📆 Hari 2 (Selasa) - Mengambil Data dengan `axios` dan Menampilkan Data Produk

🎯 Tujuan Pembelajaran Hari Ini

- Memahami keunggulan library `axios` dibandingkan native `fetch` API.
- Mampu menginstal dan menggunakan `axios` untuk membuat permintaan HTTP.
- Mengintegrasikan `axios` dengan `useEffect` untuk mengambil data dari API.
- Mampu menampilkan daftar data produk yang diterima dari backend.
- Memecah tampilan daftar produk menjadi komponen `ProductList` dan `ProductItem`.
- Tetap menerapkan penanganan loading dan error state saat menggunakan `axios`.

📚 Materi Inti (2 Jam)

1. Pengantar `axios`

- **Library HTTP Client Berbasis Promise:**
  - `axios` adalah library JavaScript populer yang berfungsi sebagai HTTP client.
  - Ia bekerja baik di browser maupun di lingkungan Node.js.
  - Seperti `fetch`, `axios` juga berbasis Promise, yang memudahkan penanganan operasi asynchronous.
- **Keunggulan `axios` Dibandingkan `fetch`:**
  - **Interceptors:** Memungkinkan Anda mencegat request atau response sebelum ditangani oleh `.then()` atau `.catch()`. Berguna untuk menambahkan header autentikasi secara otomatis, logging, atau transformasi data.
  - **Automatic JSON Transformation:** `axios` secara otomatis mengubah data respons JSON menjadi objek JavaScript, dan data permintaan (request body) dari objek JavaScript menjadi string JSON.
  - **Better Error Handling:** `axios` memperlakukan respons dengan status HTTP non-2xx (misalnya, 404, 500) sebagai error dan langsung masuk ke blok `.catch()`. Ini berbeda dengan `fetch` yang hanya menganggap error jika ada masalah jaringan.
  - **Request Cancellation:** Memiliki fitur bawaan untuk membatalkan permintaan HTTP yang sedang berjalan.
  - **Client-side Protection Against XSRF:** Fitur keamanan bawaan.
  - **Progress Tracking:** Mudah untuk melacak progress upload atau download.

2. Instalasi dan Penggunaan `axios`

- **Instalasi:**
  - Menggunakan npm: `npm install axios`
  - Menggunakan yarn: `yarn add axios`
- **Sintaks Dasar `axios`:**
  - Permintaan GET:

```
axios.get('/api/products')
  .then(response => {
    console.log(response.data); // Data respons ada di
response.data
  })
  .catch(error => {
    console.error(error); // Error status non-2xx langsung masuk
sini
  });
```

- Permintaan POST:

```
axios.post('/api/products', { name: 'New Product', price: 100 })
  .then(response => {
    console.log(response.data);
  })
  .catch(error => {
    console.error(error);
  });
```

- Metode HTTP lainnya: `axios.put()`, `axios.delete()`, dll.

- **Menggunakan `async/await` dengan `axios`:**
  - Sama seperti `fetch`, `axios` bekerja sangat baik dengan `async/await`.

```
async function createProduct(productData) {
  try {
    const response = await axios.post('/api/products', productData);
    console.log(response.data);
  } catch (error) {
    console.error('Error creating product:', error);
    // error.response.data berisi data error dari backend
    // error.response.status berisi status HTTP
  }
}
```

- **Menangani Response dan Error dengan `axios`:**
  - Data respons dari server tersedia di `response.data`.
  - Objek `error` di blok `catch` `axios` lebih informatif dibandingkan `fetch`. Anda bisa mengakses `error.response` untuk detail respons dari server (status, data, headers) jika error berasal dari respons non-2xx.

3. Menampilkan Data Produk

- **Struktur Data Produk:**
  - Asumsikan API backend mengembalikan array objek produk seperti ini:

```
[
  { "id": 1, "name": "Laptop", "price": 1200, "description": "Powerful
laptop", "imageUrl": "/images/laptop.jpg" },
  { "id": 2, "name": "Mouse", "price": 25, "description": "Wireless
mouse", "imageUrl": "/images/mouse.jpg" }
  // ... produk lainnya
]
```

- **Membuat Komponen `ProductList.jsx`:**
  - Komponen ini bertanggung jawab untuk mengambil data daftar produk dan menampilkannya.
  - Akan menggunakan `useState` untuk state `products`, `loading`, dan `error`.
  - Akan menggunakan `useEffect` untuk memanggil fungsi fetching data saat komponen mount.
  - Akan menggunakan `axios.get` untuk mengambil data dari endpoint `/api/products`.
- **Melakukan Mapping Data Produk:**
  - Setelah data produk berhasil diambil dan disimpan di state `products` (yang berupa array), kita bisa menggunakan method array `map()` untuk mengiterasi array tersebut.
  - Untuk setiap objek produk dalam array, kita akan me-render elemen UI yang sesuai.
- **Membuat Komponen `ProductItem.jsx`:**
  - Untuk menjaga komponen `ProductList` tetap rapi dan reusable, kita akan membuat komponen terpisah, `ProductItem`, yang bertanggung jawab menampilkan detail *satu* produk.
  - Komponen `ProductItem` akan menerima objek produk sebagai props.

```
// ProductItem.jsx
import React from 'react';

function ProductItem({ product }) {
  return (
    <div style={{ border: '1px solid #ccc', padding: '10px', margin:
'10px' }}>
      <img src={product.imageUrl} alt={product.name} style={{ width:
'100px' }} />
      <h3>{product.name}</h3>
      <p>Price: ${product.price}</p>
      <p>{product.description}</p>
    </div>
  );
}

export default ProductItem;
```

```
// ProductList.jsx (Contoh integrasi dengan axios dan useEffect)
import React, { useEffect, useState } from 'react';
import axios from 'axios';
import ProductItem from './ProductItem'; // Import komponen
ProductItem

function ProductList() {
```

```
  const [products, setProducts] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  useEffect(() => {
    async function fetchProducts() {
      try {
        const response = await axios.get('/api/products'); //
Menggunakan axios
        setProducts(response.data); // Data ada di response.data
      } catch (err) {
        setError(err); // axios menangani error non-2xx di sini
      } finally {
        setLoading(false);
      }
    }

    fetchProducts();

  }, []); // Dependency array kosong

  if (loading) {
    return <div>Loading products...</div>;
  }

  if (error) {
    return <div>Error loading products: {error.message}</div>;
  }

  return (
    <div>
      <h1>Product List</h1>
      <div style={{ display: 'flex', flexWrap: 'wrap' }}>
        {products.map(product => (
          // Menggunakan komponen ProductItem untuk setiap produk
          <ProductItem key={product.id} product={product} />
        ))}
      </div>
    </div>
  );
}

export default ProductList;
```

🏋 Praktik Mandiri (8 Jam)

1. **Instal `axios`:** Tambahkan `axios` ke proyek React Anda (`npm install axios` atau `yarn add axios`).
2. **Buat Komponen `ProductItem.jsx`:** Buat komponen fungsional sederhana yang menerima prop `product` dan menampilkan nama, harga, dan gambar produk.
3. **Buat Komponen `ProductList.jsx`:**

    - Gunakan `useState` untuk mengelola state `products`, `loading`, dan `error`.
    - Gunakan `useEffect` dengan dependency array kosong (`[]`).
    - Di dalam `useEffect`, buat fungsi `async` yang menggunakan `axios.get` untuk mengambil data dari dummy API produk. Anda bisa menggunakan:
        - Endpoint dummy publik seperti `https://fakestoreapi.com/products`.
        - Atau membuat file JSON lokal di folder `public` proyek Anda dan mengambilnya (misalnya, `axios.get('/products.json')`).
    - Tangani respons sukses dengan mengupdate state `products` (`setProducts(response.data)`).
    - Tangani error dengan mengupdate state `error` (`setError(err)`).
    - Atur state `loading` menjadi `false` di blok `finally`.
    - Di bagian render, tampilkan "Loading..." jika `loading` true, pesan error jika `error` tidak null, dan daftar produk menggunakan `map` dan komponen `ProductItem` jika data sudah siap.
4. **Tampilkan di Aplikasi:** Render komponen `ProductList` di komponen utama aplikasi Anda (misalnya, di `App.js`).
5. **Verifikasi:** Jalankan aplikasi dan pastikan daftar produk muncul dengan benar.
6. **(Opsional) Styling:** Tambahkan styling sederhana (misalnya menggunakan CSS biasa atau Tailwind CSS jika sudah di-setup) agar tampilan daftar produk lebih menarik.

💡 Tips Belajar Tambahan

- **Konfigurasi `axios` Global:** Anda bisa membuat instance `axios` dengan konfigurasi default (misalnya, base URL API) menggunakan `axios.create()`.
- **Interceptors:** Pelajari cara menggunakan request interceptors untuk menambahkan header autentikasi secara otomatis ke setiap permintaan yang memerlukan.
- **Error Response Detail:** Saat error terjadi, eksplorasi objek `error` yang diberikan oleh `axios`, terutama `error.response`, `error.request`, dan `error.message` untuk debugging.

🔗 Referensi Tambahan

- Dokumentasi Resmi Axios
- Contoh Penggunaan Axios dengan React Hooks

Hari ini kita sudah beralih ke `axios` yang lebih powerful untuk fetching data dan berhasil menampilkan daftar produk. Besok, kita akan menggunakan `axios` untuk membangun halaman Login dan Register, berinteraksi dengan endpoint autentikasi di backend.