El proyecto cumple con los requerimientos solicitados y la consigna principal de crear una aplicación web utilizando React para gestionar una lista de tareas:

- ✓ **Componentes Funcionales:**
  - Utilizamos tres componentes funcionales: **TaskItem**, **TaskList**, y **TaskForm**.
- ✓ **Hook useState para el Manejo del Estado:**
  - Se utilizó el hook **useState** en varios lugares para gestionar el estado local de la aplicación. Por ejemplo, en el componente principal (**App.js**), utilizamos **useState** para gestionar el estado de la lista de tareas y en el componente **TaskForm**, utilizamos **useState** para gestionar la entrada del usuario.
- ✓ **Hook useEffect para Efectos Secundarios:**
  - Se utilizó el hook **useEffect** en el componente principal (**App.js**) para simular la carga de tareas desde una fuente externa. (En una aplicación real, podríamos utilizar este hook para cargar datos desde una API, por ejemplo).
- ✓ **Eventos para Interactuar con el Usuario:**
  - Se utilizaron eventos para interactuar con el usuario, como el evento **onClick** en los botones de completar y eliminar tareas, así como el evento **onChange** en el input del formulario para agregar nuevas tareas.
- ✓ **Persistencia de Datos con localStorage:**
  - ✓ Se ha implementado la persistencia de datos utilizando **localStorage** para que las tareas persistan incluso después de recargar la página. Esto garantiza que la información de las tareas se conserve entre sesiones y mejora la experiencia del usuario.

El proyecto cumple con los requisitos establecidos y demuestra el uso de componentes funcionales, los hooks **useState** y **useEffect**, así como eventos para la interacción del usuario. A continuación, explicamos cómo y para qué se utilizaron los tres componentes funcionales en el proyecto:

- ✓ **TaskItem:**
  - Este componente representa individualmente una tarea en la lista.
  - Utiliza el estado local con **useState** para gestionar la apariencia de la tarea, por ejemplo, aplicando un tachado cuando la tarea está completada.
  - Contiene botones para completar y eliminar la tarea, y estos botones están asociados a eventos que modifican el estado de la tarea y/o notifican al componente principal sobre acciones realizadas.
- ✓ **TaskList:**
  - Este componente muestra la lista completa de tareas.
  - Recibe como propiedades la lista de tareas y funciones para gestionar eventos relacionados con las tareas (completar, eliminar, etc.).
  - Mapea la lista de tareas y renderiza un componente **TaskItem** para cada tarea, pasándole los datos de la tarea y las funciones correspondientes.
- ✓ **TaskForm:**
  - Este componente contiene un formulario para agregar nuevas tareas.
  - Utiliza el estado local con **useState** para gestionar la entrada del usuario (el nombre de la nueva tarea).

- Envía la nueva tarea al componente principal mediante una función (**onAddTask**) cuando se envía el formulario.
- El formulario tiene un input para el nombre de la tarea y un botón para agregar la tarea.

En resumen:

- ✓ **TaskItem** se encarga de representar individualmente una tarea y manejar eventos asociados a esa tarea.
- ✓ **TaskList** muestra la lista completa de tareas, utilizando **TaskItem** para cada tarea individual.
- ✓ **TaskForm** maneja la entrada del usuario para agregar nuevas tareas y utiliza el estado local para gestionar esa entrada.

Estos tres componentes trabajan juntos para crear una interfaz completa de gestión de tareas, donde **TaskList** muestra todas las tareas, **TaskItem** representa cada tarea individualmente, y **TaskForm** permite agregar nuevas tareas.