**Week 16.2**

**16.2.1 Auth using Cookies**

* Now the cookie is first set by the server and then in every subsequent request the cookie is sent by the browser.
* You can’t do localStorage in Next.js because the very first request that goes out when you enter the URL cannot access the localStorage item. In subsequent requests, we can send the auth header but not in the first request.
* Next.js server needs to have user specific data so it can do SSR and provide a pre-rendered page.
* When you log out, the cookie gets cleared.
* Browser automatically sets the set-cookie header when it comes back as a response header. It then automatically sents it in every subsequent request.
* CSRF – Cross Site Request Forgery. In CSRF attacks, cookies can be sent from any malicious website etc and the server accepts it. The thing is cookies get stored in the browser and if you go over on the malicious website then that website can send a request to our server and the cookie would be sent.
* SameSite solves this.
* You should never SameSite:none
* Lax is between none and strict. It only allows get request and top level navigation.
* Use jwt to encode cookies.
* Cors should be like this:  
  app.use(cors({

credentials: true,

origin: "http://localhost:5173"

}));

* This means that allow the server to set cookies on this origin. If we have frontend and backend on different domains then we need to do credentials: true.
* Cookies are useful for browsers and not for mobile apps.
* You can set multiple cookies using res.cookie multiple times.

**16.2.2 NextAuth**

* Catching all routes.
* Provider lets you decide how you want to implement authentication. Three basic types: OAuth (Google, Facebook, etc), Email (Password-less), Credentials (Do whatever you want)
* NextAuth would not let you create signup page. But a default sign in page is created by NextAuth CredentialsrProvider.
* NextAuth is a session management library that manages the cookies etc.
* Use useSession() hook to get user’s info in a clients side component. A client side component has ‘use client’.
* In a server side component use async await and getServerSession(). Also use this for routes and apis.
* Callbacks let you tap in the life cycle events of authentication.
* Whenever you’re using getServerSession, make sure you extract out the config in a lib/auth.ts file and pass NEXT\_AUTH as arguments to getServerSession.
* Make sure to add pages in config in the end so that users are redirected to the main sign in page and not the api/auth/… one.
* Overview: Build the signin page at api/auth/[…nextauth] using provider 🡪 At the main sign in page, use signIn(“google”), signIn(“github”) to redirect to google or github signin 🡪 Add page in the end to config so when a user clicks sign in button from the home page, they are redirected to the main sign in page.