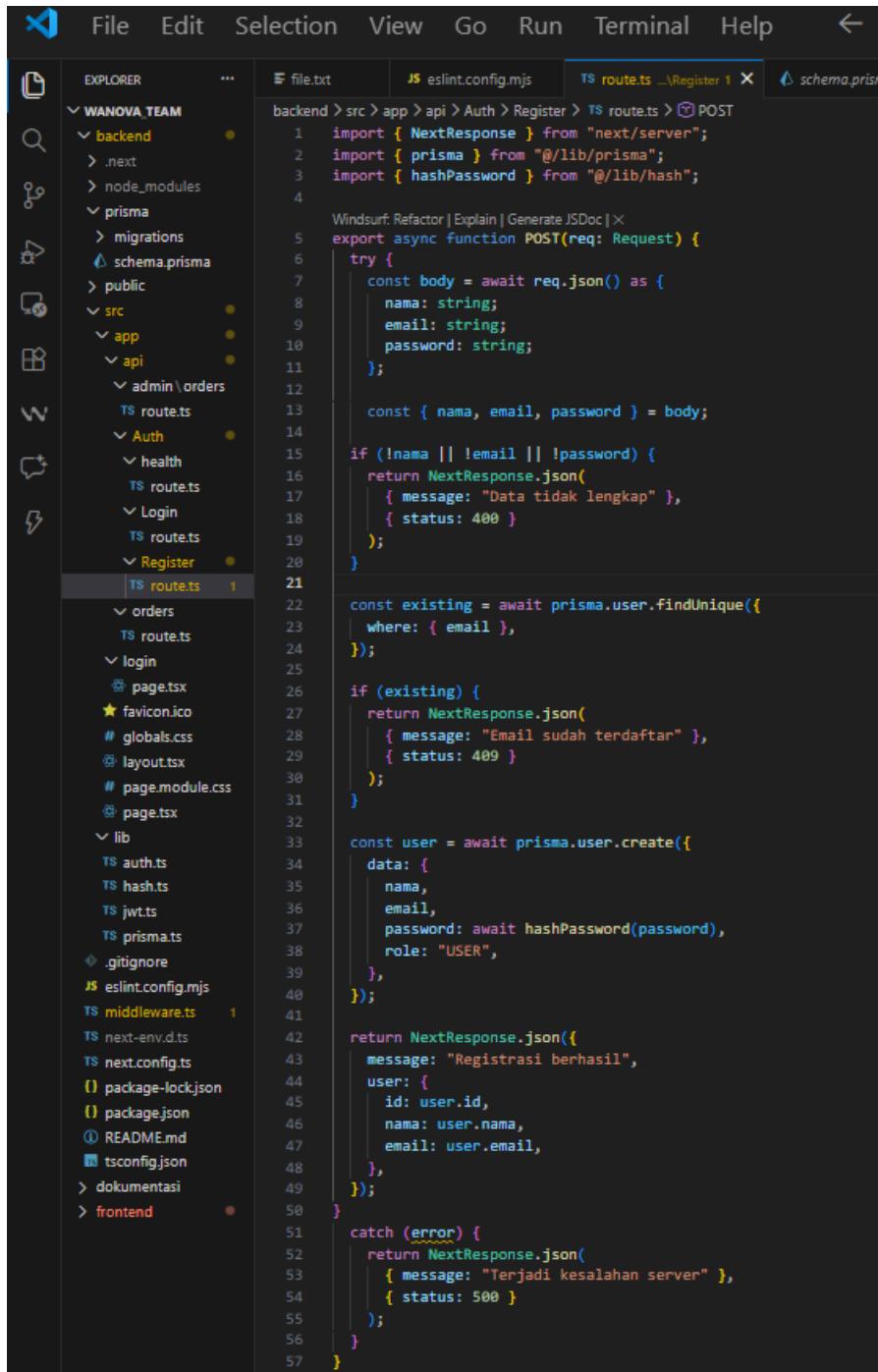


## DOKUMENTASI PROJEK PWBL

NAMA: WANDI FILEMON HOTMARTUA SIANTURI

NPM: 23313009

### 1. Backend/Register/route.ts



The screenshot shows a code editor with the following details:

- File Explorer:** Shows the project structure under "WANOVA TEAM". Key files include ".next", "node\_modules", "prisma", "schema.prisma", "public", "src", "app", "api", "admin\orders", "Auth", "health", "Login", and "Register".
- Code Editor:** The current file is "route.ts" located in the "Register" directory.
- Content:** The code is a POST endpoint for user registration. It imports "NextResponse" from "next/server", "prisma" from "@lib/prisma", and "hashPassword" from "@lib/hash". The function handles a POST request with body fields "nama", "email", and "password". It checks if all fields are present. If not, it returns a JSON response with an error message and status 400. If the email already exists in the database, it returns a JSON response with an error message and status 409. Otherwise, it creates a new user in the database using Prisma's "create" method, setting "role" to "USER". Finally, it returns a JSON response with a success message and the user's details (id, nama, email).

```
import { NextResponse } from "next/server";
import { prisma } from "@lib/prisma";
import { hashPassword } from "@lib/hash";

export async function POST(req: Request) {
  try {
    const body = await req.json() as {
      nama: string;
      email: string;
      password: string;
    };

    const { nama, email, password } = body;

    if (!nama || !email || !password) {
      return NextResponse.json(
        { message: "Data tidak lengkap" },
        { status: 400 }
      );
    }

    const existing = await prisma.user.findUnique({
      where: { email },
    });

    if (existing) {
      return NextResponse.json(
        { message: "Email sudah terdaftar" },
        { status: 409 }
      );
    }

    const user = await prisma.user.create({
      data: {
        nama,
        email,
        password: await hashPassword(password),
        role: "USER",
      },
    });

    return NextResponse.json({
      message: "Registrasi berhasil",
      user: {
        id: user.id,
        nama: user.nama,
        email: user.email,
      },
    });
  } catch (error) {
    return NextResponse.json(
      { message: "Terjadi kesalahan server" },
      { status: 500 }
    );
  }
}
```

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.

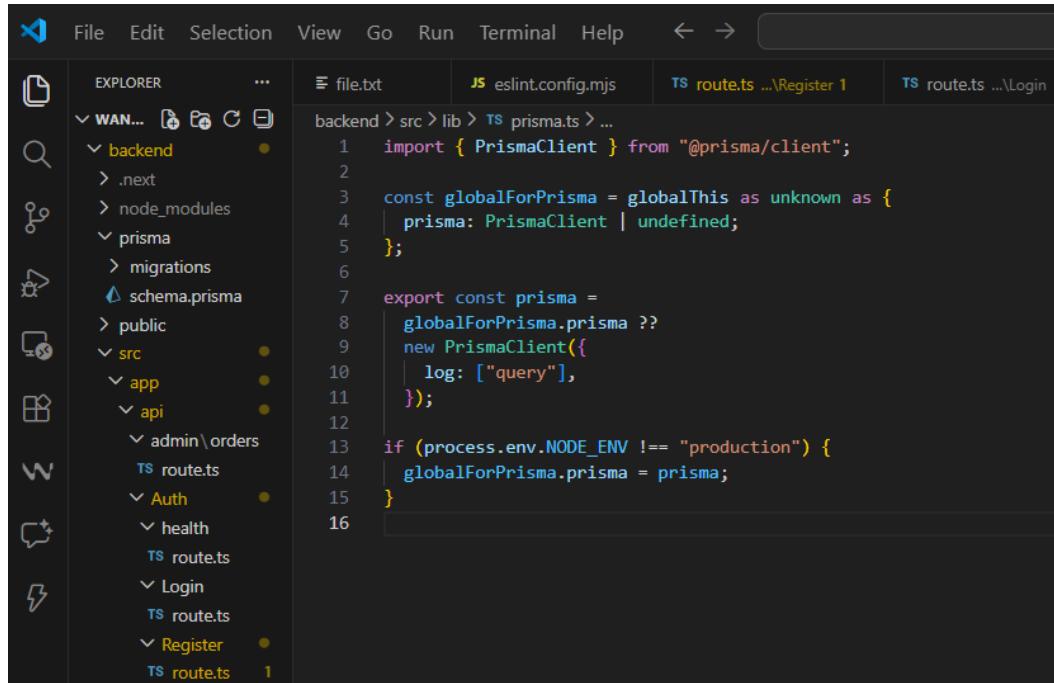
## 2. Backend/Api/auth/login

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows the project structure under "WANOVATEAM". Key components include "backend", "src", "app", "api", "Auth", "Login", "Register", "orders", "page.tsx", "lib", and "prisma".
- Code Editor:** Displays the content of the file "route.ts". The code is a Next.js API endpoint for logging in users. It imports "NextResponse" from "next/server", "prisma" from "@lib/prisma", "comparePassword" from "@lib/hash", and "signToken" from "@lib/jwt". It handles a POST request with JSON body containing "email" and "password". It finds a user by email using Prisma's findUnique query. If no user is found, it returns a 404 response with an error message. It then checks if the provided password matches the hashed password in the database using comparePassword. If they don't match, it returns a 401 response with an error message. If they do match, it generates a JWT token for the user (containing user.id, role, and user.role) using signToken and returns a success response with the message "Login berhasil", the token, and the user's role.
- Bottom Bar:** Includes tabs for "main", "BLACKBOX Agent", "Open Website", and "Generate Commit Message".

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

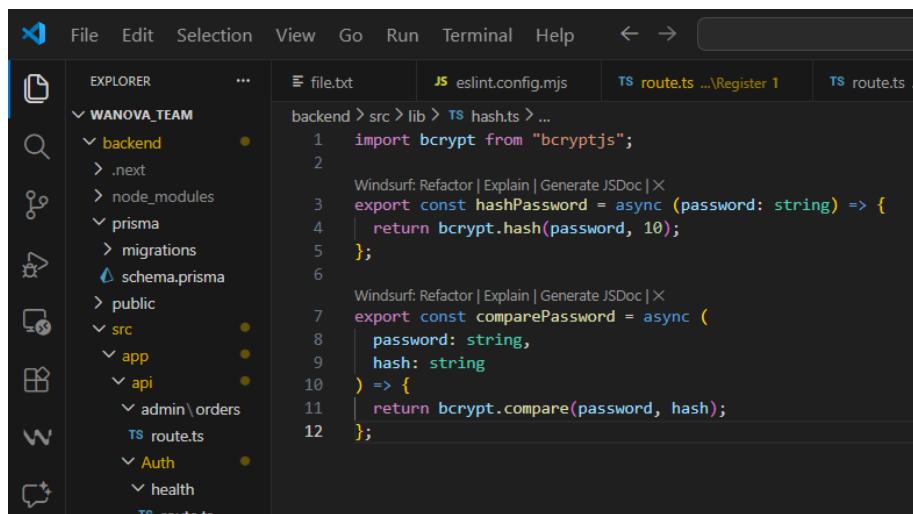


The screenshot shows the VS Code interface with the Explorer sidebar open, displaying the project structure. The current file being edited is `schema.prisma`. The code in the editor is as follows:

```
backend > src > lib > ts prisma.ts > ...
1 import { PrismaClient } from "@prisma/client";
2
3 const globalForPrisma = globalThis as unknown as {
4   prisma: PrismaClient | undefined;
5 };
6
7 export const prisma =
8   globalForPrisma.prisma ??
9   new PrismaClient({
10   log: ["query"],
11 });
12
13 if (process.env.NODE_ENV !== "production") {
14   globalForPrisma.prisma = prisma;
15 }
16
```

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



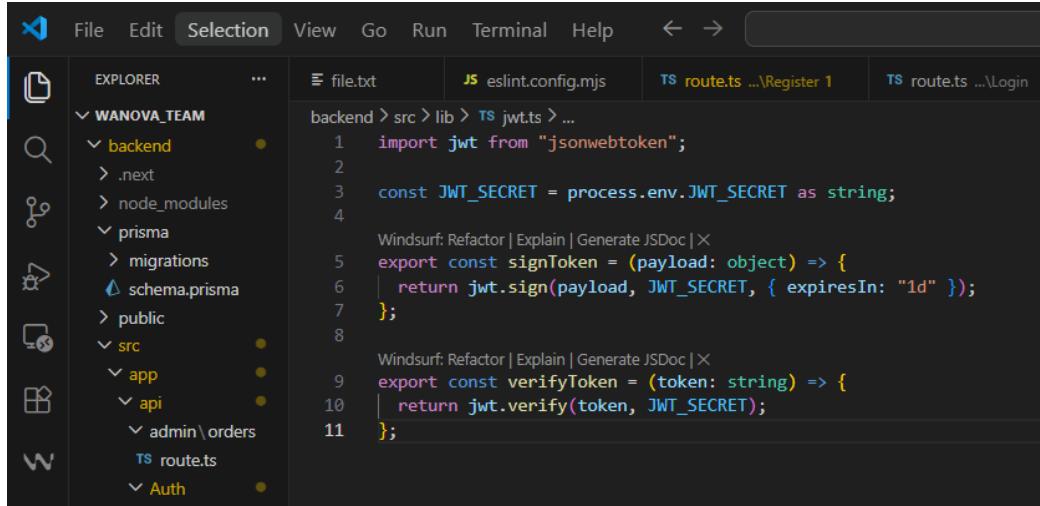
The screenshot shows the VS Code interface with the Explorer sidebar open, displaying the project structure. The current file being edited is `hash.ts`. The code in the editor is as follows:

```
backend > src > lib > ts hash.ts > ...
1 import bcrypt from "bcryptjs";
2
3 Windsurf: Refactor | Explain | Generate JSDoc | ×
4 export const hashPassword = async (password: string) => {
5   return bcrypt.hash(password, 10);
6 }
7 Windsurf: Refactor | Explain | Generate JSDoc | ×
8 export const comparePassword = async (
9   password: string,
10  hash: string
11 ) => {
12   return bcrypt.compare(password, hash);
13 }
```

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

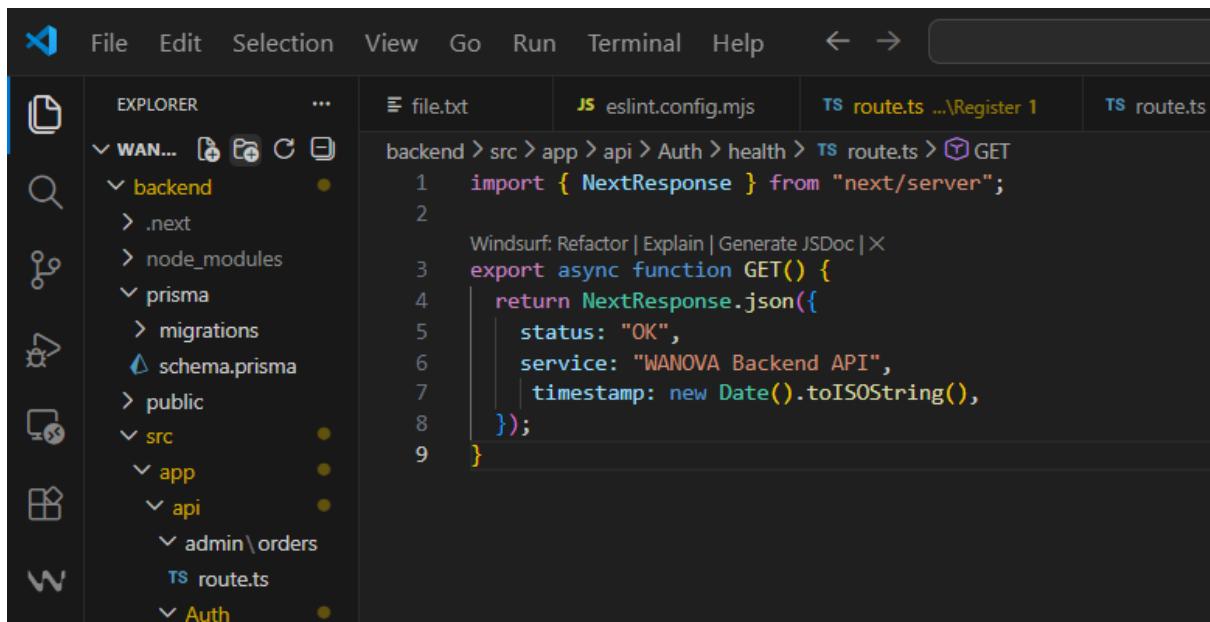


The screenshot shows the VS Code interface with the 'Selection' tab active. The Explorer sidebar shows a project structure under 'WANOVA\_TEAM' with 'backend' as the active folder. The 'route.ts' file is open in the editor, displaying the following TypeScript code:

```
backend > src > lib > ts jwt.ts > ...
1 import jwt from "jsonwebtoken";
2
3 const JWT_SECRET = process.env.JWT_SECRET as string;
4
5 export const signToken = (payload: object) => {
6   return jwt.sign(payload, JWT_SECRET, { expiresIn: "1d" });
7 }
8
9 export const verifyToken = (token: string) => {
10  return jwt.verify(token, JWT_SECRET);
11 }
```

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



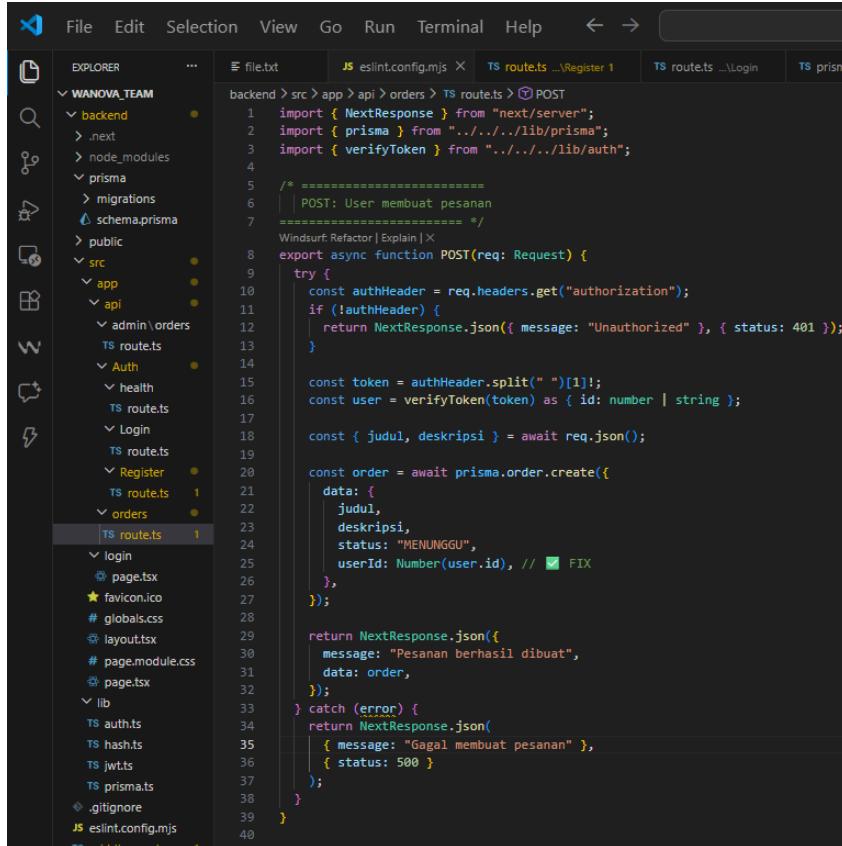
The screenshot shows the VS Code interface with the 'Selection' tab active. The Explorer sidebar shows a project structure under 'WANOVA\_TEAM' with 'backend' as the active folder. The 'route.ts' file is open in the editor, displaying the following TypeScript code:

```
backend > src > app > api > Auth > health > ts route.ts > GET
1 import { NextResponse } from "next/server";
2
3 export async function GET() {
4   return NextResponse.json({
5     status: "OK",
6     service: "WANOVA Backend API",
7     timestamp: new Date().toISOString(),
8   });
9 }
```

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



The screenshot shows a code editor with the following details:

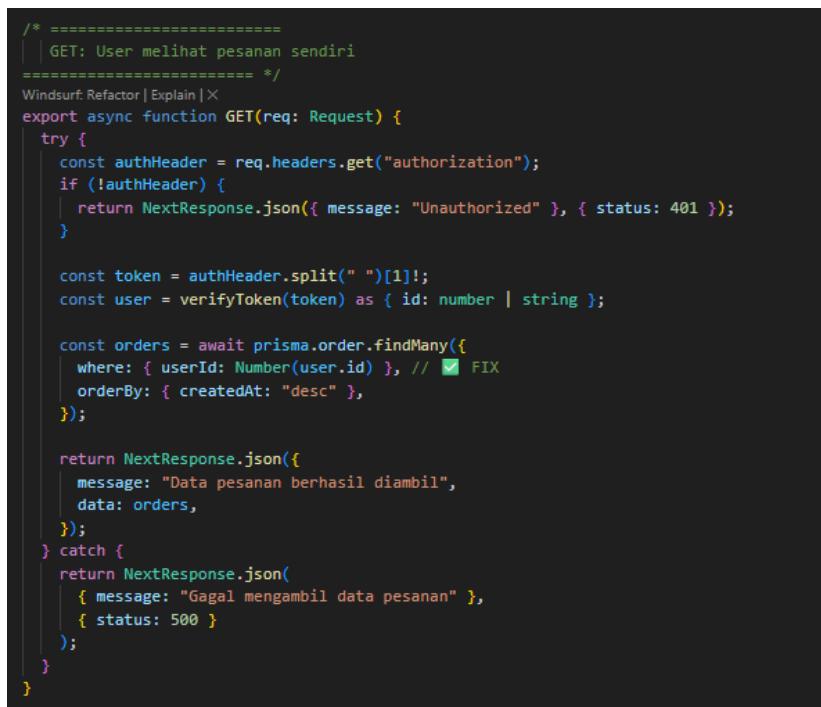
- File Explorer:** Shows the project structure under "WANOVA.TEAM". The "orders" folder is selected.
- Code Editor:** Displays the content of "route.ts" for the POST endpoint.

```
/* ===== */
| | GET: User melihat pesanan sendiri
===== */
Windsurf Refactor | Explain | X
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 }
  );
}
}
```



The screenshot shows a code editor with the following details:

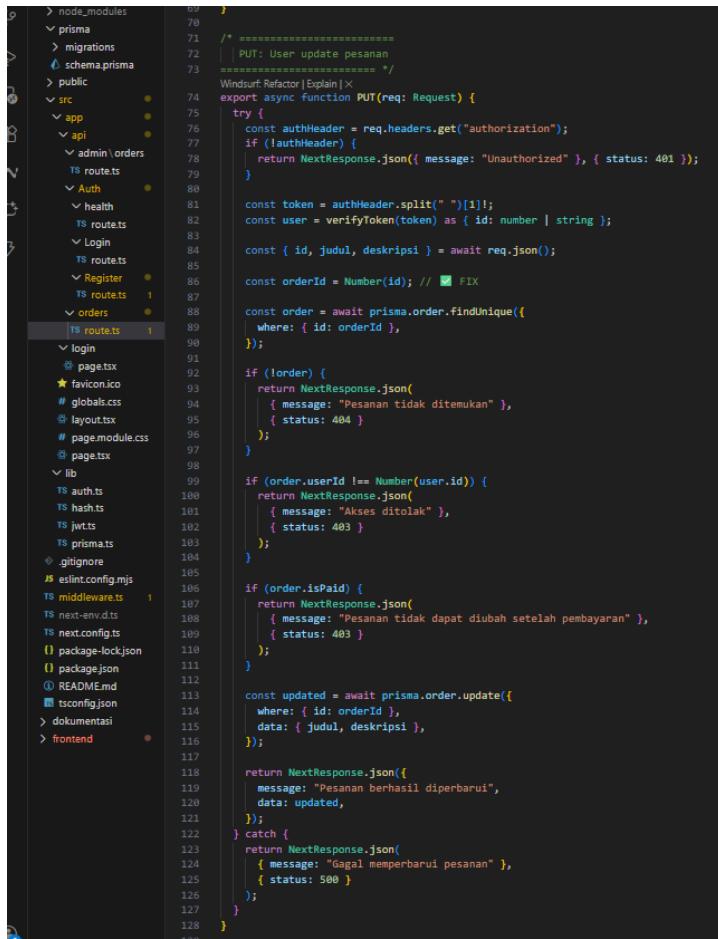
- Code Editor:** Displays the content of "route.ts" for the GET endpoint.

```
/* ===== */
| | POST: User membuat pesanan
===== */
Windsurf Refactor | Explain | X
export async function POST(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 order = await prisma.order.create({
    data: {
      judul,
      deskripsi,
      status: "MENUNGGU",
      userId: Number(user.id), // ✅ FIX
    },
  });

  return NextResponse.json({
    message: "Pesanan berhasil dibuat",
    data: order,
  });
} catch (error) {
  return NextResponse.json(
    { message: "Gagal membuat pesanan" },
    { status: 500 }
  );
}
}
```



```
 67  }
 68  */
 69  /* ===== */
 70  |  PUT: User update pesanan
 71  |=====
 72  |
 73  */
 74  export async function PUT(req: Request) {
 75  try {
 76  const authHeader = req.headers.get("authorization");
 77  if (!authHeader) {
 78  return NextResponse.json({ message: "Unauthorized" }, { status: 401 });
 79  }
 80
 81  const token = authHeader.split(" ")[1]!;
 82  const user = verifyToken(token) as { id: number | string };
 83
 84  const { id, judul, deskripsi } = await req.json();
 85
 86  const orderId = Number(id); // ✘ FIX
 87
 88  const order = await prisma.order.findUnique({
 89  where: { id: orderId },
 90  });
 91
 92  if (!order) {
 93  return NextResponse.json(
 94  { message: "Pesanan tidak ditemukan" },
 95  { status: 404 }
 96  );
 97  }
 98
 99  if (order.userId !== Number(user.id)) {
100  return NextResponse.json(
101  { message: "Akses ditolak" },
102  { status: 403 }
103  );
104
105  if (order.isPaid) {
106  return NextResponse.json(
107  { message: "Pesanan tidak dapat diubah setelah pembayaran" },
108  { status: 403 }
109  );
110
111  const updated = await prisma.order.update({
112  where: { id: orderId },
113  data: { judul, deskripsi },
114  });
115
116
117  return NextResponse.json(
118  { message: "Pesanan berhasil diperbarui" },
119  { data: updated },
120  );
121
122  } catch {
123  return NextResponse.json(
124  { message: "Gagal memperbarui pesanan" },
125  { status: 500 }
126  );
127  }
128 }
```

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

```

File Edit Selection View Go Run Terminal Help < > Q: WANOVATEAM
EXPLORER ... file.txt JS eslint.config.mjs TS route.ts ...\\Register 1 TS route.ts ...\\Login TS prismats TS layout.tsx 1
frontend > src > app > layout.tsx ...
1 import type { Metadata } from "next";
2 import { Geist, Geist_Mono } from "next/font/google";
3 import "../globals.css";
4
5 const geistSans = Geist({
6   variable: "--font-geist-sans",
7   subsets: ["latin"],
8 });
9
10 const geistMono = Geist_Mono({
11   variable: "--font-geist-mono",
12   subsets: ["latin"],
13 });
14
15 export const metadata: Metadata = {
16   title: "Sistem Pengelolaan Arsip Digital",
17   description:
18     "Implementasi Web Service Pengelolaan Arsip Digital berbasis RESTful API",
19 };
20
21 Windsurf: Refactor | Explain | Generate JSDoc | X
22 export default function RootLayout({ children }) {
23   return (
24     <html lang="id">
25       <body>
26         <div className={`${geistSans.variable} ${geistMono.variable} antialiased bg-gray-100`}>
27           {children}
28         </div>
29       </body>
30     </html>
31   );
32 }
33
34
35
36
37

```

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

```

File Edit Selection View Go Run Terminal Help < > Q: WANOVATEAM
EXPLORER ... file.txt JS eslint.config.mjs TS route.ts ...\\Register 1 TS route.ts ...\\Login TS prismats TS layout.tsx 1
frontend > src > components > sidebaradmin.tsx > SidebarAdmin
Windsurf: Refactor | Explain | Generate JSDoc | X
1 export default function SidebarAdmin() {
2   return (
3     <div className="w-60 bg-gray-200 h-screen p-4">
4       <ul className="space-y-2">
5         <li>
6           <a href="/admin/dashboard">Dashboard</a>
7         </li>
8         <li>
9           <a href="/admin/orders">Kelola Pesanan</a>
10        </li>
11      </ul>
12    </div>
13  );
14
15
16

```

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.

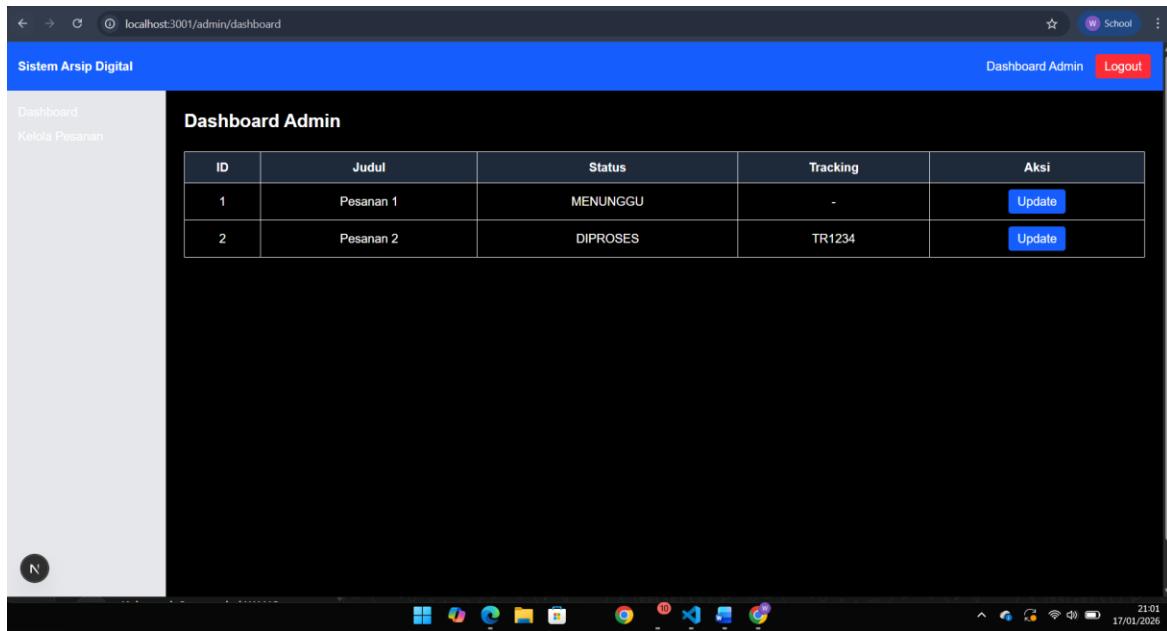
## 10. Frontend/Admin/dashboard/page.tsx

```
File Edit Selection View Go Run Terminal Help < >
EXPLORER ... # metxt x JS eslintconfig.mjs TS routes..._Register I TS routes..._Login TS prismas
V WANOWA_TEAM
  > backend
  > dokumentasi
  > frontend
    > next
    > node_modules
    > public
      > src
        > app
          > admin\dashboard
            > pages\tsx
              > login
              > register
              > user
                > favicon.ico
                > global.css
                > layout.tsx
                > page.tsx
                > components
                  > navbar.tsx
                  > sidebaradmin.tsx
                  > sidebaruser.tsx
                  > updateordermodal.tsx
                > lib
                > .gitignore
                JS eslintconfig.mjs
                TS middleware.ts
                TS next-env.dts
                TS next.config.ts
                () package-lock.json
                () package.json
                JS postcssconfig.mjs
                () README.md
                tsconfig.json

frontend > src > app > admin > dashboard > pages\tsx > AdminDashboard > handleSave
  1  "use client";
  2  import { useState } from "react";
  3  import Navbar from "../../../../../Components/navbar";
  4  import SidebarAdmin from "../../../../../Components/sidebaradmin";
  5  import UpdateOrderModal from "../../../../../Components/updateordermodal";
  6
  7  Windurf Refactor | Explain
  8  interface Order {
  9    id: number;
 10    judul: string;
 11    status: string;
 12    tracking: string | null;
 13  }
 14
 15  Windurf Refactor | Explain | Generate JSDoc
 16  export default function AdminDashboard() {
 17    const [orders, setOrders] = useState<Order>([
 18      { id: 1, judul: "Pesanan 1", status: "MENUNGGU", tracking: null },
 19      { id: 2, judul: "Pesanan 2", status: "DIPROSSES", tracking: "TR1234" },
 20    ]);
 21
 22  Windurf Refactor | Explain | Generate JSDoc
 23  const handleSave = (updatedOrder: Order) => {
 24    setOrders((prev) => {
 25      prev.map(o => (o.id === updatedOrder.id ? updatedOrder : o));
 26    });
 27    setSelectedOrder(null);
 28  }
 29
 30  return (
 31    <>
 32      <Navbar role="ADMIN" />
 33      <div className="flex min-h-screen bg-black text-white">
 34        <SidebarAdmin />
 35        <main className="p-6 w-full">
 36          <h1 className="text-2xl font-bold mb-6">Dashboard Admin</h1>
 37
 38          <table className="w-full border border-gray-700">
 39            <thead className="bg-gray-800">
 40              <tr>
 41                <th className="p-2 border" ID/>
 42                <th className="p-2 border" Judul/>
 43                <th className="p-2 border" Status/>
 44                <th className="p-2 border" Tracking/>
 45                <th className="p-2 border" Aksi/>
 46              </tr>
 47            </thead>
 48
 49            <tbody>
 50              <tr key={order.id}>
 51                <td>{order.id}</td>
 52                <td>{order.judul}</td>
 53                <td>{order.status}</td>
 54                <td>{order.tracking}</td>
 55                <td>
 56                  <button onClick={() => setSelectedOrder(order)}>
 57                    Update
 58                  </button>
 59                </td>
 60              </tr>
 61            </tbody>
 62          </table>
 63        </main>
 64      </div>
 65    </>
 66  </>
 67  <UpdateOrderModal
 68    order={selectedOrder}
 69    onClose={() => setSelectedOrder(null)}
 70    onSave={handleSave}
 71  >
 72  </UpdateOrderModal>
 73  </>
 74  </>
 75  </>
 76  </>
 77  </>
 78  </>
 79  </>
 80  </>
 81  </>
```

```
File Edit Selection View Go Run Terminal Help < >
EXPLORER ... # file.txt JS eslintconfig.mjs TS routes..._Register I TS routes..._Login TS prismas
V WANOWA_TEAM
  > backend
  > dokumentasi
  > frontend
    > next
    > node_modules
    > public
      > src
        > app
          > admin\dashboard
            > pages\tsx
              > login
              > register
              > user
                > favicon.ico
                > global.css
                > layout.tsx
                > page.tsx
                > components
                  > navbar.tsx
                  > sidebaradmin.tsx
                  > sidebaruser.tsx
                  > updateordermodal.tsx
                > lib
                > .gitignore
                JS eslintconfig.mjs
                TS middleware.ts
                TS next-env.dts
                TS next.config.ts
                () package-lock.json
                () package.json
                JS postcssconfig.mjs
                () README.md
                tsconfig.json

frontend > src > app > admin > dashboard > pages\tsx > AdminDashboard > handleSave
  14  export default function AdminDashboard() {
 30    <>
 31      <Navbar role="ADMIN" />
 32      <div className="flex min-h-screen bg-black text-white">
 33        <SidebarAdmin />
 34        <main className="p-6 w-full">
 35          <h1 className="text-2xl font-bold mb-6">Dashboard Admin</h1>
 36
 37          <table className="w-full border border-gray-700">
 38            <thead className="bg-gray-800">
 39              <tr>
 40                <th className="p-2 border" ID/>
 41                <th className="p-2 border" Judul/>
 42                <th className="p-2 border" Status/>
 43                <th className="p-2 border" Tracking/>
 44                <th className="p-2 border" Aksi/>
 45              </tr>
 46            </thead>
 47
 48            <tbody>
 49              <tr key={order.id}>
 50                <td>{order.id}</td>
 51                <td>{order.judul}</td>
 52                <td>{order.status}</td>
 53                <td>{order.tracking}</td>
 54                <td>
 55                  <button onClick={() => setSelectedOrder(order)}>
 56                    Update
 57                  </button>
 58                </td>
 59              </tr>
 60            </tbody>
 61          </table>
 62        </main>
 63      </div>
 64    </>
 65  </>
 66  <UpdateOrderModal
 67    order={selectedOrder}
 68    onClose={() => setSelectedOrder(null)}
 69    onSave={handleSave}
 70  >
 71  </UpdateOrderModal>
 72  </>
 73  </>
 74  </>
 75  </>
 76  </>
 77  </>
 78  </>
 79  </>
 80  </>
 81  </>
```



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

```

File Edit Selection View Go Run Terminal Help < >
EXPLORER file.txt eslint.config.js TS routes.ts ... \Register 1 TS routes.ts ... \Login TS prismata ...
WANNOA_TEAM
> backend
> dokumentasi
< frontend
> .next
> node_modules
> public
  < src
    < app
      < admin\dashboard
        page.tsx
      > login
      > register
      > user
      favicon.ico
      # globals.css
      layout.tsx
      page.tsx
      < components
        navbar.tsx
        sidebaradmin.tsx
        sidebaruser.tsx
        updateordermoda...
      > lib
      .gitignore
      eslint.config.js
      middleware.ts
      next-env.d.ts
      next.config.ts
      package-lock.json
      package.json
      postcss.config.js
      README.md
      tsconfig.json
    > .gitignore
    eslint.config.js
    middleware.ts
    next-env.d.ts
    next.config.ts
    package-lock.json
    package.json
    postcss.config.js
    README.md
    tsconfig.json
  > .gitignore
  eslint.config.js
  middleware.ts
  next-env.d.ts
  next.config.ts
  package-lock.json
  package.json
  postcss.config.js
  README.md
  tsconfig.json

```

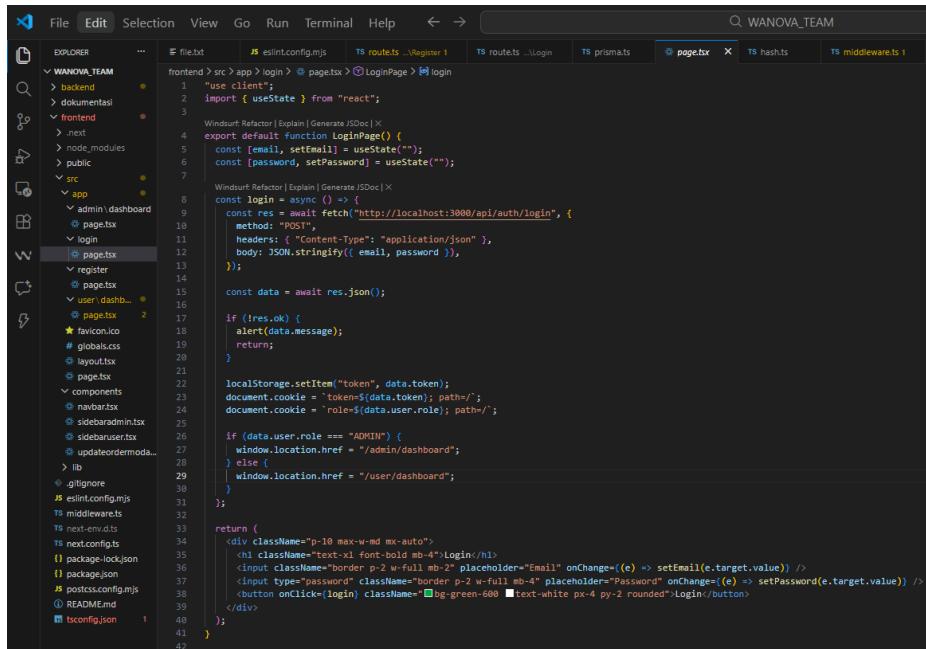
```

frontend > src > components > navbar.tsx > ...
1 "use client";
2
3 import Link from "next/link";
4 import { useRouter } from "next/navigation";
5
6 WinduSurf Refactor | Explain | Generate JSDoc | X
7 export default function Navbar({ role }: { role: "ADMIN" | "USER" }) {
8   const router = useRouter();
9
10  const handleLogout = () => {
11    localStorage.removeItem("token");
12    router.push("/login");
13  };
14
15  return (
16    <header className="bg-blue-600 text-white p-4 flex justify-between items-center">
17      <span className="font-bold">Sistem Arsip Digital</span>
18
19      <nav className="flex gap-4 items-center">
20        {role === "ADMIN" && (
21          <Link href="/admin/dashboard" className="hover:underline">
22            Dashboard Admin
23          </Link>
24        )}
25
26        {role === "USER" && (
27          <Link href="/user/dashboard" className="hover:underline">
28            Dashboard User
29          </Link>
30        )}
31
32        <button
33          onClick={handleLogout}
34          className="bg-red-500 px-3 py-1 rounded hover:bg-red-600"
35        >
36          Logout
37        </button>
38      </nav>
39    </header>
40  );
41}
42

```

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.

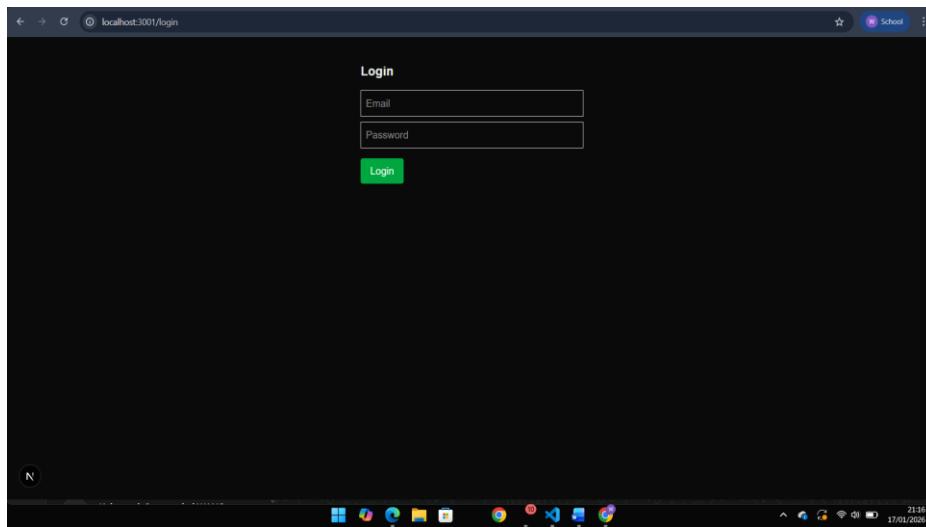
## 12. Frontend/login/page.tsx



```

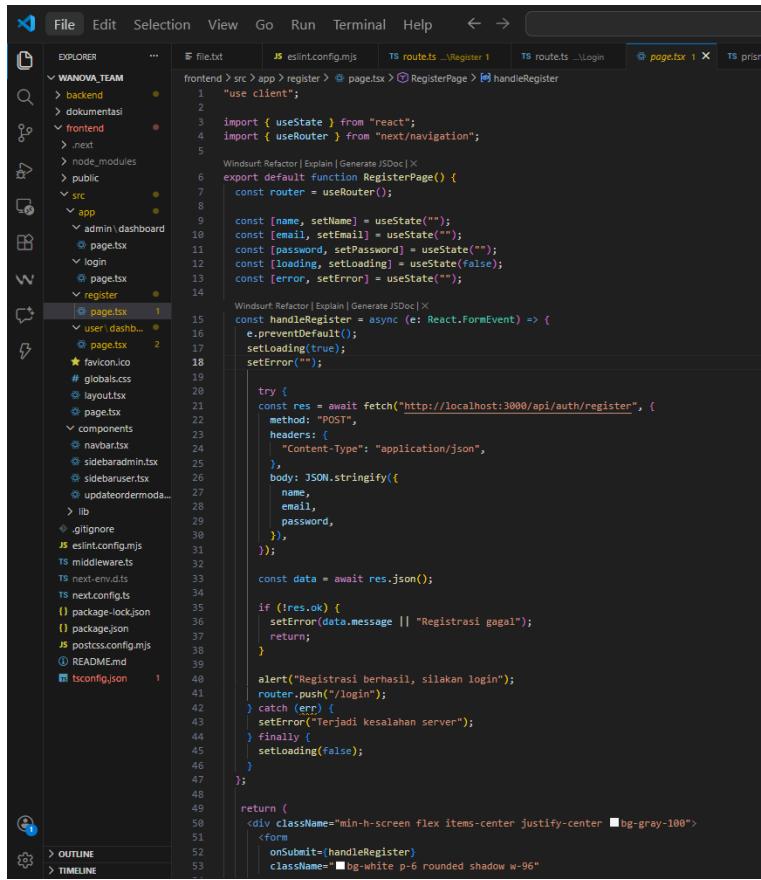
File | Edit | Selection | View | Go | Run | Terminal | Help | ← → | 🔍 WANOVATEAM
EXPLORER | file.txt | JS eslint.config.mjs | TS route.ts ...\\Register | TS route.ts ...\\Login | TS prismास | page.tsx | TS hash.ts | TS middleware.ts
frontend > src > app > login > page.tsx > LoginPage > login
1 "use client";
2 import { useState } from "react";
3
4 WindwsRefactor | Explain | GenerateJSDoc | X
5 export default function LoginPage() {
6   const [email, setEmail] = useState("");
7   const [password, setPassword] = useState("");
8
9   WindwsRefactor | Explain | GenerateJSDoc | X
10  const login = async () => {
11    const res = await fetch("http://localhost:3000/api/auth/login", {
12      method: "POST",
13      headers: { "Content-Type": "application/json" },
14      body: JSON.stringify({ email, password }),
15    });
16
17    const data = await res.json();
18
19    if (!res.ok) {
20      alert(data.message);
21      return;
22    }
23
24    localStorage.setItem("token", data.token);
25    document.cookie = `tokens${data.token}; path=/`;
26    document.cookie = `role${data.user.role}; path=/`;
27
28    if (data.user.role === "ADMIN") {
29      window.location.href = "/admin/dashboard";
30    } else {
31      window.location.href = "/user/dashboard";
32    }
33
34    return (
35      <div className="p-10 max-w-md mx-auto">
36        <h1 className="text-xl font-bold mb-4">Login</h1>
37        <input className="border p-2 w-full mb-2" placeholder="Email" onChange={(e) => setEmail(e.target.value)} />
38        <input type="password" className="border p-2 w-full mb-4" placeholder="Password" onChange={(e) => setPassword(e.target.value)} />
39        <button onClick={login} className="bg-green-500 text-white px-4 py-2 rounded">Login</button>
40      </div>
41    );
42  };

```

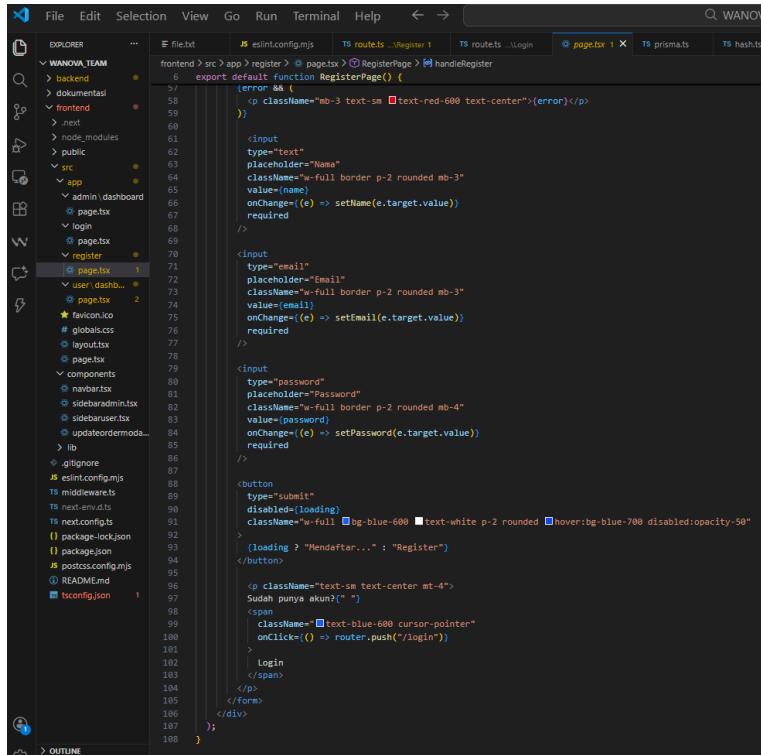


Kode ini adalah halaman login berbasis React yang memungkinkan user memasukkan email dan password, lalu mengirimkannya ke API /api/auth/login menggunakan metode POST. Jika login gagal, pesan error akan ditampilkan, dan jika berhasil, token JWT disimpan di localStorage serta cookie, kemudian sistem mengecek 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.

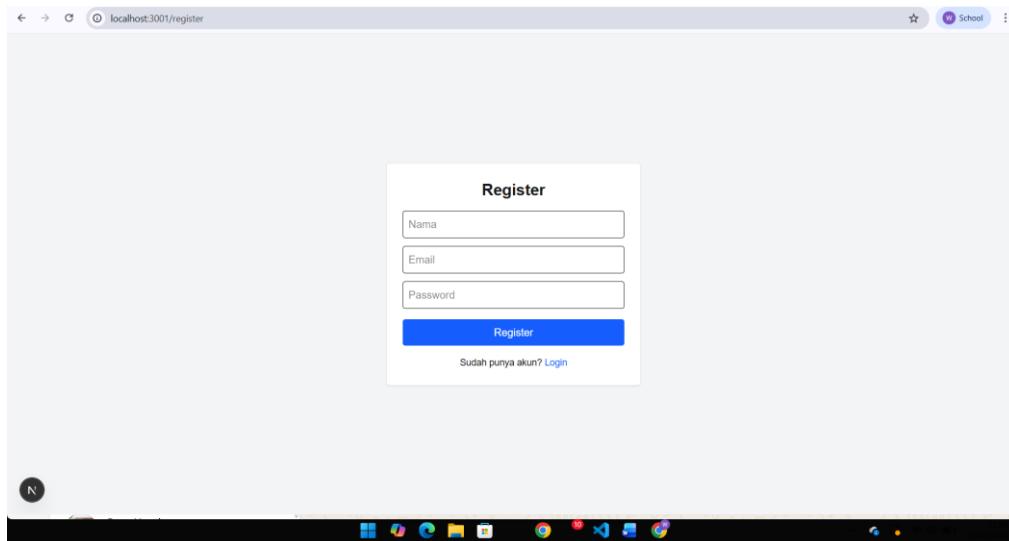
### 13. Frontend/register/page.tsx



```
1 "use client";
2
3 import { useState } from "react";
4 import { useRouter } from "next/navigation";
5
6 Windsurf Refactor | Explain | Generate JSDoc | X
7 export default function RegisterPage() {
8     const router = useRouter();
9
10    const [name, setName] = useState("");
11    const [email, setEmail] = useState("");
12    const [password, setPassword] = useState("");
13    const [loading, setLoading] = useState(false);
14    const [error, setError] = useState("");
15
16    const handleRegister = async (e: React.FormEvent) => {
17        e.preventDefault();
18        setLoading(true);
19        setError("");
20
21        try {
22            const res = await fetch("http://localhost:3000/api/auth/register", {
23                method: "POST",
24                headers: {
25                    "Content-Type": "application/json",
26                },
27                body: JSON.stringify({
28                    name,
29                    email,
30                    password,
31                }),
32            });
33
34            const data = await res.json();
35
36            if (!res.ok) {
37                setError(data.message || "Registrasi gagal");
38                return;
39            }
40
41            alert("Registrasi berhasil, silakan login");
42            router.push("/");
43        } catch (err) {
44            setError("Terjadi kesalahan server");
45        } finally {
46            setLoading(false);
47        }
48
49        return (
50            <div className="min-h-screen flex items-center justify-center bg-gray-100">
51                <form>
52                    <input type="submit" value="Register" onClick={handleRegister} className="bg-white p-6 rounded shadow w-96" />
53            </div>
54        );
55    }
56
57    return (
58        <div className="mb-3 text-sm text-red-600 text-center">{error}</div>
59    );
60
61    <input type="text" placeholder="Name" className="w-full border p-2 rounded mb-3" value={name} onChange={(e) => setName(e.target.value)} required />
62
63    <input type="email" placeholder="Email" className="w-full border p-2 rounded mb-3" value={email} onChange={(e) => setEmail(e.target.value)} required />
64
65    <input type="password" placeholder="Password" className="w-full border p-2 rounded mb-4" value={password} onChange={(e) => setPassword(e.target.value)} required />
66
67    <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>
68
69    <p className="text-sm text-center mt-4">Sudah punya akun? <a href="#">Login</a></p>
70
71    </div>
72
73    </div>
74
75    </div>
76
77    </div>
78
79    </div>
80
81    </div>
82
83    </div>
84
85    </div>
86
87    </div>
88
89    </div>
90
91    </div>
92
93    </div>
94
95    </div>
96
97    </div>
98
99    </div>
100
101
102    <Login>
103        <span>
104            <span>
105                <span>
106                    <span>
107                </span>
108            </span>
109        </span>
110    </Login>
111
112    </div>
113
114    </div>
115
116    </div>
117
118    </div>
119
120    </div>
121
122    </div>
123
124    </div>
125
126    </div>
127
128    </div>
129
130    </div>
131
132    </div>
133
134    </div>
135
136    </div>
137
138    </div>
139
140    </div>
141
142    </div>
143
144    </div>
145
146    </div>
147
148    </div>
149
150    </div>
151
152    </div>
153
154    </div>
155
156    </div>
157
158    </div>
159
160    </div>
161
162    </div>
163
164    </div>
165
166    </div>
167
168    </div>
169
170    </div>
171
172    </div>
173
174    </div>
175
176    </div>
177
178    </div>
179
180    </div>
181
182    </div>
183
184    </div>
185
186    </div>
187
188    </div>
189
190    </div>
191
192    </div>
193
194    </div>
195
196    </div>
197
198    </div>
199
200    </div>
201
202    </div>
203
204    </div>
205
206    </div>
207
208    </div>
209
210    </div>
211
212    </div>
213
214    </div>
215
216    </div>
217
218    </div>
219
220    </div>
221
222    </div>
223
224    </div>
225
226    </div>
227
228    </div>
229
230    </div>
231
232    </div>
233
234    </div>
235
236    </div>
237
238    </div>
239
240    </div>
241
242    </div>
243
244    </div>
245
246    </div>
247
248    </div>
249
250    </div>
251
252    </div>
253
254    </div>
255
256    </div>
257
258    </div>
259
260    </div>
261
262    </div>
263
264    </div>
265
266    </div>
267
268    </div>
269
270    </div>
271
272    </div>
273
274    </div>
275
276    </div>
277
278    </div>
279
280    </div>
281
282    </div>
283
284    </div>
285
286    </div>
287
288    </div>
289
290    </div>
291
292    </div>
293
294    </div>
295
296    </div>
297
298    </div>
299
300    </div>
301
302    </div>
303
304    </div>
305
306    </div>
307
308    </div>
309
310    </div>
311
312    </div>
313
314    </div>
315
316    </div>
317
318    </div>
319
320    </div>
321
322    </div>
323
324    </div>
325
326    </div>
327
328    </div>
329
330    </div>
331
332    </div>
333
334    </div>
335
336    </div>
337
338    </div>
339
340    </div>
341
342    </div>
343
344    </div>
345
346    </div>
347
348    </div>
349
350    </div>
351
352    </div>
353
354    </div>
355
356    </div>
357
358    </div>
359
360    </div>
361
362    </div>
363
364    </div>
365
366    </div>
367
368    </div>
369
370    </div>
371
372    </div>
373
374    </div>
375
376    </div>
377
378    </div>
379
380    </div>
381
382    </div>
383
384    </div>
385
386    </div>
387
388    </div>
389
390    </div>
391
392    </div>
393
394    </div>
395
396    </div>
397
398    </div>
399
400    </div>
401
402    </div>
403
404    </div>
405
406    </div>
407
408    </div>
409
410    </div>
411
412    </div>
413
414    </div>
415
416    </div>
417
418    </div>
419
420    </div>
421
422    </div>
423
424    </div>
425
426    </div>
427
428    </div>
429
430    </div>
431
432    </div>
433
434    </div>
435
436    </div>
437
438    </div>
439
440    </div>
441
442    </div>
443
444    </div>
445
446    </div>
447
448    </div>
449
450    </div>
451
452    </div>
453
454    </div>
455
456    </div>
457
458    </div>
459
460    </div>
461
462    </div>
463
464    </div>
465
466    </div>
467
468    </div>
469
470    </div>
471
472    </div>
473
474    </div>
475
476    </div>
477
478    </div>
479
480    </div>
481
482    </div>
483
484    </div>
485
486    </div>
487
488    </div>
489
490    </div>
491
492    </div>
493
494    </div>
495
496    </div>
497
498    </div>
499
500    </div>
501
502    </div>
```



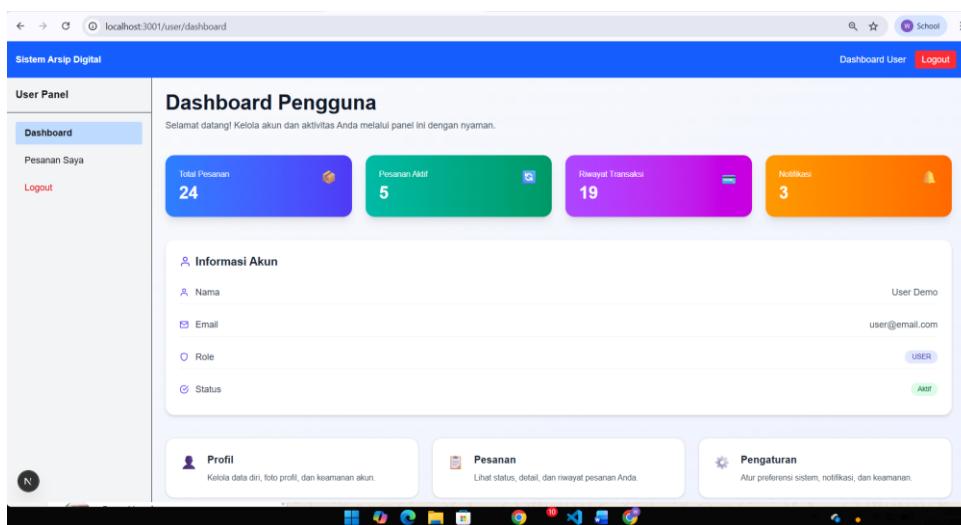
```
1 "use client";
2
3 import { useState } from "react";
4 import { useRouter } from "next/navigation";
5
6 Windsurf Refactor | Explain | Generate JSDoc | X
7 export default function RegisterPage() {
8     const router = useRouter();
9
10    const [name, setName] = useState("");
11    const [email, setEmail] = useState("");
12    const [password, setPassword] = useState("");
13    const [loading, setLoading] = useState(false);
14    const [error, setError] = useState("");
15
16    const handleRegister = async (e: React.FormEvent) => {
17        e.preventDefault();
18        setLoading(true);
19        setError("");
20
21        try {
22            const res = await fetch("http://localhost:3000/api/auth/register", {
23                method: "POST",
24                headers: {
25                    "Content-Type": "application/json",
26                },
27                body: JSON.stringify({
28                    name,
29                    email,
30                    password,
31                }),
32            });
33
34            const data = await res.json();
35
36            if (!res.ok) {
37                setError(data.message || "Registrasi gagal");
38                return;
39            }
40
41            alert("Registrasi berhasil, silakan login");
42            router.push("/");
43        } catch (err) {
44            setError("Terjadi kesalahan server");
45        } finally {
46            setLoading(false);
47        }
48
49        return (
50            <div className="min-h-screen flex items-center justify-center bg-gray-100">
51                <form>
52                    <input type="submit" value="Register" onClick={handleRegister} className="bg-white p-6 rounded shadow w-96" />
53            </div>
54        );
55    }
56
57    return (
58        <div className="mb-3 text-sm text-red-600 text-center">{error}</div>
59    );
60
61    <input type="text" placeholder="Name" className="w-full border p-2 rounded mb-3" value={name} onChange={(e) => setName(e.target.value)} required />
62
63    <input type="email" placeholder="Email" className="w-full border p-2 rounded mb-3" value={email} onChange={(e) => setEmail(e.target.value)} required />
64
65    <input type="password" placeholder="Password" className="w-full border p-2 rounded mb-4" value={password} onChange={(e) => setPassword(e.target.value)} required />
66
67    <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>
68
69    <p className="text-sm text-center mt-4">Sudah punya akun? <a href="#">Login</a></p>
70
71    </div>
72
73    </div>
74
75    </div>
76
77    </div>
78
79    </div>
80
81    </div>
82
83    </div>
84
85    </div>
86
87    </div>
88
89    </div>
90
91    </div>
92
93    </div>
94
95    </div>
96
97    </div>
98
99    </div>
100
101
102    <Login>
103        <span>
104            <span>
105                <span>
106                    <span>
107                </span>
108            </span>
109        </span>
110    </Login>
111
112    </div>
113
114    </div>
115
116    </div>
117
118    </div>
119
120    </div>
121
122    </div>
123
124    </div>
125
126    </div>
127
128    </div>
129
130    </div>
131
132    </div>
133
134    </div>
135
136    </div>
137
138    </div>
139
140    </div>
141
142    </div>
143
144    </div>
145
146    </div>
147
148    </div>
149
150    </div>
151
152    </div>
153
154    </div>
155
156    </div>
157
158    </div>
159
160    </div>
161
162    </div>
163
164    </div>
165
166    </div>
167
168    </div>
169
170    </div>
171
172    </div>
173
174    </div>
175
176    </div>
177
178    </div>
179
180    </div>
181
182    </div>
183
184    </div>
185
186    </div>
187
188    </div>
189
190    </div>
191
192    </div>
193
194    </div>
195
196    </div>
197
198    </div>
199
200    </div>
201
202    </div>
203
204    </div>
205
206    </div>
207
208    </div>
209
210    </div>
211
212    </div>
213
214    </div>
215
216    </div>
217
218    </div>
219
220    </div>
221
222    </div>
223
224    </div>
225
226    </div>
227
228    </div>
229
230    </div>
231
232    </div>
233
234    </div>
235
236    </div>
237
238    </div>
239
240    </div>
241
242    </div>
243
244    </div>
245
246    </div>
247
248    </div>
249
250    </div>
251
252    </div>
253
254    </div>
255
256    </div>
257
258    </div>
259
260    </div>
261
262    </div>
263
264    </div>
265
266    </div>
267
268    </div>
269
270    </div>
271
272    </div>
273
274    </div>
275
276    </div>
277
278    </div>
279
280    </div>
281
282    </div>
283
284    </div>
285
286    </div>
287
288    </div>
289
290    </div>
291
292    </div>
293
294    </div>
295
296    </div>
297
298    </div>
299
300    </div>
301
302    </div>
303
304    </div>
305
306    </div>
307
308    </div>
309
310    </div>
311
312    </div>
313
314    </div>
315
316    </div>
317
318    </div>
319
320    </div>
321
322    </div>
323
324    </div>
325
326    </div>
327
328    </div>
329
330    </div>
331
332    </div>
333
334    </div>
335
336    </div>
337
338    </div>
339
340    </div>
341
342    </div>
343
344    </div>
345
346    </div>
347
348    </div>
349
350    </div>
351
352    </div>
353
354    </div>
355
356    </div>
357
358    </div>
359
360    </div>
361
362    </div>
363
364    </div>
365
366    </div>
367
368    </div>
369
370    </div>
371
372    </div>
373
374    </div>
375
376    </div>
377
378    </div>
379
380    </div>
381
382    </div>
383
384    </div>
385
386    </div>
387
388    </div>
389
390    </div>
391
392    </div>
393
394    </div>
395
396    </div>
397
398    </div>
399
400    </div>
401
402    </div>
403
404    </div>
405
406    </div>
407
408    </div>
409
410    </div>
411
412    </div>
413
414    </div>
415
416    </div>
417
418    </div>
419
420    </div>
421
422    </div>
423
424    </div>
425
426    </div>
427
428    </div>
429
430    </div>
431
432    </div>
433
434    </div>
435
436    </div>
437
438    </div>
439
440    </div>
441
442    </div>
443
444    </div>
445
446    </div>
447
448    </div>
449
450    </div>
451
452    </div>
453
454    </div>
455
456    </div>
457
458    </div>
459
460    </div>
461
462    </div>
463
464    </div>
465
466    </div>
467
468    </div>
469
470    </div>
471
472    </div>
473
474    </div>
475
476    </div>
477
478    </div>
479
480    </div>
481
482    </div>
483
484    </div>
485
486    </div>
487
488    </div>
489
490    </div>
491
492    </div>
493
494    </div>
495
496    </div>
497
498    </div>
499
500    </div>
501
502    </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

```
    },
    ].map((card, idx) => (
      <div
        key={idx}
        className={ `bg-white p-6 rounded-2xl shadow-sm border border-gray-200 cursor-pointer group
          transition-all duration-300 ease-out transform hover:-translate-y-1 hover:shadow-lg ${card.hoverBg}` }
      >
        <div className="flex items-start gap-4">
          <span className="text-2xl mt-1">{card.icon}</span>
          <div>
            <h4
              className={`font-semibold text-lg text-gray-800 mb-2 group-hover:text-${card.color}-700 transition-colors`}
            >
              {card.title}
            </h4>
            <p className="text-sm text-gray-600">{card.desc}</p>
          </div>
        </div>
      </div>
    )))
  </div>
</in>
```



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.