Report on React Application Structure and State Management

Name: Dheeraj S Pawar
Email: dheerajsp0700@gmail.com

Folder Structure

The project is organized into several key folders and files, which are structured as follows:

1. **src/:**
    - **App.js:** The main application component that sets up the routing for the application.
    - **index.js:** The entry point of the application where the React app is rendered and the Redux store is provided to the app.
2. **Components/:**
    - **Counter.js:** Contains the Counter component logic and UI.
    - **Home.js:** Contains the Home component that displays user data and counter information.
    - **PrivateRouter.js:** A component that handles protected routes, allowing access only if the user is logged in.
    - **SignIn.js:** A component for the user login form.
    - **SignUp.js:** A component for the user registration form.
    - **UserDataForm.js:** A form component for entering user data, integrated with React Quill for rich text editing.
3. **redux/:**
    - **store.js:** Configures the Redux store and integrates the user reducer.
    - **userSlice.js:** Defines the user slice using Redux Toolkit, including actions and reducers for user data management.

State Management Choices

1) Local State:

- **useState Hook:**
    - **App.js:** Manages the **isLoggedIn** state to track the user's login status. The initial state is retrieved from **localStorage**.

Report on React Application Structure and State Management

Name: Dheeraj S Pawar
Email: dheerajsp0700@gmail.com

- **Counter.js:** Manages the **count** and **bgLevel** states, with initial value of count retrieved from **localStorage**.
- **SignIn.js:** Manages **loggedEmail** and **loggedPassword** states for the login form.
- **SignUp.js:** Manages **userName**, **email**, and **password** states for the registration form.
- **UserDataForm.js:** Manages the **userData** state for capturing user information, and **isDirty** to track unsaved changes.

2) **Global State:**

- **Redux:**
    - **userSlice.js:** Uses Redux Toolkit's **createSlice** to define a slice of the state called **user**. This slice includes:
        - **initialState**: An empty array to represent the initial user state.
        - **reducers**: A **saveUser** reducer to save user data to both Redux state and **localStorage**.
    - **store.js:** Configures the Redux store, combining the **user** reducer.

Routing

- **React Router:**
    - **App.js:** Uses **react-router-dom** to define the application routes. The **PrivateRouter** component ensures that routes like **/home**, **/counter**, and **/form** are accessible only when the user is logged in.
    - **PrivateRouter.js:** Checks the **isLoggedIn** prop and either renders the child components if the user is logged in or redirects to the sign-up page.

Component Descriptions

- **Counter.js:** Implements a counter with increment, decrement, and reset functionalities. The background color dynamically changes based on the counter value using **react-spring**.
- **Home.js:** Displays the current counter value and user data, providing navigation to update user info or reset the counter.

Report on React Application Structure and State Management

Name: Dheeraj S Pawar
Email: dheerajsp0700@gmail.com

- **PrivateRouter.js:** Ensures that certain routes are protected and can only be accessed by authenticated users.
- **SignIn.js:** Handles user login, validating against stored credentials and updating the **isLoggedIn** state upon successful login.
- **SignUp.js:** Handles user registration, saving credentials to **localStorage**.
- **UserDataForm.js:** Allows users to input and save their profile information, with rich text fields for better data entry.