Nama : Muhamad Dimas Stiyawan

NPM : 22312087

Environtment 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.
```

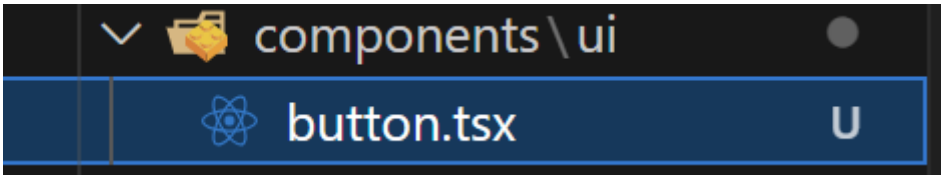
2. menginstall shadcn ui

```
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.
```

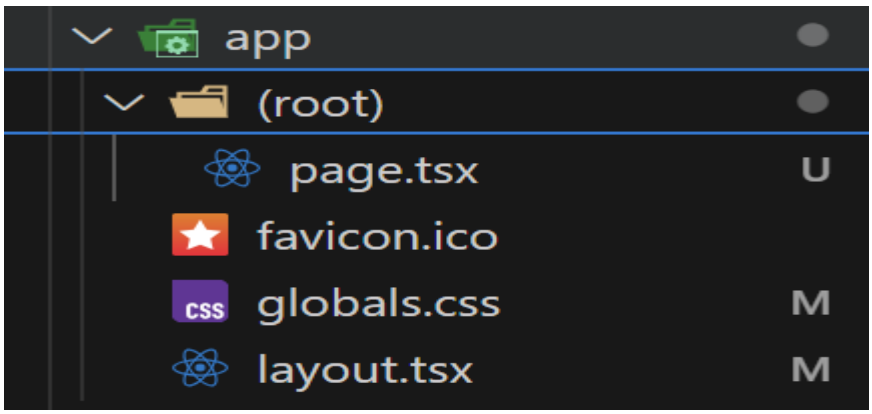
- jika sudah akan muncul folder components yg berisi file button.tsx



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

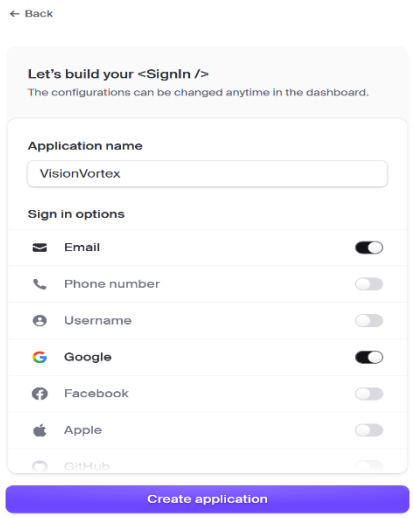
```
ecommerce-visionvortex > app > page.tsx > ...
1  import { Button } from "@/components/ui/button";
2  import Image from "next/image";
3
4  export default function Home() {
5    return (
6      <div className="p-4">
7          <Button>Click Me</Button>
8      </div>
9
10   );
11 }
```

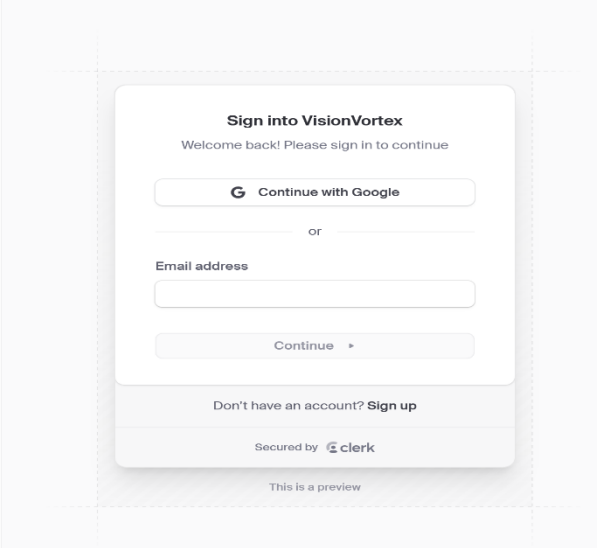
4. membuat folder (root) pada folder app dan memindahkan file page.tsx ke dalam folder (root)



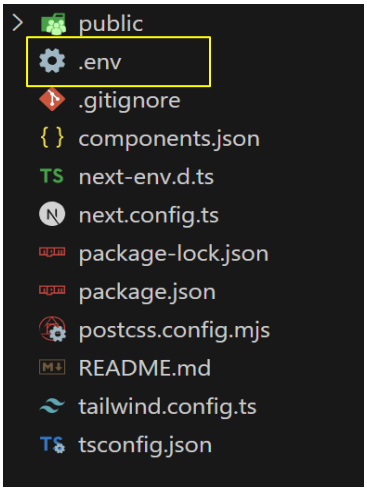
Clerk Authentication (Admin)

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

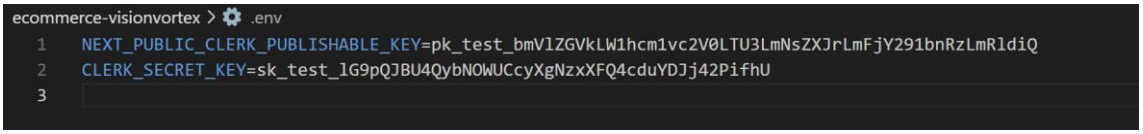




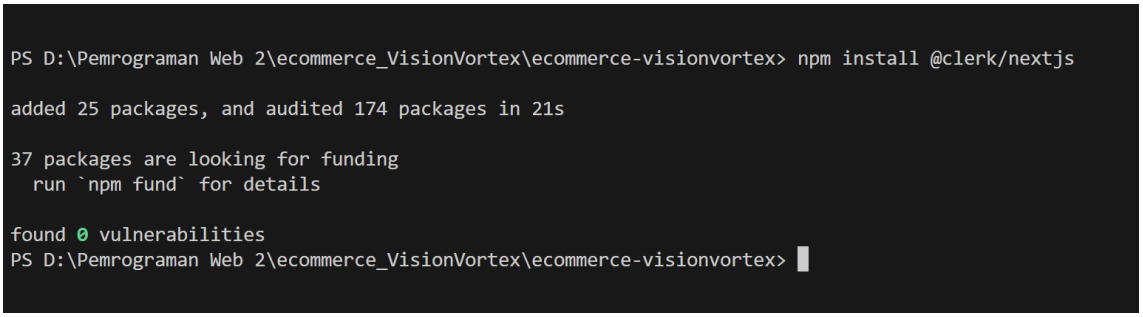
2. buat file .env pada project



3. memasukkan API Keys clerk ke dalam file .env



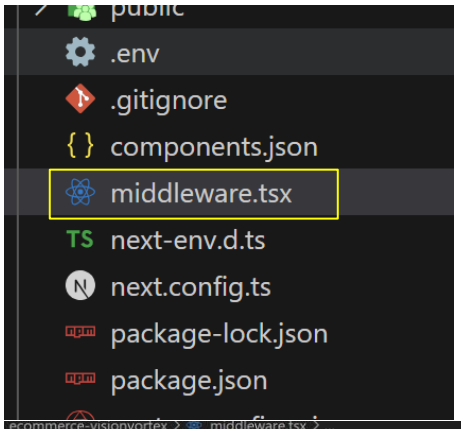
4. menginstal clerk pada project



5. mengimport clerkProvider pada file layout.tsx

```tsx
import type { Metadata } from "next";
import { Geist, Geist_Mono } from "next/font/google";
import { ClerkProvider } from "@clerk/nextjs";
import "./globals.css";

const geistSans = Geist({ ···
});

const geistMono = Geist_Mono({ ···
});

export const metadata: Metadata = {
  title: "Admin Dashboard",
  description: "Admin Dashboard",
};

export default function RootLayout({
  children,
}: Readonly<{
  children: React.ReactNode;
}>) {
  return (
    <ClerkProvider>
      <html lang="en">
        <body
          className={`${geistSans.variable} ${geistMono.variable} antialiased`}
        >
          {children}
        </body>
      </html>
    </ClerkProvider>
  );
}
```

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

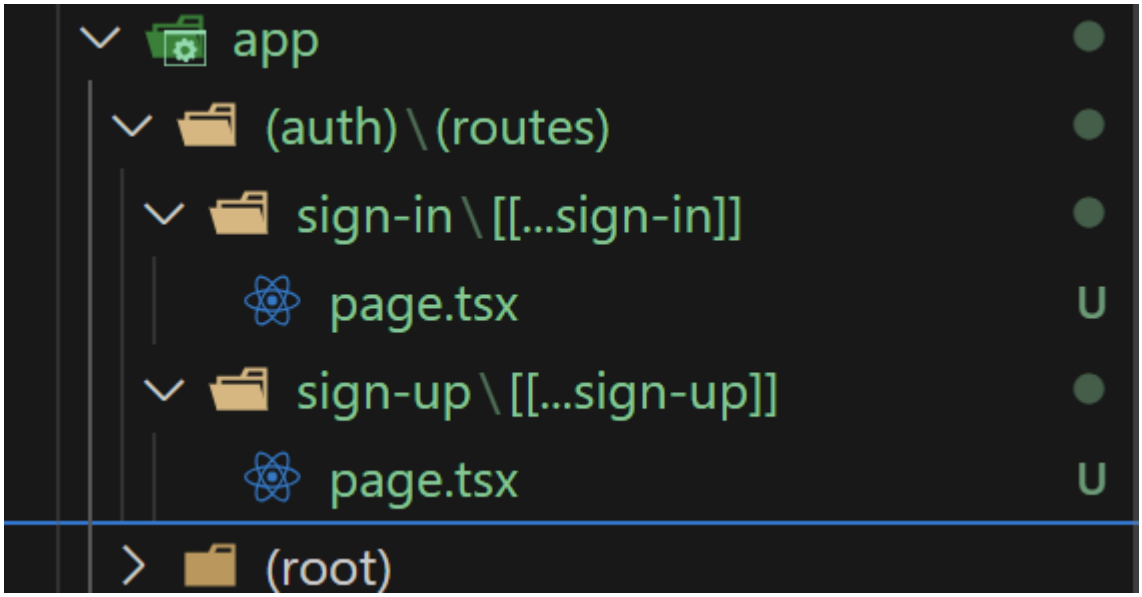


ecommerce-visionvortex > middleware.tsx > ...

```tsx
import { clerkMiddleware } from "@clerk/nextjs/server";

export default clerkMiddleware();

export const config = {
  matcher: [
    // Skip Next.js internals and all static files, unless found in search params
    '/((?!_next|[^?]*\\.(?:html?|css|js(?!on)|jpe?g|webp|png|gif|svg|ttf|woff2?|ico|csv|docx?|xlsx?|zip|webmanifest)).*)',
    // Always run for API routes
    '/(api|trpc)(.*)',
  ],
};
```

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

```tsx
import { SignIn } from '@clerk/nextjs'

export default function Page() {
  return <SignIn />
}
```

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

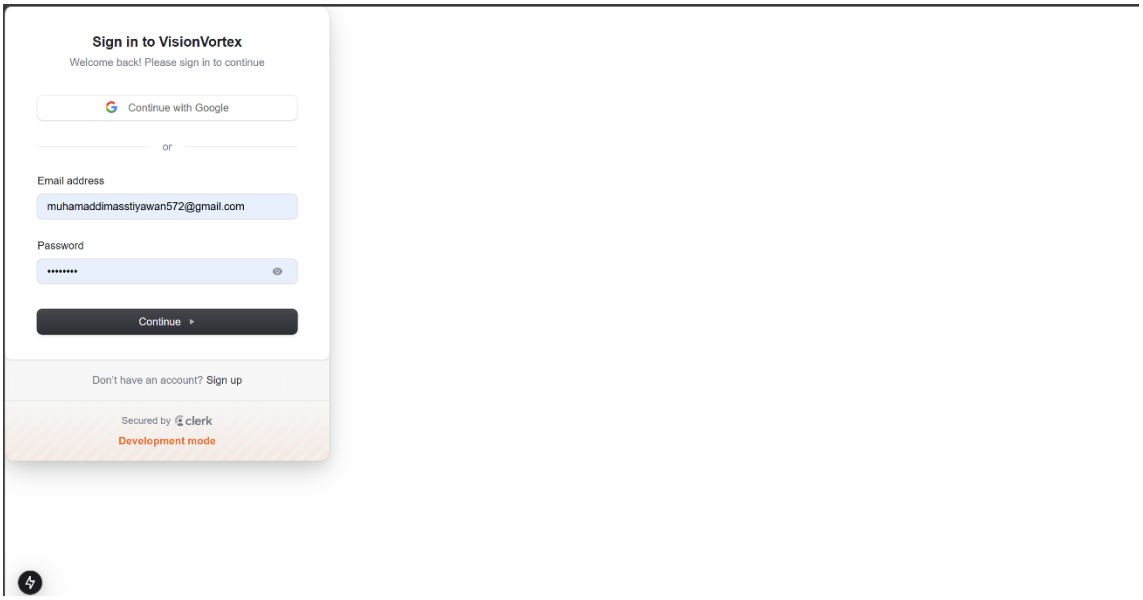
ecommerce-visionvortex > app > (auth) > (routes) > sign-up > [[...sign-up]] > page.tsx > Page

```tsx
import { SignUp } from '@clerk/nextjs'

export default function Page() {
  return <SignUp />
}
```

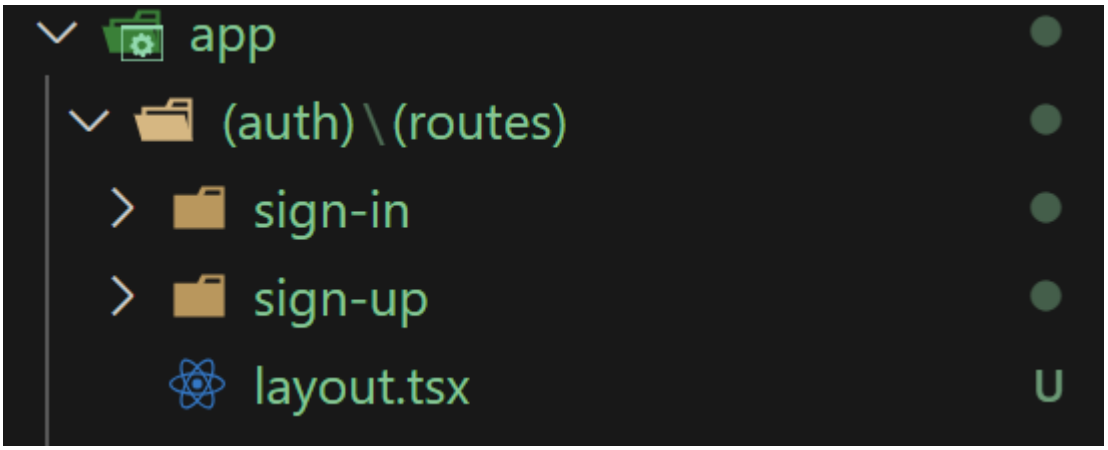
4. menambahkan program pada file .env

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

5. tampilan setelah program dijalankan



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

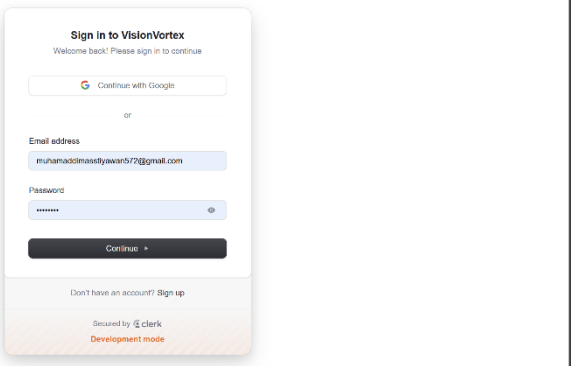


7. menambahkan program pada file layout.tsx untuk mengubah layout sign-in dan sign-up

ecommerce-visionvortex > app > (auth) > (routes) > layout.tsx > AuthLayout

```tsx
export default  function AuthLayout({
    children
}: {
    children: React.ReactNode
}) {
    return (
        <div className="flex items-center justify-center min-h-screen ">{children}</div>
    )
}
```

8. tampilan setelah program dijalankan



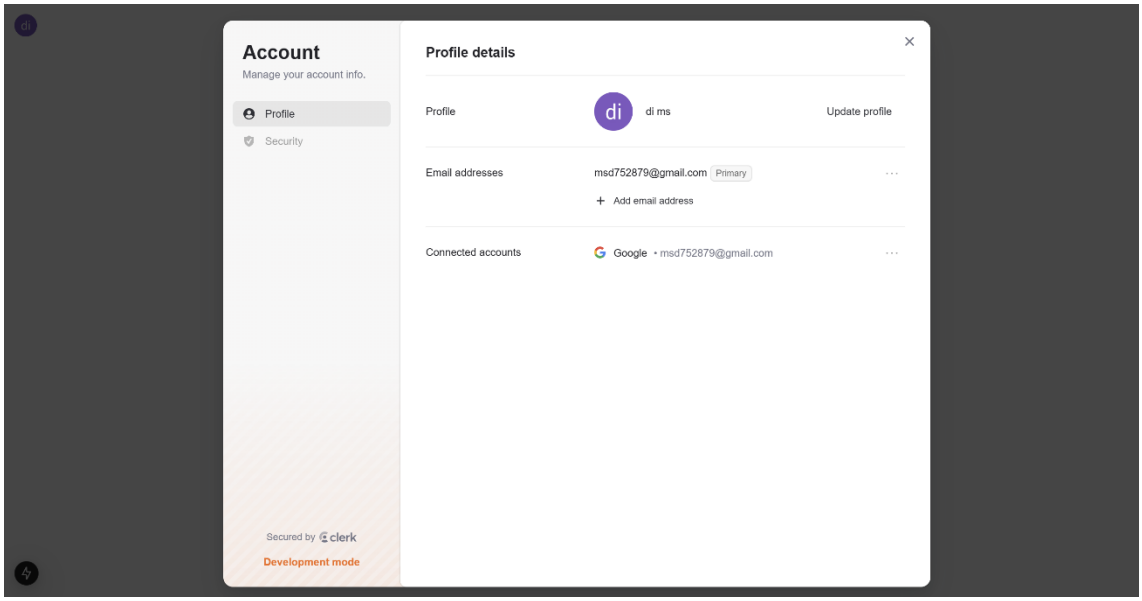
9. tampilan setelah sign-in menggunakan google

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

ecommerce-visionvortex > app > (root) > page.tsx > ...

```tsx
import { Button } from "@/components/ui/button";
import { UserButton } from "@clerk/nextjs";
import Image from "next/image";

const setUpPage = () => {
  return (
    <div className="p-4">
        <UserButton></UserButton>
    </div>

  );
}

export default setUpPage
```

11. tampilan setelah program dijalankan

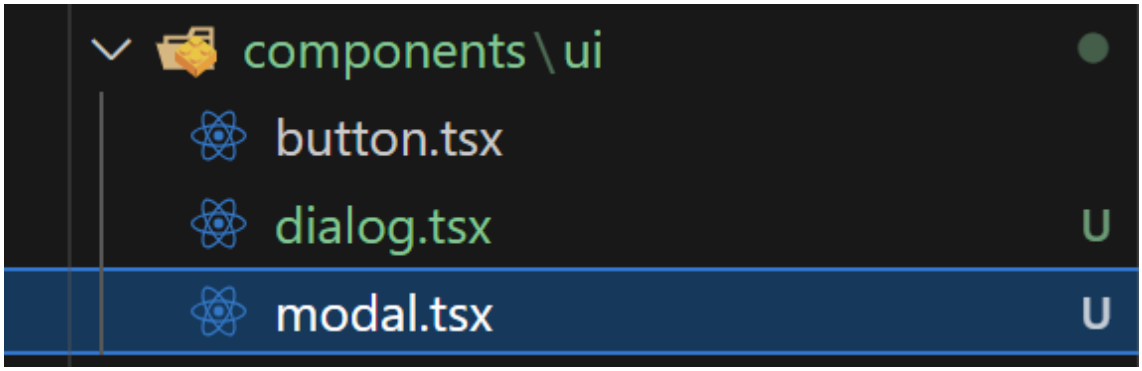


Modal Components

1. menginstal ui dialog dari Shadcnui.com

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

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



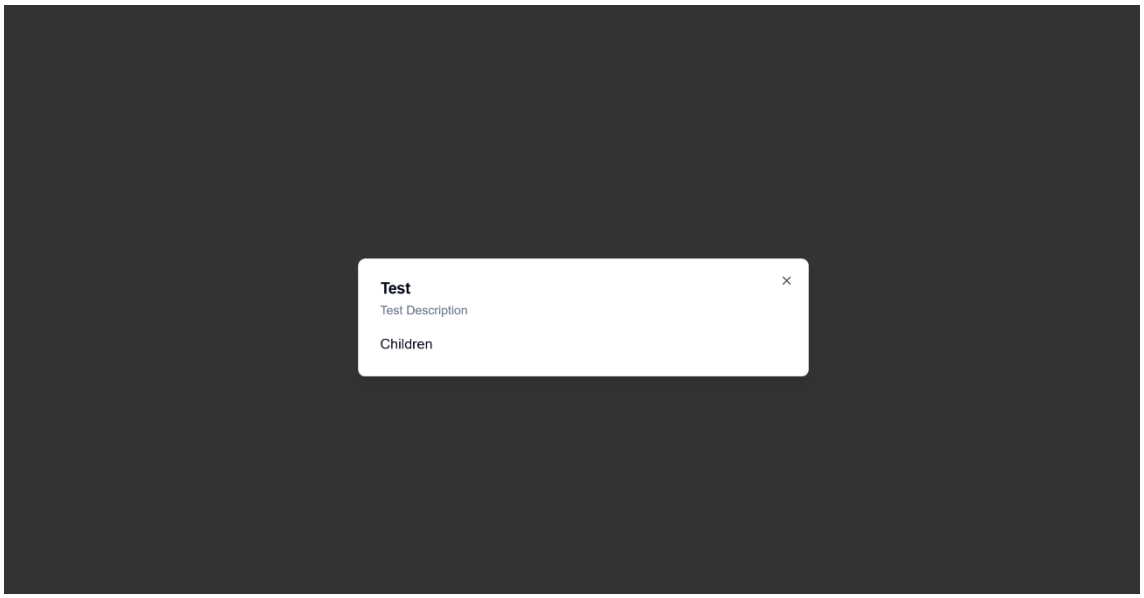
3. membuat program untum modal dialog pada file modal.tsx

```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)

```tsx
"use client"
// Mengimpor komponen `Modal` dari path `@/components/ui/modal`.
import { Modal } from "@/components/ui/modal";


// Fungsi `setUpPage` adalah komponen fungsional React yang digunakan untuk merender halaman.
const setUpPage = () => {
  return (
    <div className="p-4">
       <Modal title="Test" description="Test Description" isOpen onClose={() => {}}>
            Children
        </Modal>
    </div>

  );
}

// Mengekspor `setUpPage` sebagai default, sehingga komponen ini dapat digunakan di file lain.
export default setUpPage

```

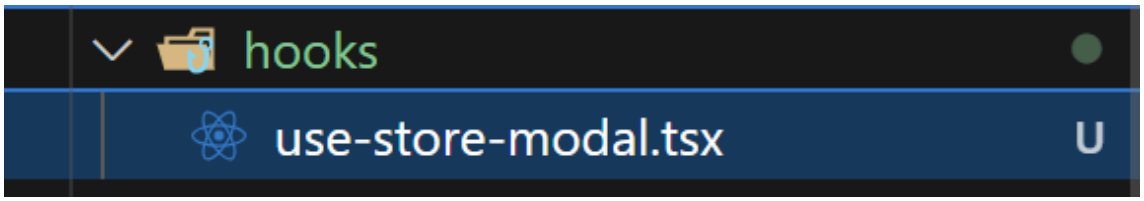
tampilan ketika program dijalankan



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

```
npm install zustand
```

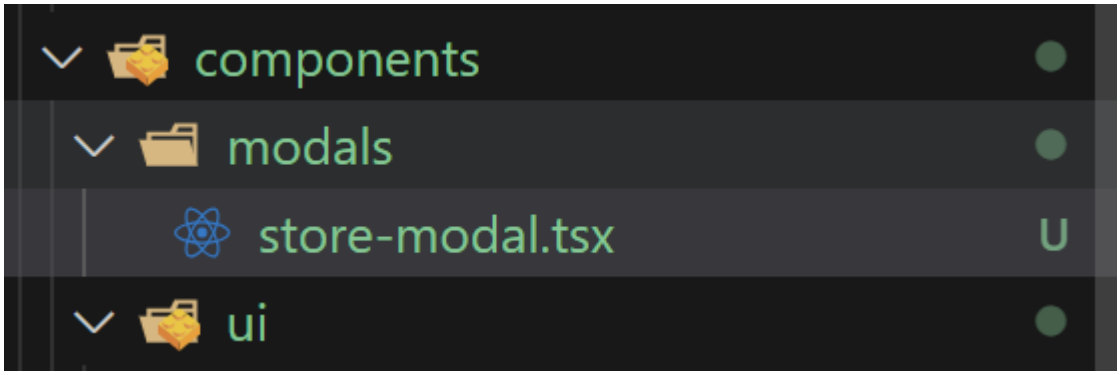
6. membuat folder hooks dalam project kemudian membuat file use-store-modal.tsx dalam folder hooks



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

```tsx
// Mengimpor fungsi `create` dari pustaka Zustand untuk membuat store global pada aplikasi React.
import { create } from "zustand";

// Mendefinisikan tipe data untuk store yang mengelola status modal.
interface useStoreModalStore{
    isOpen: boolean;
    onOpen: () => void;
    onClose: () => void;
};

// Membuat store dengan Zustand dan mendefinisikan status dan fungsi pengelolaan modal.
export  const useStoreModal = create<useStoreModalStore>((set) => ({
    isOpen: false,
    onOpen: () => set({isOpen: true}),
    onClose: () => set({isOpen: false}),
}));
```

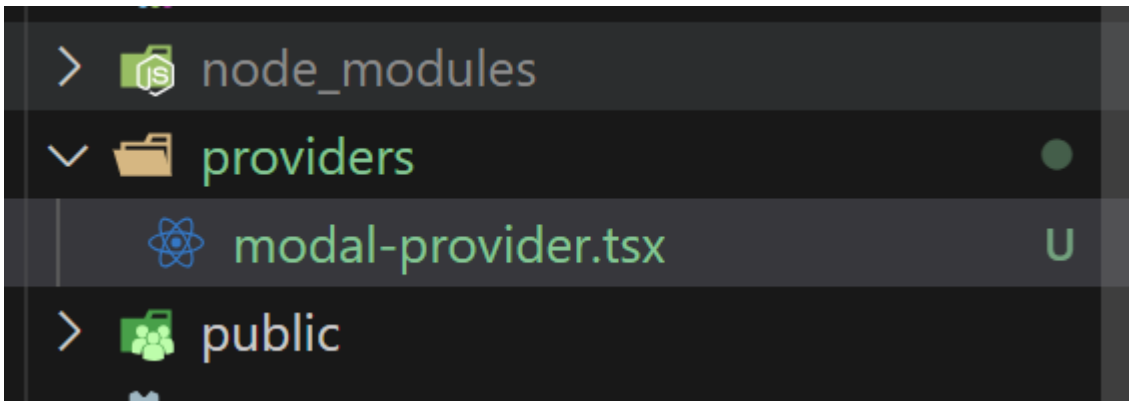
8. membuat folder modals dalam folder components, kemudian membuat file store-modal.tsx dalam folder modals



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`.**

```tsx
"use client"

// Mengimpor hook `useStoreModal` yang digunakan untuk mengakses state global modal dari Zustand.
import { useStoreModal } from "@/hooks/use-store-modal"
// Mengimpor komponen `Modal` dari path lokal. Komponen ini bertugas untuk menampilkan dialog/modal.
import { Modal } from "../ui/modal";

// Mendefinisikan komponen fungsional React bernama `StoreModal`.
 export const StoreModal = () => {
    // Mengakses state dan fungsi dari store modal menggunakan hook `useStoreModal`.
    const storeModal = useStoreModal();

    return(
        <Modal
        title="create store"
        description="Add a new store to manage products and categories"
        isOpen={storeModal.isOpen}
        onClose={storeModal.onClose}
        >
            future Create Sore Form
            {/* Konten modal yang akan muncul di dalamnya. Saat ini berupa placeholder teks,
                tetapi akan digantikan dengan form atau elemen lain di masa mendatang. */}
        </Modal>
    )
}
```

10. membuat folder providers kemudian didalmnya 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.

```tsx
"use client"
// Mengimpor komponen `StoreModal` yang bertugas untuk menampilkan modal untuk menambahkan toko.
import { StoreModal } from "@/components/modals/store-modal";
// Mengimpor hook `useEffect` dan `useState` dari React untuk mengelola efek samping dan state lokal.
import { useEffect, useState } from "react"

// `ModalProvider` yang bertugas mengelola render modal di sisi klien.
export const ModalProvider = () =>{
    // State `isMounted` untuk mengecek apakah komponen sudah ter-mount.
    const [isMounted, setIsMounted] = useState(false);

    useEffect(() =>{
        // Mengubah `isMounted` menjadi `true` setelah komponen ter-mount di klien.
        setIsMounted(true)
    }, []);

    if(!isMounted){
        // Tidak merender komponen hingga ter-mount di klien.
        return null;
    }

    return(
        <>
        {/* Render `StoreModal` setelah komponen ter-mount. */}
            <StoreModal/>
        </>
    );
};
```

12. memanggil ModalProvider pada halaman layout.tsx

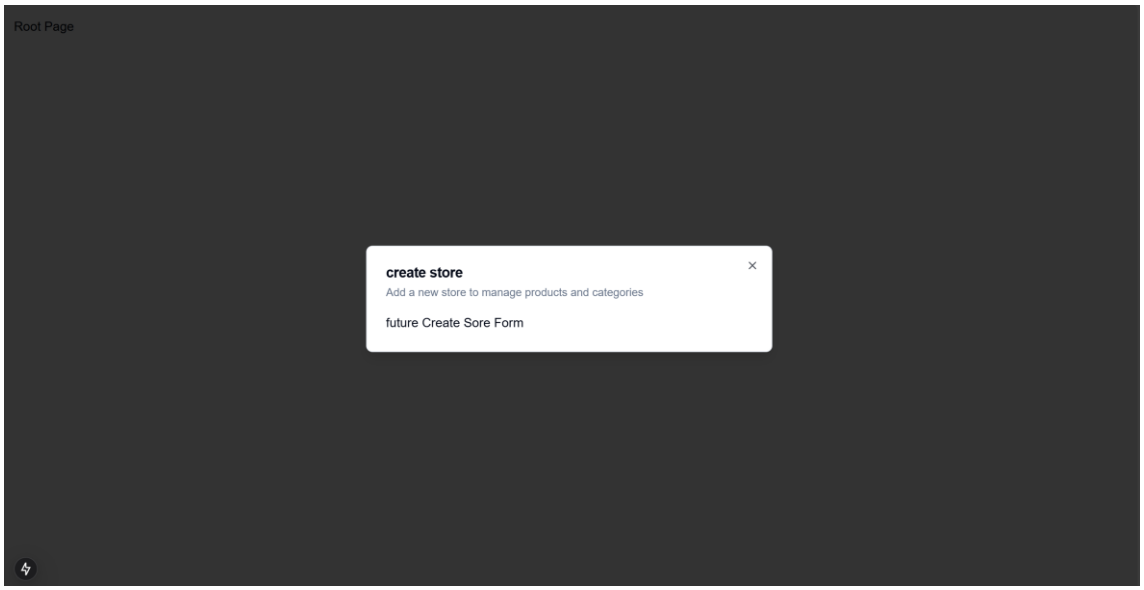
```tsx
import type { Metadata } from "next";
import { Geist, Geist_Mono } from "next/font/google";
import { ClerkProvider } from "@clerk/nextjs";

import { ModalProvider } from "@/providers/modal-provider";
import "./globals.css";


const geistSans = Geist({ ···
});

const geistMono = Geist_Mono({ ···
});

export const metadata: Metadata = {
  title: "Admin Dashboard",
  description: "Admin Dashboard",
};

export default function RootLayout({
  children,
}: Readonly<{
  children: React.ReactNode;
}>) {
  return (
    <ClerkProvider>
    <html lang="en">
      <body className={`${geistSans.variable} ${geistMono.variable} antialiased`}>
        <ModalProvider/>
        {children}
      </body>
    </html>
    </ClerkProvider>
  );
}
```

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

ecommerce-visionvortex > app > (root) > page.tsx > setUpPage

```tsx
"use client"

// Mengimpor hook `useStoreModal` untuk mengakses store modal.
import { useStoreModal } from "@/hooks/use-store-modal";

import { useEffect } from "react";


// Fungsi `setUpPage` adalah komponen fungsional React yang digunakan untuk merender halaman.
const setUpPage = () => {

  // Mengambil fungsi `onOpen` dan status `isOpen` dari store `useStoreModal`.
  const onOpen = useStoreModal((state) => state.onOpen);
  const isOpen = useStoreModal((state) => state.isOpen);

  // Menggunakan `useEffect` untuk membuka modal jika `isOpen` bernilai false.
  useEffect(() => {
    if(!isOpen){
      onOpen();
    }
  }, [isOpen,onOpen]);


  return (
    <div className="p-4">
      {/* Konten dari halaman, menampilkan teks "Root Page". */}
      Root Page
    </div>

  );
}

// Mengekspor `setUpPage` sebagai default, sehingga komponen ini dapat digunakan di file lain.
export default setUpPage
```

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