Day 1 Coding challenge

Problem Statement

Build a Task Management Application: You are required to create a simple task management application that allows users to add, remove, and toggle tasks as completed. The application should utilize React Hooks for state management, Context API for global state, and demonstrate advanced hooks. Additionally, create a custom hook for managing tasks and implement SSR to serve the initial task list.

User Stories

1. **As a user**, I want to be able to view a list of tasks when I first load the application, so that I can see what tasks I have.
2. **As a user**, I want to be able to add a new task to my task list, so that I can keep track of what I need to do.
3. **As a user**, I want to be able to remove a task from the task list, so that I can delete tasks I no longer need.
4. **As a user**, I want to be able to toggle a task as completed, so that I can easily track my progress.
5. **As a user**, I want to see the total number of tasks and the number of completed tasks, so that I can assess my productivity.
6. **As a developer**, I want to implement a custom hook for managing the task state, so that I can reuse this logic across different components.

Assignment Structure

1. Setup (10 Minutes)

- Create a new React application using `create-react-app`.
- Install necessary dependencies such as `axios` for HTTP requests and any other libraries you may need.

2. Build the Components (30 Minutes)

- **TaskProvider Component (Using Context API)**
  - Create a `TaskContext` to provide task data to the application.
  - Implement a `TaskProvider` component that uses `useReducer` to manage state.
- **TaskList Component**
  - Use `useContext` to consume the tasks from `TaskContext`.

- Render a list of tasks with the ability to toggle completion status and remove tasks.

- **AddTask Component**
    - Implement a form that allows users to add new tasks.
    - Use `useState` to manage the form input and call the function provided by `TaskContext` to add tasks.
- **TaskItem Component**
    - Create a reusable component for individual task items that displays the task text and buttons for completing/removing tasks.

3. Create Custom Hook (10 Minutes)

- **useTasks Custom Hook**
    - Create a custom hook named `useTasks` that encapsulates the logic for adding, removing, and toggling tasks.
    - This hook should return the current state and functions for managing tasks.

4. Server-Side Rendering (10 Minutes)

- **Implement SSR with Next.js**
    - Refactor the application to use Next.js, ensuring the initial task list is served from the server.
    - Utilize Next.js's `getServerSideProps` to fetch the initial list of tasks.

Submission Requirements

- Submit the code via a Git repository link.
- Include a README.md file that provides:
    - Instructions on how to run the application locally.
    - An overview of the application architecture.
    - Explanation of how React Hooks and Context API are implemented.

Evaluation Criteria

- **Functionality**: All user stories should be implemented correctly.
- **Code Quality**: Clean, readable code following best practices.
- **Use of Hooks**: Correct and efficient usage of React Hooks and Context API.
- **SSR Implementation**: Successful implementation of server-side rendering with Next.js.
- **Custom Hook**: Effective creation and usage of the custom hook for task management.

Final Note

The coding assignment is designed to test your understanding of advanced React concepts, especially in using Hooks and Context API. Please manage your time effectively to ensure all components are completed within the allotted 60 minutes. Good luck!