", "Task 2"). When a task is deleted, subsequent tasks automatically update their numbering.
  2. **Implementation**: The displayTasks() function iterates over the tasks array using the forEach() method. The task number is derived from the current index of the task in the array, ensuring that numbering remains consecutive even after deletions.

const taskNumber = index + 1; // Task numbers start from 1h3.textContent = `Task ${taskNumber}: ${task.title}`; // Display task number

### 

After deleting the Task 3 is now Task 2:



### Conclusion

These enhancements create a versatile and user-friendly Task Tracker application that not only validates user input but also provides a visually responsive interface. With dynamic task management and clear feedback mechanisms, users can efficiently track and organize their tasks.