# ***TodoContainer.js***

This is a React component for a todo app that fetches data from an API and displays it on the screen. It has functionality to add, edit, and delete todo items, as well as sort them. It also allows users to click on a todo item to view and edit its description.

The component takes in four props:

- `tableName`: a string that specifies the table to fetch data from
- `sideBar`: a boolean that determines whether or not to display the sidebar
- `searchTerm`: a string that specifies the search term to filter the data
- `setCurrentLink`: a function that sets the current link in the navigation bar

The component renders an `AddTodoForm`, a `SortButton`, and a `TodoList` component. When the user clicks on a todo item in the `TodoList`, a `ItemDescription` component is rendered on the right side of the screen.

The component uses the `useState` and `useEffect` hooks to manage state and perform side effects. It also imports functions from an `API` file to interact with the backend API.

Inside the component, it initializes several state variables using the `useState` hook:

- `todoList`: an array of todo list items.
- `isLoading`: a boolean that indicates whether the data is currently being fetched from the server.
- `direction`: a string that indicates the sorting direction, either "asc" or "desc".
- `itemDescription`: a string that represents the currently selected todo item for displaying its description.
- `showDescription`: a boolean that indicates whether to show the item description panel or not.
- `errorMessage`: a string to hold error messages that may occur during the application's execution.

The component has two `useEffect` hooks:

- The first `useEffect` hook calls the `requestGetTodo` function from `API.js` to fetch the data from the server when the component is mounted. The fetched data is then stored in the `todoList` state variable. If an error occurs, the `errorMessage` state variable is set to an appropriate message.

- The second `useEffect` hook monitors the changes in `todoList` and `isLoading` variables. When `isLoading` becomes false, it stores the `todoList` state variable in the local storage as a JSON string.
-

The component has several functions:

- `handleSort` is a function that sets the `direction` state variable to the opposite direction and then calls the `requestSortedList` function.
- `requestSortedList` function calls the `requestSortData` function from `API.js` to sort the todo list data based on the current `direction`. If the sorting is successful, the `todoList` state variable is set to the new sorted list. If an error occurs, the `errorMessage` state variable is set to an appropriate message.
- `addTodo` is a function that calls the `requestAddTodo` function from `API.js` to add a new todo item to the list. If the addition is successful, the `todoList` state variable is updated with the new list of items. If an error occurs, the `errorMessage` state variable is set to an appropriate message.
- `removeTodo` is a function that calls the `requestDeleteTodo` function from `API.js` to remove a todo item from the list. If the removal is successful, the `todoList` state variable is updated with the new list of items. If an error occurs, the `errorMessage` state variable is set to an appropriate message.
- `editTodo` is a function that calls the `requestEditTodo` function from `API.js` to edit a todo item in the list. If the editing is successful, the `todoList` state variable is updated with the new list of items.
- `editDescription` is a function that calls the `requestEditDescription` function from `API.js` to edit the description of a todo item in the list. If the editing is successful, the `todoList` state variable is updated with the new list of items.
- `handleDescription` is a function that toggles the `showDescription` state variable and sets the `itemDescription` variable to the ID of the selected todo item.