**Case Study: To-Do List App**

**Problem Statement:**

**Objective**

Build a React-based To-Do List application that allows users to manage their daily tasks efficiently. The application should enable users to add new tasks, mark tasks as complete or incomplete, and delete tasks. This project focuses on practicing state management with arrays, handling multiple event types, and creating a controlled form for user input, suitable for intermediate-level React developers.

**Requirements**

1. **Task Management**:
   * Users should be able to add new tasks to the list by entering text in an input field and submitting it.
   * Each task should have a description (text) and a status (completed or not completed).
   * Users should be able to mark tasks as completed or revert them to incomplete.
   * Users should be able to delete tasks from the list.
2. **State Management**:
   * Use React's useState hook to manage the list of tasks as an array of objects, where each object contains at least the task's text and completion status.
   * Use a separate state to manage the input field for adding new tasks, ensuring it is a controlled component (i.e., its value is controlled by React state).
3. **Event Handling**:
   * Handle form submission (onSubmit) to add new tasks.
   * Handle input changes (onChange) to update the input field's state in real-time.
   * Handle click events (onClick) for marking tasks as complete/incomplete and deleting tasks.
4. **User Interface**:
   * Display a form with a text input and a submit button to add tasks.
   * Display the list of tasks, showing their descriptions and completion status (e.g., strikethrough text for completed tasks).
   * Provide buttons for each task to toggle its completion status and delete it.
   * Ensure the UI updates dynamically as tasks are added, modified, or deleted.
5. **Validation and Usability**:
   * Prevent adding empty tasks (e.g., trim input and check for non-empty strings).
   * Clear the input field after a task is added.
   * Ensure the list renders efficiently, using unique key props for each task.

**Constraints**

* Use functional components and React hooks (useState) for state management.
* Do not mutate state directly; use immutable updates (e.g., spread operator, map, filter).
* Keep the UI simple but functional. Add styles as per your convenience

**Acceptance Criteria**

* Users can add a new task by typing in the input field and clicking a submit button or pressing Enter.
* Added tasks appear in the list with an initial "incomplete" status.
* Clicking a "Complete" button marks a task as completed, visually indicated (e.g., strikethrough text), and changes the button to "Undo" to revert the status.
* Clicking a "Delete" button removes the task from the list.
* Empty inputs are not added to the list, and the input field clears after submission.
* The application updates the UI correctly after each action without manual page refreshes.

**Optional Enhancements (for Further Practice)**

* Add basic styling (e.g., CSS or a UI library like Tailwind) to improve the visual appeal.
* Implement a "Clear All" button to remove all tasks at once.