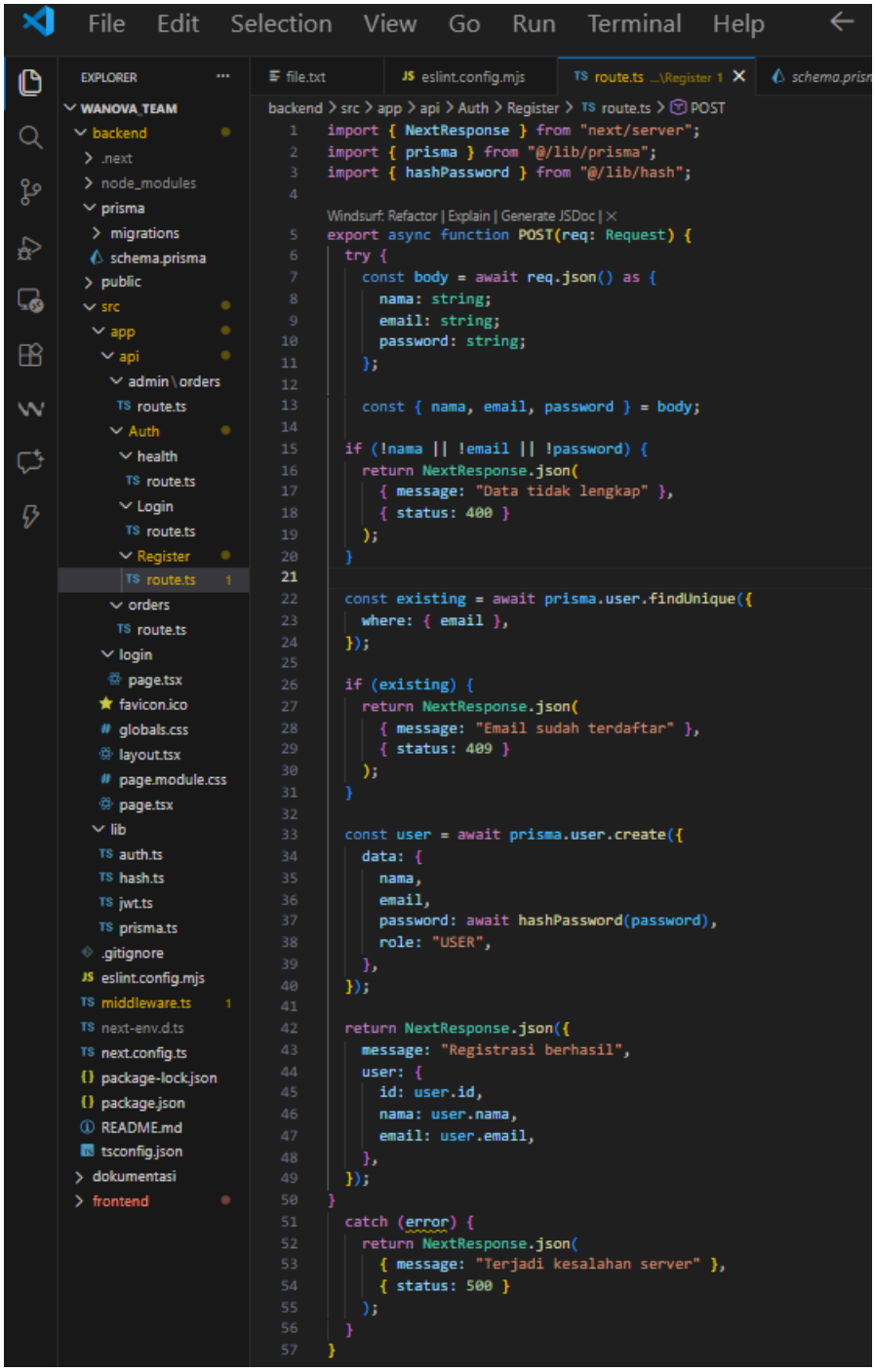
DOKUMENTASI PROJEK PWBL

NAMA: WANDI FILEMON HOTMARTUA SIANTURI

NPM: 23313009

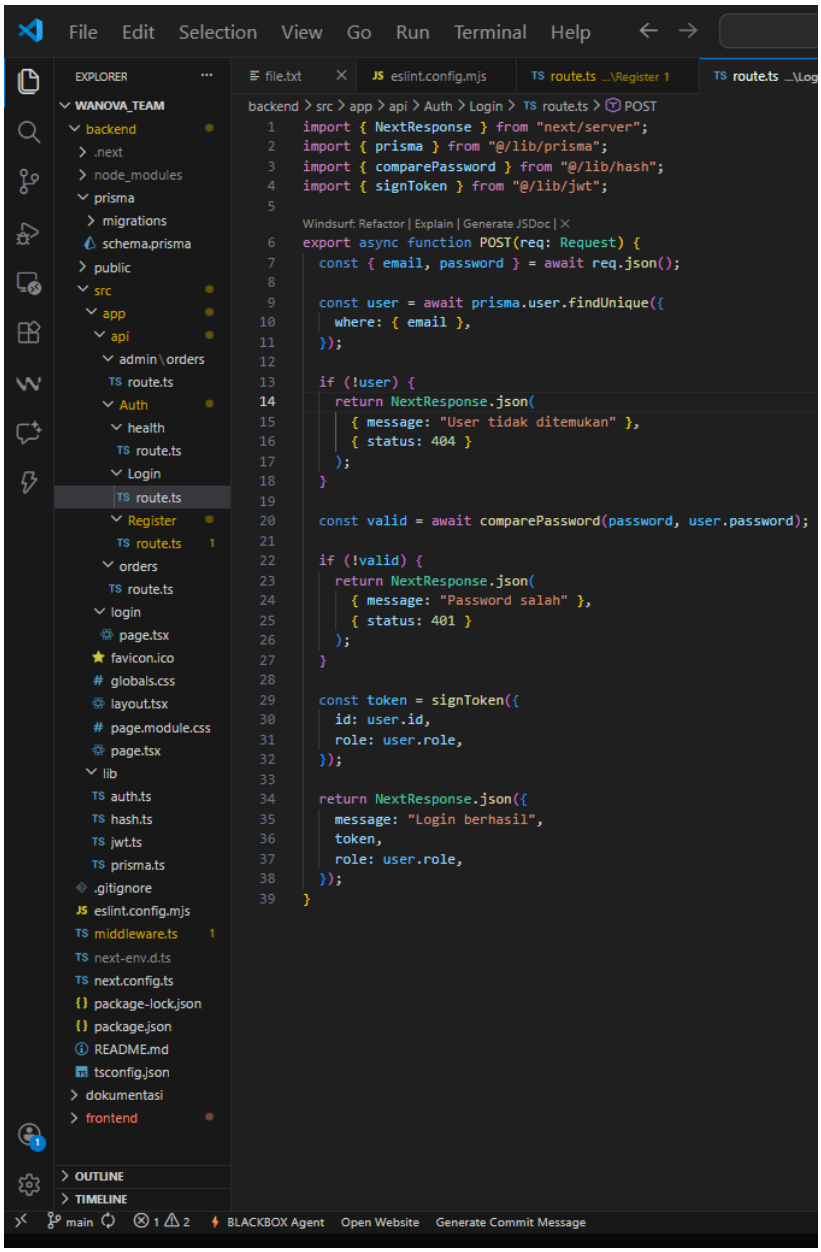
1. Backend/Register/route.ts



Kode ini adalah API endpoint di Next.js untuk proses registrasi pengguna. Saat menerima request POST, server membaca data nama, email, dan password dari body, lalu memeriksa apakah semuanya sudah diisi. Jika ada yang kosong, akan dikembalikan error. Setelah itu sistem mengecek ke database dengan Prisma apakah email sudah terdaftar; jika sudah, registrasi ditolak. Jika belum, password akan di-hash (dienkripsi satu arah) menggunakan fungsi hashPassword, kemudian data pengguna baru disimpan ke

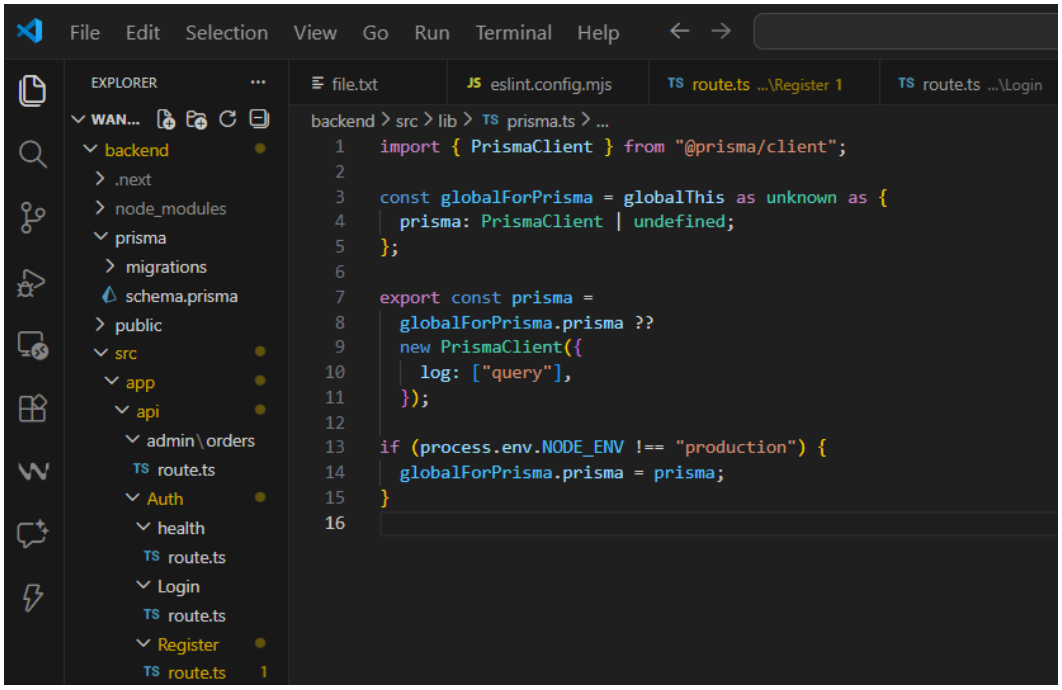
database dengan role USER. Jika berhasil, server mengirimkan respons sukses berisi pesan dan data dasar pengguna (id, nama, email), dan jika terjadi kesalahan, server akan mengembalikan pesan error.

2. Backend/Api/auth/login



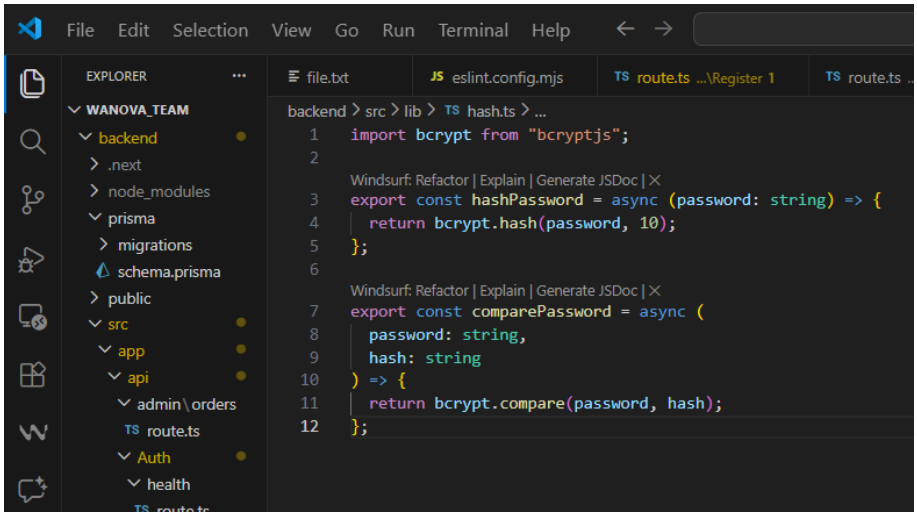
Kode ini adalah API endpoint untuk proses login. Server menerima email dan password dari request, lalu mencari data user di database menggunakan Prisma. Jika email tidak ditemukan, server mengirim error. Jika ditemukan, password yang dimasukkan akan dibandingkan dengan password terenkripsi di database menggunakan comparePassword. Jika tidak cocok, login ditolak. Jika cocok, server membuat token JWT berisi ID dan role user menggunakan signToken, lalu mengirimkan respons sukses berupa pesan, token, dan role pengguna.

3. Backend/Prisma/schema.prisma



Kode ini berfungsi untuk membuat dan mengelola satu instance Prisma Client agar tidak dibuat berulang kali, terutama saat mode development di Next.js yang sering melakukan hot-reload. Variabel globalForPrisma digunakan untuk menyimpan objek Prisma di scope global, sehingga jika sudah ada, akan dipakai kembali. Jika belum ada, maka dibuat instance baru PrismaClient dengan opsi log query. Pada mode non-production, instance tersebut disimpan ke globalThis supaya koneksi database tetap stabil dan tidak membuka koneksi baru setiap kali file dimuat ulang.

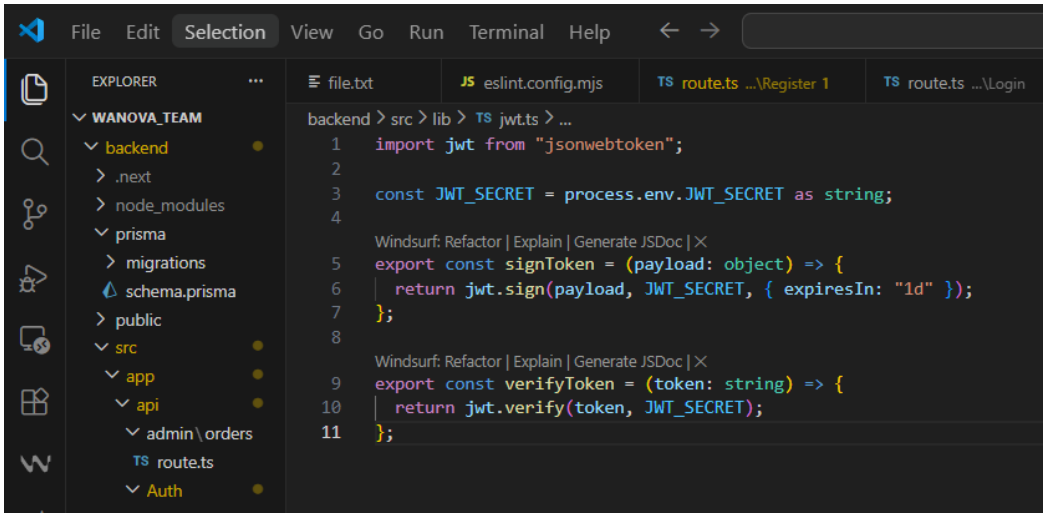
4. backned/lib/hash.ts



Kode ini digunakan untuk mengamankan password dengan algoritma bcrypt. Fungsi hashPassword mengenkripsi password asli menjadi bentuk hash dengan tingkat keamanan (salt round) 10 sebelum disimpan ke database, sehingga password tidak tersimpan dalam bentuk teks biasa. Sedangkan fungsi

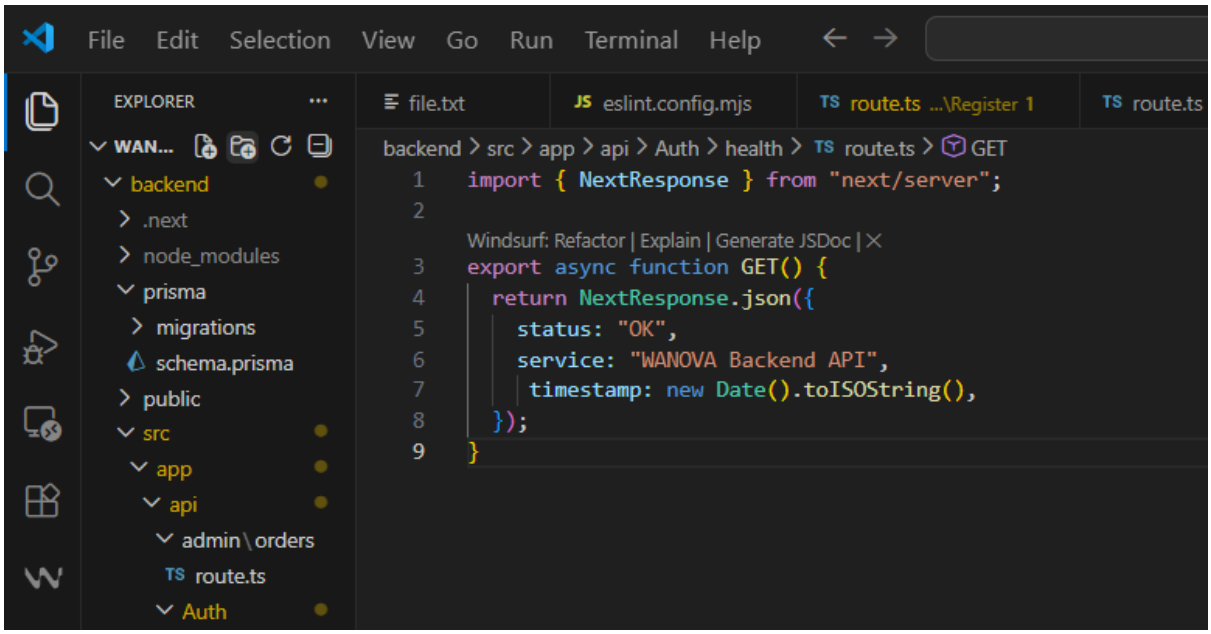
comparePassword digunakan saat login untuk membandingkan password yang dimasukkan user dengan hash password yang ada di database, dan akan mengembalikan nilai benar atau salah.

5. backend/lib/Jwt Ts



Kode ini digunakan untuk membuat dan memverifikasi JSON Web Token (JWT) sebagai sistem autentikasi. Fungsi signToken menghasilkan token berisi data user (payload) yang ditandatangani dengan kunci rahasia JWT_SECRET dan berlaku selama 1 hari, sehingga bisa dipakai sebagai bukti login. Sedangkan fungsi verifyToken digunakan untuk mengecek keaslian dan validitas token tersebut saat user mengakses halaman atau API yang dilindungi, memastikan token tidak palsu dan belum kedaluwarsa.

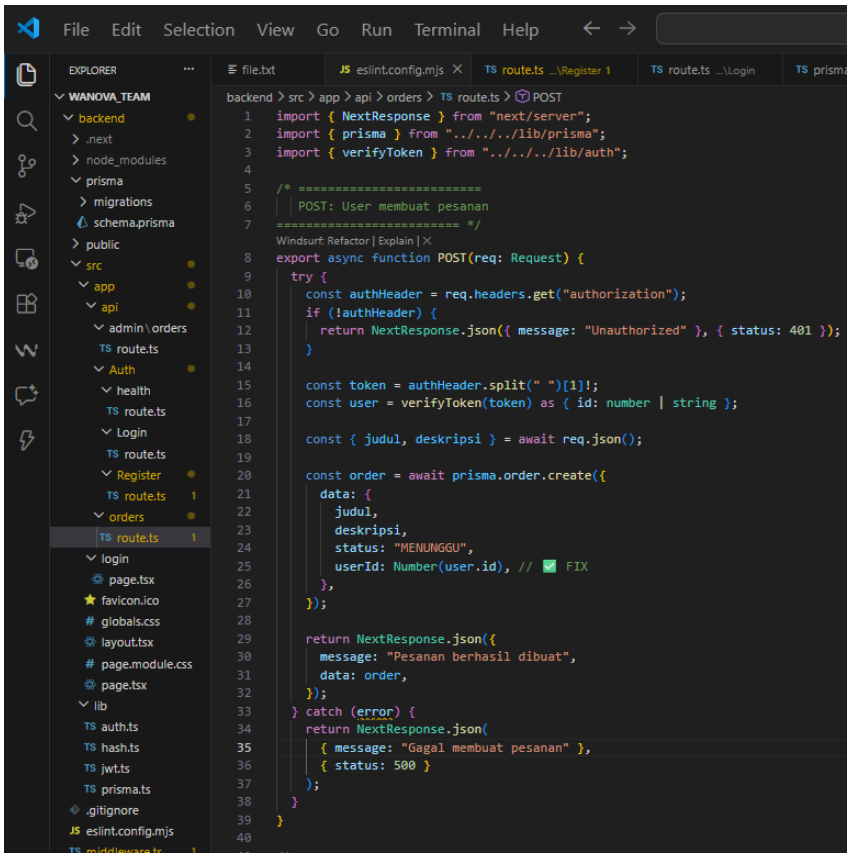
6. Backend/api/Auth/Health/route.ts



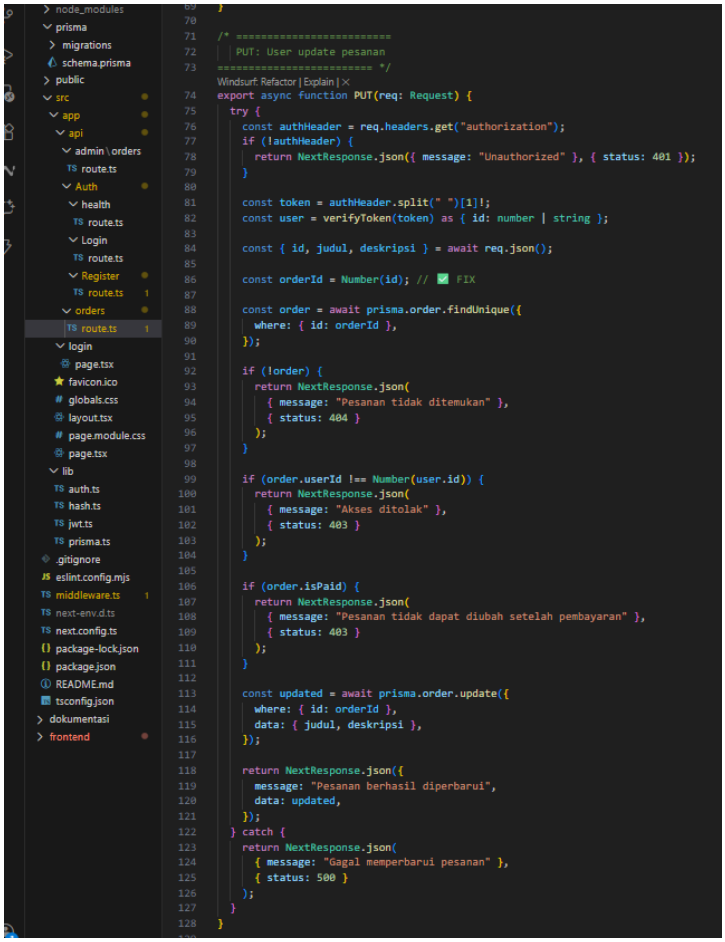
Kode ini adalah endpoint API sederhana untuk mengecek status server (health check). Saat diakses dengan metode GET, server akan mengembalikan respons JSON yang berisi status “OK”, nama layanan

“WANOVA Backend API”, dan waktu saat ini dalam format ISO, sehingga bisa digunakan untuk memastikan bahwa backend sedang berjalan dengan normal.

7. Frontend/Api/Orders/route.ts

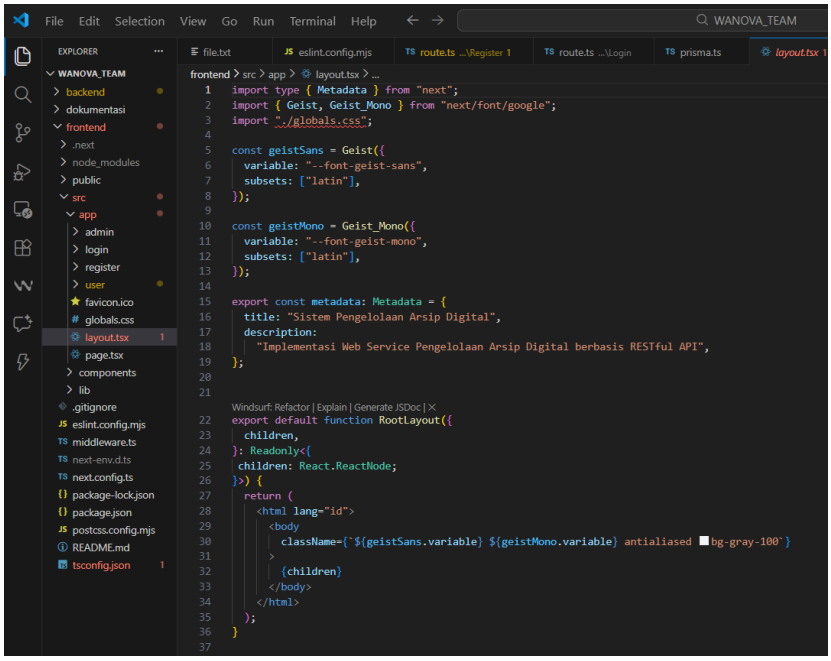



```
/* ========================
   GET: User melihat pesanan sendiri
======================== */
Windsurf: Refactor | Explain |
export async function GET(req: Request) {
  try {
    const authHeader = req.headers.get("authorization");
    if (!authHeader) {
      return NextResponse.json({ message: "Unauthorized" }, { status: 401 });
    }

    const token = authHeader.split(" ")[1]!;
    const user = verifyToken(token) as { id: number | string };

    const orders = await prisma.order.findMany({
      where: { userId: Number(user.id) }, // ✅ FIX
      orderBy: { createdAt: "desc" },
    });

    return NextResponse.json({
      message: "Data pesanan berhasil diambil",
      data: orders,
    });
  } catch {
    return NextResponse.json(
      { message: "Gagal mengambil data pesanan" },
      { status: 500 }
    );
  }
}
```



Kode ini adalah API untuk mengelola pesanan (order) yang hanya bisa diakses oleh user yang sudah login menggunakan token JWT. Pada setiap request, server mengambil token dari header Authorization, memverifikasinya untuk mendapatkan ID user, lalu: metode **POST** digunakan untuk membuat pesanan baru dengan status “MENUNGGU”, **GET** untuk menampilkan daftar pesanan milik user tersebut saja, dan **PUT** untuk mengubah judul atau deskripsi pesanan selama pesanan itu milik user yang sama dan belum dibayar. Semua proses menggunakan Prisma untuk berinteraksi dengan database, serta dilengkapi pengecekan hak akses dan penanganan error.

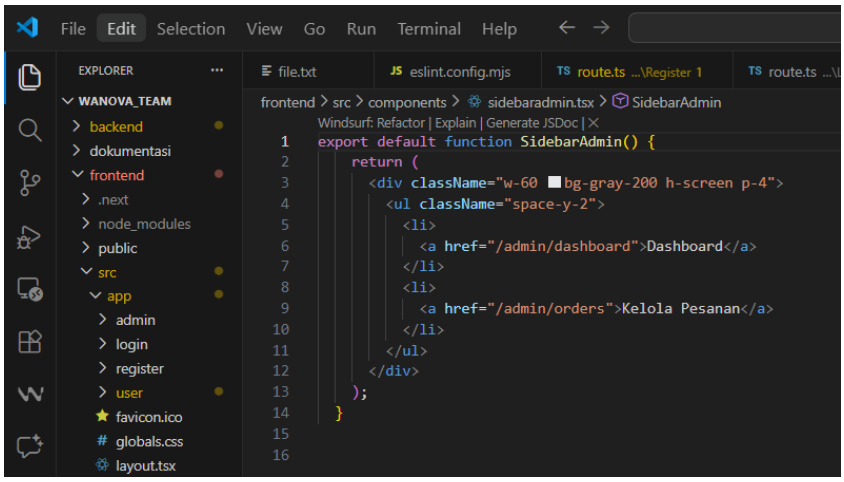
8. Frontend/layout.tsx



```tsx
import type { Metadata } from "next";
import { Geist, Geist_Mono } from "next/font/google";
import "../globals.css";

const geistSans = Geist({
  variable: "--font-geist-sans",
  subsets: ["latin"],
});

const geistMono = Geist_Mono({
  variable: "--font-geist-mono",
  subsets: ["latin"],
});

export const metadata: Metadata = {
  title: "Sistem Pengelolaan Arsip Digital",
  description:
    "Implementasi Web Service Pengelolaan Arsip Digital berbasis RESTful API",
};


Windsurf: Refactor | Explain | Generate JSDoc |
export default function RootLayout({
  children,
}: Readonly<{
  children: React.ReactNode;
}>) {
  return (
    <html lang="id">
      <body
        className={`${geistSans.variable} ${geistMono.variable} antialiased bg-gray-100`}
      >
        {children}
      </body>
    </html>
  );
}
```

Kode ini adalah layout utama (RootLayout) pada aplikasi Next.js yang mengatur font, metadata, dan tampilan dasar seluruh halaman. Di sini digunakan font Google Geist dan Geist Mono, lalu metadata seperti judul dan deskripsi website didefinisikan untuk keperluan SEO. Komponen RootLayout membungkus semua halaman dengan tag HTML berbahasa Indonesia, menerapkan font, efek antialiasing, serta latar belakang abu-abu, sehingga semua halaman memiliki gaya dan struktur yang konsisten.

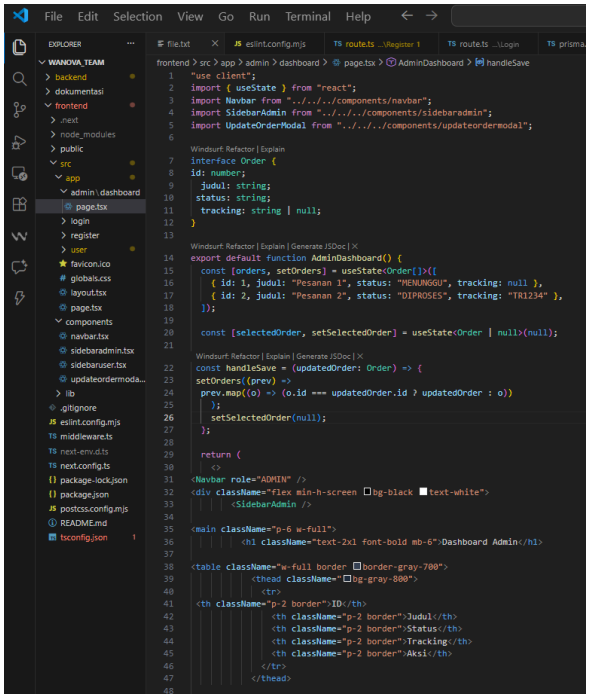
9. frontend/components/SidebarAdmin



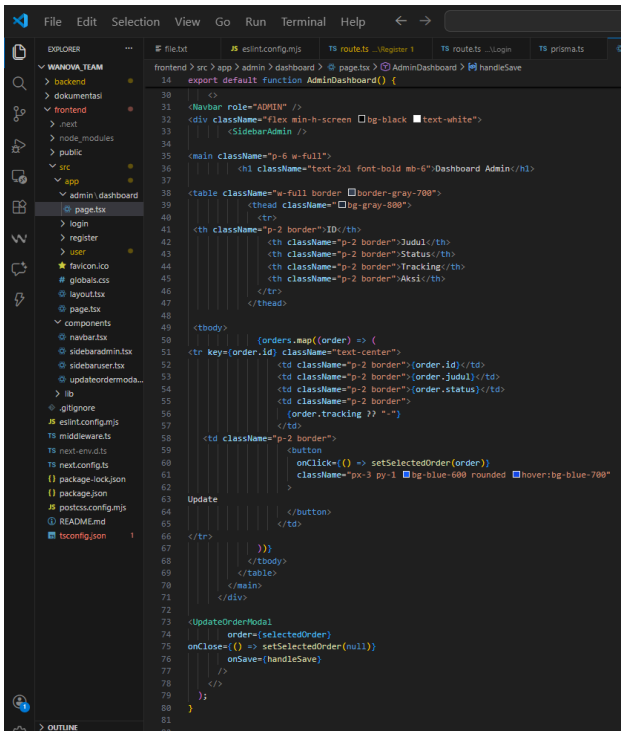
```tsx
Windsurf: Refactor | Explain | Generate JSDoc |
export default function SidebarAdmin() {
    return (
      <div className="w-60 bg-gray-200 h-screen p-4">
        <ul className="space-y-2">
          <li>
            <a href="/admin/dashboard">Dashboard</a>
          </li>
          <li>
            <a href="/admin/orders">Kelola Pesanan</a>
          </li>
        </ul>
      </div>
    );
  }
```

Kode ini adalah komponen React sederhana untuk menampilkan sidebar khusus admin. Sidebar memiliki lebar tetap, tinggi penuh layar, dan latar belakang abu-abu, lalu berisi daftar menu berupa link ke halaman “Dashboard” dan “Kelola Pesanan”. Komponen ini berfungsi sebagai navigasi agar admin dapat berpindah ke halaman utama admin dan halaman pengelolaan pesanan dengan mudah.

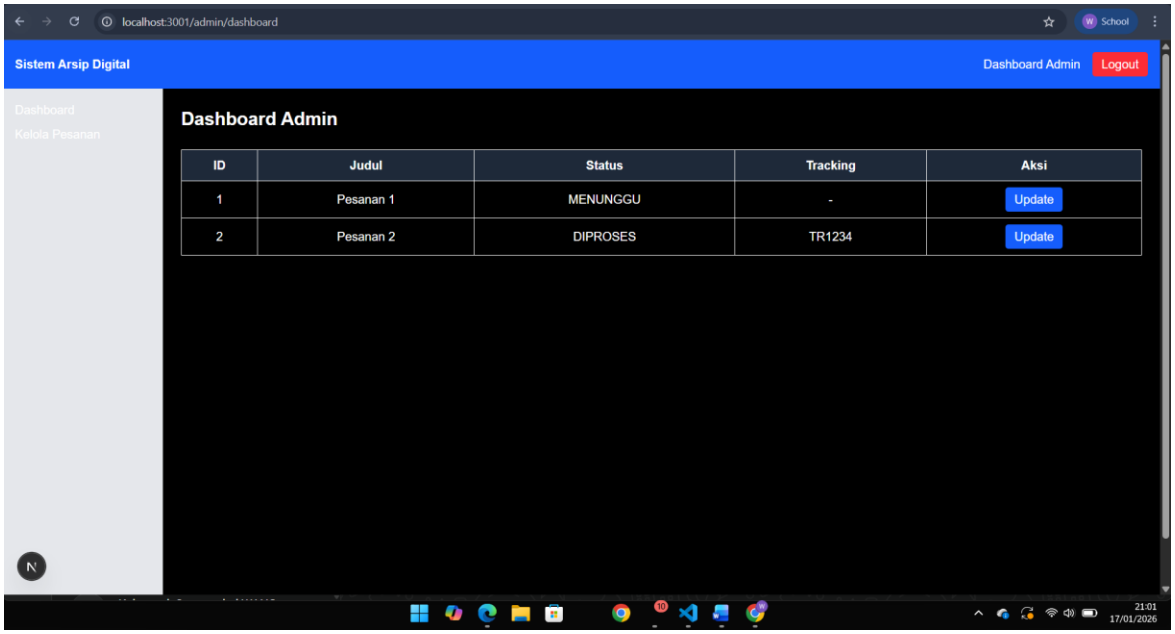
10. Frontend/Admin/dashboard/page.tsx



```
"use client";
import { useState } from "react";
import Navbar from "../../../components/navbar";
import SidebarAdmin from "../../../components/sidebaradmin";
import UpdateOrderModal from "../../../components/updateordermodal";

interface Order {
id: number;
  judul: string;
 status: string;
  tracking: string | null;
}

export default function AdminDashboard() {
  const [orders, setOrders] = useState<Order[]>([
    { id: 1, judul: "Pesanan 1", status: "MENUNGGU", tracking: null },
    { id: 2, judul: "Pesanan 2", status: "DIPROSES", tracking: "TR1234" },
  ]);

  const [selectedOrder, setSelectedOrder] = useState<Order | null>(null);

  const handleSave = (updatedOrder: Order) => {
 setOrders((prev) =>
  prev.map((o) => (o.id === updatedOrder.id ? updatedOrder : o))
    );
    setSelectedOrder(null);
  };

  return (
    <>
 <Navbar role="ADMIN" />
 <div className="flex min-h-screen bg-black text-white">
        <SidebarAdmin />

 <main className="p-6 w-full">
          <h1 className="text-2xl font-bold mb-6">Dashboard Admin</h1>

 <table className="w-full border border-gray-700">
            <thead className="bg-gray-800">
              <tr>
 <th className="p-2 border">ID</th>
                <th className="p-2 border">Judul</th>
                <th className="p-2 border">Status</th>
                <th className="p-2 border">Tracking</th>
                <th className="p-2 border">Aksi</th>
              </tr>
            </thead>
```

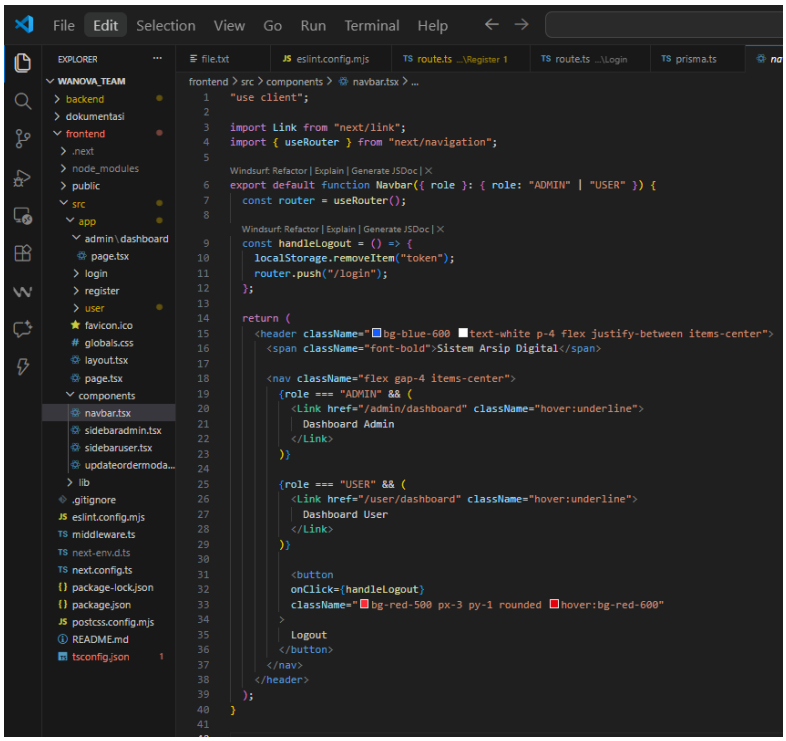


```
export default function AdminDashboard() {
    <>
<Navbar role="ADMIN" />
<div className="flex min-h-screen bg-black text-white">
        <SidebarAdmin />

<main className="p-6 w-full">
          <h1 className="text-2xl font-bold mb-6">Dashboard Admin</h1>

<table className="w-full border border-gray-700">
            <thead className="bg-gray-800">
              <tr>
 <th className="p-2 border">ID</th>
                <th className="p-2 border">Judul</th>
                <th className="p-2 border">Status</th>
                <th className="p-2 border">Tracking</th>
                <th className="p-2 border">Aksi</th>
              </tr>
            </thead>

 <tbody>
            {orders.map((order) => (
<tr key={order.id} className="text-center">
                <td className="p-2 border">{order.id}</td>
                <td className="p-2 border">{order.judul}</td>
                <td className="p-2 border">{order.status}</td>
                <td className="p-2 border">
                  {order.tracking ?? "-"}
                </td>
 <td className="p-2 border">
                  <button
                    onClick={() => setSelectedOrder(order)}
                    className="px-3 py-1 bg-blue-600 rounded hover:bg-blue-700"
                  >
Update
                  </button>
                </td>
</tr>
            ))}
          </tbody>
        </table>
      </main>
    </div>

<UpdateOrderModal
      order={selectedOrder}
onClose={() => setSelectedOrder(null)}
      onSave={handleSave}
    />
  </>
  );
}
```



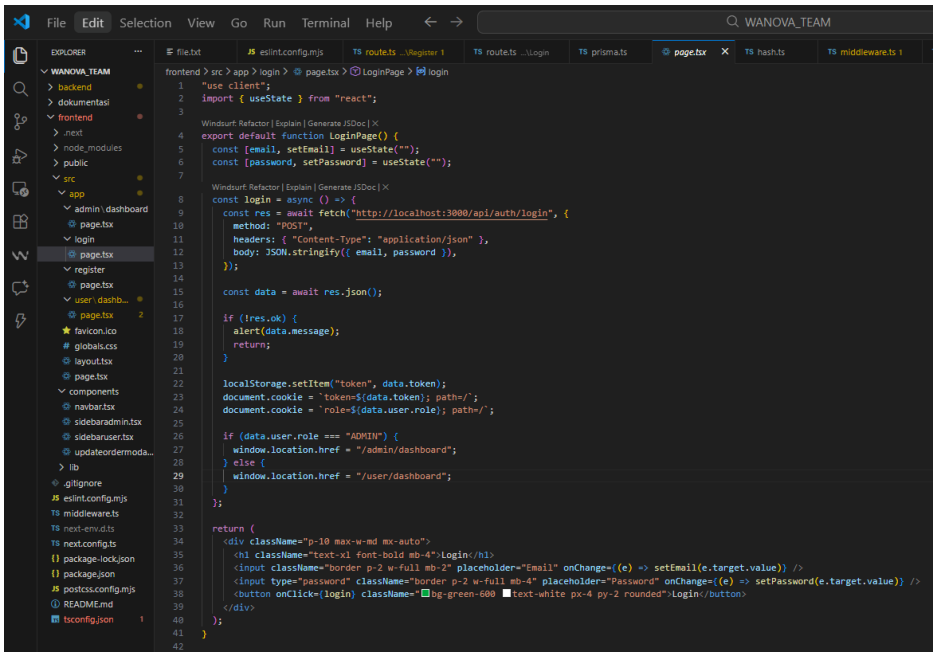
Kode ini adalah halaman dashboard admin berbasis React (Next.js) yang menampilkan daftar pesanan dalam bentuk tabel. Data pesanan disimpan di state orders, lalu ditampilkan dengan kolom ID, judul, status, dan nomor tracking. Admin dapat menekan tombol **Update** pada setiap baris untuk memilih pesanan, yang kemudian akan dibuka di komponen UpdateOrderModal. Fungsi handleSave digunakan untuk memperbarui data pesanan di state setelah diedit. Halaman ini juga menggunakan komponen Navbar (dengan role ADMIN) dan SidebarAdmin sebagai navigasi, sehingga membentuk tampilan panel admin lengkap untuk memantau dan mengelola pesanan.

11. frontend/Components/Navbar.tsx

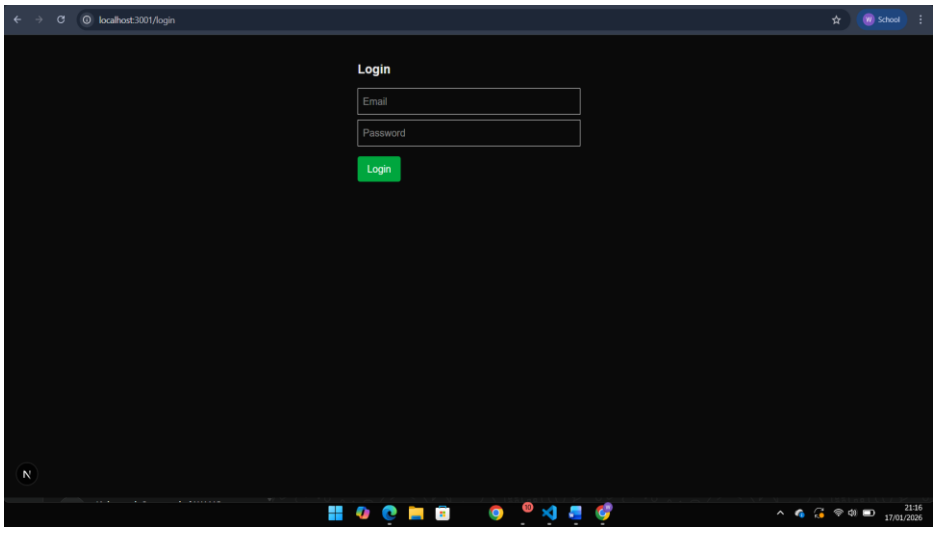


Kode ini adalah komponen Navbar untuk aplikasi yang menampilkan menu sesuai dengan peran pengguna (ADMIN atau USER). Jika role adalah ADMIN, akan muncul link ke Dashboard Admin, dan jika USER, akan muncul link ke Dashboard User. Terdapat juga tombol Logout yang berfungsi menghapus token dari localStorage lalu mengarahkan kembali ke halaman login menggunakan router Next.js. Navbar ini menjadi navigasi utama di bagian atas aplikasi dengan tampilan sederhana dan responsif.

12. Frontend/login/page.tsx

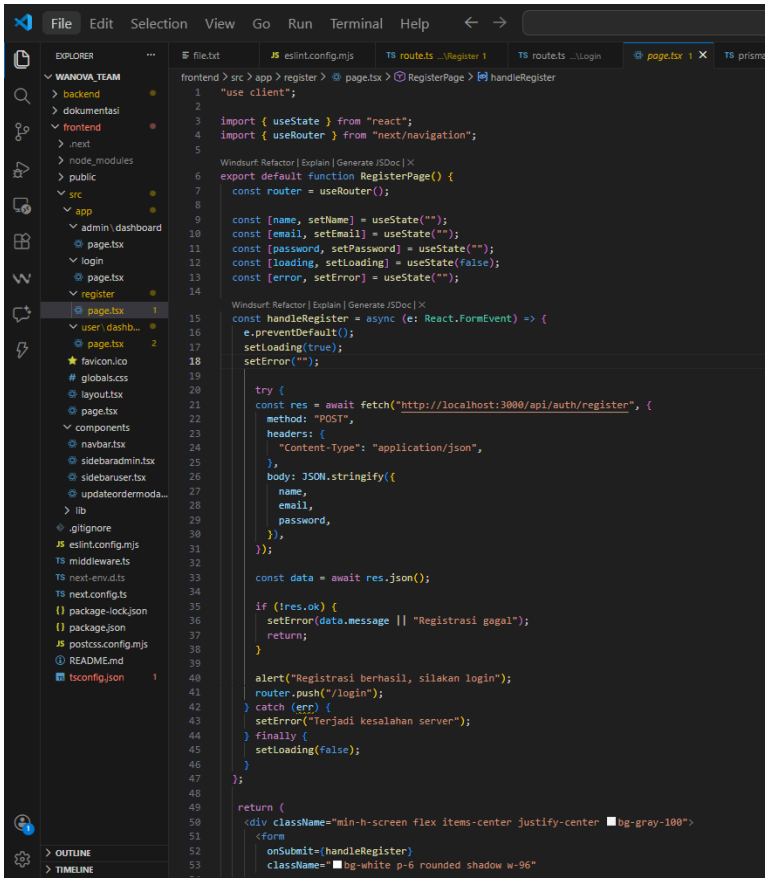


```tsx
"use client";
import { useState } from "react";

export default function LoginPage() {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");

  const login = async () => {
    const res = await fetch("http://localhost:3000/api/auth/login", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({ email, password }),
    });

    const data = await res.json();

    if (!res.ok) {
      alert(data.message);
      return;
    }

    localStorage.setItem("token", data.token);
    document.cookie = `token=${data.token}; path=/`;
    document.cookie = `role=${data.user.role}; path=/`;

    if (data.user.role === "ADMIN") {
      window.location.href = "/admin/dashboard";
    } else {
      window.location.href = "/user/dashboard";
    }
  };

  return (
    <div className="p-10 max-w-md mx-auto">
      <h1 className="text-xl font-bold mb-4">Login</h1>
      <input className="border p-2 w-full mb-2" placeholder="Email" onChange={(e) => setEmail(e.target.value)} />
      <input type="password" className="border p-2 w-full mb-4" placeholder="Password" onChange={(e) => setPassword(e.target.value)} />
      <button onClick={login} className="bg-green-600 text-white px-4 py-2 rounded">Login</button>
    </div>
  );
}
```

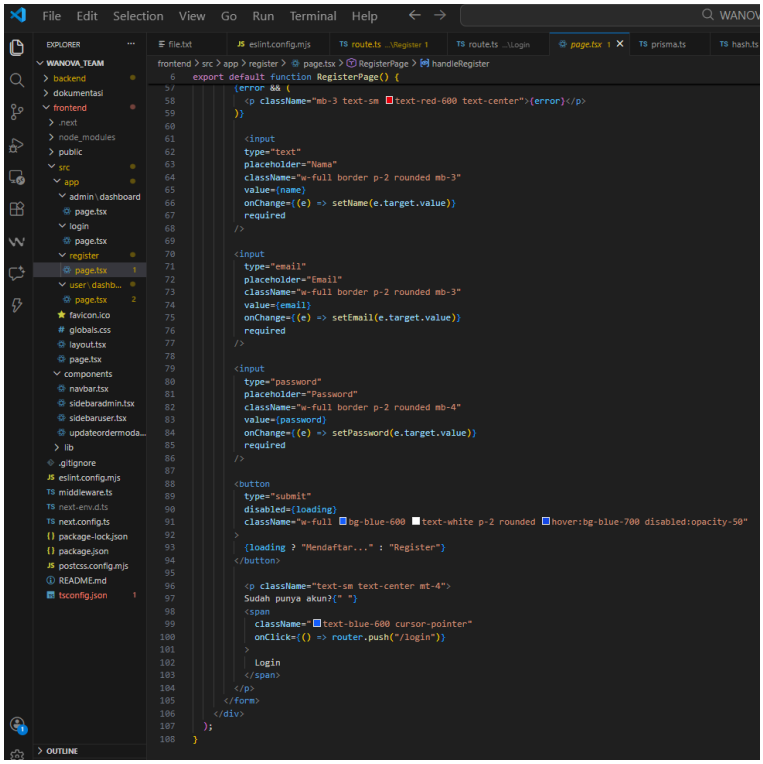


Kode ini adalah halaman login berbasis React yang memungkinkan user memasukkan email dan password, lalu mengirimkannya ke API /api/auth/login menggunakan metode POST. Jika login gagal, pesan error akan ditampilkan, dan jika berhasil, token JWT disimpan di localStorage serta cookie, kemudian sistem mengecek role user (ADMIN atau USER) untuk mengarahkan ke dashboard yang sesuai. Halaman ini menggunakan state untuk menyimpan input dan tombol Login untuk menjalankan proses autentikasi.

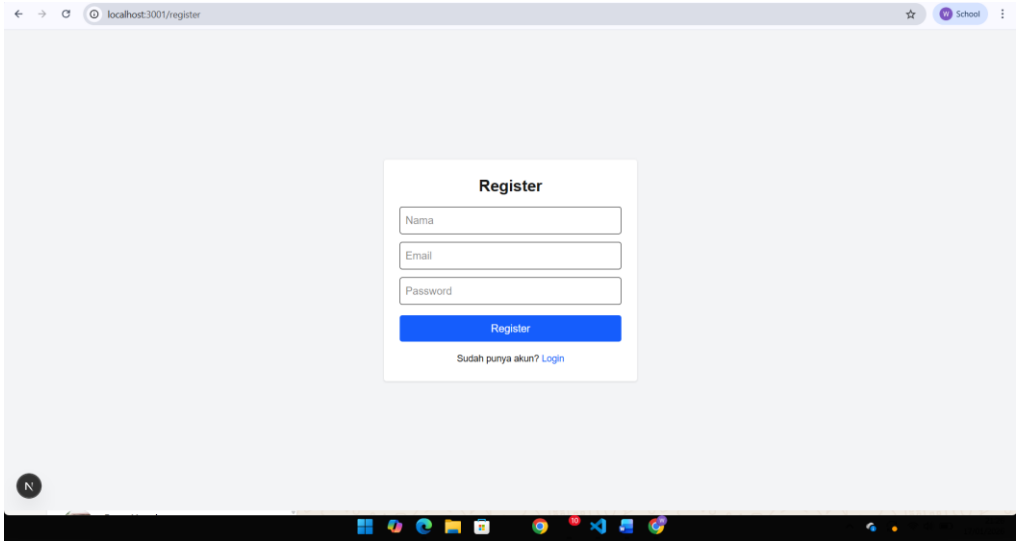
13. Frontend/register/page.tsx



```tsx
"use client";

import { useState } from "react";
import { useRouter } from "next/navigation";

export default function RegisterPage() {
  const router = useRouter();

  const [name, setName] = useState("");
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState("");

  const handleRegister = async (e: React.FormEvent) => {
    e.preventDefault();
    setLoading(true);
    setError("");

    try {
    const res = await fetch("http://localhost:3000/api/auth/register", {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify({
        name,
        email,
        password,
      }),
    });

    const data = await res.json();

    if (!res.ok) {
      setError(data.message || "Registrasi gagal");
      return;
    }

    alert("Registrasi berhasil, silakan login");
    router.push("/login");
  } catch (err) {
    setError("Terjadi kesalahan server");
  } finally {
    setLoading(false);
  }
};

  return (
    <div className="min-h-screen flex items-center justify-center bg-gray-100">
      <form
        onSubmit={handleRegister}
        className="bg-white p-6 rounded shadow w-96"
```

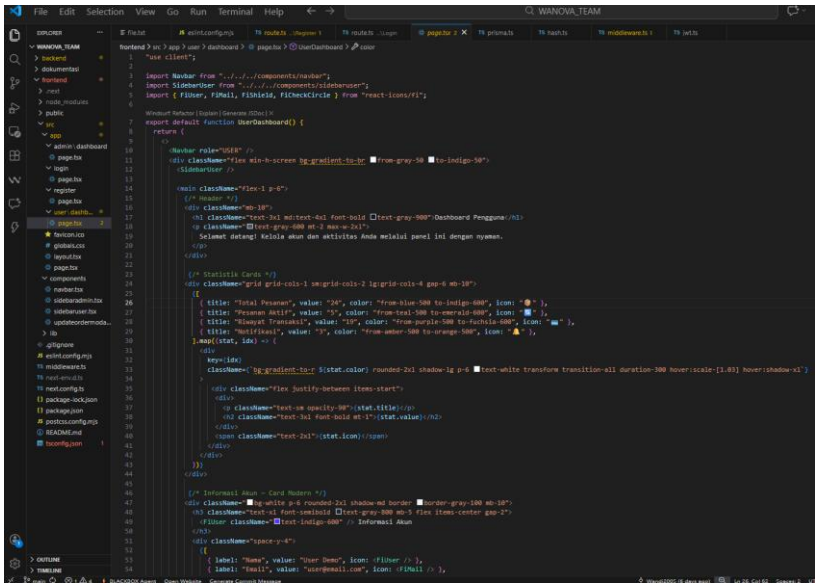


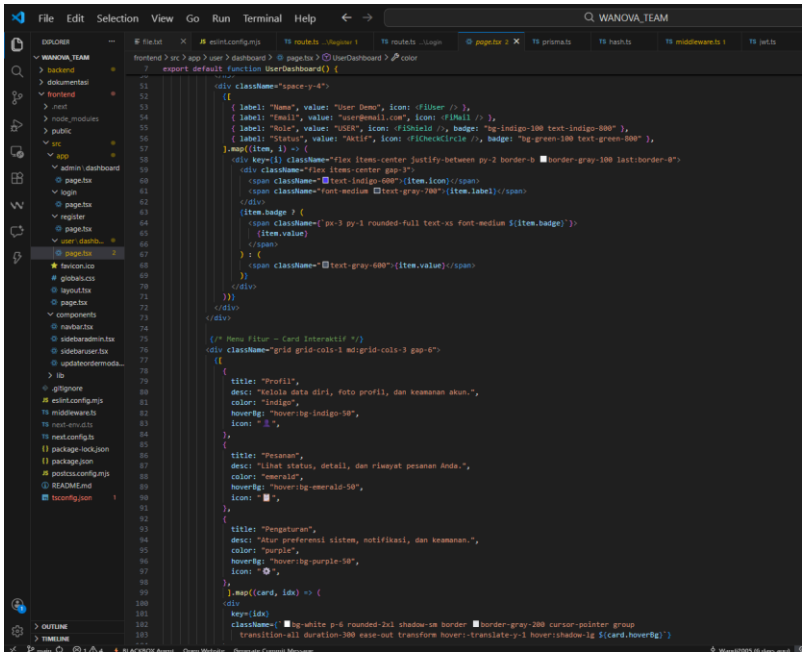
```tsx
        {error && (
          <p className="mb-3 text-sm text-red-600 text-center">{error}</p>
        )}

        <input
          type="text"
          placeholder="Nama"
          className="w-full border p-2 rounded mb-3"
          value={name}
          onChange={(e) => setName(e.target.value)}
          required
        />

        <input
          type="email"
          placeholder="Email"
          className="w-full border p-2 rounded mb-3"
          value={email}
          onChange={(e) => setEmail(e.target.value)}
          required
        />

        <input
          type="password"
          placeholder="Password"
          className="w-full border p-2 rounded mb-4"
          value={password}
          onChange={(e) => setPassword(e.target.value)}
          required
        />

        <button
          type="submit"
          disabled={loading}
          className="w-full bg-blue-600 text-white p-2 rounded hover:bg-blue-700 disabled:opacity-50"
        >
          {loading ? "Mendaftar..." : "Register"}
        </button>

        <p className="text-sm text-center mt-4">
          Sudah punya akun? {" "}
          <span
            className="text-blue-600 cursor-pointer"
            onClick={() => router.push("/login")}
          >
            Login
          </span>
        </p>
      </form>
    </div>
  );
}
```

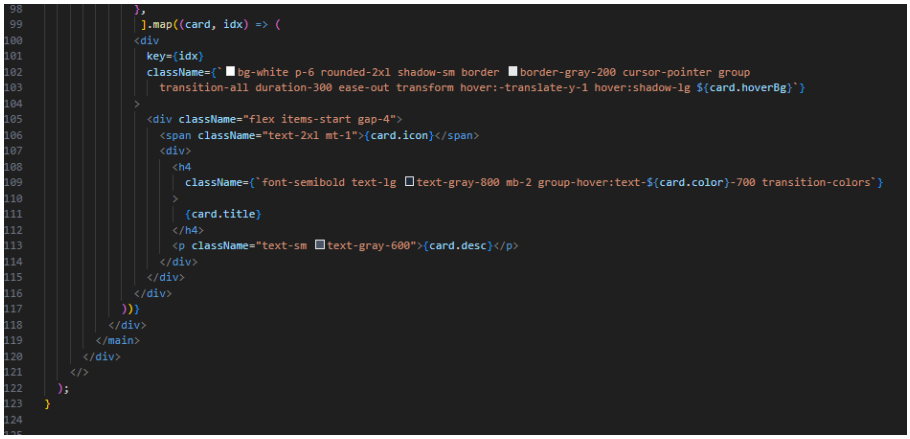


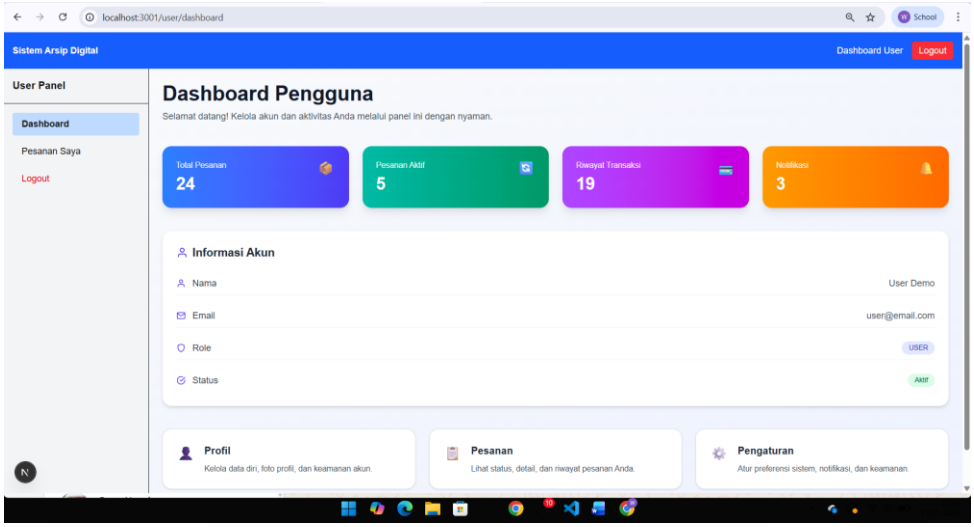
Kode ini adalah halaman registrasi user di Next.js yang menampilkan form untuk mengisi nama, email, dan password, lalu mengirimkannya ke API /api/auth/register dengan metode POST. Data input disimpan menggunakan state, dan saat tombol Register ditekan, sistem menampilkan status loading, menangani error jika gagal, serta menampilkan pesan sukses jika berhasil. Setelah registrasi sukses, user otomatis diarahkan ke halaman login. Tampilan dibuat sederhana dengan Tailwind CSS dan sudah dilengkapi validasi dasar serta navigasi ke halaman login.

14. Frontend/user/dashboard/page.tsx









Kode ini adalah halaman Dashboard User di Next.js yang menampilkan panel utama pengguna setelah login, lengkap dengan Navbar khusus role USER dan SidebarUser sebagai navigasi. Di dalamnya terdapat tampilan statistik pesanan, informasi akun (nama, email, role, status), serta kartu menu interaktif untuk fitur Profil, Pesanan, dan Pengaturan, dengan desain modern menggunakan Tailwind

CSS dan ikon dari React Icons. Halaman ini berfungsi sebagai pusat informasi dan kontrol aktivitas pengguna dalam sistem pengelolaan arsip/pesanan.