JOBSHEET 03

ROUTING & LAYOUTING PADA NEXT.JS

PEMOGRAMAN BERBASIS FRAMEWORK

REIKA AMALIA SYAHPUTRI

2341720173

TI-3E / 19

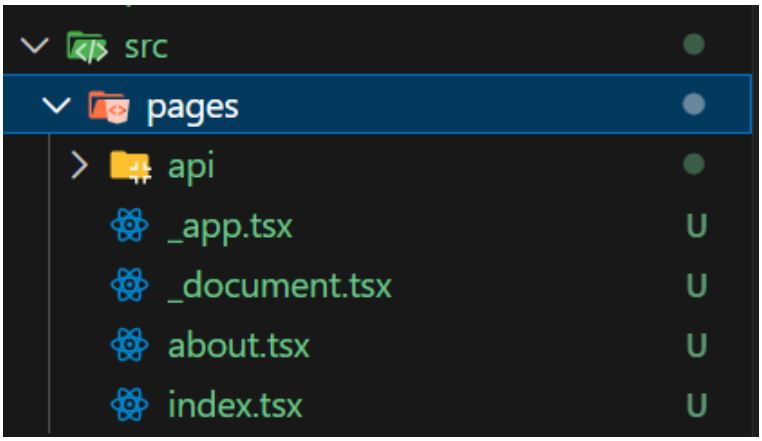
TEKNOLOGI INFORMASI

D. Langkah Praktikum

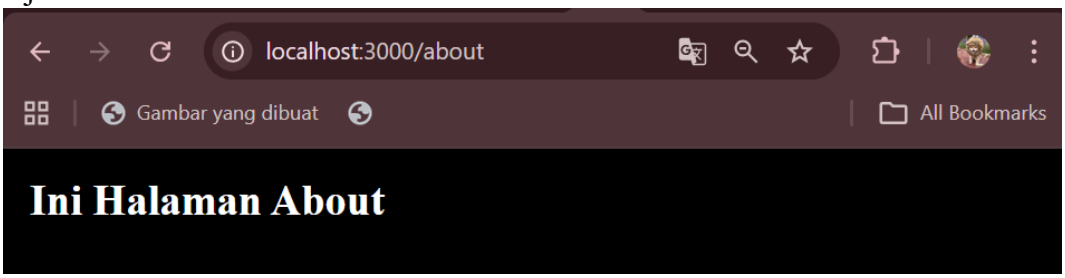
1. Routing Dasar (Static Routing)

a. Struktur Awal pages/
└── index.tsx

b. Tambahkan Halaman About



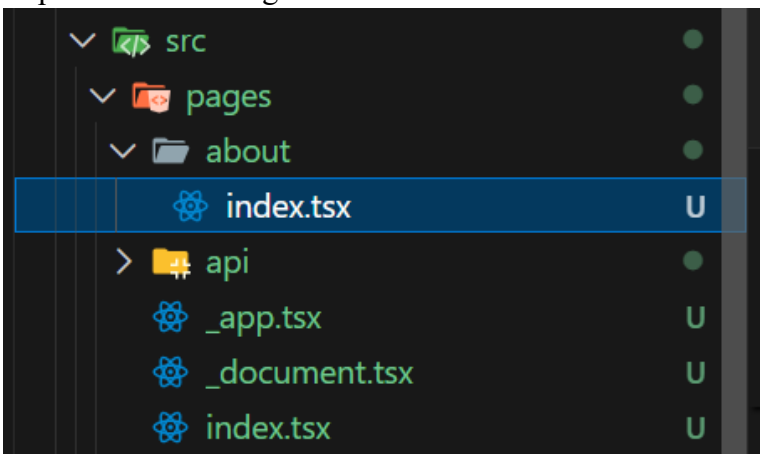
c. Uji di Browser



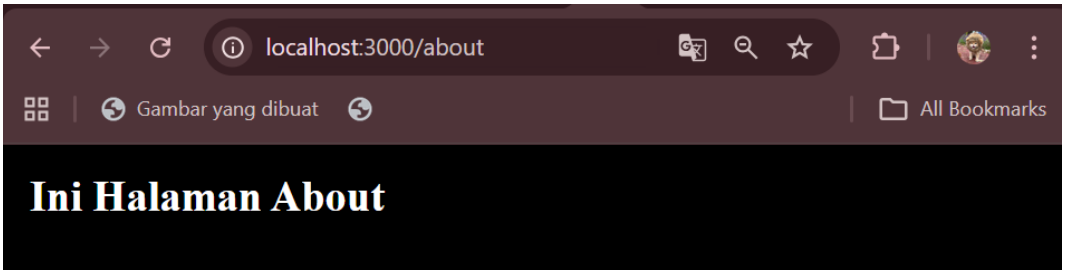
Catatan: Next.js otomatis membuat routing berdasarkan nama file tanpa konfigurasi tambahan

2. Routing Menggunakan Folder

a. Rapikan Struktur Pages



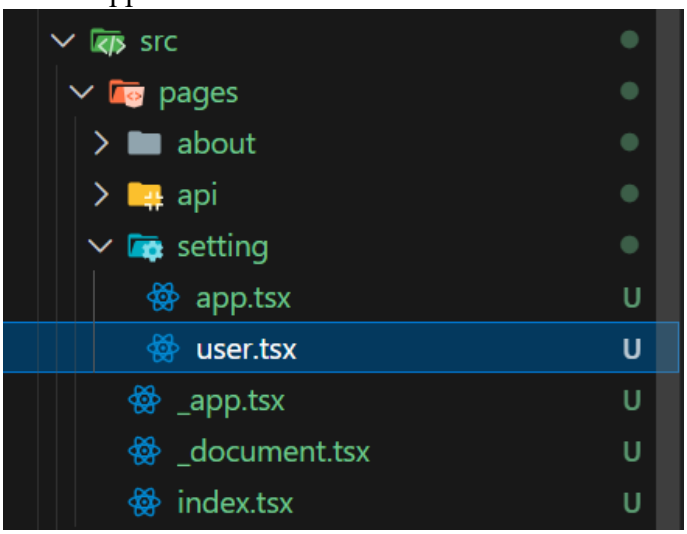
b. Akses dari halaman browser (tetap sama tetapi lebih rapi)



3. Nested Routing

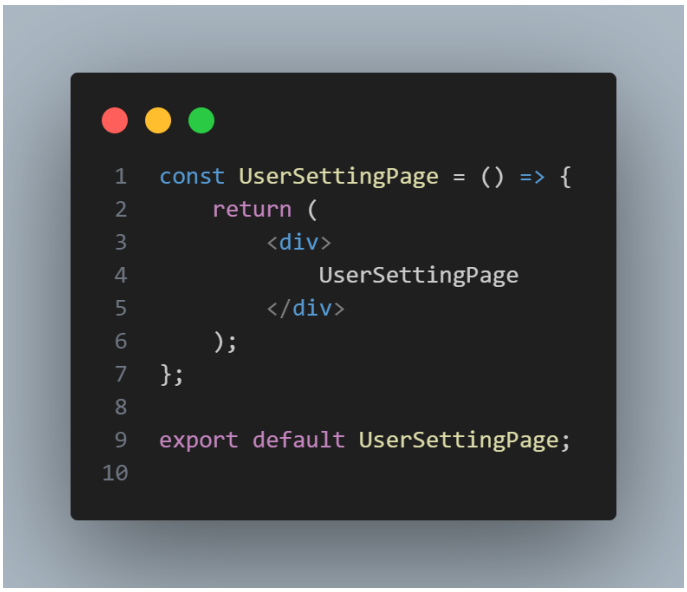
a. Buat Folder Setting pages/

```
└── setting/
    ├── user.tsx
    └── app.tsx
```



Modifikasi kodenya

- User.tsx



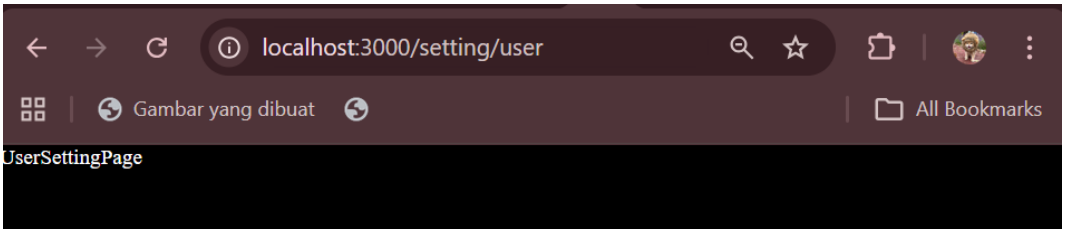
```tsx
const UserSettingPage = () => {
    return (
        <div>
            UserSettingPage
        </div>
    );
};

export default UserSettingPage;
```

- App.tsx

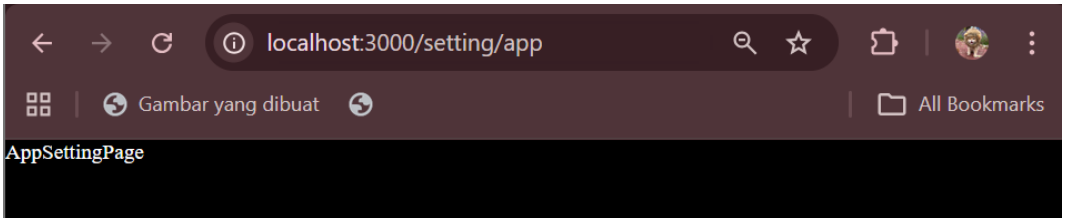
```tsx
const AppSettingPage = () => {
    return (
        <div>
            AppSettingPage
        </div>
    );
};

export default AppSettingPage;

```

Akses

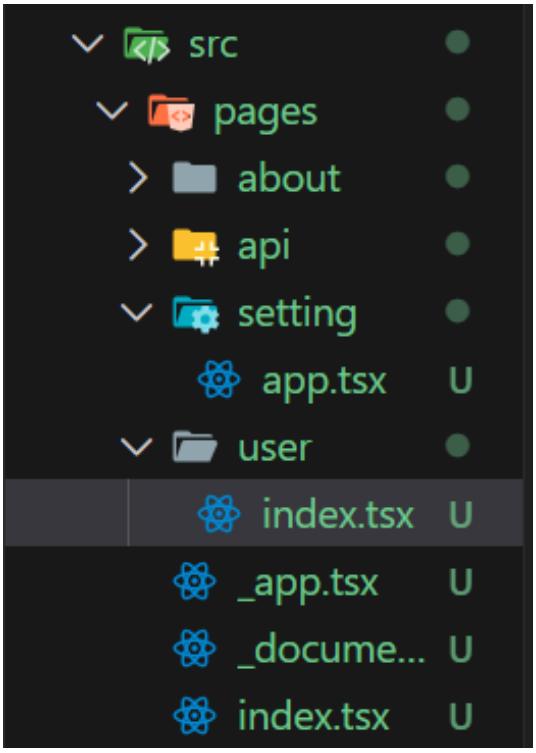
- /setting/user



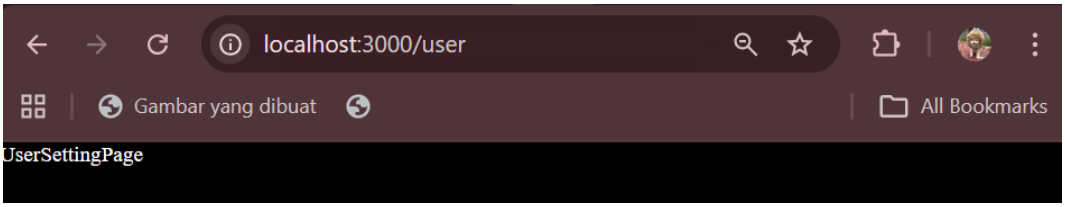
- /setting/app



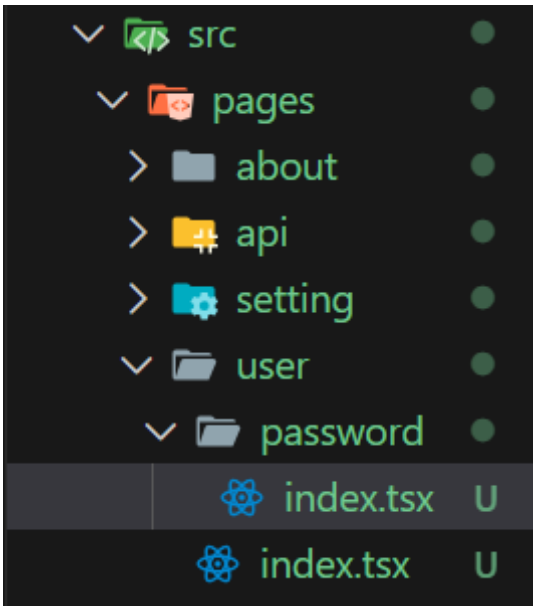
Modifikasi struktur folder pages dengan menambahkan folder user dan user.tsx pada setting dipindah ke folder user dan rubah file user.tsx menjadi index.tsx



Jalankan pada browser

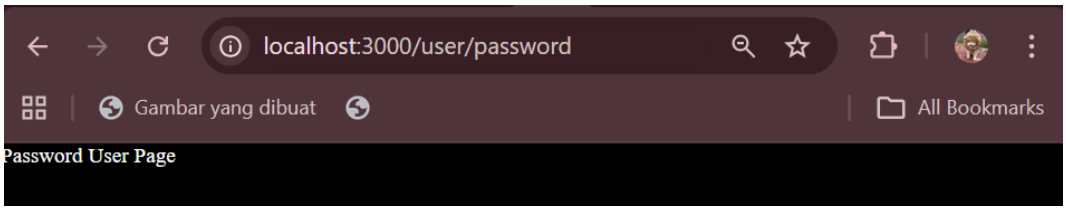


b. Nested Lebih Dalam



Akses:

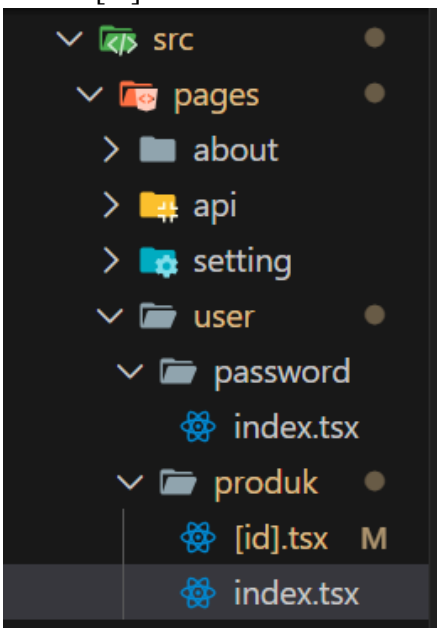
/setting/user/password



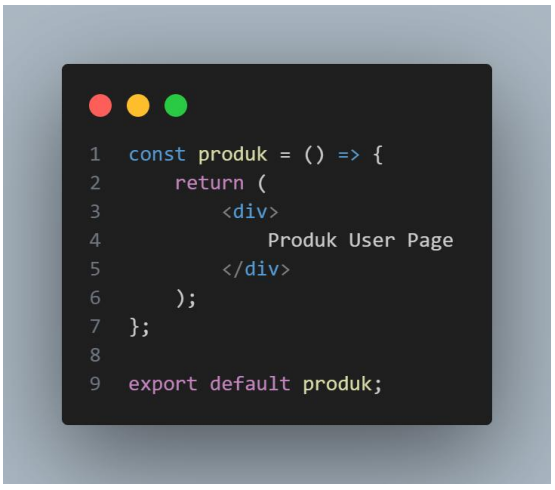
Keunggulan:
Tidak perlu konfigurasi manual seperti React Router atau Laravel Route.

4. Dynamic Routing
   a. Buat Halaman Produk

```
pages/
└── produk/
    ├── index.tsx
    └── [id].tsx
```

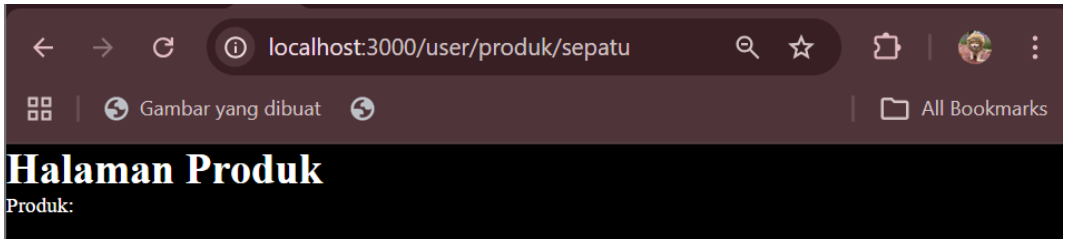


- Modifikasi index.tsx



```tsx
const produk = () => {
    return (
        <div>
            Produk User Page
        </div>
    );
};

export default produk;
```

- Modifikasi [id].tsx
  Buka browser http://localhost:3000/produk/sepatu tambahkan segment sepatu



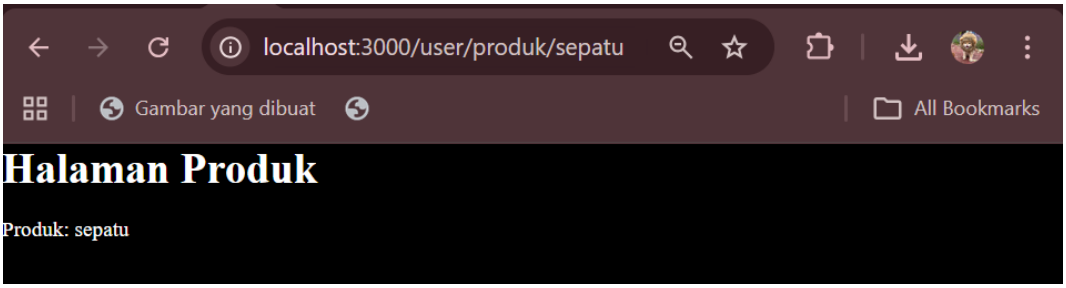
- Cek menggunakan console.log

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

const HalamanProduk = () => {
  const Router = useRouter();
  console.log(Router);

  return (
    <div>
      <h1>Halaman Produk</h1>
      <p>Produk:</p>
    </div>
  );
};

export default HalamanProduk;
```

- Modifikasi [id].tsx agar dapat mengambil nilai dari id

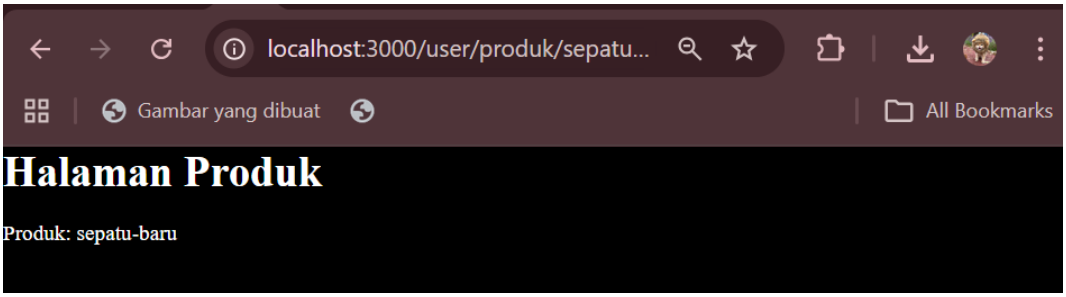
```jsx
import { useRouter } from "next/router";

const HalamanProduk = () => {
  // const Router = useRouter();
  // console.log(Router);
  const { query } = useRouter();

  return (
    <div>
      <h1>Halaman Produk</h1> <br />
      <p>Produk: {query.id}</p>
    </div>
  );
};

export default HalamanProduk;

```

- Buka browser

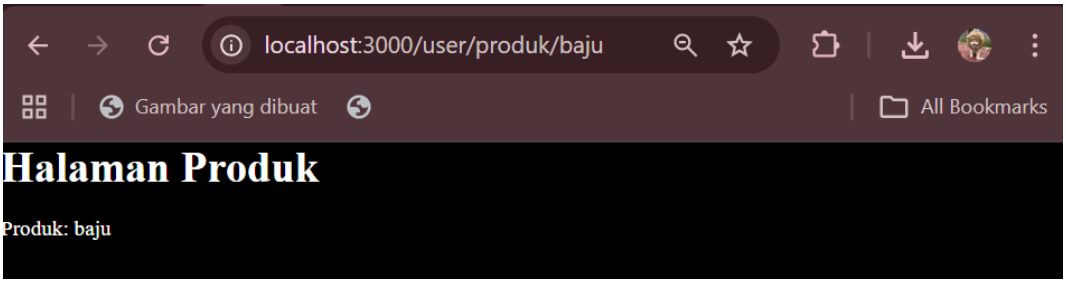


b. Uji di Browser

- /produk/sepatu-baru

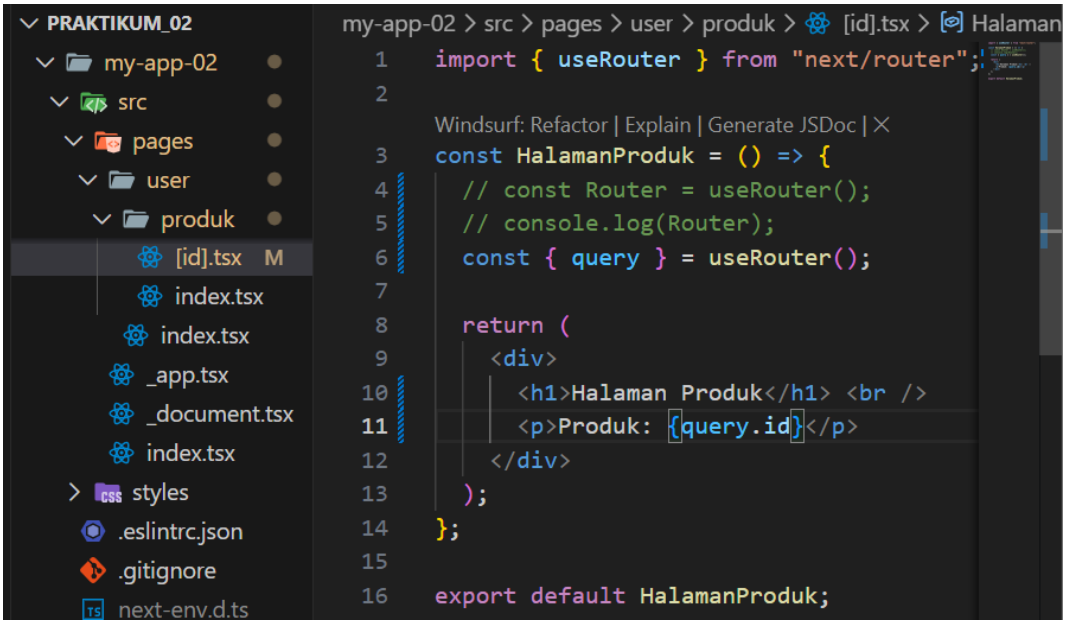


- /produk/baju



Catatan Penting:

Nama file di dalam [ ] akan menjadi parameter URL. Contoh



5. Membuat Komponen Navbar
   a. Struktur Komponen

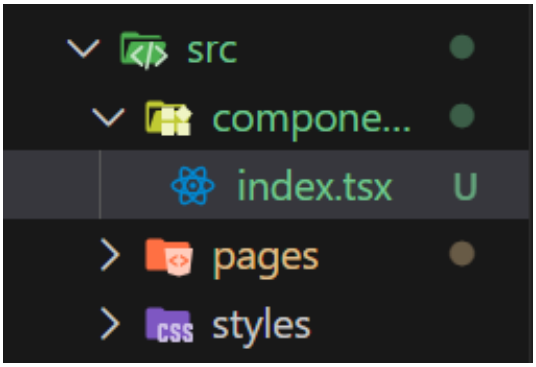
src/

└── components/

└── layouts/

└── Navbar/

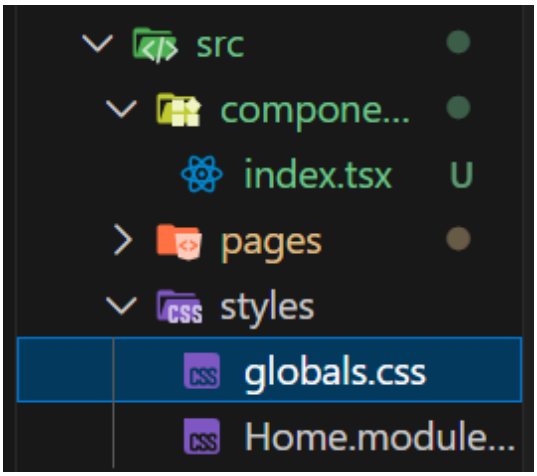
└── index.tsx



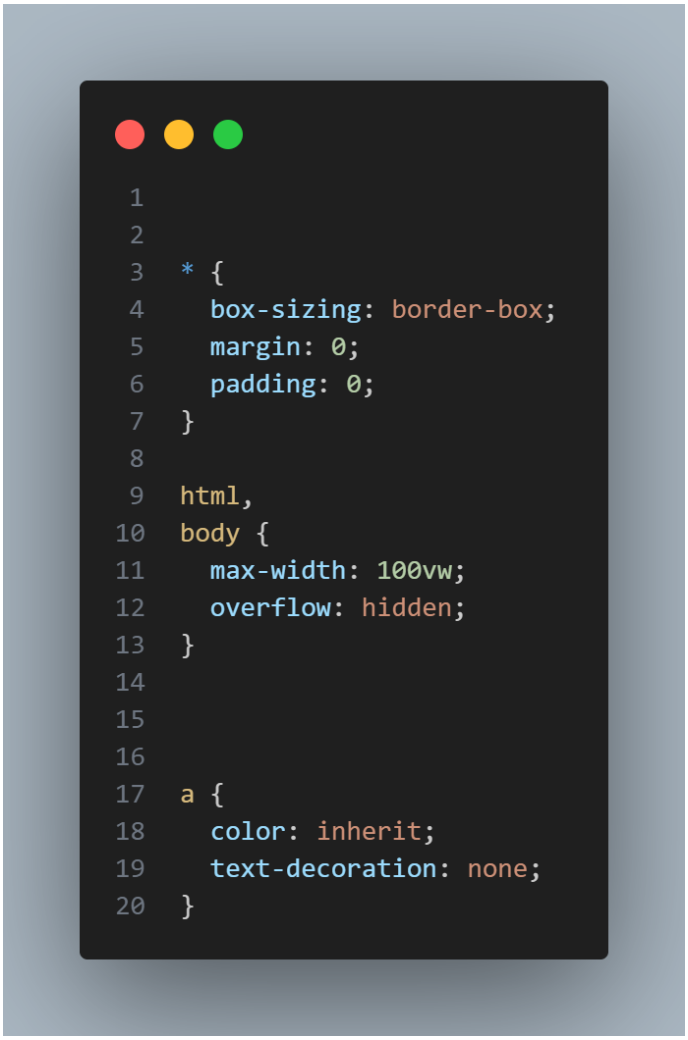
- Modifikasi index.tsx

```tsx
const Navbar = () => {
    return (
    <div className="">
        <div>navbar Component </div>
    </div>
    );
};

export default Navbar;
```

- Buka globals.css untuk nantinya digunakan pada style navbar



- Modifikasi globals.css



```
1
2
3   * {
4     box-sizing: border-box;
5     margin: 0;
6     padding: 0;
7   }
8
9   html,
10  body {
11    max-width: 100vw;
12    overflow: hidden;
13  }
14
15
16
17  a {
18    color: inherit;
19    text-decoration: none;
20  }
```

- Modifikasi index.tsx dengan menambahkan classname untuk style navbar

```
1  const Navbar = () => {
2      return (
3      <div className="navbar">
4          <div>navbar Component </div>
5      </div>
6      );
7  };
8
9  export default Navbar;
```

- Modifikasi globals.css

```
1
2
3   * {
4     /* box-sizing: border-box;
5     margin: 0;
6     padding: 0; */
7   }
8
9   html,
10  body {
11    /* max-width: 100vw;
12    overflow: hidden; */
13  }
14
15
16
17  a {
18    /* color: inherit;
19    text-decoration: none; */
20  }
21
22  .navbar {
23    width: 100%;
24    height: 60px;
25    background-color: #333;
26    color: white;
27    display: flex;
28    align-items: center;
29    padding: 0 20px;
30  }
31
```

- Modifikasi index.tsx pada folder pages

```tsx
// import Link from "next/link";
import Head from 'next/head'
import Image from "next/image"
import { Inter } from 'next/font/google'
import styles from '@/styles/Home.module.css'
import Navbar from '@/components/index'

const inter = Inter({ subsets: ['latin'] })

export default function Home() {
  return (
    <div>
      <Navbar />
      <h1>Praktikum Next.js Pages Router</h1> <br />
      <p>Mahasiswa D4 Pengembangan Web </p>
    </div>
  );
}

```

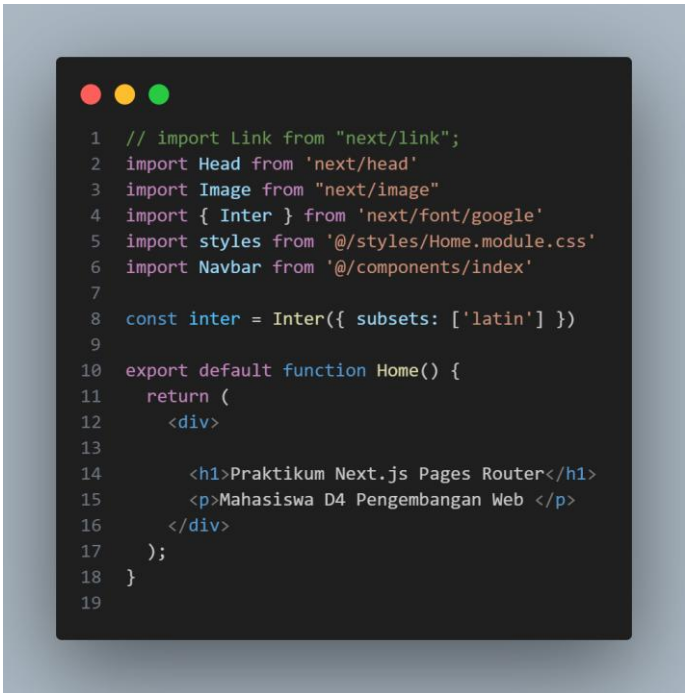
- Modifikasi _app.tsx (pastikan import styles dalam keadaan aktif)

```tsx
import '@/styles/globals.css'
import type { AppProps } from 'next/app'

export default function App({ Component, pageProps }: AppProps) {
  return <Component {...pageProps} />
}

```

- Jalankan di browser (Navbar akan tampil)

Modifikasi navbar agar tampil di semua page

- Modifikasi index.tsx pada folder page (hapus navbar)

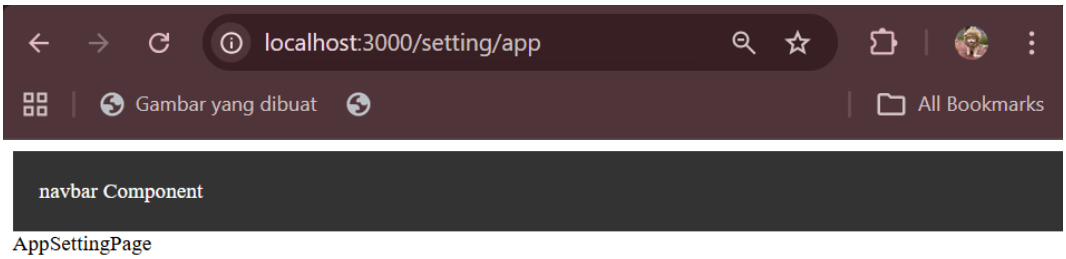


```
// import Link from "next/link";
import Head from 'next/head'
import Image from "next/image"
import { Inter } from 'next/font/google'
import styles from '@/styles/Home.module.css'
import Navbar from '@/components/index'

const inter = Inter({ subsets: ['latin'] })

export default function Home() {
  return (
    <div>

      <h1>Praktikum Next.js Pages Router</h1>
      <p>Mahasiswa D4 Pengembangan Web </p>
    </div>
  );
}
```

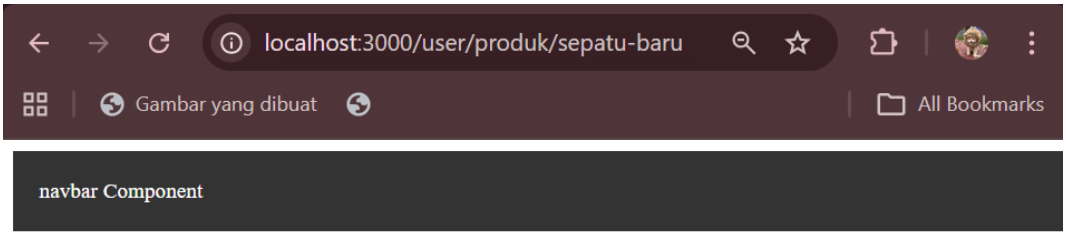
- Modifikasi_app.tsx (Menambahkan navbar)

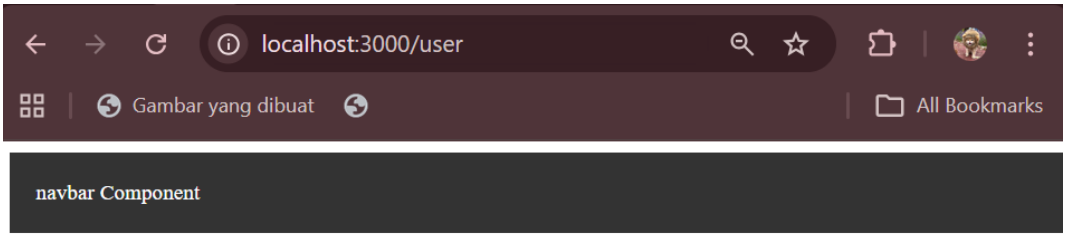
```
import '@/styles/globals.css'
import type { AppProps } from 'next/app'
import Navbar from '@/components/index'

export default function App({ Component, pageProps }: AppProps) {
  return (
    <div>
      <Navbar />
      <Component {...pageProps} />
    </div>
  )
}
```

- Jalankan browser



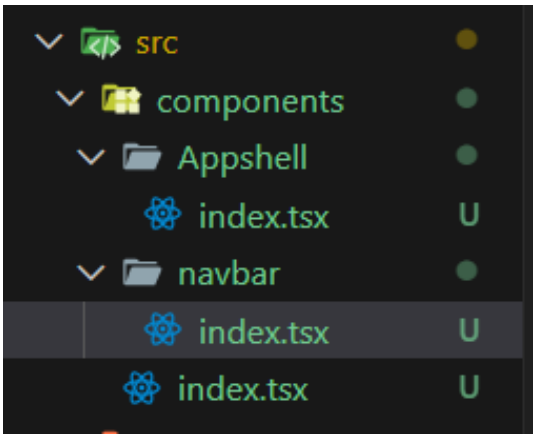
AppSettingPage





UserSettingPage

6. Membuat Layout Global (App Shell)
   a. Buat AppShell



   b. Modifikasi index.tsx pad AppShell

```tsx
import Navbar from "@/components/index";

type AppShellProps = {
children: React.ReactNode;
}

const AppShell = (props:AppShellProps) => {
    const { children } = props;
    return (
    <main>
        <Navbar />
        {children}
    </main>
    );
};

export default AppShell;
```

7. Implementasi Layout di _app.tsx

```tsx
import '@/styles/globals.css'
import type { AppProps } from 'next/app'
import AppShell from '@/components/Appshell'
import Navbar from '@/components/index'

export default function App({ Component, pageProps }: AppProps) {
  return (
    <AppShell>
      <Component {...pageProps} />
    </AppShell>
  )
}
```

Hasil: Navbar dan layout muncul di semua halaman tanpa perlu dipanggil satu per satu.contoh Modifikasi pada _app.tsx tambahkan footer seperti pada gambar dan amati hasilnya

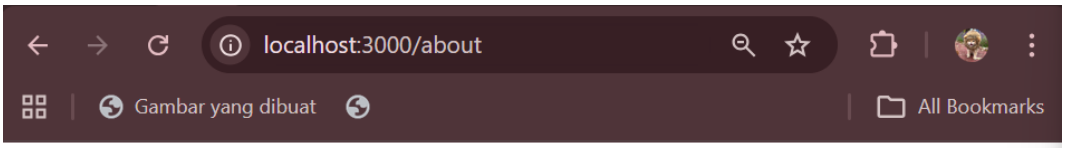
```tsx
import Navbar from "@/components/index";

type AppShellProps = {
children: React.ReactNode;
}

const AppShell = (props:AppShellProps) => {
    const { children } = props;
    return (
        <main>
        <Navbar />
        {children}
        <div>
            footer
        </div>
        </main>
    );
    };

export default AppShell;
```

navbar Component

Praktikum Next.js Pages Router

Mahasiswa D4 Pengembangan Web

footer



navbar Component

Ini Halaman About

footer

E. Tugas Praktikum

Tugas 1 – Routing

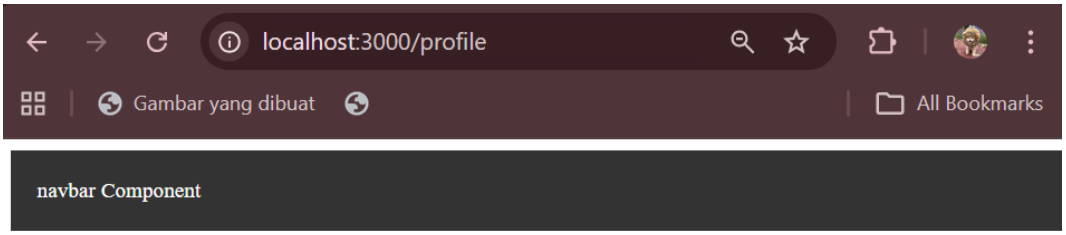
1. Buat halaman:
    - /profile

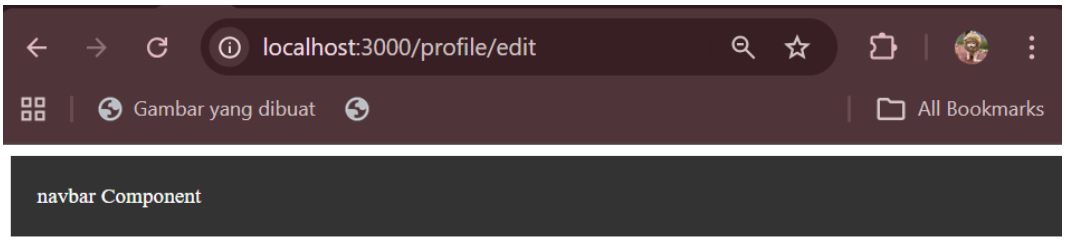
```jsx
const ProfilePage = () => {
  return (
    <div>
      <h1>Halaman Profile</h1>
    </div>
  );
};

export default ProfilePage;
```

    - /profile/edit

```jsx
const EditProfilePage = () => {
  return (
    <div>
      <h1>Edit Profile Page</h1>
    </div>
  );
};

export default EditProfilePage;
```

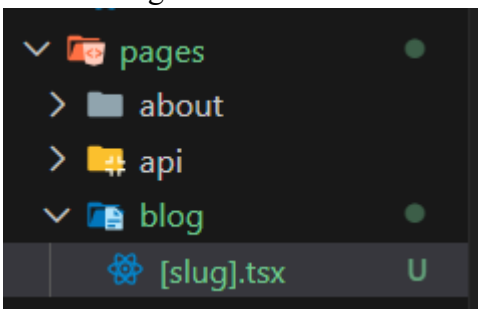
2. Pastikan routing berjalan tanpa error





Tugas 2 – Dynamic Routing

1. Buat routing:



2. /blog/[slug]

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

const DetailBlog = () => {
  const { query } = useRouter();

  return (
    <div>
      <h1>Detail Blog</h1>
      <p>Slug: {query.slug}</p>
    </div>
  );
};

export default DetailBlog;

```

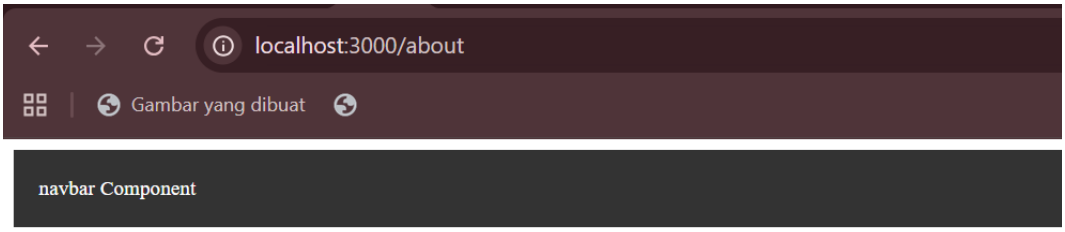
3. Tampilkan nilai slug di halaman

Tugas 3 – Layout

1. Tambahkan Footer pada AppShell

```tsx
import Navbar from "@/components/index";

type AppShellProps = {
children: React.ReactNode;
}

const AppShell = (props:AppShellProps) => {
    const { children } = props;
    return (
        <main>
        <Navbar />
        {children}
        <footer style={{ marginTop: "20px" }}>
            <hr />
            <p> © 2026 - Praktikum Next.js</p>
        </footer>
        <div>
        </div>
        </main>
    );
    };

export default AppShell;
```

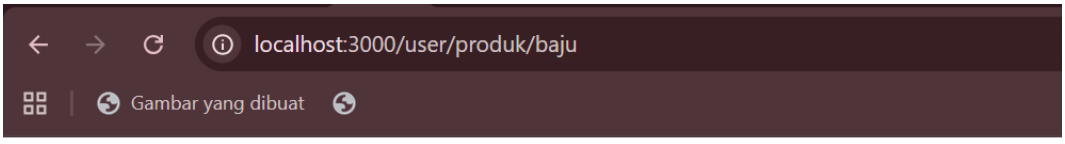
2. Footer tampil di semua halaman



navbar Component

Ini Halaman About

© 2026 - Praktikum Next.js

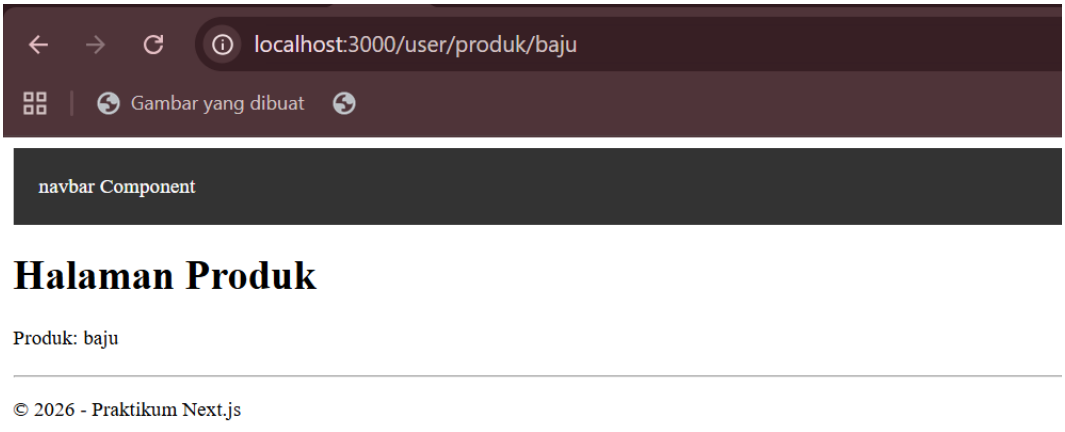


navbar Component

Halaman Produk

Produk: baju

© 2026 - Praktikum Next.js



F. Pertanyaan Refleksi

1. Apa perbedaan routing berbasis file dan routing manual?

- Routing berbasis file (file-based routing) adalah sistem routing yang secara otomatis dibuat berdasarkan struktur folder dan nama file di dalam direktori pages. Pada Next.js, setiap file di dalam folder pages akan langsung menjadi sebuah route tanpa perlu konfigurasi tambahan.

  Sedangkan routing manual adalah sistem routing yang mengharuskan developer mendefinisikan setiap route secara eksplisit dalam file konfigurasi atau kode tertentu, seperti menggunakan React Router pada React atau mendefinisikan route pada file web.php di Laravel.

  Perbedaannya:
  1. Routing berbasis file lebih sederhana dan cepat karena tidak memerlukan konfigurasi tambahan.
  2. Routing manual lebih fleksibel tetapi membutuhkan pengaturan route satu per satu.
  3. File-based routing lebih cocok untuk pengembangan cepat dan struktur yang rapi.
  4. Routing manual biasanya digunakan pada framework yang tidak memiliki sistem routing otomatis.

2. Mengapa dynamic routing penting dalam aplikasi web?

- Dynamic routing penting karena memungkinkan aplikasi web menangani banyak halaman dengan struktur yang sama tetapi memiliki konten berbeda berdasarkan parameter URL.

  Contohnya pada halaman:
  - Detail produk
  - Detail artikel blog
  - Profil pengguna

Dengan dynamic routing, cukup menggunakan satu file seperti [id].tsx atau [slug].tsx, lalu parameter pada URL dapat digunakan untuk menampilkan konten yang berbeda.

Keunggulan dynamic routing:

- Menghemat pembuatan banyak file
- Lebih efisien dan scalable
- Mendukung pengambilan data berdasarkan parameter (misalnya dari database atau API)
- Cocok untuk aplikasi berbasis data

3. Apa keuntungan menggunakan layout global dibanding memanggil komponen satu per satu?

- Keuntungannya:
    - Mengurangi duplikasi kode
    - Struktur aplikasi lebih rapi
    - Perubahan cukup dilakukan di satu tempat
    - Konsistensi tampilan antar halaman terjaga
    - Lebih mudah dikembangkan dan dipelihara (maintainable)

    Dengan menggunakan AppShell atau _app.tsx, semua halaman otomatis dibungkus oleh layout yang sama.