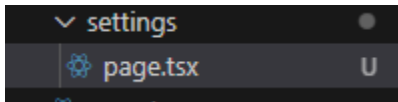


Nama: Rico Decaprio Esa Ananta

Npm:22312081

1. Membuat folder setting dan membuat file di dalam folder app



2. membuat hello setting di page.tsx

```
app > (dashboard) > [storeId] > (routes) > settings > page.tsx > default
1  const SettingsPage = () =>{
2      return(
3          <div>
4              Hello Settings
5          </div>
6      )
7  };
8
9  export default SettingsPage
```

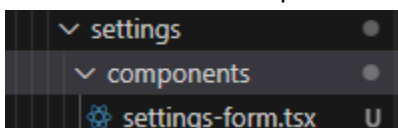
3. membuat sign-in

```
interface SettingsPageProps {
  params: {
    storeId: string;
  }
};

const SettingPage: React.FC<SettingsPageProps> = async ({
  params
}) =>{
  const { userId } = await auth();

  if (!userId) {
    redirect ("/sign-in");
  }
}
```

4. membuat folder components dan membuat file settings-form.tsx



5. Menampilkan halaman yang memiliki tata letak vertikal dengan menggunakan flexbox.

```
return (
  <div className="flex-col">
    <div className="flex-1 space-y-4 p-8 pt-6">
      <SettingsForm initialData={store}/>
    </div>
  </div>
)
```

6. menambah program pada settings- from.tsx berfungsi untuk menampilkan atau memproses data terkait pengaturan (settings).

```
app > (dashboard) > [storeid] > (routes) > settings > components > settings-form.tsx > SettingsFrom
1  "use client";
2
3  import { Store } from "@prisma/client";
4
5  interface SettingsPageProps {
6    initialData: Store;
7  }
8
9  export const SettingsFrom: React.FC <SettingsPageProps> = ({
10    initialData
11  }) => {
12    return (
13      <div>
14        Settings From
15      </div>
16    );
17  };
```

7. Kode tersebut adalah pernyataan **import** dalam JavaScript atau TypeScript

```
import { SettingsFrom } from "../components/settings-form";
```

8. potongan JSX (JavaScript XML) yang berfungsi untuk membuat struktur HTML dengan bantuan React atau Next.js.

```
<div classname="flex items-center justify-between">
  <heading
    title="Settings"
    description="Manage store preferences"
  />
</div>
```

9.

definisi sebuah komponen React bernama Heading. Komponen ini dirancang untuk menampilkan judul dan deskripsi.

```
components > ui > heading.tsx > Heading
1  interface HeadingProps {
2    title: string;
3    description: string;
4  };
5  export const Heading: React.FC<HeadingProps> = ({
6    title,
7    description
8  }) => {
9    return(
10      <div>
11        <h2>{title}</h2>
12        <p>
13          {description}
14        </p>
15      </div>
16    )
17  }
```

10.fungsi kode program tersebut komponen React yang mendefinisikan sebuah tombol dengan ikon bergambar tong sampah (trash icon).

```

<Button
  variant="destructive"
  size="icon"
  onClick={() => {}}
>
<Trash className="h-4 w-4" />
</Button>

```

11. menginstall separator

```
vortex> npx shadcn@latest add separator
```

12. Dengan Zod, Anda dapat membuat skema (schemas) yang mendefinisikan struktur

```
import * as z from "zod";
```

13. useFrom: Mengelola status

zodResolver: Mengintegrasikan Zod untuk validasi form dengan React Hook Form.

```
import { useForm } from "react-hook-form";
import { zodResolver } from "@hookform/resolvers/zod";
```

14. useState: dalah hook bawaan dari React yang digunakan untuk mengelola state di dalam komponen fungsi React.

Form:Menyediakan fitur seperti validasi, pengiriman data, atau integrasi dengan pustaka form lain (misalnya React Hook Form).

```
import { useState } from "react";
import { Form } from "@components/ui/form";
```

15. lemen-elemen input seperti <input>, <textarea>, atau <select> untuk menangani logika pengelolaan form.

```
import {
  Form,
  FormControl,
  FormField,
  FormItem,
  FormLabel
} from "@components/ui/form";
```

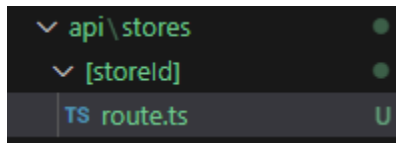
16. mengirimkan formulir (submit), dan ketika aplikasi dalam kondisi loading, tombol tersebut akan dinonaktifkan sehingga pengguna tidak bisa mengkliknya dua kali atau mengirim formulir lagi.

```

</div>
<button disabled={loading} className="ml-auto" type="submit">
  | save changes
</button>
</form>

```

17. buat folder di folder api nama folder storeId dan buat file route.ts



18. kode ini memastikan bahwa hanya pengguna yang terautentikasi yang bisa mengupdate data, dan juga memastikan bahwa parameter `name` disediakan dalam request.

```
app > api > stores > [storeId] > TS route.ts > PATCH
1 import { auth } from "@clerk/nextjs/server";
2 import { NextResponse } from "next/server";
3
4 export async function PATCH (
5   req: Request,
6   { params }: { params: { storeId: string } }
7 ) {
8   try {
9     const { userId } = await auth();
10    const body = await req.json();
11
12    const { name } = body;
13
14    if (!userId) {
15      return new NextResponse("unauthenticated", { status: 401 });
16    }
17
18    if (!name) {
19      return new NextResponse("Name is required", { status: 400 });
20    }
21
22  } catch (error) {
23    console.log("[STORE_PATCH]", error);
24    return new NextResponse("Internal error", { status: 500 });
25  }
26 }
```

18. berfungsi untuk menghapus satu atau lebih entri toko dari database berdasarkan storeId dan userId yang disertakan dalam request, kemudian mengembalikan hasilnya dalam format JSON.

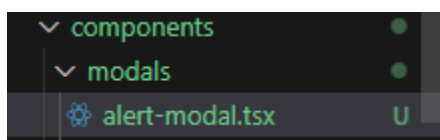
```
const store = await db.store.deleteMany({
  where: {
    id: params.storeId,
    userId
  }
});

return NextResponse.json(store);
```

19. mempermudah pembuatan form yang modular, terstruktur, dan mudah dibaca.

```
try {
  setLoading(true);
  await axios.patch(`api/stores/${params.storeId}`, data);
} catch (error) {
  toast.error("Something went wrong.");
} finally {
  setLoading(false);
}
```

20. membuat file



21.d digunakan untuk menampilkan modal konfirmasi dengan pesan peringatan. Modal ini meminta konfirmasi pengguna sebelum melakukan tindakan yang tidak dapat dibatalkan.

```
components > modals > alert-modal.tsx > AlertModal
1  "use client";
2
3  import { useEffect, useState } from "react";
4  import { Modal } from "@/components/ui/modal";
5
6  interface AlertModalProps {
7    isOpen: boolean;
8    onClose: () => void;
9    onconfirm: () => void;
10   loading:boolean;
11 }
12
13 export const AlertModal: React.FC<AlertModalProps> =({
14   isOpen,
15   onClose,
16   onconfirm,
17   loading
18 }) => {
19   const [isMounted, setIsMounted] = useState(false);
20   useEffect(() => {
21     setIsMounted(true);
22   }, [])
23
24   if (!isMounted) {
25     return null;
26   }
27   return (
28     <Modal
29       title="Are you sure?"
30       description="This action cannot be undone."
31       isOpen={isOpen}
32       onClose={onClose}
33     >
34
35     </Modal>
36   )
37 }
```

22. digunakan untuk mengimpor komponen

```
import { Button } from "@/components/ui/button";

interface AlertModalProps {
```

23.

digunakan untuk mendeskripsikan sebuah link atau item navigasi di dalam aplikasi.

```
{
  href: `/${params.storeId}/`,
  label: 'Overview',
  active: pathname === `/${params.storeId}/`,
},
```

24.menginstal add alert

```
PS D:\web Project\ecommerce-visionvortex> npx shadcn@latest add alert
● >>
✔ Checking registry
```

25 menampilkan pesan alert dengan beberapa properti dan styling yang disesuaikan berdasarkan variannya

```
components > ui > @api-alert > @ApiAlert
4  interface ApiAlertProps {
5    title: string;
6    description: string;
7    variant: "public" | "admin";
8  };
9
10 const textMap: Record<ApiAlertProps["variant"], string> = {
11   public: "public",
12   admin: "admin"
13 };
14
15 const variantMap: Record<ApiAlertProps["variant"], string> = {
16   public: "secondary",
17   admin: "destructive"
18 };
19
20 export const ApiAlert: React.FC<ApiAlertProps> = ({
21   title,
22   description,
23   variant = "public",
24 }) => {
25   return (
26     <Alert>
27       <Server className="h-4 w-4"/>
28     </Alert>
29   )
30 }
```

26. mengimpor komponen `ApiAlert` yang kemungkinan besar digunakan untuk menampilkan pemberitahuan atau alert terkait interaksi dengan API di aplikasi.

```
import { AlertModal } from "@/components/modals/alert-modal";
import { ApiAlert } from "@/components/ui/api-alert";
```

27.membuat model order dan model orderItem

```
20
21 model Order {
22   id String @id @default(uuid())
23   storeId String
24   store Store @relation("StoreToOrder", fields: [storeId], references: [id])
25   orderItems OrderItem []
26   isPaid Boolean @default(false)
27   phone String @default("")
28   address String @default("")
29   createdAt DateTime @default(now())
30   updatedAt DateTime @updatedAt
31
32   @@index([storeId])
33 }
34
35 model OrderItem {
36   id String @id @default(uuid())
37   orderId String
38   order Order @relation(fields: [orderId], references: [id])
39   productId String
40   product Product @relation(fields: [productId], references: [id])
41
42   @@index([orderId])
43   @@index([productId])
44 }
```

28.membuat orders di tampilan

```
{
  href: `/${params.storeId}/orders`,
  label: 'orders',
  active: pathname === `/${params.storeId}/orders`,
},
```

29. mengambil data dari database menggunakan Prisma ORM

```
1 import { format, formatDate } from "date-fns"
2
3 import db from "@lib/db";
4 import { BillboardClient } from "../components/client";
5 import { BillboardColumn } from "../components/columns";
6
7 const BillboardsPage = async ({
8   params
9 }): {
10   params: {storeId: string}
11 } => {
12   const billboards= await db.billboard.findMany({
13     where: {
14       storeId: params.storeId
15     },
16     orderBy: {
17       createdAt: 'desc'
18     }
19   });
20
21   const formattedBillboards: BillboardColumn[] = billboards.map(
22     (item) => ({
23       id: item.id,
24       label: item.label,
25       createdAt: formatDate(item.createdAt, "MMMM do, yyyy")
26     })
27   );
28
29   return (
30     <div className="flex-col">
31       <div className="flex-1 space-y-4 p-8 pt-6">
32         <BillboardClient data={formattedBillboards}/>
33       </div>
34     </div>
```

30.membuat ordercolumn

```
7+ export type OrderColumn = {
8+   id: string;
9+   phone: string;
10+   address: string;
11+   isPaid: boolean;
12+   totalPrice: string;
13+   products: string;
14+   createdAt: string ;
15+ }
```