Yarn add cookie

Yarn add -D @types/cookie

Import cookie frim ‘cookie’

Ve login basariliysa headerimizi cookiyi kullanarak ayarliyoruz.

Cookie.serialize tipki localStroage’ye json.stringfy yaparak auth adinda tokenimizi kaydeder gibi kaydeder.

res.setHeader('Set-Cookie', cookie.serialize('auth', token, {

          httpOnly: true, **//sadece http isteklerini kabul eder**

          secure:  process.env.NODE\_ENV !== 'development', **//https icin islemleri yapar sadece. Amaburada developmenta esit degilse yap demisiz yani sadece herhangi bir hosta yukledigimizde bu gecerli olacak.**

          sameSite: **'strict' //true veya ‘strict’ ayarlanmalidir.**

maxAge: 3600, *//1 hours*

          path: '/' //Eger bunu yapmazsak loginden sonra login icin istek yapilan yolu path olarak kabul edecektir. Diyelim login /api/login de ise path /api olarak gozukecektir.

        }))

Daha sonra asagidaki gibi req.cookies.auth ‘u kontrol ederek authenticationimizi kontrol edebiliriz veya istersek asagidaki gibi bir middleware ile bunu yapabilir ve kullanabiliriz.

**MIDDLEWARE OLUSTURULUMU**

export const authenticated = (*fn*: NextApiHandler) => async (

*req*: NextApiRequest,

*res*: NextApiResponse

) => {

  jwt.verify(req.cookies.auth!, SECRET\_KEY\_DONENV, async function(*err*, *decoded*) {

    if (!err && decoded) {

      return await fn(req, res);

    }

    res.status(401).json({ message: 'Sorry you are not authenticated' });

  });

};

**MIDDLEWARE KULLANIMI**

export default authenticated(async function getPeople(*req*:NextApiRequest, *res*:NextApiResponse) {

  const db = await open({

    filename: 'mydb.sqlite',

    driver: Database

  });

  const people = await db.all('select id, email, name from person');

  res.json(people);

});

Tarayicimizi f5 yaptigimizda cookieyi yakalayip gormuyorsa asagidaki gibi cookieyi tekrar tanimlamamiz gerekiyor. Cunku serverside renderindeki COOKIE CLIENT’a gonderilmiyor.

FETCH islemleri serverside da yapildigi icin ayarlari oradan yapacagiz.

Once headers deki cookieyi cekerek cookie degiskenine atiyoruz.

Daha sorna fetch yaparken fetch yapilan path in headerdaki cookiesini buna ayarliyoruz. Ve f5 ile yenilenmelerdeki sorunun onune geciyoruz.

People.getInitialProps = async (*ctx*: NextPageContext) => {

  const cookie = await ctx.req?.headers.cookie;

  const resp = await fetch('http://localhost:3000/api/people', {

    headers: {

      cookie: cookie!

    }

  });

  const json = await resp.json();

  return { people: json };

}

**useRef KULLANIMI**

Once useRef olsutururuz asagidaki gibi

  const emailRef = useRef<HTMLInputElement>(null);

dahasonra bunu ref olarak atamak istedigimiz birseye atariz. Ornek asagidaki gibi inputa

<input type="text" placeholder="Email" ref={emailRef} />

Vedaha sonra bunlara ulasmak istersek eger bir buton olustururuz ve bu butona onClick eventi ile asagidaki gibi tek tiklamamizlas referans verdigimiz degerlerin degerlerini current.value ile yakalariz.

<button onClick={() => {

        console.log(emailRef.current?.value, passRef.current?.value);

      }}>Login</button>

**EXAMPLE**

import { useRef } from "react"

export default function Login() {

  const emailRef = useRef<HTMLInputElement>(null);

  const passRef = useRef<HTMLInputElement>(null);

  const handleLogin = () => {

    console.log(emailRef.current?.value, passRef.current?.value);

  }

  return (

    <div>

      <input type="text" placeholder="Email" ref={emailRef} />

      <input type="text" placeholder="Password" ref={passRef} />

      <button onClick={handleLogin}>Login</button>

    </div>

  )

}