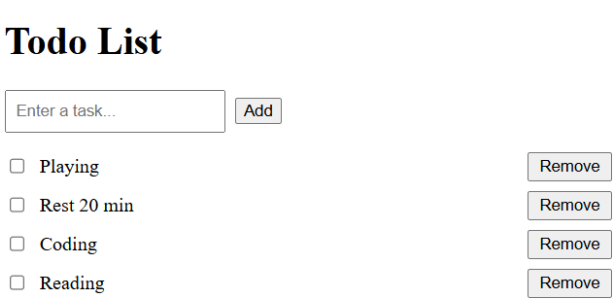
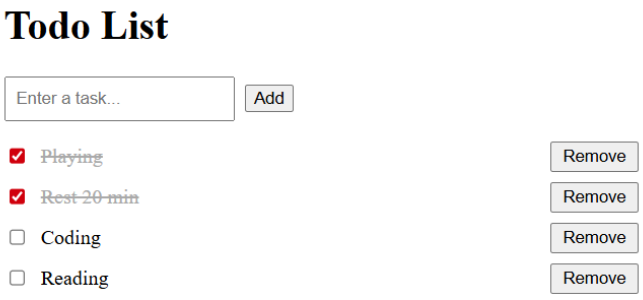
Todo List App with Redux

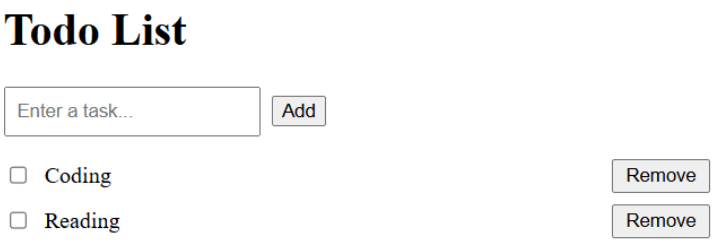
1. **Task Description**

   Develop a React-based todo list application that allows users to add, remove, and mark tasks as complete. The app must use Redux for centralized state management, ensuring all state changes (adding, toggling completion, and removing tasks) are handled through Redux actions and reducers.

2. **Task Output Screenshot**







3. **Widget/Algorithm Used In Task**

   Libraries and Widgets

- React: The core library for building the user interface components.
- Redux: For centralized state management—stores the list of todos, tracks their completion status via actions.
- React-Redux: Connects React components to the Redux store, enabling components to subscribe to state changes and dispatch actions.
- UI Components:
  - Input Field: Collects new todo text from the user.
  - Add Button: Dispatches an action to add the new todo to the Redux store.

    - Todo List: Displays all todos. Each task includes:
        - Checkbox/UI Control: Toggles the completion status (dispatches a Redux action).
        - Remove Button: Dispatches an action to delete the task from the store.
- Redux Toolkit used to simplify reducer and action creation.

Algorithms and State Management Logic

- State Structure: The Redux store holds an array of todo objects. Each todo has at least id, text, and completed properties.
- Filtering out removed todos is handled in the removeTodo reducer.
- Toggling completion is handled in the toggleTodo reducer by inverting the completed property of the matching todo.
- Dynamic UI updates strikethrough for completed tasks are achieved by rendering based on the completed property.
- Unique IDs for each todo are generated using Date.now() for simple uniqueness.