📆 Hari 3 (Rabu) - Update & Delete Item Keranjang

Tujuan Pembelajaran (2 Jam)

Setelah menyelesaikan materi ini, santri diharapkan mampu:

- Memahami kebutuhan untuk memodifikasi (mengubah kuantitas) atau menghapus item dari keranjang belanja.
- Mengimplementasikan fungsi `updateQuantity` dan `removeFromCart` dalam state manager (Context API).
- Mengintegrasikan fungsi-fungsi tersebut dengan antarmuka pengguna (UI) di halaman keranjang.
- Memastikan state keranjang diperbarui secara immutably.

Materi Inti (2 Jam)

Pada hari sebelumnya, kita sudah berhasil menambahkan produk ke keranjang dan menampilkannya. Sekarang, kita akan menambahkan fungsionalitas untuk mengelola item yang sudah ada di keranjang, yaitu mengubah kuantitas dan menghapusnya.

1. Kebutuhan Update & Delete Item Dalam skenario e-commerce nyata, pengguna seringkali perlu mengubah jumlah produk yang ingin dibeli atau bahkan membatalkan pembelian item tertentu. Fungsionalitas ini krusial untuk pengalaman pengguna yang baik.

2. Membuat Fungsi `updateQuantity` Fungsi ini akan menerima `productId` dan `newQuantity` sebagai argumen. Logikanya adalah mencari item di dalam array state keranjang berdasarkan `productId` dan memperbarui properti `quantity`-nya. Penting untuk selalu memperbarui state secara *immutable*, artinya kita tidak boleh langsung memodifikasi array atau objek state yang sudah ada. Kita harus membuat salinan baru.

Contoh implementasi di `CartContext` (menggunakan `useReducer` atau `useState`):

```
// Asumsi state keranjang adalah array seperti ini:
// [{ id: 1, name: 'Produk A', price: 100, quantity: 2 }, ...]

const updateQuantity = (productId, newQuantity) => {
  // Pastikan kuantitas tidak kurang dari 1
  const quantityToUpdate = Math.max(1, newQuantity);

  setCartItems(prevItems =>
    prevItems.map(item =>
      item.id === productId ? { ...item, quantity: quantityToUpdate } :
item
    )
  );
};

// Atau jika menggunakan useReducer:
// dispatch({ type: 'UPDATE_QUANTITY', payload: { productId, newQuantity }
});
// Dalam reducer:
// case 'UPDATE_QUANTITY':
```

```
//   return state.map(item =>
//     item.id === action.payload.productId ? { ...item, quantity:
Math.max(1, action.payload.newQuantity) } : item
//   );
```

Penjelasan:

- Kita menggunakan `map` untuk mengiterasi seluruh item di keranjang.
- Jika `item.id` cocok dengan `productId` yang ingin diupdate, kita membuat objek baru (`{ ...item, quantity: quantityToUpdate }`) dengan kuantitas yang diperbarui.
- Jika tidak cocok, item dikembalikan apa adanya.
- `Math.max(1, newQuantity)` memastikan kuantitas minimal adalah 1.

3. Membuat Fungsi `removeFromCart` Fungsi ini akan menerima `productId` sebagai argumen. Logikanya adalah menghapus item dari array state keranjang berdasarkan `productId`. Lagi-lagi, kita harus melakukannya secara *immutable*.

Contoh implementasi di `CartContext`:

```
const removeFromCart = (productId) => {
  setCartItems(prevItems =>
    prevItems.filter(item => item.id !== productId)
  );
};

// Atau jika menggunakan useReducer:
// dispatch({ type: 'REMOVE_FROM_CART', payload: { productId } });
// Dalam reducer:
// case 'REMOVE_FROM_CART':
//   return state.filter(item => item.id !== action.payload.productId);
```

Penjelasan:

- Kita menggunakan `filter` untuk membuat array baru yang hanya berisi item-item yang `id`-nya tidak cocok dengan `productId` yang ingin dihapus.

4. Mengintegrasikan dengan UI Di komponen `CartItem` (yang menampilkan detail satu item di keranjang), kita perlu menambahkan elemen UI untuk memicu fungsi `updateQuantity` dan `removeFromCart`. Ini bisa berupa tombol (+) dan (-) di samping kuantitas, atau input number, dan tombol "Hapus".

Contoh di komponen `CartItem.js`:

```
import React from 'react';
import { useCart } from '../context/CartContext'; // Asumsi hook useCart

function CartItem({ item }) {
  const { updateQuantity, removeFromCart } = useCart();

  const handleQuantityChange = (e) => {
```

```
    const newQuantity = parseInt(e.target.value, 10);
    if (!isNaN(newQuantity)) {
      updateQuantity(item.id, newQuantity);
    }
  };

  const handleIncrement = () => {
    updateQuantity(item.id, item.quantity + 1);
  };

  const handleDecrement = () => {
    updateQuantity(item.id, item.quantity - 1);
  };

  const handleRemove = () => {
    removeFromCart(item.id);
  };

  return (
    <div className="cart-item">
      <h3>{item.name}</h3>
      <p>Harga: Rp{item.price.toLocaleString('id-ID')}</p>
      <div>
        <button onClick={handleDecrement}>-</button>
        <input
          type="number"
          value={item.quantity}
          onChange={handleQuantityChange}
          min="1"
          className="w-12 text-center"
        />
        <button onClick={handleIncrement}>+</button>
      </div>
      <p>Subtotal: Rp{(item.price * item.quantity).toLocaleString('id-ID')}
</p>
      <button onClick={handleRemove} className="text-red-
500">Hapus</button>
    </div>
  );
}

export default CartItem;
```

Penjelasan:

- Kita mengambil fungsi `updateQuantity` dan `removeFromCart` dari Context menggunakan hook `useCart`.
- Tombol (+) dan (-) memanggil `updateQuantity` dengan kuantitas yang ditambah/dikurangi 1.
- Input number memanggil `updateQuantity` saat nilainya berubah.
- Tombol "Hapus" memanggil `removeFromCart`.

Praktik Mandiri (8 Jam)

1. **Modifikasi Komponen `CartItem`**: Buka kembali komponen `CartItem` yang Anda buat kemarin. Tambahkan tombol (+) dan (-) atau input number untuk kuantitas, serta tombol "Hapus". Gunakan Tailwind CSS untuk styling agar terlihat rapi.
2. **Implementasikan `updateQuantity`**: Di dalam `CartContext` (atau file state manager Anda), tambahkan fungsi `updateQuantity` sesuai contoh di atas. Pastikan logika update state dilakukan secara immutable.
3. **Implementasikan `removeFromCart`**: Di dalam `CartContext`, tambahkan fungsi `removeFromCart` sesuai contoh di atas. Pastikan logika penghapusan state dilakukan secara immutable.
4. **Hubungkan UI dengan Fungsi**: Di komponen `CartItem`, panggil fungsi `updateQuantity` dan `removeFromCart` yang sudah Anda sediakan di Context melalui hook `useCart` (atau cara lain sesuai state manager Anda).
5. **Uji Coba**: Jalankan aplikasi Anda. Buka halaman keranjang. Coba ubah kuantitas item, pastikan subtotal dan total harga keranjang ikut berubah. Coba hapus item dari keranjang, pastikan item tersebut hilang dari daftar dan total harga diperbarui.

Tips Belajar

- Perhatikan baik-baik konsep *immutability* saat mengupdate state array atau object di React. Ini adalah pola penting untuk menghindari bug dan memastikan React me-render ulang komponen dengan benar.
- Gunakan React Developer Tools untuk memeriksa state keranjang Anda saat melakukan operasi update dan delete. Pastikan state berubah sesuai harapan.
- Jika menggunakan `useReducer`, pastikan logika update dan delete berada di dalam fungsi reducer Anda.

Referensi Teknis

- Updating Arrays in State - React Docs
- Updating Objects in State - React Docs
- useContext Hook - React Docs
- useReducer Hook - React Docs