* Advance React- Styling

Styling in React can be advanced and done in several ways to improve component-based architecture and scalability.

1.1 CSS-in-JS

CSS-in-JS refers to the style that is written directly inside JavaScript files. Dynamic styling of components, where the styles are transformed based on the various states or props of the component, can also include themselves as their scopes. Libraries providing CSS-in-JS include Styled Components, Emotion, etc., as they allow the definition of styles written in JavaScript files using template literals or objects.

1.2 CSS Modules

CSS Modules allows the drawback of local scoping of CSS classes, thereby preventing naming clashes. Every component has its own CSS files, and class names will be automatically generated as unique identifiers. This method increases maintainability when developing applications on a larger scale, as global styles will definitely not affect components unintentionally.

1.3 Utility-First CSS: Example: Tailwind

With an add-on like Tailwind CSS attached, manufactures need not write various styles. All manufacturers do is to use utility classes to style specific element. This leaves developers with highly reusable and consistent UI designs since the process is quicker, easier and less custom for developing UI components.

1.4 SASS/SCSS with React

SASS is a CSS preprocessor that makes CSS more powerful and maintainable by introducing variables, nested rules, and mixins. The .scss files created by SASS can be smoothly imported into React and compiled to normal CSS. This makes it better organized to manage styles especially in a larger application environment.

1.5 React Native putting styles

In React Native, this would mean to define and style using JavaScript objects instead of using CSS. These objects will hold properties with values in them.

* React Routing ?

Routing in React gives way for navigation from one page to another page in single-page applications, which is advanced routing with a pitch of having complex URLs handling, lazy loading, and scaling of applications as these grow.

2.1 React Router

React Router is the library that is most popularly used for routing in React applications. This allows the developer to describe URL paths and map them to a specific React component. This makes the user interface dynamic, as it reflects the change in view when the user interacts with it. It supports simple as well as complex URL routes.

2.2 Nested Routes

Nested routes allow you to define among other routes, creating a hierarchically constructed URL tree. This is useful when you want to have a common layout with common elements, such as navigation, and render specific views in these common structures based on routing.

2.3 Dynamic Routing

Dynamic routing is where the route works with dynamic parameters such as user IDs or product IDs. This is to allow the application to load different content on the basis of the passed data through the URL. Like a route of the nature of /user/:userId, it dynamically loads user-specific data from the userId parameter in the URL.

2.4 Lazy Loading of Routes

Lazy loading based on the particular route makes performance better since it uses only the components it needs at the moment. It is implemented through React.lazy() and Suspense; hence components are loaded asynchronously when the user moves to that route related to that component.

2.5 Route Protection

With Route Protection, access to certain routing entities is being denied under authentication or authority level.