📆 Hari 2 (Selasa) - Kondisi Routing Login/Logout

🎯 Tujuan Pembelajaran Hari Ini

- Memahami masalah akses halaman Login/Register setelah user terautentikasi.
- Mampu mengecek status autentikasi user di komponen React.
- Mengimplementasikan conditional rendering atau redirect pada routing berdasarkan status autentikasi.
- Menggunakan hook `useNavigate` dan `useEffect` dari `react-router-dom` untuk melakukan redirect programatik.

📚 Materi Inti (2 Jam)

1. Masalah: Akses Halaman Login/Register Setelah Login

Di aplikasi web modern yang memiliki alur autentikasi, user yang sudah login seharusnya tidak bisa lagi mengakses halaman seperti `/login` atau `/register`. Jika mereka mencoba mengaksesnya (misalnya, mengetik URL langsung di browser), aplikasi harusnya mengarahkan mereka kembali ke halaman yang relevan untuk user yang sudah login, seperti halaman utama (`/`) atau dashboard (`/dashboard`).

Sebaliknya, halaman atau fitur yang membutuhkan autentikasi (misalnya, halaman profil, halaman checkout, dashboard admin) seharusnya tidak bisa diakses oleh user yang belum login. Jika user yang belum login mencoba mengakses halaman tersebut, mereka harus diarahkan ke halaman login.

Hari ini kita akan fokus pada skenario pertama: mencegah user yang sudah login mengakses halaman Login dan Register.

2. Mengecek Status Autentikasi

Untuk mengontrol akses halaman berdasarkan status login, kita perlu cara untuk mengetahui apakah user saat ini sudah terautentikasi atau belum. Cara paling sederhana, berdasarkan materi Hari 1, adalah dengan memeriksa keberadaan token autentikasi di `localStorage`.

```
// Contoh sederhana mengecek status autentikasi
const isAuthenticated = localStorage.getItem('authToken') !== null;
// Atau bisa juga:
// const token = localStorage.getItem('authToken');
// const isAuthenticated = !!token; // Mengubah string token menjadi
boolean
```

Pengecekan ini bisa dilakukan di komponen halaman Login (`LoginPage.jsx`) dan Register (`RegisterPage.jsx`).

3. Conditional Rendering pada Routing

Salah satu cara untuk menangani ini adalah dengan menambahkan logika di komponen routing utama (misalnya di `App.js` atau file yang mendefinisikan rute Anda). Anda bisa secara kondisional me-render

komponen rute yang berbeda atau menambahkan logika pengecekan di dalam elemen rute itu sendiri.

Namun, pendekatan yang lebih umum dan seringkali lebih bersih, terutama untuk redirect setelah aksi (seperti login), adalah dengan menggunakan hook navigasi di dalam komponen halaman itu sendiri.

4. Redirect Berdasarkan Status

Strategi yang akan kita gunakan adalah: di halaman Login dan Register, saat komponen dimuat, kita akan cek apakah user sudah login (dengan memeriksa `localStorage`). Jika ya, kita akan langsung mengarahkan user ke halaman lain (misalnya `/dashboard`).

- **Jika user sudah login** dan mencoba mengakses `/login` atau `/register`, **redirect ke `/dashboard`**.
- **Jika user belum login**, biarkan mereka tetap di halaman `/login` atau `/register`.

5. Menggunakan `useNavigate` dan `useEffect` untuk Redirect

Untuk melakukan redirect programatik di React Router v6, kita menggunakan hook `useNavigate`. Hook ini mengembalikan fungsi yang bisa kita panggil untuk berpindah halaman.

Untuk mengecek status autentikasi dan melakukan redirect segera setelah komponen dimuat, kita menggunakan hook `useEffect`. `useEffect` memungkinkan kita menjalankan side effect (seperti mengambil data, memanipulasi DOM, atau dalam kasus ini, melakukan navigasi) setelah komponen di-render.

Kita akan menempatkan logika pengecekan `localStorage` dan pemanggilan `navigate` di dalam `useEffect` di komponen `LoginPage.jsx` dan `RegisterPage.jsx`.

```
// Contoh di LoginPage.jsx
import React, { useEffect } from 'react';
import { useNavigate } from 'react-router-dom';

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

  useEffect(() => {
    // Cek apakah token ada di localStorage
    const token = localStorage.getItem('authToken');

    if (token) {
      // Jika token ada (user sudah login), redirect ke dashboard
      navigate('/dashboard', { replace: true }); // Gunakan replace: true
agar halaman login tidak bisa di-back
    }
  }, [navigate]); // Dependency array: effect ini akan dijalankan ulang
jika navigate berubah (biasanya hanya sekali saat mount)

  // ... sisa kode komponen LoginPage (form, state, handleSubmit, dll.)
  return (
    <div>
      <h1>Login Page</h1>
```

```
        {/* Form login di sini */}
      </div>
    );
  }

  export default LoginPage;
```

Logika yang sama akan diterapkan di `RegisterPage.jsx`. Dengan `replace: true`, kita mengganti entri saat ini di history browser, sehingga user tidak bisa menekan tombol 'back' untuk kembali ke halaman login/register setelah di-redirect.

🏋️ Praktik Mandiri (8 Jam)

1. **Lanjutkan proyek e-commerce Anda.** Pastikan Anda memiliki komponen `LoginPage.jsx` dan `RegisterPage.jsx` serta konfigurasi routing untuk `/login` dan `/register` menggunakan `react-router-dom`.
2. **Tambahkan logika redirect di `LoginPage.jsx`:** Impor `useEffect` dan `useNavigate`. Di dalam komponen `LoginPage`, panggil `useNavigate()`. Buat blok `useEffect` yang mengecek `localStorage.getItem('authToken')`. Jika token ada, panggil `navigate('/dashboard', { replace: true });`. Pastikan `[navigate]` ada di dependency array `useEffect`.
3. **Tambahkan logika redirect di `RegisterPage.jsx`:** Lakukan hal yang sama seperti langkah 2 di komponen `RegisterPage.jsx`.
4. **Buat komponen `DashboardPage.jsx` sederhana:** Buat file baru `src/pages/DashboardPage.jsx` yang hanya menampilkan teks sederhana, misalnya `<h1>Selamat Datang di Dashboard!</h1>`.
5. **Konfigurasi route `/dashboard`:** Di file routing utama Anda (misalnya `App.js`), tambahkan route untuk `/dashboard` yang me-render `DashboardPage`.
6. **Uji coba alur redirect:**
    - Pastikan Anda sudah login (token ada di `localStorage`). Coba akses `/login` atau `/register` langsung dari URL browser. Pastikan Anda langsung diarahkan ke `/dashboard`.
    - Logout (hapus token dari `localStorage`). Coba akses `/login` atau `/register`. Pastikan Anda tetap berada di halaman tersebut.
    - Login kembali. Pastikan Anda diarahkan ke `/dashboard`.

💡 Tips Belajar Tambahan

- **Dependency Array `useEffect`:** Memahami kapan `useEffect` dijalankan sangat penting. Dengan dependency array kosong `[]`, effect hanya berjalan sekali setelah render pertama (mirip `componentDidMount`). Dengan `[navigate]`, effect berjalan saat `navigate` berubah (yang jarang terjadi), efektif membuatnya berjalan sekali saat mount.
- **`replace: true`:** Memahami perbedaan antara `navigate('/path')` dan `navigate('/path', { replace: true })` penting untuk mengelola history browser.
- **Simulasi Status:** Jika backend belum sepenuhnya siap, Anda bisa mensimulasikan status login/logout dengan menambahkan/menghapus item `authToken` di `localStorage` secara manual melalui browser developer tools.

🔗 Referensi Tambahan

- React Router Documentation - useNavigate
- React Documentation - useEffect
- MDN Web Docs - Window.localStorage

Hari ini kita sudah berhasil mengontrol akses ke halaman Login dan Register berdasarkan status autentikasi user. Ini adalah langkah awal dalam mengimplementasikan alur autentikasi yang lebih lengkap. Besok, kita akan membahas cara melindungi halaman atau rute yang hanya boleh diakses oleh user yang sudah login.