# Laporan Tugas Pratikum

# Jobsheet 2


**Oleh:**

**Satrio Ahmad Ramadhani**

**2341720163**

**TI-3B/22**

**D-IV TEKNIK INFORMATIKA**

**JURUSAN TEKNOLOGI INFORMASI**

**POLITEKNIK NEGERI MALANG**

**JOBSHEET PRAKTIKUM**

**Routing, Nested Routing, Dynamic Routing, dan Layouting pada Next.js (Pages Router)**

**Mata Kuliah**

Pemrograman Framework

**Topik Praktikum**

Routing & Layouting pada Next.js (Pages Router)

**Waktu Praktikum**

2 × 50 menit

---

**A. Tujuan Praktikum**

Setelah menyelesaikan praktikum ini, mahasiswa mampu:

1. Memahami konsep **Pages Router** pada Next.js
2. Membuat **routing statis** berbasis file dan folder
3. Mengimplementasikan **nested routing**
4. Mengimplementasikan **dynamic routing** menggunakan parameter URL
5. Membuat **layout global** menggunakan komponen layout (App Shell)

---

**B. Tools & Persiapan**

- Node.js (minimal v16)
- NPM / Yarn / PNPM
- Code Editor (VS Code disarankan)
- Browser (Chrome / Firefox)
- Project Next.js (TypeScript)

npx create-next-app@latest next-routing

cd next-routing

npm run dev

---

**C. Dasar Konsep (Ringkas)**

- Folder **pages/** → otomatis menjadi routing
- index.tsx → root route (/)
- Folder di dalam pages/ → nested route

- File [param].tsx → dynamic routing
- pages/_app.tsx → entry point global aplikasi

---

**D. Langkah Praktikum**

---

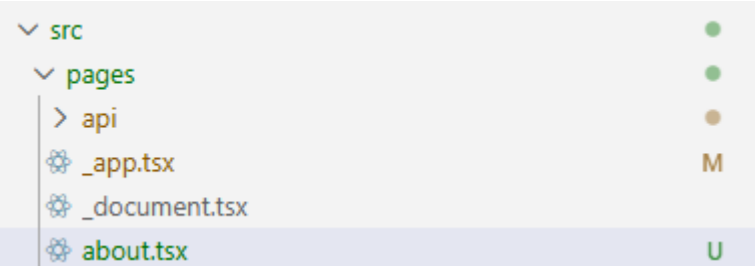
**1. Routing Dasar (Static Routing)**

**a. Struktur Awal**

pages/

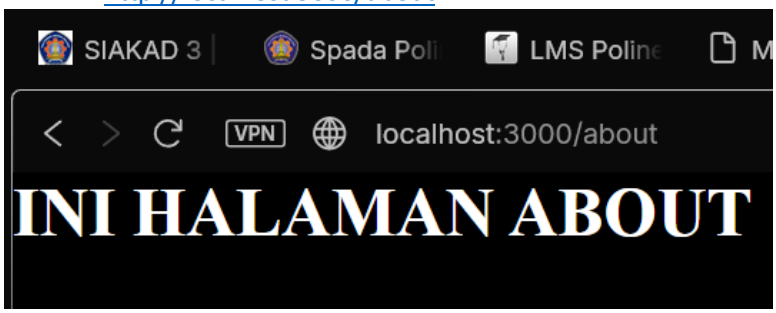
└── index.tsx

**b. Tambahkan Halaman About**




**c. Uji di Browser**

- http://localhost:3000/about

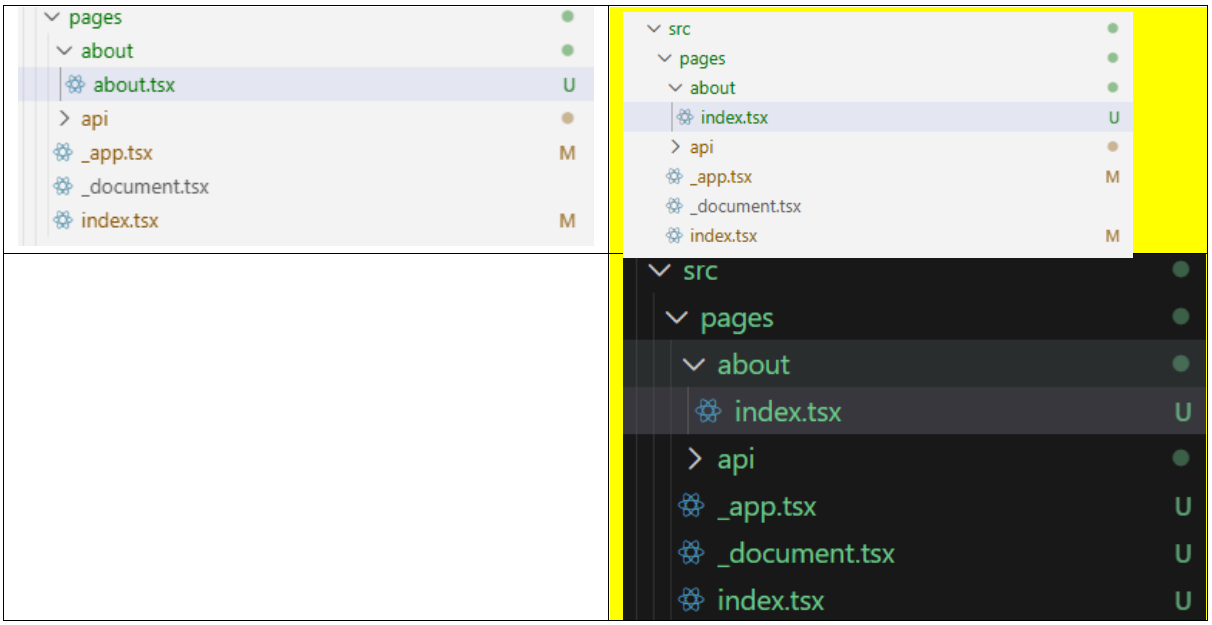



**Catatan:**
**Next.js otomatis membuat routing berdasarkan nama file tanpa konfigurasi tambahan.**

---

**2. Routing Menggunakan Folder**

**a. Rapikan Struktur Pages**




Ubah struktur menjadi:

pages/

└── about/

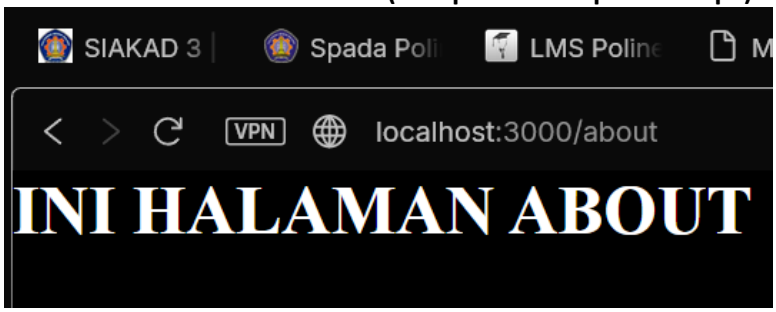
└── index.tsx **( yang sebelumnya about.tsx menjadi index.tsx )**

Akses:

/about

**Insight:**
index.tsx di dalam folder mewakili root folder tersebut.

**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

praktikum-PagesLayout > my-app > src > pages > setting > user.tsx > ...

Windsurf: Refactor | Explain | Generate JSDoc | ×

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

export default UserSettingPage;
```

- app.tsx

praktikum-PagesLayout > my-app > src > pages > setting > app.tsx > ...

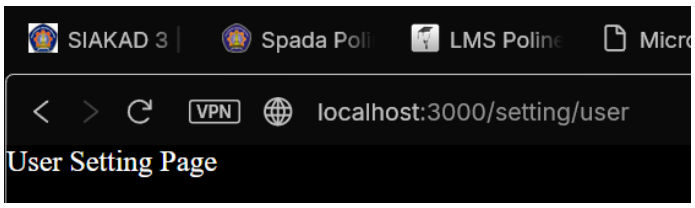
Windsurf: Refactor | Explain | Generate JSDoc | ×

```tsx
const Appsetting = () => {
  return (
  <div>
    App Setting Page
  </div>
    );
};

export default Appsetting;
```

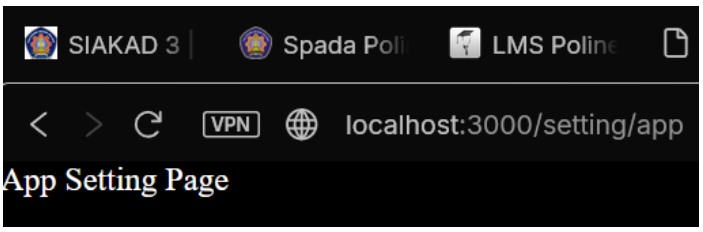
Tips: Copy paste dari user.tsx , block Usersettingpage + Control+D maka saat merubah usersettingpage menjadi Appsetting akan berubah semua tanpa harus rename satu-satu

Akses:

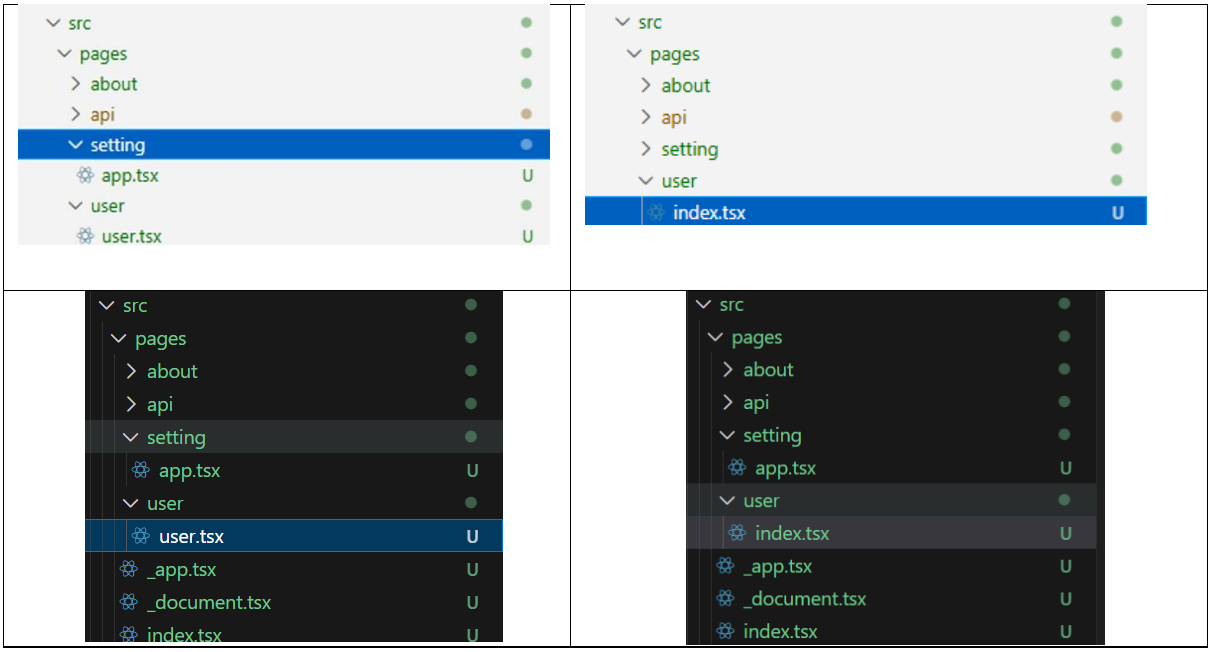
- /setting/user




- /setting/app

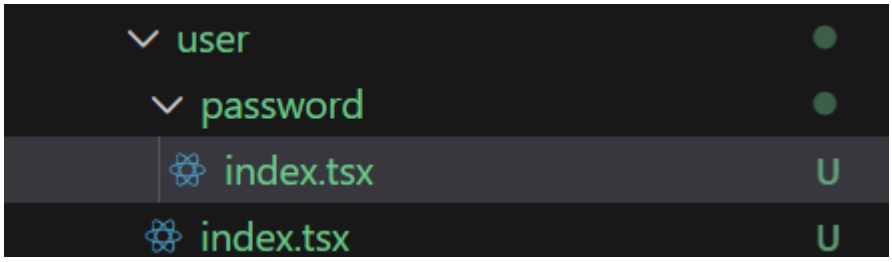


Modifikasi struktur folder pages dengan menambahkan folder user dan user.tsx pada setting dipindah ke folder user dan rubah file user.tsx menjadi index.tsx




Jalankan pada browser


**b. Nested Lebih Dalam**




pages/

└── setting/

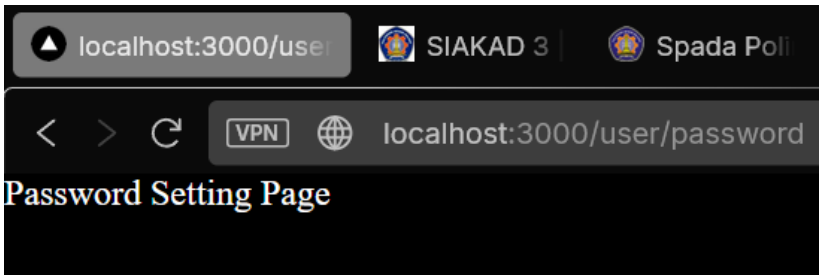
└── user/

└── password/

└── index.tsx

Akses:

/setting/user/password




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

---

1. **Dynamic Routing**

**c. Buat Halaman Produk**

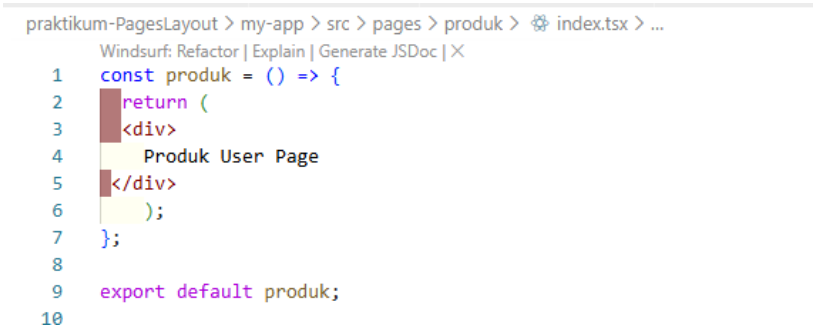
pages/

└── produk/

  ├── index.tsx

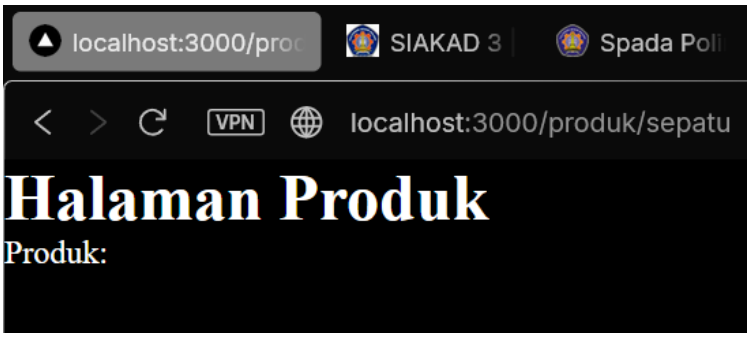
  └── [id].tsx


- **Modifikasi index.tsx**

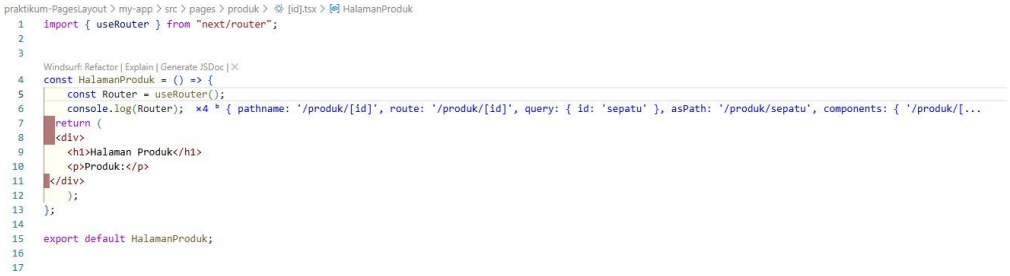



- **Modifikasi [id].tsx**

  Buka browser http://localhost:3000/produk/sepatu tambahkan segment sepatu




- **Cek menggunakan console.log**




jika berhasil maka pada console.log dapat terlihat pada id terdapat nilai sepatu.

⚡ **Catatan Penting:**

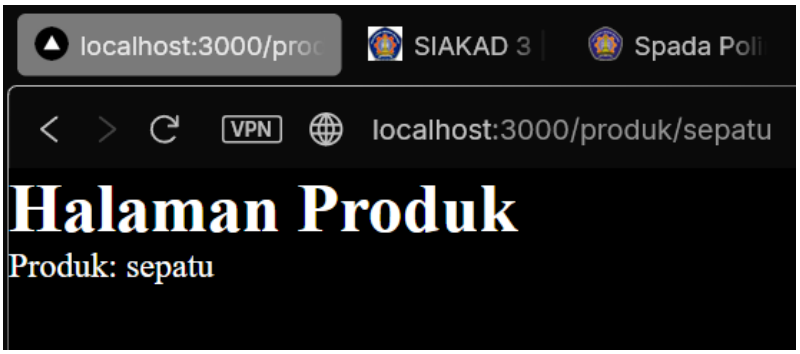
Pada console.log data langsung terlihat dikarenakan terdapat ext vscode console ninja

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

praktikum-PagesLayout > my-app > src > pages > produk > [id].tsx > ...

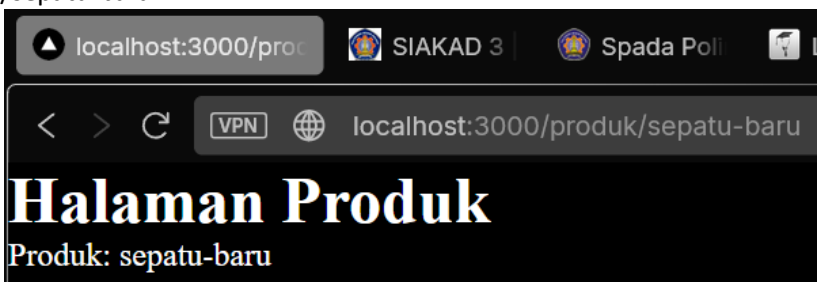
```tsx
import { useRouter } from "next/router";


Windsurf: Refactor | Explain | Generate JSDoc |
const HalamanProduk = () => {
    // const Router = useRouter();
    // console.log(Router);
    const { query } = useRouter();
  return (
  <div>
    <h1>Halaman Produk</h1>
    <p>Produk: {query.id}</p>
  </div>
    );
};

export default HalamanProduk;

```

- Buka browser



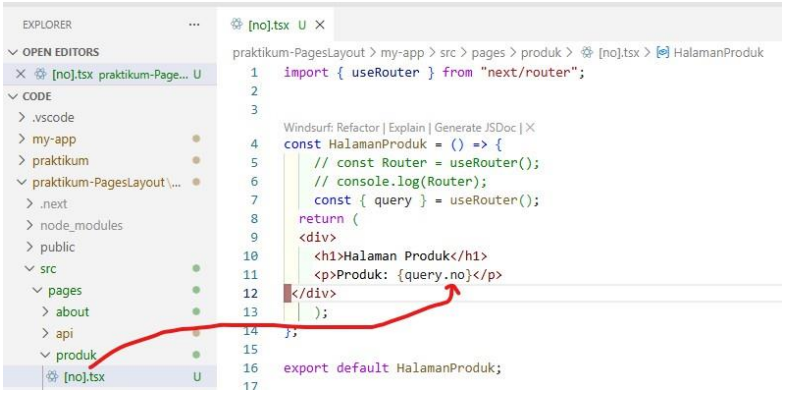

**c. Uji di Browser**

- /produk/sepatu-baru




- /produk/baju


**Catatan Penting:**
Nama file di dalam [ ] akan menjadi **parameter URL**. Contoh



2. **Membuat Komponen Navbar**

**d. Struktur Komponen**

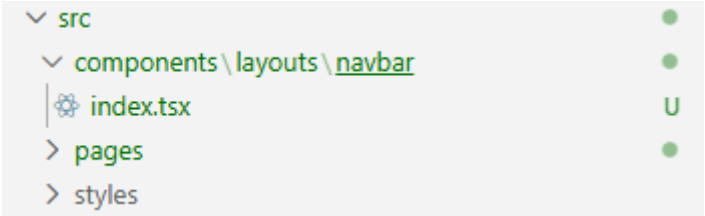
src/

└── components/

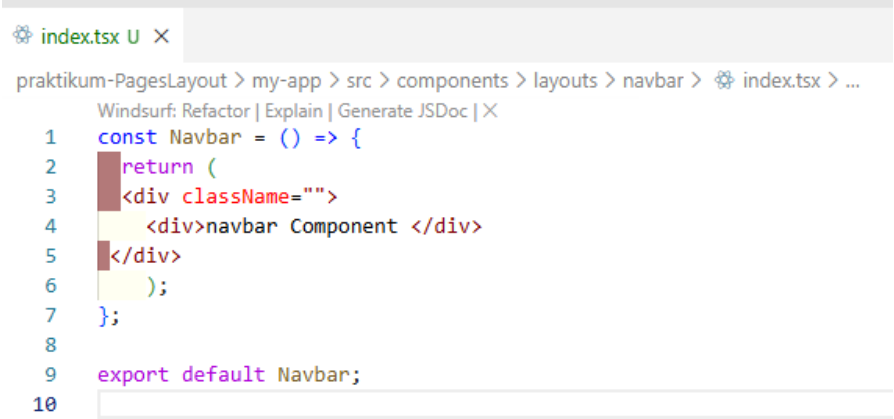
└── layouts/

└── Navbar/

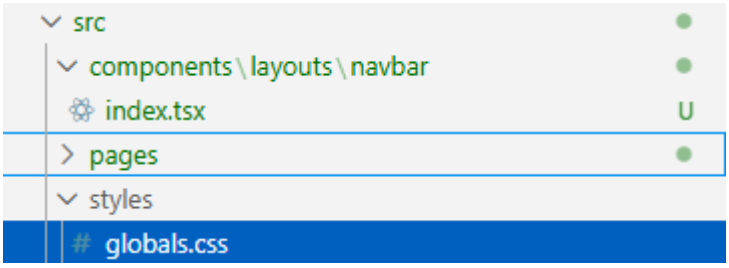
└── index.tsx




- Modifikasi index.tsx




- Buka globals.css untuk nantinya digunakan pada style navbar




  - globals.css

praktikum-PagesLayout > my-app > src > styles > # globals.css > :root

```
:root {…
}

@media (prefers-color-scheme: dark) {…
}

* {…
}

html,
body {…
}

body {…
}

a {…
}

@media (prefers-color-scheme: dark) {…
}
```

- Modifikasi global.css

praktikum-PagesLayout > my-app > src > styles > # globals.css > ...

```
* {
  box-sizing: border-box;
  padding: 0;
  margin: 0;
}

html,
body {
  max-width: 100vw;
  overflow-x: hidden;
}


a {
  color: inherit;
  text-decoration: none;
}
```

- Modifikasi index.tsx dengan menambahkan classname untuk style navbar

praktikum-PagesLayout > my-app > src > components > layouts > navbar > index.tsx > ...

Windsurf: Refactor | Explain | Generate JSDoc |

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

export default Navbar;
```

- Modifikasi globals.css

praktikum-PagesLayout > my-app > src > styles > # globals.css > .navbar

```
1
2
3  > * {…
7    }
8
9    html,
10 > body {…
13   }
14
15
16
17 > a {…
20   }
21
22
23   .navbar {
24     width: 100%;
25     height: 60px;
26     background-color: #333;
27     color: white;
28     display: flex;
29     align-items: center;
30     padding: 0 20px;
31   }
```

- Modifikasi index.tsx pada folder pages

praktikum-PagesLayout > my-app > src > pages > index.tsx > ...

```
1  import Head from 'next/head'
2  import Image from 'next/image'
3  import { Inter } from 'next/font/google'
4  import styles from '@/styles/Home.module.css'
5  import Navbar from '@/components/layouts/navbar'
6
7  const inter = Inter({ subsets: ['latin'] })
8
   Windsurf: Refactor | Explain | Generate JSDoc | X
9  export default function Home() {
10   return (
11     <div>
12       <Navbar />
13       <h1>Praktikum Next.js Pages Router</h1> <br />
14       <p>Mahasiswa D4 Pengembangan Web</p>
15     </div>
16
17   )
18 }
19
```

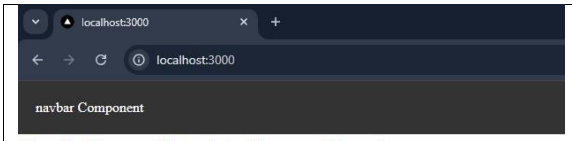
- Modifikasi _app.tsx ( pastikan import styles dalam keadaan aktif)

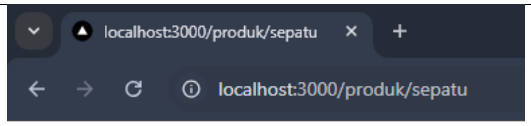
praktikum-PagesLayout > my-app > src > pages > _app.tsx > ...

```
1  import '@/styles/globals.css'
2  import type { AppProps } from 'next/app'
3
   Windsurf: Refactor | Explain | Generate JSDoc | X
4  export default function App({ Component, pageProps }: AppProps) {
5    return <Component {...pageProps} />
6  }
7
```

- Jalankan di browser ( Navbar akan tampil )


**Catatan Penting:**
navbar hanya akan muncul pada index page, pada page produk navbar tidak akan muncul. Contoh




- Modifikasi navbar agar tampil di semua page
  - Modifikasi index.tsx pada folder page ( hapus navbar )

praktikum-PagesLayout > my-app > src > pages > index.tsx > ...

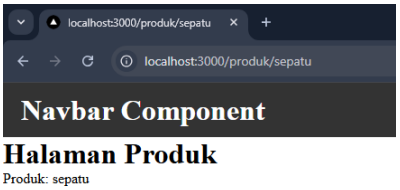
```tsx
import Head from 'next/head'
import Image from 'next/image'
import { Inter } from 'next/font/google'
import styles from '@/styles/Home.module.css'


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

Windsurf: Refactor | Explain | Generate JSDoc | X
export default function Home() {
  return (
    <div>

      <h1>Praktikum Next.js Pages Router</h1> <br />
      <p>Mahasiswa D4 Pengembangan Web</p>
    </div>

  )
}

```

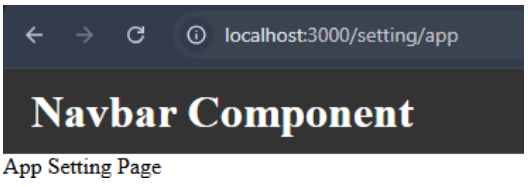
  - Modifikasi _app.tsx ( Menambahkan navbar )

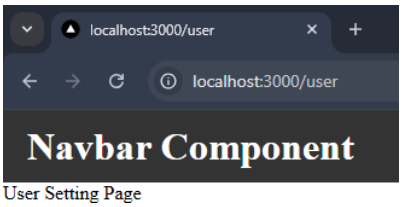
praktikum-PagesLayout > my-app > src > pages > _app.tsx > ...

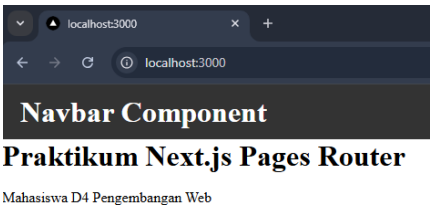
```tsx
import '@/styles/globals.css'
import type { AppProps } from 'next/app'
import Navbar from '@/components/layouts/navbar'

Windsurf: Refactor | Explain | Generate JSDoc | X
export default function App({ Component, pageProps }: AppProps) {
  return (
  <div>
    <Navbar />
    <Component {...pageProps} />
  </div>
  );
};

```

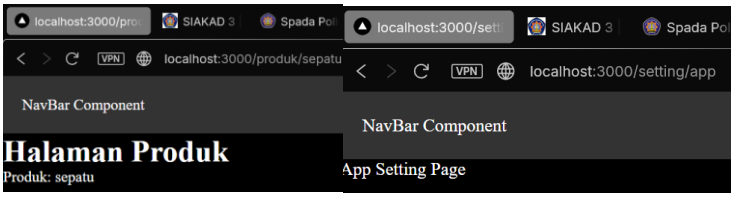
- Jalankan browser

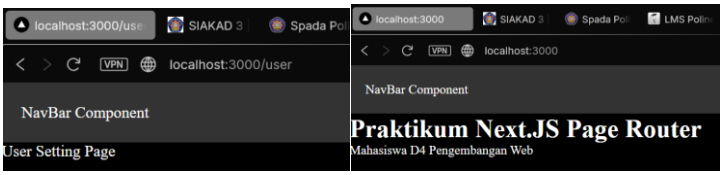


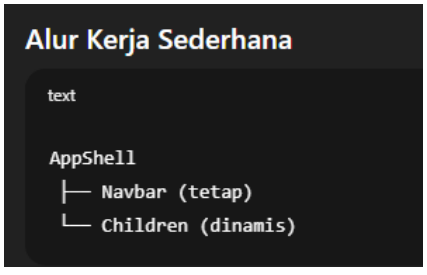



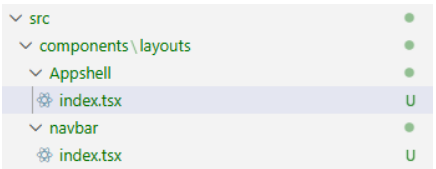




---

### 3. Membuat Layout Global (App Shell)

**e. Buat AppShell**



- **Modifikasi index.tsx pad AppShell**

praktikum-PagesLayout > my-app > src > components > layouts > Appshell > index.tsx > AppShell

```tsx
import Navbar from "../navbar";

Windsurf: Refactor | Explain
type AppShellProps = {
    children: React.ReactNode;
}

Windsurf: Refactor | Explain | Generate JSDoc | X
const AppShell = (props:AppShellProps) => {
    const { children } = props;
  return (
  <main>
    <Navbar />
    {children}
  </main>

    );
};

export default AppShell;

```

---

### 4. Implementasi Layout di _app.tsx

praktikum-PagesLayout > my-app > src > pages > _app.tsx > App

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

Windsurf: Refactor | Explain | Generate JSDoc | X
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

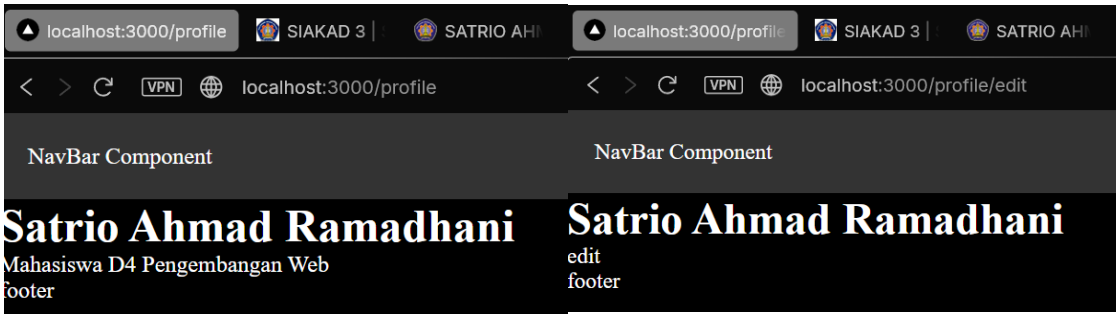
Modifikasi pada **_app.tsx tambahkan footer seperti pada gambar dan amati hasilnya**

```tsx
  */
  const { children } = props;
  return (
  <main>
    <Navbar />
    {children}
  <div>
    footer
  </div>
  </main>
```


**E. Tugas Praktikum**

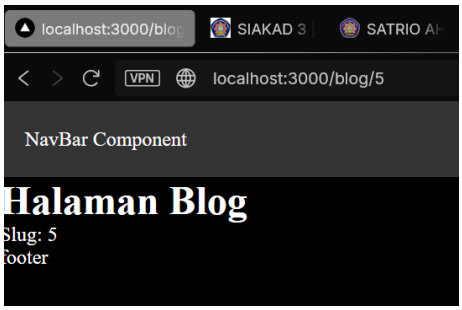
**Tugas 1 – Routing**

1. Buat halaman:
    - /profile
    - /profile/edit
2. Pastikan routing berjalan tanpa error




**Tugas 2 – Dynamic Routing**

1. Buat routing:
2. /blog/[slug]
3. Tampilkan nilai slug di halaman




**Tugas 3 – Layout**

1. Tambahkan Footer pada AppShell
2. Footer tampil di semua halaman

**F. Pertanyaan Refleksi**

1. Apa perbedaan routing berbasis file dan routing manual?
2. Mengapa dynamic routing penting dalam aplikasi web?
3. Apa keuntungan menggunakan layout global dibanding memanggil komponen satu per satu?

**G. Jawaban**

1. Routing berbasis file di Next.js otomatis mencocokkan struktur folder pages/ dengan URL sehingga cukup membuat file baru untuk menambahkan rute, sedangkan pada routing manual kita harus menulis sendiri aturan URL-komponen dengan kode seperti app.get atau router.push, memberi fleksibilitas lebih tapi menambah boilerplate dan potensi kesalahan.
2. Dynamic routing penting karena memungkinkan aplikasi memakai parameter di URL (misalnya /blog/123 atau /produk/abc), jadi konten dapat ditampilkan berdasarkan ID atau nama yang berubah-ubah; tanpa itu, kita terpaksa mendefinisikan rute terpisah untuk setiap entitas, yang tidak praktis dan mengganggu SEO.
3. Menggunakan layout global berarti elemen seperti header, footer, atau sidebar hanya ditentukan sekali sehingga setiap halaman mewarisinya, menghindari duplikasi komponen di banyak file, memperkuat konsistensi tampilan, serta memudahkan pemeliharaan karena struktur umum aplikasi terpusat.

---

**H. Kesimpulan**

Melalui praktikum ini, mahasiswa telah memahami bahwa **Next.js Pages Router**:

- Menghemat waktu konfigurasi routing
- Mendukung nested dan dynamic routing secara natural
- Memudahkan pengelolaan layout global menggunakan _app.ts