

Nama : Indri Tri Maharani
NPM : 22312035

1. First Commit
2. Membuat file page.js, layout.js, global.css, page.module.css pada folder app

```
# globals.css
JS layout.js
JS page.js
# page.module.css
```

3. Membuat File page.js

```
'use client';
```

Kode ini menunjukkan bahwa file ini adalah komponen React Client-side. React Server Components dapat berjalan di server, sedangkan komponen yang diawali 'use client' berjalan di browser (client). Digunakan untuk memberikan sinyal kepada framework bahwa seluruh kode dalam file ini akan dijalankan di sisi klien.

```
import { useState } from 'react';
import Header from './components/Header';
import Banner from './components/Banner';
import Categories from './components/Categories';
import ProductList from './components/ProductList';
import ContactForm from './components/ContactForm';
import Footer from './components/Footer';
```

useState merupakan Hook React untuk mengelola state (data yang berubah-ubah) dalam komponen.

Import komponen:

- import Header : Komponen untuk menampilkan bagian header.
- import Banner : Komponen untuk menampilkan (banner) di halaman.
- import Categories : Komponen untuk memilih kategori produk.
- import ProductList : Komponen untuk menampilkan daftar produk.
- import ContactForm : Komponen untuk menampilkan formulir kontak.
- import Footer : Komponen untuk menampilkan footer halaman.

Semua komponen diatas diintegrasikan untuk membangun tampilan halaman.

```
export default function Page() {
  const [selectedCategory, setSelectedCategory] = useState('Semua');
```

- Function Page adalah fungsi React yang menjadi komponen utama halaman.
- useState digunakan untuk membuat state selectedCategory yang menyimpan kategori yang dipilih oleh pengguna. Nilai awalnya adalah 'Semua', artinya menampilkan semua kategori produk.
- setSelectedCategory adalah fungsi untuk mengubah nilai selectedCategory.

```

return (
  <div>
    <Header />
    <main>
      <Banner />
      <Categories onSelectCategory={setSelectedCategory} />
      <ProductList selectedCategory={selectedCategory} />
      <ContactForm />
    </main>
    <Footer />
  </div>
);
}

```

- <div> Semua elemen halaman dibungkus dalam satu elemen utama.
- <Header /> Menampilkan komponen header, biasanya berisi logo atau navigasi.
- <main> Elemen utama untuk konten halaman berisi :
 - <Banner /> Menampilkan spanduk/banner, berupa gambar atau informasi utama.
 - <Categories onSelectCategory={setSelectedCategory} /> berisi Komponen untuk memilih kategori produk.
 - onSelectCategory adalah callback yang memanggil fungsi setSelectedCategory untuk mengubah kategori yang dipilih.
 - <ProductList selectedCategory={selectedCategory} /> Menampilkan daftar produk berdasarkan kategori yang dipilih.
 - selectedCategory digunakan untuk menyaring daftar produk.
 - <ContactForm /> merupakan komponen untuk menampilkan formulir kontak pengguna.
- <Footer /> digunakan untuk menampilkan footer, berisi link, atau kontak tambahan.

4. Membuat File globals.css

```

@tailwind base;
@tailwind components;
@tailwind utilities;

```

- @tailwind base; digunakan untuk mengimpor reset CSS untuk memberikan dasar yang konsisten pada elemen HTML.
- @tailwind components; digunakan untuk menyediakan kelas bawaan untuk membuat komponen UI seperti tombol atau kartu.
- @tailwind utilities; menyediakan berbagai kelas utilitas kecil yang fleksibel, seperti pengaturan warna, margin, padding, dan tata letak, yang memungkinkan penyesuaian detail gaya dengan mudah.

5. Membuat File layout.js

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

- Geist dan Geist_Mono : Mengimpor font Google melalui fitur bawaan Next.js (next/font/google). Geist dan Geist_Mono adalah font yang digunakan dalam proyek.
- "./globals.css" : Mengimpor file CSS global yang berisi aturan gaya (style) yang berlaku di seluruh aplikasi. globals.css digunakan untuk mendefinisikan gaya dasar atau global aplikasi.

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

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

Menyediakan pengaturan font untuk digunakan di aplikasi. geistSans dan geistMono yang mengonfigurasi masing-masing font (Geist dan Geist_Mono).

- Variable : Mendefinisikan CSS custom property (--font-geist-sans dan --font-geist-mono) untuk font.
- Subsets : Menentukan subset karakter yang digunakan (misalnya, "latin").

```
export const metadata = {
  title: "Create Next App",
  description: "Generated by create next app",
};
```

metadata adalah objek yang berisi informasi metadata halaman. Seperti, title : Judul halaman (ditampilkan di tab browser). description: Deskripsi singkat halaman (sering digunakan untuk SEO). Metadata digunakan untuk memberikan informasi tentang halaman secara global.

```
Codeium: Refactor | Explain | Generate JSDoc | ✕
export default function RootLayout({ children }) {
  return (
    <html lang="en">
      <body className={` ${geistSans.variable} ${geistMono.variable}`}>
        {children}
      </body>
    </html>
  );
}
```

- RootLayout : Komponen layout utama aplikasi Next.js yang digunakan untuk membungkus seluruh konten aplikasi. RootLayout menentukan struktur HTML dasar

untuk seluruh aplikasi. Membungkus semua konten aplikasi di dalam elemen `<html>` dan `<body>`.

- Children : Properti bawaan React yang berisi elemen atau komponen anak yang akan dirender di dalam layout.
- Struktur jsx :
 - `<html lang="en">` Merupakan elemen root HTML dengan atribut `lang="en"`, menunjukkan bahwa bahasa halaman adalah bahasa Inggris.
 - `<body>` Merupakan elemenn HTML yang memuat seluruh konten aplikasi, seperti `className` yang digunakan untuk menggabungkan custom property CSS dari `geistSans` (`--font-geist-sans`) dan `geistMono` (`--font-geist-mono`) untuk menerapkan font ke seluruh halaman.

6. Membuat file `page.module.css`

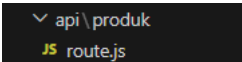
```
app > # page.module.css > .page
1  .page {
2    --gray-rgb: 0, 0, 0;
3    --gray-alpha-200: rgba(var(--gray-rgb), 0.08);
4    --gray-alpha-100: rgba(var(--gray-rgb), 0.05);
5
6    --button-primary-hover: #383838;
7    --button-secondary-hover: #f2f2f2;
8
9    display: grid;
10   grid-template-rows: 20px 1fr 20px;
11   align-items: center;
12   justify-items: center;
13   min-height: 100svh;
14   padding: 80px;
15   gap: 64px;
16   font-family: var(--font-geist-sans);
17 }
18
19 @media (prefers-color-scheme: dark) {
20   .page {
21     --gray-rgb: 255, 255, 255;
22     --gray-alpha-200: rgba(var(--gray-rgb), 0.145);
23     --gray-alpha-100: rgba(var(--gray-rgb), 0.06);
24
25     --button-primary-hover: #ccc;
26     --button-secondary-hover: #1a1a1a;
27   }
28 }
29
30 .main {
31   display: flex;
32   flex-direction: column;
33   gap: 32px;
34   grid-row-start: 2;
35 }
```

Kode ini berfungsi untuk membuat halaman web yang responsif, mendukung mode terang/gelap, dengan tata letak rapi menggunakan grid dan flexbox.

- Css

- Variabel seperti `--gray-rgb`, `--gray-alpha-200`, `--button-primary-hover` didefinisikan untuk menyimpan nilai warna atau properti tertentu yang dapat digunakan ulang.
- Mode gelap diatur dengan media query `@media (prefers-color-scheme: dark)`.
- `.page` Kelas ini mendefinisikan tata letak utama halaman dengan properti seperti:
 - Grid layout membagi halaman menjadi 3 baris : header, konten utama, dan footer.
 - `align-items` dan `justify-items` : Mengatur perataan isi grid.
 - Font: Menggunakan font dari variabel `--font-geist-sans`.
 - `padding: 80px` : Memberikan ruang di sekitar isi halaman.
 - `gap: 64px` : Memberikan jarak antara elemen dalam grid.
- `.main` (Konten Utama)
 - Menggunakan flexbox untuk tata letak vertikal (`flex-direction: column`).
 - Memberi jarak antar elemen dengan `gap: 32px`.
 - Elemen daftar terurut (ol) diberi gaya khusus :
Font monospace: Menggunakan variabel `--font-geist-mono`. Tidak ada padding dan margin. Gaya huruf kecil dengan jarak antar baris (`line-height`) lebih besar.
- `.ctas`
 - `display: flex`: Menampilkan tombol secara horizontal. `gap: 16px`: Memberi jarak antar tombol.
 - Tombol `a.primary` dan `a.secondary` memiliki gaya berbeda.
 - `a.primary` : Warna latar dari `--foreground` dan teks dari `--background`.
 - `a.secondary` : Memiliki border dengan warna transparansi abu-abu.
- Footer
Menggunakan `grid-row-start: 3` untuk menempatkan footer di baris ketiga grid. Elemen footer seperti link dan gambar memiliki gaya khusus seperti, link diberi jarak antar ikon dan teks (`gap: 8px`), dan gambar di footer tidak mengecil dengan `flex-shrink: 0`.
- Responsif
(`max-width: 600px`) berarti tombol tersusun vertical, padding dan ukuran font diperkecil, footer elemen disusun ulang agar tidak meluber.
- Mode Gelap

7. Membuat Folder api yang berisi folder produk dan file route.js



```

  api \ produk
    JS route.js
  
```

8. Membuat File route.js

```
import { PrismaClient } from '@prisma/client';
```

`import { PrismaClient } from '@prisma/client'`; artinya mengimpor `PrismaClient` dari library `@prisma/client`. `PrismaClient` adalah library untuk berinteraksi dengan database secara otomatis menggunakan query builder.

```
const prisma = new PrismaClient();
```

`const prisma = new PrismaClient();` artinya membuat instance baru dari `PrismaClient` bernama `prisma`. Instance ini digunakan untuk menjalankan semua operasi database seperti membaca, menambah, memperbarui, dan menghapus data.

```
// membaca Semua Produk
Codeium: Refactor | Explain | ✕
export async function GET() {
  const products = await prisma.product.findMany();
  return new Response(JSON.stringify(products), {
    headers: { 'Content-Type': 'application/json' },
  });
}
```

- Function GET adalah endpoint API yang digunakan untuk membaca seluruh data produk dari tabel product.
- `prisma.product.findMany()` mengambil semua data yang ada di tabel product.
- Hasilnya diubah menjadi format JSON menggunakan `JSON.stringify()` dan dikembalikan dalam response dengan header `Content-Type: application/json`.

```
// Menambah Produk Baru
Codeium: Refactor | Explain | ✕
export async function POST(request) {
  const newProduct = await request.json();
  const createdProduct = await prisma.product.create({
    data: {
      name: newProduct.name,
      category: newProduct.category,
      price: newProduct.price,
    },
  });
  return new Response(JSON.stringify(createdProduct), { status: 201 });
}
```

- Function POST ini digunakan untuk menambahkan produk baru ke database.
- Data produk yang dikirim oleh klien (dalam `request.body`) diambil menggunakan `request.json()`.
- `prisma.product.create()` menambahkan data baru ke tabel product dengan properti `name`, `category`, dan `price`.
- Hasil data yang berhasil ditambahkan dikembalikan dalam format JSON dengan status 201 Created.

```
// Memperbaharui Produk
Codeium: Refactor | Explain | X
export async function PUT(request) {
  const updatedProduct = await request.json();
  const product = await prisma.product.update({
    where: { id: updatedProduct.id },
    data: {
      name: updatedProduct.name,
      category: updatedProduct.category,
      price: updatedProduct.price,
    },
  });

  if (!product) {
    return new Response(JSON.stringify({ message: 'Product not found' }), { status: 404 });
  }

  return new Response(JSON.stringify(product), { status: 200 });
}
```

- Function PUT digunakan untuk memperbarui data produk yang sudah ada di database berdasarkan id.
- Data yang diterima dari klien (dalam request.body) diambil menggunakan request.json().
- prisma.product.update() memperbarui data produk sesuai dengan nilai id yang dikirim.
- Jika produk tidak ditemukan, fungsi ini mengembalikan pesan error Product not found dengan status 404.
- Jika berhasil, data produk yang diperbarui dikembalikan dalam format JSON dengan status 200 OK.

```
// Menghapus Produk
Codeium: Refactor | Explain | X
export async function DELETE(request) {
  const { id } = await request.json();

  try { ...
} catch (error) {
  return new Response(JSON.stringify({ message: 'Product not found' }), { status: 404 });
}
}
```

- Function DELETE digunakan untuk menghapus produk dari database berdasarkan id.
- Data id yang dikirim oleh klien diambil menggunakan request.json().
- prisma.product.delete() menghapus produk yang sesuai dengan id tersebut.
- Jika produk tidak ditemukan, try-catch menangkap error, lalu mengembalikan pesan error dengan status 404.
- Jika berhasil, fungsi ini mengembalikan pesan sukses Product deleted dengan status 200 OK.