⚛ React – Advanced Topics: Styling & Routing

🎨 1. React Styling

□ Theory Assignment

Q1: What are the different ways to apply styling in React?
React provides several ways to style components:

1. **Inline CSS:**
   Styles applied directly within an element using the style attribute.
   Example:

2. <h1 style={{ color: "blue", fontSize: "24px" }}>Hello World</h1>

3. **CSS Stylesheets:**
   You can import a regular .css file and use class names.
   Example:

4. import "./App.css";

5. <h1 className="title">Welcome</h1>

6. **CSS Modules:**
   Provide **scoped CSS** that applies only to a specific component, preventing name conflicts.
   Example:

7. import styles from "./App.module.css";

8. <h1 className={styles.header}>React Styling</h1>

9. **Styled Components (CSS-in-JS):**
   Uses libraries like **styled-components** for dynamic and reusable styles.
   Example:

10. import styled from "styled-components";

11. const Button = styled.button`

12. background-color: blue;

13. color: white;

14. padding: 10px;

15. `;

Q2: What are the advantages of using CSS Modules or Styled Components?

- Prevents global CSS conflicts.
- Enables dynamic styling using props.
- Maintains modularity and reusability.
- Improves component-level customization.

Q3: What is conditional styling in React?
Conditional styling changes element styles based on state or props.
Example:

```
const [active, setActive] = useState(false);

<h1 style={{ color: active ? "green" : "red" }}>Status</h1>
```

□ Lab Assignment

Task 1: Inline Styling Example

```
import React from "react";


function InlineStyle() {
  return <h2 style={{ color: "purple", backgroundColor: "lavender" }}>Hello from Inline Style</h2>;
}
export default InlineStyle;
```

Task 2: CSS Stylesheet Example
App.css

```
.title {
  color: blue;
  text-align: center;
```

```
  font-size: 24px;
}
```

App.js

```
import React from "react";
import "./App.css";

function App() {
  return <h1 className="title">React Styling Example</h1>;
}
export default App;
```

Task 3: CSS Module Example

App.module.css

```
.header {
  color: darkgreen;
  font-size: 22px;
}
```

App.js

```
import React from "react";
import styles from "./App.module.css";

function App() {
  return <h2 className={styles.header}>Using CSS Modules in React</h2>;
}
export default App;
```

Task 4: Styled Components Example

npm install styled-components

App.js

```
import React from "react";
import styled from "styled-components";

const Button = styled.button`
  background-color: #007bff;
  color: white;
  padding: 10px;
  border-radius: 6px;
  border: none;
`;

function App() {
  return <Button>Styled Component Button</Button>;
}
export default App;
```

□ 2. React Routing

□ Theory Assignment

Q1: What is React Router?
React Router is a **routing library** for React applications that enables **navigation** between different views or pages without refreshing the browser.
It makes Single Page Applications (SPA) behave like multi-page apps.

Q2: What are the main components of React Router?

1. **BrowserRouter:** Wraps the entire app to enable routing.
2. **Routes:** Container for all routes.
3. **Route:** Defines path-component mapping.
4. **Link / NavLink:** Used to navigate without reloading the page.
5. **useNavigate / useParams:** React hooks for dynamic navigation and parameter access.

Q3: How does React Router work?
React Router listens to the URL changes in the browser and displays the component that matches the route path. It uses the HTML5 **History API** to manage navigation state without full page reloads.

Q4: What are the advantages of React Router?

- Enables seamless navigation in SPAs.
- Provides dynamic and nested routing.
- Reduces load time by avoiding page refreshes.
- Improves user experience.

□ Lab Assignment

Task 1: Basic Routing Example

npm install react-router-dom

App.js

```
import React from "react";

import { BrowserRouter as Router, Routes, Route, Link } from "react-router-dom";


function Home() {

  return <h2>Home Page</h2>;
```

```
}
function About() {
  return <h2>About Page</h2>;
}
function Contact() {
  return <h2>Contact Page</h2>;
}

function App() {
  return (
    <Router>
      <nav>
        <Link to="/">Home</Link> |
        <Link to="/about">About</Link> |
        <Link to="/contact">Contact</Link>
      </nav>

      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/about" element={<About />} />
        <Route path="/contact" element={<Contact />} />
      </Routes>
    </Router>
  );
}
```

```
export default App;
```

Task 2: Nested Routing Example

```
import React from "react";
import { BrowserRouter as Router, Routes, Route, Link, Outlet } from "react-router-dom";

function Dashboard() {
  return (
    <div>
      <h2>Dashboard</h2>
      <Link to="profile">Profile</Link> |
      <Link to="settings">Settings</Link>
      <Outlet />
    </div>
  );
}

function Profile() {
  return <h3>User Profile</h3>;
}
function Settings() {
  return <h3>Settings Page</h3>;
}
```

```
function App() {
  return (
    <Router>
      <Routes>
        <Route path="/" element={<h1>Home</h1>} />
        <Route path="dashboard" element={<Dashboard />}>
          <Route path="profile" element={<Profile />} />
          <Route path="settings" element={<Settings />} />
        </Route>
      </Routes>
    </Router>
  );
}
export default App;
```

Task 3: 404 Not Found Page

```
<Route path="*" element={<h2>404 - Page Not Found</h2>} />
```

Task 4: Dynamic Route Example

```
import { useParams } from "react-router-dom";

function User() {
  const { id } = useParams();
  return <h3>User ID: {id}</h3>;
}
```

<Route path="/user/:id" element={<User />} />
