React useReducer Hook Assignment

Todo List Application

Task

- Build a simple todo list application using the **useReducer** hook.
- Implement features to add new todos, mark todos as completed, and delete todos.
- Use context to manage the state of the todo list across components.
- Provide functionality to filter todos based on their completion status (e.g., all, active, completed).
- Design a user-friendly interface with intuitive controls for managing todos.

***NOTE*: Read the task carefully and understand the problem statement and think of what steps you should take to solve this problem on a paper before reading the given steps to solve this problem.**

Steps to solve the problem:

1. Define Initial State:

- Decide on the structure of your todo list state. Each todo item might have properties like **id**, **text**, and **completed**.
- Initialize an empty array to hold the initial state of the todo list.

2. Create Reducer Function:

- Define a reducer function (**todoReducer**) that takes the current state and an action, and returns a new state based on the action type.
- Implement cases for different action types such as adding a todo, toggling the completion status of a todo, and deleting a todo.

3. Implement Component:

- Create a functional component (e.g., **TodoList**) to manage the todo list state using the **useReducer** hook.
- Import **useState** and **useReducer** hooks from React.
- Initialize state variables for the todo list items and the input text for adding new todos using **useState**.
- Use the **useReducer** hook to manage state transitions based on user actions.

4. Define Action Types:

- Define action types that represent different actions that can be performed on the todo list (e.g., **ADD_TODO**, **TOGGLE_TODO**, **DELETE_TODO**).
- Each action type should have a corresponding payload (data) associated with it.

5. Handle User Interactions:

- Implement event handlers to handle user interactions such as adding new todos, toggling the completion status of todos, and deleting todos.
- Dispatch actions to the reducer function based on user interactions.

6. Update State:

- Update the state based on the action type and payload received from the user interactions.
- Modify the state immutably to ensure React detects changes properly.

7. Render Todo List:

- Render the todo list items dynamically based on the current state of the todo list.
- Display checkboxes to toggle completion status and delete buttons to remove todos.
- Apply conditional styling to visually indicate completed todos.

8. Styling and User Interface:

- Style the todo list application to enhance the user experience.
- Use CSS or a UI library like Bootstrap to style components and improve visual appeal.
- Ensure that the user interface is intuitive and easy to navigate.