📆 Hari 5 (Jum'at) - Simulasi Transaksi & Tugas Mingguan

Tujuan Pembelajaran (2 Jam)

Setelah menyelesaikan materi ini, santri diharapkan mampu:

- Mereview kembali seluruh alur fitur Cart & Checkout yang telah dipelajari.
- Memahami konsep simulasi transaksi sederhana di frontend sebagai alternatif jika backend nyata belum tersedia.
- Mempersiapkan diri untuk mengerjakan Tugas Mingguan dengan memahami semua persyaratan fungsional dan opsional.

Materi Inti (2 Jam)

Pada hari ini, kita akan mereview kembali seluruh materi Cart & Checkout dari Hari 1 hingga Hari 4, serta mempersiapkan diri untuk tugas mingguan.

1. Review Alur Cart & Checkout Mari kita rekap kembali perjalanan kita dalam membangun fitur keranjang belanja dan checkout:

- **Hari 1**: Kita memulai dengan konsep dasar state keranjang, memilih Context API sebagai state manager, dan mengimplementasikan fungsi `addToCart` dengan penanganan duplikasi produk. Ini adalah fondasi untuk menyimpan data keranjang di frontend.
- **Hari 2**: Kita fokus pada tampilan. Bagaimana cara mengambil data dari Context dan menampilkannya di UI? Kita membuat komponen `CartItem` dan `CartList`, serta menghitung total harga keranjang. Ini memastikan pengguna bisa melihat apa saja yang ada di keranjang mereka.
- **Hari 3**: Kita menambahkan interaktivitas. Pengguna perlu bisa mengubah kuantitas atau menghapus item. Kita mengimplementasikan fungsi `updateQuantity` dan `removeFromCart` dengan prinsip *immutability*, lalu mengintegrasikannya dengan tombol (+) (-), input, dan tombol "Hapus" di `CartItem`.
- **Hari 4**: Puncak dari alur ini adalah proses checkout. Kita membahas bagaimana data keranjang dikirim dari frontend ke backend (menggunakan `axios` atau `fetch`), data apa saja yang perlu disertakan, dan bagaimana menangani respons sukses atau gagal dari server. Kita juga menyentuh simulasi transaksi jika backend belum siap.

Memahami setiap tahapan ini secara menyeluruh akan sangat membantu dalam mengerjakan tugas mingguan.

2. Membahas Simulasi Transaksi Sederhana di Frontend Dalam pengembangan, seringkali frontend dan backend dikerjakan secara paralel. Jika backend untuk proses checkout belum selesai, kita bisa mensimulasikan responsnya di frontend. Ini memungkinkan kita untuk terus mengembangkan UI dan logika frontend tanpa harus menunggu backend.

Bagaimana Mensimulasikan? Seperti yang dibahas di Hari 4, kita bisa menggunakan `setTimeout` untuk menunda eksekusi dan mengembalikan respons palsu. Contoh:

```
const simulateCheckout = (cartData) => {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
```

```
      // Simulasi sukses 80% kemungkinan
      if (Math.random() < 0.8) {
        console.log('Simulasi Checkout Sukses:', cartData);
        resolve({ success: true, message: 'Order berhasil diproses!',
orderId: 'ORD' + Date.now() });
      } else {
        console.error('Simulasi Checkout Gagal:', cartData);
        reject({ success: false, message: 'Terjadi kesalahan saat memproses
order. Silakan coba lagi.' });
      }
    }, 2000); // Simulasi delay 2 detik
  });
};

// Cara menggunakannya di komponen:
// const handleCheckout = async () => {
//   try {
//     const result = await simulateCheckout(cartItems);
//     alert(result.message);
//     // Kosongkan keranjang, redirect, dll.
//   } catch (error) {
//     alert(error.message);
//   }
// };
```

Simulasi ini sangat berguna untuk:

- Menguji alur UI setelah checkout (halaman sukses, halaman error).
- Mengembangkan *loading state* dan *error handling* di frontend.
- Memungkinkan tim frontend bekerja secara independen.

3. Persiapan Tugas Mingguan Bagian terpenting hari ini adalah memahami tugas mingguan. Tugas ini akan menguji pemahaman Anda tentang seluruh materi Cart & Checkout. Bacalah setiap persyaratan dengan seksama.

Tugas Mingguan (Waktu Pengerjaan: Sisa Minggu + Akhir Pekan)

Tema Tugas: Implementasi Fitur Cart & Checkout di Aplikasi E-commerce.

Persyaratan Fungsional:

1. **Manajemen Cart:** Implementasikan state keranjang belanja menggunakan Context API atau state management lain yang Anda pilih. Pastikan state ini bisa diakses secara global.
2. **Tambah ke Keranjang:** Buat fungsi untuk menambahkan produk ke keranjang dari halaman produk (misalnya, dari daftar produk atau halaman detail produk). Pastikan penanganan duplikasi produk (menambah kuantitas jika produk sudah ada).
3. **Tampilkan Cart:** Buat halaman terpisah (misalnya `/cart`) untuk menampilkan semua item di keranjang. Tampilkan detail seperti nama produk, harga, kuantitas, dan subtotal per item.
4. **Update Kuantitas:** Sediakan fungsionalitas untuk mengubah kuantitas item di keranjang (misalnya, dengan tombol (+) dan (-) atau input number). Pastikan kuantitas tidak bisa kurang dari 1.
5. **Hapus Item:** Sediakan tombol untuk menghapus item tertentu dari keranjang.

6. **Total Harga:** Tampilkan total harga semua item di keranjang secara dinamis, yang akan diperbarui setiap kali ada perubahan kuantitas atau penghapusan item.
7. **Proses Checkout:** Implementasikan logika untuk mengirim data keranjang ke backend. Anda bisa menggunakan simulasi seperti yang dibahas di atas, atau jika Anda sudah memiliki backend, integrasikan dengan endpoint checkout nyata.
8. **Kosongkan Cart:** Setelah proses checkout berhasil (baik simulasi maupun nyata), pastikan state keranjang dikosongkan.
9. **Dokumentasi (`README.md`):** Perbarui file `README.md` di repositori proyek Anda. Jelaskan secara singkat fitur cart & checkout yang telah Anda implementasikan, cara menjalankannya, dan teknologi yang digunakan.

Fitur Opsional (Nilai Tambah):

- **Penyimpanan Lokal:** Menyimpan state keranjang di `localStorage` atau `sessionStorage` agar data keranjang tidak hilang saat pengguna me-refresh halaman atau menutup browser.
- **Validasi Stok:** Tambahkan validasi sederhana (misalnya, di frontend) untuk mensimulasikan pengecekan stok atau ketersediaan produk saat menambah ke keranjang atau saat checkout.
- **Notifikasi (Toast):** Tampilkan notifikasi visual (misalnya, menggunakan library seperti `react-toastify` atau implementasi sederhana) saat item berhasil ditambahkan ke keranjang.
- **Halaman Sukses/Riwayat Order:** Implementasi halaman sukses order setelah checkout, atau bahkan halaman riwayat order (ini akan sangat membutuhkan integrasi backend yang lebih dalam).

Output Tugas:

- Kode sumber aplikasi React yang sudah disempurnakan dengan fitur Cart & Checkout.
- File `README.md` yang diperbarui dengan penjelasan fitur.
- Push semua perubahan ke repositori GitHub pribadi Anda.

🗣️ Sesi Presentasi (Waktu disesuaikan)

Siapkan diri Anda untuk sesi presentasi singkat di awal minggu berikutnya. Fokus presentasi adalah:

- **Demo:** Tunjukkan alur lengkap: menambah item ke cart dari halaman produk, melihat/mengubah kuantitas/menghapus item di cart, dan proses checkout (hingga keranjang kosong).
- **Penjelasan Implementasi:** Jelaskan bagaimana Anda mengelola state cart (misalnya, struktur Context API Anda, reducer jika pakai `useReducer`). Jelaskan juga bagaimana data dikirim saat checkout dan bagaimana Anda menangani responsnya.
- **Tantangan & Solusi:** Ceritakan tantangan teknis yang Anda hadapi saat mengerjakan tugas ini dan bagaimana Anda menemukan solusi untuk mengatasinya. Ini menunjukkan kemampuan *problem-solving* Anda.

Tips Belajar

- **Mulai dari yang Fungsional:** Fokus pada persyaratan fungsional terlebih dahulu. Pastikan semua itu bekerja dengan baik sebelum beralih ke fitur opsional.
- **Modularisasi:** Pisahkan logika dan UI ke dalam komponen-komponen yang reusable (misalnya, `CartProvider`, `CartItem`, `CartPage`).
- **Testing:** Lakukan pengujian secara berkala setiap kali Anda menambahkan fungsionalitas baru. Jangan menunggu sampai semua selesai baru diuji.

- **Manfaatkan Referensi:** Gunakan materi yang sudah diberikan dan referensi teknis yang relevan. Jangan ragu mencari solusi di dokumentasi resmi atau sumber terpercaya lainnya.
- **Istirahat Cukup:** Jangan memaksakan diri. Jika buntu, istirahat sejenak dan kembali dengan pikiran yang lebih segar.

Referensi Teknis

- React Context API Documentation
- Axios GitHub Repository
- MDN Web Docs - Fetch API
- Working with localStorage - MDN Web Docs
- React Toastify (contoh library notifikasi)