📆 Hari 3 (Rabu) - Halaman Login dan Register

🎯 Tujuan Pembelajaran Hari Ini

- Memahami konsep dasar autentikasi user (login dan register).
- Mampu membuat form input di React dan mengelola state-nya.
- Menggunakan `axios` untuk mengirim data form (kredensial login, data register) ke API backend.
- Memahami cara menyimpan token autentikasi di frontend.
- Mampu melakukan navigasi programatik menggunakan `react-router-dom` setelah aksi form berhasil.
- Mengimplementasikan penanganan error sederhana untuk form login dan register.

📚 Materi Inti (2 Jam)

1. Konsep Autentikasi Sederhana

- **Proses Login:**
  - User memasukkan kredensial (misalnya, email dan password) di form login.
  - Frontend mengirim kredensial ini ke endpoint login di backend (biasanya menggunakan metode POST).
  - Backend memverifikasi kredensial terhadap data user di database.
  - Jika kredensial valid, backend menghasilkan token autentikasi (misalnya, JWT - JSON Web Token) atau membuat sesi.
  - Backend mengirim token/sesi kembali ke frontend dalam respons.
  - Frontend menyimpan token/sesi ini (misalnya, di `localStorage`, `sessionStorage`, atau Context API/State Management).
  - Token ini akan digunakan untuk otorisasi pada permintaan ke endpoint yang dilindungi di masa mendatang.
- **Proses Register:**
  - User memasukkan data user baru (nama, email, password, dll.) di form register.
  - Frontend mengirim data ini ke endpoint register di backend (biasanya menggunakan metode POST).
  - Backend memvalidasi data (misalnya, cek format email, cek apakah email sudah terdaftar).
  - Jika data valid, backend membuat user baru di database.
  - Backend mengirim respons sukses ke frontend.
  - Frontend biasanya mengarahkan user ke halaman login setelah register berhasil.
- **Pentingnya Keamanan:**
  - **Jangan Simpan Password di Frontend:** Password yang dimasukkan user hanya boleh dikirim ke backend untuk verifikasi (saat login) atau penyimpanan (setelah di-hash, saat register). Jangan pernah menyimpan password dalam bentuk plain text di frontend.
  - **Gunakan HTTPS:** Selalu gunakan protokol HTTPS untuk komunikasi antara frontend dan backend. Ini mengenkripsi data yang dikirim, termasuk kredensial login, sehingga tidak bisa dibaca jika dicegat.
  - **Hashing Password:** Backend harus selalu menyimpan password user dalam bentuk hash (menggunakan algoritma seperti bcrypt), bukan plain text.

- **Validasi Input:** Lakukan validasi input baik di frontend (untuk user experience instan) maupun di backend (untuk keamanan, karena validasi frontend bisa dilewati).

2. Membuat Halaman Login

- **Membuat Komponen `LoginPage.jsx`:**
  - Komponen ini akan berisi form untuk input email/username dan password.

```
// LoginPage.jsx
import React, { useState } from 'react';
import axios from 'axios';
import { useNavigate } from 'react-router-dom'; // Untuk navigasi
programatik

function LoginPage() {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [error, setError] = useState(null);
  const navigate = useNavigate(); // Hook untuk navigasi

  const handleSubmit = async (e) => {
    e.preventDefault(); // Mencegah refresh halaman
    setError(null); // Reset error state

    try {
      // Mengirim kredensial ke endpoint login backend
      const response = await axios.post('/api/auth/login', {
        email,
        password,
      });

      // Asumsikan backend mengembalikan token di response.data.token
      const token = response.data.token;

      // Menyimpan token (contoh menggunakan localStorage)
      localStorage.setItem('token', token);

      // Navigasi ke halaman dashboard atau home setelah login
berhasil
      navigate('/dashboard'); // Ganti dengan path yang sesuai

    } catch (err) {
      // Menangani error dari backend
      console.error('Login failed:', err);
      if (err.response && err.response.data &&
err.response.data.message) {
        setError(err.response.data.message); // Tampilkan pesan error
dari backend
      } else {
        setError('Login failed. Please try again.');
      }
    }
```

```
    };

    return (
      <div>
        <h1>Login Page</h1>
        <form onSubmit={handleSubmit}>
          <div>
            <label>Email:</label>
            <input
              type="email"
              value={email}
              onChange={(e) => setEmail(e.target.value)}
              required
            />
          </div>
          <div>
            <label>Password:</label>
            <input
              type="password"
              value={password}
              onChange={(e) => setPassword(e.target.value)}
              required
            />
          </div>
          {error && <div style={{ color: 'red' }}>{error}</div>} {/*
  Tampilkan error */}
          <button type="submit">Login</button>
        </form>
      </div>
    );
  }

  export default LoginPage;
```

- **Menggunakan State (`useState`) untuk Input Form:**
  - Setiap input field pada form akan memiliki state-nya sendiri yang dikelola oleh `useState`.
  - Event handler `onChange` pada input akan mengupdate state setiap kali nilai input berubah.
- **Menangani Event `onSubmit`:**
  - Event `onSubmit` pada elemen `<form>` akan dipicu saat form disubmit (misalnya, saat tombol submit diklik atau user menekan Enter di salah satu input field).
  - Di dalam handler `onSubmit`, panggil `e.preventDefault()` untuk mencegah browser melakukan refresh halaman default.
  - Lakukan logika pengiriman data ke backend di dalam handler ini.
- **Menggunakan `axios.post` untuk Login:**
  - Gunakan `axios.post` untuk mengirim objek yang berisi kredensial user ke endpoint login backend.
  - Data yang dikirim (request body) adalah objek JavaScript, `axios` akan otomatis mengubahnya menjadi JSON.
- **Menyimpan Token Autentikasi:**
  - Setelah backend merespons sukses dengan token, simpan token tersebut.

    - `localStorage` adalah pilihan sederhana untuk menyimpan token agar tetap ada bahkan setelah browser ditutup, namun kurang aman terhadap serangan XSS. Pilihan lain yang lebih aman (tapi lebih kompleks) adalah menggunakan cookie `HttpOnly` atau menyimpan di memori aplikasi (Context API/State Management) dan hanya menggunakan cookie untuk token refresh.
- **Navigasi Programatik (`useNavigate`):**
    - Hook `useNavigate` dari `react-router-dom` memungkinkan Anda berpindah halaman secara programatik (melalui kode) setelah suatu aksi selesai, seperti setelah login berhasil.
    - Panggil `navigate('/path-tujuan')`.

3. Membuat Halaman Register

- **Membuat Komponen `RegisterPage.jsx`:**
    - Mirip dengan halaman login, komponen ini akan memiliki form untuk data user baru.

```
// RegisterPage.jsx
import React, { useState } from 'react';
import axios from 'axios';
import { useNavigate, Link } from 'react-router-dom'; // Import Link
juga

function RegisterPage() {
  const [name, setName] = useState('');
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [error, setError] = useState(null);
  const navigate = useNavigate();

  const handleSubmit = async (e) => {
    e.preventDefault();
    setError(null);

    try {
      // Mengirim data user baru ke endpoint register backend
      await axios.post('/api/auth/register', {
        name,
        email,
        password,
      });

      // Navigasi ke halaman login setelah register berhasil
      alert('Registration successful! Please login.'); // Pesan sukses
sederhana
      navigate('/login'); // Ganti dengan path halaman login

    } catch (err) {
      // Menangani error dari backend
      console.error('Registration failed:', err);
      if (err.response && err.response.data &&
err.response.data.message) {
        setError(err.response.data.message); // Tampilkan pesan error
dari backend
```

```
        } else {
          setError('Registration failed. Please try again.');
        }
      }
    };

    return (
      <div>
        <h1>Register Page</h1>
        <form onSubmit={handleSubmit}>
          <div>
            <label>Name:</label>
            <input
              type="text"
              value={name}
              onChange={(e) => setName(e.target.value)}
              required
            />
          </div>
          <div>
            <label>Email:</label>
            <input
              type="email"
              value={email}
              onChange={(e) => setEmail(e.target.value)}
              required
            />
          </div>
          <div>
            <label>Password:</label>
            <input
              type="password"
              value={password}
              onChange={(e) => setPassword(e.target.value)}
              required
            />
          </div>
          {error && <div style={{ color: 'red' }}>{error}</div>}
          <button type="submit">Register</button>
        </form>
        <p>Already have an account? <Link to="/login">Login here</Link>
</p> {/* Link ke halaman login */}
      </div>
    );
}

export default RegisterPage;
```

- **Menggunakan `axios.post` untuk Register:**
  - Kirim objek data user baru ke endpoint register backend.
- **Navigasi Setelah Register:**

    - Setelah register berhasil, arahkan user ke halaman login agar mereka bisa langsung mencoba login dengan akun baru mereka.
- **Menambahkan Link Antar Halaman:**
    - Gunakan komponen `<Link>` dari `react-router-dom` untuk membuat tautan navigasi antar halaman Login dan Register.

4. Menangani Error Form dan Validasi Sederhana

- **Menampilkan Pesan Error dari Backend:**
    - Saat backend merespons dengan status error (misalnya, 400 Bad Request, 401 Unauthorized, 409 Conflict), `axios` akan masuk ke blok `catch`.
    - Backend seringkali menyertakan detail error dalam body respons (misalnya, `{"message": "Invalid credentials"}`). Akses ini melalui `err.response.data`.
    - Simpan pesan error ini di state (`error`) dan tampilkan di UI form.
- **Implementasi Validasi Input Sederhana di Frontend:**
    - Lakukan validasi dasar sebelum mengirim data ke backend, misalnya:
        - Memastikan field yang required tidak kosong.
        - Memeriksa format email.
        - Memeriksa panjang minimum password.
    - Validasi ini bisa dilakukan di handler `onSubmit` sebelum memanggil `axios.post`.
    - Tampilkan pesan validasi di bawah input field yang relevan atau di area pesan error umum.

🏋 Praktik Mandiri (8 Jam)

1. **Buat Komponen `LoginPage.jsx`:**
    - Buat form dengan input email/username dan password.
    - Gunakan `useState` untuk mengelola nilai input.
    - Implementasikan handler `onSubmit`.
    - Gunakan `axios.post` untuk mengirim data ke endpoint dummy (misalnya, `https://reqres.in/api/login` untuk simulasi login, atau simulasikan respons sukses/gagal di frontend).
    - Simpan token (jika ada) di `localStorage`.
    - Gunakan `useNavigate` untuk pindah halaman setelah sukses.
    - Tampilkan pesan error jika ada.
2. **Buat Komponen `RegisterPage.jsx`:**
    - Buat form dengan input nama, email, password, dll.
    - Gunakan `useState` untuk mengelola nilai input.
    - Implementasikan handler `onSubmit`.
    - Gunakan `axios.post` untuk mengirim data ke endpoint dummy (misalnya, `https://reqres.in/api/register` untuk simulasi register, atau simulasikan respons sukses/gagal).
    - Gunakan `useNavigate` untuk pindah ke halaman login setelah sukses.
    - Tampilkan pesan error jika ada.
3. **Konfigurasi Routing:**
    - Pastikan `react-router-dom` sudah terinstal dan dikonfigurasi.

    - Tambahkan Route untuk `/login` yang me-render `LoginPage` dan Route untuk `/register` yang me-render `RegisterPage`.
4. **Tambahkan Link:**
    - Tambahkan `<Link>` dari halaman login ke register, dan sebaliknya.
5. **Validasi Sederhana:**
    - Tambahkan validasi sederhana di frontend, misalnya cek apakah field email dan password tidak kosong sebelum submit.
6. **Verifikasi:** Jalankan aplikasi Anda, coba akses halaman login dan register, isi form, dan lihat respons (simulasi) dan navigasinya.

💡 Tips Belajar Tambahan

- **Form Libraries:** Untuk form yang lebih kompleks, pertimbangkan menggunakan library seperti Formik atau React Hook Form yang membantu mengelola state form, validasi, dan submission.
- **Context API atau State Management:** Untuk mengelola status autentikasi (apakah user sudah login atau belum, data user) di seluruh aplikasi, gunakan Context API atau library state management seperti Zustand atau Redux.
- **Protected Routes:** Implementasikan "Protected Routes" yang hanya bisa diakses oleh user yang sudah login. Ini bisa dilakukan dengan memeriksa keberadaan token atau status autentikasi sebelum me-render komponen halaman.

🔗 Referensi Tambahan

- React Documentation - Working with Forms
- React Router DOM - useNavigate
- MDN Web Docs - Sending form data

Hari ini kita sudah berhasil membuat halaman Login dan Register, langkah penting dalam membangun aplikasi yang memerlukan autentikasi. Kita sudah belajar cara mengirim data form menggunakan `axios` dan mengelola navigasi. Besok, kita akan melanjutkan dengan menampilkan detail produk dan bagaimana mengirim header autentikasi untuk mengakses endpoint yang dilindungi.