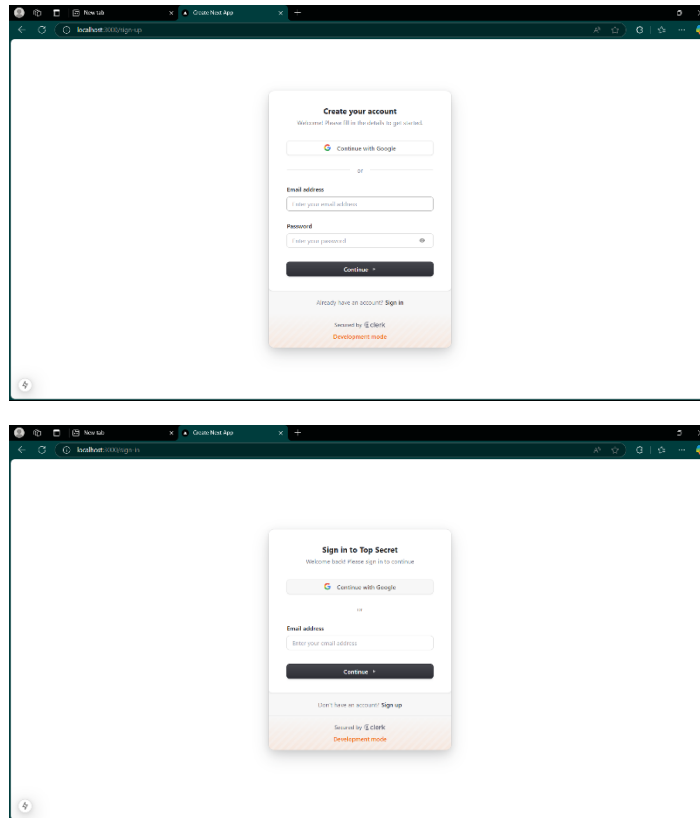


Nama : Maulidan Akmal Gandi

NPM : 22312137

Dokumentasi Project Pemograman Web 2

1. Pembuatan Tampilan Sign-In dan Sign-Up menggunakan Clerk



Sign-In

```
1 import { SignIn } from '@clerk/nextjs'
2
3 export default function Page() {
4   return <SignIn />
5 }
```

Sign-Up

```
1 import { SignUp } from '@clerk/nextjs'
2
3 export default function Page() {
4   return <SignUp />
5 }
```

Digunakan untuk membuat akun dan masuk ketika mempunyai akun dan masuk kedalam website yang sudah dibuat

2. Membuat Database

```
Generate
7  ∨ generator client {
8    |   provider = "prisma-client-js"
9    | }
10
11 ∨ datasource db {
12   |   provider = "postgresql"
13   |   url      = env("DATABASE_URL")
14   | }
15
16 ∨ model Store{
17   |   id String @id @default(uuid())
18   |   name String
19   |   userId String
20   |   createdAt DateTime @default(now())
21   |   updatedAt DateTime @updatedAt
22   | }
```

Membuat database dengan menggunakan provider “postgresql” dan table dengan nama “Store”

3. Membuat API dan Memperbaiki page.tsx di dalam folder app/(root)

```
app > api > store > route.ts > POST > status
1  import { auth } from "@clerk/nextjs"
2  import { NextResponse } from "next/server"
3
4  export async function POST(req: Request){
5    try {
6      const { userId } = auth()
7      const body = await req.json();
8
9      const {name} = body
10
11     if (!userId){
12       return new NextResponse("Unauthorized", {status:401})
13     }
14
15   } catch (error) {
16     console.log("[STORES_POST]", error)
17     return new NextResponse("Internal Error", {status:500})
18   }
19 }
```

Pembuatan API(Belum Selesai)

Root Page

Buat Store

Tambahkan Store untuk membuat produk dan kategori

Nama

Cancel Continue

Diatas adalah tampilan page.tsx dalam folder app/(root) yang sudah diperbaiki dan akan muncul tampilan Buat Store

4. Memperbaiki API agar dapat menambahkan data ke dalam database, menambahkan fungsi agar bisa check data apakah ada data yang belum terisi, dan juga membuat fungsi untuk dapat memasukkan data kedalam prisma

```
1  import db from "@lib/db";
2  import { auth } from "@clerk/nextjs";
3  import { NextResponse } from "next/server";
4
5  export async function POST(req: Request) {
6    try {
7      const { userId } = auth();
8      const body = await req.json();
9
10     const { name } = body;
11
12     if (!userId) {
13       return new NextResponse("Unauthorized", { status: 401 });
14     }
15
16     if (!name) {
17       return new NextResponse("Nama Toko Perlu di Tambahkan", { status: 400 });
18     }
19
20     const store = await db.store.create({
21       data: {
22         name,
23         userId,
24       },
25     });
26     return NextResponse.json(store);
27   } catch (error) {
28     console.log("[STORES_POST]", error);
29     return new NextResponse("Internal Error", { status: 500 });
30   }
31 }
32
```

Ini adalah kode untuk API

```
19  export const StoreModal = () => {
20
21    const [loading, setLoading] = useState(false)
22
23    const storeModal = useStoreModal();
24
25    const form = useForm<z.infer<typeof formSchema>>({
26      resolver: zodResolver(formSchema),
27      defaultValues: {
28        name: ""
29      }
30    })
31
32    const onSubmit = async (values: z.infer<typeof formSchema>) => {
33      try{
34        setLoading(true)
35
36        const response = await axios.post("/api/stores", values);
37        console.log(response.data);
38      }catch(error){
39        console.log(error);
40      }
41    };
42  }
```

Ini adalah fungsi untuk memasukan data kedalam prisma menggunakan axios

5. Membuat provider toast dan memberikan alert jika terjadi kesalahan saat membuat toko dan membuat alert ketika berhasil membuat toko

Membuat Provider Toast

```
1  "use client"
2
3  import { Toaster } from "react-hot-toast"
4
5  export const ToastProvider = () => {
6    return <Toaster >
7
8    </Toaster>;
9  }
```

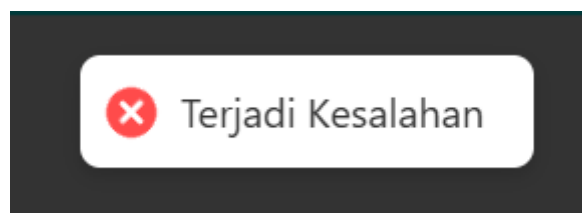
Fungsi untuk membuat alert menggunakan toast

```
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");
  }catch(error){
    toast.error("Terjadi Kesalahan");
  }
  finally{
    setLoading(false);
  }
};
```

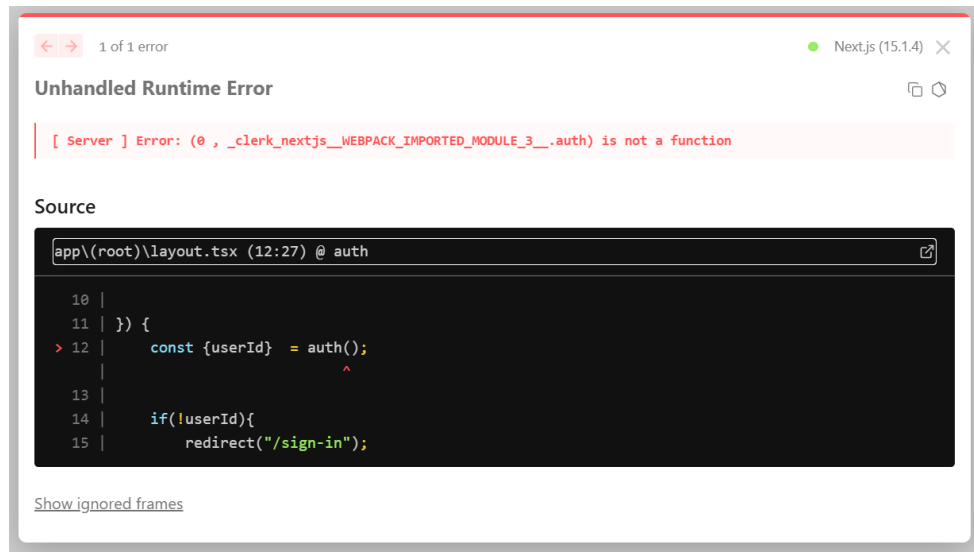
Hasil alert Ketika terjadi kesalahan saat membuat toko



Dan hasil alert Ketika berhasil membuat toko terjadi error karena post tidak terbaca di on submit

```
✖ POST C:\Users\akmal_ghand...\store-modal.tsx:36
http://localhost:3000/api/stores 500 (Internal Server Error)
onSubmit @ C:\Users\akmal_ghand...\store-modal.tsx:36
Show ignore-listed frames
```

6. Membuat dashboard layout dan membuat root layout agar setelah membuat toko langsung dapat menuju dashboard layout



Belum dapat terbaca dikarenakan error pada auth di dalam file layout tsx pada folder (root).

```
1 import db from "@lib/db";  
2 import { auth } from "@clerk/nextjs";  
3 import { redirect } from "next/navigation";  
4  
5 export default async function DashboardLayout({  
6   children,  
7   params,  
8 }): {  
9   children: React.ReactNode  
10  params: {storeId: string}  
11 } {  
12   const { userId } = auth();  
13  
14   if(!userId){  
15     redirect("/sign-in");  
16   }  
17  
18   const store = await db.store.findFirst({  
19     where:{  
20       id: params.storeId,  
21       userId  
22     }  
23   });  
24  
25   if(!store){  
26     redirect("/");  
27   }  
28  
29   return(  
30     <>  
31     <div>Ini Navigasi</div>  
32     {children}  
33     </>  
34   )  
35 }
```

Berikut adalah kode di dalam folder Dashboard/storeid yang didalamnya diberi nama DashboardLayout

```

1  const DashboardPage = () => {
2    |   return (
3    |     <div>Ini Dashboard</div>
4    |   );
5  }
6
7  export default DashboardPage;

```

Berikut adalah kode di dalam folder Dashboard/storied/routes yang didalamnya diberi nama DashboardPage

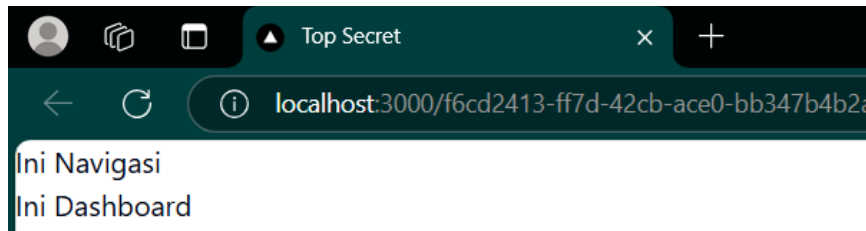
```

1  ∨ import db from "@lib/db";
2  import { auth } from "@clerk/nextjs"
3  import { redirect } from "next/navigation";
4
5  ∨ export default async function SetupLayout({
6    |   children,
7    |
8  ∨ }): {
9    |   children: React.ReactNode;
10   |
11   }) {
12     const {userId} = auth();
13
14     ∨ if(!userId){
15     |   redirect("/sign-in");
16     | }
17
18     ∨ const store = await db.store.findFirst({
19     ∨   where:{
20     |     userId: userId
21     |   }
22     | });
23
24     ∨ if(store){
25     |   redirect(`/${store.id}`);
26     | }
27
28     ∨ return(
29     ∨   <>
30     |   {children}
31     | </>
32     | );
33   };

```

Berikut adalah kode di dalam folder root yang didalamnya diberi nama SetupLayout

- Memperbaiki layout pada dashboard layout dan setup layout agar memunculkan tampilan halamannya



berikut adalah tampilan dashboard layout

```
app > (dashboard) > [storeId] > layout.tsx > DashboardLayout
1 import db from "@lib/db";
2 import { auth } from "@clerk/nextjs/server";
3 import { redirect } from "next/navigation";
4
5 export default async function DashboardLayout({
6   children,
7   params,
8 }): {
9   children: React.ReactNode
10  params: {storeId: string}
11 } {
12   const { userId } = await auth();
13
14   if(!userId){
15     redirect("/sign-in");
16   }
17
18   const store = await db.store.findFirst({
19     where:{
20       id: params.storeId,
21       userId
22     }
23   });
24
25   if(!store){
26     redirect("/");
```

Berikut kode yang sudah diperbaiki pada baris 12 dan baris 2 di dalam dashboard layout

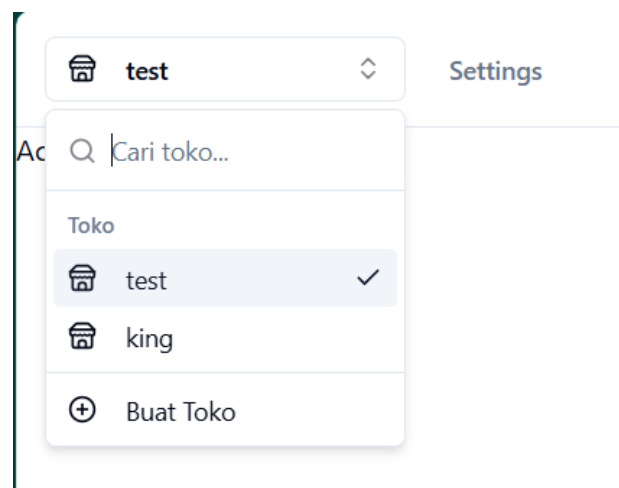
```
1 import db from "@lib/db";
2 import { auth } from "@clerk/nextjs/server";
3 import { redirect } from "next/navigation";
4
5 export default async function SetupLayout({
6   children,
7   params,
8 }): {
9   children: React.ReactNode;
10 } {
11 } {
12   const {userId} = await auth();
13
14   if(!userId){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

Berikut kode yang sudah diperbaiki pada baris 12 dan baris 2 di dalam setup layout

8. Memperbaiki bar navigasi agar bisa menampilkan toko yang sudah di buat dan bisa membuat toko

```
1 import { UserButton } from "@clerk/nextjs";
2 import { MainNav } from "../main-nav";
3 import StoreSwitcher from "../store-switcher";
4 import { auth } from "@clerk/nextjs/server";
5 import { redirect } from "next/navigation";
6 import db from "@lib/db";
7
8 const Navbar = async () => {
9   const { userId } = await auth();
10
11   if(!userId){
12     redirect("/sign-in");
13   }
14
15   const stores = await db.store.findMany({
16     where:{
17       userId: userId
18     }
19   });
20
21   return (
22     <div className="border-b">
23       <div className="flex h-16 items-center px-4 ">
24         <UserButton />
25         <StoreSwitcher items={stores} />
26       </div>
27     </div>
28   );
29 }
```

Berikut fungsi kode yang diperbaiki dan digunakan untuk menampilkan toko yang sudah dibuat dan bisa membuat toko di bar navigasi



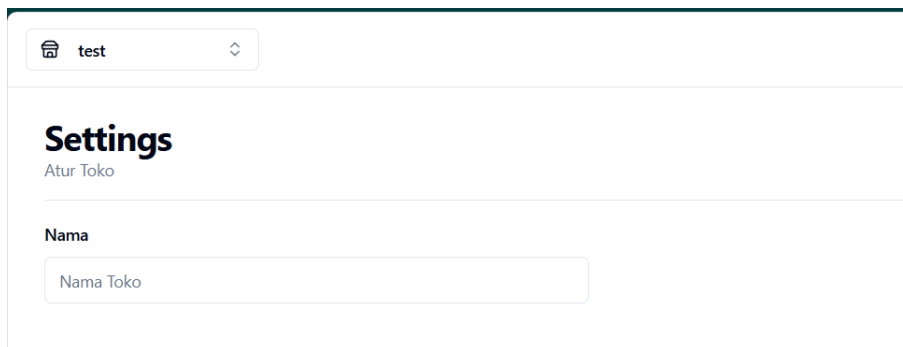
Berikut bar navigasi yang dibuat yang menampilkan nama toko yang sudah dibuat dan bisa membuat toko

9. Memperbaiki fungsi untuk menu settings agar bisa menampilkan apa yang ada di dalam menu settings

```
44 const useFormField = () => {
45   const fieldContext = React.useContext(FormFieldContext)
46   const itemContext = React.useContext(FormItemContext)
47   const { getFieldState, formState } = useFormContext() || {}
48
49   const fieldState = fieldContext?.name ? getFieldState?.(fieldContext.name, formState) : {}
50
51   if (!fieldContext.name) {
52     throw new Error("useFormField should be used within <FormField>")
53   }
54 }
```

```
6 const Separator = React.forwardRef<
7   HTMLDivElement,
8   React.HTMLAttributes<HTMLDivElement> & {
9     orientation?: "horizontal" | "vertical"
10    decorative?: boolean
11  }
12 >((
13   {
14     className, orientation = "horizontal", decorative = true, ...props },
15     ref
16   ) => {
17     <div
18       ref={ref}
19       role={decorative ? "none" : "separator"}
20       aria-orientation={orientation}
21       className={cn(
22         "shrink-0 bg-border",
23         orientation === "horizontal" ? "h-[1px] w-full" : "h-full w-[1px]",
24         className
25       )}
26     />
```

Berikut adalah kode yang diperbaiki agar menampilkan tampilan menu settings



Berikut adalah tampilan menu settings yang sudah ditampilkan tetapi masih ada yang perlu ditambahkan

10. Membuat fungsi untuk update data dan hapus data

```
1 import db from "@lib/db";
2 import { auth } from "@clerk/nextjs/server"
3 import { NextResponse } from "next/server"
4
5 export async function PATCH(
6   req: Request,
7   {params}: {params: {storeId: string}}
8 ){
9   try {
10     const { userId } = await auth()
11     const body = await req.json();
12
13     const { name } = body;
14
15     if (!userId){
16       return new NextResponse("Unauthenticated", {status: 401})
17     }
18
19     if (!name){
20       return new NextResponse("Silahkan Input Nama Terlebih Dahulu", {status: 400})
21     }
22     if (!params){
23       return new NextResponse("Store ID dibutuhkan", {status: 400})
24     }
25
26     const store = await db.store.updateMany({
27       where: {
28         id: params.storeId,
29         userId
30       },
31       data:{
32         name
33       }
34     })
35     return NextResponse.json(store)
36
37   } catch (error) {
38     console.log("[STORE_PATCH]", error)
39     return new NextResponse("Internal Error", {status: 500})
40   }
41 }
```

Berikut adalah fungsi untuk mengupdate data

```
43 export async function DELETE(
44   req: Request,
45   {params}: {params: {storeId: string}}
46 ){
47   try {
48     const { userId } = await auth()
49
50
51     if (!userId){
52       return new NextResponse("Unauthenticated", {status: 401})
53     }
54     if (!params){
55       return new NextResponse("Store ID dibutuhkan", {status: 400})
56     }
57
58     const store = await db.store.deleteMany({
59       where: {
60         id: params.storeId,
61         userId
62       }
63     })
64     return NextResponse.json(store)
65
66   } catch (error) {
67     console.log("[STORE_DELETE]", error)
68     return new NextResponse("Internal Error", {status: 500})
69   }
70 }
```

Berikut adalah fungsi untuk menghapus data

11. Menarik fungsi untuk mengupdate dan menghapus toko dari folder api/stores/[storeId]

```

30 export const SettingsForm: React.FC<SettingPageProps> = (
31   {initialData}
32 ) => {
33   const params = useParams()
34   const router = useRouter()
35
36   const [loading, setLoading] = useState(false)
37
38   const form = useForm<SettingsFormValue>({
39     resolver: zodResolver(formSchema),
40     defaultValues: initialData,
41   });
42
43   const onSubmit = async(data: SettingsFormValue) => {
44     try {
45       setLoading(true);
46       await axios.patch(`/api/stores/${params.storeId}`, data)
47       router.refresh()
48       toast.success("Toko Berhasil diupdate")
49     } catch (error) {
50       toast.error("cek kembali data yang sudah di inputkan")
51     } finally{
52       setLoading(false)
53     }
54   }
55 }

```

Berikut kode yang dibuat untuk menarik fungsi mengupdate toko yang sudah diinputkan

Berikut adalah tampilan Ketika berhasil mengupdate toko

	id	name	userid	createAt	updateAt
web-admin	7743ee45-d8e5-400a-954...	astrostore	user_2rLeG17kqyIUPjhe3...	2025-01-17 09:37:21.531	2025-01-21 09:38:58.561
public	f6cd2413-ff7d-42cb-ace...	test-store	user_2rLeG17kqyIUPjhe3...	2025-01-17 08:56:32.434	2025-01-21 09:34:15.451

Berikut tampilan data yang sudah di ubah di dalam database

```

59
60     const onDelete = async () => {
61       try {
62         setLoading(true)
63         await axios.delete(`/api/stores/${params.storeId}`)
64         toast.success("Toko Berhasil Dihapus")
65         if (router) {
66           router.push("/")
67         }
68       } catch (error) {
69         toast.error("Cek kembali data dan koneksi anda")
70       } finally {
71         setLoading(false)
72         setOpen(false)
73       }
74     }
75
76     return(
77       <>
78       <AlertModal
79         isOpen={open}
80         onClose={() => setOpen(false)}
81         onConfirm={onDelete}
82         loading={loading}
83       />

```

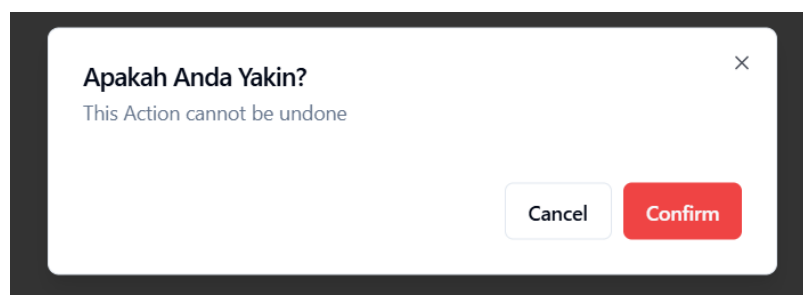
Berikut kode yang dibuat untuk menarik fungsi menghapus toko yang sudah diinputkan

```

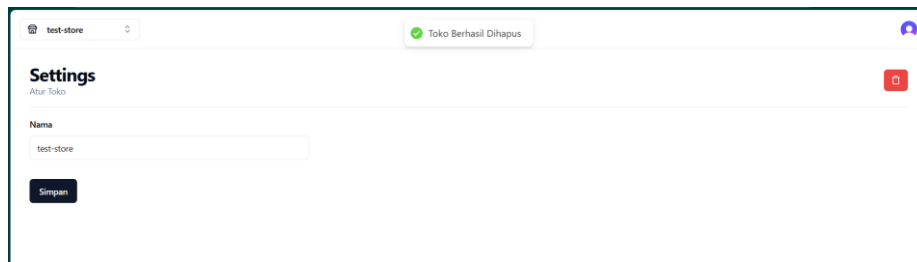
1  "use client"
2
3  import { useEffect, useState } from "react";
4  import Modal from "../ui/modal";
5  import { Button } from "../ui/button";
6
7  interface AlertModalProps {
8    isOpen: boolean;
9    onClose: () => void;
10   onConfirm: () => void;
11   loading: boolean;
12 }
13
14 export const AlertModal: React.FC<AlertModalProps> = ({
15   isOpen,
16   onClose,
17   onConfirm,
18   loading
19 }) => {
20   const [isMounted, setIsMounted] = useState(false)
21   useEffect(() => {
22     setIsMounted(true)
23   }, [])
24   if (!isMounted) {
25     return null
26   }
27   return(
28     <Modal title="Apakah Anda Yakin?" description="This Action cannot be undone" isOpen={isOpen} onClose={onClose}>
29       <div className="pt-6 space-x-2 flex items-center justify-end w-full">
30         <Button disabled={loading} variant="outline" onClick={onClose}>
31           Cancel
32         </Button>
33         <Button disabled={loading} variant="destructive" onClick={onConfirm}>
34           Confirm
35         </Button>
36       </div>
37     </Modal>
38   )
39 }

```

Berikut adalah kode alert yang dibuat untuk menampilkan alert mengkonfirmasi atau membatalkan menghapus toko



Berikut adalah alert untuk mengkonfirmasi dan membatalkan untuk menghapus toko



Berikut adalah tampilan data yang sudah berhasil dihapus

web-admin	public	Search...	Store
<div> <div>1 row • 1s</div> <div> <div>50</div> <div>0</div> </div> </div>			
id	name	userid	createAt
7743ee45-d8e5-400a-954...	astrostore	user_2rLe6L7kqyIUPjhe3...	2025-01-17 09:37:21.531

Berikut adalah tampilan data yang sudah hilang di dalam database Ketika berhasil dihapus di website

12. Menambahkan table Banner di dalam database

```

16 model Store {
17   id      String  @id @default(uuid())
18   name    String
19   userId  String
20   banners Banner[] @relation("StoreToBanner")
21   createdAt DateTime @default(now())
22   updatedAt DateTime @updatedAt
23 }
24
25 model Banner {
26   id      String  @id @default(uuid())
27   storeId String
28   store   Store   @relation("StoreToBanner", fields: [storeId], references: [id])
29   label   String
30   createdAt DateTime @default(now())
31   updatedAt DateTime @updatedAt
32
33   @@index([storeId])
34 }

```

Berikut adalah kode untuk Membuat table Banner di dalam database

web-admin	public	Search...	Banner	Store
<div> <div>0 rows • 489ms</div> <div> <div>50</div> <div>0</div> </div> </div>				
id	storeId	label	createdAt	updatedAt

Berikut adalah tampilan table Banner di dalam database web-admin

13. Membuat tombol dan fungsi untuk upload gambar Banner di navigasi banner lalu klik add new

```
"use client";

import { useEffect, useState } from "react";
import { Button } from "../button";
import { ImagePlus, Trash } from "lucide-react";
import Image from "next/image";
import { CldUploadWidget } from "next-cloudinary";

interface ImageUploadProps {
  disabled?: boolean;
  onChange: (value: string) => void;
  onRemove: (value: string) => void;
  value: string[];
}

const ImageUpload: React.FC<ImageUploadProps> = ({
  disabled,
  onChange,
  onRemove,
  value,
}) => {
  const [isMounted, setIsMounted] = useState(false);

  useEffect(() => {
    setIsMounted(true);
  }, []);

  const onUpload = (result: any) => {
    onChange(result.info.secure_url);
  };

  if (!isMounted) {
    return null;
  }

  return (
    <div>
      <div className="mb-4 flex items-center gap-4">
        {value &&
          value.map((url) => (
            <div
              key={url}
              className="relative w-[200px] h-[200px] rounded-md
overflow-hidden"
            >
              <div className="z-10 absolute top-2 right-2">
```

```

        <Button
          type="button"
          onClick={() => onRemove(url)}
          variant={"destructive"}
          size={"icon"}
        >
          <Trash className="h-4 w-4"></Trash>
        </Button>
      </div>
      <Image
        fill
        className="object-cover"
        alt="Image"
        src={url}
      ></Image>
    </div>
  )))
</div>
<CldUploadWidget onSuccess={onUpload} uploadPreset="tscret20">
  ({open}) => {
    const onClick = () => {
      open();
    };
    return (
      <Button
        type="button"
        disabled={disabled}
        variant={"secondary"}
        onClick={onClick}
      >
        <ImagePlus className="h-4 w-4 mr-2"></ImagePlus>
        Upload Gambar Banner
      </Button>
    );
  }
</CldUploadWidget>
</div>
);
};

export default ImageUpload;

```

```

<FormItem>
  <FormLabel>Image</FormLabel>
  <FormControl>
    <ImageUpload
      disabled={loading}
      onChange={(url) => field.onChange(url)}
      onRemove={() => field.onChange("")}
      value={field.value ? [field.value] : []}
    />
  </FormControl>
  <FormMessage />
</FormItem>

```

Berikut adalah code untuk Membuat fungsi dan tombol Upload Gambar Banner

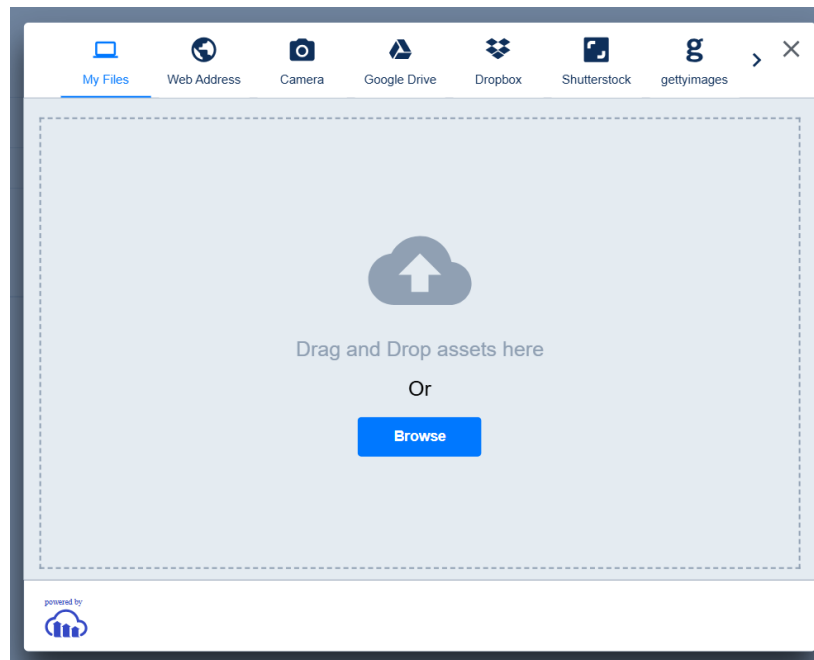
```

1  import type { NextConfig } from "next";
2
3  const nextConfig: NextConfig = {
4    images: {
5      domains: ["res.cloudinary.com"]
6    },
7  };
8
9  export default nextConfig;
10  | Ctrl+L to chat, Ctrl+K to generate

```

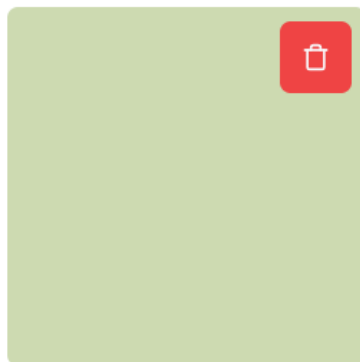
Berikut adalah kode untuk merubah konfigurasi agar cloudinary terbaca oleh next config

Berikut hasil tombol untuk upload gambar banner di navigasi add new banner



Berikut adalah tempat upload banner menggunakan clouldinary

Image



Upload Gambar Banner

Berikut adalah gambar banner yang dipilih

14. Membuat fungsi untuk membuat banner dan menghapus banner

```
58     const onSubmit = async(data: BannerFormValue) => {
59         setloading(true);
60         try {
61             if (initialData) {
62                 await axios.patch(`/api/${params.storeId}/banners/${params.bannerId}`, data);
63             }else{
64                 await axios.post(`/api/${params.storeId}/banners`, data);
65             }
66         }
67         toast.success(toastMessage);
68         if (router) {
69             router.refresh("/")
70         }
71     } catch (error) {
72         toast.error("Cek kembali data yang sudah di inputkan");
73     } finally {
74         setloading(false);
75     }
76 };
77
78     const onDelete = async () => {
79         try {
80             setloading(true)
81             await axios.delete(`/api/${params.storeId}/banners/${params.bannerId}`)
82             toast.success("Banner Berhasil Dihapus")
83             if (router) {
84                 router.push("/")
85             }
86         } catch (error) {
87             toast.error("Cek kembali data dan koneksi anda")
88         }finally{
89             setloading(false)
90             setOpen(false)
91         }
92     }
```

Berikut adalah kode fungsi untuk membuat dan menghapus banner

15. Membuat fungsi untuk check data banner apakah sudah diisi atau belum

```
import db from "@/lib/db";
import { auth } from "@clerk/nextjs/server";
import { NextResponse } from "next/server";

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

        const { label, imageUrl } = body;

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

        if (!label) {
            return new NextResponse("Nama Banner Perlu di Tambahkan", { status: 400
        });
        }

        if (!imageUrl) {
```

```

        return new NextResponse("Image Banner Perlu di Tambahkan", { status:
400 });
    }
    if(!params.storeId){
        return new NextResponse("Store id URL dibutuhkan")
    }
    const storeByUserId = await db.store.findFirst({
        where: {
            id: params.storeId,
            userId: userId
        }
    })

    if(!storeByUserId){
        return new NextResponse("Unauthorized", {status: 403});
    }

    const banner = await db.banner.create({
        data: {
            label,
            imageUrl,
            storeId: params.storeId
        },
    });
    return NextResponse.json(banner);
} catch (error) {
    console.error("[BANNERS_POST]", error);
    return new NextResponse(JSON.stringify({ message: "Internal Error" })), {
status: 500 });
}
}

export async function GET(req: Request,
    {params} : {params: {storeId: string}}) {
    try {
        if(!params.storeId){
            return new NextResponse("Store id URL dibutuhkan")
        }
        const banner = await db.banner.findMany({
            where: {
                storeId: params.storeId
            }
        });
        return NextResponse.json(banner);
    } catch (error) {
        console.error("[BANNERS_GET]", error);
        return new NextResponse(JSON.stringify({ message: "Internal Error" })), {
status: 500 });
    }
}

```

```
}  
}
```

Berikut adalah kode untuk membuat fungsi check data banner

16. Membuat fungsi untuk menyimpan data banner ke dalam database dan terbaca table di halaman banner yang sudah dibuat

```
'use client'  
  
import * as z from 'zod'  
import { use, useState } from 'react'  
  
import { Button } from "@/components/ui/button"  
import { Heading } from "@/components/ui/heading"  
import { Separator } from "@/components/ui/separator"  
import { Store } from "@prisma/client"  
import { Trash } from "lucide-react"  
import { useForm } from 'react-hook-form'  
import { zodResolver } from '@hookform/resolvers/zod'  
import { Form, FormControl, FormField, FormItem, FormLabel, FormMessage }  
from '@/components/ui/form'  
import { Input } from '@/components/ui/input'  
import toast from 'react-hot-toast'  
import axios from 'axios'  
import { useParams } from 'next/navigation'  
import { useRouter } from 'next/compat/router'  
import { AlertModal } from '@/components/modals/alert-modal'  
import { ApiAlert } from '@/components/ui/api-alert'  
import { UseOrigin } from '@/hooks/use-origin'  
import ImageUpload from '@/components/ui/image-upload'  
  
interface BannerFormProps {  
  initialData: Banner | null;  
}  
  
const formSchema = z.object({  
  label: z.string().min(1),  
  imageUrl: z.string().min(1)  
})  
  
type BannerFormValue = z.infer<typeof formSchema>  
  
export const BannerForm: React.FC<BannerFormProps> = (  
  {initialData}  
) => {  
  const params = useParams();  
  const router = useRouter();  
  const origin = UseOrigin();
```

```

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

const title = initialData ? "Edit Banner" : "Buat Banner"
const description = initialData ? "Edit Banner toko" : "Buat Banner
toko"
const toastMessage = initialData ? "Banner Berhasil di edit" : "Banner
Berhasil dibuat"
const action = initialData ? "Simpan Banner" : "Buat Banner"

const form = useForm<BannerFormValue>({
  resolver: zodResolver(formSchema),
  defaultValues: initialData || {
    label: '',
    imageUrl: ''
  }
});

const onSubmit = async(data: BannerFormValue) => {
  try {
    setLoading(true);
    if (initialData) {
      await
axios.patch(`/api/${params.storeId}/banners/${params.bannerId}`, data);
    }else{
      await axios.post(`/api/${params.storeId}/banners`, data);
    }
    toast.success("Toko Berhasil diupdate");
    if (router) {
      router.refresh("/")
    }
  } catch (error) {
    toast.error("Cek kembali data yang sudah di inputkan");
  } finally {
    setLoading(false);
  }
};

const onDelete = async () => {
  try {
    setLoading(true)
    await
axios.delete(`/api/${params.storeId}/banners/${params.bannerId}`);
    toast.success("Banner Berhasil Dihapus")
    if (router) {
      router.push("/")
    }
  } catch (error) {

```

```

        toast.error("Cek kembali data dan koneksi anda")
    }finally{
        setLoading(false)
        setOpen(false)
    }
}

return(
    <>
    <AlertModal
    isOpen={open}
    onClose={() => setOpen(false)}
    onConfirm={onDelete}
    loading={loading}
    />
    <div className="flex items-center justify-between">
        <Heading
            title={title}
            description={description}
        />
        {initialData &&(
            <Button
                disabled={loading}
                variant="destructive"
                size="sm"
                onClick={() => setOpen(true)}
            >
                <Trash className="h-4 w-4"/>
            </Button>
        )}
    </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="label"
                render={({field}) => (
                    <FormItem>
                        <FormLabel>Label</FormLabel>
                        <FormControl>
                            <Input placeholder="Label Banner"
disabled={loading} value={field.value} onChange={field.onChange} />
                        </FormControl>
                        <FormMessage />
                    </FormItem>
                )}
            />

```

```

    })
  />

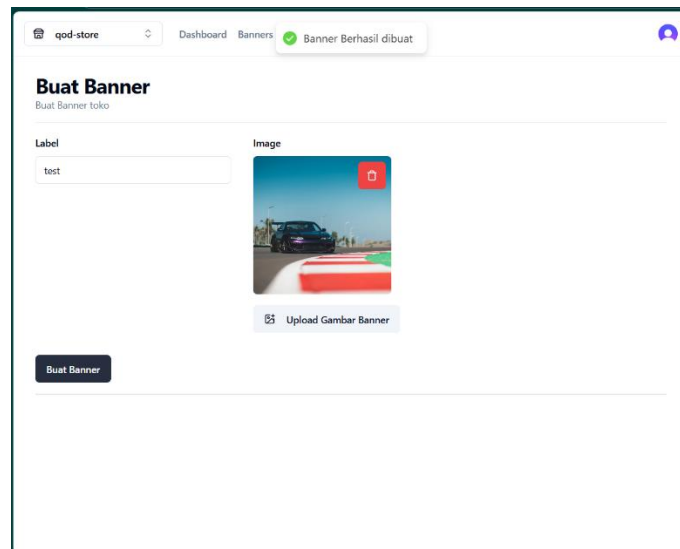
  <FormField
    control={form.control}
    name="imageUrl"
    render={({field}) => (
      <FormItem>
        <FormLabel>Image</FormLabel>
        <FormControl>
          <ImageUpload
            disabled={loading}
            onChange={(url: string) =>
field.onChange(url)}
            onRemove={() => field.onChange("")}
            value={field.value ? [field.value] :
[]}]
          />
        </FormControl>
        <FormMessage />
      </FormItem>
    )}
  />

</div>
<Button
  disabled={loading}
type='submit'
  >
    {action}
  </Button>
</form>
</Form>
<Separator />
</>

);
};

```

Berikut adalah fungsi untuk tombol agar di klik akan menyimpan data banner



Berikut adalah tampilan jika sudah dapat mengupload banner

label	date
qod	Jan 25th, 2025
test	Jan 25th, 2025

Previous Next

Berikut adalah tampilan table data banner yang sudah di buat

Tables

web-admin public Banner Store

Search...

To exit full screen, move mouse to top of screen or press and hold **Esc**

id	storeId	label	createdAt	updatedAt	imageUrl	Store
83c3b2d4-82c9-4769-a92...	88a9b88-6954-4c05-8c8...	test	2025-01-25 13:12:54.987	2025-01-25 13:12:54.987	https://res.cloudinary...	Store
43489980-2959-497a-958...	88a9b88-6954-4c05-8c8...	qod	2025-01-25 13:13:41.841	2025-01-25 13:13:41.841	https://res.cloudinary...	Store

Berikut adalah data banner yang masuk Ketika di klik Buat Banner

17. Update database menambahkan imageUrl di dalam database

```

25 model Banner {
26   id      String @id @default(uuid())
27   storeId String
28   store   Store @relation("StoreToBanner", fields: [storeId], referen
29   label   String
30+  imageUrl String
31   createdAt DateTime @default(now())
32   updatedAt DateTime @updatedAt
33
34   @@index([storeId])
35 }

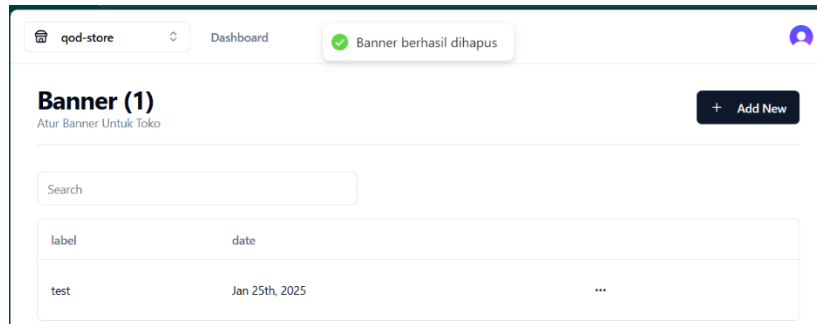
```

Berikut adalah kode imageUrl yang ditambahkan di schema.prisma dan sudah di push ke dalam database

18. Memperbaiki fungsi tombol hapus untuk menghapus table di dalam menu banner dan di database

```
setLoading(true);  
await axios.delete(`/api/${params.storeId}/banners`, {data});  
router.refresh();
```

Berikut adalah kode yang di perbaiki untuk menghapus data banner di dalam table menu banner dan database



Berikut adalah tampilan data yang berhasil di hapus yang bernama banner qod

Tables

web-admin

public

Search...

Banner

Store

Filters

Columns

+ Add record

1 row • 2s

id	storeId	label	createdAt	updatedAt	imageUrl	Store
83c1883d-8dca-476e-a922-8ba9d988-8354-4c05-8d8c		test	2025-01-25 13:12:54.587	2025-01-25 13:12:54.587	https://res.cloudinary.com/qod-store	Store

Berikut adalah tampilan data di dalam database yang telah dihapus bernama qod