Report on Component Structure and State Management Choices

1. SignIn Component

Structure

This component features a form where users can input their email and password.

It also includes a Google login button.

State Management

The component uses useState to manage the email, password, and user data states.

Upon successful login, user data is stored in localStorage.

2. SignUp Component

Structure

Similar to the SignIn component, this one includes a form for user input (first name, last name, email, and password).

It provides navigation to the Dashboard page.

State Management

The component uses useState to manage form inputs (first name, last name, email, password).

The form data is saved to state upon form submission, and the user is navigated to the dashboard.

3. Home Component

Structure

It holds several Link components that lead to different pages like Counter, User Form, Editor, and Dashboard.

4. CounterComp Component

Structure

The component holds the counter's display and three buttons for controlling the counter: increment, decrement, and reset.

The background color is dynamically updated based on the count value.

The state count manages the counter's value.

State Management

The component uses useState to manage the count.

Actions such as increment, decrement, and reset modify the state accordingly.

5. UserComp Component

Structure

This component is a form that allows users to input their personal data, including name, address, email, and phone number.

It includes a "Save" button, and the form data is saved to localStorage.

State Management

The component uses useState to manage form input values (formData), as well as whether the form has been modified (isFormDirty).

useEffect is used to prevent accidental page reloads when the form is dirty, giving users a warning if they try to navigate away.

6. RichTextComp Component

Structure

This component includes a ReactQuill rich text editor, allowing users to type and format text.

It stores the content of the editor in local state and also saves it to localStorage.

State Management

The component uses useState to manage the editorContent state, which represents the content in the text editor.

7.DashboardComp Component

Structure

Uses react-chartjs-2 and Chart.js for rendering a **line chart**.

Displays **user activity trends**.

Chart updates dynamically when the counter value changes.

State Management

The component uses useState to manage the counter state.

State and Data Persistent

For most components, useState is the ideal choice to manage local state. This ensures that components react to user interaction and are able to update the UI dynamically.

In cases where data needs to be persistent (like user data or editor content), the use of localStorage is a good approach.

Usage of React Router

The usage of Link components and routing logic is consistent across multiple components, providing a seamless navigation experience throughout the application.

Usage of Material UI (@mui/material)

Used in User Data Form to enhance User Interface.

Provides TextField, Button, Typography, and Paper components for a professional UI.

Usage of React Spring (@react-spring/web)

Use in App.jsx file for Smooth transitions for page changes.

Usage of React Chart (chart.js)

Used for **data visualization** in Dashboard file

Helps in representing user statistics, analytics, and trends.

Use **line charts.**