

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.storeId);
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

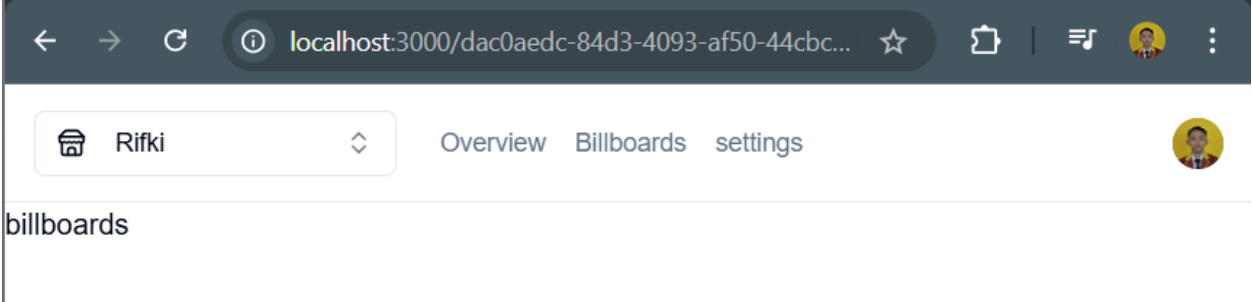
```
PS D:\Universitas Teknokrat Indonesia\Semester 5\WEB 2\ecommerce-visionvortex> npx prisma db push
```

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;
```

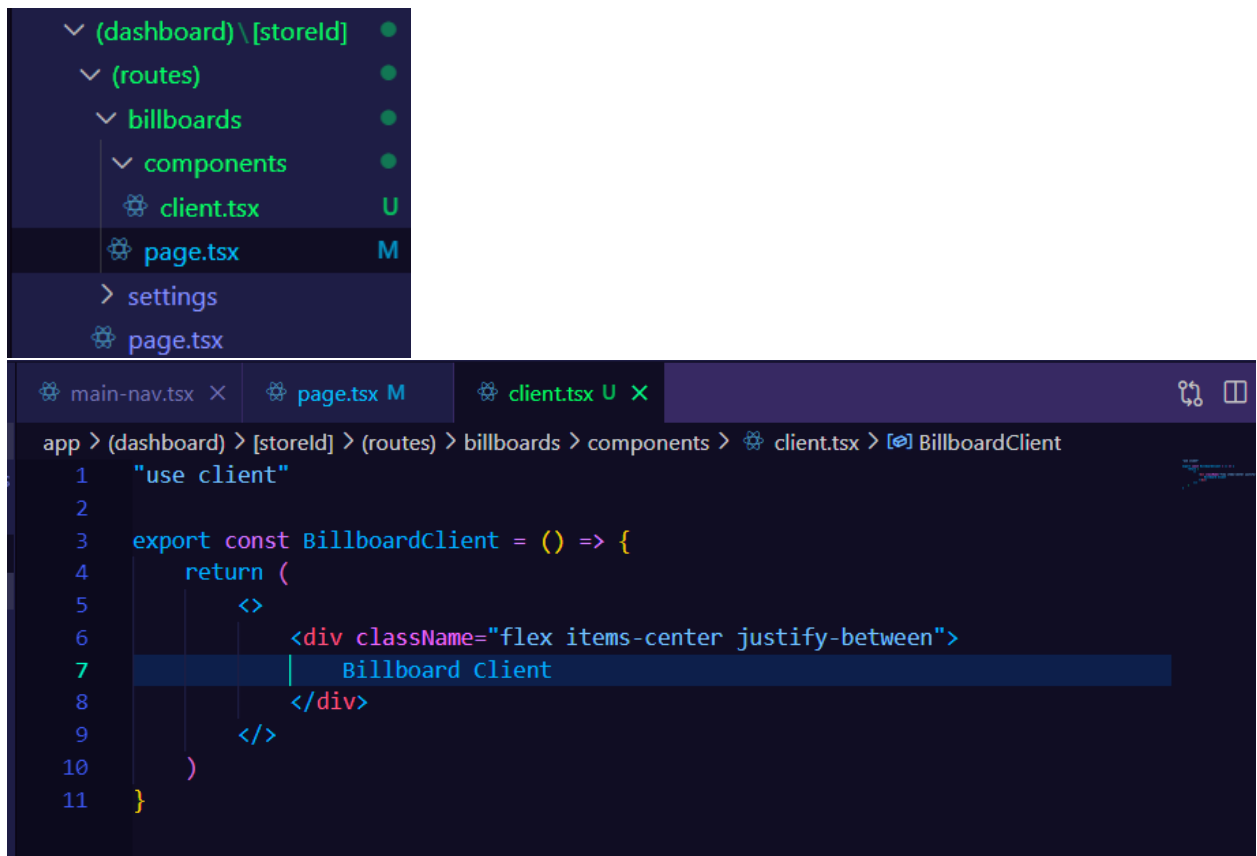
Membuat Lembar papan iklan/Billboards.

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
```

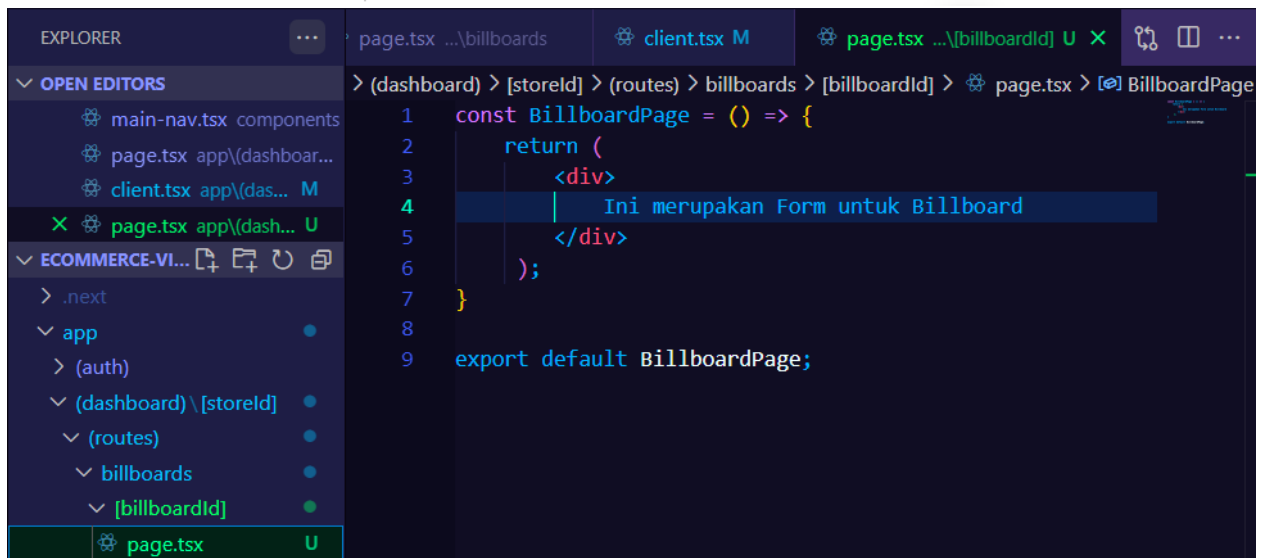
Membuat onClick pada button tambah billboards

hasil :



404

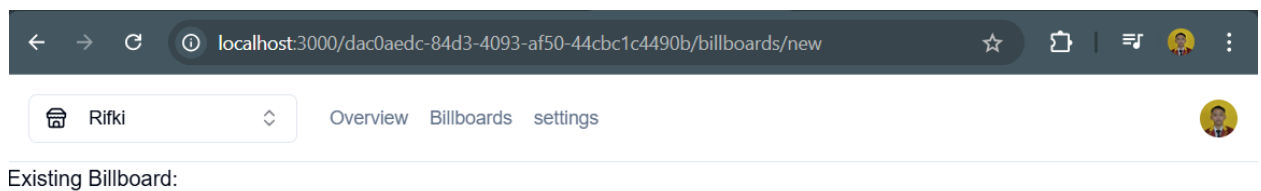
This page could not be found.



```
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
main-nav.tsx components
page.tsx app\(dashboard)\st...
client.tsx app\(dashboard)\st...
page.tsx app\(dashbo... M
billboard-form.tsx ap... U
ECCOMMERCE-VISIONVORTEX
> next
> app
  > (auth)
  > (dashboard) [storeId]
    > (routes)
      > billboards
        > [billboardId]
          > components
            @ billboard-form.tsx U
            @ page.tsx M
            > components
              111

```

```

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 { billboardId } = params;
  72
  73     if (!userId) {
  74       return new NextResponse("Unauthenticated", { status: 401 });
  75     }
  76
  77     if (!billboardId) {
  78       return new NextResponse("Id billboard dibutuhkan", { status: 400 });
  79     }
  80
  81     const billboard = await db.billboard.delete({
  82       where: {
  83         id: billboardId,
  84       },
  85     });
  86
  87     return new NextResponse("Billboard berhasil dihapus", { status: 200 });
  88   } catch (error) {
  89     console.log("[BILLBOARD_DELETE]", error);
  90     return new NextResponse("Internal error", { status: 500 });
  91   }
  92 }
  93
  94 // Membuat entri baru pada tabel 'store' dalam database.
  95 const store = await db.store.create({
  96   data: {
  97     name: "Store Baru",
  98     address: "Jl. Merdeka No. 123",
  99     city: "Jakarta",
 100     state: "DKI Jakarta",
 101     country: "Indonesia",
 102     zip: "10110",
 103   },
104 });
105
106 return new NextResponse(JSON.stringify(store), { status: 201 });
107 } catch (error) {
108   console.log("[STORE_CREATE]", error);
109   return new NextResponse("Internal error", { status: 500 });
110 }
111 }
112
113 export async function GET(
114   req: Request,
115   { params: { storeId: string } }
116 ) {
117   try {
118     const { userId } = await auth();
119     const { storeId } = params;
120
121     if (!userId) {
122       return new NextResponse("Unauthenticated", { status: 401 });
123     }
124
125     if (!storeId) {
126       return new NextResponse("Id store dibutuhkan", { status: 400 });
127     }
128
129     const store = await db.store.findUnique({
130       where: {
131         id: storeId,
132       },
133     });
134
135     return new NextResponse(JSON.stringify(store), { status: 200 });
136   } catch (error) {
137     console.log("[STORE_GET]", error);
138     return new NextResponse("Internal error", { status: 500 });
139   }
140 }
141
142 export async function PATCH(
143   req: Request,
144   { params: { storeId: string } }
145 ) {
146   try {
147     const { userId } = await auth();
148     const { storeId } = params;
149     const body = await req.json();
150
151     const { name, address, city, state, country, zip } = body;
152
153     if (!userId) {
154       return new NextResponse("Unauthenticated", { status: 401 });
155     }
156
157     if (!name) {
158       return new NextResponse("Nama toko harus diisi", { status: 400 });
159     }
160
161     if (!address) {
162       return new NextResponse("Alamat toko harus diisi", { status: 400 });
163     }
164
165     if (!city) {
166       return new NextResponse("Kota harus diisi", { status: 400 });
167     }
168
169     if (!state) {
170       return new NextResponse("Provinsi harus diisi", { status: 400 });
171     }
172
173     if (!country) {
174       return new NextResponse("Negara harus diisi", { status: 400 });
175     }
176
177     if (!zip) {
178       return new NextResponse("Kode pos harus diisi", { status: 400 });
179     }
180
181     const store = await db.store.update({
182       where: {
183         id: storeId,
184       },
185       data: {
186         name,
187         address,
188         city,
189         state,
190         country,
191         zip,
192       },
193     });
194
195     return new NextResponse(JSON.stringify(store), { status: 200 });
196   } catch (error) {
197     console.log("[STORE_PATCH]", error);
198     return new NextResponse("Internal error", { status: 500 });
199   }
200 }
201
202 export async function DELETE(
203   req: Request,
204   { params: { storeId: string } }
205 ) {
206   try {
207     const { userId } = await auth();
208     const { storeId } = params;
209
210     if (!userId) {
211       return new NextResponse("Unauthenticated", { status: 401 });
212     }
213
214     if (!storeId) {
215       return new NextResponse("Id store dibutuhkan", { status: 400 });
216     }
217
218     const store = await db.store.delete({
219       where: {
220         id: storeId,
221       },
222     });
223
224     return new NextResponse("Store berhasil dihapus", { status: 200 });
225   } catch (error) {
226     console.log("[STORE_DELETE]", error);
227     return new NextResponse("Internal error", { status: 500 });
228   }
229 }
230
231 // Membuat entri baru pada tabel 'billboard' dalam database.
232 const billboard = await db.billboard.create({
233   data: {
234     label: "Billboard Baru",
235     imageUrl: "https://example.com/image.jpg",
236     storeId,
237   },
238 });
239
240 return new NextResponse(JSON.stringify(billboard), { status: 201 });
241 } catch (error) {
242   console.log("[BILLBOARD_CREATE]", error);
243   return new NextResponse("Internal error", { status: 500 });
244 }
245 }
246
247 export async function GET(
248   req: Request,
249   { params: { billboardId: string } }
250 ) {
251   try {
252     const { userId } = await auth();
253     const { billboardId } = params;
254
255     if (!userId) {
256       return new NextResponse("Unauthenticated", { status: 401 });
257     }
258
259     if (!billboardId) {
260       return new NextResponse("Id billboard dibutuhkan", { status: 400 });
261     }
262
263     const billboard = await db.billboard.findUnique({
264       where: {
265         id: billboardId,
266       },
267     });
268
269     return new NextResponse(JSON.stringify(billboard), { status: 200 });
270   } catch (error) {
271     console.log("[BILLBOARD_GET]", error);
272     return new NextResponse("Internal error", { status: 500 });
273   }
274 }
275
276 export async function PATCH(
277   req: Request,
278   { params: { billboardId: string } }
279 ) {
280   try {
281     const { userId } = await auth();
282     const { billboardId } = params;
283     const body = await req.json();
284
285     const { label, imageUrl } = body;
286
287     if (!userId) {
288       return new NextResponse("Unauthenticated", { status: 401 });
289     }
290
291     if (!label) {
292       return new NextResponse("Label menginput nama", { status: 400 });
293     }
294
295     if (!imageUrl) {
296       return new NextResponse("menginput Gambar Url", { status: 400 });
297     }
298
299     const billboard = await db.billboard.update({
300       where: {
301         id: billboardId,
302       },
303       data: {
304         label,
305         imageUrl,
306       },
307     });
308
309     return new NextResponse(JSON.stringify(billboard), { status: 200 });
310   } catch (error) {
311     console.log("[BILLBOARD_PATCH]", error);
312     return new NextResponse("Internal error", { status: 500 });
313   }
314 }
315
316 export async function DELETE(
317   req: Request,
318   { params: { billboardId: string } }
319 ) {
320   try {
321     const { userId } = await auth();
322     const { billboardId } = params;
323
324     if (!userId) {
325       return new NextResponse("Unauthenticated", { status: 401 });
326     }
327
328     if (!billboardId) {
329       return new NextResponse("Id billboard dibutuhkan", { status: 400 });
330     }
331
332     const billboard = await db.billboard.delete({
333       where: {
334         id: billboardId,
335       },
336     });
337
338     return new NextResponse("Billboard berhasil dihapus", { status: 200 });
339   } catch (error) {
340     console.log("[BILLBOARD_DELETE]", error);
341     return new NextResponse("Internal error", { status: 500 });
342   }
343 }
344
345 // Membuat entri baru pada tabel 'store' dalam database.
346 const store = await db.store.create({
347   data: {
348     name: "Store Baru",
349     address: "Jl. Merdeka No. 123",
350     city: "Jakarta",
351     state: "DKI Jakarta",
352     country: "Indonesia",
353     zip: "10110",
354   },
355 });
356
357 return new NextResponse(JSON.stringify(store), { status: 201 });
358 } catch (error) {
359   console.log("[STORE_CREATE]", error);
360   return new NextResponse("Internal error", { status: 500 });
361 }
362 }
363
364 export async function GET(
365   req: Request,
366   { params: { storeId: string } }
367 ) {
368   try {
369     const { userId } = await auth();
370     const { storeId } = params;
371
372     if (!userId) {
373       return new NextResponse("Unauthenticated", { status: 401 });
374     }
375
376     if (!storeId) {
377       return new NextResponse("Id store dibutuhkan", { status: 400 });
378     }
379
380     const store = await db.store.findUnique({
381       where: {
382         id: storeId,
383       },
384     });
385
386     return new NextResponse(JSON.stringify(store), { status: 200 });
387   } catch (error) {
388     console.log("[STORE_GET]", error);
389     return new NextResponse("Internal error", { status: 500 });
390   }
391 }
392
393 export async function PATCH(
394   req: Request,
395   { params: { storeId: string } }
396 ) {
397   try {
398     const { userId } = await auth();
399     const { storeId } = params;
400     const body = await req.json();
401
402     const { name, address, city, state, country, zip } = body;
403
404     if (!userId) {
405       return new NextResponse("Unauthenticated", { status: 401 });
406     }
407
408     if (!name) {
409       return new NextResponse("Nama toko harus diisi", { status: 400 });
410     }
411
412     if (!address) {
413       return new NextResponse("Alamat toko harus diisi", { status: 400 });
414     }
415
416     if (!city) {
417       return new NextResponse("Kota harus diisi", { status: 400 });
418     }
419
420     if (!state) {
421       return new NextResponse("Provinsi harus diisi", { status: 400 });
422     }
423
424     if (!country) {
425       return new NextResponse("Negara harus diisi", { status: 400 });
426     }
427
428     if (!zip) {
429       return new NextResponse("Kode pos harus diisi", { status: 400 });
430     }
431
432     const store = await db.store.update({
433       where: {
434         id: storeId,
435       },
436       data: {
437         name,
438         address,
439         city,
440         state,
441         country,
442         zip,
443       },
444     });
445
446     return new NextResponse(JSON.stringify(store), { status: 200 });
447   } catch (error) {
448     console.log("[STORE_PATCH]", error);
449     return new NextResponse("Internal error", { status: 500 });
450   }
451 }
452
453 export async function DELETE(
454   req: Request,
455   { params: { storeId: string } }
456 ) {
457   try {
458     const { userId } = await auth();
459     const { storeId } = params;
460
461     if (!userId) {
462       return new NextResponse("Unauthenticated", { status: 401 });
463     }
464
465     if (!storeId) {
466       return new NextResponse("Id store dibutuhkan", { status: 400 });
467     }
468
469     const store = await db.store.delete({
470       where: {
471         id: storeId,
472       },
473     });
474
475     return new NextResponse("Store berhasil dihapus", { status: 200 });
476   } catch (error) {
477     console.log("[STORE_DELETE]", error);
478     return new NextResponse("Internal error", { status: 500 });
479   }
480 }
481
482 // Membuat entri baru pada tabel 'billboard' dalam database.
483 const billboard = await db.billboard.create({
484   data: {
485     label: "Billboard Baru",
486     imageUrl: "https://example.com/image.jpg",
487     storeId,
488   },
489 });
490
491 return new NextResponse(JSON.stringify(billboard), { status: 201 });
492 } catch (error) {
493   console.log("[BILLBOARD_CREATE]", error);
494   return new NextResponse("Internal error", { status: 500 });
495 }
496 }
497
498 export async function GET(
499   req: Request,
500   { params: { billboardId: string } }
501 ) {
502   try {
503     const { userId } = await auth();
504     const { billboardId } = params;
505
506     if (!userId) {
507       return new NextResponse("Unauthenticated", { status: 401 });
508     }
509
510     if (!billboardId) {
511       return new NextResponse("Id billboard dibutuhkan", { status: 400 });
512     }
513
514     const billboard = await db.billboard.findUnique({
515       where: {
516         id: billboardId,
517       },
518     });
519
520     return new NextResponse(JSON.stringify(billboard), { status: 200 });
521   } catch (error) {
522     console.log("[BILLBOARD_GET]", error);
523     return new NextResponse("Internal error", { status: 500 });
524   }
525 }
526
527 export async function PATCH(
528   req: Request,
529   { params: { billboardId: string } }
530 ) {
531   try {
532     const { userId } = await auth();
533     const { billboardId } = params;
534     const body = await req.json();
535
536     const { label, imageUrl } = body;
537
538     if (!userId) {
539       return new NextResponse("Unauthenticated", { status: 401 });
540     }
541
542     if (!label) {
543       return new NextResponse("Label menginput nama", { status: 400 });
544     }
545
546     if (!imageUrl) {
547       return new NextResponse("menginput Gambar Url", { status: 400 });
548     }
549
550     const billboard = await db.billboard.update({
551       where: {
552         id: billboardId,
553       },
554       data: {
555         label,
556         imageUrl,
557       },
558     });
559
560     return new NextResponse(JSON.stringify(billboard), { status: 200 });
561   } catch (error) {
562     console.log("[BILLBOARD_PATCH]", error);
563     return new NextResponse("Internal error", { status: 500 });
564   }
565 }
566
567 export async function DELETE(
568   req: Request,
569   { params: { billboardId: string } }
570 ) {
571   try {
572     const { userId } = await auth();
573     const { billboardId } = params;
574
575     if (!userId) {
576       return new NextResponse("Unauthenticated", { status: 401 });
577     }
578
579     if (!billboardId) {
580       return new NextResponse("Id billboard dibutuhkan", { status: 400 });
581     }
582
583     const billboard = await db.billboard.delete({
584       where: {
585         id: billboardId,
586       },
587     });
588
589     return new NextResponse("Billboard berhasil dihapus", { status: 200 });
590   } catch (error) {
591     console.log("[BILLBOARD_DELETE]", error);
592     return new NextResponse("Internal error", { status: 500 });
593   }
594 }
595
596 // Membuat entri baru pada tabel 'store' dalam database.
597 const store = await db.store.create({
598   data: {
599     name: "Store Baru",
600     address: "Jl. Merdeka No. 123",
601     city: "Jakarta",
602     state: "DKI Jakarta",
603     country: "Indonesia",
604     zip: "10110",
605   },
606 });
607
608 return new NextResponse(JSON.stringify(store), { status: 201 });
609 } catch (error) {
610   console.log("[STORE_CREATE]", error);
611   return new NextResponse("Internal error", { status: 500 });
612 }
613 }
614
615 export async function GET(
616   req: Request,
617   { params: { storeId: string } }
618 ) {
619   try {
620     const { userId } = await auth();
621     const { storeId } = params;
622
623     if (!userId) {
624       return new NextResponse("Unauthenticated", { status: 401 });
625     }
626
627     if (!storeId) {
628       return new NextResponse("Id store dibutuhkan", { status: 400 });
629     }
630
631     const store = await db.store.findUnique({
632       where: {
633         id: storeId,
634       },
635     });
636
637     return new NextResponse(JSON.stringify(store), { status: 200 });
638   } catch (error) {
639     console.log("[STORE_GET]", error);
640     return new NextResponse("Internal error", { status: 500 });
641   }
642 }
643
644 export async function PATCH(
645   req: Request,
646   { params: { storeId: string } }
647 ) {
648   try {
649     const { userId } = await auth();
650     const { storeId } = params;
651     const body = await req.json();
652
653     const { name, address, city, state, country, zip } = body;
654
655     if (!userId) {
656       return new NextResponse("Unauthenticated", { status: 401 });
657     }
658
659     if (!name) {
660       return new NextResponse("Nama toko harus diisi", { status: 400 });
661     }
662
663     if (!address) {
664       return new NextResponse("Alamat toko harus diisi", { status: 400 });
665     }
666
667     if (!city) {
668       return new NextResponse("Kota harus diisi", { status: 400 });
669     }
670
671     if (!state) {
672       return new NextResponse("Provinsi harus diisi", { status: 400 });
673     }
674
675     if (!country) {
676       return new NextResponse("Negara harus diisi", { status: 400 });
677     }
678
679     if (!zip) {
680       return new NextResponse("Kode pos harus diisi", { status: 400 });
681     }
682
683     const store = await db.store.update({
684       where: {
685         id: storeId,
686       },
687       data: {
688         name,
689         address,
690         city,
691         state,
692         country,
693         zip,
694       },
695     });
696
697     return new NextResponse(JSON.stringify(store), { status: 200 });
698   } catch (error) {
699     console.log("[STORE_PATCH]", error);
700     return new NextResponse("Internal error", { status: 500 });
701   }
702 }
703
704 export async function DELETE(
705   req: Request,
706   { params: { storeId: string } }
707 ) {
708   try {
709     const { userId } = await auth();
710     const { storeId } = params;
711
712     if (!userId) {
713       return new NextResponse("Unauthenticated", { status: 401 });
714     }
715
716     if (!storeId) {
717       return new NextResponse("Id store dibutuhkan", { status: 400 });
718     }
719
720     const store = await db.store.delete({
721       where: {
722         id: storeId,
723       },
724     });
725
726     return new NextResponse("Store berhasil dihapus", { status: 200 });
727   } catch (error) {
728     console.log("[STORE_DELETE]", error);
729     return new NextResponse("Internal error", { status: 500 });
730   }
731 }
732
733 // Membuat entri baru pada tabel 'billboard' dalam database.
734 const billboard = await db.billboard.create({
735   data: {
736     label: "Billboard Baru",
737     imageUrl: "https://example.com/image.jpg",
738     storeId,
739   },
740 });
741
742 return new NextResponse(JSON.stringify(billboard), { status: 201 });
743 } catch (error) {
744   console.log("[BILLBOARD_CREATE]", error);
745   return new NextResponse("Internal error", { status: 500 });
746 }
747 }
748
749 export async function GET(
750   req: Request,
751   { params: { billboardId: string } }
752 ) {
753   try {
754     const { userId } = await auth();
755     const { billboardId } = params;
756
757     if (!userId) {
758       return new NextResponse("Unauthenticated", { status: 401 });
759     }
760
761     if (!billboardId) {
762       return new NextResponse("Id billboard dibutuhkan", { status: 400 });
763     }
764
765     const billboard = await db.billboard.findUnique({
766       where: {
767         id: billboardId,
768       },
769     });
770
771     return new NextResponse(JSON.stringify(billboard), { status: 200 });
772   } catch (error) {
773     console.log("[BILLBOARD_GET]", error);
774     return new NextResponse("Internal error", { status: 500 });
775   }
776 }
777
778 export async function PATCH(
779   req: Request,
780   { params: { billboardId: string } }
781 ) {
782   try {
783     const { userId } = await auth();
784     const { billboardId } = params;
785     const body = await req.json();
786
787     const { label, imageUrl } = body;
788
789     if (!userId) {
790       return new NextResponse("Unauthenticated", { status: 401 });
791     }
792
793     if (!label) {
794       return new NextResponse("Label menginput nama", { status: 400 });
795     }
796
797     if (!imageUrl) {
798       return new NextResponse("menginput Gambar Url", { status: 400 });
799     }
800
801     const billboard = await db.billboard.update({
802       where: {
803         id: billboardId,
804       },
805       data: {
806         label,
807         imageUrl,
808       },
809     });
810
811     return new NextResponse(JSON.stringify(billboard), { status: 200 });
812   } catch (error) {
813     console.log("[BILLBOARD_PATCH]", error);
814     return new NextResponse("Internal error", { status: 500 });
815   }
816 }
817
818 export async function DELETE(
819   req: Request,
820   { params: { billboardId: string } }
821 ) {
822   try {
823     const { userId } = await auth();
824     const { billboardId } = params;
825
826     if (!userId) {
827       return new NextResponse("Unauthenticated", { status: 401 });
828     }
829
830     if (!billboardId) {
831       return new NextResponse("Id billboard dibutuhkan", { status: 400 });
832     }
833
834     const billboard = await db.billboard.delete({
835       where: {
836         id: billboardId,
837       },
838     });
839
840     return new NextResponse("Billboard berhasil dihapus", { status: 200 });
841   } catch (error) {
842     console.log("[BILLBOARD_DELETE]", error);
843     return new NextResponse("Internal error", { status: 500 });
844   }
845 }
846
847 // Membuat entri baru pada tabel 'store' dalam database.
848 const store = await db.store.create({
849   data: {
850     name: "Store Baru",
851     address: "Jl. Merdeka No. 123",
852     city: "Jakarta",
853     state: "DKI Jakarta",
854     country: "Indonesia",
855     zip: "10110",
856   },
857 });
858
859 return new NextResponse(JSON.stringify(store), { status: 201 });
860 } catch (error) {
861   console.log("[STORE_CREATE]", error);
862   return new NextResponse("Internal error", { status: 500 });
863 }
864 }
865
866 export async function GET(
867   req: Request,
868   { params: { storeId: string } }
869 ) {
870   try {
871     const { userId } = await auth();
872     const { storeId } = params;
873
874     if (!userId) {
875       return new NextResponse("Unauthenticated", { status: 401 });
876     }
877
878     if (!storeId) {
879       return new NextResponse("Id store dibutuhkan", { status: 400 });
880     }
881
882     const store = await db.store.findUnique({
883       where: {
884         id: storeId,
885       },
886     });
887
888     return new NextResponse(JSON.stringify(store), { status: 200 });
889   } catch (error) {
890     console.log("[STORE_GET]", error);
891     return new NextResponse("Internal error", { status: 500 });
892   }
893 }
894
895 export async function PATCH(
896   req: Request,
897   { params: { storeId: string } }
898 ) {
899   try {
900     const { userId } = await auth();
901     const { storeId } = params;
902     const body = await req.json();
903
904     const { name, address, city, state, country, zip } = body;
905
906     if (!userId) {
907       return new NextResponse("Unauthenticated", { status: 401 });
908     }
909
910     if (!name) {
911       return new NextResponse("Nama toko harus diisi", { status: 400 });
912     }
913
914     if (!address) {
915       return new NextResponse("Alamat toko harus diisi", { status: 400 });
916     }
917
918     if (!city) {
919       return new NextResponse("Kota harus diisi", { status: 400 });
920     }
921
922     if (!state) {
923       return new NextResponse("Provinsi harus diisi", { status: 400 });
924     }
925
926     if (!country) {
927       return new NextResponse("Negara harus diisi", { status: 400 });
928     }
929
930     if (!zip) {
931       return new NextResponse("Kode pos harus diisi", { status: 400 });
932     }
933
934     const store = await db.store.update({
935       where: {
936         id: storeId,
937       },
938       data: {
939         name,
940         address,
941         city,
942         state,
943         country,
944         zip,
945       },
946     });
947
948     return new NextResponse(JSON.stringify(store), { status: 200 });
949   } catch (error) {
950     console.log("[STORE_PATCH]", error);
951     return new NextResponse("Internal error", { status: 500 });
952   }
953 }
954
955 export async function DELETE(
956   req: Request,
957   { params: { storeId: string } }
958 ) {
959   try {
960     const { userId } = await auth();
961     const { storeId } = params;
962
963     if (!userId) {
964       return new NextResponse("Unauthenticated", { status: 401 });
965     }
966
967     if (!storeId) {
968       return new NextResponse("Id store dibutuhkan", { status: 400 });
969     }
970
971     const store = await db.store.delete({
972       where: {
973         id: storeId,
974       },
975     });
976
977     return new NextResponse("Store berhasil dihapus", { status: 200 });
978   } catch (error) {
979     console.log("[STORE_DELETE]", error);
980     return new NextResponse("Internal error", { status: 500 });
981   }
982 }
983
984 // Membuat entri baru pada tabel 'billboard' dalam database.
985 const billboard = await db.billboard.create({
986   data: {
987     label: "Billboard Baru",
988     imageUrl: "https://example.com/image.jpg",
989     storeId,
990   },
991 });
992
993 return new NextResponse(JSON.stringify(billboard), { status: 201 });
994 } catch (error) {
995   console.log("[BILLBOARD_CREATE]", error);
996   return new NextResponse("Internal error", { status: 500 });
997 }
998 }
999
1000 export async function GET(
1001   req: Request,
1002   { params: { billboardId: string } }
1003 ) {
1004   try {
1005     const { userId } = await auth();
1006     const { billboardId } = params;
1007
1008     if (!userId) {
1009       return new NextResponse("Unauthenticated", { status: 401 });
1010     }
1011
1012     if (!billboardId) {
1013       return new NextResponse("Id billboard dibutuhkan", { status: 400 });
1014     }
1015
1016     const billboard = await db.billboard.findUnique({
1017       where: {
1018         id: billboardId,
1019       },
1020     });
1021
1022     return new NextResponse(JSON.stringify(billboard), { status: 200 });
1023   } catch (error) {
1024     console.log("[BILLBOARD_GET]", error);
1025     return new NextResponse("Internal error", { status: 500 });
1026   }
1027 }
1028
1029 export async function PATCH(
1030   req: Request,
1031   { params: { billboardId: string } }
1032 ) {
1033   try {
1034     const { userId } = await auth();
1035     const { billboardId } = params;
1036     const body = await req.json();
1037
1038     const { label, imageUrl } = body;
1039
1040     if (!userId) {
1041       return new NextResponse("Unauthenticated", { status: 401 });
1042     }
1043
1044     if (!label) {
1045       return new NextResponse("Label menginput nama", { status: 400 });
1046     }
1047
1048     if (!imageUrl) {
1049       return new NextResponse("menginput Gambar Url", { status: 400 });
1050     }
1051
1052     const billboard = await db.billboard.update({
1053       where: {
1054         id: billboardId,
1055       },
1056       data: {
1057         label,
1058         imageUrl,
1059       },
1060     });
1061
1062     return new NextResponse(JSON.stringify(billboard), { status: 200 });
1063   } catch (error) {
1064     console.log("[BILLBOARD_PATCH]", error);
1065     return new NextResponse("Internal error", { status: 500 });
1066   }
1067 }
1068
1069 export async function DELETE(
1070   req: Request,
1071   { params: { billboardId: string } }
1072 ) {
1073   try {
1074     const { userId } = await auth();
1075     const { billboardId } = params;
1076
1077     if (!userId) {
1078       return new NextResponse("Unauthenticated", { status: 401 });
1079     }
1080
1081     if (!billboardId) {
1082       return new NextResponse("Id billboard dibutuhkan", { status: 400 });
1083     }
1084
1085     const billboard = await db.billboard.delete({
1086       where: {
1087         id: billboardId,
1088       },
1089     });
1090
1091     return new NextResponse("Billboard berhasil dihapus", { status: 200 });
1092   } catch (error) {
1093     console.log("[BILLBOARD_DELETE]", error);
1094     return new NextResponse("Internal error", { status: 500 });
1095   }
1096 }
1097
1098 // Membuat entri baru pada tabel 'store' dalam database.
1099 const store = await db.store.create({
1100   data: {
1101     name: "Store Baru",
1102     address: "Jl. Merdeka No. 123",
1103     city: "Jakarta",
1104     state: "DKI Jakarta",
1105     country: "Indonesia",
1106     zip: "10110",
1107   },
1108 });
1109
1110 return new NextResponse(JSON.stringify(store), { status: 201 });
1111 } catch (error) {
1112   console.log("[STORE_CREATE]", error);
1113   return new NextResponse("Internal error", { status: 500 });
1114 }
1115 }
1116
1117 export async function GET(
1118   req: Request,
1119   { params: { storeId: string } }
1120 ) {
1121   try {
1122     const { userId } = await auth();
1123     const { storeId } = params;
1124
1125     if (!userId) {
1126       return new NextResponse("Unauthenticated", { status: 401 });
1127     }
1128
1129     if (!storeId) {
1130       return new NextResponse("Id store dibutuhkan", { status: 400 });
1131     }
1132
1133     const store = await db.store.findUnique({
1134       where: {
1135         id: storeId,
1136       },
1137     });
1138
1139     return new NextResponse(JSON.stringify(store), { status: 200 });
1140   } catch (error) {
1141     console.log("[STORE_GET]", error);
1142     return new NextResponse("Internal error", { status: 500 });
1143   }
1144 }
1145
1146 export async function PATCH(
1147   req: Request,
1148   { params: { storeId: string } }
1149 ) {
1150   try {
1151     const { userId } = await auth();
1152     const { storeId } = params;
1153     const body = await req.json();
1154
1155     const { name, address, city, state, country, zip } = body;
1156
1157     if (!userId) {
1158       return new NextResponse("Unauthenticated", { status: 401 });
1159     }
1160
1161     if (!name) {
1162       return new NextResponse("Nama toko harus diisi", { status: 400 });
1163     }
1164
1165     if (!address) {
1166       return new NextResponse("Alamat toko harus diisi", { status: 400 });
1167     }
1168
1169     if (!city) {
1170       return new NextResponse("Kota harus diisi", { status: 400 });
1171     }
1172
1173     if (!state) {
1174       return new NextResponse("Provinsi harus diisi", { status: 400 });
1175     }
1176
1177     if (!country) {
1178       return new NextResponse("Negara harus diisi", { status: 400 });
1179     }
1180
1181     if (!zip) {
1182       return new NextResponse("Kode pos harus diisi", { status: 400 });
1183     }
1184
1185     const store = await db.store.update({
1186       where: {
1187         id: storeId,
1188       },
1189       data: {
1190         name,
1191         address,
1192        
```

```

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 :

