📆 Hari 2 (Selasa) - Menampilkan Isi Keranjang

🎯 Tujuan Pembelajaran Hari Ini

- Mampu mengakses state keranjang belanja yang dikelola oleh Context API (atau state manager lain).
- Mampu membuat komponen UI untuk menampilkan daftar item di keranjang.
- Mampu menampilkan detail setiap item keranjang (nama, harga, kuantitas, subtotal).
- Mampu menghitung dan menampilkan total harga keseluruhan item di keranjang.

📚 Materi Inti (2 Jam)

1. Mengakses State Keranjang

Di Hari 1, kita sudah membuat `CartContext` dan `CartProvider` untuk mengelola state `cartItems`. Sekarang, di komponen yang perlu menampilkan isi keranjang, kita bisa mengakses state tersebut menggunakan hook `useCart()` yang sudah kita definisikan.

```
// Contoh di CartPage.jsx (komponen yang akan kita buat)
import React from 'react';
import { useCart } from '../contexts/CartContext'; // Import hook useCart

const CartPage = () => {
  const { cartItems } = useCart(); // Ambil state cartItems dari Context

  // ... logika untuk menampilkan item dan total ...

  return (
    <div>
      <h1>Keranjang Belanja</h1>
      {/* ... render daftar item ... */}
      {/* ... tampilkan total harga ... */}
    </div>
  );
};

export default CartPage;
```

Dengan memanggil `useCart()`, kita mendapatkan akses ke objek `value` yang disediakan oleh `CartProvider`, termasuk array `cartItems`.

2. Membuat Komponen untuk Menampilkan Item Keranjang (`CartItem`, `CartList`)

Untuk menjaga kode tetap bersih dan modular, kita sebaiknya memisahkan tampilan setiap item keranjang ke dalam komponen tersendiri, misalnya `CartItem.jsx`. Komponen ini akan menerima satu objek item keranjang sebagai prop dan bertanggung jawab menampilkan detailnya.

Komponen parent, misalnya `CartPage.jsx`, akan menggunakan komponen `CartItem` ini dalam sebuah daftar (`CartList` bisa jadi komponen terpisah lagi jika daftarnya kompleks, tapi untuk saat ini cukup di `CartPage`).

Struktur komponen bisa seperti ini:

```
src/
  components/
    CartItem.jsx
    Header.jsx // Untuk link ke CartPage
  contexts/
    CartContext.js
  pages/
    CartPage.jsx
    // ... halaman lain
```

3. Menampilkan Detail Setiap Item

Komponen `CartItem` akan menerima satu objek item dari array `cartItems`. Objek ini memiliki struktur `{ product: {...}, quantity: N }`. Komponen `CartItem` akan menampilkan informasi dari objek `product` dan nilai `quantity`.

```jsx
// src/components/CartItem.jsx
import React from 'react';

const CartItem = ({ item }) => {
  const { product, quantity } = item;
  const itemSubtotal = product.price * quantity; // Hitung subtotal per
item

  return (
    <div style={{ border: '1px solid #ccc', margin: '10px', padding: '10px'
}}>
      <h4>{product.name}</h4>
      <p>Harga: Rp {product.price.toLocaleString()}</p>
      <p>Kuantitas: {quantity}</p>
      <p>Subtotal: Rp {itemSubtotal.toLocaleString()}</p>
      {/* Tombol update/delete akan ditambahkan besok */}
    </div>
  );
};

export default CartItem;
```

Di komponen `CartPage`, kita akan me-render daftar `CartItem` dengan melakukan map pada array `cartItems`:

```jsx
// src/pages/CartPage.jsx
import React from 'react';
import { useCart } from '../contexts/CartContext';
import CartItem from '../components/CartItem'; // Import CartItem

const CartPage = () => {
  const { cartItems } = useCart();

  return (
    <div>
      <h1>Keranjang Belanja</h1>
      {
        cartItems.length === 0 ? (
          <p>Keranjang Anda kosong.</p>
        ) : (
          <div>
            {cartItems.map((item) => (
              <CartItem key={item.product.id} item={item} /> // Render
CartItem untuk setiap item
            ))}
          </div>
        )
      }
      {/* ... total harga akan ditambahkan di bagian selanjutnya ... */}
    </div>
  );
};

export default CartPage;
```

Penting: Saat me-render daftar elemen dari array menggunakan `map`, selalu sertakan prop `key` dengan nilai unik untuk setiap elemen (dalam kasus ini, `item.product.id`). Ini membantu React mengidentifikasi elemen mana yang berubah, ditambahkan, atau dihapus, yang penting untuk performa dan stabilitas UI.

4. Menampilkan Total Harga Keranjang

Untuk menampilkan total harga semua item di keranjang, kita perlu mengiterasi array `cartItems` dan menjumlahkan subtotal dari setiap item (`product.price * quantity`). Kita bisa menggunakan method array seperti `reduce` untuk melakukan ini.

```jsx
// src/pages/CartPage.jsx (Lanjutan)
import React from 'react';
import { useCart } from '../contexts/CartContext';
import CartItem from '../components/CartItem';

const CartPage = () => {
  const { cartItems } = useCart();

  // Hitung total harga
  const totalCartPrice = cartItems.reduce((total, item) => {
```

```
      return total + (item.product.price * item.quantity);
    }, 0); // Nilai awal total adalah 0

    return (
      <div>
        <h1>Keranjang Belanja</h1>
        {
          cartItems.length === 0 ? (
            <p>Keranjang Anda kosong.</p>
          ) : (
            <div>
              {cartItems.map((item) => (
                <CartItem key={item.product.id} item={item} />
              ))}
              {/* Tampilkan total harga */}
              <div style={{ marginTop: '20px', fontWeight: 'bold' }}>
                Total Belanja: Rp {totalCartPrice.toLocaleString()}
              </div>
            </div>
          )
        }
      </div>
    );
  };

  export default CartPage;
```

Method `reduce` sangat berguna untuk menghitung nilai tunggal dari sebuah array. Dalam kasus ini, kita menggunakannya untuk menjumlahkan subtotal setiap item menjadi total keseluruhan.

🏋️ Praktik Mandiri (8 Jam)

1. **Lanjutkan proyek e-commerce Anda.** Pastikan Anda sudah menyelesaikan praktik Hari 1 dan state `cartItems` di `CartContext` sudah bisa diisi saat menambah produk.
2. **Buat komponen `CartItem.jsx`:** Buat file baru `src/components/CartItem.jsx` dan salin kode komponen `CartItem` seperti contoh di atas. Sesuaikan styling jika Anda menggunakan Tailwind CSS atau styling lainnya.
3. **Buat halaman `CartPage.jsx`:** Buat file baru `src/pages/CartPage.jsx` dan salin kode dasar komponen `CartPage`. Impor `useCart` dan `CartItem`.
4. **Konfigurasi Routing:** Di file routing utama Anda (misalnya `App.js`), tambahkan route baru untuk `/cart` yang me-render `CartPage`.

   ```
   // src/App.js (atau file routing utama Anda)
   import { BrowserRouter as Router, Routes, Route } from 'react-router-
   dom';
   // ... import komponen lain ...
   import CartPage from './pages/CartPage'; // Import CartPage

   function App() {
   ```

```
    return (
      <Router>
        {/* Pastikan AuthProvider dan CartProvider membungkus Router
atau komponen root */}
        <Routes>
          {/* ... rute lain ... */}
          <Route path="/cart" element={<CartPage />} /> {/* Tambahkan
rute CartPage */}
          {/* ... rute terproteksi jika ada ... */}
        </Routes>
      </Router>
    );
}

export default App;
```

5. **Tambahkan Link ke Cart:** Di komponen Header/Navbar Anda, tambahkan `<Link to="/cart">Keranjang ({cartItems.length})</Link>`. Anda perlu mengimpor `useCart()` di Header untuk mendapatkan jumlah item (`cartItems.length`) dan menampilkannya di samping link.
6. **Implementasikan tampilan daftar item:** Di `CartPage.jsx`, gunakan method `map` pada `cartItems` untuk me-render komponen `CartItem` untuk setiap item. Jangan lupa prop `key`.
7. **Implementasikan perhitungan dan tampilan total harga:** Di `CartPage.jsx`, gunakan method `reduce` untuk menghitung total harga dan tampilkan di bawah daftar item.
8. **Uji coba tampilan keranjang:**
    - Tambahkan beberapa produk ke keranjang dari halaman produk.
    - Navigasi ke halaman `/cart`.
    - **Verifikasi:** Anda seharusnya melihat daftar produk yang sudah ditambahkan, kuantitasnya, subtotal per item, dan total harga keseluruhan.
    - Coba tambahkan/hapus item (meskipun fungsi update/delete belum diimplementasikan), perhatikan bagaimana tampilan di halaman cart otomatis update karena menggunakan state dari Context.

💡 Tips Belajar Tambahan

- **Styling:** Gunakan CSS atau Tailwind CSS untuk membuat tampilan halaman keranjang lebih menarik dan terstruktur.
- **State Kosong:** Pastikan halaman keranjang menampilkan pesan yang informatif saat keranjang kosong (`cartItems.length === 0`).
- **Format Mata Uang:** Gunakan `toLocaleString()` untuk memformat harga agar lebih mudah dibaca (misalnya, menambahkan pemisah ribuan).
- **Komponen Reusable:** Komponen `CartItem` adalah contoh bagus dari komponen reusable yang menerima data melalui props dan menampilkan UI berdasarkan data tersebut.

🔗 Referensi Tambahan

- React Documentation - Lists and Keys

- MDN Web Docs - Array.prototype.map()
- MDN Web Docs - Array.prototype.reduce()
- MDN Web Docs - Number.prototype.toLocaleString()

Hari ini kita telah berhasil menampilkan isi keranjang belanja dan menghitung total harganya. Ini adalah langkah penting untuk memberikan feedback visual kepada user mengenai item yang telah mereka pilih. Besok, kita akan menambahkan fungsionalitas untuk mengupdate kuantitas item dan menghapus item dari keranjang.