📆 Hari 4 (Kamis) - Proses Checkout (Frontend ke Backend)

Tujuan Pembelajaran (2 Jam)

Setelah menyelesaikan materi ini, santri diharapkan mampu:

- Memahami alur proses checkout dari sisi frontend hingga interaksi dengan backend.
- Mengidentifikasi data yang diperlukan untuk proses checkout.
- Mengirim data keranjang dari frontend ke backend menggunakan `axios` atau `fetch`.
- Menangani respons dari backend (sukses atau gagal) dan memberikan umpan balik kepada pengguna.
- Mensimulasikan proses checkout jika backend nyata belum tersedia.

Materi Inti (2 Jam)

Setelah pengguna selesai memilih produk dan mengatur kuantitas di keranjang, langkah selanjutnya adalah menyelesaikan pembelian melalui proses checkout. Proses ini melibatkan pengiriman data keranjang dari frontend ke backend untuk diproses menjadi sebuah order.

1. Alur Checkout Frontend ke Backend Secara umum, alur checkout melibatkan langkah-langkah berikut:

- Pengguna mengklik tombol "Checkout" di halaman keranjang.
- Frontend mengumpulkan data yang relevan (isi keranjang, mungkin informasi pengiriman, dll.).
- Frontend mengirimkan data ini ke endpoint API spesifik di backend (misalnya, `/api/orders`).
- Backend menerima data, melakukan validasi, memproses order (misalnya, mengurangi stok, mencatat transaksi di database), dan mengembalikan respons (sukses atau gagal).
- Frontend menerima respons dan memberikan umpan balik kepada pengguna (misalnya, menampilkan halaman sukses order atau pesan error).

2. Data yang Dikirim saat Checkout Data minimal yang perlu dikirim ke backend biasanya adalah daftar item di keranjang, termasuk `productId` dan `quantity` untuk setiap item. Data tambahan bisa mencakup:

- `userId` (jika pengguna login)
- Alamat pengiriman
- Metode pembayaran (jika relevan)
- Kode kupon (jika ada)

Struktur data yang dikirim biasanya berupa objek JSON, misalnya:

```
{
  "userId": "user123",
  "shippingAddress": {
    "street": "Jl. Contoh No. 1",
    "city": "Kota Contoh",
    "postalCode": "12345"
  },
  "items": [
    { "productId": 1, "quantity": 2 },
    { "productId": 5, "quantity": 1 }
```

```
    ]
  }
```

3. Membuat Endpoint API di Backend (Simulasi/Nyata) Jika Anda memiliki backend Express.js, Anda perlu membuat route dan controller untuk menangani request POST ke endpoint checkout (misalnya, `/api/orders`). Controller ini akan menerima data dari frontend, memprosesnya, dan menyimpan order ke database.

Jika backend nyata belum ada, kita bisa mensimulasikan respons backend di frontend menggunakan `setTimeout` untuk menunda respons dan berpura-pura ada proses di backend.

4. Mengirim Data dari Frontend Kita akan menggunakan library seperti `axios` atau fungsi bawaan `fetch` untuk mengirim data ke backend. Request yang umum digunakan untuk membuat resource baru (seperti order) adalah POST.

Contoh menggunakan `axios`:

```
import axios from 'axios';

const handleCheckout = async (cartItems, userId, shippingAddress) => {
  const orderData = {
    userId: userId, // Ambil dari state auth atau Context
    shippingAddress: shippingAddress, // Ambil dari form input atau state
    items: cartItems.map(item => ({
      productId: item.id,
      quantity: item.quantity
    })),
  };

  try {
    const response = await axios.post('/api/orders', orderData);
    console.log('Checkout successful:', response.data);
    // Lakukan sesuatu setelah sukses, misalnya kosongkan keranjang dan
redirect
    return { success: true, data: response.data };
  } catch (error) {
    console.error('Checkout failed:', error);
    // Lakukan sesuatu jika gagal, misalnya tampilkan pesan error
    return { success: false, error: error.message };
  }
};
```

Contoh menggunakan `fetch`:

```
const handleCheckout = async (cartItems, userId, shippingAddress) => {
  const orderData = {
    userId: userId,
    shippingAddress: shippingAddress,
    items: cartItems.map(item => ({
```

```
        productId: item.id,
        quantity: item.quantity
      })),
    };

    try {
      const response = await fetch('/api/orders', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          // Tambahkan header Authorization jika perlu
          // 'Authorization': `Bearer ${yourAuthToken}`
        },
        body: JSON.stringify(orderData),
      });

      if (!response.ok) {
        throw new Error(`HTTP error! status: ${response.status}`);
      }

      const result = await response.json();
      console.log('Checkout successful:', result);
      // Lakukan sesuatu setelah sukses
      return { success: true, data: result };
    } catch (error) {
      console.error('Checkout failed:', error);
      // Lakukan sesuatu jika gagal
      return { success: false, error: error.message };
    }
  };
```

Penjelasan:

- Kita menyiapkan objek `orderData` yang berisi informasi yang dibutuhkan backend.
- Menggunakan `axios.post` atau `fetch` dengan method 'POST' dan mengirim `orderData` di body request.
- Menggunakan `async/await` untuk menangani operasi asynchronous.
- Menggunakan blok `try...catch` untuk menangani potensi error saat request.
- Memeriksa `response.ok` saat menggunakan `fetch` untuk memastikan request berhasil (status code 2xx).

5. Menangani Respons dan Umpan Balik Setelah menerima respons dari backend, frontend perlu bertindak sesuai status respons:

- **Sukses**: Biasanya backend akan mengembalikan detail order yang berhasil dibuat. Frontend bisa mengosongkan state keranjang (karena item sudah dibeli) dan mengarahkan pengguna ke halaman konfirmasi atau detail order.
- **Gagal**: Backend akan mengembalikan status error dan mungkin pesan kesalahan. Frontend harus menangkap error ini dan menampilkan pesan yang informatif kepada pengguna (misalnya, "Stok tidak cukup", "Alamat tidak valid", dll.).

Praktik Mandiri (8 Jam)

1. **Buat Tombol Checkout**: Di halaman keranjang (`CartPage`), tambahkan tombol "Checkout".
2. **Buat Fungsi `handleCheckout`**: Buat fungsi asynchronous `handleCheckout` di komponen `CartPage` atau di Context API jika Anda ingin logika checkout terpusat.
3. **Ambil Data Keranjang**: Di dalam `handleCheckout`, akses state keranjang dari Context API atau state manager Anda.
4. **Siapkan Data untuk Backend**: Format data keranjang menjadi objek yang siap dikirim ke backend, sesuai dengan struktur yang Anda harapkan di backend (atau struktur simulasi).
5. **Kirim Data ke Backend**: Gunakan `axios.post` atau `fetch` untuk mengirim data yang sudah disiapkan ke endpoint checkout Anda. Jika tidak ada backend nyata, lewati langkah ini dan langsung ke simulasi respons.
6. **Simulasi Respons Backend (Jika Perlu)**: Jika tidak ada backend, di dalam `handleCheckout`, gunakan `setTimeout` untuk mensimulasikan penundaan (misalnya 2 detik), lalu simulasikan respons sukses (misalnya, kosongkan keranjang dan tampilkan alert "Checkout Berhasil!") atau respons gagal (tampilkan alert "Checkout Gagal!").
7. **Tangani Respons Nyata (Jika Ada Backend)**: Jika Anda mengirim request ke backend nyata, tangani respons sukses (kosongkan keranjang, redirect) dan respons gagal (tampilkan pesan error) menggunakan blok `try...catch`.
8. **Kosongkan Keranjang Setelah Sukses**: Setelah checkout berhasil (nyata atau simulasi), panggil fungsi untuk mengosongkan state keranjang Anda (misalnya, tambahkan fungsi `clearCart` di Context).
9. **Uji Coba**: Jalankan aplikasi. Tambahkan beberapa item ke keranjang. Buka halaman keranjang. Klik tombol "Checkout". Pastikan proses berjalan (atau simulasi berjalan) dan keranjang dikosongkan setelah sukses.

Tips Belajar

- Pastikan Anda memahami perbedaan antara `fetch` dan `axios` serta cara menangani respons dan error di keduanya.
- Pertimbangkan *loading state* saat proses checkout sedang berjalan untuk memberikan umpan balik visual kepada pengguna (misalnya, disable tombol "Checkout" atau tampilkan spinner).
- Untuk backend nyata, pastikan endpoint checkout Anda aman dan melakukan validasi data yang diterima dari frontend.

Referensi Teknis

- Making Requests - Axios Docs
- Using Fetch - MDN Web Docs
- Handling Asynchronous Operations in React with useEffect and async/await
- HTTP POST Method - MDN Web Docs