Day 3: State Management and Redux, based on One-Way Data Flow, Flux, and Redux.

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 the completion status of tasks. The application should demonstrate the use of **Flux** for state management and **Redux** for optimizing state management.

User Stories

1. **As a user**, I want to add new tasks so that I can keep track of my to-do items.
2. **As a user**, I want to remove tasks from my list so that I can manage my tasks effectively.
3. **As a user**, I want to toggle the completion status of tasks so that I can mark them as done.
4. **As a developer**, I want to understand and implement one-way data flow in my application.
5. **As a developer**, I want to set up and use Redux for managing application state effectively.

Assignment Structure

1. Setup (10 Minutes)

- Create a new React application using `create-react-app`.
- Install necessary dependencies for Redux (e.g., `redux`, `react-redux`).

2. Build the Flux-Based Task Management (20 Minutes)

- **Understanding One-Way Data Flow**
  - Implement a simple Flux architecture to manage the state of the task management system.
  - Create a `Dispatcher`, `Store`, and `Action Creators` to handle adding, removing, and toggling tasks.
- **Build the UI**
  - Create a UI with an input field to add tasks and a list to display them.
  - Implement buttons for removing tasks and toggling their completion status.

3. Integrate Redux for State Management (20 Minutes)

- **Set Up Redux**

    - Create actions and reducers for managing tasks (add, remove, toggle).
    - Set up the Redux store and integrate it into the React application.
- **Refactor the Application**
    - Refactor the existing Flux-based application to use Redux for state management.
    - Ensure the UI components are connected to the Redux store using `connect` or `useSelector` and `useDispatch`.

4. Final Touches (10 Minutes)

- **Optimization and Best Practices**
    - Review and optimize the Redux implementation, ensuring that state updates trigger appropriate re-renders.
    - Discuss scenarios where Redux may be overkill and alternatives like the Context API.

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 architecture, including the roles of Flux and Redux.
    - Screenshots of the application and explanations of key components.

Evaluation Criteria

- **Functionality**: All user stories should be implemented correctly with a seamless user experience.
- **Code Quality**: Clean, organized code adhering to best practices in React and Redux.
- **State Management**: Effective use of Flux and Redux for managing application state.
- **User Interface**: The application should be user-friendly and visually appealing.
- **Understanding**: Clear documentation explaining the implementation details and design decisions.

Final Note

This coding assignment is designed to assess your ability to implement and optimize state management solutions using **Flux** and **Redux** in a React application. Please manage your time effectively to ensure all components are completed within the allotted 60 minutes. Good luck!