

Nama : Rifki Abdullah

NPM : 22312080

DOKUMENTASI Pengerjaan

Membuat Komponen Form

Tanggal : 07 Januari 2025

Waktu : 18.54

- Menginstall komponen **form** dengan perintah **npx shadcn@latest add form** dari Pustaka **shadcn/ui** kedalam proyek, dengan dokumentasi terminal berikut ini :

```
PS D:\ecommerce-visionvortex> npx shadcn@latest add form
✓ Checking registry.
  Installing dependencies.

It looks like you are using React 19.
Some packages may fail to install due to peer dependency issues in npm (see https://ui.shadcn.com/react-19).

✓ How would you like to proceed? » Use --legacy-peer-deps
✓ Installing dependencies.
✓ The file button.tsx already exists. Would you like to overwrite? ... yes
✓ Created 2 files:
  - components\ui\form.tsx
  - components\ui\label.tsx
i Updated 1 file:
  - components\ui\button.tsx

npm notice
npm notice New major version of npm available! 10.9.0 -> 11.0.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.0.0
npm notice To update run: npm install -g npm@11.0.0
npm notice
PS D:\ecommerce-visionvortex>
```

- Berikut ini hasil yang sudah terinstall dengan menghasilkan file baru berupa **form.tsx** dan **label.tsx** pada folder **ui** didalam **components** :



Tanggal : 07 Januari 2025

Waktu : 19.21

- Menambahkan komponen **input** dengan perintah **npx shadcn@latest add input** dari Pustaka **shadcn/ui** kedalam proaiek, dengan dokumentasi terminal berikut ini :

```
PS D:\ecommerce-visionvortex> npx shadcn@latest add input
✓ Checking registry.
✓ Created 1 file:
  - components\ui\input.tsx
```

- Berikut ini hasil yang sudah terinstall dengan menghasilkan file baru berupa **input.tsx** pada pada folder **ui** didalam **components** :



Tanggal : 08 Januari 2025

Waktu : 00.02

Melanjutkan membuat form komponen

Pada file **store-modal.tsx**

```
3 import * as z from "zod";
4 import { useForm } from "react-hook-form";
5 import { zodResolver } from "@hookform/resolvers/zod";
```

Menambahkan import seperti diatas seperti zod untuk membuat dan memvalidasi skema data, lalu import useForm yaitu hook dari Pustaka react hook Form, lalu menambahkan import zodResolver supaya zod dengan React Hook Form terintegrasikan.

```

11 import {
12     Form,
13     FormControl,
14     FormField,
15     FormItem,
16     FormLabel,
17     FormMessage
18 } from "@components/ui/form";
19 import { Input } from "@components/ui/input";
20 import { Button } from "@components/ui/button";

```

Menambahkan import berbagai komponen UI terkait (Form, FormControl, FormField, FormItem, FormLabel, FormMessage). Mengimport komponen (Input), dan mengimport komponen tombol (Button).

```

22 const formSchema = z.object({
23     name: z.string().min(1),
24 });

```

Menambahkan const formSchema untuk mendefinisikan skema validasi formulir menggunakan Zod dan `name` adalah field teks wajib dengan minimal 1 karakter

```

const form = useForm<z.infer<typeof formSchema>>({
    resolver: zodResolver(formSchema),
    defaultValues: {
        name: "",
    },
});

```

Menambahkan const baru untuk menginisialisasi hook `useForm` dengan resolver Zod untuk validasi skema dengan `defaultValues` menetapkan nilai default untuk input formulir, di sini `name` diinisialisasi sebagai string kosong.

```

const onSubmit = async (values: z.infer<typeof formSchema>) => {
    console.log(values);
    // TODO: Create Store
}

```

Menambahkan const baru untuk mendefinisikan fungsi `onSubmit`, yang akan dipanggil saat formulir dikirimkan.

```

44     <Modal
45       title="create store"
46       description="Add a new store to manage products and categories"
47       isOpen={storeModal.isOpen}
48       onClose={storeModal.onClose}
49     >
50       <div>
51         <div className="space-y-4 py-2 pb-4">
52           <Form {...form}>
53             <form onSubmit={form.handleSubmit(onSubmit)}>
54               <FormField
55                 control={form.control}
56                 name="name"
57                 render={({ field }) => (
58                   <FormItem>
59                     <FormLabel>Name</FormLabel>
60                     <FormControl>
61                       <Input placeholder="E-Commerce" {...field}/>
62                     </FormControl>
63                     <FormMessage />
64                   </FormItem>
65                 )}
66               />
67             <div className="pt-6 space-x-2 flex items-center justify-end w-full">
68               <Button
69                 variant="outline"
70                 onClick={storeModal.onClose}>Cancel
71               </Button>
72               <Button type="submit">Continue</Button>
73             </div>
74           </form>
75         </Form>
76       </div>
77     </div>
78   </Modal>

```

Menambahkan isi didalam Form yang terdapat :

- Membungkus formulir HTML dengan komponen `Form` untuk integrasi dengan React Hook Form dan `handleSubmit` menangani validasi dan pemrosesan data sebelum memanggil `onSubmit`.
- Komponen `FormField` agar terhubung dengan input form
- Menambahkan `render` supaya mengonfigurasi elemen input dan menampilkan pesan error jika ada.
- Dan menambahkan Input dengan isi placeholder "E-Commerce" dan dihubungkan ke field "name".
- Menambahkan tombol Button pada form yang terdiri dari Cancel dan Continue
- **Hasil form yang ditambahkan diatas**

create store



Add a new store to manage products and categories

Name

Rifki Abdullah

Cancel

Continue

- Jika table tidak terisikan

create store



Add a new store to manage products and categories

Name

E-Commerce

String must contain at least 1 character(s)

Cancel

Continue

Tanggal : 09 Januari 2025

Waktu : 14.34 s/d ...

Membuat HomePage pada Admin

- Membuat folder (dashboard) dan lalu didalamnya terdapat folder [storeId] lalu membuat file layout.tsx



- Mengerjakan isi layout.tsx

```

1 import db from "@lib/db";
2 import { auth } from "@clerk/nextjs/server";
3 import { redirect } from "next/navigation";
4 import React from "react";

```

Mengimpor modul db, mengimpor fungsi auth dari clerk, mengimpor fungsi redirect dari next.js, mengimpor modul react.

```

export default async function DashboardLayout({
  children,
  params,
}): {
  children: React.ReactNode;
  params: {storeId: string}
}) {

```

Membuat komponen layout untuk dashboard. Menambahkan komponen children, params. Untuk memastikan hanya pengguna yang login.

```

}) {
  const {userId} = await auth(); ←
  if (!userId) {
    redirect("sign-in")
  }
  const store = await db.store.findFirst({
    where: {
      id: params.storeId,
      userId
    }
  })

```

Mendapatkan data autentikasi

```

  }) {
    const {userId} = await auth();
    if (!userId) {
      redirect("sign-in")
    }
    const store = await db.store.findFirst({
      where: {
        id: params.storeId,
        userId
      }
    })
  }
}

```

Membuat supaya pengguna jika blm login diarahkan ke halaman sign-in

```

  }) {
    const {userId} = await auth();
    if (!userId) {
      redirect("sign-in")
    }
    const store = await db.store.findFirst({
      where: {
        id: params.storeId,
        userId
      }
    })
  }
}

```

Membuat query ke database untuk mencari data store pertama yang memenuhi kriteria.

```

if (!store) {
  redirect("/")
}
return (
  <>
    <div>This is Navbar</div>
    {children}
  </>
)

```

Melakukan pengecekan jika store tidak ada

- Dokumentasi hasil perubahan dari [storeId]



This is Navbar

This is Dashboard

Dia mendapatkan Id store di Alamat halaman diatas.

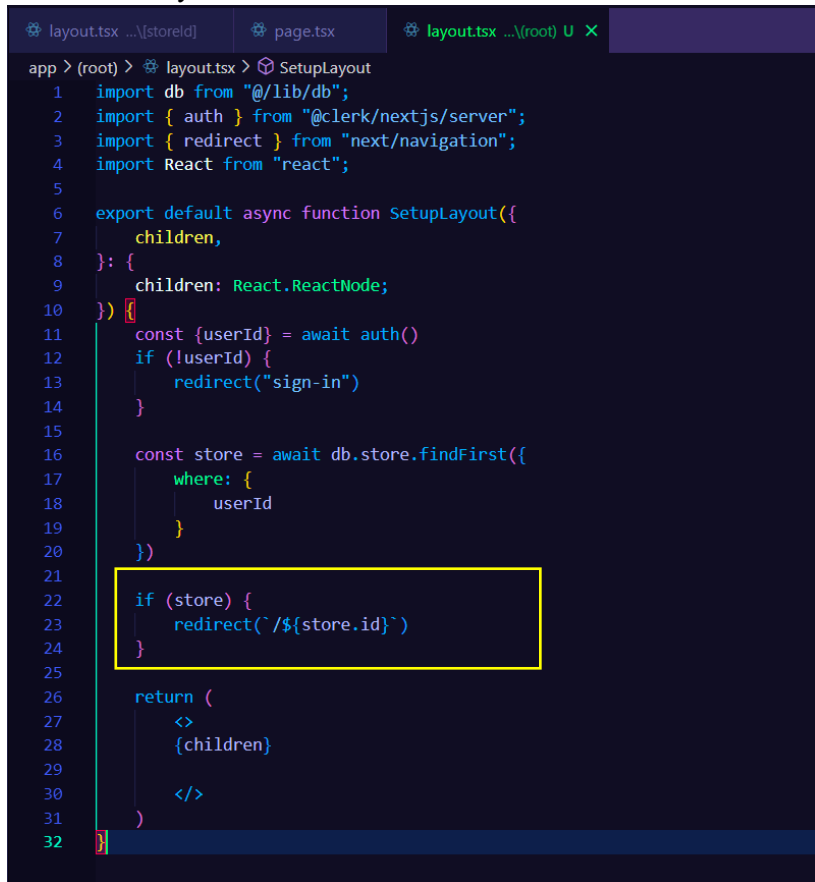
- Membuat Folder baru didalam [storeId] dengan nama (routes) dan lalu membuat file dengan nama page.tsx
- Isi page.tsx

```
const DashboardPage = () => {  
  return (  
    <div>  
      This is Dashboard  
    </div>  
  );  
}  
  
export default DashboardPage;
```

- Membuat file baru didalam folder (root) yaitu file layout.tsx



- Membuat isi nya :



```

app > (root) > layout.tsx > SetupLayout
1 import db from "@lib/db";
2 import { auth } from "@clerk/nextjs/server";
3 import { redirect } from "next/navigation";
4 import React from "react";
5
6 export default async function SetupLayout({
7   children,
8 }): {
9   children: React.ReactNode;
10 } {
11   const {userId} = await auth()
12   if (!userId) {
13     redirect("sign-in")
14   }
15
16   const store = await db.store.findFirst({
17     where: {
18       userId
19     }
20   })
21
22   if (store) {
23     redirect(`/${store.id}`)
24   }
25
26   return (
27     <>
28       {children}
29     </>
30   )
31 }
32

```

Pada isinya hamper sama dengan file layout.tsx pada (dashboard)/[storeId] hanya saja menambahkan isi yang ditandai untuk memastikan store yang diminta oleh pengguna benar-benar ada dan dapat diakses.

- Membuat folder baru didalam folder (root) dengan nama (routes) dan memindahkan file page.tsx yang awalnya di (root) pindah ke (routes).



- Melakukan **npx prisma migrate reset** untuk mereset database ke keadaan awal berdasarkan migrasi prisma.

```
PS D:\Universitas Teknokrat Indonesia\Semester 5\WEB 2\ecommerce-visionvortex> npx prisma generate
```

- Pada bash tersebut melakukan perintah tersebut untuk menghasilkan client prisma yang akan digunakan.

```
PS D:\Universitas Teknokrat Indonesia\Semester 5\WEB 2\ecommerce-visionvortex> npx prisma db push
```

Push skema Prisma langsung ke database

```
const onSubmit = async (values: z.infer<typeof formSchema>) => {
  try {
    setLoading(true)
    const response = await axios.post('/api/stores', values)
    console.log(response.data);
    toast.success("Berhasil Membuat Toko");
    window.location.assign(`/${response.data.id}`)
  } catch (error) {
    toast.error("Gagal Membuat Toko")
  } finally{
    setLoading(false)
  }
}
```

- Menambahkan kode untuk mengarahkan pengguna ke URL tertentu berdasarkan data yang diterima dari respons server.
- Memodifikasi pada file page.tsx pada folder (dashboard)/[storeId]/(routes)

```
2
3 interface DashboardPageProps {
4   params: {storeId: string}
5 }
```

Menambahkan interface tersebut untuk mendeskripsikan tipe properti yang diterima oleh komponen.

```
7 const DashboardPage = async ({params} : DashboardPageProps) => {
8
```

Menambahkan properti params (dari tipe DashboardPageProps)

```
8
9   const store = await db.store.findFirst({
10     where: {
11       id: params.storeId
12     }
13   })
```

Menambahkan const baru untuk mencari data di tabel store. Query ini mencari satu entri dalam tabel store yang memiliki id sesuai dengan nilai params.storeId.

```
return (
  <div>
    Active Store: {store?.name}
  </div>
);
```

Menambahkan untuk menampilkan keterangan berdasarkan nama storenya.

Hasil penampilannya:



Tanggal : 10 Januari 2025

Waktu : 07.23 s/d ...

Membuat Navbar pada Admin



Merubah isi tampilan pada file layout.tsx pada (dashboard)/[storeId]



Lalu membuat file baru pada folder components dengan nama file navbar.tsx dan serta memberikan isinya untuk dipanggil Navbar pada layout.tsx sebelumnya.

```
layout.tsx ...[storeId]  navbar.tsx M X  store-modal.tsx  page.tsx  components > navbar.tsx > default
1  import { UserButton } from "@clerk/nextjs";
2
3  const Navbar = () => {
4    return (
5      <div className="border-b">
6        <div className="flex h-16 items-center px-4">
7          <div>Store Switcher</div>
8          <div>main nav</div>
9          <div className="ml-auto flex items-center space-x-4">
10             <UserButton afterSignOutUrl="/" />
11          </div>
12        </div>
13      </div>
14    );
15  }
16
17  export default Navbar;
```

Membuat isi navbar berupa bordernya, mengatur ukurannya, membuat userbutton-nya.

```
<MainNav />
```

Mengubah main nav sebelumnya menjadi seperti itu untuk memanggil hasil dari main-nav.tsx

```
layout.tsx ...[storeId]  navbar.tsx M  main-nav.tsx U X  sto
components > main-nav.tsx > MainNav
1  'use client'
2
3  export function MainNav({
4    className,
5
6  }: React.HTMLAttributes<HTMLElement>) {
7    return (
8      <nav>
9        This is Main Nav
10      </nav>
11    )
12  }
```

Mendefinisikan MainNav untuk dipanggil dan di render.



Hasilnya.

```
const pathname = usePathname();
const params = useParams();

const routes = [
  {
    href: `/${params.storeId}/settings`,
    label: 'settings',
    active: pathname === `/${params.storeId}/settings`,
  }
]
```

Menambahkan code pada main-nav.tsx untuk mendefinisikan const pathname dan const params. Dan membuat const routes.

```
{routes.map((route) => (
  <Link
    key={route.href}
    href={route.href}
    className={cn(
      "text-sm font-medium transition-colors hover:text-primary",
      route.active ? "text-black dark:text-white" : "text-muted-foreground"
    )}
  >
    {route.label}
  </Link>
)}
```

Merender navigasi dinamis menggunakan data dari array routes pada main-nav.tsx

```
<MainNav className="mx-6"/>
```

Dan memberikan jarak nya.



Active Store: Toko01

-

Dan ini hasilnya.

- `PS D:\Universitas Teknokrat Indonesia\Semester 5\WEB 2\ecommerce-visionvortex> npx shadcn@latest add popover`

Menginstall popover

- `PS D:\Universitas Teknokrat Indonesia\Semester 5\WEB 2\ecommerce-visionvortex> npx shadcn@latest add command`

Menginstall command

```
import db from "@/lib/db";

const Navbar = async () => {
  const {userId} = await auth();

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

  const stores = await db.store.findMany({
    where: {
      userId,
    }
  })

  return (
    <div className="border-b">
      <div className="flex h-16 items-center px-4">
        <StoreSwitcher items={stores} />
      </div>
    </div>
  )
}
```

-

Menambahkan const baru untuk mengaitkan StoreSwitcher dan menambahkan const baru

```
components > store-switcher.tsx > StoreSwitcher > [currentStore]
1  'use client';
2
3  import { Check, ChevronsUpDown, PlusCircle, StethoscopeIcon, Store as StoreIcon } from "lucide-react";
4  import { useState } from "react";
5  import { Store } from "@prisma/client";
6  import { useParams, useRouter } from "next/navigation";
7
8  import { Popover, PopoverContent, PopoverTrigger } from "@components/ui/popover";
9  import { Button } from "@components/ui/button";
10 import { useStoreModal } from "@hooks/use-store-modal";
11 import { cn } from "@lib/Utils";
12 import { Command, CommandEmpty, CommandGroup, CommandInput, CommandItem, CommandList } from "cmdk";
13
14 type PopoverTriggerProps = React.ComponentPropsWithRef<typeof PopoverTrigger>
15
16 interface StoreSwitcherProps extends PopoverTriggerProps {
17   items: Store[];
18 };
19
20 export default function StoreSwitcher({
21   className,
22   items = []
23 }: StoreSwitcherProps) {
24   const storeModal = useStoreModal();
25   const params = useParams();
26   const router = useRouter();
27
28   const formattedItems = items.map((item) => ({
29     label: item.name,
30     value: item.id
31   }));
32
33   const currentStore = formattedItems.find((item) => item.value === params.store);
34
35   const [open, setOpen] = useState(false);
36
37   const onStoreSelect = (store: {value: string, label: string}) => {
```

Mendefinisikan StoreSwitcher didalamnya membuat button took berupa searcing took, nama toko, tambah toko.

Tanggal : 12 Januari 2025

Waktu : 14.44 s/d ...


```
settings-form.tsx M × layout.tsx use-origin.tsx U ×
hooks > use-origin.tsx > useOrigin
1 import { useState, useEffect } from "react";
2
3 export const useOrigin = () => {
4   const [mounted, setMounted] = useState(false);
5   const origin = typeof window !== "undefined" && window.location.origin ? window.location.origin : '';
6
7   useEffect(() => {
8     setMounted(true);
9   }, []);
10
11   if (!mounted) {
12     return '';
13   }
14
15   return origin;
16 }
```

Menambahkan custom React hook bernama useOrigin. Untuk tampilan settings pada admin.

```
27 import { useOrigin } from "@/hooks/use-origin";
28
29
30
31 interface SettingsFormProps {
32   initialData: Store;
33 }
34
35 const formSchema = z.object({
36   name: z.string().min(1)
37 });
38 type SettingsFormValues = z.infer<typeof formSchema>;
39
40 export const SettingsForm: React.FC<SettingsFormProps> =
41   ({ initialData }) => {
42     const params = useParams();
43     const router = useRouter();
44     const origin = useOrigin();
```

Menambahkan const origin dan mengimport dari useOrigin yang sebelumnya dibuat.

```
schema.prisma M X settings-form.tsx
prisma > schema.prisma > Billboard
15 }
16
17 model Store{
18   id String @id @default(uuid())
19   name String
20   userId String
21   billboards Billboard[] @relation("StoreToBillboard")
22   createdAt DateTime @default(now())
23   updatedAt DateTime @updatedAt
24 }
25
26 model Billboard {
27   id String @id @default(uuid())
28   storeId String
29   store Store @relation("StoreToBillboard", fields: [storeId], references: [id])
30   label String
31   imageUrl String
32   createdAt DateTime @default(now())
33   updatedAt DateTime @updatedAt
34
35   @@index([storeId])
36 }
```

Membuat model baru Billboard dan mendefinisikannya

```
PS D:\Universitas Teknokrat Indonesia\Semester 5\WEB 2\ecommerce-visionvortex> npx prisma generate
```

Melakukan generate prisma

```
PS D:\Universitas Teknokrat Indonesia\Semester 5\WEB 2\ecommerce-visionvortex> npx prisma db push
```

Melakukan push prisma

```
main-nav.tsx M × page.tsx U ×
app > (dashboard) > [storeId] > (routes) > billboards > page.tsx > default
1  const BillboardsPage = () => {
2      return (
3          <div>
4              billboards
5          </div>
6      );
7  }
8
9  export default BillboardsPage;
```

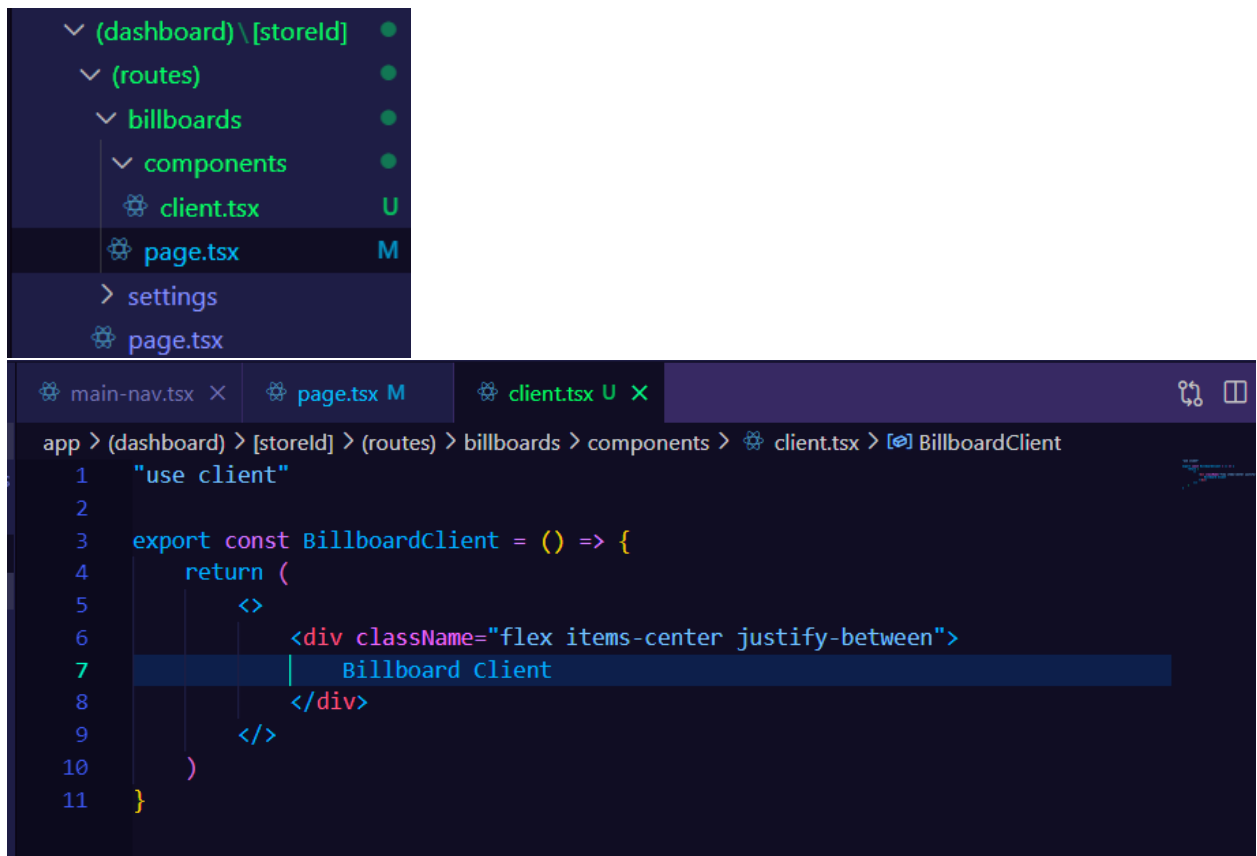
Membuat Lembar papan iklan/Billboards.

hasil lembar



```
main-nav.tsx × page.tsx M × client.tsx U
app > (dashboard) > [storeId] > (routes) > billboards > page.tsx > default
1  import { BillboardClient } from "../components/client";
2
3  const BillboardsPage = () => {
4      return (
5          <div className="flex-col">
6              <div className="flex-1 space-y-4 p-8 pt-6">
7                  <BillboardClient />
8              </div>
9          </div>
10     );
11 }
12
13 export default BillboardsPage;
```

Memperbarui isi pada halaman page billboards dan lalu membuat folder baru didalam folder billboards Bernama folder components dan berisi file client.tsx



The image shows a VS Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders (dashboard)\[storeId], (routes), and billboards, and files client.tsx, page.tsx, and settings. The code editor shows the content of client.tsx, which defines a BillboardClient component.

```
app > (dashboard) > [storeId] > (routes) > billboards > components > client.tsx > BillboardClient
1  "use client"
2
3  export const BillboardClient = () => {
4    return (
5      <>
6        <div className="flex items-center justify-between">
7          Billboard Client
8        </div>
9      </>
10    )
11  }
```

Pada file client berisikan isi untuk dipanggil pada BillboardClient yang dibuat pada file BillboardsPage sebelumnya.

Hasil Pemanggilan :



```
main-nav.tsx x page.tsx client.tsx M x
app > (dashboard) > [storeId] > (routes) > billboards > components > client.tsx > BillboardClient
1  "use client"
2
3  import { Plus } from "lucide-react"
4
5  import { Button } from "@/components/ui/button"
6  import { Heading } from "@/components/ui/heading"
7  import { Separator } from "@/components/ui/separator"
8
9  export const BillboardClient = () => {
10    return (
11      <>
12        <div className="flex items-center justify-between">
13          <Heading
14            title="Billboards (0)"
15            description="Manage Billboards for your store"
16          />
17          <Button>
18            <Plus className="mr-2 h-4 w-4"/>
19            Add New
20          </Button>
21        </div>
22        <Separator />
23      </>
24    )
25  }
```

Menambahkan heading dan button tambah pada tampilan.

Hasil Menambahkan tampilan :



```
main-nav.tsx x page.tsx client.tsx M x
app > (dashboard) > [storeId] > (routes) > billboards > components > client.tsx > [0] BillboardClient
1  "use client"
2
3  import { Plus } from "lucide-react"
4  import { useParams, useRouter } from "next/navigation"
5
6  import { Button } from "@components/ui/button"
7  import { Heading } from "@components/ui/heading"
8  import { Separator } from "@components/ui/separator"
9
10 export const BillboardClient = () => {
11   const router = useRouter();
12   const params = useParams();
13
14   return (
15     <>
16       <div className="flex items-center justify-between">
17         <Heading
18           title="Billboards (0)"
19           description="Kelola Billboard untuk toko Anda"
20         />
21         <Button onClick={() => router.push(`/${params.storeId}/billboards/new`)}>
22           <Plus className="mr-2 h-4 w-4"/>
23           Add New
24         </Button>
25       </div>
26       <Separator />
27     </>
28   )
29 }
30
```

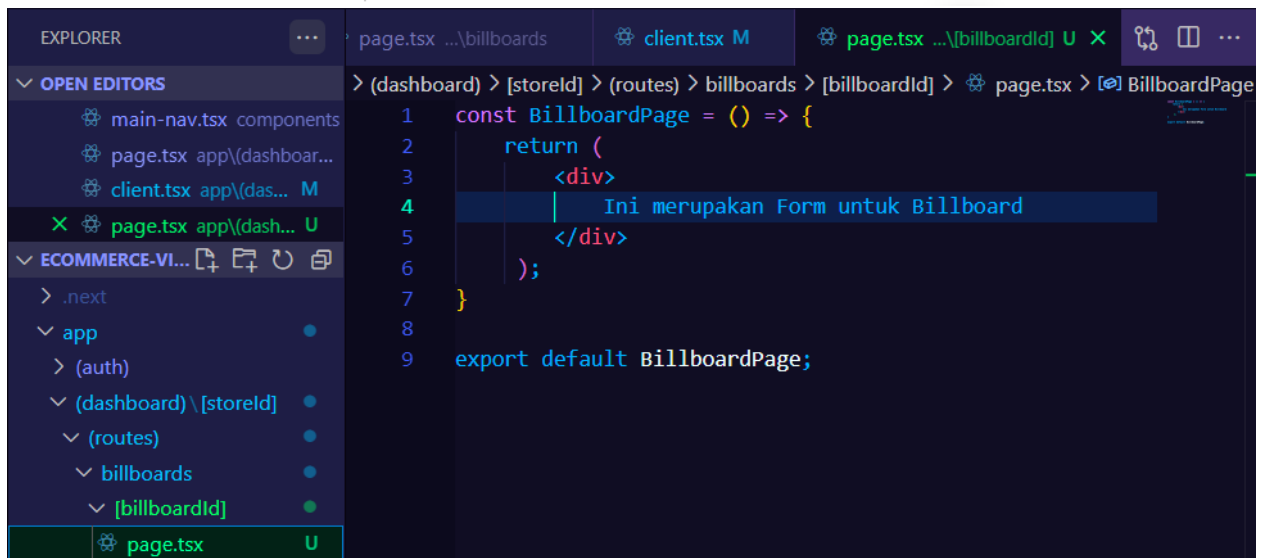
Membuat onClick pada button tambah billboards

hasil :



404

This page could not be found.

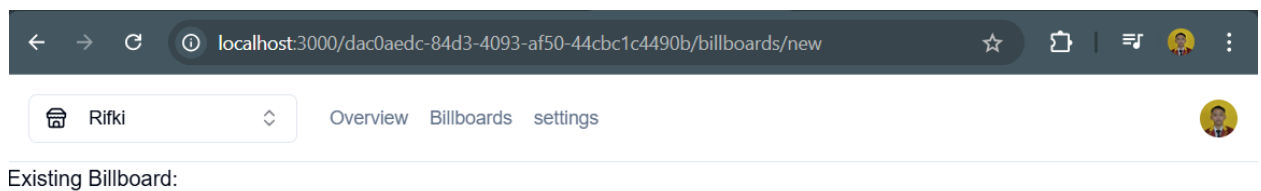


Membuat folder baru Bernama [billboardId] didalam folder billboards dan membuat lembar baru Bernama page.tsx untuk membuat lembar form

```
page.tsx ...\billboards client.tsx page.tsx ...\[billboardId] M X
> (dashboard) > [storeId] > (routes) > billboards > [billboardId] > page.tsx > BillboardPage
1 import db from "@lib/db";
2
3 const BillboardPage = async({
4   params
5 }: {
6   params: {billboardId: string}
7 }) => {
8   const billboard = await db.billboard.findUnique({
9     where: {
10       id: params.billboardId
11     }
12   })
13
14   return (
15     <div>
16       Existing Billboard: {billboard?.label}
17     </div>
18   );
19 }
20
21 export default BillboardPage;
```

Membeikan id halaman berdasarkan label pada billboard

hasil:




```
tsx | page.tsx ...\billboards | client.tsx | page.tsx ...\[billboardId] M X | billboard-form.tsx U | ...
app > (dashboard) > [storeId] > (routes) > billboards > [billboardId] > page.tsx > [BillboardPage]
1  import db from "@lib/db";
2  import { BillboardForm } from "../components/billboard-form";
3
4  const BillboardPage = async({
5    params
6  }): {
7    params: {billboardId: string}
8  } => {
9    const billboard = await db.billboard.findUnique({
10      where: {
11        id: params.billboardId
12      }
13    })
14
15    return (
16      <div className="flex-col">
17        <div className="flex-1 space-y-4 p-8 pt-6">
18          <BillboardForm initialData={billboard}/>
19        </div>
20      </div>
21    );
22  }
23
24  export default BillboardPage;
```

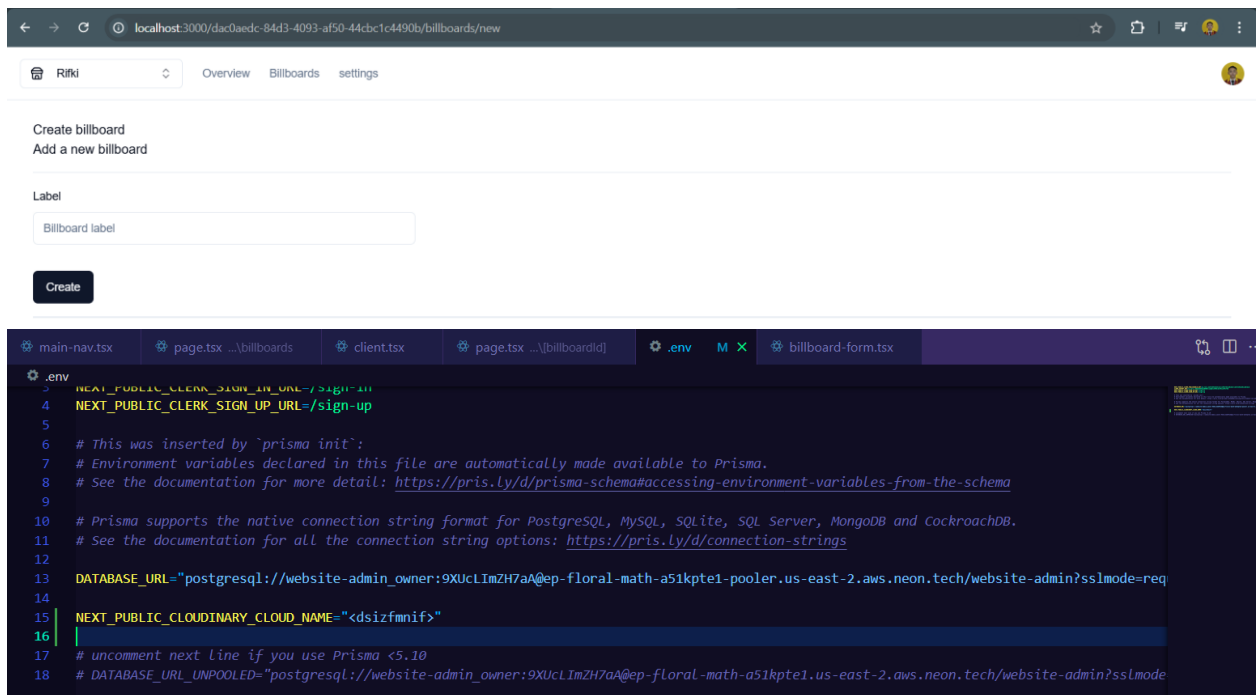
Menambahkan untuk memanggil BillboardForm pada Page/[billboardId]

```
File Edit Selection View Go Run Terminal Help | ecommerce-visionvortex | ...
EXPLORER | main-nav.tsx | page.tsx ...\billboards | client.tsx | page.tsx ...\[billboardId] M | billboard-form.tsx U | ...
OPEN EDITORS | app > (dashboard) > [storeId] > (routes) > billboards > [billboardId] > components > billboard-form.tsx > [BillboardForm]
41  export const BillboardForm: React.FC<BillboardFormProps> = ({
93    <AlertModal
94      isOpen={open}
95      onClose={() => setOpen(false)}
96      onconfirm={onDelete}
97      loading={loading}
98    />
99    <div className="flex items-center justify-between">
100      <Heading
101        title={title}
102        description={description}
103      />
104      {initialData && (
105        <Button
106          disabled={!loading}
107          variant="destructive"
108          size="icon"
109          onClick={() => setOpen(true)}
110        >
111        <Trash className="h-4 w-4" />

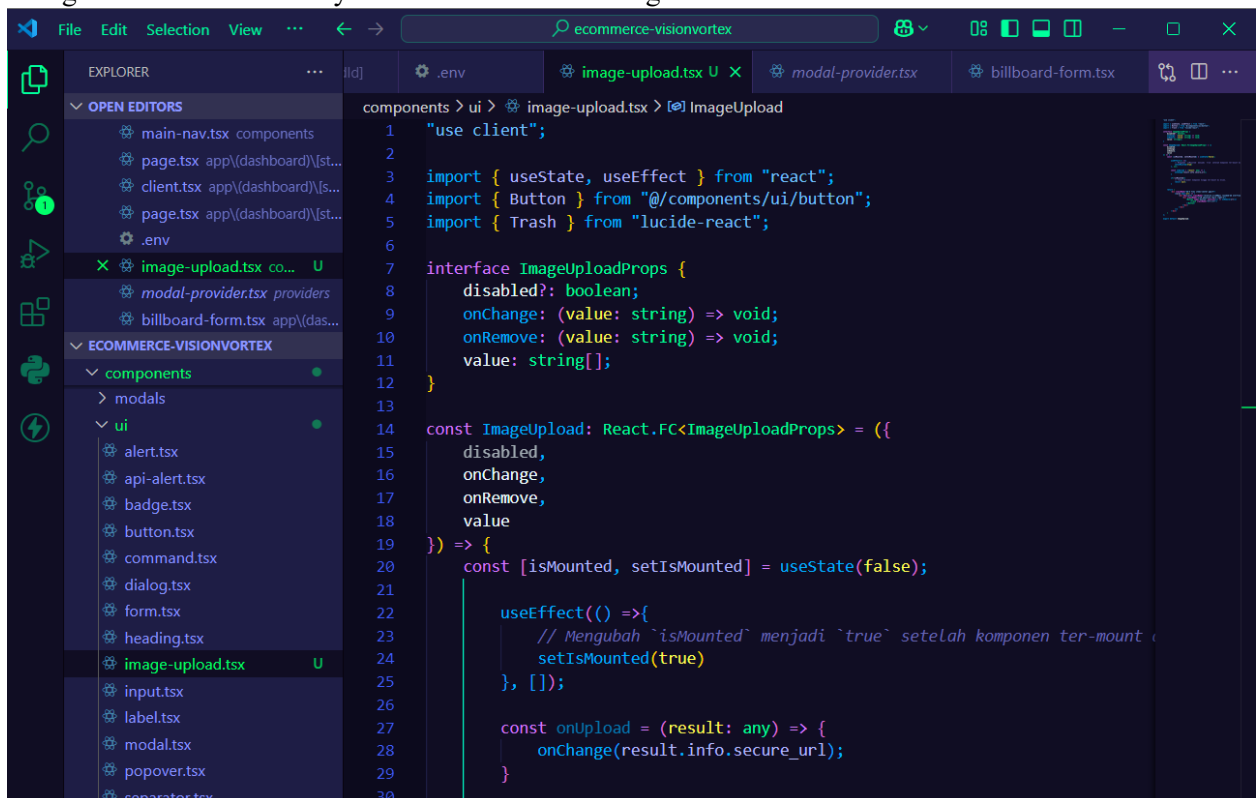
```

Membuat component baru dari folder [billboardId] dan lalu mengcopy file settings-form.tsx dan di paste di dalam folder components dan memberikan nama file baru billboard-form.tsx dan modifikasi isi dalam form tersebut.

hasil:



Meng install next-cloudinary dan menambahkan konfigurasi di .env



Membuat file baru didalam components/ui/ yang diberi nama image-upload.tsx yang didalamnya kita isikan untuk meng upload image pada billboards.

```
components > ui > image-upload.tsx > ImageUpload > value.map() callback
16  const ImageUpload: React.FC<ImageUploadProps> = ({
39    return (
40      <div>
41        <div className="mb-4 flex items-center gap-4">
42          {value.map((url) => (
43            <div
44              key={url}
45              className="relative w-[200px] h-[200px] rounded-md overflow-hidden"
46            >
47              <div className="z-10 absolute top-2 right-2">
48                <Button
49                  type="button"
50                  onClick={() => onRemove(url)}
51                  variant="destructive"
52                  size="icon"
53                >
54                  <Trash className="h-4 w-4" />
55                </Button>
56              </div>
57              <Image fill className="object-cover" alt="Image" src={url} />
58            </div>
59          ))}
60        </div>
61        <CldUploadWidget onUpload={onUpload} uploadPreset="visionvortex">
62          ({open} => {
63            const onClick = () => {
64              open();
65            };
66            return (
67              <Button
68                type="button"
69                disabled={disabled}
70                variant="secondary"
71                onClick={onClick}
72              >
73                <ImagePlus className="h-4 w-4 mr-2" />
74                Upload image
75              </Button>

```

Menampilkan daftar gambar yang dapat dihapus oleh pengguna serta menyediakan tombol untuk mengunggah gambar baru menggunakan widget upload Cloudinary.

```

119 <FormField
120   control={form.control}
121   name="imageUrl"
122   render={({field}) => (
123     <FormItem>
124       <FormLabel>Baground Image</FormLabel>
125       <FormControl>
126         <ImageUpload
127           value={field.value ? [field.value] : []}
128           disabled={loading}
129           onChange={(url) => field.onChange(url)}
130           onRemove={() => field.onChange("")}
131         />
132       </FormControl>
133       <FormMessage />
134     </FormItem>
135   )}
136 />

```

Membuat form untuk mengunggah dan mengatur URL gambar latar (background image)

Hasil :



```

65 const onSubmit = async (data: BillboardFormValues) =>{
66     try {
67         setLoading(true);
68         if (initialData) {
69             await axios.patch(`/api/${params.storeId}/billboards/${params.billboardId}`, data);
70         } else {
71             await axios.post(`/api/${params.storeId}/billboards`, data);
72         }
73         router.refresh();
74         toast.success(toastMessage);
75     } catch (error) {
76         toast.error("Something went wrong.");
77     } finally{
78         setLoading(false);
79     }
80 };
81
82 const onDelete = async () => {
83     try {
84         setLoading(true)
85         await axios.delete(`/api/${params.storeId}/billboards/${params.billboardId}`);
86         router.refresh();
87         router.push("/")
88         toast.success("Billboard deleted.")
89     } catch (error) {
90         toast.error("Make sure you removed all categories using this billboard first.");
91     } finally{
92         setLoading(false)
93         setOpen(false)
94     }
95 }

```

mengupdate untuk file billboard-form.tsx

```
1 import db from "@lib/db";
2 import { auth } from "@clerk/nextjs/server";
3 import { useParams } from "next/navigation";
4 import { NextResponse } from "next/server";
5
6 export async function POST(
7   req: Request,
8   {params}: {params: {storeId: string}}
9 ) {
10   try{
11     const { userId } = await auth()
12     const body = await req.json();
13
14     const {label, imageUrl} = body
15
16     if (!userId){
17       return new NextResponse("Unauthenticated", {status: 401})
18     }
19
20     if (!label){
21       return new NextResponse("label Toko Perlu diinput", {status: 400})
22     }
23
24     if (!imageUrl){
25       return new NextResponse("Gambar URL Perlu diinput", {status: 400})
26     }
27
28     if (!params.storeId) {
29       return new NextResponse("Id Toko Perlu diinput", {status: 400})
30     }
31
32     const storeByUserId = await db.store.findFirst({
33       where: {
34         id: params.storeId,
35         userId
36       }
37     });
38   }
```

membuat folder baru berupa billboards dan file baru route.ts dan memberikan fungsi billboards

```
EXPLORER ***
app > api > [storeId] > billboards > TS routes > GET > @ billboard > where
1 import db from "@lib/db";
2 import { auth } from "@clerk/nextjs/server";
3 import { NextResponse } from "next/server";
4
5
6 export async function GET(
7   req: Request,
8   { params: { billboardId: string } }
9 ) {
10   try {
11     if (!params.billboardId) {
12       return new NextResponse("Papan iklan id dibutuhkan", { status: 400 });
13     }
14
15     const billboard = await db.billboard.findUnique({
16       where: {
17         id: params.billboardId,
18       },
19     });
20
21     return NextResponse.json(billboard);
22   } catch (error) {
23     console.log("[BILLBOARD_GET]", error);
24     return new NextResponse("Internal error", { status: 500 });
25   }
26 }
27
28 export async function PATCH(
29   req: Request,
30   { params: { storeId: string, billboardId: string } }
31 ) {
32   try {
33     const { userId } = await auth();
34     const body = await req.json();
35
36     const { label, imageUrl } = body;
37
38     if (!userId) {
39       return new NextResponse("Unauthenticated", { status: 401 });
40     }
41
42     if (!label) {
43       return new NextResponse("Label menginput nama", { status: 400 });
44     }
45
46     if (!imageUrl) {
47       return new NextResponse("menginput Gambar Url", { status: 400 });
48     }
49
50     const storeByUserId = await db.store.findFirst({
51       where: {
52         id: params.storeId,
53         userId,
54       },
55     });
56
57     if (!storeByUserId) {
58       return new NextResponse("Unauthorized", { status: 403 });
59     }
60
61     // Membuat entri baru pada tabel 'store' dalam database.
62     const billboard = await db.billboard.create({
63       data: {
64         label,
65         imageUrl,
66         storeId: params.storeId,
67       },
68     });
69
70     return new NextResponse(JSON.stringify(billboard), { status: 201 });
71   } catch (error) {
72     console.log("[BILLBOARD_PATCH]", error);
73     return new NextResponse("Internal error", { status: 500 });
74   }
75 }
76
77 export async function DELETE(
78   req: Request,
79   { params: { billboardId: string } }
80 ) {
81   try {
82     const billboardId = params.billboardId;
83
84     if (!billboardId) {
85       return new NextResponse("Papan iklan id dibutuhkan", { status: 400 });
86     }
87
88     const billboard = await db.billboard.delete({
89       where: {
90         id: billboardId,
91       },
92     });
93
94     return new NextResponse("Papan iklan berhasil dihapus", { status: 200 });
95   } catch (error) {
96     console.log("[BILLBOARD_DELETE]", error);
97     return new NextResponse("Internal error", { status: 500 });
98   }
99 }
100
101 export async function OPTIONS(
102   req: Request,
103   { params: { billboardId: string } }
104 ) {
105   return new NextResponse("Options method not implemented", { status: 405 });
106 }
107
108 export default GET;
```

Membuat fungsi GET, PATCH, DELETE pada billboardId

```
File Edit Selection View Go Run Terminal Help
e-commerce-visionvortex
EXPLORER ***
app > api > [storeId] > billboards > TS routes > POST > @ storeByUserId > where
1 import db from "@lib/db";
2 import { auth } from "@clerk/nextjs/server";
3 import { useParams } from "next/navigation";
4 import { NextResponse } from "next/server";
5
6 export async function POST(
7   req: Request,
8   { params: { storeId: string } }
9 ) {
10   try {
11     const { userId } = await auth();
12     const body = await req.json();
13
14     const { label, imageUrl } = body;
15
16     if (!userId) {
17       return new NextResponse("Unauthenticated", { status: 401 });
18     }
19
20     if (!label) {
21       return new NextResponse("Label Toko Perlu diinput", { status: 400 });
22     }
23
24     if (!imageUrl) {
25       return new NextResponse("Gambar URL Perlu diinput", { status: 400 });
26     }
27
28     if (!params.storeId) {
29       return new NextResponse("Id Toko Perlu diinput", { status: 400 });
30     }
31
32     const storeByUserId = await db.store.findFirst({
33       where: {
34         id: params.storeId,
35         userId,
36       },
37     });
38
39     if (!storeByUserId) {
40       return new NextResponse("Unauthorized", { status: 403 });
41     }
42
43     // Membuat entri baru pada tabel 'store' dalam database.
44     const billboard = await db.billboard.create({
45       data: {
46         label,
47         imageUrl,
48         storeId: params.storeId,
49       },
50     });
51
52     return new NextResponse(JSON.stringify(billboard), { status: 201 });
53   } catch (error) {
54     console.log("[BILLBOARD_POST]", error);
55     return new NextResponse("Internal error", { status: 500 });
56   }
57 }
58
59 export default POST;
```

Memperbaiki route pada billboard

```

39
40 model Category {
41   id String @id @default(uuid())
42   storeId String
43   store Store @relation("StoreToCategory", fields: [storeId], references: [id])
44   billboardId String
45   billboard Billboard @relation(fields: [billboardId], references: [id])
46   name String
47   createdAt DateTime @default(now())
48   updatedAt DateTime @updatedAt
49
50   @@index([storeId])
51   @@index([billboardId])
52 }

```

Menambahkan model prisma baru berupa model Category.

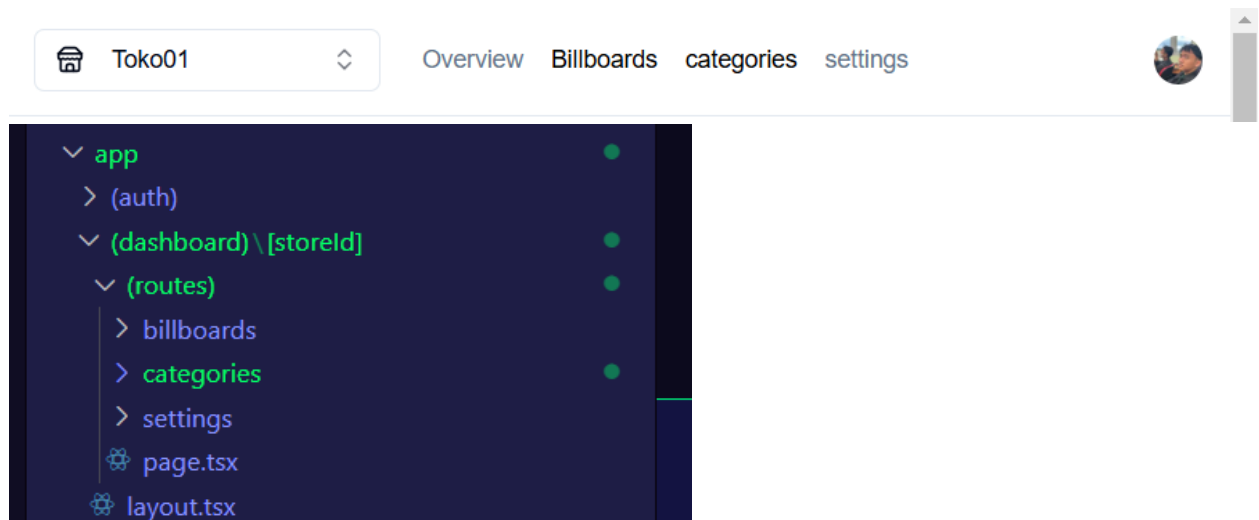
```

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

```

Menambahkan navbar berupa menu baru yaitu categories.

Hasil :



Mengcopy folder billboards dan merename menjadi categories hingga menu categories pun bisa terpanggil.

hasil :

←

→

↺

localhost:3000/232157a8-c166-43ab-8102-731cc8570d73/cat... ☆

🏠

📑

👤

⋮

Toko01 ▾

Overview Billboards categories settings

👤

Billboards(1)

Kelola Billboard untuk toko Anda

+ Add New

Search

Label	Date	
Rifki Ganteng	January 20th, 2025	...

Previous

Next

API

API calls for Billboards

📄 GET public

`http://localhost:3000/ap/232157a8-c166-43ab-8102-731cc8570d73/billboards` 📄

⚡ 📄 GET public

```

7  const CategoriesPage = async ({
8    params
9  }): {
10    params: {storeId: string}
11  }) => {
12    const categories= await db.category.findMany({
13      where: {
14        storeId: params.storeId
15      },
16      include: {
17        billboard: true,
18      },
19      orderBy: {
20        createdAt: 'desc'
21      }
22    });
23
24    const formattedCategories: CategoryColumn[] = categories.map((item) => ({
25      id: item.id,
26      name: item.name,
27      billboardLabel: item.billboard.label,
28      createdAt: formatDate(item.createdAt, "MMMM do, yyyy" )
29    }));
30
31    return (
32      <div className="flex-col">
33        <div className="flex-1 space-y-4 p-8 pt-6">
34          <BillboardClient data={formattedCategories}/>
35        </div>
36      </div>
37    );
38  }
39
40  export default CategoriesPage;

```

Pada page.tsx folder categories kita memakai file lama dan lalu menyesuaikan Namanya menjadi CategoriesPage dan menyesuaikan lainnya dan menambahkan include pada const categories yang sudah saya kotakin orange, dan merubah dan menambahkan isi const pada formattedCategories berupa name dan billboardLabel yang saya kotakin warna yellow.

```

8 export type CategoryColumn = {
9   id: string
10  name: string
11  billboardLabel: string
12  createdAt: string
13 }
14
15 export const columns: ColumnDef<CategoryColumn>[] = [
16   {
17     accessorKey: "name",
18     header: "Name",
19   },
20   {
21     accessorKey: "billboard",
22     header: "Billboard",
23     cell: ({ row }) => row.original.billboardLabel,
24   },
25   {
26     accessorKey: "createdAt",
27     header: "Date",
28   },
29   {
30     id: "Actions",
31     cell: ({row}) => <CellAction data={row.original}/>
32   }
33 ]
34

```

Lalu pada columns.tsx menyesuaikan namasebelumnya menjadi CategoryColumn. Menambahkan string baru berupa name dan billboardLabel seperti yang saya kotakin Orange. Dan lalu menyesuaikan isi const columns seperti yang saya kotakin Yellow.

```

10 import { DataTable } from "@components/ui/data-table"
11 import { Apilist } from "@components/ui/api-list"
12 import { CategoryColumn, columns } from "./columns"
13
14 interface CategoryClientProps{
15   data: CategoryColumn[]
16 }
17
18 export const CategoryClient: React.FC<CategoryClientProps> = ({
19   data

```

Melakukan penyesuaian pada client.tsx dan mengganti menjadi Category.

```

return (
  <>
    <div className="flex items-center justify-between">
      <Heading
        title={`Categories(${data.length})`}
        description="Kelola Kategori untuk toko Anda"
      />
      <Button onClick={() => router.push(`/${params.storeId}/categories/new`)}>
        <Plus className="mr-2 h-4 w-4"/>
        Add New
      </Button>
    </div>
    <Separator />
    <DataTable searchKey="label" columns={columns} data={data}/>
    <Heading title="API" description="API calls for Categories"/>
    <Separator/>
    <ApiList entityName="categories" entityIdName="categoryId"/>
  </>
)

```

Melakukan perbaikan pada return dari yang sebelumnya billboard menjadi categories.

Hasil :



- `<DataTable searchKey="name" columns={columns} data={data}/>`

Merubah searchKey menjadi name dan lalu merename folder [billboardId] menjadi [categoryId] seperti dibawah ini :

```

1  import db from "@/lib/db";
2  import { CategoryForm } from "../components/category-form";
3
4  const CategoryPage = async({
5    params
6  }): {
7    params: {categoryId: string}
8  } => {
9    const category = await db.category.findUnique({
10      where: {
11        id: params.categoryId
12      }
13    })
14
15    return (
16      <div className="flex-col">
17        <div className="flex-1 space-y-4 p-8 pt-6">
18          <CategoryForm initialData={category}/>
19        </div>
20      </div>
21    );
22  }
23
24  export default CategoryPage;

```

Lalu pada page.tsx menyesuaikan menjadi category seperti kotak warna orange.

```

categories
├── [categoryId]
│   └── components
│       ├── category-form.tsx
│       └── page.tsx

```

Merename file sebelumnya menjadi category-form.tsx

```

const formSchema = z. object ({
  name: z.string().min(1),
  billboardId: z.string().min(1)
});

type CategoryFormValues = z. infer<typeof formSchema>;

interface CategoryFromProps {
  initialData: Category | null;
}

export const CategoryForm: React.FC<CategoryFromProps> = ({
  initialData
}) => {
  const params = useParams();
  const router = useRouter();

  const [open, setOpen] = useState(false);
  const [loading, setLoading] = useState(false);

  const title = initialData ? "Edit category" : "Create category";
  const description = initialData ? "Edit category" : "Add a new category";
  const toastMessage = initialData ? "Category update" : "Category created";
  const action = initialData ? "Save change" : "Create";

  const form = useForm<CategoryFormValues>({
    resolver: zodResolver(formSchema),
    defaultValues: initialData || {
      name: '',
      billboardId: '',
    }
  });
};

```

Pada const formSchema diganti menjadi const name dan billboardId. Dan initialData diganti menjadi category. Lalu pada const form diganti menjadi name dan billboardId, seperti warna orange.

Merubah nama billboard menjadi category pada kotak yellow.

```

</div>
<Separator />
<Form {...form}>
  <form onSubmit={form.handleSubmit(onSubmit)} className="space-y-8 w-full">
    <div className="grid grid-cols-3 gap-8">
      <FormField
        control={form.control}
        name="name"
        render={({field}) => (
          <FormItem>
            <FormLabel>Name</FormLabel>
            <FormControl>
              <Input disabled={loading} placeholder="Category name" {...fi
            </FormControl>
            <FormMessage />
          </FormItem>
        )}
      />
    </div>
    <Button disabled={loading} className="ml-auto" type="submit">
      {action}
    </Button>
  </form>
</Form>

```

Membuat form categorynya.

Hasil :

Create category

Add a new category

Name

Create

```

PS D:\Universitas Teknokrat Indonesia\Semester 5\WEB 2\ecommerce-visionvortex> npx shadcn@latest add select
✓ Checking registry.
✓ Installing dependencies.
✓ Created 1 file:
  - components/ui/select.tsx

```

Menginstall ui select.

```
<FormField
  control={form.control}
  name="billboardId"
  render={({field}) => (
    <FormItem>
      <FormLabel>Billboard</FormLabel>
      <Select
        disabled={loading}
        onChange={field.onChange}
        value={field.value}
        defaultValue={field.value}
      >
        <FormControl>
          <SelectTrigger>
            <SelectValue
              defaultValue={field.value}
              placeholder="Select a Billboard"
            />
          </SelectTrigger>
        </FormControl>
        <SelectContent>
          </SelectContent>
        </Select>
        <FormMessage />
      </FormItem>
    )
  )
/>
```

Membuat FormField baru yang berisikan select pada form category.

```
const CategoryPage = async({
  params
}): {
  params: {categoryId: string, storeId: string}
} => {
```

Menambahkan string baru berupa storeId, pada page.tsx

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

Menambahkan categoryForm berupa billboards pada page.tsx.

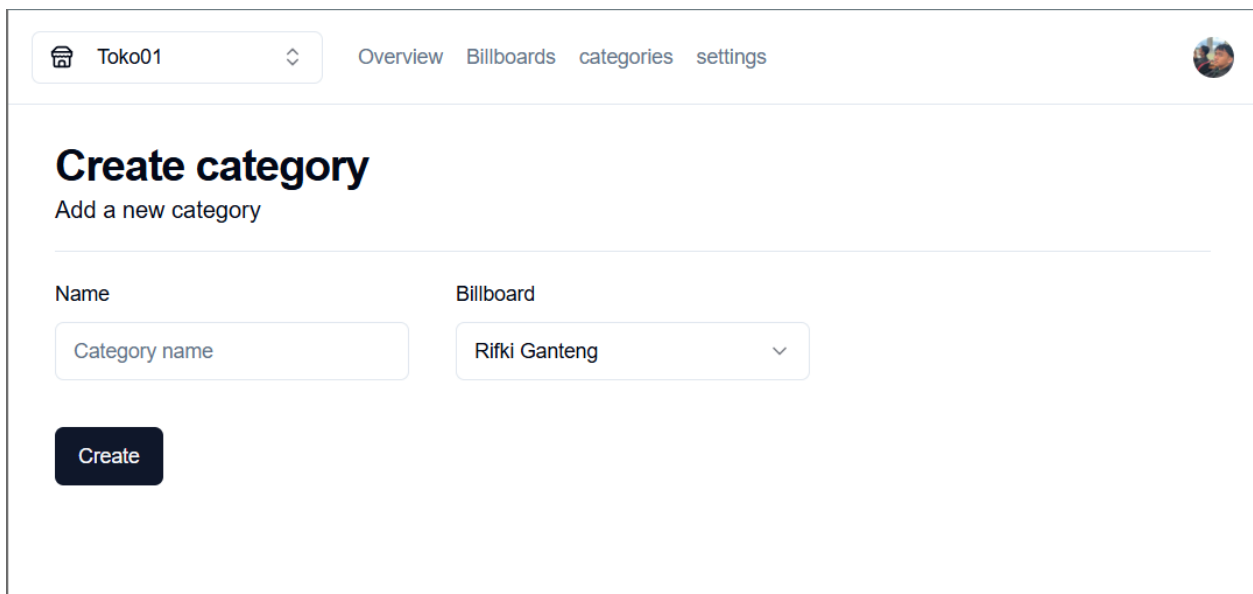

```
interface CategoryFromProps {
  initialData: Category | null;
  billboards: Billboard[];
}
```

Pada categoryForm menambahkan interface baru yaitu billboards.

```
<SelectContent>
  {billboards.map((billboards) => (
    <SelectItem
      key={billboards.id}
      value={billboards.id}
    >
      {billboards.label}
    </SelectItem>
  ))}
</SelectContent>
```

Menambahkan select content sesuai pada label billboard.

Hasil :



The screenshot shows a web application interface for creating a new category. At the top, there is a header bar with a store icon and the name 'Toko01', followed by navigation links: 'Overview', 'Billboards', 'categories', and 'settings'. A user profile picture is visible on the right. Below the header, the main section is titled 'Create category' with the subtitle 'Add a new category'. The form consists of two input fields: 'Name' with the placeholder text 'Category name' and 'Billboard' which is a dropdown menu currently showing 'Rifki Ganteng'. A dark blue 'Create' button is located at the bottom left of the form area.

```

const onSubmit = async (data: CategoryFormValues) =>{
  try {
    setLoading(true);
    if (initialData) {
      await axios.patch(`/api/${params.storeId}/categories/${params.categoryId}`, data);
    } else {
      await axios.post(`/api/${params.storeId}/categories`, data);
    }
    router.refresh();
    router.push(`/api/${params.storeId}/categories`)
    toast.success(toastMessage);
  } catch (error) {
    toast.error("Something went wrong.");
  }finally{
    setLoading(false);
  }
};

const onDelete = async () => {
  try {
    setLoading(true)
    await axios.delete(`/api/${params.storeId}/categories/${params.categoryId}`);
    router.refresh();
    router.push(`/api/${params.storeId}/categories`)
    toast.success("Category deleted.")
  } catch (error) {
    toast.error("Make sure you removed all products using this category first.");
  }finally{
    setLoading(false)
    setOpen(false)
  }
}

```

Merubah code sebelumnya menjadi categories dan category.

```

  api
  [storeId]
    > billboards
    > categories
    > stores

```

Mengcopy folder billboards dan mem-paste lalu merename menjadi categories.

```

export async function POST(
  req: Request,
  {params}: {params: {storeId: string}}
) {
  try{
    const { userId } = await auth()
    const body = await req.json();

    const {name, billboardId} = body

    if (!userId){
      return new NextResponse("Unauthenticated", {status: 401})
    }

    if (!name){
      return new NextResponse("Nama Toko Perlu diinput", {status: 400})
    }

    if (!billboardId){
      return new NextResponse("Billboard Perlu diinput", {status: 400})
    }

    if (!params.storeId) {
      return new NextResponse("Id Toko Perlu diinput", {status: 400})
    }

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

```

Pada fungsi post memberikan perbaikan yaitu memperbaiki fungsi name dan billboardId seperti kotak orange.

```

const category = await db.category.create({
  data: {
    name,
    billboardId,
    storeId: params.storeId
  },
});

```

Pada const category menambahkan name, billboardId.

```

return NextResponse.json(category);
} catch (error) {
  console.log("[CATEGORIES_POST]", error)
  return new NextResponse("Internal error", {status: 500})
}

```

Melakukan return category.

```

try{
  if (!params.storeId) {
    return new NextResponse("Id Toko Perlu diinput", {status: 400})
  }

  const categories = await db.category.findMany({
    where: {
      storeId: params.storeId,
    }
  });

  return NextResponse.json(categories);
} catch (error) {
  console.log("[CATEGORIES_GET]", error)
  return new NextResponse("Internal error", {status: 500})
}

```

Memperbaiki menjadi categories seperti kotak orange

```

▼ categories ●
▼ [categoryId] ●
TS route.ts U
TS route.ts

```

Mencopy-paste dan merename menjadi [categoryId]

```

export async function GET(
  req: Request,
  { params }: { params: { categoryId: string } }
) {
  try {
    if (!params.categoryId) {
      return new NextResponse("kategori id dibutuhkan", { status: 400 });
    }

    const category = await db.category.findUnique({
      where: {
        id: params.categoryId,
      },
    });

    return NextResponse.json(category);
  } catch (error) {
    console.log("[CATEGORY_GET]", error);
    return new NextResponse("Internal error", { status: 500 });
  }
}

```

Didalam route.ts memperbaiki code sebelumnya pada fungsi get dari billboard menjadi category.

```

export async function PATCH(
  req: Request,
  { params }: { params: { storeId: string, categoryId: string } }
) {
  try {
    const { userId } = await auth();
    const body = await req.json();

    const { name, billboardId } = body;

    if (!userId) {
      return new NextResponse("Unauthenticated", { status: 401 });
    }
    if (!name) {
      return new NextResponse("Name menginput nama", { status: 400 });
    }
    if (!billboardId) {
      return new NextResponse("menginput billboard", { status: 400 });
    }
    if (!params.categoryId) {
      return new NextResponse("Kategori Diinputkan", { status: 400 });
    }

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

```

Melakukan hal yang sama pada fungsi patch

```

app > api > [storeId] > categories > [categoryId] > TS route.ts > PATCH > billboardId
79 export async function DELETE(
80   req: Request,
81   { params }: { params: { storeId: string, categoryId: string } }
82 ) {
83   try {
84     const { userId } = await auth();
85
86
87     if (!userId) {
88       return new NextResponse("Unauthenticated", { status: 401 });
89     }
90
91     if (!params.categoryId) {
92       return new NextResponse("Kategori id dibutuhkan", { status: 400 });
93     }
94
95     const storeByUserId = await db.store.findFirst({
96       where: {
97         id: params.storeId,
98         userId
99       }
100     });
101
102     if (!storeByUserId) {
103       return new NextResponse("Unauthorized", { status: 403 });
104     }
105
106     const category = await db.category.deleteMany({
107       where: {
108         id: params.categoryId,
109       },
110     });
111
112     return NextResponse.json(category);
113   } catch (error) {
114     console.log("[CATEGORY_DELETE]", error);
115     return new NextResponse("Internal error", { status: 500 });

```

Pada fungsi delete melakukan hal yang sama merubah billboard menjadi category.

Hasil :

Toko01

OverviewBillboardscategoriessettings

Categories(1)

Kelola Kategori untuk toko Anda

+ Add New

Search

Name	Billboard	Date	
Testing-1	Rifki Ganteng	January 21st, 2025	...

Previous

Next

API

API calls for Categories

GET public

http://localhost:3000/ap/232157a8-c166-43ab-8102-731cc8570d73/categories

GET public

Berhasil menambahkan data kategori.

```
interface CellActionProps {
  data: CategoryColumn
}
```

Menambahkan interface datanya categoryColumn.

```
const onDelete = async () => {
  try {
    setLoading(true)
    await axios.delete(`/api/${params.storeId}/categories/${data.id}`);
    router.refresh();
    toast.success("Category deleted.")
  } catch (error) {
    toast.error("Make sure you removed all products using this category first.");
  } finally {
    setLoading(false)
    setOpen(false)
  }
}
```

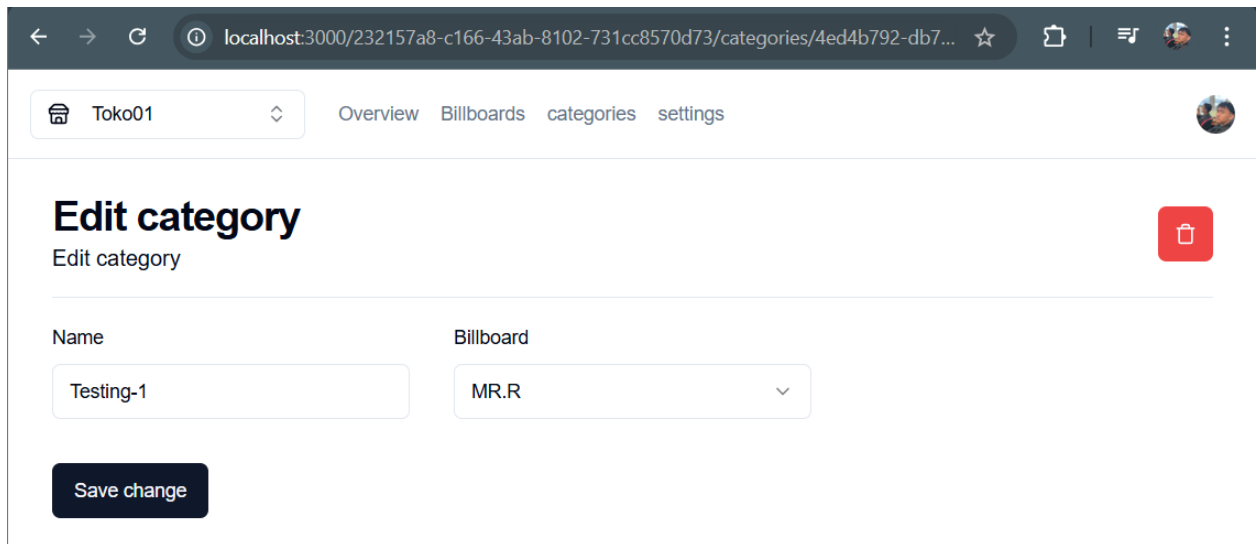

Pada onDelete di perbaiki menjadi categories.

```
<DropdownMenuItem onClick={() => router.push(`/ ${params.storeId}/categories/${data.id}`)}>  
  <Edit className="mr-2 h-4 w-4" />  
  Update  
</DropdownMenuItem>
```

Melakukan DropdownMenuItem untuk update categories.

Hasil :

Update:



The screenshot shows a web browser at localhost:3000. The application has a dark header with navigation links: Toko01, Overview, Billboards, categories, and settings. The main content area is titled 'Edit category' and contains a form with two fields: 'Name' (with value 'Testing-1') and 'Billboard' (with value 'MR.R'). A 'Save change' button is at the bottom left of the form. A red trash icon is visible in the top right corner of the form area.

Delete:

localhost:3000/232157a8-c166-43ab-8102-731cc8570d73/categories

☆

Toko01

OverviewBillboardscategoriessettings

Categories(4)

Kelola Kategori untuk toko Anda

+ Add New

Search

Name	Billboard	Date	
Testing-3			...
Test-1			...
Testing-2	Rifki Ganteng	January 21st, 2025	...
Testing-1	MR.R	January 21st, 2025	...

PreviousNext

Are you sure?

This action cannot be undone.

CancelContinue

←

→

↺


📍

localhost:3000/232157a8-c166-43ab-8102-731cc8570d73/categories


☆

🔖

☰




⋮

 Toko01

⇅

OverviewBillboardscategoriessettings



Categories(3)

Kelola Kategori untuk toko Anda

+ Add New

Search

Name	Billboard	Date	
Testing-3	MR.R	January 21st, 2025	...
Testing-2	Rifki Ganteng	January 21st, 2025	...
Testing-1	MR.R	January 21st, 2025	...

Previous

Next

Search :

←

→

↺


📍

localhost:3000/232157a8-c166-43ab-8102-731cc8570d73/categories


☆

🔖

☰




⋮

 Toko01

⇅

OverviewBillboardscategoriessettings



Categories(3)

Kelola Kategori untuk toko Anda

+ Add New

2

Name	Billboard	Date	
Testing-2	Rifki Ganteng	January 21st, 2025	...

Previous

Next

```

84 model Product {
85     id String @id @default(uuid())
86     storeId String
87     store Store @relation("StoreToProduct", fields: [storeId], references: [id])
88     categoryId String
89     category Category @relation("CategoryToProduct", fields: [categoryId], references: [id])
90     name String
91     price Decimal
92     isFuture Boolean @default(false)
93     isArchived Boolean @default(false)
94     sizeId String
95     size Size @relation(fields: [sizeId], references: [id])
96     colorId String
97     color Color @relation(fields: [colorId], references: [id])
98     images Image[]
99     createdAt DateTime @default(now())
100    updatedAt DateTime @updatedAt
101
102    @@index([storeId])
103    @@index([categoryId])
104    @@index([sizeId])
105    @@index([colorId])
106 }

```

menambahkan model baru berupa Product

```

108 model Image {
109     id String @id @default(uuid())
110     productId String
111     product Product @relation(fields: [productId], references: [id], onDelete: Cascade)
112     url String
113     createdAt DateTime @default(now())
114     updatedAt DateTime @updatedAt
115
116     @@index([productId])
117 }

```

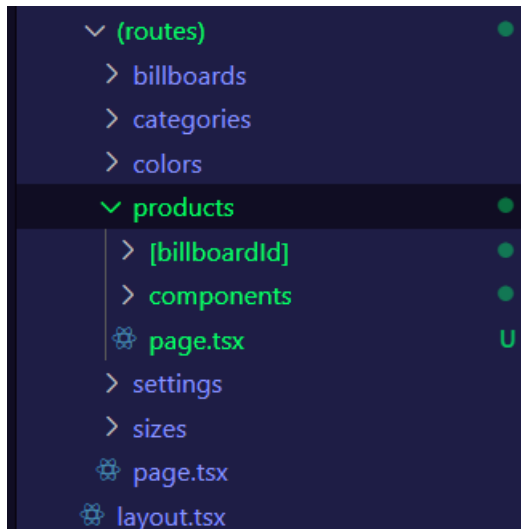
Menambahkan model baru yaitu Image

```

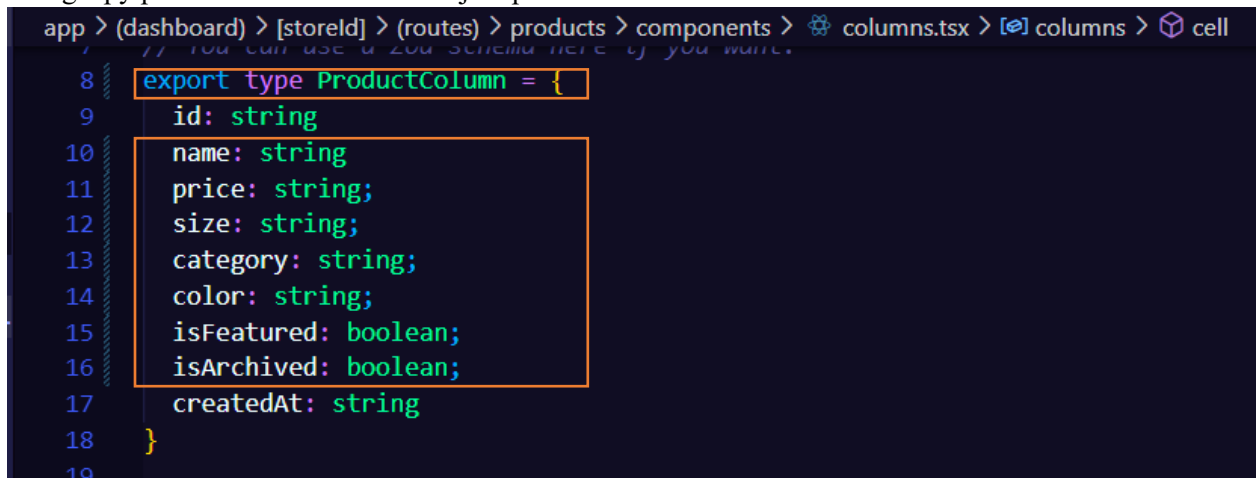
{
    href: `/${params.storeId}/products`,
    label: 'Products',
    active: pathname === `/${params.storeId}/products`,
},

```

menambahkan navigasi baru berupa products



Mengcopy paste folder billboards menjadi products



Pada column merubah sebelumnya menjadi productColumn dan menambahkan type data

```
export const columns: ColumnDef<ProductColumn>[] = [
  {
    accessorKey: "name",
    header: "Name",
  },
  {
    accessorKey: "isArchived",
    header: "Archived",
  },
  {
    accessorKey: "isFeatured",
    header: "Featured",
  },
  {
    accessorKey: "price",
    header: "Price",
  },
  {
    accessorKey: "category",
    header: "Category",
  },
  {
    accessorKey: "size",
    header: "Size",
  },
],
```

Merubah BillboardtColumn menjadi ProductColumn dan menambahkan isi export const baru pada column

```
{
  accessorKey: "color",
  header: "Color",
  cell: ({row}) => (
    <div className="h-6 w-6 rounded-full border">
      {row.original.color}
      <div
        className="h-6 w-6 rounded-full border"
        style={{ backgroundColor: row.original.color
      />
    </div>
  )
},
```

untuk bagian color menambahkan div class baru untuk bordernya.

```
schema.prisma M × main-nav.tsx page.tsx M client.tsx M X columns.tsx M TS utils.ts M
app > (dashboard) > [storeId] > (routes) > products > components > client.tsx > ProductClient
13
14 interface ProductClientProps{
15   data: ProductColumn[]
16 }
17
18 export const ProductClient: React.FC<ProductClientProps> = ({
19   data
20 }) => {
21   const router = useRouter();
22   const params = useParams();
23
24
25   return (
26     <>
27       <div className="flex items-center justify-between">
28         <Heading
29           title={`Products(${data.length})`}
30           description="Kelola Produk untuk toko Anda"
31         />
32         <Button onClick={() => router.push(`/${params.storeId}/products/new`)}>
33           <Plus className="mr-2 h-4 w-4"/>
34           Add New
35         </Button>
36       </div>
37       <Separator />
38       <DataTable searchKey="label" columns={columns} data={data}/>
39       <Heading title="API" description="API calls for Products"/>
40       <Separator/>
41       <ApiList entityName="products" entityIdName="productId"/>
42     </>
43   )
44 }
```

• merubah dan mengganti Billboard sebelumnya menjadi Products/product

```
const ProductsPage = async ({
  params
}): {
  params: {storeId: string}
} => {
  const products= await db.product.findMany({
    where: {
      storeId: params.storeId
    },
    include: {
      category: true,
      size: true,
      color: true,
    },
    orderBy: {
      createdAt: 'desc'
    }
  });

  const formattedProducts: ProductColumn[] = products.map((item) => {
    id: item.id,
    name: item.name,
    isFeatured: item.isFeatured,
    isArchived: item.isArchived,
    price: formatter.format(item.price.toNumber()),
    category: item.category.name,
    size: item.category.name,
    color: item.color.value,
    createdAt: formatDate(item.createdAt, "MMMM do, yyyy" )
  }));

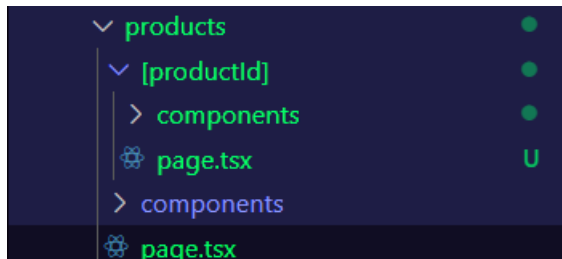
  return (
    <div className="flex-col">
      <div className="flex-1 space-y-4 p-8 pt-6">
        <ProductClient data={formattedProducts}/>
      </div>
    </div>
  );
};
```

pada bagian page merubah kata billboards menjadi products dan menambahkan include baru didalamnya.

```
const formattedProducts: ProductColumn[] = products.map((item) => {
  id: item.id,
  name: item.name,
  isFeatured: item.isFeatured,
  isArchived: item.isArchived,
  price: formatter.format(item.price.toNumber()),
  category: item.category.name,
  size: item.category.name,
  color: item.color.value,
  createdAt: formatDate(item.createdAt, "MMMM do, yyyy" )
}));

return (
  <div className="flex-col">
    <div className="flex-1 space-y-4 p-8 pt-6">
      <ProductClient data={formattedProducts}/>
    </div>
  </div>
);
```

merubah billboard menjadi product/products dan menambahkan const baru didalamnya.



merename billboardId menjadi productid

- ```

36 const formSchema = z. object ({
37 name: z.string().min(1),
38 images: z.object({ url: z.string() }).array(),
39 price: z.coerce.number().min(1),
40 categoryId: z.string().min(1),
41 colorId: z.string().min(1),
42 sizeId: z.string().min(1),
43 isFeatured: z.boolean().default(false).optional(),
44 isArchived: z.boolean().default(false).optional(),
45 });
46

```

pada product form menambahkan isi const form schema

- ```

49  interface ProductFromProps {
50      initialData: Product & {
51          images: Image[]
52      } | null;
53      categories: Category[];
54      colors: Color[];
55      sizes: Size[];
56  }

```

pada interfacenya menambahkan categories, colors, dan sizes

- ```

58 export const ProductForm: React.FC<ProductFromProps> = ({
59 initialData,
60 categories,
61 colors,
62 sizes
63 }) => {

```

menambahkan isi const product form berupa categories, colors, sizes.

```
const form = useForm<ProductFormValues>({
 resolver: zodResolver(formSchema),
 defaultValues: initialData ? {
 ...initialData,
 price: parseFloat(String(initialData?.price)),
 } : {
 name: '',
 images: [],
 price: 0,
 categoryId: '',
 colorId: '',
 sizeId: '',
 isFeatured: false,
 isArchived: false,
 }
});
```

default values menambahkan isinya seperti diatas

```
<ImageUpload
 value={field.value.map((image) => image.url)}
 disabled={loading}
 onChange={(url) => field.onChange([...field.value, { url }])}
 onRemove={(url) => field.onChange([...field.value.filter((current) => current.url !== url])]}
/>
```

menambahkan field pada image upload di form image

```
<FormField
 control={form.control}
 name="name"
 render={({field}) => (
 <FormItem>
 <FormLabel>Name</FormLabel>
 <FormControl>
 <Input disabled={loading} placeholder="Product name" {...field} />
 </FormControl>
 <FormMessage />
 </FormItem>
)}
/>
```

menambahkan berupa form field nama

```

<FormField
 control={form.control}
 name="price"
 render={({field}) => (
 <FormItem>
 <FormLabel>Price</FormLabel>
 <FormControl>
 <Input type="number" disabled={loading} placeholder="100000"{...field} />
 </FormControl>
 <FormMessage />
 </FormItem>
)}
/>

```

menambahkan form field price pada form product

```

199 <FormField
200 control={form.control}
201 name="categoryId"
202 render={({field}) => (
203 <FormItem>
204 <FormLabel>Category</FormLabel>
205 <Select
206 disabled={loading}
207 onChange={field.onChange}
208 value={field.value}
209 defaultValue={field.value}
210 >
211 <FormControl>
212 <SelectTrigger>
213 <SelectValue
214 defaultValue={field.value}
215 placeholder="Select a Category"
216 />
217 </SelectTrigger>
218 </FormControl>
219 <SelectContent>
220 {categories.map((category) => (
221 <SelectItem
222 key={category.id}
223 value={category.id}
224 >
225 {category.name}
226 </SelectItem>
227))}
228 </SelectContent>
229 </Select>
230 <FormMessage />
231 </FormItem>
232)}

```

menambahkan form field category product


```

/>
<FormField
 control={form.control}
 name="sizeId"
 render={({field}) => (
 <FormItem>
 <FormLabel>Size</FormLabel>
 <Select
 disabled={loading}
 onValueChange={field.onChange}
 value={field.value}
 defaultValue={field.value}
 >
 <FormControl>
 <SelectTrigger>
 <SelectValue
 defaultValue={field.value}
 placeholder="Select a Size"
 />
 </SelectTrigger>
 </FormControl>
 <SelectContent>
 {sizes.map((size) => (
 <SelectItem
 key={size.id}
 value={size.id}
 >
 {size.name}
 </SelectItem>
))}
 </SelectContent>
 </Select>
 <FormMessage />
 </FormItem>
)}
)


```

- menambahkan form field size pada produk

- Hasil :

 Rifki

OverviewBillboardsCategoriesSizesProductsColorssettings




## Buat Product

Buat Product Toko

---

Images

 Upload image

|                                           |                                |                                                |
|-------------------------------------------|--------------------------------|------------------------------------------------|
| Name                                      | Price                          | Category                                       |
| <input type="text" value="Product name"/> | <input type="text" value="0"/> | <input type="text" value="Select a Category"/> |

Size

Buat Product

```

<FormItem>
 <FormLabel>Color</FormLabel>
 <Select
 disabled={loading}
 onValueChange={field.onChange}
 value={field.value}
 defaultValue={field.value}
 >
 <FormControl>
 <SelectTrigger>
 <SelectValue
 defaultValue={field.value}
 placeholder="Select a Color"
 />
 </SelectTrigger>
 </FormControl>
 <SelectContent>
 {colors.map((color) => (
 <SelectItem
 key={color.id}
 value={color.id}
 >
 {color.name}
 </SelectItem>
))}
 </SelectContent>
 </Select>
 <FormMessage />
</FormItem>

```

menambahkan berupa select color pada form field product

```

PS D:\Universitas Teknokrat Indonesia\Semester 5\WEB 2\ecommerce-visionvortex> npx shadcn@latest add checkbox
✓ Checking registry.
✓ Installing dependencies.
✓ Created 1 file:
 - components\ui\checkbox.tsx

```

install checkbox.