📆 Hari 3 (Rabu) - React Router

🎯 Tujuan Pembelajaran

- Memahami konsep routing di Single Page Application (SPA).
- Mengimplementasikan navigasi antar halaman menggunakan React Router.
- Memahami cara kerja komponen utama React Router seperti `BrowserRouter`, `Routes`, `Route`, dan `Link`.
- Mengelola parameter URL dan nested routes.

🧠 Materi Inti (2 Jam)

1. Konsep Routing di Aplikasi SPA

- **Apa itu Routing?**
  - Routing adalah proses menentukan bagaimana aplikasi merespons permintaan URL tertentu. Dalam aplikasi web tradisional (MPA), routing ditangani oleh server.
  - Di SPA, routing ditangani di sisi klien (browser) menggunakan JavaScript. Ketika pengguna mengklik link, browser tidak memuat ulang halaman, melainkan JavaScript mencegat event klik dan mengubah konten yang ditampilkan di halaman tanpa meninggalkan halaman utama (`index.html`).
- **Mengapa Butuh Client-Side Routing?**
  - Memberikan pengalaman pengguna yang mulus dan cepat, mirip aplikasi desktop.
  - Memungkinkan navigasi tanpa pemuatan ulang halaman penuh, menghemat bandwidth dan waktu.
  - Mempertahankan state aplikasi saat navigasi (misalnya, state scroll position, data yang sudah dimuat).
- **Bagaimana Client-Side Routing Bekerja?**
  - Menggunakan History API (`pushState`, `replaceState`) untuk memanipulasi URL browser tanpa memicu permintaan server.
  - Mendengarkan event perubahan URL (misalnya, saat tombol back/forward browser diklik).
  - Menampilkan komponen UI yang sesuai berdasarkan URL saat ini.

2. Pengantar dan Implementasi React Router

- **Apa itu React Router?**
  - React Router adalah library routing standar de facto untuk React. Menyediakan komponen-komponen deklaratif untuk mengelola routing di aplikasi React Anda.
- **Instalasi:**
  - React Router tersedia sebagai package npm. Untuk aplikasi web, kita biasanya menggunakan `react-router-dom`.

```
npm install react-router-dom
# atau menggunakan yarn
yarn add react-router-dom
```

- **Komponen Utama React Router:**
  - `BrowserRouter`: Menggunakan History API HTML5 (`pushState`, `replaceState`) untuk menjaga UI tetap sinkron dengan URL. Ini adalah router yang paling umum digunakan untuk aplikasi web.
  - `Routes`: Wadah untuk semua `Route` Anda. Hanya satu `Route` di dalam `Routes` yang akan di-render pada satu waktu, yang cocok dengan lokasi saat ini.
  - `Route`: Mendefinisikan path URL dan komponen yang akan di-render ketika path tersebut cocok dengan lokasi saat ini.
  - `Link`: Komponen untuk membuat link navigasi. Mencegah pemuatan ulang halaman penuh dan menggunakan History API untuk mengubah URL.
  - `useNavigate`: Hook untuk navigasi programatik (misalnya, setelah submit form atau login).
  - `useParams`: Hook untuk membaca parameter dari URL (misalnya, `/users/:id`).

3. Navigasi Antar Halaman dengan React Router

- **Struktur Dasar Aplikasi dengan React Router:**
  - Aplikasi React Anda perlu dibungkus dengan `BrowserRouter` (biasanya di level teratas, misalnya di `src/index.js` atau `src/App.js`).
  - Definisikan rute-rute Anda di dalam komponen `Routes`.
  - Gunakan komponen `Link` untuk navigasi.
- **Contoh Implementasi:**
  - Misalkan kita punya 3 halaman: Home, About, dan Contact.
  - Buat komponen untuk setiap halaman: `Home.js`, `About.js`, `Contact.js`.
  - Di `App.js`, atur routing:

```
// src/App.js
import React from 'react';
import { BrowserRouter as Router, Routes, Route, Link } from
'react-router-dom';
import Home from './pages/Home'; // Asumsikan komponen halaman
ada di folder pages
import About from './pages/About';
import Contact from './pages/Contact';

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

```
          </ul>
        </nav>

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

  export default App;
```

- Buat file komponen halaman sederhana (misalnya, `src/pages/Home.js`):

```
  // src/pages/Home.js
  import React from 'react';

  function Home() {
    return (
      <div>
        <h1>Halaman Utama</h1>
        <p>Selamat datang di aplikasi SPA kami!</p>
      </div>
    );
  }

  export default Home;
```

- Lakukan hal yang sama untuk `About.js` dan `Contact.js`.

- **Navigasi Programatik dengan `useNavigate`:**
  - Kadang kita perlu navigasi berdasarkan logika (misalnya, setelah berhasil login). Gunakan hook `useNavigate`.

```
import { useNavigate } from 'react-router-dom';

function LoginForm() {
  const navigate = useNavigate();

  const handleSubmit = (event) => {
    event.preventDefault();
    // Logika login...
    // Jika berhasil:
    navigate('/dashboard'); // Navigasi ke halaman dashboard
  };

  // ... JSX form
}
```

- **Parameter URL dengan `useParams`:**
  - Untuk rute dinamis seperti `/products/123` atau `/users/john-doe`.
  - Definisikan rute dengan placeholder (`:`):

```
<Route path="/products/:productId" element={<ProductDetail />} />
```

  - Di komponen `ProductDetail`, gunakan `useParams`:

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

function ProductDetail() {
  const { productId } = useParams();

  return (
    <div>
      <h1>Detail Produk</h1>
      <p>ID Produk: {productId}</p>
      {/* Ambil data produk berdasarkan productId */}
    </div>
  );
}
```

- **Nested Routes:**
  - Rute di dalam rute lain. Berguna untuk layout yang kompleks (misalnya, sidebar yang berubah tergantung sub-rute).
  - Definisikan rute anak di dalam rute induk:

```
<Route path="/dashboard" element={<DashboardLayout />}>
  <Route path="overview" element={<DashboardOverview />} />
  <Route path="settings" element={<DashboardSettings />} />
</Route>
```

  - Di komponen `DashboardLayout`, gunakan `<Outlet />` untuk menandai di mana rute anak akan di-render:

```
import { Outlet, Link } from 'react-router-dom';

function DashboardLayout() {
  return (
    <div>
      <h2>Dashboard</h2>
      <nav>
        <Link to="/dashboard/overview">Overview</Link>
```

```
          <Link to="/dashboard/settings">Settings</Link>
        </nav>
        <hr />
        {/* Rute anak akan di-render di sini */}
        <Outlet />
      </div>
    );
  }
```

- URL akan menjadi `/dashboard/overview` dan `/dashboard/settings`.

📝 Praktik Mandiri (8 Jam)

Praktik ini akan memandu Anda membangun aplikasi SPA sederhana dengan navigasi menggunakan React Router.

1. **Setup Project React:**
   - Gunakan project yang sudah ada dari Hari 1 atau buat project baru dengan CRA/Vite.

```
# Jika menggunakan project Hari 1
cd my-spa-exploration
npm install react-router-dom
# Jika project baru
npx create-react-app my-react-router-app
cd my-react-router-app
npm install react-router-dom
# atau dengan Vite
npm create vite@latest my-react-router-app --template react
cd my-react-router-app
npm install
npm install react-router-dom
```

   - **Tujuan:** Memastikan project siap dengan library React Router.
2. **Buat Komponen Halaman:**
   - Buat folder `src/pages`.
   - Di dalamnya, buat 3 file komponen sederhana: `HomePage.js`, `AboutPage.js`, `ContactPage.js`. Masing-masing hanya perlu menampilkan judul halaman (misalnya, `<h1>Home Page</h1>`).
   - **Tujuan:** Menyediakan komponen yang akan dirender oleh router.
3. **Konfigurasi React Router:**
   - Buka `src/App.js`.
   - Import `BrowserRouter`, `Routes`, `Route`, dan `Link` dari `react-router-dom`.
   - Import komponen halaman yang baru dibuat.
   - Bungkus seluruh konten yang akan di-route dengan `BrowserRouter`.

- Di dalam `BrowserRouter`, buat elemen `<nav>` dengan beberapa `<Link>` ke `/`, `/about`, dan `/contact`.
- Di bawah `<nav>`, buat elemen `<Routes>`.
- Di dalam `<Routes>`, definisikan `<Route>` untuk setiap path (`/`, `/about`, `/contact`) dan hubungkan dengan komponen halaman yang sesuai menggunakan prop `element`.
- **Tujuan:** Menyiapkan struktur routing dasar aplikasi.

4. **Uji Navigasi Dasar:**

   - Jalankan aplikasi (`npm start` atau `npm run dev`).
   - Buka di browser.
   - Klik link Home, About, dan Contact di navigasi.
   - Amati URL di address bar browser dan konten halaman yang berubah tanpa pemuatan ulang penuh.
   - Gunakan tombol back/forward browser dan amati bagaimana React Router menangani navigasi ini.
   - **Tujuan:** Memverifikasi bahwa navigasi dasar menggunakan `Link` dan `Route` berfungsi dengan baik.

5. **Implementasi Navigasi Programatik:**

   - Buat komponen baru, misalnya `src/components/RedirectButton.js`. Komponen ini memiliki tombol.
   - Di dalam komponen `RedirectButton`, gunakan hook `useNavigate`.
   - Buat fungsi handler untuk event `onClick` tombol yang memanggil `navigate('/about')` (atau path lain).
   - Import dan gunakan komponen `RedirectButton` di `HomePage.js`.
   - **Tujuan:** Memahami cara melakukan navigasi menggunakan kode JavaScript, bukan hanya dengan mengklik link.

6. **Implementasi Parameter URL:**

   - Buat komponen baru, misalnya `src/pages/UserPage.js`. Komponen ini akan menampilkan ID pengguna.
   - Di dalam `UserPage.js`, gunakan hook `useParams` untuk mendapatkan parameter ID.
   - Di `src/App.js`, tambahkan rute baru: `<Route path="/users/:userId" element={<UserPage />} />`.
   - Di `HomePage.js` atau komponen lain, tambahkan beberapa `<Link>` ke rute ini dengan ID yang berbeda, misalnya `<Link to="/users/1">User 1</Link>` dan `<Link to="/users/john-doe">User John Doe</Link>`.
   - **Tujuan:** Mempelajari cara membuat rute dinamis dan membaca nilai parameter dari URL.

7. **Implementasi Nested Routes (Opsional tapi Direkomendasikan):**

   - Buat folder `src/layouts` dan file `DashboardLayout.js` di dalamnya.
   - Buat folder `src/pages/dashboard` dan file `DashboardOverview.js` serta `DashboardSettings.js` di dalamnya.
   - Di `DashboardLayout.js`, import `Outlet` dan `Link` dari `react-router-dom`. Buat struktur layout dengan navigasi internal ke `/dashboard/overview` dan `/dashboard/settings`, dan

tambahkan `<Outlet />` di tempat konten rute anak akan dirender.

- Di `src/App.js`, ubah rute `/dashboard` menjadi rute induk dengan rute anak:

```
<Route path="/dashboard" element={<DashboardLayout />}>
  <Route path="overview" element={<DashboardOverview />} />
  <Route path="settings" element={<DashboardSettings />} />
  {/* Opsional: rute index untuk /dashboard */}
  <Route index element={<DashboardOverview />} />
</Route>
```

- Tambahkan link ke `/dashboard/overview` atau `/dashboard` di navigasi utama (`<nav>` di `App.js`).
- **Tujuan:** Memahami cara membuat struktur rute yang lebih kompleks dengan layout bersama.

8. **Dokumentasikan Proses Routing:**
   - Tulis catatan singkat tentang:
     - Komponen React Router apa saja yang Anda gunakan dan perannya masing-masing.
     - Bagaimana Anda mengatur rute di `App.js`.
     - Bagaimana cara navigasi menggunakan `Link` dan `useNavigate`.
     - Bagaimana cara membaca parameter URL dengan `useParams`.
     - (Jika dilakukan) Bagaimana cara kerja nested routes dan `<Outlet />`.
   - **Tujuan:** Memperkuat pemahaman konsep melalui refleksi dan dokumentasi.

💡 Tips Belajar Tambahan

- **Pahami Perbedaan `BrowserRouter` dan `HashRouter`:** `BrowserRouter` menggunakan History API (URL terlihat bersih tanpa `#`), sementara `HashRouter` menggunakan hash (`#`) di URL. `BrowserRouter` lebih umum untuk aplikasi modern, tetapi `HashRouter` bisa berguna untuk static site atau lingkungan yang tidak mendukung konfigurasi server untuk SPA routing.
- **Pelajari Konsep `index` Route:** Di nested routes, `<Route index element={<Component />} />` digunakan untuk menentukan komponen default yang akan dirender ketika path induk cocok persis (misalnya, `/dashboard`).
- **Error Handling (404 Page):** Tambahkan rute terakhir di dalam `<Routes>` dengan `path="*"` untuk menangani URL yang tidak cocok dengan rute manapun (halaman 404 Not Found).

```
<Routes>
  {/* ... rute lainnya */}
  <Route path="*" element={<NotFoundPage />} />
</Routes>
```

- **Redirects:** Gunakan komponen `<Navigate replace to="/new-path" />` untuk mengarahkan pengguna dari satu rute ke rute lain.
- **Baca Dokumentasi Resmi React Router:** Dokumentasi di reactrouter.com adalah sumber terbaik untuk detail lebih lanjut dan contoh penggunaan.

📚 Referensi

- React Router Official Docs
- React Router Tutorial (reactrouter.com)
- Client-Side Routing vs Server-Side Routing
- HTML5 History API
- Getting Started with React Router v6