

Next Auth

`npx create-next-app@latest`

Add Shadcn ui

`npx shadcn-ui@latest init`

Add Shadcn button component

`npx shadcn-ui@latest add button`

In Routing in NextJs

(auth) this folder name won't be added in routing and we can have separate layout

`_component` : this folder will be always skipped from routing and wont be included in routing even if we start with `page.tsx`

By default server component

Server component can be turned into asynchronous, we can actually call the database inside the server components and return it to the client component.

```
You, 54 seconds ago | 1 author (You)
1  import { Button } from "@/components/ui/button";
2  import { useEffect } from "react";
3
4  export default async function Home() {
5    const users = await db.collection("users").get();    Cannot find name 'db'.
6
7    return (
8      <ClientComponent data={users} />    'ClientComponent' is not defined.
9    )
10
11    return (
12      <Button size="lg">
13        Click me
14      </Button>
15    )
16  }
```

In Global.css modify the file

```
html,
body,
:root{
  height: 100%;
}
```

Home page

Let's work on basic UI of the Landing page

Task to do:

- Add Poppins font from google font, choose weights
- Give a Heading and a paragraph and a Sign in button

```
// adding font
import { Poppins } from "next/font/google";


import {cn} from '@lib/utils'
// now we can utilize existing class with the font

import {Button} from "@components/ui/button"

const font = Poppins({
  subsets:['latin'],
  weight:['600']
})

export default function Home() {
  return (
    <main className="flex h-full flex-col items-center justify-center
bg-[radial-gradient(ellipse_at_top,_var(--tw-gradient-stops))]
from-sky-400 to-blue-800">
      <div className="space-y-6 text-center">
        <h1 className={cn(
          "text-6xl font-semibold text-white drop-shadow-md",
```

```

        font.className,
      ) }>
       Auth
    </h1>
    <p className="text-white text-lg">Simple Authentication Service</p>
    <div>
      <Button variant="secondary" size="lg">
        Sign in
      </Button>
    </div>
  </div>
</main>
);
}

```

Create a folder inside components folder naming auth/login-button.tsx

We will be creating LoginButton, a Auth component

```

'use client'
// it will have interactive interfaces

interface LoginButtonProps{
  children:React.ReactNode;
  mode?:"modal"|"redirect";
  asChild?:boolean;
}

// Auth Components called LoginButton
// if user doesn't pass anything then it will redirect
export const LoginButton = ({
  children,
  mode="redirect",
  asChild
}:LoginButtonProps) =>{
  const onClick = () =>{
    console.log("LOGIN BUTTON CLICKED")
  }
}

```

```

return (
  <span onClick={onClick} className="cursor-pointer">
    {children}
  </span>
)
}

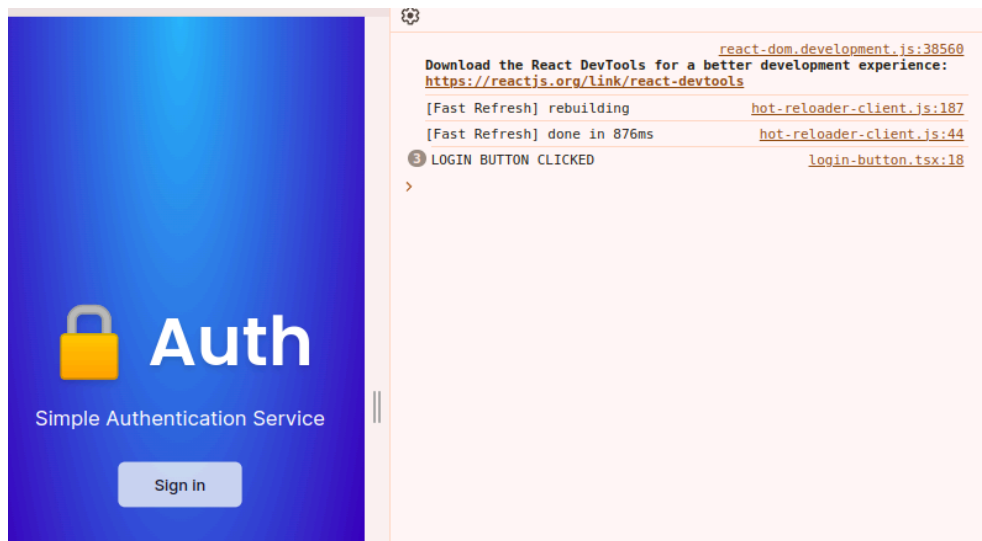
```

Now Wrap the SignIn button in Homepage with this component

```

<div>
  <LoginButton>
    /* Now this Button component need not to be button it can be
anything but still act like button */
    <Button variant="secondary" size="lg">
      Sign in
    </Button>
  </LoginButton>
</div>

```



Now on click the Sign in button we will be redirected to the Login Page

```

import { useRouter } from "next/navigation";

export const LoginButton = ({
  children,
  mode="redirect",

```

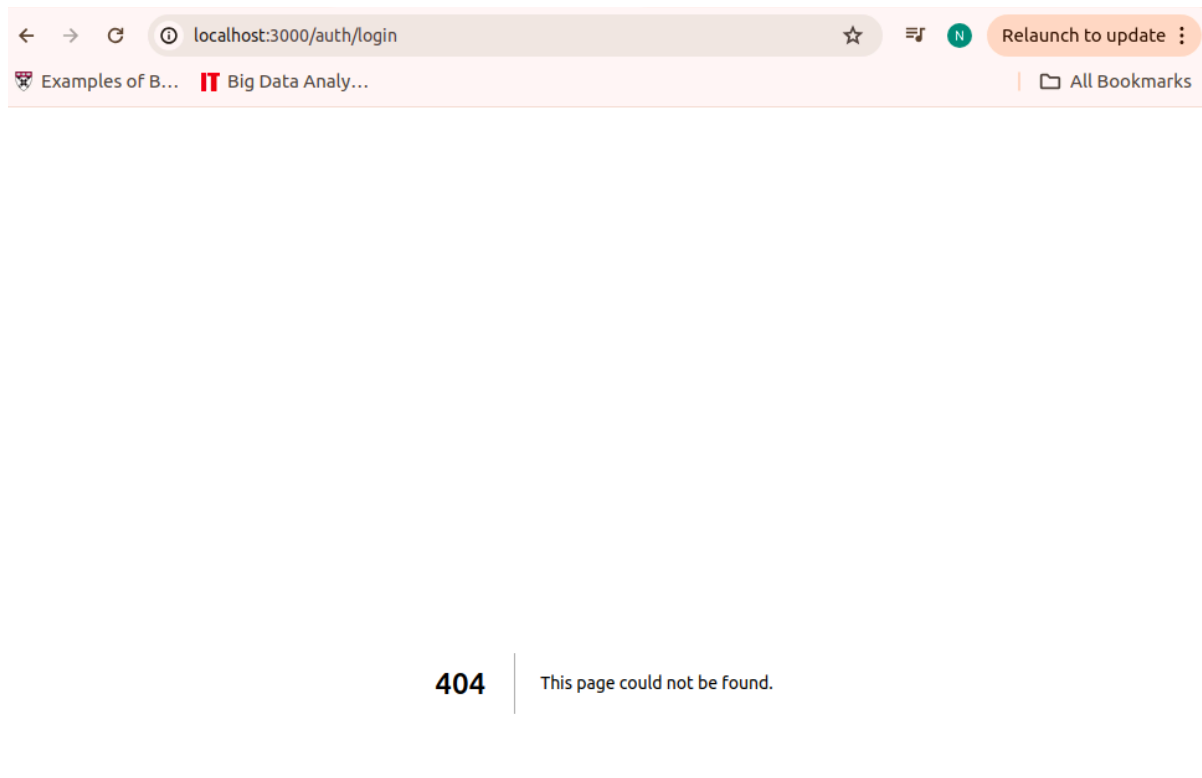
```

    asChild
  ):LoginButtonProps) =>{
    const router = useRouter();

    const onClick = () =>{
      // console.log("LOGIN BUTTON CLICKED")
      router.push("/auth/login");
      // this will be our login route in future
    }
  }

```

Currently it will show 404 since there is no such page now.



Now we will make a Form component

app/auth/Login/page.tsx Login page

```

import { LoginForm } from '@components/auth/login-form'
import React from 'react'

const LoginPage = () => {

```

```

return (
  // We will be returning Login Form component
  <LoginForm />
  // components/auth
)
}

export default LoginPage

```

app/auth/layout.tsx Common Layout to the auth folder

```

const AuthLayout = ({ children }: { children: React.ReactNode }) => {
  return(
    <div className="h-full flex items-center justify-center
bg-[radial-gradient(ellipse_at_top,_var(--tw-gradient-stops))]
from-sky-400 to-blue-800">
      {children}
    </div>
  );
};

export default AuthLayout;

```

components/login-form.tsx Will be reused

```

// we are not exporting as default since it is NOT a page and it is just a
component

export const LoginForm = ()=>{
  return(
    <div>
      Login Form
    </div>
  )
}

```

We will create a Card , everything such as new User Verification, Login, Register will all be rendered under Identical card model

npx shadcn-ui@latest add card it will add a Card component from Shadcn in ui folder

```
export const LoginForm = ()=>{
  return(
    <CardWrapper>
      Login Form
    </CardWrapper>
  )
}
```

Now create a new file called card-wrapper.tsx in compnents/auth

```
"use client"

// You can also use @ to be consistent throughout the project
import {
  Card,
  CardContent,
  CardFooter,
  CardHeader
} from "../ui/card";

interface CardWrapperProps{
  children:React.ReactNode;
  headerLabel:string;
  backButtonLabel:string;
  backButtonHref:string;
  // optional
  showSocial?:boolean;
};

export const CardWrapper = ({
  children,
  headerLabel,
  backButtonLabel,
  backButtonHref,
  showSocial
```

```

}: CardWrapperProps) => {
  return(
    <Card className="w-[400px] shadow-md">
      {children}
    </Card>
  )
}

```

Now login-form.tsx looks like

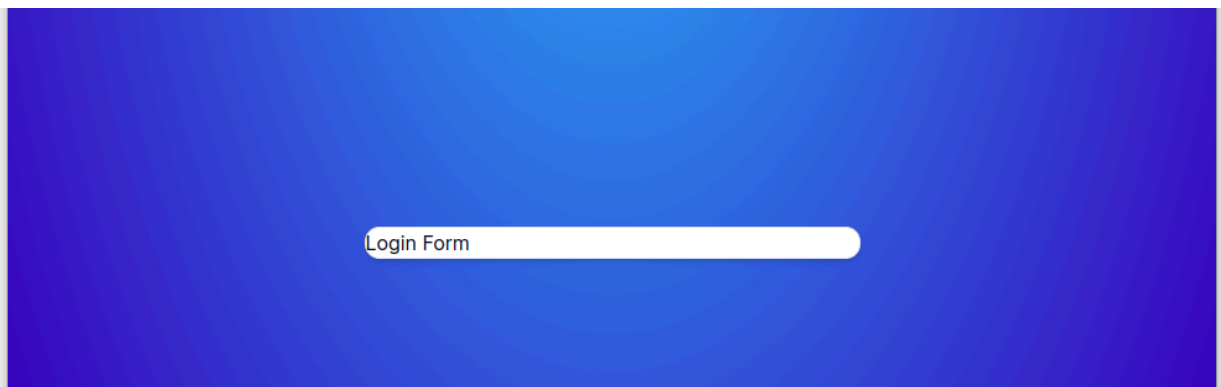
```

// we are not exporting as default since it is NOT a page and it is just a
component

import { CardWrapper } from "../card-wrapper"

export const LoginForm = ()=>{
  return(
    <CardWrapper
      headerLabel="Welcome back"
      backButtonLabel="Don't have an account?"
      backButtonHref="/auth/register"
      showSocial
    >
      Login Form
    </CardWrapper>
  )
}

```



Now we will create a Reusable Header component

```
import { Poppins } from "next/font/google";
import { cn } from "@/lib/utils";

const font = Poppins({
  subsets: ["latin"],
  weight: ["600"],
});

interface HeaderProps {
  label: string;
}

export const Header = ({
  label,
}: HeaderProps) => {
  return (
    <div className="w-full flex flex-col gap-y-4 items-center justify-center">
      <h1 className={cn("text-3xl font-semibold", font.className)}>
        🔒 Auth
      </h1>
      <p className="text-muted-foreground text-sm">
        /* This will be reused for different components */
        {label}
      </p>
    </div>
  )
}
```

Now we will use the headerLabel of CardWrapper component to render our Header component

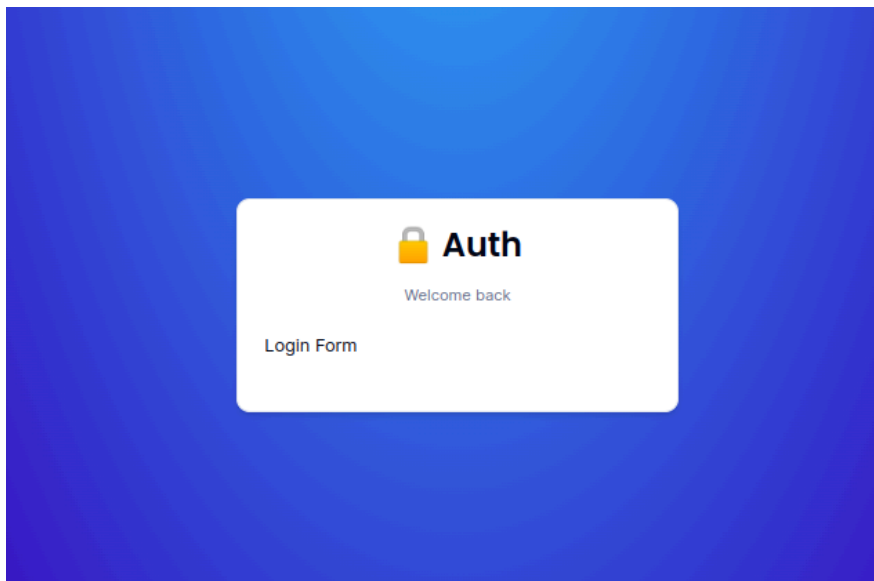
```
export const CardWrapper = ({
  children,
  headerLabel,
```

```

    backButtonLabel,
    backButtonHref,
    showSocial,
  }: CardWrapperProps) => {
    return (
      <Card className="w-[400px] shadow-md">
        <CardHeader>
          <Header label={headerLabel} />
        </CardHeader>
        <CardContent>{children}</CardContent>
        {showSocial && (
          <CardFooter>
            <Social />
          </CardFooter>
        )}
      </Card>
    );
  };
};

```

At the moment our login page will look like this



Now lets Create the Social components

On removing

Install package called React-icons for getting the google icons

npm i react-icons

```

"use client";

import { FcGoogle } from "react-icons/fc";
import { FaGithub } from "react-icons/fa";
import { Button } from "../ui/button";

export const Social = () => {
  return (
    <div className="flex items-center w-full gap-x-2">
      { /* Google */ }
      <Button
        size="lg"
        className="w-full"
        variant="outline"
        onClick={() =>{

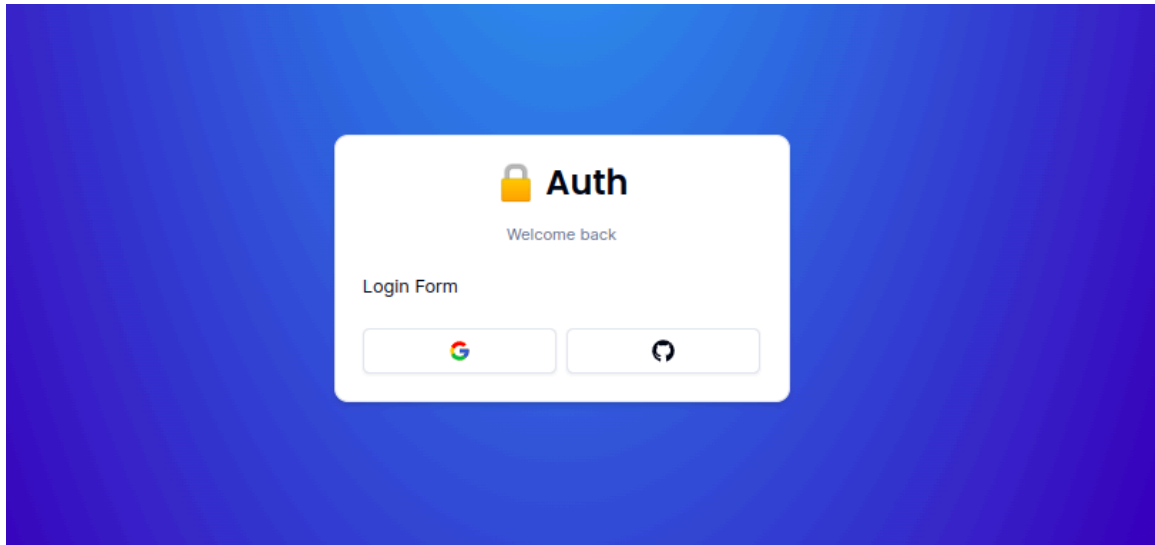
        }}
      >
        <FcGoogle className="h-5 w-5"/>
      </Button>

      { /* Github */ }
      <Button
        size="lg"
        className="w-full"
        variant="outline"
        onClick={() =>{

        }}
      >
        <FaGithub className="h-5 w-5"/>
      </Button>
    </div>
  )
}

```

Depending on the props showSocial we will hide or use them



Now lets use another component that will be used in CardWrapper component

```
<Card className="w-[400px] shadow-md">
  <CardHeader>
    <Header label={headerLabel} />
  </CardHeader>
  <CardContent>{children}</CardContent>
  {showSocial && (
    <CardFooter>
      <Social />
    </CardFooter>
  )}
  <CardFooter>
    <BackButton
      label={backButtonLabel}
      href={backButtonHref}
    />
  </CardFooter>
</Card>
```

BackButton component inside the component/auth folder

```
"use client";

import Link from "next/link";
import { Button } from "../ui/button";
```

```

interface BackButtonProps {
  href:string;
  label:string;
}

export const BackButton = ({
  href,
  label,
}:BackButtonProps) => {
  return(
    <Button
      variant="link"
      className="font-normal w-full"
      size="sm"
      asChild
      // so that we can properly render the link property which is
inside
    >
      <Link href={href}>
        {label}
      </Link>
    </Button>
  )
}

```

Card wrapper looks like

```

<CardFooter>
  <BackButton
    label={backButtonLabel}
    href={backButtonHref}
  />
</CardFooter>

```

Login form

```

import { CardWrapper } from "../card-wrapper"
export const LoginForm = ()=>{
  return(
    <CardWrapper

```

```

        headerLabel="Welcome back"
        backButtonLabel="Don't have an account?"
        backButtonHref="/auth/register"
        showSocial
    >
        Login Form
    </CardWrapper>
)
}

```



On clicking the Don't have an account it goes to this Link

Code till now

https://github.com/nthapa000/authentication_next/tree/fe2d54def6916a37af7dfa67c10ac40757bd1e5a

Login Form

Now add form component from shadcn

npx shadcn-ui@latest add form

Now add input component from shadcn

npx shadcn-ui@latest add input

Now we will create a form schema will we will use

In root directory create a folder **schemas/index.tsx** From here we will do the validation on the frontend as well as the backend

```
import * as z from "zod";

export const LoginSchema = z.object({
  email: z.string().email(),
  password: z.string(),
  // for registering new users we will also add the minimum length of 6
  // but we should keep it in login since it is possible before we keep min(6)
  // there would have been users who created the password of length less than 6
});
```

imports required

```
// we are not exporting as default since it is NOT a page and it is just a
// component
"use client"
// since we are using hook
import * as z from "zod"

import {useForm} from "react-hook-form"
import {zodResolver} from "@hookform/resolvers/zod"

import {
  Form,
  FormControl,
```

```

    FormItem,
    FormLabel,
    FormMessage,
  } from "@components/ui/form"

import { LoginSchema } from "@schemas"
import { CardWrapper } from "../card-wrapper"

```

Now lets see the login-form page

```

// we are not exporting as default since it is NOT a page and it is just a
component
"use client";
// since we are using hook
import * as z from "zod";

import { useForm } from "react-hook-form";
import { zodResolver } from "@hookform/resolvers/zod";

import {
  Form,
  FormControl,
  FormField,
  FormItem,
  FormLabel,
  FormMessage,
} from "@components/ui/form";

import { LoginSchema } from "@schemas";
import { CardWrapper } from "../card-wrapper";
import { Input } from "../ui/input";
import { Button } from "../ui/button";

export const LoginForm = () => {
  const form = useForm<z.infer<typeof LoginSchema>>({
    resolver: zodResolver(LoginSchema),
    defaultValues: {
      email: "",
      password: "",
    },
  })

```



```

    },
  });

const onSubmit = (values: z.infer<typeof LoginSchema>) => {
  console.log(values);
}

return (
  <CardWrapper
    headerLabel="Welcome back"
    backButtonLabel="Don't have an account?"
    backButtonHref="/auth/register"
    showSocial
  >
    <Form {...form}>
      <form
        onSubmit={form.handleSubmit(onSubmit)}
        className="space-y-6"
      >
        <div className="space-y-4">

          <FormField
            control={form.control}
            name="email"
            render={({field}) => (
              <FormItem>
                <FormLabel>Email</FormLabel>
                <FormControl>
                  <Input
                    {...field}
                    placeholder="john.wick@don.com"
                    type="email"
                  />
                </FormControl>
                /* Message of the form , we can also change
this message by going in the Login Schema*/
                // email: z.string().email({
                //   message: "Email is Required"
                // })
              </FormItem>
            )}
          />
        </div>
      </form>
    </Form>
  </CardWrapper>
)

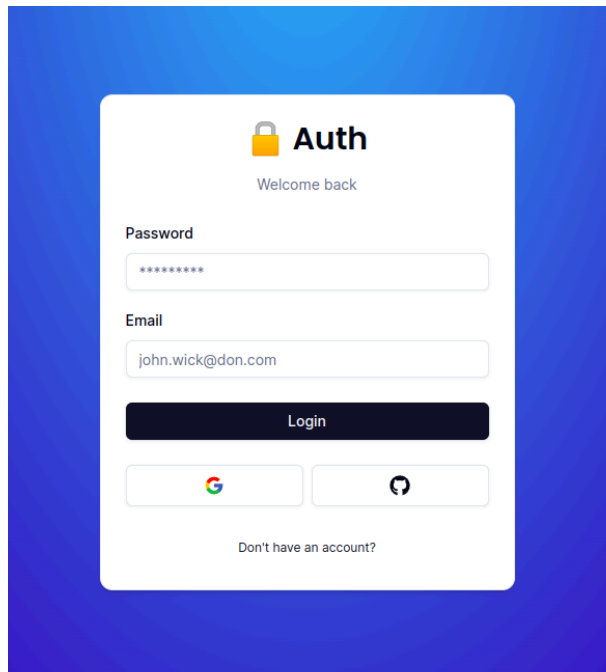
```

```

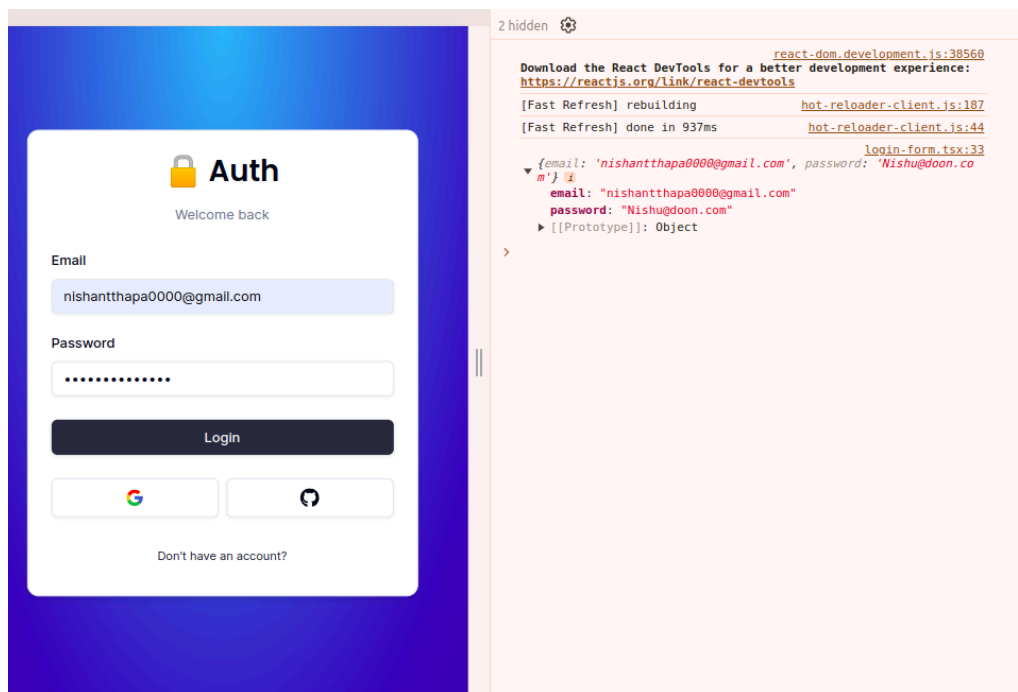
        <FormMessage />
    </FormItem>
  )}
/>
<FormField
  control={form.control}
  name="email"
  render={({field})=>(
    <FormItem>
      <FormLabel>Password</FormLabel>
      <FormControl>
        <Input
          {...field}
          placeholder="*****"
          type="password"
        />
      </FormControl>
      {/* Message of the form , we can also change
this message by going in the Login Schema*/
        // email:z.string().email({
        //   message:"Email is Required"
        // })
      }
    <FormMessage />
  </FormItem>
  )}
/>
</div>
{/* Button component */}
<Button
  type="submit"
  className="w-full"
>
  Login
</Button>
</form>
</Form>
</CardWrapper>
);
};

```

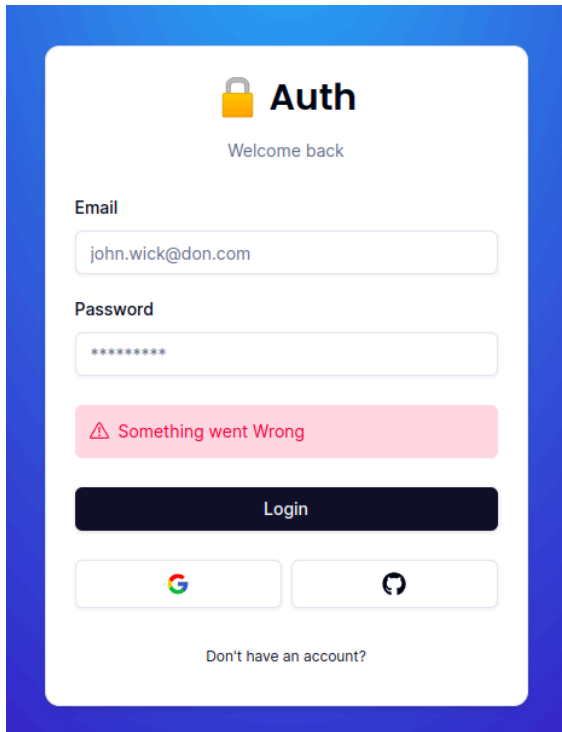
Lets see how it looks




Let see what happens when we click on the Login button



Now we will work on Error Component




 **Auth**

Welcome back



Email

john.wick@don.com

Password

 Something went Wrong

Login

Don't have an account?

```
import {ExclamationTriangleIcon} from "@radix-ui/react-icons";
interface FormErrorProps{
  message?:string;
};

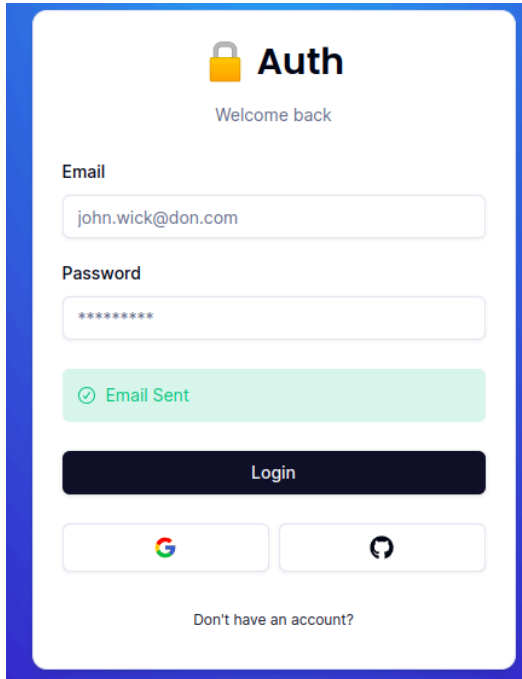
export const FormError =({
  message,
}:FormErrorProps)=>{
  if(!message) return null;

  return(
    <div className="bg-destructive/15 p-3 rounded-md flex items-center
gap-x-2 text-sm text-destructive">
      <ExclamationTriangleIcon className="h-4 w-4"/>
      <p>{message}</p>
    </div>
  )
}
```

login-form.tsx

```
<FormError message="Something went Wrong"/>
```

Now we will work on form success component



```
import {CheckCircledIcon} from "@radix-ui/react-icons";

interface FormSuccessProps{
  message?:string;
};

export const FormSuccess = ({
  message,
}:FormSuccessProps)=>{
  if(!message) return null;

  return(
    <div className="bg-emerald-500/15 p-3 rounded-md flex items-center gap-x-2 text-sm text-emerald-500">
      <CheckCircledIcon className="h-4 w-4"/>
      <p>{message}</p>
    </div>
  )
}
```

```
}
```

Repository till this point:

https://github.com/nthapa000/authentication_next/tree/2a09f0e3ba7d372b552f79be3f06ce5a8c3c74b5

Find a way to transfer these values from client to the server

(Server Actions)

Create a new folder in the root directory called Actions

Create a folder called login.ts

First thing is to use “use server”

```
"use server"

export const login = (values:any)=>{
  console.log(values);
  // This is valid server actions
}
```

Till now we could on pressing the login button could see the email and password only on the browser. Now we will see on the server(for us it is vs code terminal)

login-form.tsx

```
const onSubmit = (values:z.infer<typeof LoginSchema>)=>{
  login(values)
}
```

```

▲ Next.js 14.2.3
- Local:      http://localhost:3000

✓ Starting...
✓ Ready in 2.3s
o Compiling /auth/login ...
✓ Compiled /auth/login in 1959ms (627 modules)
GET /auth/login 200 in 2342ms
✓ Compiled in 305ms (297 modules)
✓ Compiled /favicon.ico in 167ms (360 modules)
GET /favicon.ico 200 in 264ms
{ email: 'BhaiteraDesi@123.gmail.com', password: 'asdadasdasda' }
POST /auth/login 200 in 37ms

```

We will use `useTransition` for pending state

```
import { useTransition } from "react";
```

Now wrap make a state

```
export const LoginForm = () => {
  const [isPending, startTransition]=useTransition();
}
```

On the onSubmit wrap the login with startTransition function

```
const onSubmit = (values:z.infer<typeof LoginSchema>)=>{
  startTransition(()=>{
    login(values)
  })
}
```

Now use `isPending` to disable all the state which we want

```
disabled={isPending}
```

Disable input of email and password when it is pending so that user cannot send so many request to Server

`startTranscation` will be useful when we will use NextJs cache function **`revalidatePath()`** and **`revalidateTag()`** .

Now we will think how to validate the fields on the server.

```
"use server"
```

```
import * as z from "zod";
import { LoginSchema } from "@schemas";

export const login = (values:z.infer<typeof LoginSchema>)=>{
  console.log(values);
  // This is valid server actions
}
```

Now lets do validation it is always easier to bypass the client side validation, whereas in backend we can't manipulate

```
"use server"

import * as z from "zod";
import { LoginSchema } from "@schemas";

export const login = (values:z.infer<typeof LoginSchema>)=>{
  const validateFields = LoginSchema.safeParse(values);
  if(!validateFields.success){
    return{error:"Invalid fields"}
  }
  return{success:"Email sent!"}
}
```

Now lets try to check these errors

We will make two state field one for Error and one for success

```
const [error,setError] = useState<string | undefined>("");
const [success,setSuccess] = useState<string | undefined>("");
```

Now lets see the onSubmit button

```
const onSubmit = (values:z.infer<typeof LoginSchema>)=>{
  // Everytime we hit the new submit then we will clear all successs and
  error message
  setError("");
  setSuccess("");

  startTransition(()=>{
    login(values)
      .then((data)=>{
```



```

      setError(data.error);
      setSuccess(data.success);
    })
  })
  // if we didn't want to do the server action then we could do in such a
way
  // axios.post("/your/api/route",values)
}

```

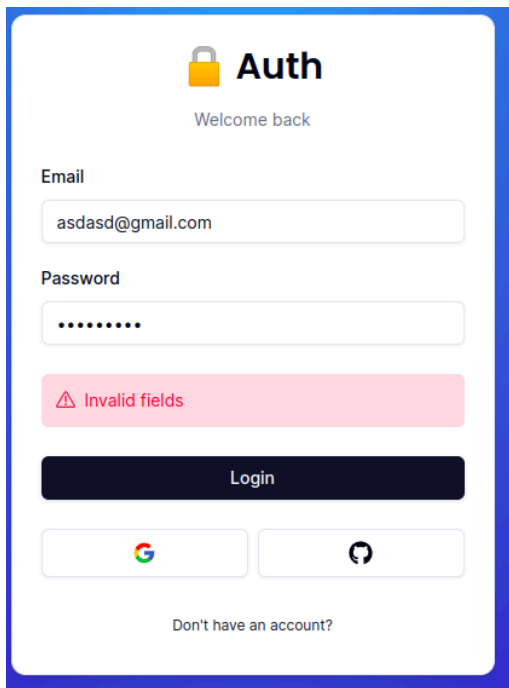
Now we will pass the error and success state to the FormError and FormSuccess component


```

<FormError message={error}/>
<FormSuccess message={success}/>

```

Now on our login server action now lets forcefully send the error message on every input or onsubmit from the button



 **Auth**


Welcome back

Email



asdasd@gmail.com

Password

.....

 Invalid fields

Login

Don't have an account?

This is our first server action , getting pendingState , this code is never bundled with the client due to “use server”

Register Form

We have to resolve a 404 route currently

app/auth/register/page.tsx

```
const Register = () => {  
  return (  
    <div>Register</div>  
  )  
}  
  
export default Register
```

Now lets work on Register Schema

```
export const RegisterSchema = z.object({  
  email: z.string().email({  
    message: "Email is required",  
  }),  
  password: z.string().min(6, {  
    message: "Minimum 6 character required",  
  }),  
  name: z.string().min(1, {  
    message: "Name is required"  
  }),  
});
```

Now make a register form component

```
"use client";  
import * as z from "zod";  
  
import { useForm } from "react-hook-form";  
import { zodResolver } from "@hookform/resolvers/zod";  
import { useState, useTransition } from "react";  
  
import {  
  Form,  
  FormControl,  
  FormField,  
  FormItem,  
  FormLabel,  
}
```

```

    FormMessage,
  } from "@components/ui/form";

import { RegisterSchema } from "@schemas";
import { CardWrapper } from "../card-wrapper";
import { Input } from "../ui/input";
import { Button } from "../ui/button";
import { FormError } from "../form-errors";
import { FormSuccess } from "../form-success";
import { login } from "@actions/login";

export const RegisterForm = () => {
  const [error, setError] = useState<string | undefined>("");
  const [success, setSuccess] = useState<string | undefined>("");
  const [isPending, startTransition] = useTransition();

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

  const onSubmit = (values: z.infer<typeof RegisterSchema>) => {
    setError("");
    setSuccess("");

    startTransition(() => {
      login(values)
        .then((data) => {
          setError(data.error);
          setSuccess(data.success);
        })
    })
  }

  return (
    <CardWrapper

```

```

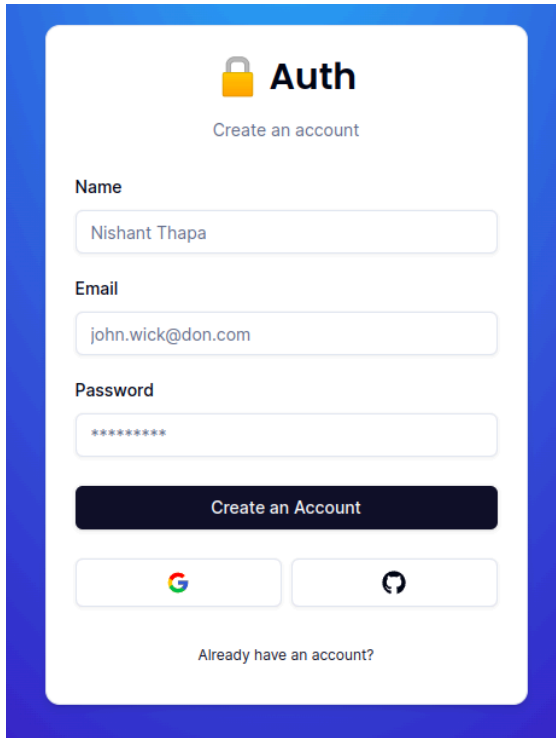
headerLabel="Create an account"
backButtonLabel="Already have an account?"
backButtonHref="/auth/login"
showSocial
>
<Form {...form}>
  <form
    onSubmit={form.handleSubmit(onSubmit)}
    className="space-y-6"
  >
    <div className="space-y-4">
      <FormField
        control={form.control}
        name="name"
        render={({field})=>(
          <FormItem>
            <FormLabel>Name</FormLabel>
            <FormControl>
              <Input
                {...field}
                disabled={isPending}
                placeholder="Nishant Thapa"
              />
            </FormControl>
            <FormMessage />
          </FormItem>
        )}
      />
      <FormField
        control={form.control}
        name="email"
        render={({field})=>(
          <FormItem>
            <FormLabel>Email</FormLabel>
            <FormControl>
              <Input
                {...field}
                disabled={isPending}
                placeholder="john.wick@don.com"
                type="email"


```

```

        />
        </FormControl>
        <FormMessage />
    </FormItem>
    })
  />
  <FormField
    control={form.control}
    name="password"
    render={({field})=>(
      <FormItem>
        <FormLabel>Password</FormLabel>
        <FormControl>
          <Input
            {...field}
            placeholder="*****"
            type="password"
            disabled={isPending}
          />
        </FormControl>
        <FormMessage />
      </FormItem>
    )}
  />
</div>
<FormError message={error}/>
<FormSuccess message={success}/>
<Button
  type="submit"
  className="w-full"
  disabled={isPending}
>
  Create an Account
</Button>
</form>
</Form>
</CardWrapper>
);
};

```



 **Auth**

Create an account

Name



Nishant Thapa

Email

john.wick@don.com

Password

Create an Account

Already have an account?

Now we will have register action

```
"use server"

import * as z from "zod";
import { RegisterSchema } from "@schemas";

// since we are using promises
export const register = async (values: z.infer<typeof RegisterSchema>) => {
  const validateFields = RegisterSchema.safeParse(values);
  if (!validateFields.success) {
    return { error: "Invalid fields" }
  }
  return { success: "Email sent!" }
}
```

register-form.tsx

```
import { register } from "@actions/register";
```

```
startTransition(()=>{
  register(values)
    .then((data)=>{
      setError(data.error);
      setSuccess(data.success);
    })
  })
})
```

Repository at the moment

https://github.com/nthapa000/authentication_next/tree/d7fd51196a8cd628c814485c9fdcb24cee3f215a