JOBSHEET PRAKTIKUM

Mata Kuliah : Pemograman Framework

Program Studi : D4 (Sarjana Terapan)

Topik Praktikum : Catch-All Routing, Optional Catch-All, Linking & Navigating pada Next.js Pages Router

Alokasi Waktu : 3 × 50 menit

A. **Tujuan Praktikum**

Setelah menyelesaikan praktikum ini, mahasiswa mampu:

1. Membuat catch-all route untuk menangkap banyak segmen URL.
2. Menggunakan optional catch-all route agar halaman tetap dapat diakses tanpa parameter.
3. Mengambil parameter URL berbentuk array menggunakan useRouter.
4. Menerapkan navigasi antar halaman menggunakan Link.
5. Melakukan navigasi imperatif menggunakan router.push().
6. Mengimplementasikan redirect sederhana berbasis kondisi (simulasi login).

B. **Dasar Teori Singkat**

1. **Segment & Slug pada URL**

- URL dapat terdiri dari beberapa segmen, contoh:

  /product/clothes/tops/t-shirt

- Setiap bagian dipisahkan oleh / dan disebut segmen.

2. Catch-All Route

Next.js memungkinkan menangkap semua segmen URL menggunakan:

- [...slug].js

Hasil parameter akan berbentuk array.

3. Optional Catch-All Route

Agar halaman tetap bisa diakses meskipun tanpa parameter:

- [[...slug]].js

4. Navigasi di Next.js

- Deklaratif: Link dari next/link
- Imperatif: router.push() dari next/router

C. Alat dan Bahan

Perangkat Lunak

- Node.js (LTS)
- NPM
- Visual Studio Code
- Browser (Chrome / Firefox)

Prasyarat

- Project Next.js Pages Router sudah tersedia
- Server dapat dijalankan (npm run dev)

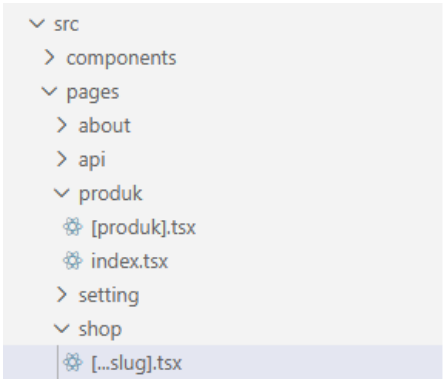
D. Langkah Kerja Praktikum

Langkah 1 – Menjalankan Project

1. Buka folder project Next.js.
2. Jalankan server:
3. npm run dev
4. Akses:
5. http://localhost:3000

Langkah 2 – Membuat Catch-All Route

1. Masuk ke folder pages.
2. Buat folder shop dan file […slug].tsx:



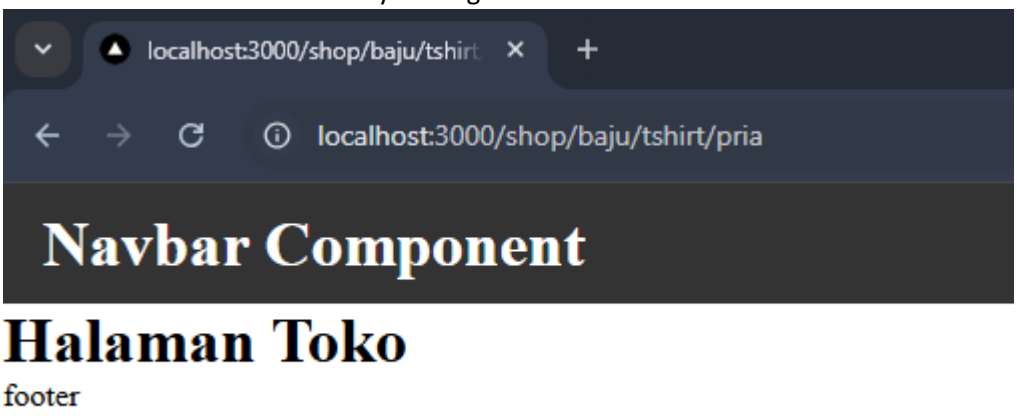
3. Modifikasi Isi file […slug].tsx dengan kode berikut:

praktikum-LinkNavigation > my-app > src > pages > shop > [...slug].tsx > .

```tsx
import { useRouter } from "next/router";


Windsurf: Refactor | Explain | Generate JSDoc | X
const halamanToko = () => {
    const Router = useRouter();
    console.log(Router);
    // const { query } = useRouter();
  return (
  <div>
    <h1>Halaman Toko</h1>

</div>
    );
};

export default halamanToko;

```

Cek menggunakan console.log apakah nilai segment berhasil didapat

- Jalankan browser dan ketik urlnya sebagai berikut



- Cek Vscode jika pada console.log dapat menampilkan nilai querynya berarti berhasil

- Modifikasi [...slug].tsx untuk menampilkan nilai query

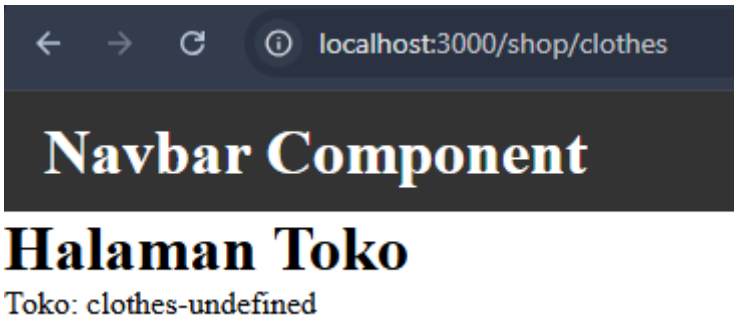
praktikum-LinkNavigation > my-app > src > pages > shop > [...slug].tsx > ...

```tsx
import { useRouter } from "next/router";


Windsurf: Refactor | Explain | Generate JSDoc |
const halamanToko = () => {
   // const Router = useRouter();
    //console.log(Router);
    const { query } = useRouter();
  return (
  <div>
    <h1>Halaman Toko</h1>
    <p>Toko: {`${query.slug && query.slug[0]+"-"+ query.slug[1]}`}</p>  {/* menggunakan backtick bukan petik satu*/}

  </div>
    );
};

export default halamanToko;
```

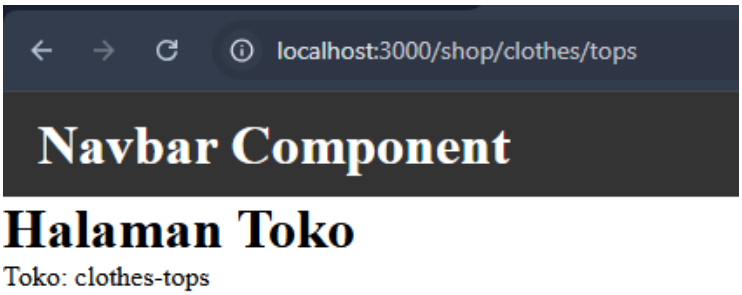
Langkah 3 – Pengujian Catch-All Route

Akses URL berikut di browser:

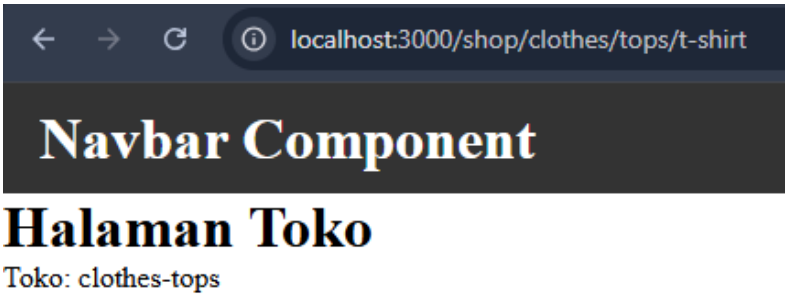
/shop/clothes



/shop/clothes/tops



/shop/clothes/tops/t-shirt



Jika dilihat ada yang terbaca undifined dan ada yang tidak terbaca ini dikarena segmennya dibatasi Cuma array[0] dan array[1]. Solusinya bagaimana ?

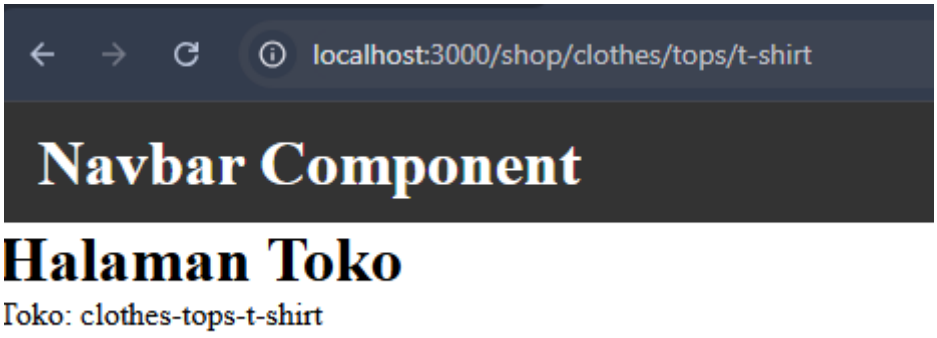
Modifikasi […slug].tsx menjadi berikut

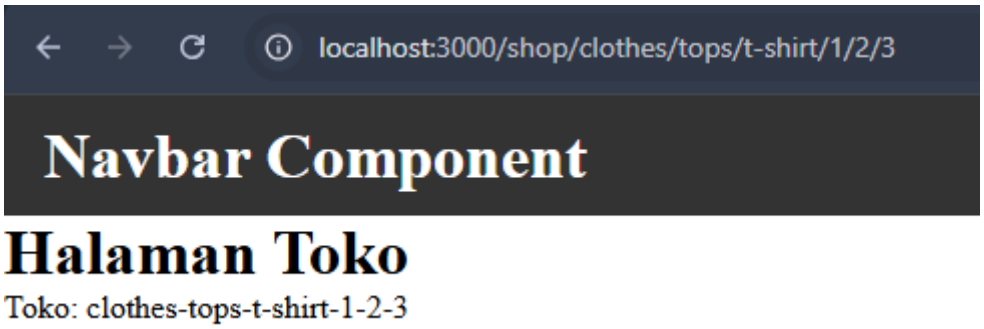
[…slug].tsx

praktikum-LinkNavigation > my-app > src > pages > shop > […slug].tsx > ...

```tsx
import { useRouter } from "next/router";


Windsurf: Refactor | Explain | Generate JSDoc |
const halamanToko = () => {
   // const Router = useRouter();
    //console.log(Router);
    const { query } = useRouter();
  return (
  <div>
     <h1>Halaman Toko</h1>
     {/* <p>Toko: {`${query.slug && query.slug[0]+"-"+ query.slug[1]}`}</p>  menggunakan backtick bukan petik satu */}
      <p>
          Toko: {Array.isArray(query.slug) ? query.slug.join("-") : query.slug}
      </p>

  </div>
    );
};

export default halamanToko;

```

Jalankan browser : **Berapapun banyaknya seqment tetap terbaca**





Untuk saat ini gunakan

praktikum-LinkNavigation > my-app > src > pages > shop > [...slug].tsx > ...

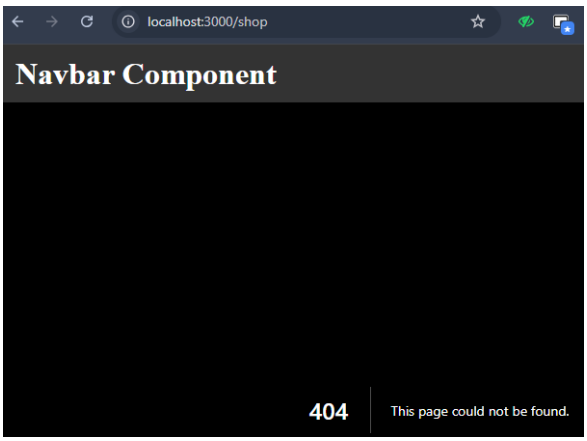
```jsx
import { useRouter } from "next/router";


Windsurf: Refactor | Explain | Generate JSDoc | X
const halamanToko = () => {
   // const Router = useRouter();
    //console.log(Router);
    const { query } = useRouter();
  return (
  <div>
     <h1>Halaman Toko</h1>
     <p>Toko: {`${query.slug && query.slug[0]+"-"+ query.slug[1]}`}</p>  {/* menggunakan backtick bukan petik satu*/}

  </div>
    );
};

export default halamanToko;
```

Perhatikan bahwa:

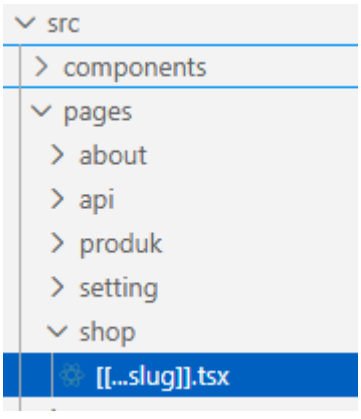
- slug berbentuk array
- Isi halaman berubah sesuai URL

Langkah 4 – Optional Catch-All Route

1. Jika menggunakan [...slug].js maka ketika mengakses shop akan terjadi error

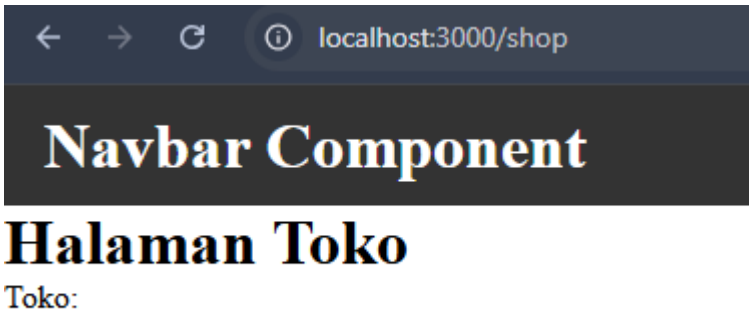


2. Solusinya dengan Rename file: [...slug].js → [[...slug]].js



3. Sekarang akses:

/shop



4. Halaman dapat diakses meskipun tanpa parameter.

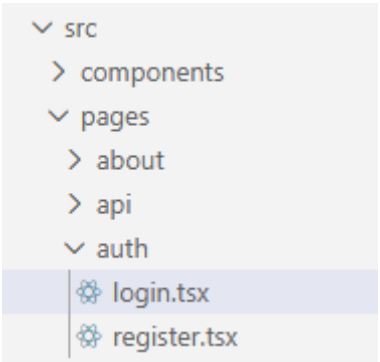
Langkah 5 – Validasi Parameter

Tambahkan validasi agar tidak error saat slug kosong:

Kategori: {slug ? slug[0] : "Semua Kategori"}

Langkah 6 – Membuat Halaman Login & Register

1. Buat folder:
2. pages/auth
3. Buat file:
    - login.jsx
    - register.jsx



Modifikasi login.jsx:

```
praktikum-LinkNavigation > my-app > src > pages > auth > login.tsx > halamanLogin
import Link from "next/link";

Windsurf: Refactor | Explain | Generate JSDoc | X
const halamanLogin = () => {
  return (
    <div>
      <h1>Halaman Login</h1>
        <Link href="/auth/register">Ke Halaman Register</Link>
    </div>
  );
};

export default halamanLogin;
```

Modifikasi register.jsx:

```
praktikum-LinkNavigation > my-app > src > pages > auth > register.tsx > halamanRegister
import Link from "next/link";

Windsurf: Refactor | Explain | Generate JSDoc | X
const halamanRegister = () => {
  return (
    <div>
      <h1>Halaman Register</h1>
      <Link href="/auth/login">Ke Halaman Login</Link>
    </div>
  );
};

export default halamanRegister;
```

Langkah 7 – Navigasi Imperatif (router.push)

1. Tambahkan button login:

```
praktikum-LinkNavigation > my-app > src > pages > auth > login.tsx > halamanLogin
import Link from "next/link";
import { useRouter } from "next/router";

Windsurf: Refactor | Explain | Generate JSDoc | X
const halamanLogin = () => {
    const {push} = useRouter();
    Windsurf: Refactor | Explain | Generate JSDoc | X
    const handlerLogin = () => {
        // logic login disini
        push('/produk');
    }
  return (
    <div>

      <h1>Halaman Login</h1>
        <button onClick={handlerLogin}>Login</button> <br />
        <button onClick={() => push('/produk')}>Login</button> <br />
        <button onClick={() => handlerLogin()}>Login</button> <br />
        <Link href="/auth/register">Ke Halaman Register</Link>
    </div>
  );
};

export default halamanLogin;

```

Kode	Cara Kerja	Kapan Dipanggil	Kelebihan	Kekurangan	Rekomendasi
onClick={handlerLogin}	Mengirim referensi fungsi ke event handler	Saat tombol diklik	Paling bersih, efisien, dan best practice	Tidak bisa kirim parameter langsung	**Sangat direkomendasikan**
onClick={() => push('/produk')}	Arrow function memanggil fungsi di dalamnya	Saat tombol diklik	Praktis untuk aksi sederhana	Kurang reusable jika logika bertambah	Direkomendasikan untuk navigasi sederhana
onClick={() => handlerLogin()}	Arrow function membungkus pemanggilan fungsi	Saat tombol diklik	Fleksibel untuk kirim parameter	Redundant jika tanpa parameter	Direkomendasikan hanya jika perlu argumen
onClick={handlerLogin()}	Fungsi langsung dieksekusi	Saat render	Tidak ada	Bug: fungsi tidak menunggu klik	**Tidak direkomendasikan**

Saat ini gunakan **onClick={() => handlerLogin()}**

2. Klik tombol dan perhatikan perpindahan halaman tanpa reload.
3. Jika di klik button login maka akan menuju /produk

Note :

- Pastikan code untuk redirect tidak aktif , jika aktif maka ketika masuk ke produk akan lansung redirect ke login

6.praktikum-ApiRoutes > my-app > src > pages > produk > index.tsx > ...

```tsx
import { useRouter } from "next/router";
import { useEffect, useState } from "react";

Windsurf: Refactor | Explain | Generate JSDoc | X
const produk = () => {
  // const [isLogin, setIsLogin] = useState(false);
  // const { push } = useRouter();
  // useEffect(() => {
  //   if (!isLogin) {
  //     push("/auth/login");
  //   }
  //   },[]);
  return (
  <div>Produk User Page</div>
  );
};

export default produk;

```

Langkah 8 – Simulasi Redirect (Belum Login)

1. Di halaman product, pada index.tsx tambahkan beberapa code berikut:

praktikum-LinkNavigation > my-app > src > pages > produk > index.tsx > ...

```tsx
import { useRouter } from "next/router";
import { useEffect, useState } from "react";

Windsurf: Refactor | Explain | Generate JSDoc | X
const produk = () => {
  const [isLogin, setIsLogin] = useState(false);
  const { push } = useRouter();
  useEffect(() => {
    if (!isLogin) {
      push("/auth/login");
    }
    },[]);
  return (
  <div>Produk User Page</div>
  );
};

export default produk;
Windsurf: Generate (Ctrl+I)

```

2. Jika Akses /product → otomatis diarahkan ke login.

E. Tugas Praktikum

Tugas 1 (Wajib)

- Buat catch-all route:
- /category/[...slug].js
- Tampilkan seluruh parameter URL dalam bentuk list.

Tugas 2 (Wajib)

- Buat navigasi:
    - Login → Product (imperatif)

- Login ↔ Register (Link)

Tugas 3 (Pengayaan)

- Terapkan redirect otomatis ke login jika user belum login.

F. Pertanyaan Evaluasi

1. Apa perbedaan [id].js dan [...slug].js?
2. Mengapa slug berbentuk array?
3. Kapan sebaiknya menggunakan Link dan router.push()?
4. Mengapa navigasi Next.js tidak me-refresh halaman?