Lab 6: Browser Events

Submission Instruction

Please submit zipped (compressed) file with roll number as filename. The zip should contain 1 directory with three files as part of submission:

[Roll_No].zip

|----Q1

|---- index.html

|---- script.js

|---- style.css

1. This activity is an individual submission, NOT a group activity.
2. Submissions should be done via the course portal only.
3. The deadline mentioned is strictly immutable. No extensions will be given.
4. Any naming conventions mentioned in this activity must be followed strictly or marks may be deducted.
5. Any plagiarized content will fetch zero marks for the current lab and will be followed by strict action. However, discussion of ideas is allowed.

Task: The Interactive Task List

Your goal is to build a clean and intuitive to-do application for managing daily tasks. The application should provide a fluid and responsive experience, allowing users to add, remove, and reorder tasks seamlessly.

Functional Requirements:

Part 1: Task Creation Functionality (5 Marks)

- The application must have a form for user input, allowing them to define a new task.

- Upon submission of this form, the page **must not** reload. This behavior must be intercepted.
- A new task item, containing the user's text and a "Remove" button, must be dynamically created and appended to the task list.
- The text input field must be cleared after the task is successfully added.

Part 2: Task Removal Functionality (5 Marks)

- Every task in the list must have a "Remove" button.
- When a user clicks this button, the corresponding task item must be removed from the DOM.
- This functionality must work for any task, including those that are added after the initial page load.

Part 3: Task Reordering Functionality (Drag and Drop) (10 Marks)

- Users must be able to click and drag any task item to change its position within the list.
- While a task is being dragged, it must have a distinct visual style (e.g., reduced opacity) to indicate it is the active drag element. This style should be removed when the drag operation concludes.
- The task list itself must function as a "drop zone." The browser's default behavior, which typically prevents dropping, must be overridden.
- When a task is dropped in a new position, the DOM must be updated to reflect the new order.

Constraints

The implementation **must** adhere to the following rules:

- **Event Handling:** All event listeners must be attached using the element.addEventListener() method. Inline HTML event attributes (e.g., onclick) are strictly forbidden.
- **Form Interception:** You must use event.preventDefault() within the form's submit event handler to manage the task creation process with JavaScript.

- **Efficient Deletion:** Task removal **must be implemented using the Event Delegation pattern.** A single event listener on the parent <ul> element should be responsible for handling all "Remove" button clicks.
- **Drag and Drop Events:** The reordering functionality must be built using the appropriate sequence of drag and drop events (dragstart, dragend, dragover, drop).
