React Router

React Router is a standard library for routing in React. It enables the navigation among views of various components in a React Application, allows changing the browser URL, and keeps the UI in sync with the URL.

Step 1:

Create a new project

npx create-react-app Demo_route

Step 2:

Installing React Router: React Router can be installed via npm in your React application. Follow the steps given below to install Router in your React application:

Step 2.1:cd into your project directory i.e**Demo_route**.

Step 2.2: To install the React Router use the following command:

npm install – -save react-router-dom

Step 3:

Add the following component in App.js

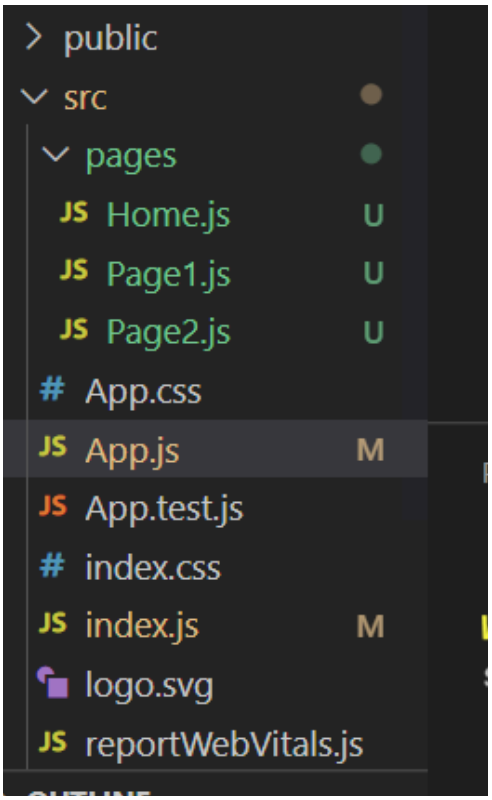
```
import {BrowserRouter, Routes,Route, Link} from 'react-router-dom';
```

Adding React Router Components: The main Components of React Router are:

- **BrowserRouter:**BrowserRouter is a router implementation that uses the HTML5 history API(pushState, replaceState and the popstate event) to keep your UI in sync with the URL. It is the parent component that is used to store all of the other components.
- **Routes:** It's a new component introduced in the v6 and a upgrade of the component. The main advantages of Routes over Switch are:
    - Relative s and s
    - Routes are chosen based on the best match instead of being traversed in order.
- **Route:** Route is the conditionally shown component that renders some UI when its path matches the current URL.
- **Link:** Link component is used to create links to different routes and implement navigation around the application. It works like HTMLanchor tag.

Step 4:

Using React Router: To use React Router, let us first create few components in the react application. In your project directory, create a folder named **pages** inside the src folder and now add 3 files named **Home.js**, **Page1.js** and Page2**.js** to the component folder.



Step 5:

Add the following code in Home.js

```
function Home()
{
return(<h4>Home</h4> );
}
export default Home;
```

Step 6

Add the following in Page1.js

```
function Page1()
{return(<h1>Page 1</h1>);
}
export default Page1;
```

Step 7

Add the following in Page 2.js

```
function Page2()
{
return(<h1>Page 2</h1>);
}
export default Page2;
```

Step 8:

Add the following in App.js

```
import './App.css';
import {BrowserRouter, Routes,Route, Link} from 'react-router-dom';
import Home from './pages/Home';
import Page1 from './pages/Page1';
import Page2 from './pages/Page2';

function App(props) {
return(
<BrowserRouter>
<nav>
<ul>
<li>
<Link to="/">Home</Link>
</li>
<li>
<Link to="/page1">About Us</Link>
</li>
<li>
<Link to="/page2">Contact Us</Link>
</li>
```

```
</ul>
</nav>
<Routes>
<Route   exact path='/' element={< Home />}></Route>
<Route   exact path='/page1' element={< Page1/>}></Route>
<Route   exact path='/page2' element={< Page2 />}></Route>
</Routes>
</BrowserRouter>
 );
}
export default App;
```

Notes:

- **BrowserRouter:** Add BrowserRouter to your app.js file in order to wrap all the other components. BrowserRouter is a parent component and can have only single child.
- **Link:** Let us now create links to our components. Link component uses the **to** prop to describe the location where the links should navigate to.
- **Routes: To render a single component, wrap all the routes inside the Routes Component.**
- **Route:** Route component will now help us to establish the link between component's UI and the URL. To include routes to the application, add the code give below to your app.js.
    - Let us now try to understand the props associated with the Route component
        - exact: It is used to match the exact value with the URL. For Eg., exact path='/about' will only render the component if it exactly matches the path but if we remove exact from the syntax, then UI will still be rendered even if the structure is like /about/10.
        - path: Path specifies a pathname we assign to our component.
        - element: It refers to the component which will render on matching the path.