Module-11) React -Advance React- Styling , Routing

1. **What is React Router? How does it handle routing in single-page applications?**

ANS: React Router is a library for managing navigation in React application.

- It helps you create and manage routes in single page applications (SPAs).
- In an SPA, there is only page HTML file, and React Router makes it look like you are navigating between different pages.
- Instead of loading new HTML pages from the server, React Router dynamically updates the content on the page to match the URL.

How it Works:

- You define different "routes" in your app (like, /home, /about, /contact).
- Each route is connected to a specific React component, which acts like a "page".
- React Router listens to the browser's URL changes (e.g., clicking links or using the back button) and updates the content accordingly without refreshing the page.
- When you navigate, React Router swaps out components to show the content for the new route, all while staying on the same HTML file.

For example:

- /home might display a home component.
- /about might display an about component.
- When you click a link, React Router updates the page content instantly without reloading the whole page.

It makes your app feel fast and seamless, like a traditional multi-page website but with the speed and efficiency of a single-page application.

2. Explain the difference between BrowserRouter, Route, Link, and Switch components in React Router.

ANS:

1. **BrowserRouter**
   - Acts as the "base" for routing in your app.
   - It wraps your entire app to enable routing functionality.

- It uses the browser's history API to keep track of URLs and display the correct content based on the current route.
- The "manger" of your app's routes.
- Example:

```jsx
<BrowserRouter>
  <App />
</BrowserRouter>
```

2. Route

- Defines which component should be shown for a specific URL path.
- When the URL matches the path you define in a <Route>, it displays the corresponding component.
- A rule that maps a URL path to a component.
- Example:

```jsx
<Route path="/home" element={<Home />} />
```

This means the <Home/> component will show when the URL is /home.

3. Link

- Provides clickable navigation links in your app.
- When clicked, it updates the URL without refreshing the page and tells React Route to show the appropriate component.
- A React-friendly version of an <a> tag for navigation.
- Example:

```jsx
<Link to="/about">Go to About</Link>
```

Clicking this will take you to /about and show the corresponding component.

4. Switch(Routes)

- Ensures that only one route is rendered at a time.
- It goes through your defined routes and renders the first one that matches the current URL.
- A "filter" to prevent multiple routes from rendering at once.

Example:

```jsx
<Switch>
  <Route path="/home" component={Home} />
  <Route path="/about" component={About} />
</Switch>
```

- Only one of these routes will render at a time.

```jsx
<Routes>
  <Route path="/home" element={<Home />} />
  <Route path="/about" element={<About />} />
</Routes>
```

Styling

1. What is styling ?

ANS: Styling in React.js refer to applying visual design to your React components to make them look attractive and functional.

React provides multiple ways to style components, giving developers flexibility based on their needs.

1. Inline styling

You can apply styles directly to elements using the style attribute in JSX.

The styles are written as JavaScript objects.

2. CSS Stylesheets

You can write your styles in a separate ".css" file and import it into your React component.

- Create a CSS file.
- Import the CSS file in your React component.

You can use pre-built CSS frameworks like Bootstrap for styling components quickly.

Tailwind CSS is a utility-first CSS frameworks that can be easily integrated with React.