

Nama : Muhamad Dimas Stiyawan

NPM : 22312087

Enviroment Setup

1. Menginstall module nextjs

```
Warning: Powershell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes. If you want to re-enable it, run 'Import-Module PSReadLine'.

PS D:\Pemrograman Web 2\eCommerce_VisionVortex> npx create-next-app@latest
Need to install the following packages:
create-next-app@15.1.3
Ok to proceed? (y) y

? What is your project named? ... eCommerce-visionvortex
? Would you like to use TypeScript? ... No / Yes
? Would you like to use ESLint? ... No / Yes
? Would you like to use Tailwind CSS? ... No / Yes
? Would you like your code inside a `src/` directory? ... No / Yes
? Would you like to use App Router? (recommended) ... No / Yes
? Would you like to use Turbopack for `next dev`? ... No / Yes
? Would you like to customize the import alias (`@/*` by default)? ... No / Yes
Creating a new Next.js app in D:\Pemrograman Web 2\eCommerce_VisionVortex\eCommerce-visionvortex.
```

2. menginstall shadcn ui

```
PS D:\Pemrograman Web 2\eCommerce_VisionVortex\eCommerce-visionvortex> npx shadcn@latest init
✓ Preflight checks.
✓ Verifying framework. Found Next.js.
✓ Validating Tailwind CSS.
✓ Validating import alias.
✓ Which style would you like to use? » Default
✓ Which color would you like to use as the base color? » Slate
✓ Would you like to use CSS variables for theming? ... no / yes
✓ Writing components.json.
✓ Checking registry.
✓ Updating tailwind.config.ts
✓ Updating app\globals.css
Installing dependencies.
```

- Menginstall button ui

```
PS D:\Pemrograman Web 2\eCommerce_VisionVortex\eCommerce-visionvortex> npx shadcn@latest add button
✓ Checking registry.

PS D:\Pemrograman Web 2\eCommerce_VisionVortex> cd eCommerce-visionvortex
PS D:\Pemrograman Web 2\eCommerce_VisionVortex\eCommerce-visionvortex> npx shadcn@latest add button
✓ Checking registry.
Installing dependencies.
PS D:\Pemrograman Web 2\eCommerce_VisionVortex\eCommerce-visionvortex> npx shadcn@latest add button
✓ Checking registry.
Installing dependencies.
Installing dependencies.
```

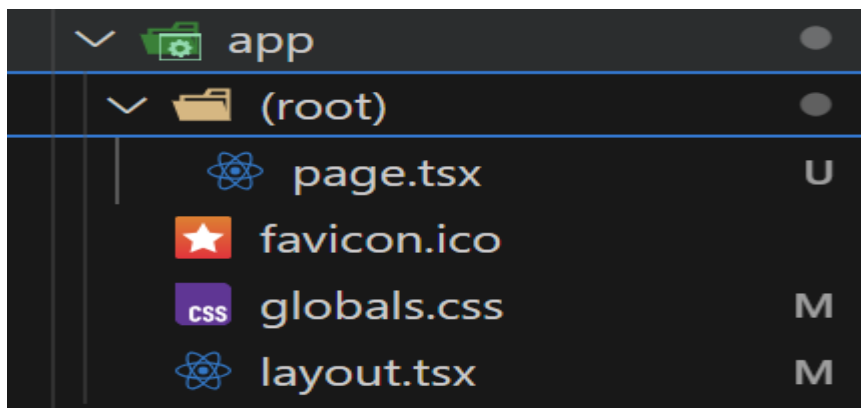
- jika sudah akan muncul folder components yg berisi file button.tsx



3. Memanggil button yang telah diinstal pada halaman page.tsx

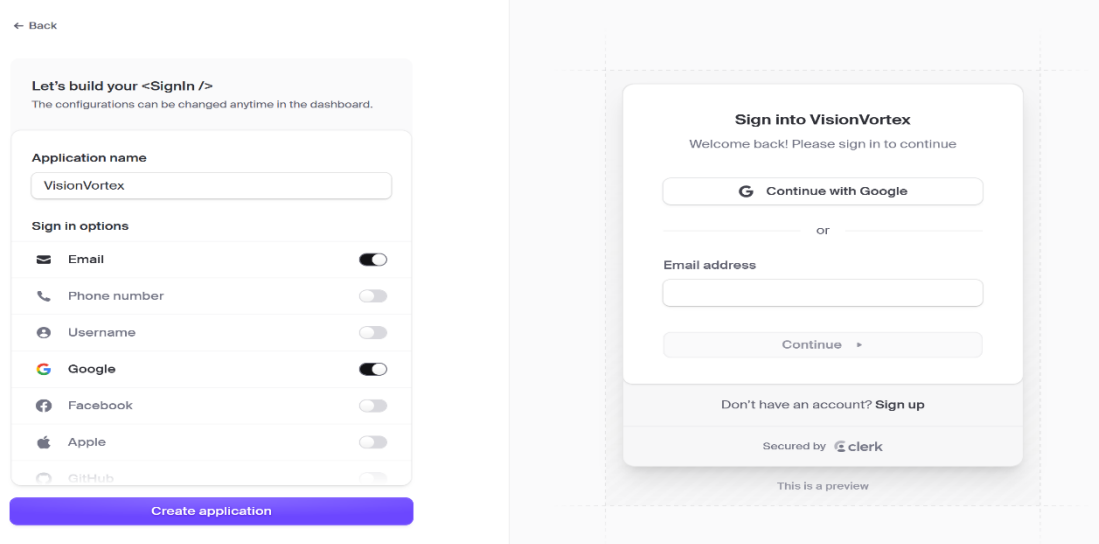
```
ecommerce-visionvortex > app > page.tsx > ...
1  import { Button } from "@components/ui/button";
2  import Image from "next/image";
3
4  export default function Home() {
5    return (
6      <div className="p-4">
7        <Button>Click Me</Button>
8      </div>
9    );
10 }
11
```

4. membuat folder (root) pada folder app dan memindahkan file page.tsx ke dalam folder (root)



Clerk Authentication (Admin)

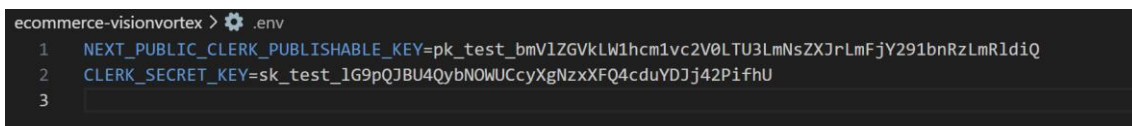
1. buat akun clerk di <https://clerk.com> , kemudian pilih autentikasi email dan google



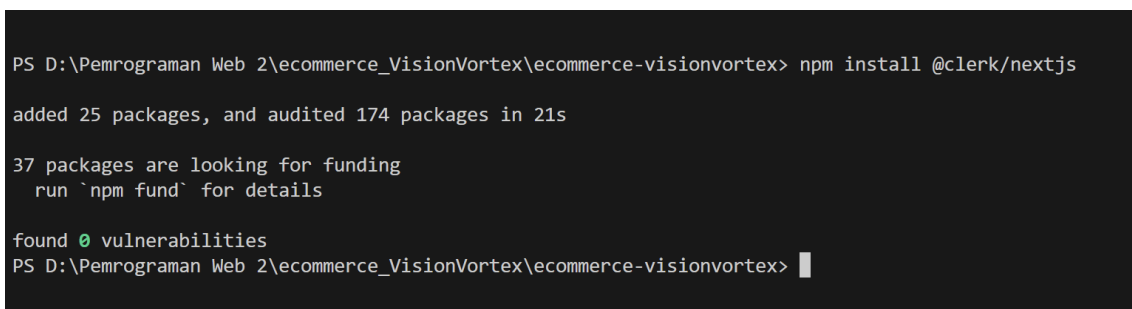
2. buat file .env pada project



3. memasukkan API Keys clerk ke dalam file .env



4. menginstal clerk pada project



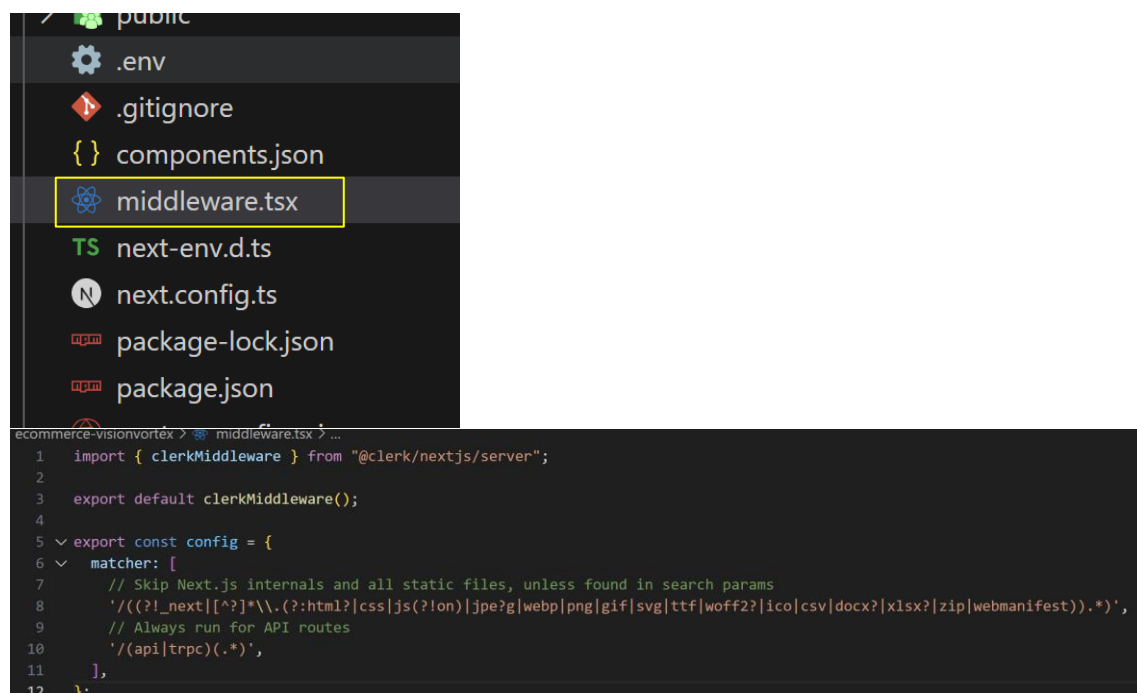
5. mengimport clerkProvider pada file layout.tsx

```

1 import type { Metadata } from "next";
2 import { Geist, Geist_Mono } from "next/font/google";
3 import { ClerkProvider } from "@clerk/nextjs";
4 import "../globals.css";
5
6 > const geistSans = Geist({...
9 });
10
11 > const geistMono = Geist_Mono({...
14 });
15
16 export const metadata: Metadata = {
17   title: "Admin Dashboard",
18   description: "Admin Dashboard",
19 };
20
21 export default function RootLayout({
22   children,
23 }: Readonly<{
24   children: React.ReactNode;
25 }>) {
26   return (
27     <ClerkProvider>
28       <html lang="en">
29         <body
30           className={` ${geistSans.variable} ${geistMono.variable} antialiased`>
31           >
32             {children}
33           </body>
34         </html>
35       </ClerkProvider>
36     );
37 }
38

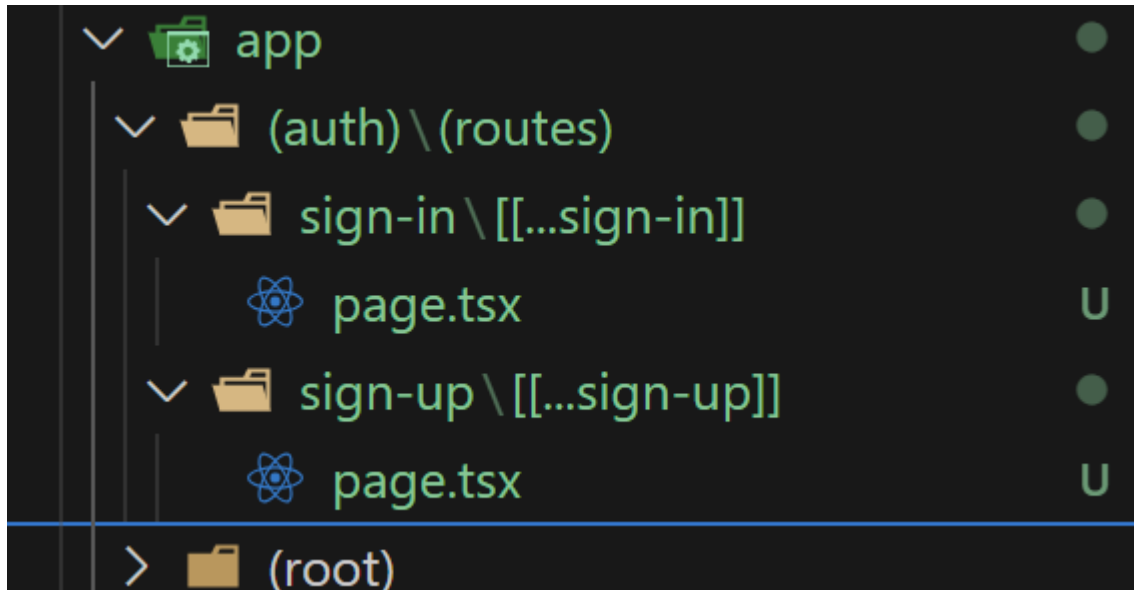
```

6. membuat file middleware.tsx dan menambahkan program config clerk authentication



SIGN-UP dan SIGN-IN

1. membuat folder (auth) kemudian didalam folder (auth) terdapat folder (routes) dan didalam folder routes membuat folder sign-in kemudian dalam folder sign-in terdapat folder [...sign-in] kemudian membuat file page.tsx dalam folder [...sign-in] begitu juga dengan sign-up seperti digambar.



2. memasukkan program sign-in pada file page.tsx dalam folder sign-in

```
ecommerce-visionvortex > app > (auth) > (routes) > sign-in > [...sign-in] > page.tsx > Page
1  import { SignIn } from '@clerk/nextjs'
2
3  export default function Page() {
4    return <SignIn />
5  }
```

3. memasukkan program sign-up pada file page.tsx dalam folder sign-up

```
ecommerce-visionvortex > app > (auth) > (routes) > sign-up > [...sign-up] > page.tsx > Page
1  import { SignUp } from '@clerk/nextjs'
2
3  export default function Page() {
4    return <SignUp />
5  }
```

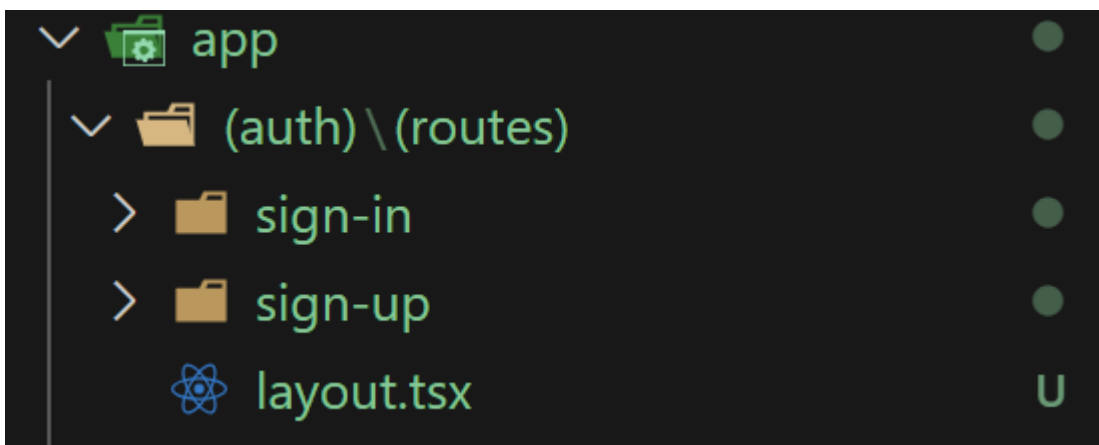
4. menambahkan program pada file .env

```
3  NEXT_PUBLIC_CLERK_SIGN_IN_URL=/sign-in
4  NEXT_PUBLIC_CLERK_SIGN_UP_URL=/sign-up
```

5. tampilan setelah program dijalankan



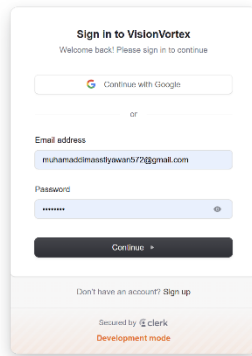
6. membuat file layout.tsx pada folder (routes)



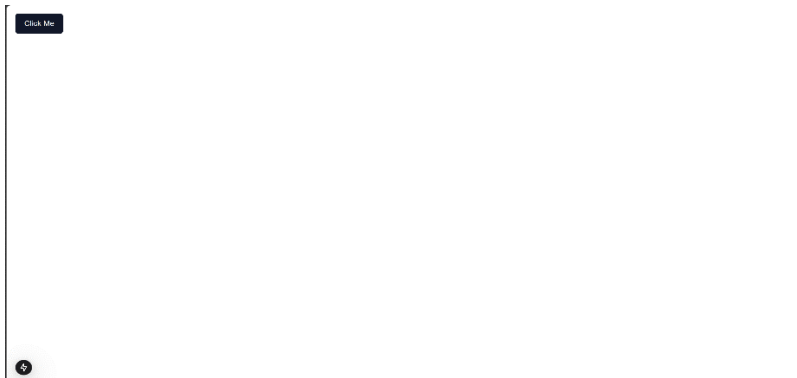
7. menambahkan program pada file layout.tsx untuk mengubah layout sign-in dan sign-up

```
ecommerce-visionvortex > app > (auth) > (routes) > layout.tsx > AuthLayout
1  export default function AuthLayout({
2    children
3  }): {
4    children: React.ReactNode
5  } {
6    return (
7      <div className="flex items-center justify-center min-h-screen ">{children}</div>
8    )
9  }
```

8. tampilan setelah program dijalankan



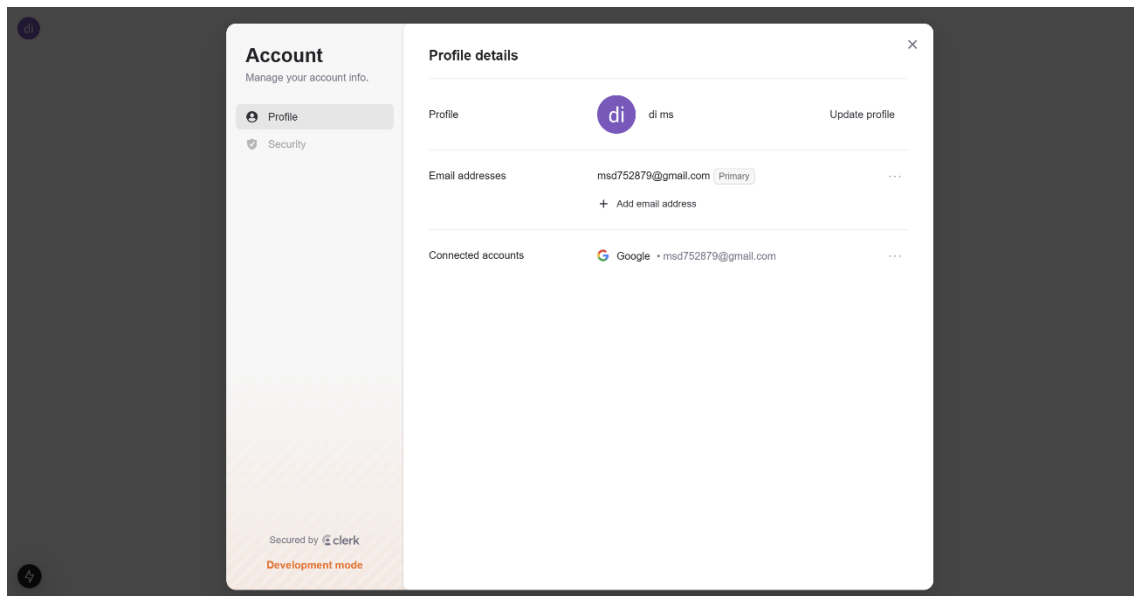
9. tampilan setelah sign-in menggunakan google



10. mengubah program pada file page.tsx dalam folder (root) untuk memanggil profile user

```
ecommerce-visionvortex > app > (root) > page.tsx > ...
1  import { Button } from "@components/ui/button";
2  import { UserButton } from "@clerk/nextjs";
3  import Image from "next/image";
4
5  const setUpPage = () => {
6    return (
7      <div className="p-4">
8        <UserButton></UserButton>
9      </div>
10    );
11  };
12
13  export default setUpPage
14
15
```

11. tampilan setelah program dijalankan

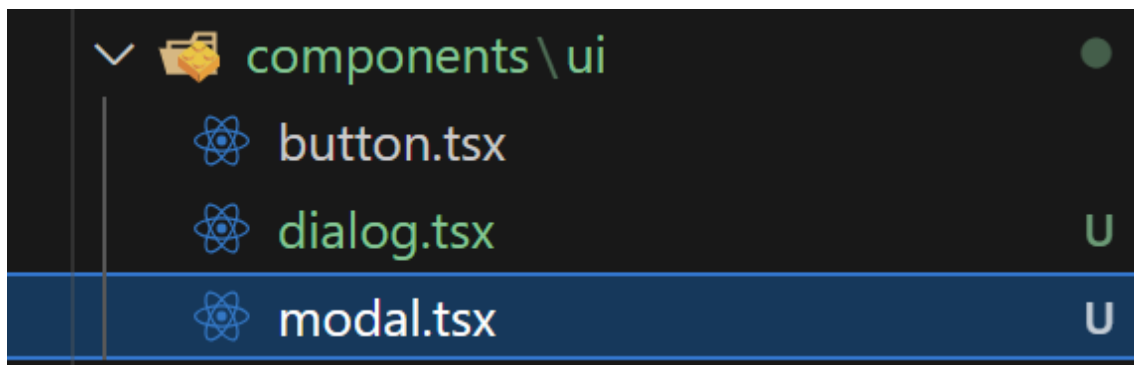


Modal Components

1. menginstal ui dialog dari Shadcnui.com

```
npx shadcn@latest add dialog
```

2. membuat file modal.tsx pada folder components/ui



3. membuat program untuk modal dialog pada file modal.tsx

```
"use client"

// Mengimpor komponen yang dibutuhkan dari library U
import { Dialog, DialogContent, DialogDescription, DialogHeader, DialogTitle } from "@/components/ui/dialog";

// Interface untuk mendefinisikan properti yang akan diterima oleh komponen Modal
interface ModalProps {
  title: string;
  description: string;
  isOpen: boolean;
  onClose: () => void;
  children?: React.ReactNode;
}

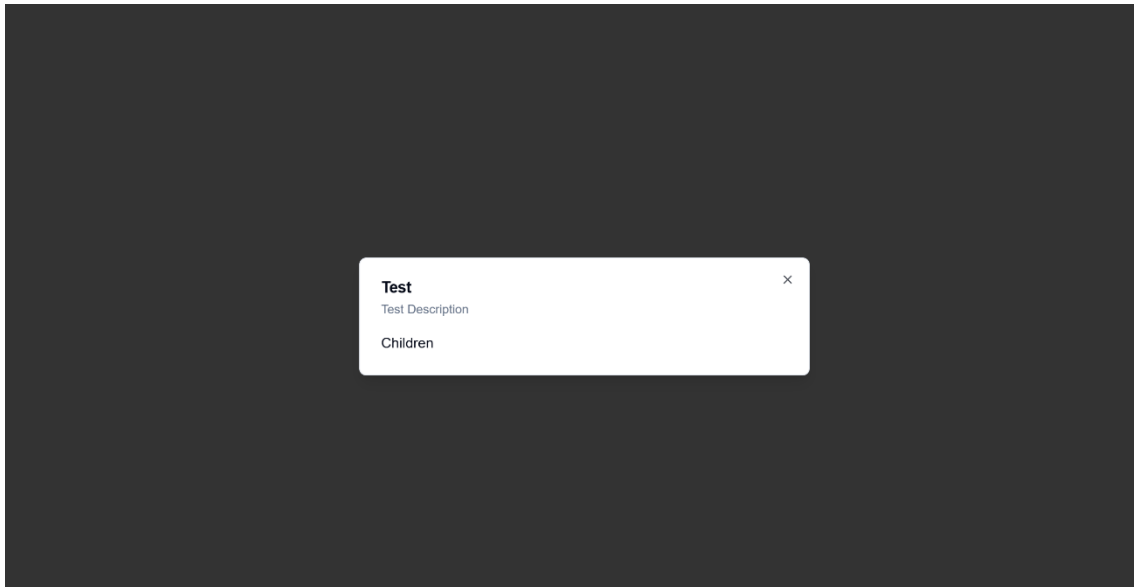
// Deklarasi komponen Modal menggunakan React.FC dengan properti yang sesuai dengan interface ModalProps
export const Modal: React.FC<ModalProps> = ({
  title,
  description,
  isOpen,
  onClose,
  children
}) => {
  // Fungsi untuk menangani perubahan status dialog (terbuka atau tertutup)
  const onChange = (open: boolean) => {
    if (!open) {
      onClose();
    }
  };

  // Komponen yang dirender
  return (
    <Dialog open={isOpen} onOpenChange={onChange}>
      <DialogContent>
        <DialogHeader>
          <DialogTitle>{title}</DialogTitle>
          <DialogDescription>
            {description}
          </DialogDescription>
        </DialogHeader>
        <div>
          {children}
        </div>
      </DialogContent>
    </Dialog>
  );
};
```

4. mengubah program pada file page.tsx dalam folder (root)

```
1  "use client"
2  // Mengimpor komponen `Modal` dari path `@/components/ui/modal`.
3  import { Modal } from "@/components/ui/modal";
4
5
6  // Fungsi `setUpPage` adalah komponen fungsional React yang digunakan untuk merender halaman.
7  const setUpPage = () => {
8    return (
9      <div className="p-4">
10       <Modal title="Test" description="Test Description" isOpen onClose={() => {}}>
11         Children
12       </Modal>
13     </div>
14   );
15 }
16
17
18 // Mengekspor `setUpPage` sebagai default, sehingga komponen ini dapat digunakan di file lain.
19 export default setUpPage
20
```

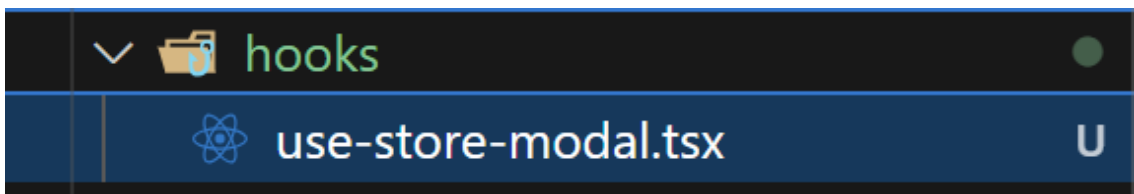
tampilan ketika program dijalankan



5. menginstall library zustand untuk mengontrol apakah modal kita terbuka atau tidak.

```
npm install zustand
```

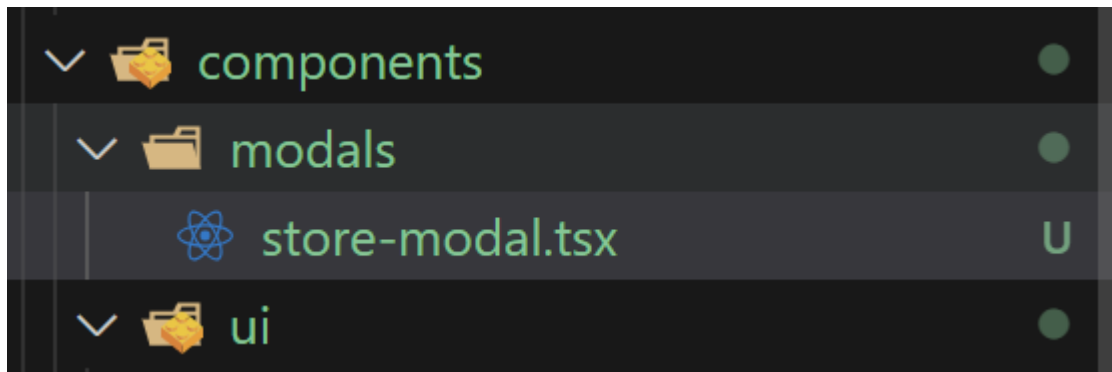
6. membuat folder hooks dalam project kemudian membuat file use-store-modal.tsx dalam folder hooks



7. menambahkan program pada file use-store-modal.tsx berfungsi untuk mengelola status modal secara global menggunakan pustaka Zustand.

```
1 // Mengimpor fungsi 'create' dari pustaka Zustand untuk membuat store global pada aplikasi React.
2 import { create } from "zustand";
3
4 // Mendefinisikan tipe data untuk store yang mengelola status modal.
5 interface useStoreModalStore{
6   isOpen: boolean;
7   onOpen: () => void;
8   onClose: () => void;
9 };
10
11 // Membuat store dengan Zustand dan mendefinisikan status dan fungsi pengelolaan modal.
12 export const useStoreModal = create<useStoreModalStore>((set) => ({
13   isOpen: false,
14   onOpen: () => set({isOpen: true}),
15   onClose: () => set({isOpen: false}),
16 }));
```

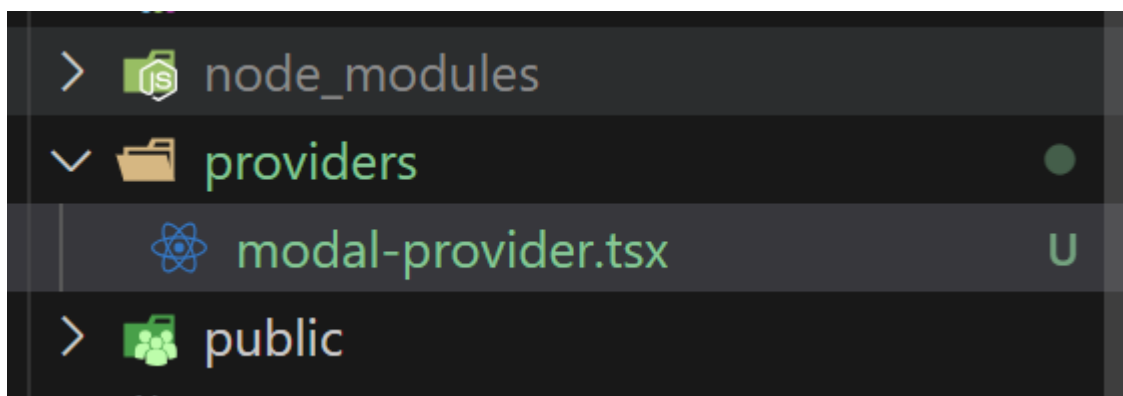
8. membuat folder modals dalam folder components, kemudian membuat file store-modal.tsx dalam folder modals



9. menambahkan program pada file store-modal.tsx untuk menampilkan sebuah modal bernama `StoreModal`, yang digunakan untuk menambahkan toko (store) baru. Modal ini dikelola menggunakan Zustand store melalui hook `useStoreModal`.

```
1  "use client"
2
3  // Mengimpor hook `useStoreModal` yang digunakan untuk mengakses state global modal dari Zustand.
4  import { useStoreModal } from "@hooks/use-store-modal"
5  // Mengimpor komponen `Modal` dari path lokal. Komponen ini bertugas untuk menampilkan dialog/modal.
6  import { Modal } from "../ui/modal";
7
8  // Mendefinisikan komponen fungsional React bernama `StoreModal`.
9  export const StoreModal = () => {
10     // Mengakses state dan fungsi dari store modal menggunakan hook `useStoreModal`.
11     const storeModal = useStoreModal();
12
13     return (
14       <Modal
15         title="create store"
16         description="Add a new store to manage products and categories"
17         isOpen={storeModal.isOpen}
18         onClose={storeModal.onClose}
19       >
20         future Create Sore Form
21         { /* Konten modal yang akan muncul di dalamnya. Saat ini berupa placeholder teks,
22            tetapi akan digantikan dengan form atau elemen lain di masa mendatang. */ }
23       </Modal>
24     )
25   }
```

10. membuat folder providers kemudian didalamnya terdapat file modal-provider.tsx



11. menambahkan program pada file modal-provider.tsx yang berfungsi sebagai provider untuk modal. Komponen ini memastikan bahwa modal hanya dirender di sisi klien (bukan di server) dan memanfaatkan modal `StoreModal` sebagai salah satu contoh modal yang dikelola.

```

1  "use client"
2  // Mengimport komponen `StoreModal` yang bertugas untuk menampilkan modal untuk menambahkan toko.
3  import { StoreModal } from "@components/modals/store-modal";
4  // Mengimport hook `useEffect` dan `useState` dari React untuk mengelola efek samping dan state lokal.
5  import { useEffect, useState } from "react"
6
7  // `ModalProvider` yang bertugas mengelola render modal di sisi klien.
8  export const ModalProvider = () =>{
9    // State `isMounted` untuk mengecek apakah komponen sudah ter-mount.
10    const [isMounted, setIsMounted] = useState(false);
11
12    useEffect(() =>{
13      // Mengubah `isMounted` menjadi `true` setelah komponen ter-mount di klien.
14      setIsMounted(true)
15    }, []);
16
17    if(!isMounted){
18      // Tidak merender komponen hingga ter-mount di klien.
19      return null;
20    }
21
22    return(
23      <>
24        {/* Render `StoreModal` setelah komponen ter-mount. */}
25        <StoreModal/>
26      </>
27    );
28  };

```

12. memanggil ModalProvider pada halaman layout.tsx

```

ecommerce-visionvortex > app > layout.tsx > ...
1  import type { Metadata } from "next";
2  import { Geist, Geist_Mono } from "next/font/google";
3  import { ClerkProvider } from "@clerk/nextjs";
4
5  import { ModalProvider } from "@providers/modal-provider";
6  import "./globals.css";
7
8
9  > const geistSans = Geist({ ...
12 });
13
14 > const geistMono = Geist_Mono({ ...
17 });
18
19 export const metadata: Metadata = {
20   title: "Admin Dashboard",
21   description: "Admin Dashboard",
22 };
23
24 export default function RootLayout({
25   children,
26 }: Readonly<{
27   children: React.ReactNode;
28 }>) {
29   return (
30     <ClerkProvider>
31     <html lang="en">
32       <body className={` ${geistSans.variable} ${geistMono.variable} antialiased`>
33         <ModalProvider/>
34         {children}
35       </body>
36     </html>
37     </ClerkProvider>
38   );
39 }

```

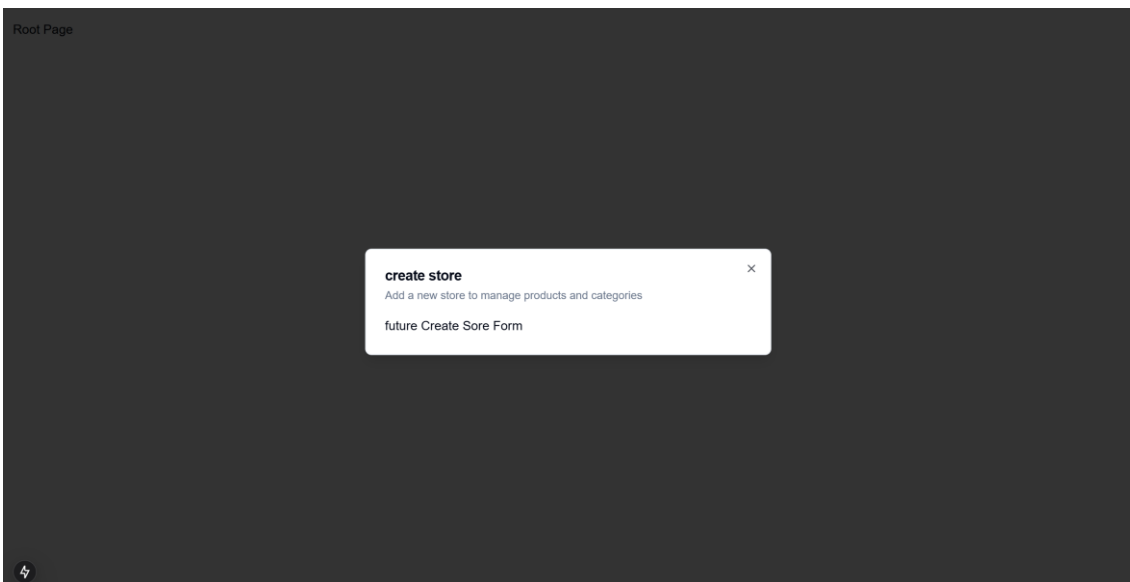
13. Mengubah dan menambahkan program pada file page.tsx yang terdapat pada folder (root).

```

ecommerce-visionvortex > app > (root) > page.tsx > [e] setUpPage
1  "use client"
2
3  // Mengimpor hook `useStoreModal` untuk mengakses store modal.
4  import { useStoreModal } from "@hooks/use-store-modal";
5
6  import { useEffect } from "react";
7
8
9  // Fungsi `setUpPage` adalah komponen fungsional React yang digunakan untuk merender halaman.
10 const setUpPage = () => {
11
12     // Mengambil fungsi `onOpen` dan status `isOpen` dari store `useStoreModal`.
13     const onOpen = useStoreModal((state) => state.onOpen);
14     const isOpen = useStoreModal((state) => state.isOpen);
15
16     // Menggunakan `useEffect` untuk membuka modal jika `isOpen` bernilai false.
17     useEffect(() => {
18         if(!isOpen){
19             onOpen();
20         }
21     }, [isOpen, onOpen]);
22
23
24     return (
25         <div className="p-4">
26             {/* Konten dari halaman, menampilkan teks "Root Page". */}
27             Root Page
28         </div>
29     );
30
31 }
32
33 // Mengekspor `setUpPage` sebagai default, sehingga komponen ini dapat digunakan di file lain.
34 export default setUpPage
35

```

14. Tampilan ketika program dijalankan



BACKEND PRISMA, NEON DAN PostgreSQL

1. menginstal prisma

```
npm install -D prisma
```

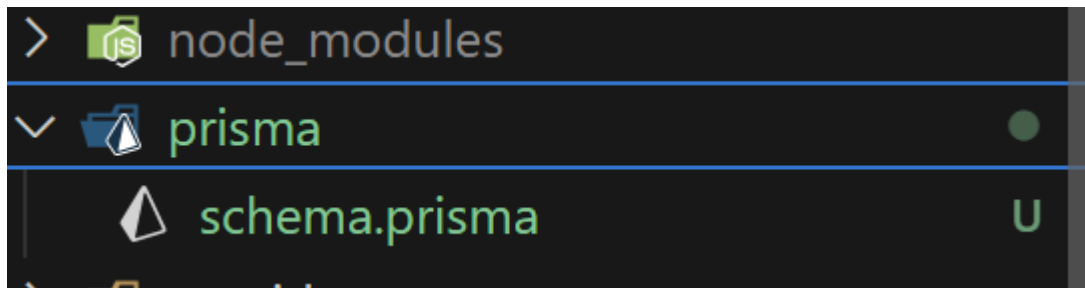
2. menginstal prisma client

```
npm install @prisma/client
```

3. menginstal prisma init

```
npx prisma init
```

Setelah menginstal prisma init akan muncul folder baru yang bernama prisma



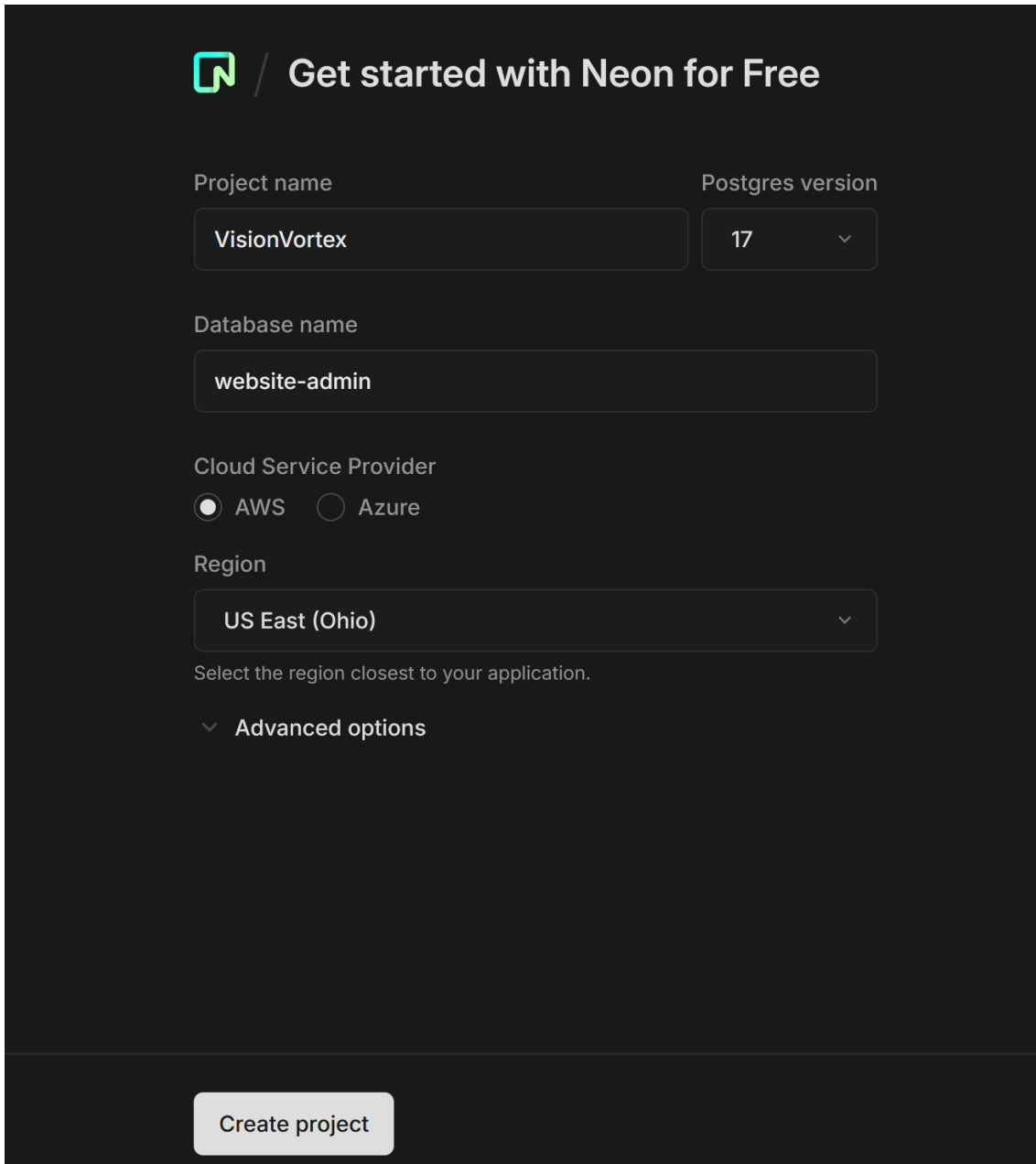
4. membuat file baru bernama db.ts pada folder lib




5. menambahkan program pada file db.ts untuk konfigurasi menggunakan Prisma Client

```
ecommerce-visionvortex > lib > db.tsx > ...
1  import {PrismaClient} from "@prisma/client"
2
3  // Mendeklarasikan variabel global bernama 'prisma'
4  declare global{
5    var prisma: PrismaClient | undefined
6  }
7
8  // Membuat instans PrismaClient
9  const db = global.globalThis.prisma || new PrismaClient();
10 // Menyimpan instans PrismaClient ke 'globalThis.prisma'
11 if (process.env.NODE_ENV !== "production") globalThis.prisma = db
12
13 export default db;
14
```

6. membuat akun dan database di neon.tech



The screenshot shows the 'Get started with Neon for Free' form on the Neon.tech website. The form is set against a dark background with light-colored text and input fields. At the top left is the Neon logo, a green square with a white 'N' inside. To its right is the heading 'Get started with Neon for Free'. Below this, the form is organized into several sections. The first section contains two fields: 'Project name' with the value 'VisionVortex' and 'Postgres version' with a dropdown menu showing '17'. The second section has a 'Database name' field with the value 'website-admin'. The third section, 'Cloud Service Provider', has two radio buttons: 'AWS' (which is selected) and 'Azure'. Below this is a 'Region' dropdown menu showing 'US East (Ohio)'. A small instruction 'Select the region closest to your application.' is placed below the region dropdown. Further down is an expandable section titled 'Advanced options' with a downward arrow icon. At the bottom of the form is a large, light-colored button labeled 'Create project'.

 / Get started with Neon for Free

Project name: VisionVortex

Postgres version: 17

Database name: website-admin

Cloud Service Provider: ☒ AWS ☐ Azure

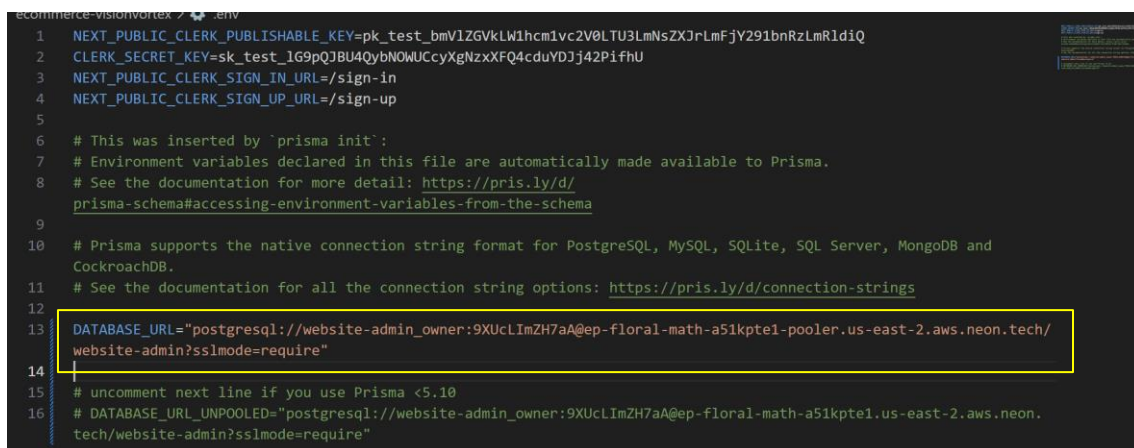
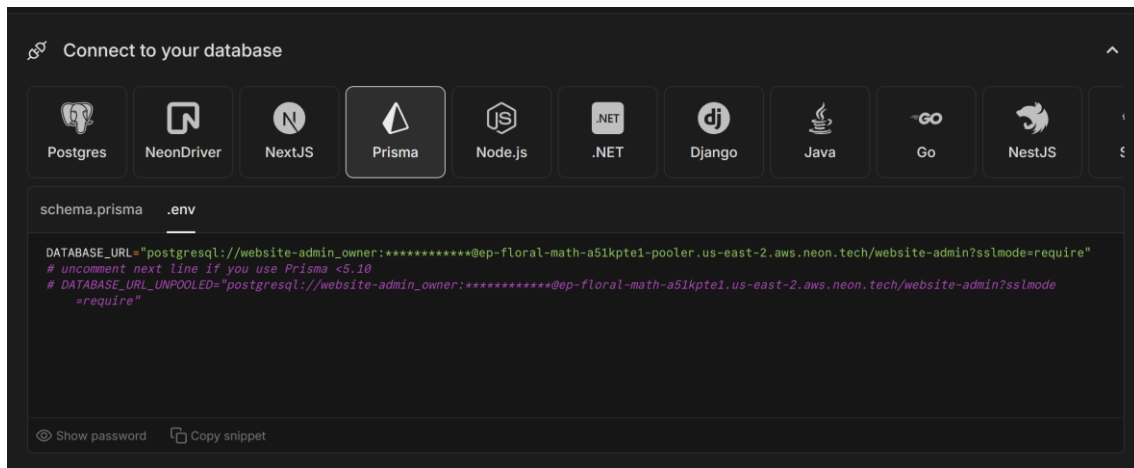
Region: US East (Ohio)

Select the region closest to your application.

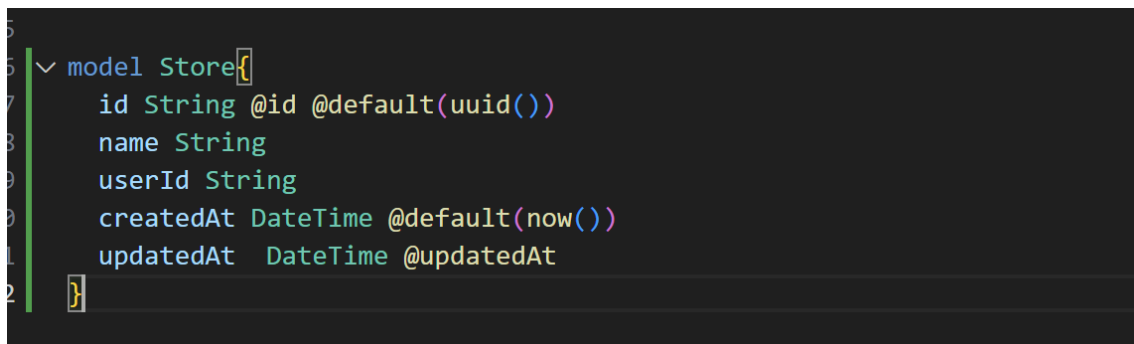
▼ Advanced options

Create project

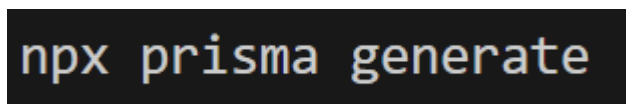
7. menyalin database url pada halaman neon.tech kemudian di tempelkan pada file .env



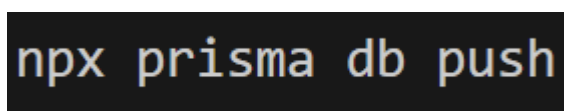
8. Membuat model database pada schema.prisma



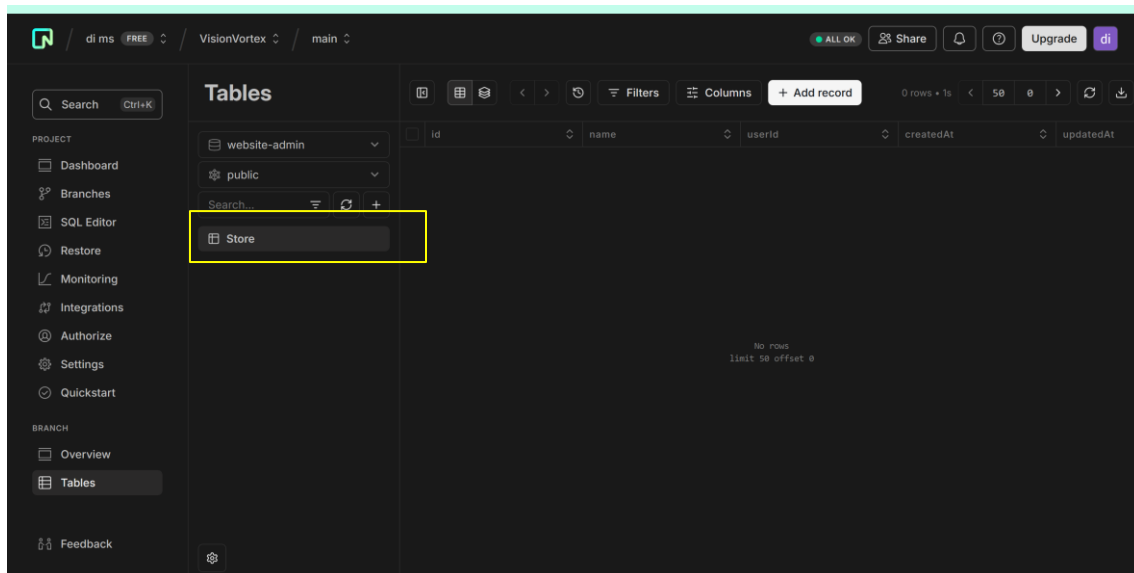
Lalu ketik `npx prisma generate` pada terminal



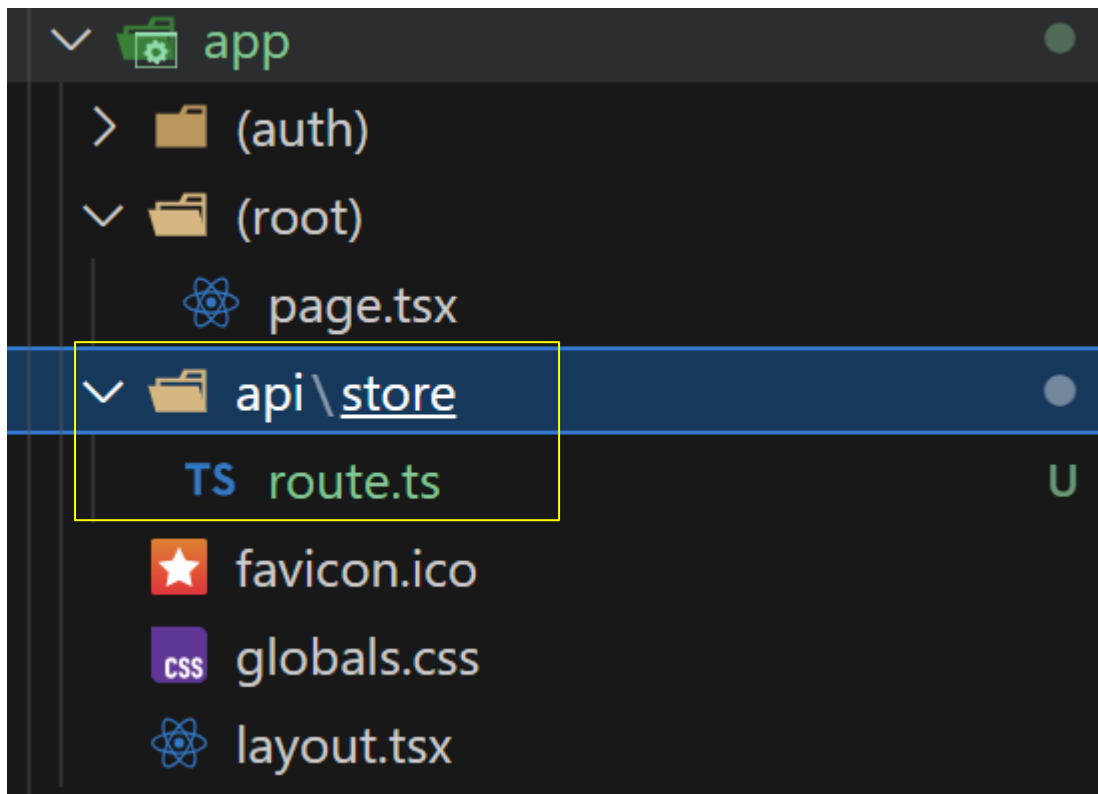
Push schema ke dalam database



Jika berhasil maka didalam web neon akan seperti ini



9. Membuat folder api kemudian didalam folder api tambahkan folder store, didalam folder store tambahkan file route.ts



10. menambahkan program pada file route.ts untuk memproses permintaan POST, dengan autentikasi melalui Clerk, yang merupakan library untuk autentikasi pengguna.

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

11.menambahkan program pada file route.ts untuk Membuat Entri baru pada tabel "store "

```
// Membuat entri baru pada tabel 'store' dalam database.
const store = await db.store.create({
  data: {
    name,
    userId
  },
});
```

12. menginstall axios pada project. *Axios merupakan library opensource yang digunakan untuk request data melalui http*

```
npm install axios
```

13. Menambahkan program pada file store-modal.tsx

Menambahkan variabel dan fungsi loading dan setLoading pada bagian Komponen StoreModal

```
// Mendefinisikan komponen fungsional React bernama `StoreModal`.
export const StoreModal = () => {
  const [loading, setLoading] = useState(false)

  // Mengakses state dan fungsi dari store modal menggunakan hook `useStoreModal`.
  const storeModal = useStoreModal();
```

Menambahkan Program fungsi onSubmit menangani proses pengiriman data dari form ke server melalui API. Fungsi ini mengaktifkan status loading saat pengiriman data dimulai, mengirimkan data ke server menggunakan axios,

menampilkan respons yang diterima dari server jika berhasil, menangani kesalahan jika terjadi error, dan akhirnya mematikan status loading setelah proses selesai, baik berhasil maupun gagal.

```
// Mendefinisikan komponen fungsional React bernama StoreModal :
export const StoreModal = () => {
  const [loading, setLoading] = useState(false)

  // Mengakses state dan fungsi dari store modal menggunakan hook `useStoreModal`.
  const storeModal = useStoreModal();

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

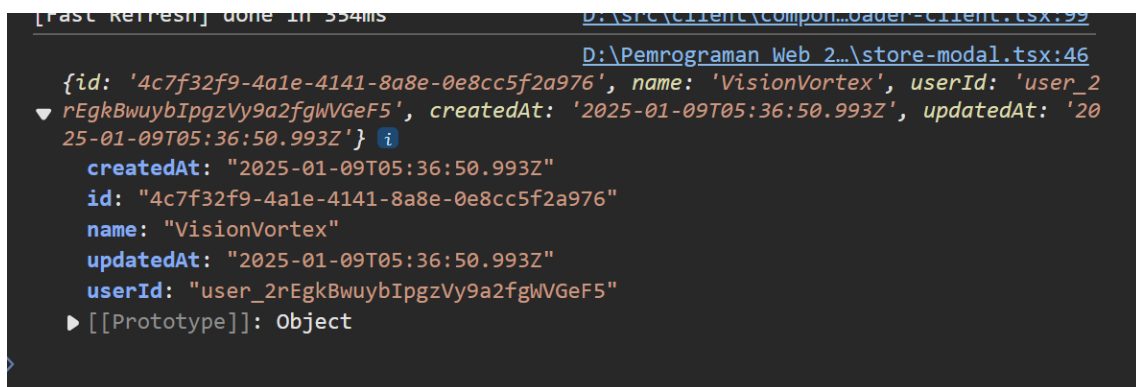
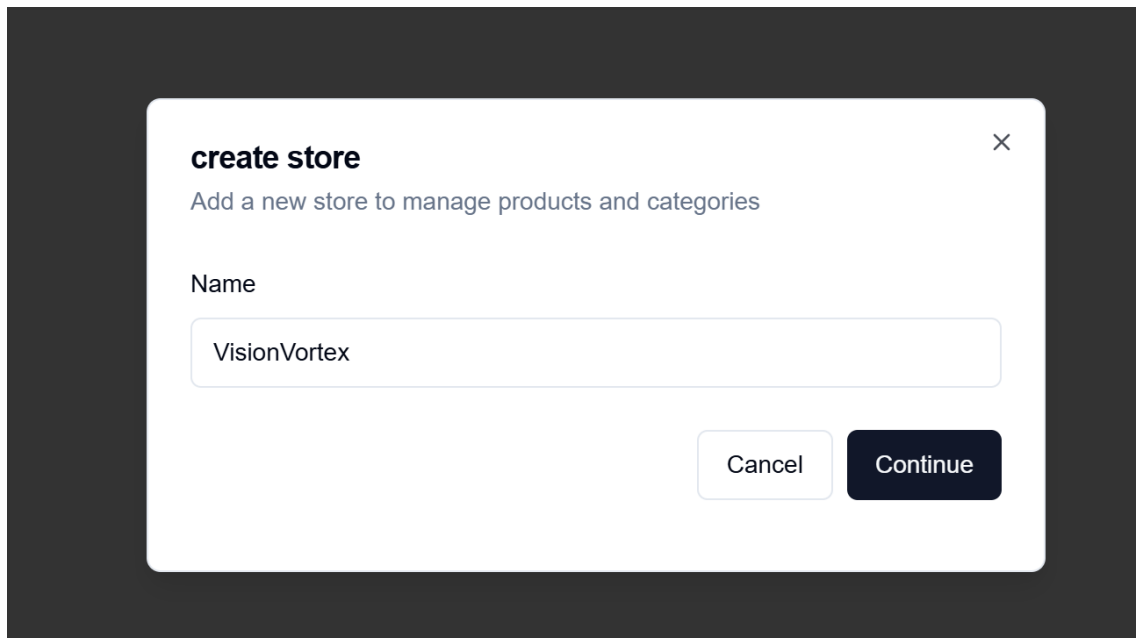
  const onSubmit = async (values: z.infer<typeof formSchema>) => {
    try {
      setLoading(true)
      const response = await axios.post('/api/stores', values)
      console.log(response.data)
    } catch (error) {
      console.log(error)
    } finally{
      setLoading(false)
    }
  }
}
```

Menambahkan property disabled

```
<FormLabel>Name</FormLabel>
<FormControl>
  <Input
    disabled={loading}
    placeholder="E-Commerce"
    {...field}/>
</FormControl>
<FormMessage />
</FormItem>
</Form>

<div className="pt-6 space-x-2 flex items-center justify-end w-full">
  <Button
    disabled={loading}
    variant="outline"
    onClick={storeModal.onClose}>Cancel
  </Button>
  <Button disabled={loading} type="submit">Continue</Button>
</div>
</form>
</Form>
```

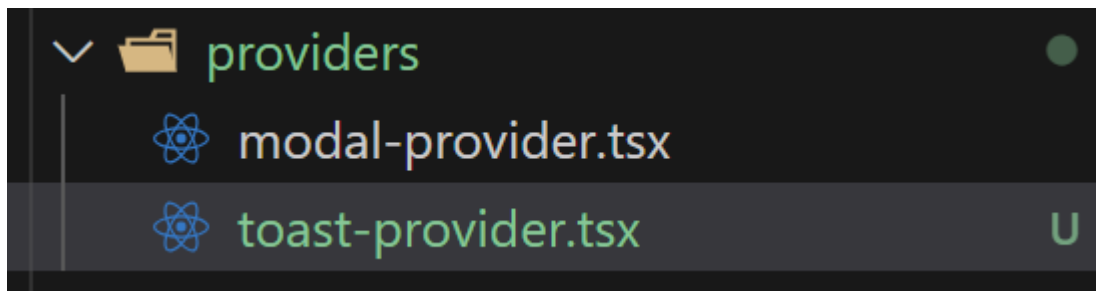
14. tampilan ketika program berhasil dijalankan



15. Menginstal react-hot-toast untuk menambahkan notifikasi toast ke aplikasi React dengan mudah dan intuitif

```
npm install react-hot-toast
```

16. menambahkan file toast-provider.tsx pada folder providers



Menambahkan program pada file toast-provider.tsx

```

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

```

17. Memanggil ToastProvider pada file layout.tsx

```

return (
  <Clerk (property) HTMLAttributes<T>.lang?: string | undefined
    <html lang="en">
      <body className={` ${geistSans.variable} ${geistMono.variable} antialiased`} >
        <ToasterProvider/>
        <ModalProvider/>
        {children}
      </body>
    </html>
  </ClerkProvider>
);
}

```

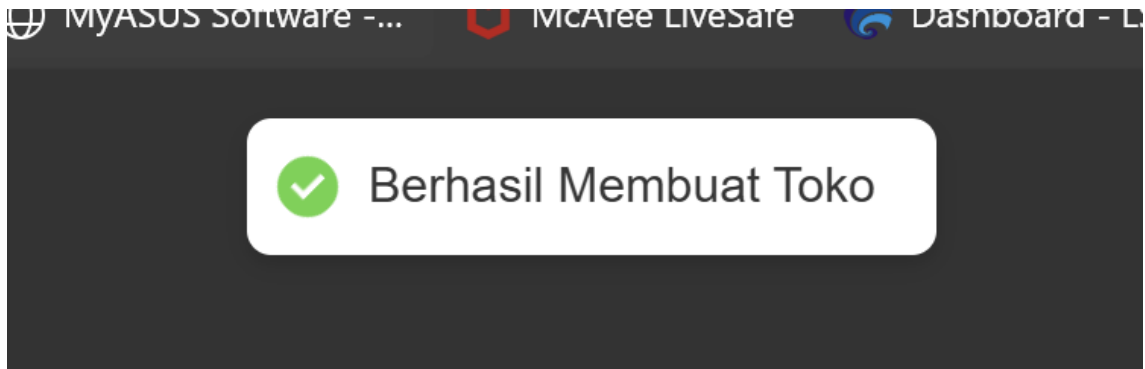
18. Menambahkan program untuk menampilkan notifikasi berhasil dan tidak berhasil menggunakan toast pada file store-modal.tsx

```

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("Gagal Membuat Toko")
  } finally{
    setLoading(false)
  }
}

```

19. Jika berhasil Maka tampilannya akan seperti berikut jika berhasil menambahkan toko:



Data Table (Admin)

1. menambahkan program pada folder billboards file page.tsx

```
import db from "@/lib/db";
import { BillboardClient } from "../components/client";

const BillboardsPage = async ({
  params
}: {
  params: {storeId: string}
}) => {
  const billboards = await db.billboard.findMany({
    where: {
      storeId: params.storeId
    },
    orderBy: {
      createdAt: 'desc'
    }
  })
  return (
    <div className="flex-col">
      <div className="flex-1 space-y-4 p-8 pt-6">
        <BillboardClient data={billboards}/>
      </div>
    </div>
  );
}

export default BillboardsPage;
```

2. menambahkan program pada file client.tsx untuk menampilkan banyak data yang di upload

```

"use client"

import { Plus } from "lucide-react"
import { useParams, useRouter } from "next/navigation"

import { Button } from "@components/ui/button"
import { Heading } from "@components/ui/heading"
import { Separator } from "@components/ui/separator"
import { Billboard } from "@prisma/client"

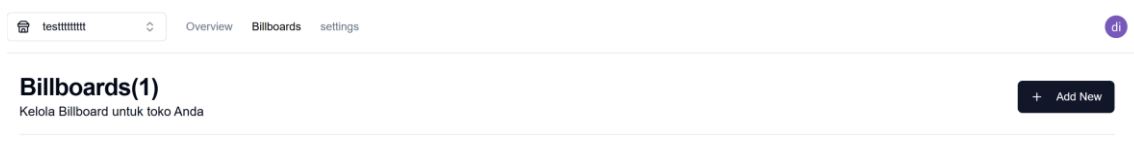
interface BillboardClientProps {
  data: Billboard[]
}

export const BillboardClient: React.FC<BillboardClientProps> = ({
  data
}) => {
  const router = useRouter()
  const params = useParams()

  return (
    <>
      <div className="flex items-center justify-between">
        <Heading
          title={`Billboards(${data.length})`}
          description="Kelola Billboard untuk toko Anda"
        />
        <Button onClick={() => router.push(`/${params.storeId}/billboards/new`)}>
          <Plus className="mr-2 h-4 w-4"/>
          Add New
        </Button>
      </div>
      <Separator />
    </>
  )
}

```

Tampilan ketika program dijalankan



3. menginstall components data table dari shadcnui.com

```
> npx shadcn@latest add table
```

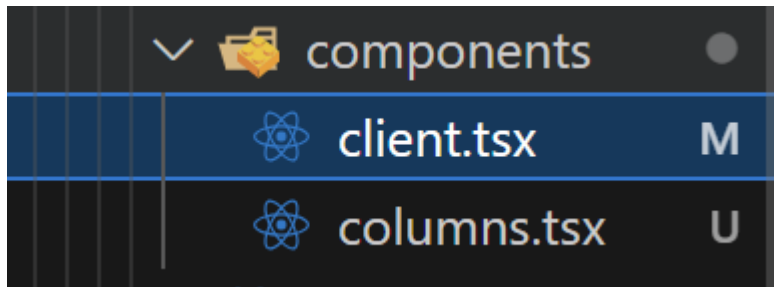
4. menginstal package tanstack

```
npm install @tanstack/react-table
```

5. menginstall package date

```
npm i date-fns
```

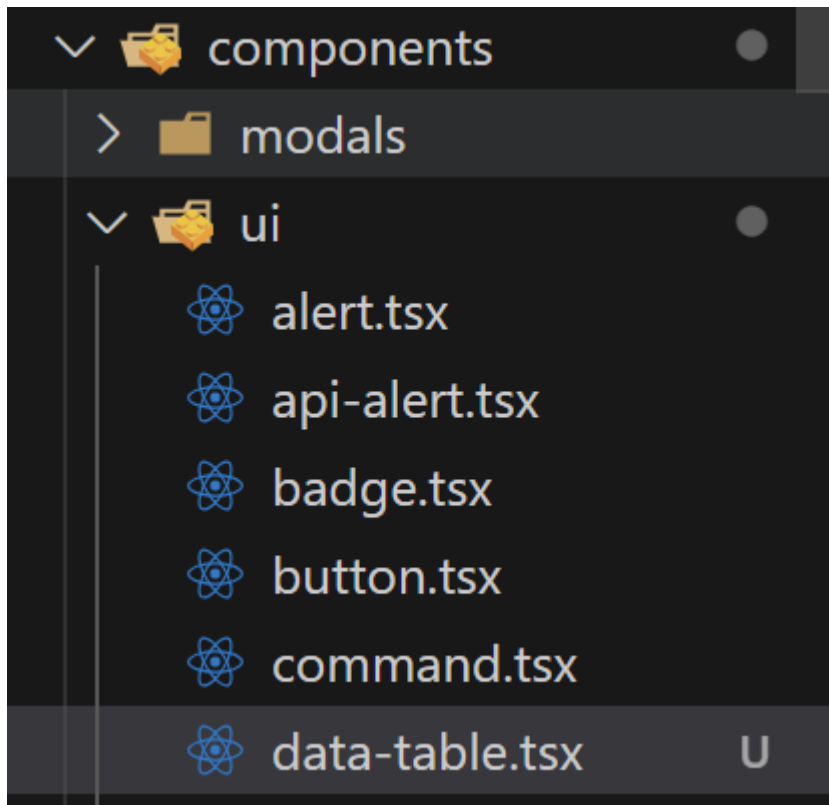
6. membuat file column.tsx pada folder components billboards



7. menambahkan program untuk membuat column pada file column.tsx

```
1  "use client"
2
3  import { ColumnDef } from "@tanstack/react-table"
4
5  // This type is used to define the shape of our data.
6  // You can use a Zod schema here if you want.
7  export type BillboardColumn = {
8    id: string
9    label: string
10   createdAt: string
11 }
12
13 export const columns: ColumnDef<BillboardColumn>[] = [
14   {
15     accessorKey: "label",
16     header: "Label",
17   },
18   {
19     accessorKey: "createdAt",
20     header: "Date",
21   },
22 ]
```

8. membuat file data-table.tsx pada folder components/ui



9. menambhkn program pada file data-table.tsx

```
"use client"

import { ... }

interface DataTableProps<TData, TValue> {
  columns: ColumnDef<TData, TValue>[]
  data: TData[]
}

export function DataTable<TData, TValue>({
  columns,
  data,
}: DataTableProps<TData, TValue>) {
  const table = useReactTable({
    data,
    columns,
    getCoreRowModel: getCoreRowModel(),
  })

  return (
    <div className="rounded-md border">
      <Table>
        <TableHeader>
          {table.getHeaderGroups().map((headerGroup) => (
            <TableRow key={headerGroup.id}>
              {headerGroup.headers.map((header) => {
                return (
                  <TableHead key={header.id}>
                    {header.isPlaceholder
                      ? null
                      : flexRender(
                        header.column.columnDef.header,
                        header.getContext()
                      )}
                  </TableHead>
                )
              })}
            </TableRow>
          ))}
        </TableHeader>
        <TableBody>
          {table.getRowModel().rows.map((row) => (
            <TableRow key={row.id}>
              {row.getVisibleCells().map((cell) => (
                <TableBodyCell>
                  {flexRender(
                    cell.column.columnDef.cell,
                    cell.getContext()
                  )}
                </TableBodyCell>
              ))}
            </TableRow>
          ))}
        </TableBody>
      </Table>
    </div>
  )
}
```

10. menambahkan program dalam file data-table.tsx untuk menampilkan pagination pada table

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

```
import {
  ColumnDef,
  flexRender,
  getCoreRowModel,
  getPaginationRowModel,
  useReactTable,
} from "@tanstack/react-table"
```

```
export function DataTable<TData, TValue>({
  columns,
  data,
}: DataTableProps<TData, TValue>) {
  const table = useReactTable({
    data,
    columns,
    getCoreRowModel: getCoreRowModel(),
    getPaginationRowModel: getPaginationRowModel(),
  })
}
```

```
</div>
<div className="flex items-center justify-end space-x-2 py-4">
  <Button
    variant="outline"
    size="sm"
    onClick={() => table.previousPage()}
    disabled={!table.getCanPreviousPage()}
  >
    Previous
  </Button>
  <Button
    variant="outline"
    size="sm"
    onClick={() => table.nextPage()}
    disabled={!table.getCanNextPage()}
  >
    Next
  </Button>
</div>
```

Tampilan ketika program dijalankan

testtttttt Overview Billboards settings

Billboards(2)

Kelola Billboard untuk toko Anda

+ Add New

Label	Date
TEST2	January 13th, 2025
TEST	January 13th, 2025

Previous

Next

11. Menambahkan program di file data-table.tsx untuk menampilkan filter

Mengimport input dari components/ui/input

```
import { Input } from "@/components/ui/input"
```

Mengimport ColumnFilterState, dan getFilteredRowModel dari shadcnui.com

```
5 import {
6   ColumnDef,
7   ColumnFiltersState,
8   flexRender,
9   getCoreRowModel,
10  getFilteredRowModel,
11  getPaginationRowModel,
12  useReactTable,
13 } from "@tanstack/react-table"
14
```

```
import { useState } from "react"
```

Membuat fungsi searchkey untuk searching

```
data: TData[]
searchKey: string;
```

membuat fungsi columnFilters dan setColumnFilters menggunakan usestate

```
33 data,
34 searchKey,
35 ): DataTableProps<TData, TValue> {
36   const [columnFilters, setColumnFilters] = useState<ColumnFiltersState>(
37     []
38   )
39
```

memanggil columnFilters pada table

```
state: {
  columnFilters,
},
```

Membuat tampilan dan fungsi filter

```
<div className="flex items-center py-4">
  <Input
    placeholder="Search"
    value={{(table.getColumn(searchKey)?.getFilterValue() as string) ?? ""}}
    onChange={(event) =>
      table.getColumn(searchKey)?.setFilterValue(event.target.value)
    }
    className="max-w-sm"
  />
</div>
```

Memanggil fungsi searchKey pada client.tsx

```
<Separator />
<DataTable searchKey="label" columns={columns} data={data}/>
</>
```

Tampilan ketika program dijalankan

testtttttt Overview Billboards settings

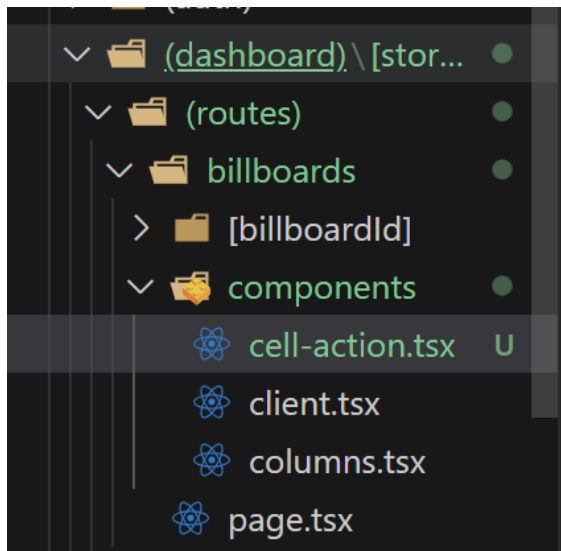
Billboards(3) Kelola Billboard untuk toko Anda + Add New

3

Label	Date
Test3	January 13th, 2025

Previous Next

Membuat file baru yang bernama cell-action.tsx pada folder billboards/components



Menambahkan program pada file cell-action yang berupa fungsi CellAction yang didalamnya terdapat div Action

```
app > (dashboard) > [storeId] > (routes) > billboards > components > cell-action.tsx > ...
1  export const CellAction = () => {
2    return(
3      <div>Action</div>
4    );
5  };
```

Menambahkan program untuk memanggil fungsi CellAction yang tadi dibuat pada file column.tsx

```
{
  id: "Actions",
  cell: () => <CellAction/>
}
```

Tampilan ketika program dijalankan

Billboards(3)

Kelola Billboard untuk toko Anda

+ Add New

Label	Date	
Test3	January 13th, 2025	Action
TEST2	January 13th, 2025	Action
TEST	January 13th, 2025	Action

Menginstal components dropdown menu dari shadcn ui

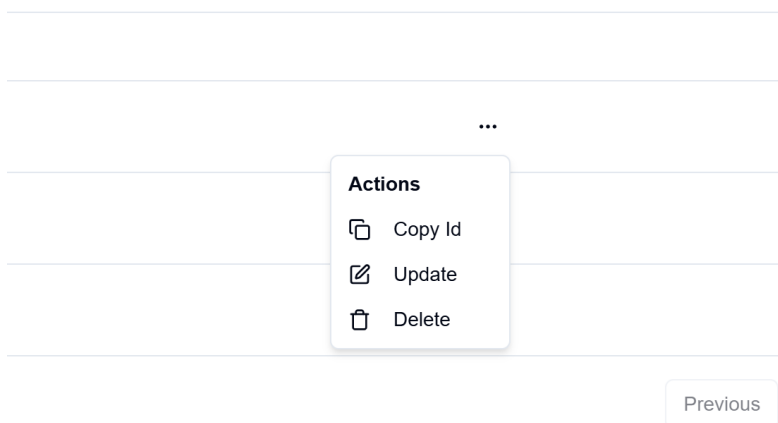
```
npx shadcn@latest add dropdown-menu
```

Mengedit Program Action dengan menambahkan program untuk membuat Dropdown yang telah diinstal diatas

```
return(  
  <DropdownMenu>  
    <DropdownMenuTrigger asChild>  
      <Button variant="ghost" className="h-8 w-8 p-0">  
        <span className="sr-only"> Open Menu</span>  
        <MoreHorizontal className="h-4 w-4"/>  
      </Button>  
    </DropdownMenuTrigger>  
    <DropdownMenuContent align="end">  
      <DropdownMenuLabel>  
        Actions  
      </DropdownMenuLabel>  
      <DropdownMenuItem>  
        <Copy className="mr-2 h-4 w-4"/>  
        Copy Id  
      </DropdownMenuItem>  
      <DropdownMenuItem>  
        <Edit className="mr-2 h-4 w-4"/>  
        Update  
      </DropdownMenuItem>  
      <DropdownMenuItem>  
        <Trash className="mr-2 h-4 w-4"/>  
        Delete  
      </DropdownMenuItem>  
    </DropdownMenuContent>  
  </DropdownMenu>  

```

Tampilan dropdown ketika program dijalankan



Menambahkan program Fungsi copy pada file cell-action.tsx

```
}) => {  
  const onCopy = (id: string) => {  
    navigator.clipboard.writeText(id);  
    toast.success("Billboard Id copied to the clipboard")  
  };  
}
```

Lalu memanggilnya pada dropdown bagian copy

```
<DropdownMenuItem onClick={() => onCopy(data.id)}>  
  <Copy className="mr-2 h-4 w-4"/>  
  Copy Id  
</DropdownMenuItem>
```

Untuk Membuat fungsi update

Membuat fungsi router yang diimport dari fungsi useRouter dan membuat fungsi params yang diimport dari fungsi useParams

```
const router = useRouter();  
const params = useParams();
```

```
import { useParams, useRouter } from "next/navigation";
```

Tambahkan fungsi onClick pada bagian update data dengan mengambil data/push dari database selanjutnya edit pada halaman billboards

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

Membuat program fungsi delete yang akan digunakan untuk menghapus data pada tabel

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

```

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

```

Lalu pada bagian return memanggil fungsi AlertModal dengan property isOpen,onClose,onConfirm dan loading.

```

<AlertModal
  isOpen={Open}
  onClose={() => setOpen(false)}
  onConfirm={onDelete}
  loading={loading}
/>

```

Setelah itu pada dropdown delete tambahkan fungsi onClick dengan memanggil setOpen yang nilainya true

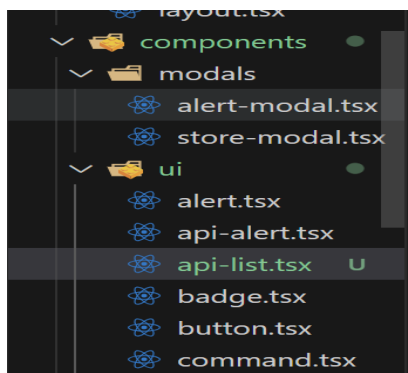
```

<DropdownMenuItem onClick={() => setOpen(true)}>
  <Trash className="mr-2 h-4 w-4"/>
  Delete
</DropdownMenuItem>

```

Setelah program dijalankan maka data pada tabel sudah bisa dihapus.

Membuat file baru yang bernama api-list.tsx pada folder componnets/ui




```
interface ApiListProps{
  entityName: string;
  entityIdName: string;
}
```

Interface ini mendefinisikan properti yang harus diterima oleh komponen `ApiList`.

- **entityName:** Nama entitas yang menjadi target endpoint API (contoh: `products`, `users`, dll.).
- **entityIdName:** Nama parameter ID yang digunakan untuk mengidentifikasi entitas tertentu (contoh: `productId`, `userId`, dll.).

```
export const ApiList: React.FC<ApiListProps> = ({
  entityName,
  entityIdName
}) => {

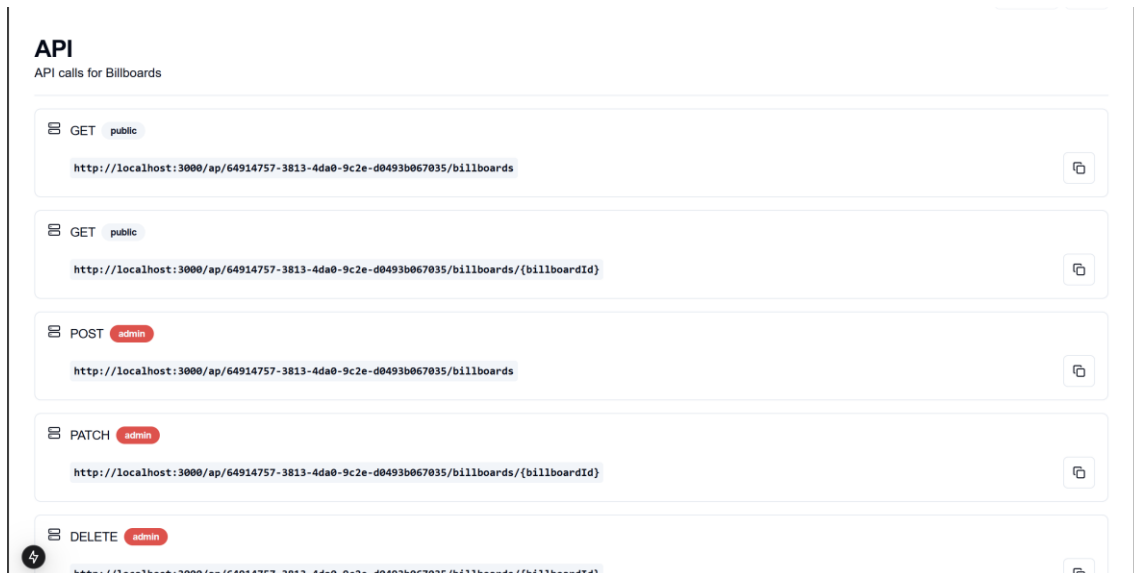
  const params = useParams();
  const origin = useOrigin();

  const baseUrl = `${origin}/ap/${params.storeId}`;
```

- **params:** Mengambil parameter URL, seperti `storeId` dari path dinamis `store/[storeId]`.
- **origin:** Mengambil URL dasar aplikasi
- **baseUrl:** Membentuk URL dasar API berdasarkan `origin` dan `storeId`,

```
return(
  <>
    <ApiAlert
      title="GET"
      variant="public"
      description={` ${baseUrl}/${entityName}`}
    />
```

- **title:** Menunjukkan metode HTTP yang digunakan untuk endpoint (GET, POST, PATCH, DELETE).
- **variant:** Menentukan aksesibilitas endpoint:
 - `public:` Dapat diakses oleh semua pengguna.
 - `admin:` Hanya dapat diakses oleh admin.
- **description:** URL endpoint API `baseUrl/entityname`



Tampilan ketika program dijalankan.

SIZE Entity(ADMIN)

```

54
55 model Size{
56   id String @id @default(uuid())
57   storeId String
58   store Store @relation("StoreToSize", fields: [storeId], references: [id])
59   name String
60   value String
61   createdAt DateTime @default(now())
62   updatedAt DateTime @updatedAt
63
64   @@index([storeId])
65 }

```

Membuat Model schema Prisma yang mendefinisikan tabel **Size** untuk menyimpan informasi ukuran produk dengan penjelasan berikut:

- **id**: Primary key dengan UUID unik.
- **storeId**: Foreign key yang menghubungkan Size ke model Store.
- **Relasi (store)**: Menghubungkan model Size ke Store menggunakan storeId.
- **name & value**: Menyimpan nama dan nilai ukuran.
- **createdAt & updatedAt**: Menyimpan waktu pembuatan dan pembaruan data secara otomatis.
- **@@index([storeId])**: Membuat indeks pada kolom storeId untuk meningkatkan performa query.

