

Nama : Deo Cahyo Anggoro
NPM : 22312168

Dokumentasi Project

The screenshot shows the VS Code interface with the following details:

- Explorer View:** Shows the project structure. A file named `schema.prisma` is selected and highlighted in blue. Other files visible include `.next`, `app`, `favicon.ico`, `# globals.css`, `layout.tsx`, `page.tsx`, `node_modules`, and `prisma`.
- Code Editor:** The current file is `schema.prisma`. The code content is as follows:

```
3 // Looking for ways to speed up your queries, or scale easily with your serverless or edge f
4 // Try Prisma Accelerate: https://pris.ly/cli/accelerate-init
5
6 Generate
7 generator client {
8   provider = "prisma-client-js"
9 }
10
11 datasource db {
12   provider = "postgresql"
13   url      = env("DATABASE_URL")
14 }
15
```

- Terminal:** Shows the command history and output of the terminal commands run in the project directory:

 - `tikayesikristiani@tika project % npm install prisma --save-dev`
 - `added 7 packages, and audited 150 packages in 14s`
 - `34 packages are looking for funding`
 - `run `npm fund` for details`
 - `found 0 vulnerabilities`
 - `tikayesikristiani@tika project % npx prisma init`
 - `Your Prisma schema was created at prisma/schema.prisma`

Menginstall prisma dengan cara mengetik `npm install prisma --save-dev` di terminal dan `npx prisma init` untuk membuat file `schema.prisma` dan `.env`

The screenshot shows two open files in VS Code:

- schema.prisma:** The code editor shows the Prisma schema definition. An arrow points from the word `db` in the code to the `schema.prisma` tab in the title bar.
- .env:** The code editor shows the environment variables. An arrow points from the `.env` tab in the title bar to the file content, which includes the `DATABASE_URL` variable.

Karena Saya menggunakan mysql ganti postgresql ke mysql di file `schema.prisma` dan `.env` untuk menghubungkan ke database mysql
Penjelasan di file `.env` root adalah usernya dan localhost:3306 adalah portnya

Menginstall DaisyUI dengan cara mengetik `npm i -D daisyui@latest` di terminal, `@latest` maksudnya adalah versi terbaru dari daisyui

```
rs tailwind.config.ts > [⚙️] default
  3   export default {
  4     ...
  5     ...
  6     ...
  7     ...
  8     ...
  9     ...
 10     ...
 11     ...
 12     ...
 13     ...
 14     ...
 15     ...
 16     },
 17     plugins: [
 18       require('daisyui'),
 19     ],
 20   }
 21 }
```

Lalu masukan require('daisyui'), kedalam plungins di file tailwind.config.ts untuk menambahkan daisyui

layout.tsx M ●

app > layout.tsx > RootLayout

```
6  export default function RootLayout({  
11    return [  
12      <html lang="en" data-theme="lemonade">  
13        <body>  
14          {/* Area Header */}  
15          <header></header>          ←  
16  
17          {/* Area Content */}  
18          <section>←  
19            {children}  
20          </section>  
21  
22          {/* Area Footer */}←  
23          <footer></footer>          You, 3 minutes ago • Uncommitted changes  
24        </body>  
25      </html>  
26    ]  
27 }
```

Menambahkan Header, Section, dan Footer bertujuan untuk memisahkan area header, conten, dan footer untuk menciptakan struktur yang lebih terorganisir dan semantic

```
/* Area Footer */  
<footer>  
Copyright © 2025 – Manusia Biasa Team You, 1 second ago • U  
</footer>
```

Menambahkan text copyright di footer, © adalah logo copyright

```
> ⚙️ layout.tsx > ...  
You, 1 minute ago | 3 authors (You and others)  
// import file "globals.css"  
import "./globals.css" You, 1
```

Mengimport globals.css di layout.tsx

```
/* Area Footer */  
<footer className="flex justify-center bg-blue-700 text-white py-2.5">  
Copyright © 2025 – Manusia Biasa Team  
</footer>
```



Memperbaiki tampilan footer

flex justify-center
agar textnya berada di tengah
bg-blue-700
membuat background berwarna biru
text-white
membuat text berwarna putih
py-2.5
memberikan padding sebesar 10 px

```
<html lang="en" data-theme="lemonade">
  <body>
    {/* Area Header */}
    <header>
      <Link href="/">Ecommerce</Link>
      <nav>
        <Link href="/">Home</Link>
        <Link href="/">About</Link>
        <Link href="/">Products</Link>
        <Link href="/">Contact</Link>
      </nav>
    </header>
```

Menambahkan Link yang berguna untuk tombol navigasi

```
/* Area Header */
<header>
  <Link href="/">Ecommerce</Link>
  <nav className="flex justify-end">
    <Link href="/" className="mr-2.5">Home</Link>
    <Link href="/" className="mx-2.5">About</Link>
    <Link href="/" className="mx-2.5">Products</Link>
    <Link href="/" className="ml-2.5 mr-5">Contact</Link>
  </nav>
</header>
```

You, 2 days ago • deo: menambahkan area header, conten, dan footer

Menambahkan flex justify-end agar tombol berada di sebelah kanan

Home About Products Contact

Dan menambahkan margin agar tombolnya memiliki jarak
mr untuk margin kanan
ml untuk margin kiri
dan 2.5 adalah ukurannya sebesar 10px

```
/* Area Header */
<header className="flex justify-between items-center px-5 py-3">
  <Link href="/">Ecommerce</Link>
  <nav className="flex justify-end">
    <Link href="/" className="mr-2.5">Home</Link>
    <Link href="/" className="mx-2.5">About</Link>
    <Link href="/" className="mx-2.5">Products</Link>
    <Link href="/" className="ml-2.5">Contact</Link>
  </nav>
</header>
```

You, 1 hour ago • deo : membuat tombol di navbar



`flex justify-between items-center px-5 py-3`

Berguna agar tata letaknya sejajar
 flex justify-between Mengatur jarak antar flex items secara horizontal sehingga item pertama berada di sisi kiri, item terakhir di sisi kanan
 items-center Menyelaraskan (align) flex items secara vertikal di tengah-tengah flex container
 px-5 untuk mengatur padding kiri dan kanan sebesar 20px
 py-3 untuk mengatur padding atas dan bawah sebesar 12px

```
model tb_barang{
  id Int @id @default(autoincrement())
  nama_barang String @db.VarChar(50)
  deskripsi String @db.VarChar(100)
  harga Float @db.Double
  kategori String @db.VarChar(20)
  status status
}

enum status {
  Y
  N
}
```

Menambah tabel barang di schema.prisma
 Int adalah tipe data integer untuk bilangan bulat
 @id menandai bahwa itu adalah primary key

@default(autoincrement()) untuk membuat nilai secara otomatis

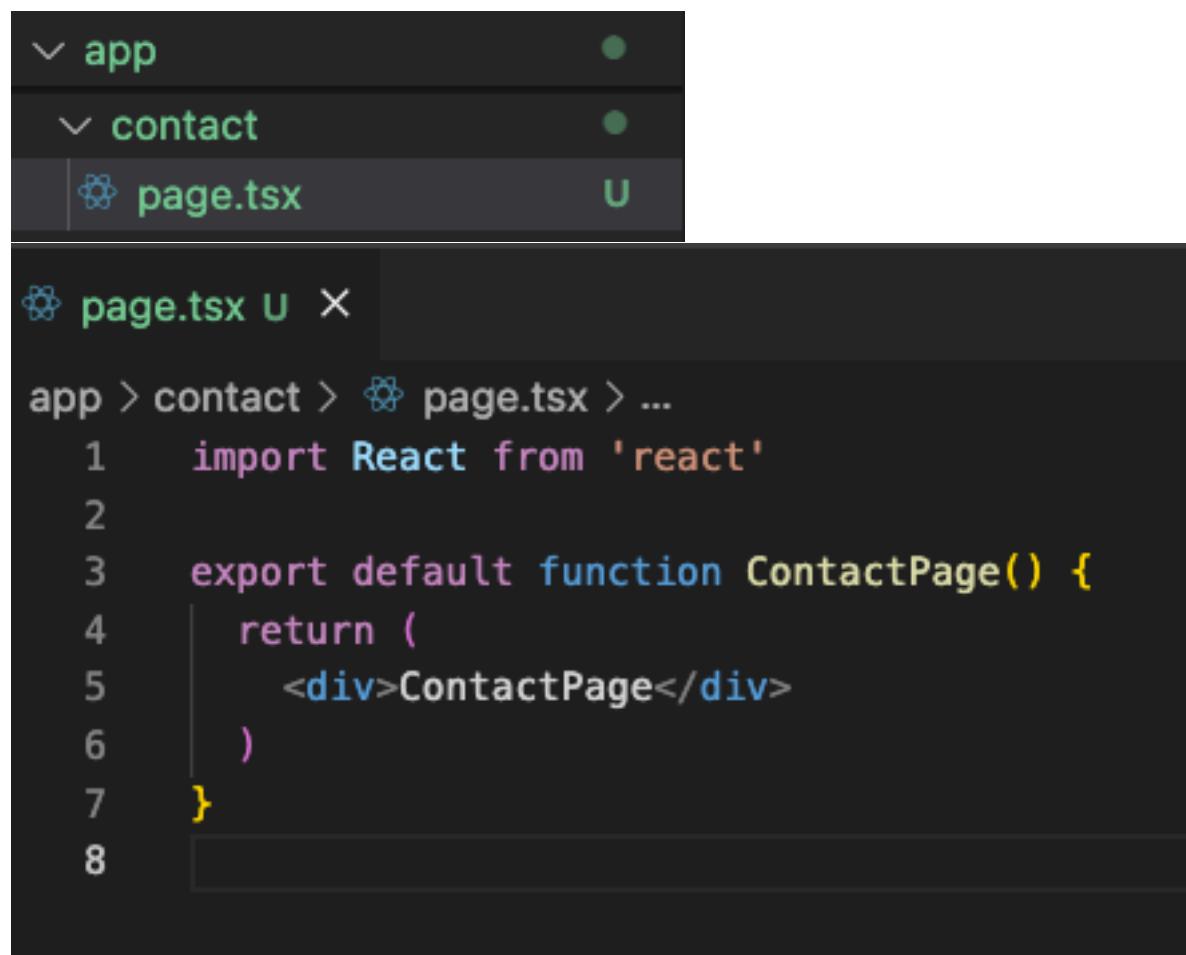
String adalah tipe data untuk teks

@db.VarChar(50) untuk tipe data ini di database adalah varchar dan untuk nilainya adalah panjang maksimum karakter

Float adalah tipe data untuk bilangan pecahan

@db.Double untuk tipe data di database adalah double

Untuk melakukan migrate ketikkan npx prisma migrate dev di terminal lalu beri nama lalu prisama akan membuat migration file



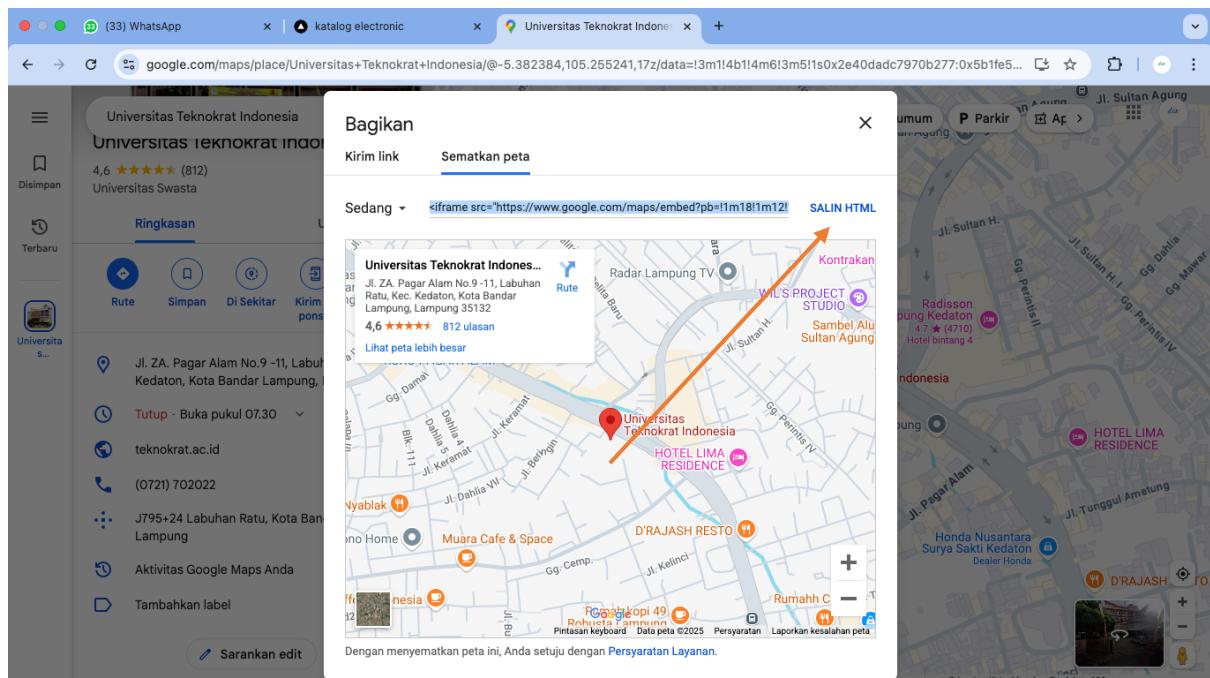
The screenshot shows a code editor with a dark theme. At the top, there is a file tree:

- app
- └ contact
- page.tsx

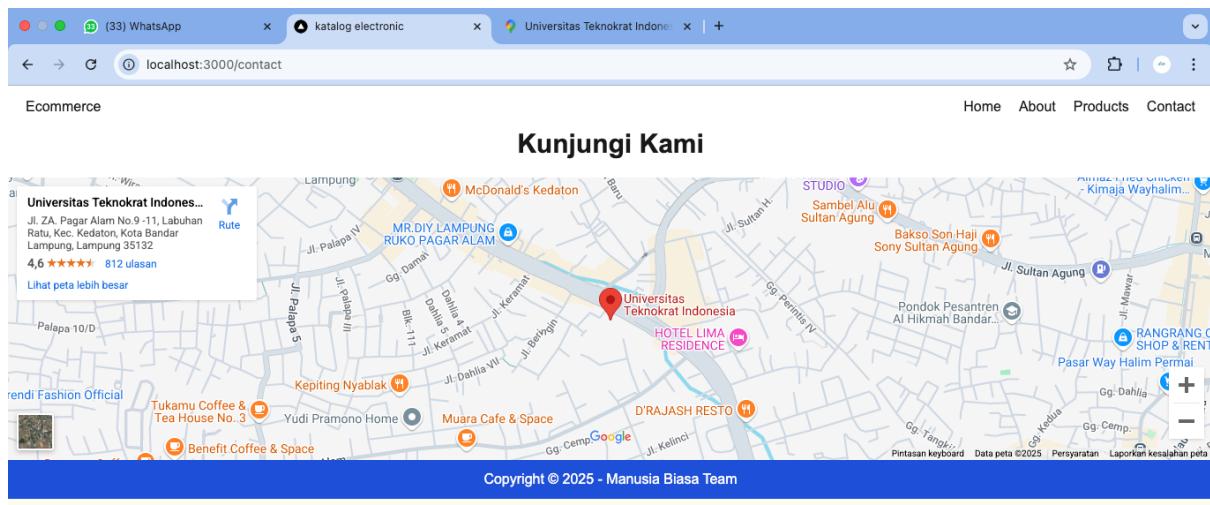
The file "page.tsx" is selected and open in the editor. The code inside is:

```
app > contact > page.tsx > ...
1 import React from 'react'
2
3 export default function ContactPage() {
4   return (
5     <div>ContactPage</div>
6   )
7 }
8
```

Buat folder baru didalam folder app dengan nama contact lalu buat file dengan nama page.tsx
Lalu ketikkan rfc lalu enter dan ganti page menjadi ContactPage



Untuk menambah map dapat langsung mengcopynya dari google map
 Lalu atur ukuran mapnya menggunakan tailwind dengan cara mengganti defaultnya width dan height dengan className="" w dan h untuk width dan height
 Saya menggunakan w-full h-80 untuk width full dari kanan ke kiri atau 100% dan height 20rem atau 320px



Lalu menambah text judul menggunakan <h1></h1> dan diatur menggunakan tailwind
`<h1 className="text-3xl font-bold mb-5 text-center">Kunjungi Kami</h1>`

text-3xl untuk mengatur font size sebesar 30px dan line height sebesar 36px
 font bold untuk menebalkan textnya
 mb-5 untuk memberikan margin dibawah sebesar 20px
 text center agar text berada ditengah

```

    22 You, 1 second ago | 1 author (You)
    23 model tb_barang{
    24   id Int @id @default(autoincrement())
    25   nama_barang String @db.VarChar(50)
    26   deskripsi String @db.VarChar(100)
    27   harga Float @db.Double
    28   kategori String @db.VarChar(20)
    29   link_product String @db.VarChar(100)
    30   status status
    31 }
    32
    33 enum status {
    34   Y
    35   N
    36 }

```

Untuk menambah field di prisma pertama masuk ke file schema.prisma lalu tambahkan field ke dalam tabel

Disini saya menambahkan link_product dengan tipe data string varchar sebanyak dengan max 100 huruf

```

    * History restored
    • tikayesikristiani@tika project % brew services start mysql
    => Successfully started `mysql` (label: homebrew.mxcl.mysql)
    ○ tikayesikristiani@tika project % npx prisma migrate dev

```

Setelah itu lakukan migrate dengan cara mengetikan npx prisma migrate dev

```

    • tikayesikristiani@tika project % brew services start mysql
    => Successfully started `mysql` (label: homebrew.mxcl.mysql)
    ○ tikayesikristiani@tika project % npx prisma migrate dev
    Environment variables loaded from .env
    Prisma schema loaded from prisma/schema.prisma
    Datasource "db": MySQL database "db_katalog" at "localhost:3306"
    ? Enter a name for the new migration: > tb_barang_rev02

```

Lalu beri nama migration datanya, disini saya menamainya tb_barang_rev02 karena ini adalah revisi ke 2

```

    PROBLEMS OUTPUT TERMINAL PORTS GITLENS DEBUG CONSOLE

    The following migration(s) have been created and applied from new schema changes:
    migrations/
      20250113052038_tb_barang_rev02/
        migration.sql

    Your database is now in sync with your schema.

    ✓ Generated Prisma Client (v6.1.0) to ./node_modules/@prisma/client in 271ms

    ○ tikayesikristiani@tika project %

```

Setelah selesai maka akan muncul file migration yang dibuat tadi dan fieldnya akan masuk ke database

```
EXPLORER            ...
PROJECT          ...
app
  models
    ...
  product
    page.tsx
    favicon.ico
    globals.css
    layout.tsx
page.tsx .../contact      schema.prisma      page.tsx .../product X
app > product > page.tsx > ProductPage
You, 1 hour ago | 1 author (You)
1 import React from 'react'
2
3 export default function ProductPage() {
4   return (
5     <div>ProductPage</div>
6   )
7 }
```

Buat folder baru dengan nama product dan file page.tsx untuk nantinya dijadikan halaman product

Lalu ketikkan rfc untuk mempermudah membuat export default function

```
<Link href="/">Home</Link>
<Link href="/about">About</Link>
<Link href="/product">Products</Link>
<Link href="/contact">Contact</Link>
</nav>
</header>
```

Lalu tambahkan /product di rootlayout agar navigasinya mengarah ke file page yang ada di dalam folder product

```
use server";
import { PrismaClient } from "@prisma/client";
//buat variabel "prisma"
const prisma = new PrismaClient();
// buat fungsi untuk ambil data barang
export async function getData() {}
```

Buka file barang.ts lalu membuat export async function getData(){}{}

Export digunakan agar file ini dapat di impor di file lain

Async menunjukkan bahwa fungsi ini bersifat asynchronous

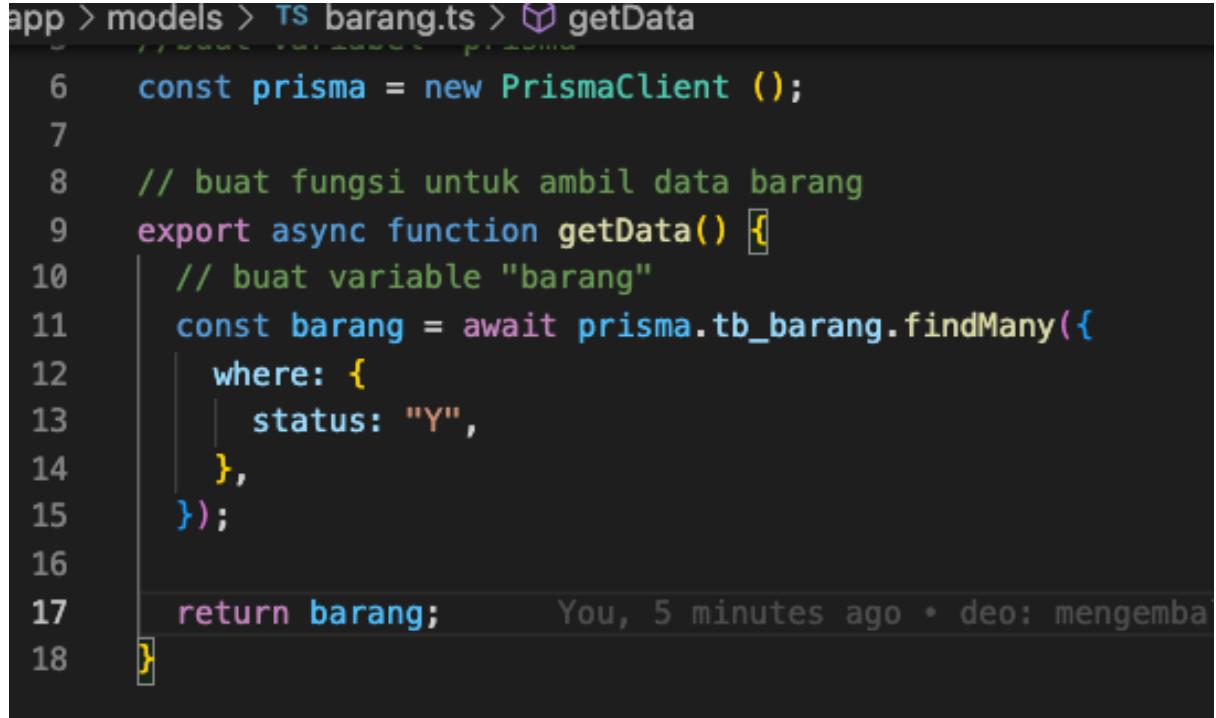
function getData ini digunakan untuk mengambil data dari database

```
app > models > barang.ts > getData > barang
You, 4 minutes ago | 2 authors (You and one other)
1 "use server";
2
3 import { PrismaClient } from "@prisma/client";
4
5 //buat variabel "prisma"
6 const prisma = new PrismaClient ();
7
8 // buat fungsi untuk ambil data barang
9 export async function getData() {
10   // buat variable "barang"
11   const barang = await prisma.tb_barang.findMany();
12 }
```

Membuat variable barang const barang = await prisma.tb_barang.findMany(); digunakan untuk mengambil semua data dari tabel tb_barang dalam database menggunakan Prisma

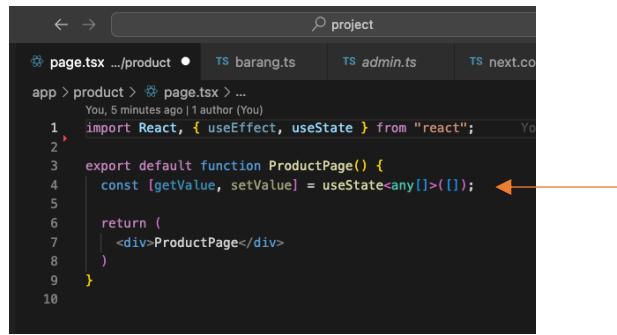
```
app > models > barang.ts > getData > barang > where
You, 12 minutes ago | 2 authors (You and one other)
1 "use server";
2
3 import { PrismaClient } from "@prisma/client";
4
5 //buat variabel "prisma"
6 const prisma = new PrismaClient ();
7
8 // buat fungsi untuk ambil data barang
9 export async function getData() {
10   // buat variable "barang"
11   const barang = await prisma.tb_barang.findMany({
12     where: {
13       status: "Y",
14     },
15   });
16 }
```

Where: {status:"Y",} ini maksudnya data yang akan ditampilkan hanya data yang enum statusnya adalah Y



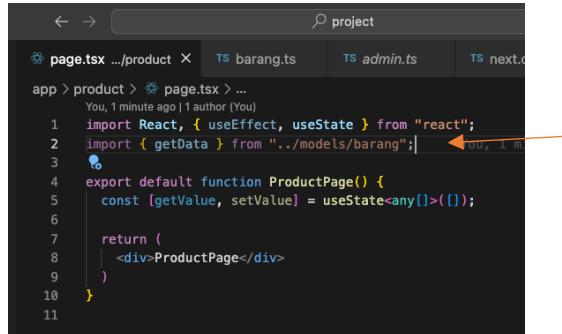
```
app > models > TS barang.ts > ⌂ getData
6   const prisma = new PrismaClient();
7
8   // buat fungsi untuk ambil data barang
9   export async function getData() {
10     // buat variable "barang"
11     const barang = await prisma.tb_barang.findMany({
12       where: {
13         status: "Y",
14       },
15     });
16
17     return barang;
18 }
```

Return barang; digunakan untuk mengembalikan data yang telah diambil dari database
Dan kenapa harus dikembalikan? Karena data ini nantinya akan dibuat untuk ditampilkan



```
app > product > page.tsx > ...
1 import React, { useEffect, useState } from "react";
2
3 export default function ProductPage() {
4   const [getValue, setValue] = useState<any>([]); ←
5
6   return (
7     <div>ProductPage</div>
8   )
9 }
10
```

Bagian kode ini adalah bagian dari komponen React yang menggunakan Hooks (useState dan useEffect)



```
1 import React, { useEffect, useState } from "react";
2 import { getData } from "../models/barang";
```

Mengimport get data dari file barang.ts yang ada didalam folder models



```
1 import React, { useEffect, useState } from "react";
2 import { getData } from "../models/barang";
3
4 export default function ProductPage() {
5   //buat hook dengan "use state"
6   const [getValue, setValue] = useState<any>([]);
7
8   // buat fungsi untuk panggil "getData"
9   async function fetchData() {
10     const data = await getData();
11     setValue(data);
12   }
13
14 }
```

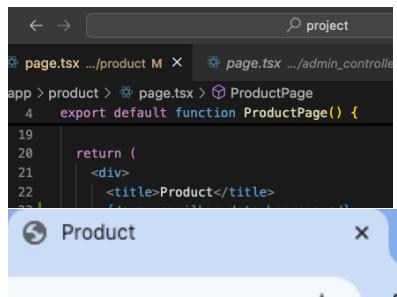
Bagian ini adalah fungsi asynchronous yang mengambil data menggunakan getData(), menunggu hasilnya dengan await, dan kemudian memperbarui state getValue menggunakan setValue(data) dengan data yang diperoleh.



```
// buat fungsi untuk panggil "getData"
async function fetchData() {
  const data = await getData();
  setValue(data);
}

// hook dengan "use effect"
useEffect(() => {
  // panggil fungsi "getData"
  fetchData();
}, []);
```

Bagian ini memanggil fungsi fetchData sekali saat komponen pertama kali dirender, karena array dependency kosong ([]), yang berarti efek hanya dijalankan sekali.



```
1 export default function ProductPage() {
2   return (
3     <div>
4       <title>Product</title>
```

Menambahkan title agar saat di menu product di bagian title web tertulis title

```
page.tsx .../product M
app > product > page.tsx
  You, 3 seconds ago | author [X]
  1 "use client";
  2 import React, { useEffect, useState } from "react";
  3
  4 export default function ProductPage() {
  5   const [data, setData] = useState<any>([]);
  6
  7   useEffect(() => {
  8     const fetchData = async () => {
  9       const response = await fetch("https://api.example.com/products");
 10       const jsonData = await response.json();
 11       setData(jsonData);
 12     };
 13
 14     fetchData();
 15   }, []);
 16
 17   return (
 18     <div>
 19       <title>Product</title>
 20       /* menampilkan data barang */
 21       {Object.values(getValue).map((data: any, index: number) => (
 22         <div key={index}>
 23           <div>
 24             {data.nama_barang} - {data.deskripsi} - {data.harga}
 25           </div>
 26         </div>
 27       ))}
 28     </div>
 29   );
 30 }
 31
 32 }
```

"use client" memastikan komponen dirender di browser, bukan di server, biasanya untuk menggunakan fitur React seperti useState atau useEffect.

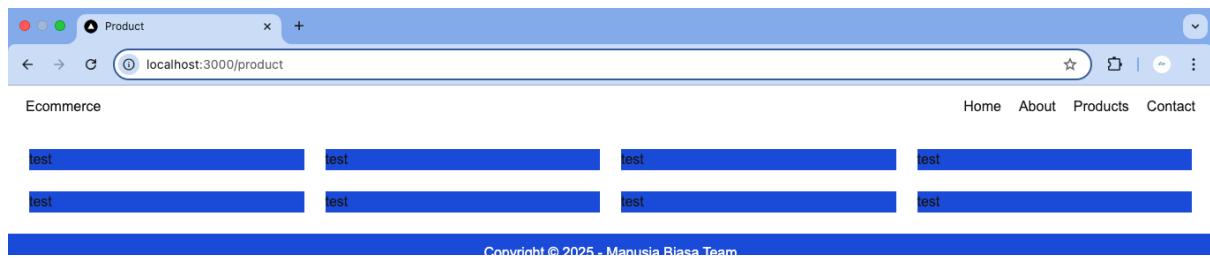
```
page.tsx .../product M
app > product > page.tsx
  You, 1 second ago * Uncommitted changes
  1 "use client";
  2 import React, { useEffect, useState } from "react";
  3
  4 export default function ProductPage() {
  5   const [data, setData] = useState<any>([]);
  6
  7   useEffect(() => {
  8     const fetchData = async () => {
  9       const response = await fetch("https://api.example.com/products");
 10       const jsonData = await response.json();
 11       setData(jsonData);
 12     };
 13
 14     fetchData();
 15   }, []);
 16
 17   return (
 18     <div>
 19       <title>Product</title>
 20       /* menampilkan data barang */
 21       {Object.values(getValue).map((data: any, index: number) => (
 22         <div key={index}>
 23           <div>
 24             {data.nama_barang} - {data.deskripsi} - {data.harga}
 25           </div>
 26         </div>
 27       ))}
 28     </div>
 29   );
 30 }
 31
 32 }
```

Kode ini digunakan untuk menampilkan data barang dari state getValue. Data diakses dengan Object.values(getValue) lalu di-loop menggunakan map, menampilkan nama_barang, deskripsi, dan harga setiap item dalam elemen <div>.

```
page.tsx .../product
page.tsx .../contact
  1 "use client";
  2 import React, { useEffect, useState } from "react";
  3 import { getData } from "../models/barang";
  4 import Image from "next/image";
  5
  6 export default function ProductPage() {
  7   //buat hook dengan "use state"
  8   const [getValue, setValue] = useState<any>([]);
  9
 10   // buat fungsi untuk panggil "getData"
 11
 12   useEffect(() => {
 13     const fetchData = async () => {
 14       const response = await fetch("https://api.example.com/products");
 15       const jsonData = await response.json();
 16       setValue(jsonData);
 17     };
 18
 19     fetchData();
 20   }, []);
 21
 22   return (
 23     <div>
 24       <title>Product</title>
 25       /* menampilkan data barang */
 26       {getValue.map((data: any, index: number) => (
 27         <div key={index}>
 28           <div>
 29             {data.nama_barang} - {data.deskripsi} - {data.harga}
 30           </div>
 31         </div>
 32       ))}
 33     </div>
 34   );
 35 }
```

Kode ini tidak memerlukan Object.values() karena getValue sudah diinisialisasi sebagai array dengan useState<any>([])

Jika Object.values() tetap ada meskipun getValue sudah berupa array, itu tidak akan menyebabkan error tetapi menjadi redundan dan tidak efisien



Membuat grid menggunakan tailwind.css

```
<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-6 p-6">
```

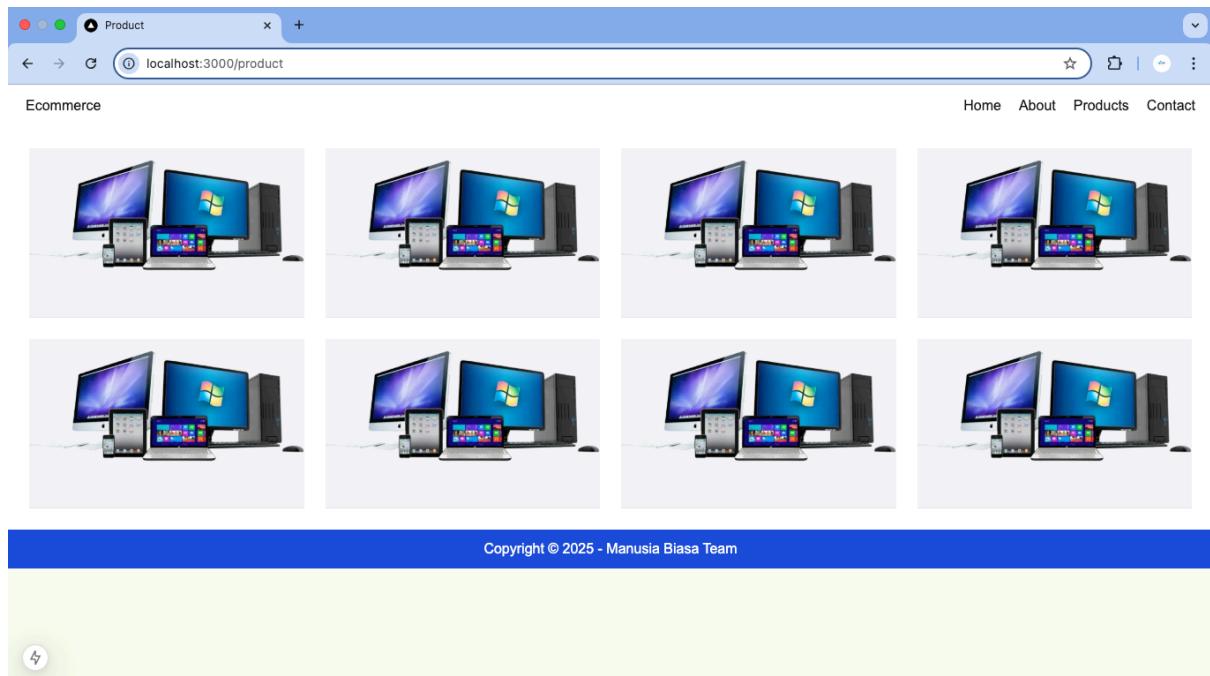
grid-cols-1: Mengatur grid untuk memiliki 1 kolom pada layar kecil (default).

md:grid-cols-2: Pada layar berukuran medium (tablet) atau lebih besar, grid akan memiliki 2 kolom.

lg:grid-cols-4: Pada layar berukuran large (desktop) atau lebih besar, grid akan memiliki 4 kolom.

gap-6: Mengatur jarak antara elemen-elemen grid (baik baris maupun kolom) sebesar 1.5rem (atau 24px).

p-6: Memberikan padding di seluruh sisi container sebesar 1.5rem (atau 24px). I



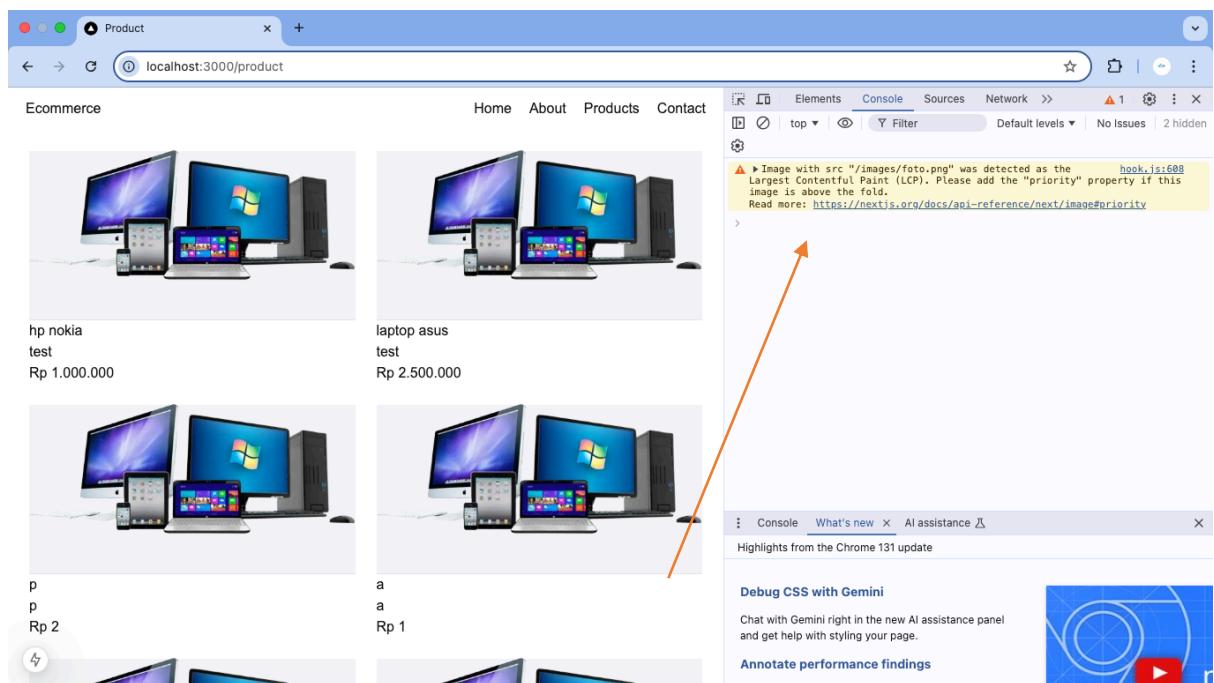
Buat tampilan untuk display

```
<div className="w-full h-48 bg-gray-100 flex items-center justify-center border-b border-gray-200">
<Image
src="/images/foto.png"
alt={data.nama_barang}
width={500}
height={300}
/>
```

Disini menggunakan tailwind
w-full = width: 100%
h-48 = height: 12rem / 192px
bg-gray-100 = background-color: (abu-abu terang)
flex = display: flex
items-center = align-items: center
justify-center = justify-content: center
border-b = border-bottom: 1px solid
border-gray-200 = border-color: (abu-abu terang)

```
<div>
  /* Nama Barang */
  <h2>
    {data.nama_barang}
  </h2>
  /* Deskripsi */
  <p>
    {data.deskripsi}
  </p>
  /* Harga */
  <p>
    Rp {data.harga.toLocaleString("id-ID")}
  </p>
</div>
```

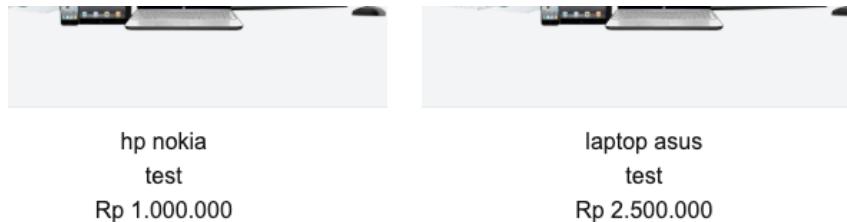
Menampilkan nama barang, deskripsi, dan harga yang diambil dari database
Untuk bagian haraga menggunakan toLocaleString("id-ID"),
toLocaleString("id-ID") digunakan untuk format angka agar sesuai dengan format lokal
Indonesia
misalnya Rp 1000000000 menjadi Rp 1.000.000.000



Tambahkan Priority pada komponen <Image>

```
<Image  
src="/images/foto.png"  
alt={data.nama_barang}  
width={500}  
height={300}  
priority  
/>
```

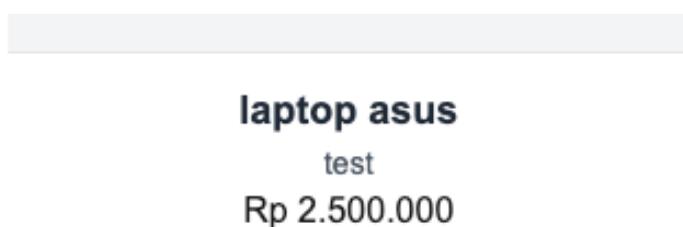
Di Next.js, atribut priority pada komponen <Image> digunakan untuk memberi tahu Next.js agar memprioritaskan pemuatannya.



Tambahkan tailwind className="p-3 text-center" untuk membuat padding sebesar 12px dan membuat teksnya ketengah



Dan tambahkan className="text-lg font-semibold text-gray-800" untuk membuat nama barang menjadi semi bold menggunakan font-semibold
Lalu text-lg untuk membuat font size 18px dan line height 28px
Lalu ganti warna text menggunakan text-gray-800



Edit text deskripsinya menggunakan className="text-gray-600 text-sm" untuk mengganti warna dan nerubah ukurannya menjadi font size 14px dan line height 20px

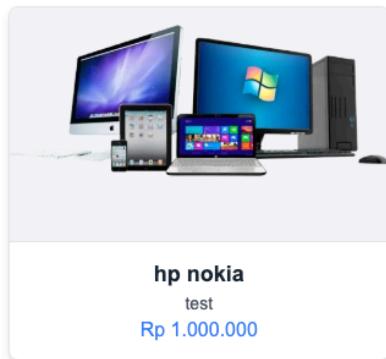
laptop asus

test

Rp 2.500.000

Dan untuk tampilan harganya className="text-base font-medium text-blue-500"

- text-base: font-size: 1rem line-height: 1.5rem
- font-medium: font-weight: 500
- text-blue-500: color: biru



hp nokia

test

Rp 1.000.000



laptop asus

test

Rp 2.500.000

Buat bentuk card menggunakan className="bg-white border border-gray-200 rounded-lg shadow-md overflow-hidden hover:shadow-lg transition-shadow duration-300"
penjelasan

- bg-white**: background-color: putih;
- border**: border-width: 1px;
- border-gray-200**: border-color: abu abu;
- rounded-lg**: border-radius: 0.5rem;
- shadow-md**: box-shadow: 0 4px 6px -1px rgba(0, 0, 0, 0.1), 0 2px 4px -2px rgba(0, 0, 0, 0.1);
- overflow-hidden**: overflow: hidden;
- hover:shadow-lg**: (on hover) box-shadow: 0 10px 15px -3px rgba(0, 0, 0, 0.1), 0 4px 6px -4px rgba(0, 0, 0, 0.1);
- transition-shadow**: transition-property: box-shadow;
- duration-300**: transition-duration: 300ms;



Tambah tombol menggunakan Link, untuk menulis link ketik dulu <Link Agar link diimport lalu href={data.link_product || "#"} target="_blank" untuk mengarahkan link dari link yang diambil dari database dan target blank untuk membuat tab baru ke link dan untuk tailwindnya :

- **bg-blue-600**: background-color: biru;
- **text-white**: color: putih;
- **py-1**: padding-top: 0.25rem; padding-bottom: 0.25rem;
- **px-3**: padding-left: 0.75rem; padding-right: 0.75rem;
- **rounded-md**: border-radius: 0.375rem;
- **text-sm**: font-size: 0.875rem; line-height: 1.25rem;
- **hover:bg-blue-700**: (on hover) background-color: biru lebih tua;



Mengganti src gambar dengan data.image_url bertujuan untuk membuat gambar sesuai gambar yang di upload
data.image_url adalah alamat gambarnya yg tertulis didatabase

```

<div className="w-full aspect-square bg-gray-100 border-b border-gray-200 overflow-hidden">
  <Image
    src={data.image_url}
    alt={data.nama_barang}
    width={500}
    height={300}
    priority
    className="object-cover w-full h-full" />
  You, 12 hours ago • deo: menambah select dari da

```

Ganti tailwind di bagian gambar menggunakan aspect-square untuk membuat gambar product yg ditampilkan memiliki rasio gambar 1:1 dan overflow-hidden untuk menyembunyikan jika ada gambar yang terpotong

className="object-cover w-full h-full"

- object-cover:** Memastikan gambar memenuhi elemen tanpa distorsi dengan memotong bagian luar.
- w-full:** Lebar gambar mengikuti 100% elemen induk.
- h-full:** Tinggi gambar mengikuti 100% elemen induk.

Select with border

Preview HTML JSX

Who shot first?

Select with border

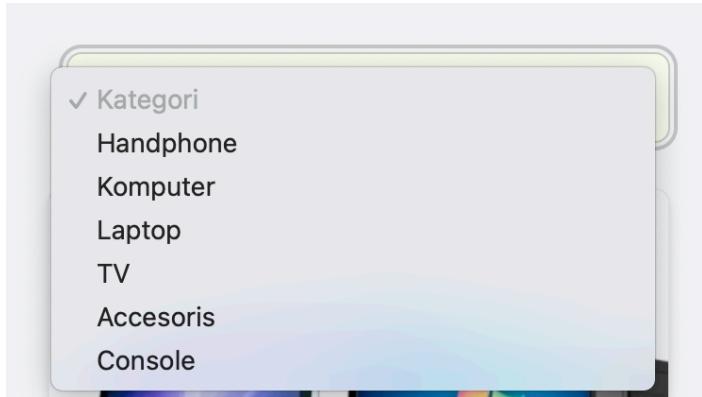
Preview HTML **JSX**

```

<select className="select select-bordered w-full max-w-xs">
  <option disabled selected>Who shot first?</option>
  <option>Han Solo</option>
  <option>Greedo</option>
</select>

```

Ambil select dari daisy ui untuk membuat pilih kategori



Lalu ubah bagian optionnya sesuai nama kategori yang mau ditampilkan

```
o > models > ts barang.ts > ⚙ getData
You, 12 hours ago | 2 authors (maeche11 and one other)
1 "use server";
2
3 import { PrismaClient } from "@prisma/client";
4
5 //buat variabel "prisma"
6 const prisma = new PrismaClient ();
7
8 // buat fungsi untuk ambil data barang
9 export async function getData(category: string | null = null) {
  // buat variable "barang"
10  const barang = await prisma.tb_barang.findMany({
11    where: {
12      status: "Y",
13      ...(category && { kategori: category }),
14    },
15  });
16
17  return barang;
18}
19
```

Tambahkan Kategori ke barang.ts untuk nantinya dibuat menampilkan barang sesuai kategori yang dipilih

```
export default function ProductPage() {
  //buat hook dengan "use state"
  const [getValue, setValue] = useState<any>([]);
  const [selectedCategory, setSelectedCategory] = useState<string | null>(null);
```

Tambahkan state untuk kategori

Kode ini mendeklarasikan sebuah state di React menggunakan hook useState.

selectedCategory adalah variabel yang menyimpan nilai kategori yang dipilih, yang bisa berupa string atau null. Fungsi setSelectedCategory digunakan untuk mengubah nilai dari selectedCategory. Tipe data state ini adalah string atau null, yang berarti kategori bisa berupa teks atau tidak ada pilihan sama sekali (null).

```
// buat fungsi untuk panggil "getData" dengan filter kategori
async function fetchData(category: string | null = null) {
  const data = await getData(category);
  setValue(data);
}

You, 2 weeks ago • deo: memanggil fungsi fetchData
```

Lalu tambahkan filter kategori

```
// Fungsi untuk mengubah kategori
function handleCategoryChange(event: React.ChangeEvent<HTMLSelectElement>) {
  const category = event.target.value;
  setSelectedCategory(category);
  fetchData(category); // Ambil data berdasarkan kategori yang dipilih
}
```

Buat fungsi untuk mengubah kategori

Kode ini mendefinisikan fungsi handleCategoryChange yang menangani perubahan kategori. Ketika kategori dipilih dari elemen <select>, fungsi ini akan:

1. Mengambil nilai kategori yang dipilih (event.target.value).
2. Mengubah state selectedCategory menggunakan setSelectedCategory dengan nilai kategori yang dipilih.
3. Memanggil fungsi fetchData untuk mengambil data yang relevan berdasarkan kategori yang dipilih.

```
<div className="flex justify-center">
  {/* Dropdown Kategori */}
  <select
    className="select select-bordered w-full max-w-xs mr-3"
    value={selectedCategory || ""}
    onChange={handleCategoryChange}>
    <option value="" disabled>
      Kategori
    </option>
    <option value="Handphone">Handphone</option>
    <option value="Komputer">Komputer</option>
    <option value="Laptop">Laptop</option>
    <option value="TV">TV</option>
    <option value="Accesoris">Accesoris</option>
    <option value="Console">Console</option>
  </select>
```

Lalu isi value sesuai barang yang mau di filter

```
# Text input with border
```

Preview HTML JSX



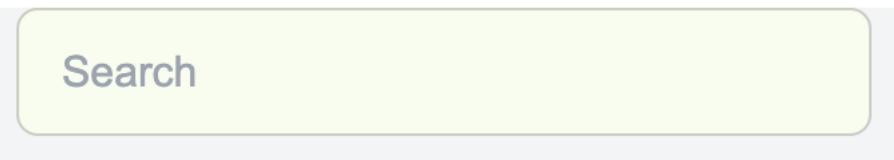
```
# Text input with border
```

Preview HTML JSX

```
<input type="text" placeholder="Type here" className="input input-bordered w-full max-w-xs" />
```



Ambil text input dari daisy ui untuk membuat search



Lalu ubah textnya menjadi search dibagian placeholder="Type here"
Menjadi placeholder="Search"

```
export default function ProductPage() {
  //buat hook dengan "use state"
  const [getValue, setValue] = useState<any>([]);
  const [selectedCategory, setSelectedCategory] = useState<string | null>(null);
  const [searchTerm, setSearchTerm] = useState<string>("");
```

Buat state untuk search

```
// Fungsi untuk mengubah nilai search
function handleSearchChange(event: React.ChangeEvent<HTMLInputElement>) {
  setSearchTerm(event.target.value);
}

// Filter data barang berdasarkan search term
const filteredData = getValue.filter(data =>
  data.nama_barang.toLowerCase().includes(searchTerm.toLowerCase())
);
```

Kode ini mendefinisikan dua bagian:

1. **Fungsi handleSearchChange:** Fungsi ini menangani perubahan input pencarian. Ketika nilai input berubah, fungsi ini memperbarui state searchTerm dengan nilai baru dari input.
2. **Filter Data:** filteredData adalah array yang berisi barang yang disaring. Data barang difilter berdasarkan apakah nama barang (data.nama_barang) mengandung kata kunci pencarian (searchTerm), tanpa memperhatikan huruf besar/kecil (toLowerCase()) digunakan untuk pencocokan yang tidak sensitif terhadap kapitalisasi).

```
/* Search */
<input
  type="text"
  placeholder="Search"
  className="input input-bordered w-full max-w-xs ml-3"
  value={searchTerm}
  onChange={handleSearchChange} You, 4 hours ago • deo : menambah
/>
::
```

Tambahkan value dan onchange

Nilai input diikat dengan state searchTerm, sehingga setiap perubahan pada input akan memperbarui state tersebut melalui handleSearchChange.

Fungsi onChange digunakan untuk menangani perubahan nilai input.

```
<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-2
  /* menampilkan data barang */
  {filteredData.map((data: any, index: number) => (
    <div
      key={index}
```

Lalu tampilkan data menggunakan filteredData

filteredData adalah hasil dari penyaringan berdasarkan kata kunci pencarian (searchTerm).