

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:



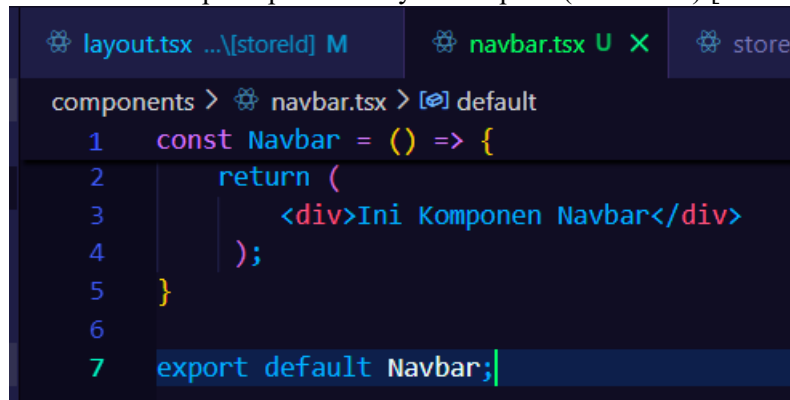
Tanggal : 10 Januari 2025

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
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/popove
9 import { Button } from "@components/ui/button";
10 import { useStoreModal } from "@hooks/use-store-modal";
11 import { cn } from "@lib/utls";
12 import { Command, CommandEmpty, CommandGroup, CommandInput, CommandItem, Comm
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.stor
34
35   const [open, setOpen] = useState(false);
36
37   const onStoreSelect = (store: {value: string, label: string}) => {
```

Mendefinisikan StoreSwitcher didalamnya membuat button took berupa searcging 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. The billboards folder contains a components subfolder with client.tsx and page.tsx files. The code editor shows the content of client.tsx, which defines a BillboardClient component. The component is a function that returns a JSX element with a div containing the text 'Billboard Client'.

```
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
```

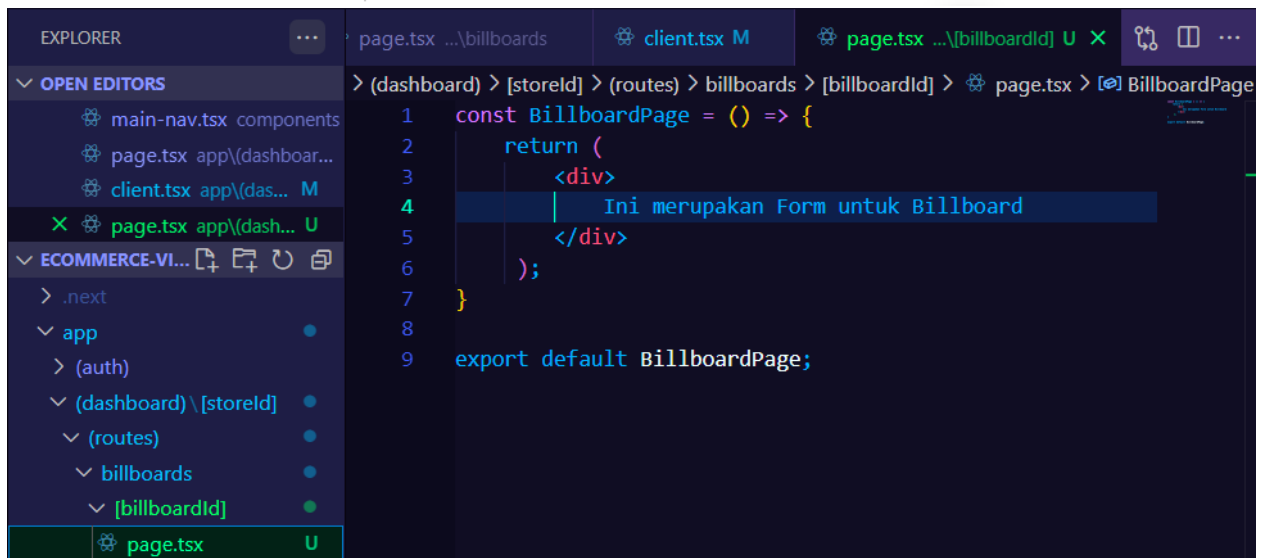
Membuat onClick pada button tambah billboards

hasil :



404

This page could not be found.

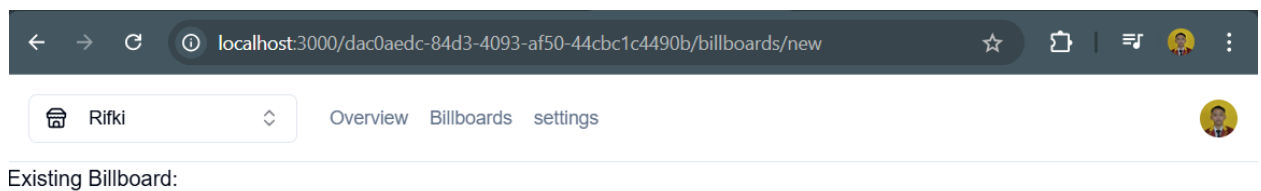


Membuat folder baru Bernama [billboardId] didalam folder billboards dan membuat lembar baru Bernama page.tsx untuk membuat lembar form

```
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 billboard = await db.billboard.create({
  51       data: {
  52         label,
  53         imageUrl,
  54         storeId,
  55       },
  56     });
  57
  58     return new NextResponse(JSON.stringify(billboard), { status: 201 });
  59   } catch (error) {
  60     console.log("[BILLBOARD_PATCH]", error);
  61     return new NextResponse("Internal error", { status: 500 });
  62   }
  63 }
  64
  65 export async function DELETE(
  66   req: Request,
  67   { params: { billboardId: string } }
  68 ) {
  69   try {
  70     const { userId } = await auth();
  71     const body = await req.json();
  72
  73     const { label, imageUrl } = body;
  74
  75     if (!userId) {
  76       return new NextResponse("Unauthenticated", { status: 401 });
  77     }
  78
  79     if (!label) {
  80       return new NextResponse("Label menginput nama", { status: 400 });
  81     }
  82
  83     if (!imageUrl) {
  84       return new NextResponse("menginput Gambar Url", { status: 400 });
  85     }
  86
  87     const billboard = await db.billboard.create({
  88       data: {
  89         label,
  90         imageUrl,
  91         storeId,
  92       },
  93     });
  94
  95     return new NextResponse(JSON.stringify(billboard), { status: 201 });
  96   } catch (error) {
  97     console.log("[BILLBOARD_DELETE]", error);
  98     return new NextResponse("Internal error", { status: 500 });
  99   }
  100 }
  101
  102 export async function GET(
  103   req: Request,
  104   { params: { billboardId: string } }
  105 ) {
  106   try {
  107     if (!params.billboardId) {
  108       return new NextResponse("Papan iklan id dibutuhkan", { status: 400 });
  109     }
  110
  111     const billboard = await db.billboard.findUnique({
  112       where: {
  113         id: params.billboardId,
  114       },
  115     });
  116
  117     return NextResponse.json(billboard);
  118   } catch (error) {
  119     console.log("[BILLBOARD_GET]", error);
  120     return new NextResponse("Internal error", { status: 500 });
  121   }
  122 }
  123
  124 export async function PATCH(
  125   req: Request,
  126   { params: { storeId: string, billboardId: string } }
  127 ) {
  128   try {
  129     const { userId } = await auth();
  130     const body = await req.json();
  131
  132     const { label, imageUrl } = body;
  133
  134     if (!userId) {
  135       return new NextResponse("Unauthenticated", { status: 401 });
  136     }
  137
  138     if (!label) {
  139       return new NextResponse("Label menginput nama", { status: 400 });
  140     }
  141
  142     if (!imageUrl) {
  143       return new NextResponse("menginput Gambar Url", { status: 400 });
  144     }
  145
  146     const billboard = await db.billboard.create({
  147       data: {
  148         label,
  149         imageUrl,
  150         storeId,
  151       },
  152     });
  153
  154     return new NextResponse(JSON.stringify(billboard), { status: 201 });
  155   } catch (error) {
  156     console.log("[BILLBOARD_PATCH]", error);
  157     return new NextResponse("Internal error", { status: 500 });
  158   }
  159 }
  160
  161 export async function DELETE(
  162   req: Request,
  163   { params: { billboardId: string } }
  164 ) {
  165   try {
  166     const { userId } = await auth();
  167     const body = await req.json();
  168
  169     const { label, imageUrl } = body;
  170
  171     if (!userId) {
  172       return new NextResponse("Unauthenticated", { status: 401 });
  173     }
  174
  175     if (!label) {
  176       return new NextResponse("Label menginput nama", { status: 400 });
  177     }
  178
  179     if (!imageUrl) {
  180       return new NextResponse("menginput Gambar Url", { status: 400 });
  181     }
  182
  183     const billboard = await db.billboard.create({
  184       data: {
  185         label,
  186         imageUrl,
  187         storeId,
  188       },
  189     });
  190
  191     return new NextResponse(JSON.stringify(billboard), { status: 201 });
  192   } catch (error) {
  193     console.log("[BILLBOARD_DELETE]", error);
  194     return new NextResponse("Internal error", { status: 500 });
  195   }
  196 }
  197
  198 export async function GET(
  199   req: Request,
  200   { params: { billboardId: string } }
  201 ) {
  202   try {
  203     if (!params.billboardId) {
  204       return new NextResponse("Papan iklan id dibutuhkan", { status: 400 });
  205     }
  206
  207     const billboard = await db.billboard.findUnique({
  208       where: {
  209         id: params.billboardId,
  210       },
  211     });
  212
  213     return NextResponse.json(billboard);
  214   } catch (error) {
  215     console.log("[BILLBOARD_GET]", error);
  216     return new NextResponse("Internal error", { status: 500 });
  217   }
  218 }
  219
  220 export async function PATCH(
  221   req: Request,
  222   { params: { storeId: string, billboardId: string } }
  223 ) {
  224   try {
  225     const { userId } = await auth();
  226     const body = await req.json();
  227
  228     const { label, imageUrl } = body;
  229
  230     if (!userId) {
  231       return new NextResponse("Unauthenticated", { status: 401 });
  232     }
  233
  234     if (!label) {
  235       return new NextResponse("Label menginput nama", { status: 400 });
  236     }
  237
  238     if (!imageUrl) {
  239       return new NextResponse("menginput Gambar Url", { status: 400 });
  240     }
  241
  242     const billboard = await db.billboard.create({
  243       data: {
  244         label,
  245         imageUrl,
  246         storeId,
  247       },
  248     });
  249
  250     return new NextResponse(JSON.stringify(billboard), { status: 201 });
  251   } catch (error) {
  252     console.log("[BILLBOARD_PATCH]", error);
  253     return new NextResponse("Internal error", { status: 500 });
  254   }
  255 }
  256
  257 export async function DELETE(
  258   req: Request,
  259   { params: { billboardId: string } }
  260 ) {
  261   try {
  262     const { userId } = await auth();
  263     const body = await req.json();
  264
  265     const { label, imageUrl } = body;
  266
  267     if (!userId) {
  268       return new NextResponse("Unauthenticated", { status: 401 });
  269     }
  270
  271     if (!label) {
  272       return new NextResponse("Label menginput nama", { status: 400 });
  273     }
  274
  275     if (!imageUrl) {
  276       return new NextResponse("menginput Gambar Url", { status: 400 });
  277     }
  278
  279     const billboard = await db.billboard.create({
  280       data: {
  281         label,
  282         imageUrl,
  283         storeId,
  284       },
  285     });
  286
  287     return new NextResponse(JSON.stringify(billboard), { status: 201 });
  288   } catch (error) {
  289     console.log("[BILLBOARD_DELETE]", error);
  290     return new NextResponse("Internal error", { status: 500 });
  291   }
  292 }
  293
  294 export async function GET(
  295   req: Request,
  296   { params: { billboardId: string } }
  297 ) {
  298   try {
  299     if (!params.billboardId) {
  300       return new NextResponse("Papan iklan id dibutuhkan", { status: 400 });
  301     }
  302
  303     const billboard = await db.billboard.findUnique({
  304       where: {
  305         id: params.billboardId,
  306       },
  307     });
  308
  309     return NextResponse.json(billboard);
  310   } catch (error) {
  311     console.log("[BILLBOARD_GET]", error);
  312     return new NextResponse("Internal error", { status: 500 });
  313   }
  314 }
  315
  316 export async function PATCH(
  317   req: Request,
  318   { params: { storeId: string, billboardId: string } }
  319 ) {
  320   try {
  321     const { userId } = await auth();
  322     const body = await req.json();
  323
  324     const { label, imageUrl } = body;
  325
  326     if (!userId) {
  327       return new NextResponse("Unauthenticated", { status: 401 });
  328     }
  329
  330     if (!label) {
  331       return new NextResponse("Label menginput nama", { status: 400 });
  332     }
  333
  334     if (!imageUrl) {
  335       return new NextResponse("menginput Gambar Url", { status: 400 });
  336     }
  337
  338     const billboard = await db.billboard.create({
  339       data: {
  340         label,
  341         imageUrl,
  342         storeId,
  343       },
  344     });
  345
  346     return new NextResponse(JSON.stringify(billboard), { status: 201 });
  347   } catch (error) {
  348     console.log("[BILLBOARD_PATCH]", error);
  349     return new NextResponse("Internal error", { status: 500 });
  350   }
  351 }
  352
  353 export async function DELETE(
  354   req: Request,
  355   { params: { billboardId: string } }
  356 ) {
  357   try {
  358     const { userId } = await auth();
  359     const body = await req.json();
  360
  361     const { label, imageUrl } = body;
  362
  363     if (!userId) {
  364       return new NextResponse("Unauthenticated", { status: 401 });
  365     }
  366
  367     if (!label) {
  368       return new NextResponse("Label menginput nama", { status: 400 });
  369     }
  370
  371     if (!imageUrl) {
  372       return new NextResponse("menginput Gambar Url", { status: 400 });
  373     }
  374
  375     const billboard = await db.billboard.create({
  376       data: {
  377         label,
  378         imageUrl,
  379         storeId,
  380       },
  381     });
  382
  383     return new NextResponse(JSON.stringify(billboard), { status: 201 });
  384   } catch (error) {
  385     console.log("[BILLBOARD_DELETE]", error);
  386     return new NextResponse("Internal error", { status: 500 });
  387   }
  388 }
  389
  390 export async function GET(
  391   req: Request,
  392   { params: { billboardId: string } }
  393 ) {
  394   try {
  395     if (!params.billboardId) {
  396       return new NextResponse("Papan iklan id dibutuhkan", { status: 400 });
  397     }
  398
  399     const billboard = await db.billboard.findUnique({
  400       where: {
  401         id: params.billboardId,
  402       },
  403     });
  404
  405     return NextResponse.json(billboard);
  406   } catch (error) {
  407     console.log("[BILLBOARD_GET]", error);
  408     return new NextResponse("Internal error", { status: 500 });
  409   }
  410 }
  411
  412 export async function PATCH(
  413   req: Request,
  414   { params: { storeId: string, billboardId: string } }
  415 ) {
  416   try {
  417     const { userId } = await auth();
  418     const body = await req.json();
  419
  420     const { label, imageUrl } = body;
  421
  422     if (!userId) {
  423       return new NextResponse("Unauthenticated", { status: 401 });
  424     }
  425
  426     if (!label) {
  427       return new NextResponse("Label menginput nama", { status: 400 });
  428     }
  429
  430     if (!imageUrl) {
  431       return new NextResponse("menginput Gambar Url", { status: 400 });
  432     }
  433
  434     const billboard = await db.billboard.create({
  435       data: {
  436         label,
  437         imageUrl,
  438         storeId,
  439       },
  440     });
  441
  442     return new NextResponse(JSON.stringify(billboard), { status: 201 });
  443   } catch (error) {
  444     console.log("[BILLBOARD_PATCH]", error);
  445     return new NextResponse("Internal error", { status: 500 });
  446   }
  447 }
  448
  449 export async function DELETE(
  450   req: Request,
  451   { params: { billboardId: string } }
  452 ) {
  453   try {
  454     const { userId } = await auth();
  455     const body = await req.json();
  456
  457     const { label, imageUrl } = body;
  458
  459     if (!userId) {
  460       return new NextResponse("Unauthenticated", { status: 401 });
  461     }
  462
  463     if (!label) {
  464       return new NextResponse("Label menginput nama", { status: 400 });
  465     }
  466
  467     if (!imageUrl) {
  468       return new NextResponse("menginput Gambar Url", { status: 400 });
  469     }
  470
  471     const billboard = await db.billboard.create({
  472       data: {
  473         label,
  474         imageUrl,
  475         storeId,
  476       },
  477     });
  478
  479     return new NextResponse(JSON.stringify(billboard), { status: 201 });
  480   } catch (error) {
  481     console.log("[BILLBOARD_DELETE]", error);
  482     return new NextResponse("Internal error", { status: 500 });
  483   }
  484 }
  485
  486 export async function GET(
  487   req: Request,
  488   { params: { billboardId: string } }
  489 ) {
  490   try {
  491     if (!params.billboardId) {
  492       return new NextResponse("Papan iklan id dibutuhkan", { status: 400 });
  493     }
  494
  495     const billboard = await db.billboard.findUnique({
  496       where: {
  497         id: params.billboardId,
  498       },
  499     });
  500
  501     return NextResponse.json(billboard);
  502   } catch (error) {
  503     console.log("[BILLBOARD_GET]", error);
  504     return new NextResponse("Internal error", { status: 500 });
  505   }
  506 }
  507
  508 export async function PATCH(
  509   req: Request,
  510   { params: { storeId: string, billboardId: string } }
  511 ) {
  512   try {
  513     const { userId } = await auth();
  514     const body = await req.json();
  515
  516     const { label, imageUrl } = body;
  517
  518     if (!userId) {
  519       return new NextResponse("Unauthenticated", { status: 401 });
  520     }
  521
  522     if (!label) {
  523       return new NextResponse("Label menginput nama", { status: 400 });
  524     }
  525
  526     if (!imageUrl) {
  527       return new NextResponse("menginput Gambar Url", { status: 400 });
  528     }
  529
  530     const billboard = await db.billboard.create({
  531       data: {
  532         label,
  533         imageUrl,
  534         storeId,
  535       },
  536     });
  537
  538     return new NextResponse(JSON.stringify(billboard), { status: 201 });
  539   } catch (error) {
  540     console.log("[BILLBOARD_PATCH]", error);
  541     return new NextResponse("Internal error", { status: 500 });
  542   }
  543 }
  544
  545 export async function DELETE(
  546   req: Request,
  547   { params: { billboardId: string } }
  548 ) {
  549   try {
  550     const { userId } = await auth();
  551     const body = await req.json();
  552
  553     const { label, imageUrl } = body;
  554
  555     if (!userId) {
  556       return new NextResponse("Unauthenticated", { status: 401 });
  557     }
  558
  559     if (!label) {
  560       return new NextResponse("Label menginput nama", { status: 400 });
  561     }
  562
  563     if (!imageUrl) {
  564       return new NextResponse("menginput Gambar Url", { status: 400 });
  565     }
  566
  567     const billboard = await db.billboard.create({
  568       data: {
  569         label,
  570         imageUrl,
  571         storeId,
  572       },
  573     });
  574
  575     return new NextResponse(JSON.stringify(billboard), { status: 201 });
  576   } catch (error) {
  577     console.log("[BILLBOARD_DELETE]", error);
  578     return new NextResponse("Internal error", { status: 500 });
  579   }
  580 }
  581
  582 export async function GET(
  583   req: Request,
  584   { params: { billboardId: string } }
  585 ) {
  586   try {
  587     if (!params.billboardId) {
  588       return new NextResponse("Papan iklan id dibutuhkan", { status: 400 });
  589     }
  590
  591     const billboard = await db.billboard.findUnique({
  592       where: {
  593         id: params.billboardId,
  594       },
  595     });
  596
  597     return NextResponse.json(billboard);
  598   } catch (error) {
  599     console.log("[BILLBOARD_GET]", error);
  600     return new NextResponse("Internal error", { status: 500 });
  601   }
  602 }
  603
  604 export async function PATCH(
  605   req: Request,
  606   { params: { storeId: string, billboardId: string } }
  607 ) {
  608   try {
  609     const { userId } = await auth();
  610     const body = await req.json();
  611
  612     const { label, imageUrl } = body;
  613
  614     if (!userId) {
  615       return new NextResponse("Unauthenticated", { status: 401 });
  616     }
  617
  618     if (!label) {
  619       return new NextResponse("Label menginput nama", { status: 400 });
  620     }
  621
  622     if (!imageUrl) {
  623       return new NextResponse("menginput Gambar Url", { status: 400 });
  624     }
  625
  626     const billboard = await db.billboard.create({
  627       data: {
  628         label,
  629         imageUrl,
  630         storeId,
  631       },
  632     });
  633
  634     return new NextResponse(JSON.stringify(billboard), { status: 201 });
  635   } catch (error) {
  636     console.log("[BILLBOARD_PATCH]", error);
  637     return new NextResponse("Internal error", { status: 500 });
  638   }
  639 }
  640
  641 export async function DELETE(
  642   req: Request,
  643   { params: { billboardId: string } }
  644 ) {
  645   try {
  646     const { userId } = await auth();
  647     const body = await req.json();
  648
  649     const { label, imageUrl } = body;
  650
  651     if (!userId) {
  652       return new NextResponse("Unauthenticated", { status: 401 });
  653     }
  654
  655     if (!label) {
  656       return new NextResponse("Label menginput nama", { status: 400 });
  657     }
  658
  659     if (!imageUrl) {
  660       return new NextResponse("menginput Gambar Url", { status: 400 });
  661     }
  662
  663     const billboard = await db.billboard.create({
  664       data: {
  665         label,
  666         imageUrl,
  667         storeId,
  668       },
  669     });
  670
  671     return new NextResponse(JSON.stringify(billboard), { status: 201 });
  672   } catch (error) {
  673     console.log("[BILLBOARD_DELETE]", error);
  674     return new NextResponse("Internal error", { status: 500 });
  675   }
  676 }
  677
  678 export async function GET(
  679   req: Request,
  680   { params: { billboardId: string } }
  681 ) {
  682   try {
  683     if (!params.billboardId) {
  684       return new NextResponse("Papan iklan id dibutuhkan", { status: 400 });
  685     }
  686
  687     const billboard = await db.billboard.findUnique({
  688       where: {
  689         id: params.billboardId,
  690       },
  691     });
  692
  693     return NextResponse.json(billboard);
  694   } catch (error) {
  695     console.log("[BILLBOARD_GET]", error);
  696     return new NextResponse("Internal error", { status: 500 });
  697   }
  698 }
  699
  700 export async function PATCH(
  701   req: Request,
  702   { params: { storeId: string, billboardId: string } }
  703 ) {
  704   try {
  705     const { userId } = await auth();
  706     const body = await req.json();
  707
  708     const { label, imageUrl } = body;
  709
  710     if (!userId) {
  711       return new NextResponse("Unauthenticated", { status: 401 });
  712     }
  713
  714     if (!label) {
  715       return new NextResponse("Label menginput nama", { status: 400 });
  716     }
  717
  718     if (!imageUrl) {
  719       return new NextResponse("menginput Gambar Url", { status: 400 });
  720     }
  721
  722     const billboard = await db.billboard.create({
  723       data: {
  724         label,
  725         imageUrl,
  726         storeId,
  727       },
  728     });
  729
  730     return new NextResponse(JSON.stringify(billboard), { status: 201 });
  731   } catch (error) {
  732     console.log("[BILLBOARD_PATCH]", error);
  733     return new NextResponse("Internal error", { status: 500 });
  734   }
  735 }
  736
  737 export async function DELETE(
  738   req: Request,
  739   { params: { billboardId: string } }
  740 ) {
  741   try {
  742     const { userId } = await auth();
  743     const body = await req.json();
  744
  745     const { label, imageUrl } = body;
  746
  747     if (!userId) {
  748       return new NextResponse("Unauthenticated", { status: 401 });
  749     }
  750
  751     if (!label) {
  752       return new NextResponse("Label menginput nama", { status: 400 });
  753     }
  754
  755     if (!imageUrl) {
  756       return new NextResponse("menginput Gambar Url", { status: 400 });
  757     }
  758
  759     const billboard = await db.billboard.create({
  760       data: {
  761         label,
  762         imageUrl,
  763         storeId,
  764       },
  765     });
  766
  767     return new NextResponse(JSON.stringify(billboard), { status: 201 });
  768   } catch (error) {
  769     console.log("[BILLBOARD_DELETE]", error);
  770     return new NextResponse("Internal error", { status: 500 });
  771   }
  772 }
  773
  774 export async function GET(
  775   req: Request,
  776   { params: { billboardId: string } }
  777 ) {
  778   try {
  779     if (!params.billboardId) {
  780       return new NextResponse("Papan iklan id dibutuhkan", { status: 400 });
  781     }
  782
  783     const billboard = await db.billboard.findUnique({
  784       where: {
  785         id: params.billboardId,
  786       },
  787     });
  788
  789     return NextResponse.json(billboard);
  790   } catch (error) {
  791     console.log("[BILLBOARD_GET]", error);
  792     return new NextResponse("Internal error", { status: 500 });
  793   }
  794 }
  795
  796 export async function PATCH(
  797   req: Request,
  798   { params: { storeId: string, billboardId: string } }
  799 ) {
  800   try {
  801     const { userId } = await auth();
  802     const body = await req.json();
  803
  804     const { label, imageUrl } = body;
  805
  806     if (!userId) {
  807       return new NextResponse("Unauthenticated", { status: 401 });
  808     }
  809
  810     if (!label) {
  811       return new NextResponse("Label menginput nama", { status: 400 });
  812     }
  813
  814     if (!imageUrl) {
  815       return new NextResponse("menginput Gambar Url", { status: 400 });
  816     }
  817
  818     const billboard = await db.billboard.create({
  819       data: {
  820         label,
  821         imageUrl,
  822         storeId,
  823       },
  824     });
  825
  826     return new NextResponse(JSON.stringify(billboard), { status: 201 });
  827   } catch (error) {
  828     console.log("[BILLBOARD_PATCH]", error);
  829     return new NextResponse("Internal error", { status: 500 });
  830   }
  831 }
  832
  833 export async function DELETE(
  834   req: Request,
  835   { params: { billboardId: string } }
  836 ) {
  837   try {
  838     const { userId } = await auth();
  839     const body = await req.json();
  840
  841     const { label, imageUrl } = body;
  842
  843     if (!userId) {
  844       return new NextResponse("Unauthenticated", { status: 401 });
  845     }
  846
  847     if (!label) {
  848       return new NextResponse("Label menginput nama", { status: 400 });
  849     }
  850
  851     if (!imageUrl) {
  852       return new NextResponse("menginput Gambar Url", { status: 400 });
  853     }
  854
  855     const billboard = await db.billboard.create({
  856       data: {
  857         label,
  858         imageUrl,
  859         storeId,
  860       },
  861     });
  862
  863     return new NextResponse(JSON.stringify(billboard), { status: 201 });
  864   } catch (error) {
  865     console.log("[BILLBOARD_DELETE]", error);
  866     return new NextResponse("Internal error", { status: 500 });
  867   }
  868 }
  869
  870 export async function GET(
  871   req: Request,
  872   { params: { billboardId: string } }
  873 ) {
  874   try {
  875     if (!params.billboardId) {
  876       return new NextResponse("Papan iklan id dibutuhkan", { status: 400 });
  877     }
  878
  879     const billboard = await db.billboard.findUnique({
  880       where: {
  881         id: params.billboardId,
  882       },
  883     });
  884
  885     return NextResponse.json(billboard);
  886   } catch (error) {
  887     console.log("[BILLBOARD_GET]", error);
  888     return new NextResponse("Internal error", { status: 500 });
  889   }
  890 }
  891
  892 export async function PATCH(
  893   req: Request,
  894   { params: { storeId: string, billboardId: string } }
  895 ) {
  896   try {
  897     const { userId } = await auth();
  898     const body = await req.json();
  899
  900     const { label, imageUrl } = body;
  901
  902     if (!userId) {
  903       return new NextResponse("Unauthenticated", { status: 401 });
  904     }
  905
  906     if (!label) {
  907       return new NextResponse("Label menginput nama", { status: 400 });
  908     }
  909
  910     if (!imageUrl) {
  911       return new NextResponse("menginput Gambar Url", { status: 400 });
  912     }
  913
  914     const billboard = await db.billboard.create({
  915       data: {
  916         label,
  917         imageUrl,
  918         storeId,
  919       },
  920     });
  921
  922     return new NextResponse(JSON.stringify(billboard), { status: 201 });
  923   } catch (error) {
  924     console.log("[BILLBOARD_PATCH]", error);
  925     return new NextResponse("Internal error", { status: 500 });
  926   }
  927 }
  928
  929 export async function DELETE(
  930   req: Request,
  931   { params: { billboardId: string } }
  932 ) {
  933   try {
  934     const { userId } = await auth();
  935     const body = await req.json();
  936
  937     const { label, imageUrl } = body;
  938
  939     if (!userId) {
  940       return new NextResponse("Unauthenticated", { status: 401 });
  941     }
  942
  943     if (!label) {
  944       return new NextResponse("Label menginput nama", { status: 400 });
  945     }
  946
  947     if (!imageUrl) {
  948       return new NextResponse("menginput Gambar Url", { status: 400 });
  949     }
  950
  951     const billboard = await db.billboard.create({
  952       data: {
  953         label,
  954         imageUrl,
  955         storeId,
  956       },
  957     });
  958
  959     return new NextResponse(JSON.stringify(billboard), { status: 201 });
  960   } catch (error) {
  961     console.log("[BILLBOARD_DELETE]", error);
  962     return new NextResponse("Internal error", { status: 500 });
  963   }
  964 }
  965
  966 export async function GET(
  967   req: Request,
  968   { params: { billboardId: string } }
  969 ) {
  970   try {
  971     if (!params.billboardId) {
  972       return new NextResponse("Papan iklan id dibutuhkan", { status: 400 });
  973     }
  974
  975     const billboard = await db.billboard.findUnique({
  976       where: {
  977         id: params.billboardId,
  978       },
  979     });
  980
  981     return NextResponse.json(billboard);
  982   } catch (error) {
  983     console.log("[BILLBOARD_GET]", error);
  984     return new NextResponse("Internal error", { status: 500 });
  985   }
  986 }
  987
  988 export async function PATCH(
  989   req: Request,
  990   { params: { storeId: string, billboardId: string } }
  991 ) {
  992   try {
  993     const { userId } = await auth();
  994     const body = await req.json();
  995
  996     const { label, imageUrl } = body;
  997
  998     if (!userId) {
  999       return new NextResponse("Unauthenticated", { status: 401 });
  1000     }
  1001
  1002     if (!label) {
  1003       return new NextResponse("Label menginput nama", { status: 400 });
  1004     }
  1005
  1006     if (!imageUrl) {
  1007       return new NextResponse("menginput Gambar Url", { status: 400 });
  1008     }
  1009
  1010     const billboard = await db.billboard.create({
  1011       data: {
  1012         label,
  1013         imageUrl,
  1014         storeId,
  1015       },
  1016     });
  1017
  1018     return new NextResponse(JSON.stringify(billboard), { status: 201 });
  1019   } catch (error) {
  1020     console.log("[BILLBOARD_PATCH]", error);
  1021     return new NextResponse("Internal error", { status: 500 });
  1022   }
  1023 }
  1024
  1025 export async function DELETE(
  1026   req: Request,
  1027   { params: { billboardId: string } }
  1028 ) {
  1029   try {
  1030     const { userId } = await auth();
  1031     const body = await req.json();
  1032
  1033     const { label, imageUrl } = body;
  1034
  1035     if (!userId) {
  1036       return new NextResponse("Unauthenticated", { status: 401 });
  1037     }
  1038
  1039     if (!label) {
  1040       return new NextResponse
```

```

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 :

Toko01 Overview Billboards categories settings

```

v app
  > (auth)
  v (dashboard) \ [storeId]
    v (routes)
      > billboards
      > categories
      > settings
    ⚙ page.tsx
    ⚙ layout.tsx

```

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]
├── compone...
│   ├── cell-action.tsx
│   ├── client.tsx M
│   ├── columns.tsx
│   └── page.tsx
└── category-form.tsx U
  
```

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.