**[Next.js14] NextAuth v5 (1) — SignIn/SignOut**

[[](https://medium.com/@youngjun625?source=post_page-----7e30cce52e7f--------------------------------)](https://medium.com/@youngjun625?source=post_page-----7e30cce52e7f--------------------------------)

[YJ Kim](https://medium.com/@youngjun625?source=post_page-----7e30cce52e7f--------------------------------)

·

Follow

7 min read

·

4 days ago

43



This is a tutorial on using NextAuth v5 on Nextjs14. I’ve noticed that there aren’t much sources about NextAuth v5 since it’s still on beta version.

This tutorial has been written based on [Next.js14 official tutorial](https://nextjs.org/learn?utm_source=next-site&utm_medium=homepage-cta&utm_campaign=home) + some of the things I learned on the way of developing with it.

The full tutorial can be found below😊

YJ Kim

[YJ Kim](https://medium.com/@youngjun625?source=post_page-----7e30cce52e7f--------------------------------)

**Nextjs14 & NextAuthV5 Tutorial**

[View list](https://medium.com/@youngjun625/list/nextjs14-nextauthv5-tutorial-4f931159cf01?source=post_page-----7e30cce52e7f--------------------------------)

3 stories







**Preparation**

👉**Creating Next App**

First create a next app using the command npx create-next-app@latest . I’ve used typescript, tailwind css, src/ directory and app router.

👉**Installing Packages**

Next, install next-auth with the command npm install next-auth@beta  
Make sure "next-auth": "^5.0.0-beta.4" or above has been added to package.json



**👉 Genrating secret key**

Generate a secret key using openssl with the following commad.

openssl rand -base64 32

There are many tutorials on installing openssl. Make sure not to share the secret key externally!



**👉 Adding environment variable inside .env file**

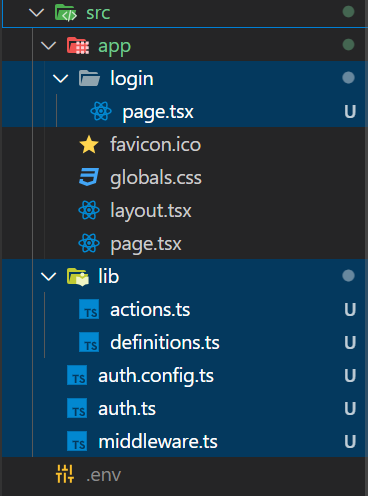
Lastly add a ‘AUTH\_SECRET’ variable inside the .env file and we are ready to go. (**Make sure .env is added to .gitignore file**)

# .env file  
AUTH\_SECRET=2rhmxBZsnncgb392RxtB...

**Code**

**👉 Folder Structure**

The folder structure of the project is the following



*src/app/login folder,  
src/lib folder,  
src/auth.config.ts,  
src/auth.ts,  
src/middleware.ts*

has been added to the basic Nextjs14 app.

**👉 App router**

Let’s briefly talk about the app router before we go on. All folders within the app directory becomes a route. For example, running the app with the folder structure above and heading to <http://localhost:3000/login> will show the component written inside (src/app/login/page.tsx)

Next has few special files like the ‘pages.tsx’ file that is used for special cases. More information can be found [here](https://nextjs.org/docs/app/building-your-application/routing)

**👉 Login page**

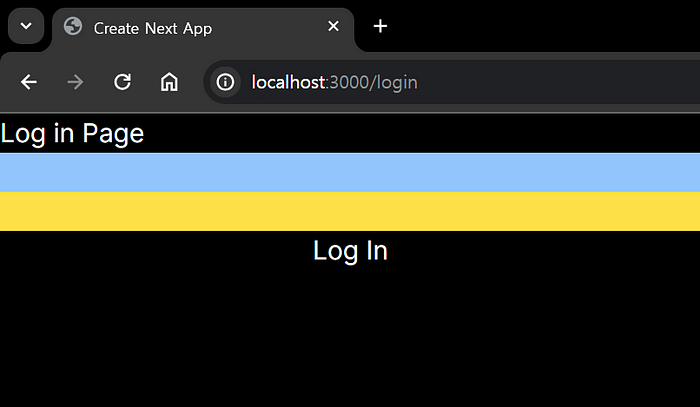
// src/app/login/page.tsx  
"use client"  
  
// import { authenticate } from "@/lib/actions"  
// import { useFormState } from "react-dom"  
  
export default function Page() {  
 // authenticate() method will be declared later  
 // const [errorMsg, dispatch] = useFormState(authenticate, undefined)  
 return (  
 <div>  
 <h1>Log in Page</h1>  
 <form className="flex flex-col" > {/\* action={dispatch} \*/}  
 <input className="bg-blue-300 text-black" name="id"></input>  
 <input className="bg-yellow-300 text-black" name="password" type="password"></input>  
 <button>  
 Log In  
 </button>  
 {/\* errorMsg will be declared later   
 <p>{errorMsg}</p> \*/}  
 </form>  
 </div>  
 )  
 }

**👉 Home page**

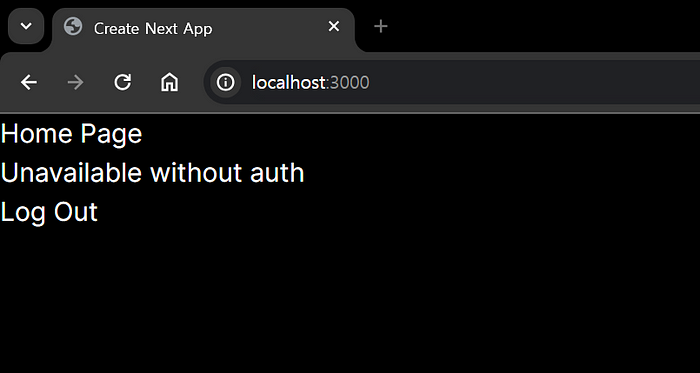
// src/app/page.tsx  
// import { signOut } from "../auth"  
  
export default async function Home() {  
 return (  
 <div>  
 <h1>Home Page</h1>  
 <h2>Unavailable without auth</h2>  
 <form  
 // action={async () => {  
 // // signOut() Method will be declared later  
 // 'use server';  
 // await signOut();  
 // }}  
 >  
 <button>  
 Log Out  
 </button>  
 </form>  
  
 </div>  
 )  
}

Now let’s run the app with npx next dev or npm run dev or yarn depending on your environment.

<http://localhost:3000/login>



<http://localhost:3000/>



Let’s move on once we are able to render these 2 pages!

**👉 Authentication and ProtectedRoutes**

Within the src folder, let’s add 3 files : auth.config.ts, auth.ts, middleware.ts  
and within scr/lib folder, let’s add 2 files : definitions.ts, actions.ts

In Next.js, a special file name ‘middleware’ always runs before a request is completed. For example, if user requests a connection to <http://localhost:3000/> , the code within ‘middleware.ts’ is ran before the request is accepted.

Therefore adding the authentication logic within ‘middleware.ts’ will protect certain routes. Authentication logic will be something like “if authenticated then redirect to homepage, else redirect to login page”.

Sometimes the middleware doesn’t run properly. If that happens, try moving the 3 files outside of the src folder (at the root directory) and see if it fixes the issue

First, let’s define the type ‘User’

// src/lib/definitions.ts  
export type User = {  
 id: string  
 email: string  
 name : string  
};

Authentication Logic Code

// src/auth.config.ts  
import type { NextAuthConfig } from 'next-auth';  
   
export const authConfig = {  
 pages: {  
 signIn: '/login',  
 },  
 callbacks: {  
 authorized({ auth, request: { nextUrl } }) {  
 // Check if the user is authenticated  
 const isLoggedIn = !!auth?.user;  
 // Initialize protected routes  
 // Here, all routes except the login page is protected  
 const isOnProtected = !(nextUrl.pathname.startsWith('/login'));  
   
 if (isOnProtected) {  
 if (isLoggedIn) return true;  
 return false; // redirect to /login  
 } else if (isLoggedIn) {  
 // redirected to homepage  
 return Response.redirect(new URL('/', nextUrl));  
 }  
 return true;  
 },  
 },  
 providers: [], // Add providers with an empty array for now  
} satisfies NextAuthConfig;

Inititializing Next Auth

// src/auth.ts  
import NextAuth from 'next-auth';  
import { authConfig } from './auth.config';  
import Credentials from 'next-auth/providers/credentials';  
import { User } from '@/lib/definitions';  
  
export const { signIn, signOut } = NextAuth({  
 ...authConfig,  
 providers: [  
 Credentials({  
 async authorize(credentials) {  
 if (credentials.id && credentials.password) {  
 // Add you backend code here  
 // let loginRes = await backendLogin(credentials.id, credentials.password)  
 let loginRes = {  
 success : true,  
 data : {  
 user: {  
 ID: "john\_doe",  
 NAME: "John Doe",  
 EMAIL: "email@email.email",  
 },  
 }  
 }  
 // Failed logging in  
 if (!loginRes.success) return null;  
 // Successful log in  
 const user = {  
 id: loginRes.data.user.ID ?? '',  
 name: loginRes.data.user.NAME ?? '',  
 email: loginRes.data.user.EMAIL ?? '',  
 } as User;  
 return user;  
 }  
 return null;  
 },  
 })  
 ],  
 callbacks: {  
 async session({ session, token, user }) {  
 session.user = token.user as User  
 return session;  
 },  
 async jwt({ token, user, trigger, session }) {  
 if (user) {  
 token.user = user;  
 }  
 return token;  
 },  
 },  
});

Finally, run NextAuth inside middleware.ts

// src/middleware.ts  
import NextAuth from 'next-auth';  
import { authConfig } from './auth.config';  
   
export default NextAuth(authConfig).auth;  
  
export const config = {  
 // https://nextjs.org/docs/app/building-your-application/routing/middleware#matcher  
 matcher: ['/((?!api|\_next/static|\_next/image|.\*\\.png$).\*)'],  
};

Next, set up the log in method that will be called by the client

// src/lib/actions.ts  
"use server"  
  
import { signIn } from "../auth";  
import { AuthError } from "next-auth";  
  
export async function authenticate(  
 prevState: string | undefined,  
 formData: FormData,  
 ) {  
 try {  
 await signIn('credentials', formData);  
 } catch (error) {  
 if (error instanceof AuthError) {  
 return 'log in failed'  
 }  
 throw error;  
 }  
 }

**🔴Now let’s uncomment some lines on src/app/login/page.tsxand src/app/page.tsx🔴**

**👉 Login page**

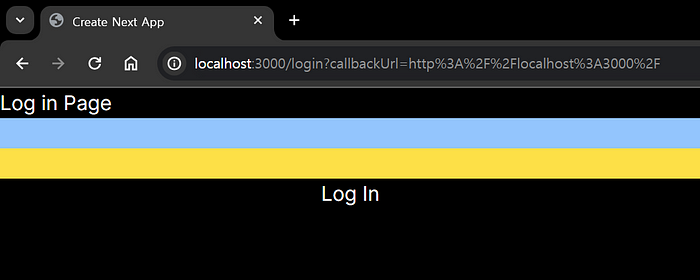
// src/app/login/page.tsx  
"use client"  
  
import { authenticate } from "@/lib/actions"  
import { useFormState } from "react-dom"  
  
export default function Page() {  
 const [errorMsg, dispatch] = useFormState(authenticate, undefined)  
 return (  
 <div>  
 <h1>Log in Page</h1>  
 <form className="flex flex-col" action={dispatch}>  
 <input className="bg-blue-300 text-black" name="id"></input>  
 <input className="bg-yellow-300 text-black" name="password" type="password"></input>  
 <button>  
 Log In  
 </button>  
 <p>{errorMsg}</p>  
 </form>  
 </div>  
 )  
 }

**👉 Home page**

// src/app/page.tsx  
import { signOut } from "../auth"  
  
export default async function Home() {  
 return (  
 <div>  
 <h1>Home Page</h1>  
 <h2>Unavailable without auth</h2>  
 <form  
 action={async () => {  
 'use server';  
 await signOut();  
 }}  
 >  
 <button>  
 Log Out  
 </button>  
 </form>  
 </div>  
 )  
}

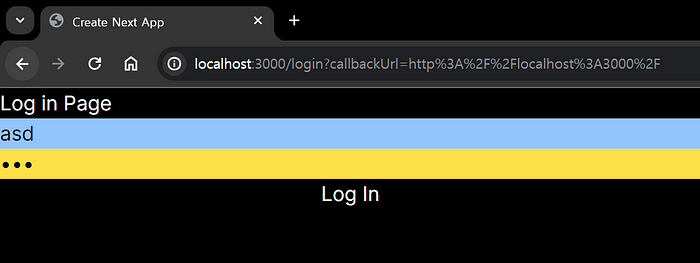
**✅ Results**

Heading to <http://localhost:3000/> without loggin in

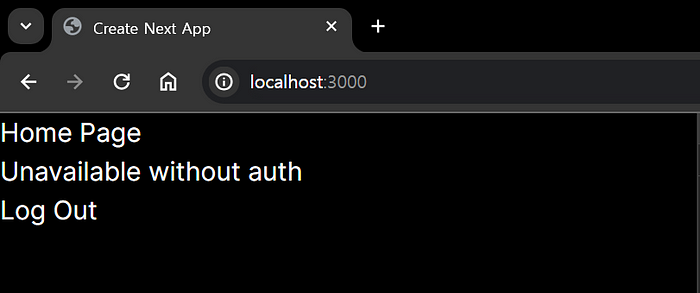


We are redirected to the login page.

Now type in any credentials



Then press on the ‘Log In’ button



We are successfully logged in and authenticated to view the home page. Try clicking on the ‘Log Out’ button to see what happens as well.

For now typing in any credentials allows authentication because we have hard coded the use data in src/auth.ts file as

user: {  
 ID: "john\_doe",  
 NAME: "John Doe",  
 EMAIL: "email@email.email",  
},

Therefore all you need to do is fix the loginRes and User type definition according to your backend code and you are good to go

**🔴Important Notes when Deploying🔴**

Before deploying your app, make sure to add the following variables inside you .env file

NEXTAUTH\_URL= ${deploying domain url}  
NEXT\_PUBLIC\_API\_URL= ${deploying domain url}

If the above is not satisfied, the callback url will be set to localhost:3000 and errors will occur.

**Upcoming & Source Code**

Next tutorial is about calling sessions to view user data and ways to update the session!

**[[Next.js14] NextAuth v5 (2) — Session/Update](https://medium.com/@youngjun625/next-js14-nextauth-v5-2-session-update-b977cb6afd47?source=post_page-----7e30cce52e7f--------------------------------" \t "_blank)**

[This is a tutorial on using NextAuth v5 on Nextjs14. Part 2 is about calling sessions and updating sessions](https://medium.com/@youngjun625/next-js14-nextauth-v5-2-session-update-b977cb6afd47?source=post_page-----7e30cce52e7f--------------------------------" \t "_blank)

[medium.com](https://medium.com/@youngjun625/next-js14-nextauth-v5-2-session-update-b977cb6afd47?source=post_page-----7e30cce52e7f--------------------------------" \t "_blank)

You can checkout the full source code below!

**[GitHub - youngjun625/nextauth14-nextauthv5: [Next.js14] NextAuth v5 tutorial](https://github.com/youngjun625/nextauth14-nextauthv5?source=post_page-----7e30cce52e7f--------------------------------" \t "_blank)**

[Next.js14] NextAuth v5 tutorial. Contribute to youngjun625/nextauth14-nextauthv5 development by creating an account on…](https://github.com/youngjun625/nextauth14-nextauthv5?source=post_page-----7e30cce52e7f--------------------------------" \t "_blank)

[github.com](https://github.com/youngjun625/nextauth14-nextauthv5?source=post_page-----7e30cce52e7f--------------------------------" \t "_blank)

[Nextjs](https://medium.com/tag/nextjs?source=post_page-----7e30cce52e7f---------------nextjs-----------------)

[Nextjs14](https://medium.com/tag/nextjs14?source=post_page-----7e30cce52e7f---------------nextjs14-----------------)

[Authentication](https://medium.com/tag/authentication?source=post_page-----7e30cce52e7f---------------authentication-----------------)

[Nextauth](https://medium.com/tag/nextauth?source=post_page-----7e30cce52e7f---------------nextauth-----------------)

[Login](https://medium.com/tag/login?source=post_page-----7e30cce52e7f---------------login-----------------)