UJIAN AKHIR SEMESTER - CLIENT SIDE PROGRAMMING

Waktu Pengerjaan: 120 Menit

Sifat Ujian: Open Catatan/Referensi di Internet (Next.js, Tailwind)

I. Deskripsi Skenario

Anda diminta untuk membangun sebuah **Aplikasi Portal Karyawan Sederhana** (*Employee Portal*). Aplikasi ini berfungsi sebagai gerbang masuk bagi karyawan untuk melihat status profil mereka.

Aplikasi ini harus aman, cepat, dan menggunakan teknologi terbaru dari ekosistem React Server Components (RSC).

II. Spesifikasi Teknis (Wajib)

1. **Framework:** Next.js versi 15 atau 16 (terbaru).
2. **Architecture:** App Router (wajib, tidak boleh Pages Router).
3. **Backend & Auth:** Supabase (menggunakan package `@supabase/ssr`).
4. **Language:** TypeScript.
5. **Styling:** Tailwind CSS.

III. Instruksi Pengerjaan (Step-by-Step)

Bagian 1: Konfigurasi & Setup (Bobot: 15%)

1. Buat project Next.js baru.
2. Buat project baru di dashboard Supabase.
3. Konfigurasikan Environment Variables (`.env.local`) yang berisi `NEXT_PUBLIC_SUPABASE_URL` dan `NEXT_PUBLIC_SUPABASE_ANON_KEY`.
4. Setup Supabase Client untuk sisi server (Server Actions/Component) dan middleware menggunakan library `@supabase/ssr`.

Bagian 2: Halaman Registration (Bobot: 25%)

Buatlah halaman di rute `/register` dengan ketentuan:

1. Memiliki form dengan input **Email** dan **Password**.
2. Proses register **HARUS** menggunakan **Server Actions** (tandai dengan `'use server'`). Jangan melakukan logika login autentikasi di Client Component (`useEffect` atau `onSubmit` client-side handler biasa).
3. Jika register gagal, tampilkan pesan error yang sesuai di UI.
4. Jika register berhasil, pengguna otomatis diarahkan (redirect) ke halaman `/login`.

Bagian 3: Halaman Login (Bobot: 25%)

Buatlah halaman di rute `/login` dengan ketentuan:

5. Memiliki form dengan input **Email** dan **Password**.

6. Proses login **HARUS** menggunakan **Server Actions** (tandai dengan 'use server'). Jangan melakukan logika login autentikasi di Client Component (useEffect atau onSubmit client-side handler biasa).
7. Jika login gagal, tampilkan pesan error yang sesuai di UI.
8. Jika login berhasil, pengguna otomatis diarahkan (redirect) ke halaman /dashboard.

Bagian 4: Proteksi Rute (Middleware) (Bobot: 15%)

Implementasikan middleware.ts pada root project untuk menangani logika sesi:

1. **Skenario A:** Jika pengguna **belum login** mencoba mengakses /dashboard, mereka harus dipaksa redirect kembali ke /login.
2. **Skenario B:** Jika pengguna **sudah login** mencoba mengakses /login, mereka harus dipaksa redirect masuk ke /dashboard.
3. Pastikan sesi Supabase diperbarui (*refresh session*) di dalam middleware ini.

Bagian 5: Halaman Dashboard (Bobot: 10%)

Buatlah halaman di rute /dashboard yang merupakan **Server Component**. Halaman ini harus menampilkan:

1. **Informasi User:** Tampilkan Email pengguna yang sedang login (ambil data session menggunakan fungsi getUser dari Supabase di sisi server).
2. **Data Dinamis:**
    - Buat tabel dummy di database Supabase Anda bernama announcements (kolom: id, title, content, created_at).
    - Isi tabel tersebut dengan minimal 3 baris data dummy.
    - Fetch data tersebut langsung di Server Component /dashboard dan render ke layar dalam bentuk Card atau List.

Bagian 6: Fitur Logout (Bobot: 10%)

1. Tambahkan tombol "Logout" di dalam halaman Dashboard.
2. Tombol ini harus memicu **Server Action** untuk menghapus sesi pengguna di Supabase.
3. Setelah logout berhasil, pengguna harus dilempar kembali ke halaman /login.

IV. Kriteria Penilaian Kode (Rubrik)

Komponen	Poin Kurang (0-50)	Poin Cukup (51-80)	Poin Sempurna (81-100)
Next.js 15/16 Config	Masih ada sisa setup Pages router atau salah config Turbopack.	Config benar, namun struktur folder berantakan.	Menggunakan App Router murni, async params (jika ada), dan struktur folder rapi.

Supabase Auth	Autentikasi berjalan di Client Side sepenuhnya (tidak aman).	Autentikasi Server Side, tapi penanganan error minim.	Autentikasi via Server Actions, Error Handling rapi, Type Safety terjaga.
Middleware	Middleware tidak jalan atau user bisa tembus tanpa login.	Middleware jalan tapi kadang *infinite redirect*.	Middleware robust, refresh token berjalan, redirect mulus.
Data Fetching	Menggunakan `useEffect` untuk fetch data dashboard.	Fetch di Server Component tapi tidak di-handle loading state (Suspense).	Fetching native di Server Component (`await`), menggunakan Suspense untuk loading UI.

V. Format Pengumpulan

Kumpulkan jawaban Anda dalam bentuk file teks/PDF yang berisi:

1. **Link Repository GitHub** (Pastikan *public* atau invite dosen sebagai *collaborator*).
2. **Link Demo Aplikasi** (Deploy ke Vercel).
3. **Screenshot** bukti tabel Supabase yang sudah dibuat.

Lampiran: Snippet Bantuan (Hint)

Hint 1: Struktur Server Action untuk Login (`actions/auth.ts`)

TypeScript

None

```
'use server'

import { createClient } from '@/utils/supabase/server'
import { redirect } from 'next/navigation'

export async function login(formData: FormData) {
  const supabase = await createClient()

  // Ambil data dari formData
  // Validasi input
```

```
  // Panggil supabase.auth.signInWithPassword

  // Jika error, return error
  // Jika sukses:
  redirect('/dashboard')
}
```

Hint 2: Fetching Data di Dashboard (`app/dashboard/page.tsx`)

TypeScript

None

```
import { createClient } from '@/utils/supabase/server'
import { redirect } from 'next/navigation'

export default async function DashboardPage() {
  const supabase = await createClient()

  const { data: { user }, error } = await supabase.auth.getUser()
  if (error || !user) {
    redirect('/login')
  }

  // Fetch data announcements
  const { data: notes } = await supabase.from('announcements').select()

  return (
    <div className="p-8">
      <h1>Welcome, {user.email}</h1>
      {/* Render notes here */}
    </div>
  )
}
```