

Nama : Muhammad Syafiq Nasrullah
NPM : 22312140

Dokumentasi Pemrograman WEB 2

Project Akhir Vanguard

1. Update Footer



Penjelasan :

Membuat footer dengan tag footer dengan warna background biru-500, lalu untuk tulisan dengan text-center agar tetap di Tengah, dan tulisan copyright dapat di klik sebagai link.

2. Install Fontawesome

```
PS C:\Users\user\Documents\PERKULIAHAN MSN\KULIAH TEKNOKRAT MSN\PW2-22C-MSN\vanguard> npm i --save @fortawesome/fontawesome-svg-core
PS C:\Users\user\Documents\PERKULIAHAN MSN\KULIAH TEKNOKRAT MSN\PW2-22C-MSN\vanguard> npm i --save @fortawesome/free-solid-svg-icons
>> npm i --save @fortawesome/free-regular-svg-icons
>> npm i --save @fortawesome/free-brands-svg-icons

PS C:\Users\user\Documents\PERKULIAHAN MSN\KULIAH TEKNOKRAT MSN\PW2-22C-MSN\vanguard> npm i --save @fortawesome/react-fontawesome@latest
added 3 packages, and audited 158 packages in 3s
32 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
```

Penjelasan :

Penginstallan fontawesome untuk dapat menggunakan icon secara gratis. Penginstallan dapat dilakukan secara sederhana dengan mengikuti petunjuk penginstallan di terminal, seperti pada gambar diatas

3. Update footer dengan icon social media



Tambah icon dengan cara membuka <https://fontawesome.com/icons> lalu cari icon yang ditambahkan, buka icon tersebut, cari react individual import, copy code contoh: `<FontAwesomeIcon icon={faInstagram} />`. Paste ke dalam codingan. (jangan lupa untuk import library nya)

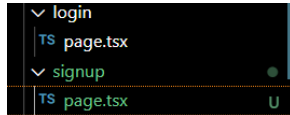
4. Update Navbar



Penjelasan :

Menambahkan Nama web di awal navbar “Barang Second”, lalu memindahkan Link Dashboard dan akun di akhir navbar.

5. Tambah Fitur Sign Up



Penjelasan :

Menambahkan signup folder dalam app, dan membuat file page.tsx

6. Update file signup/page.tsx



Penjelasan :

Menambahkan div sebagai latar putih dibelakang, lalu memasukkan div lagi untuk card sign up dan form email dan password, lalu dibawahnya ada button sign up.

7. Update signup/page.tsx

```
<div className="mb-4">
  <label htmlFor="password" className="block text-sm font-medium text-gray-700">
    Password
  </label>
  <input
    type="password"
    id="password"
    name="password"
    className="mt-1 block w-full px-3 py-2 border border-gray-300 rounded shadow-sm focus:ring-blue-500 focus:border-blue-500"
    required
  />
</div>
```

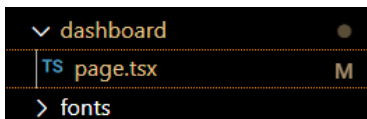
```
{/* link masuk akun jika sudah ada akun. */}
<p className="mt-4 text-sm text-gray-600 text-center">
  Sudah punya akun ? <Link href="/login" className="text-blue-500 hover:underline">Sign In</Link>
</p>
```

The image shows a 'Sign Up' form. It has a title 'Sign Up' at the top. Below the title are two input fields: 'Email' and 'Password'. Both fields are currently empty. Below the 'Password' field is a blue button labeled 'Sign Up'. At the bottom of the form, there is a text link that says 'Sudah punya akun ? Sign In'.

Penjelasan :

Update padding (jarak) antar form dan button, lalu menambahkan text “sudah punya akun ?” dan Link menuju sign in.

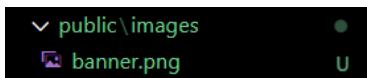
8. Add dashboard/page.tsx



Penjelasan :

Add file page.tsx pada folder dashboard.

9. Add folder public/images



Penjelasan :

Add folder public, lalu didalamnya add folder images, lalu tambahkan image nya, dalam hal ini menambahkan banner.png

10. Update dashboard menambahkan banner

```
export default function dashboardPage() {
  return (
    <div>
      <header>
        <div className="relative h-72 w-full">
          <Image
            className="absolute inset-0 h-full w-full object-cover opacity-70"
            src={"/images/banner.png"}
            alt={"Banner Barang Second"}
            width={1851}
            height={222}
          />
        </div>
      </header>
    </div>
  )
}
```

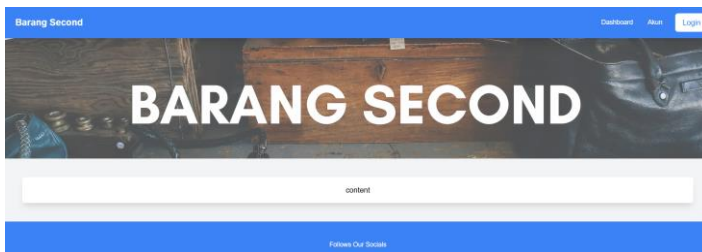


Penjelasan :

Tambahkan header banner dengan cara panggil Image dari public/images/banner.png

11. Update Dashboard untuk content

```
<div className='flex items-center justify-center w-auto h-auto p-10 bg-gray-100'>
  <div className='bg-white p-4 rounded-md shadow-lg w-full'>
    <h2 className='text-black text-center'>
      content
    </h2>
  </div>
</div>
```

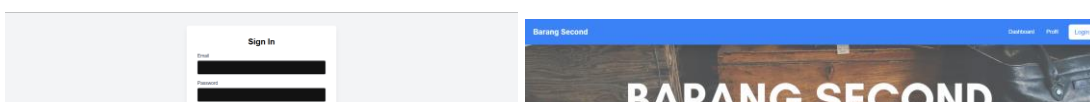


Penjelasan :

Menambahkan div sebagai tempat menaruh content dan didalamnya di isi div dengan padding sisinya sebesar 16 pixel

12. Update letak Navbar

```
(/* navbar */)
<header className='mb-10'>
  {/* navbar */}
  <nav>
    <div className='navbar bg-blue-500 fixed top-0 left-0 w-full z-50 shadow-xl'>
      <div className='navbar-start'>
        <Link href='(/dashboard)' className='navbar-center btn btn-ghost text-xl'>
          Barang Second
        </Link>
      </div>
      <div className='navbar-end lg:flex'>
        <ul className='menu menu-horizontal px-1'>
          <li>
            <Link href='(/dashboard)'>Dashboard</Link>
          </li>
          <li>
            <Link href='(/profil)'>Profil</Link>
          </li>
        </ul>
        <Link href='(/login)' className='p-2'>
          <button className='bg-white text-blue-500 px-4 py-2 rounded-md hover:bg-gray-200'>
            login
          </button>
        </Link>
      </div>
    </div>
  </nav>
</header>
```

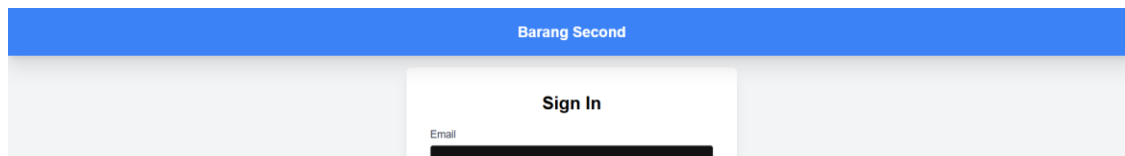


Penjelasan :

Mengubah letak navbar dari rootLayout ke Dashboard page dan Profile page. Hal ini dilakukan agar navbar tidak tampil pada page login dan signup, karena navbar yang akan di gunakan di login dan signup page akan berbeda.

13. Update Navbar pada page signup dan login

```
/* Navbar login */
/* Navbar */
<header className="mb-10">
  /* Navbar */
  <nav>
    <div className="navbar bg-blue-500 fixed top-0 left-0 w-full z-50 shadow-xl flex justify-center">
      <Link href="/dashboard" className="btn btn-ghost text-xl">
        Barang Second
      </Link>
    </div>
  </nav>
</header>
```



Penjelasan:

Menambahkan navbar yang berbeda untuk page login dan page signup. Dengan meletakkan nama web di tengah dan tidak memiliki bentuk yang sama dengan navbar di dashboard page.

14. Add database dan connect prisma orm

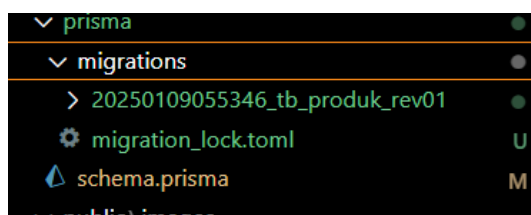
```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "mysql"
  url      = env("DATABASE_URL")
}

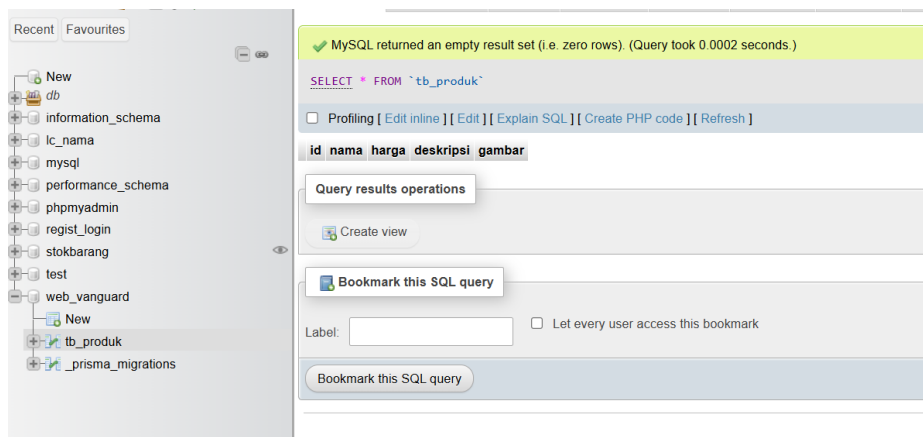
model tb_produk {
  id      Int      @id @default(autoincrement())
  nama    String   @db.VarChar(255)
  harga   Float
  deskripsi String? @db.Text
  gambar  String   @db.VarChar(255)
}
```

Penjelasan :

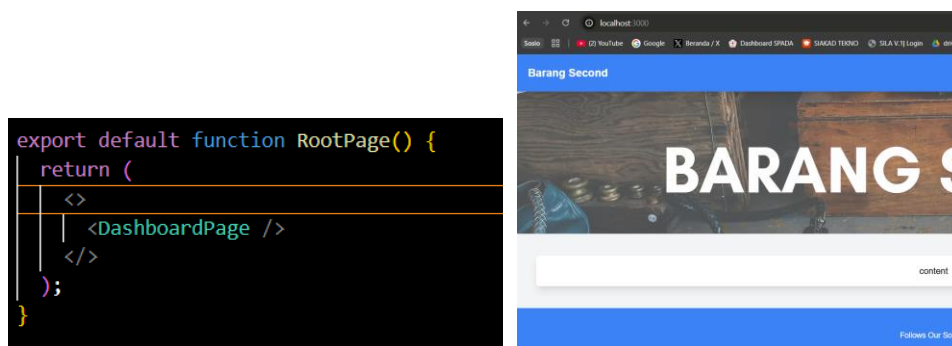
Langkah pertama adalah menyesuaikan datasource db dengan phpmy admin, buat database, lalu edit provider dan url dan env harus disesuaikan agar terhubung dengan baik, seperti : `mysql://root:@localhost:3306/web_vanguard`. Lalu jalankan `npx prisma migrate dev`.



Lalu akan muncul migrasi model yang telah dibuat bisa cek di folder prisma. Hasilnya akan seperti ini jika sudah connect dengan database phpnya.



15. Update page root



Penjelasan :

Mengubah root page agar ketika menjalankan project dari localhost/3000 langsung masuk ke dalam dashboard page

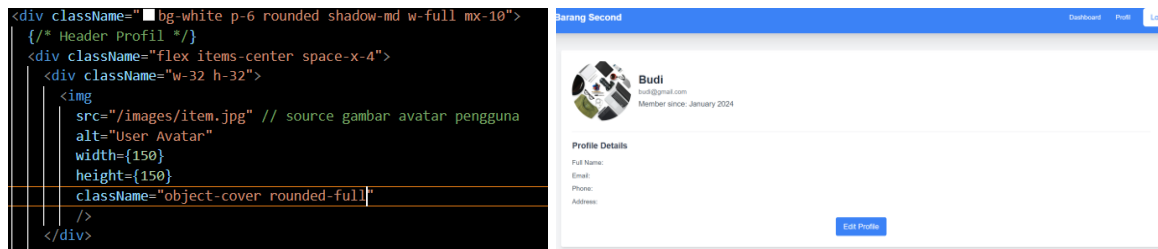
16. Update dashboard page



Penjelasan:

Update page dashboard, dengan menambahkan card produk, untuk memuat produk yang akan di add.

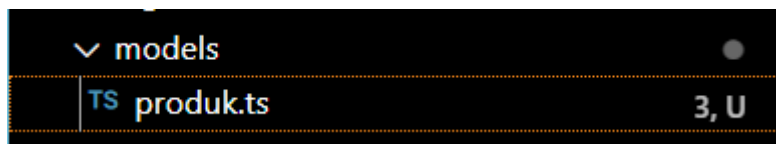
17. Update profil page



Penjelasan :

Mengubah lebar card profil, dan menambahkan gambar profil dengan gambar yang sudah ada.

18. Update dashboard page menampilkan data



Penjelasan :

Membuat models/produk.ts agar menjadi fungsi untuk mengambil data agar bisa di tampilkan di page dashboard

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

const prisma = new PrismaClient();

// Fungsi untuk menampilkan data produk
export async function getData() {
  const products = await prisma.tb_produk.findMany();
  return products;
}

// Endpoint untuk menangani permintaan produk via API
export default async function handler(req, res) {
  if (req.method === 'GET') {
    try {
      const products = await getData();
      res.status(200).json(products);
    } catch (error) {
      res.status(500).json({ error: 'Failed to fetch products' });
    }
  } else {
    res.status(405).json({ error: 'Method not allowed' });
  }
}
```

Buat fungsi getData dengan ORM Prisma agar bisa memanggil data dari database lalu di export agar bisa di import atau di masukkan di dashboard page,

```
const [products, setProducts] = useState<any[]>([]); // Menyimpan data produk dalam state

// Fungsi untuk mengambil data produk dari model
async function fetchData() {
  const data = await getData(); // Mendapatkan data produk
  setProducts(data); // Menyimpan data produk ke state
}

// useEffect untuk mengambil data produk ketika halaman dimuat
useEffect(() => {
  fetchData();
}, []);
```

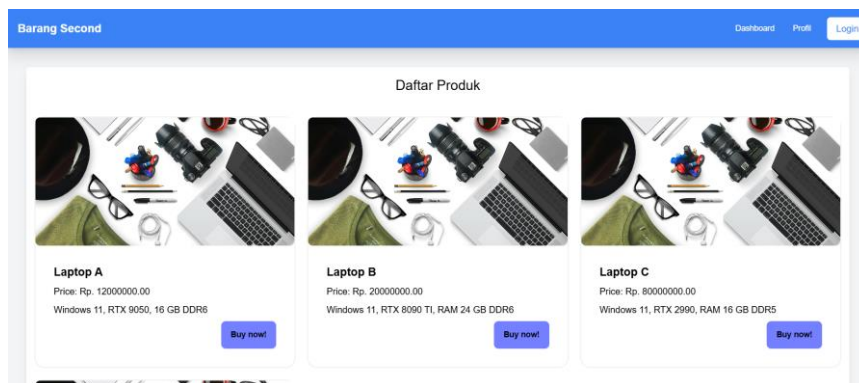
Lalu panggil data dengan membuat function fetch data agar bisa memanggil getData sebelumnya, lalu set data sebagai product agar bisa di masukkan di card produk untuk ditampilkan di web,

```

<h2 className="text-black text-center mb-10 text-2xl">Daftar Produk</h2>
<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">
  {products.map((product) => (
    <div key={product.id} className="card glass">
      
      <div className="card-body text-black">
        <h2 className="card-title">{product.nama}</h2>
        <p>Price: Rp. {product.harga.toFixed(2)}</p>
        <p>{product.deskripsi || "No description available."}</p>
        <div className="card-actions justify-end">
          <button className="btn btn-primary">Buy now!</button>
        </div>
      </div>
    </div>
  ))}
</div>

```

Lalu, modify sedikit code card sebelumnya untuk menampilkan data yang ada di database, dengan menggantinya dengan `product.nama`, `product.harga` dan `product.deskripsi`. hal ini bertujuan untuk memanggil data yang didapatkan untuk di tampilkan di web. Hasilnya :



19. Update add data

```

<div className="p-10 bg-gray-100">
  <div className="bg-white p-6 rounded-md shadow-md max-w-lg mx-auto">
    <h2 className="text-2xl font-bold mb-6 text-center text-black">Tambah Produk Baru</h2>
    <label className="input input-bordered flex items-center my-5">
      Nama Produk :
      <input type="text" className="grow" placeholder="Produk A" />
    </label>
    <label className="input input-bordered flex items-center my-5">
      Harga :
      <input type="text" className="grow" placeholder="Rp. 200.000.000" />
    </label>
    <label className="input input-bordered flex flex-col my-5 h-40 p-4">
      Deskripsi Produk :
      <textarea className="grow p-3 border rounded mt-2 mb-4" placeholder="Windows 11, RTX 3050 TI, dll"></textarea>
    </label>
    <div className="flex justify-center">
      <button className="btn btn-xs lg:btn-lg bg-blue-500 text-white">Submit</button>
    </div>
  </div>

```


Tambah Produk Baru

Nama Produk : Produk A

Harga : Rp. 200.000.000

Deskripsi Produk :

Windows 11, RTX 3050 TI, dll

Submit

Penjelasan :

Mememodifikasi bentuk dan place holder dari text field untuk menambah data.

20. Update fungsi pada model produk.ts

```
export async function setSaveData(nama: string, harga: number, deskripsi?: string) {
  try {
    // Pastikan harga adalah angka (integer) sebelum disimpan
    const hargaInt = Number.isNaN(harga) ? 0 : Math.floor(harga);

    // Simpan data produk ke database
    await prisma.tb_produk.create({
      data: {
        nama: nama,
        harga: hargaInt, // Pastikan harga adalah integer
        deskripsi: deskripsi || null, // Deskripsi opsional
      },
    });
  } catch (error) {
    console.error("Gagal menyimpan data produk:", error);
    throw new Error("Gagal menyimpan data produk. Silakan periksa log untuk detail lebih lanjut.");
  }
}
```

Penjelasan :

Menambahkan fungsi add pada model produk, fungsi menggunakan setsavedata yang menerima 3 jenis data yaitu nama, harga dan deskripsi. Lalu dilengkapi juga keharusan untuk mengisi harga dengan variable number agar tidak eror, lalu terakhir menambahkan return jika terjadi eror.

21. Update add page

```
// Panggil fungsi setSaveData
const handleSubmit = async (e: React.FormEvent) => {
  e.preventDefault(); // Mencegah reload halaman

  if (!nama || !harga) {
    alert("Nama dan harga produk wajib diisi!");
    return;
  }

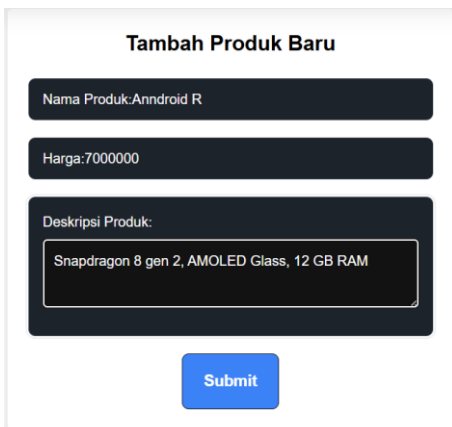
  try {
    // Panggil fungsi setSaveData
    await setSaveData(nama, parseFloat(harga), deskripsi);
    alert("Produk berhasil ditambahkan!");
    // Reset form setelah submit
    setName("");
    setHarga("");
    setDescription("");
  } catch (error) {
    console.error("Gagal menyimpan produk:", error);
    alert("Terjadi kesalahan saat menambahkan produk.");
  }
};
```

Penjelasan :

Handle diatas digunakan untuk mengecek apakah data yang di isi sudah lengkap agar nantinya bisa di masukkan ke database

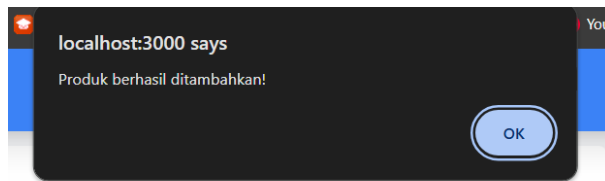
```
type="text"
className="grow"
placeholder="Produk A"
value={nama}
onChange={(e) => setName(e.target.value)}
/>
</label>
<label className="input input-bordered flex items-center my-5">
  Harga:
  <input
    type="number"
    className="grow"
    placeholder="200000000"
    value={harga}
    onChange={(e) => setHarga(e.target.value)}
  />
</label>
<label className="input input-bordered flex flex-col my-5 h-40 p-4">
  Deskripsi Produk:
  <textarea
    className="grow p-3 border rounded mt-2 mb-4"
    placeholder="Windows 11, RTX 3050 TI, dll"
    value={deskripsi}
    onChange={(e) => setDescription(e.target.value)}
  />
</label>
```

Mengubah code pada text field dan menyesuaikan agar text yang di masukan memiliki tipe data yang sesuai dengan tipe data pada database setelah menekan tombol submit, ketika data di tambahkan akan memberikan alert diatas.



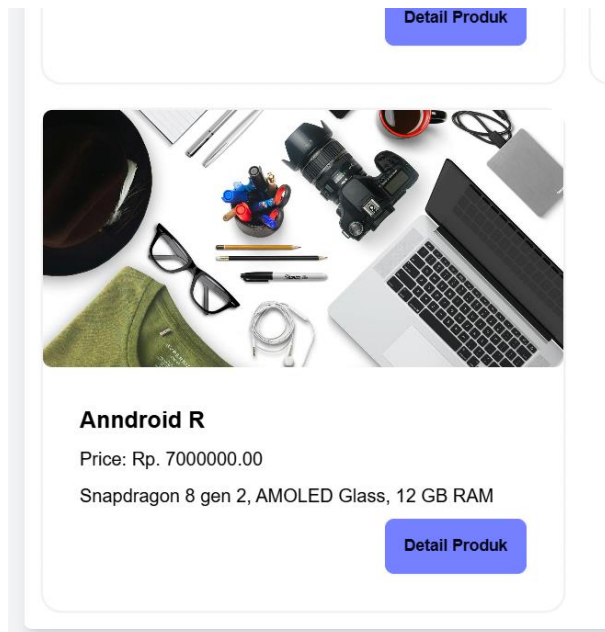
The screenshot shows a web form titled "Tambah Produk Baru". It contains three input fields: "Nama Produk" with the value "Anndroid R", "Harga" with the value "7000000", and "Deskripsi Produk" with the value "Snapdragon 8 gen 2, AMOLED Glass, 12 GB RAM". A blue "Submit" button is located at the bottom of the form.

(penjelasan : Tambah Data dengan input form)



Tambah Produk Baru

(Alert : Ketika data berhasil di tambahkan)



(Data tampil di dashboard)