📆 Hari 5 (Jum'at) - Review, State Management Lanjutan (Pengantar Redux/Zustand), & Tugas Mingguan

🎯 Tujuan Pembelajaran Hari Ini

- Mereview kembali seluruh alur autentikasi yang telah dibangun dari Hari 1 hingga Hari 4.
- Memahami kelebihan dan kekurangan penggunaan Context API untuk state global, terutama pada aplikasi yang lebih besar.
- Mendapatkan pengantar tentang library state management lanjutan seperti Redux Toolkit dan Zustand.
- Memahami kapan harus menggunakan Context API vs. library state management eksternal.
- Mempersiapkan diri untuk mengerjakan tugas mingguan implementasi alur autentikasi lengkap.

📚 Materi Inti (2 Jam)

1. Review Auth Flow

Mari kita rekap kembali alur autentikasi yang sudah kita bangun selama seminggu ini:

1. **Login/Register:** User mengirimkan kredensial ke backend. Backend memvalidasi dan mengembalikan token (misalnya JWT).
2. **Penyimpanan Token:** Token diterima di frontend dan disimpan di `localStorage` (Hari 1).
3. **Initial Load:** Saat aplikasi pertama kali dimuat atau di-refresh, `AuthProvider` membaca token dari `localStorage` menggunakan `useEffect` dan memperbarui state `isAuthenticated` di Context (Hari 4).
4. **Conditional Routing (Login/Register):** Di halaman Login/Register, kita menggunakan `useEffect` dan `useNavigate` untuk mengecek status `isAuthenticated` dari Context. Jika user sudah login, mereka langsung diarahkan ke halaman lain (misalnya `/dashboard`) (Hari 2, disempurnakan di Hari 4).
5. **Protected Routes:** Komponen `ProtectedRoute` membaca status `isAuthenticated` dari Context. Jika `true`, ia me-render child route (`<Outlet />`). Jika `false`, ia me-render `<Navigate />` ke halaman login (Hari 3, disempurnakan di Hari 4).
6. **UI Updates:** Komponen UI lain (seperti Header) menggunakan hook `useAuth()` untuk mendapatkan status `isAuthenticated` dan data `user`, lalu menampilkan elemen yang relevan (tombol Logout vs Login/Register) (Hari 4).
7. **Logout:** Saat user mengklik Logout, fungsi `logout()` dari Context dipanggil. Ini menghapus token dari `localStorage`, memperbarui state `isAuthenticated` menjadi `false`, dan memicu re-render komponen yang menggunakan Context (Hari 4).

Alur ini mencakup langkah-langkah fundamental dalam mengelola autentikasi di aplikasi SPA React.

2. Kelebihan & Kekurangan Context API untuk State Global

Context API sangat berguna dan merupakan solusi bawaan React untuk state global. Namun, ia juga memiliki keterbatasan:

Kelebihan Context API:

- **Bawaan React:** Tidak perlu menginstal library tambahan.
- **Sederhana:** Cukup mudah dipelajari dan diimplementasikan untuk kebutuhan state global yang tidak terlalu kompleks.
- **Cocok untuk Data Statis/Jarang Berubah:** Ideal untuk data seperti tema, bahasa, atau informasi user yang tidak sering diubah oleh banyak aksi.

Kekurangan Context API:

- **Performa:** Jika nilai Context sering berubah, semua komponen yang mengonsumsi Context tersebut (bahkan jika mereka hanya menggunakan sebagian kecil dari nilai Context) akan di-render ulang. Ini bisa menjadi masalah performa pada aplikasi besar dengan banyak konsumen Context dan update yang sering.
- **Kompleksitas Logic:** Mengelola state yang kompleks dengan banyak aksi dan interaksi antar bagian state bisa menjadi rumit hanya dengan `useState` dan `useReducer` di dalam Provider.
- **Debugging:** Melacak dari mana sebuah perubahan state berasal bisa lebih sulit dibandingkan dengan library state management yang memiliki devtools canggih.
- **Skalabilitas:** Untuk aplikasi yang sangat besar dengan state yang kompleks dan banyak aksi, Context API saja mungkin tidak cukup terstruktur dan sulit dikelola.

3. Pengantar State Management Libraries (Lanjutan): Redux (Toolkit), Zustand

Untuk aplikasi yang lebih besar atau memiliki state global yang kompleks, library state management eksternal seringkali menjadi pilihan yang lebih baik. Mereka menyediakan struktur, pola, dan fitur tambahan untuk mengelola state dengan lebih terorganisir dan efisien.

- **Redux (dengan Redux Toolkit):**
    - Salah satu library state management paling populer di ekosistem React.
    - Menggunakan konsep *single source of truth* (satu store global), *actions* (objek yang mendeskripsikan apa yang terjadi), dan *reducers* (fungsi murni yang menentukan bagaimana state berubah berdasarkan action).
    - **Redux Toolkit** adalah cara yang direkomendasikan untuk menggunakan Redux saat ini. Ia menyederhanakan setup, mengurangi boilerplate, dan menyediakan utilitas bawaan (seperti Immer untuk immutable updates, RTK Query untuk data fetching).
    - Cocok untuk aplikasi besar dengan state yang kompleks dan banyak interaksi.
    - Memiliki ekosistem yang matang dan devtools yang sangat membantu untuk debugging.
- **Zustand:**
    - Library state management yang lebih modern dan minimalis.
    - Menggunakan hook (`useStore`) untuk mengakses state, membuatnya terasa lebih 'React-native'.
    - Setup sangat sederhana dan boilerplate minimal.
    - Cocok untuk aplikasi berukuran sedang hingga besar yang mencari solusi state management yang lebih ringan dan mudah digunakan dibandingkan Redux.
    - Performa yang baik karena hanya me-render ulang komponen yang benar-benar menggunakan bagian state yang berubah.

4. Memilih Library

Kapan menggunakan Context API, Redux Toolkit, atau Zustand?

- **Context API:** Untuk state global yang sederhana, jarang berubah, dan hanya dibutuhkan oleh komponen-komponen yang berada dalam sub-tree yang sama. Cocok untuk tema, bahasa, atau data user sederhana di aplikasi kecil/menengah.
- **Redux Toolkit:** Untuk aplikasi besar dengan state yang sangat kompleks, banyak aksi, dan kebutuhan debugging yang canggih. Jika Anda sudah familiar dengan konsep Redux atau membutuhkan ekosistem yang matang.
- **Zustand:** Untuk aplikasi berukuran sedang hingga besar yang membutuhkan solusi state global yang lebih terstruktur dari Context API, tetapi lebih ringan dan mudah digunakan daripada Redux Toolkit. Jika Anda mencari performa yang baik dan developer experience yang menyenangkan.

Untuk tugas mingguan kali ini, kita akan tetap menggunakan **Context API** karena sudah cukup memadai untuk skala aplikasi e-commerce sederhana yang kita kerjakan dan agar fokus pada pemahaman alur autentikasi itu sendiri.

5. Persiapan Tugas Mingguan

Tugas mingguan adalah kesempatan untuk mengkonsolidasikan semua yang telah Anda pelajari tentang autentikasi dan routing terproteksi dengan mengimplementasikannya secara penuh di proyek e-commerce Anda.

Pastikan Anda memahami semua persyaratan fungsional dan fitur opsional. Gunakan materi harian dari Minggu 8 ini sebagai panduan langkah demi langkah.

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

Tema Tugas: Menyempurnakan Alur Autentikasi dan Routing di Aplikasi E-commerce.

Persyaratan Fungsional:

1. **Implementasi Auth Flow Lengkap:** Pastikan alur login dan logout berfungsi penuh, termasuk komunikasi dengan backend (simulasi atau nyata jika ada).
2. **Conditional Routing:** User yang sudah login tidak bisa mengakses halaman Login/Register dan otomatis diarahkan ke halaman lain (misalnya Home atau Dashboard).
3. **Protected Routes:** Halaman-halaman tertentu (minimal `/dashboard`) hanya bisa diakses oleh user yang sudah login. User yang belum login akan diarahkan ke halaman Login.
4. **State Autentikasi Global (Context API):** Gunakan Context API untuk menyimpan dan mengelola status autentikasi (`isAuthenticated`, `user`) secara global.
5. **Integrasi UI dengan Auth State:** Komponen UI (misalnya Header/Navbar) harus menampilkan informasi yang relevan berdasarkan status autentikasi (misalnya, tampilkan tombol Login/Register saat belum login, tampilkan nama user dan tombol Logout saat sudah login).
6. **Refactor Kode:** Pastikan kode Anda bersih, mudah dibaca, dan mengikuti praktik terbaik yang telah dipelajari.
7. **Dokumentasi (`README.md`):** Perbarui file `README.md` di root proyek Anda. Jelaskan cara menjalankan proyek, endpoint API yang digunakan (jika ada simulasi), dan fitur autentikasi yang telah

diimplementasikan.

Fitur Opsional (Nilai Tambah):

- **Interceptor `axios` untuk header `Authorization`:** Tambahkan interceptor pada instance `axios` Anda untuk secara otomatis menyertakan header `Authorization: Bearer <token>` pada setiap request ke backend jika user sudah login.
- **Menangani refresh token:** Jika backend Anda mendukung refresh token, implementasikan logika untuk menggunakan refresh token saat access token expired.
- **Menggunakan Zustand atau Redux Toolkit:** Ganti Context API dengan salah satu library ini untuk manajemen state autentikasi.
- **Menampilkan pesan error login/register yang informatif:** Tangani respons error dari backend saat login/register dan tampilkan pesan yang jelas kepada user di UI.

Output Tugas:

- Kode sumber aplikasi React yang sudah disempurnakan.
- File `README.md` yang diperbarui.
- 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 login, akses halaman terproteksi, dan logout di aplikasi Anda.
- **Penjelasan Implementasi:** Jelaskan bagaimana Anda mengimplementasikan `AuthContext` (atau state management lain jika menggunakan fitur opsional) dan komponen `ProtectedRoute`.
- **Adaptasi UI:** Jelaskan bagaimana komponen UI Anda beradaptasi secara dinamis berdasarkan status autentikasi.
- **Tantangan & Solusi:** Ceritakan tantangan yang Anda hadapi saat mengerjakan tugas dan bagaimana Anda menyelesaikannya.

💡 Tips Belajar Tambahan

- **Pecah Tugas:** Jangan mencoba menyelesaikan semuanya sekaligus. Pecah tugas mingguan menjadi langkah-langkah kecil (misalnya, selesaikan Protected Routes dulu, lalu integrasikan Context, lalu perbaiki UI, dst).
- **Gunakan Git:** Lakukan commit secara berkala dengan pesan yang jelas. Ini membantu Anda melacak perubahan dan kembali ke versi sebelumnya jika ada kesalahan.
- **Baca Dokumentasi:** Jangan ragu merujuk kembali ke dokumentasi React Router, Context API, atau library lain yang Anda gunakan.
- **Simulasi Backend:** Jika Anda tidak memiliki backend nyata, simulasikan respons API (sukses login dengan token, error login, dll.) langsung di frontend untuk tujuan pengembangan.

🔗 Referensi Tambahan

- React Documentation - Context
- React Router Documentation - Auth Examples

- Redux Toolkit Official Website
- Zustand Official Website
- MDN Web Docs - Using the Web Storage API

Selamat! Anda telah menyelesaikan materi pembelajaran untuk Minggu 8 tentang Auth Flow & Routing Terproteksi. Topik ini sangat fundamental dalam pengembangan aplikasi web modern. Sekarang saatnya mengaplikasikan semua yang telah Anda pelajari dalam tugas mingguan. Jika ada pertanyaan atau kesulitan, jangan ragu untuk bertanya!