#### Authentication and Session

Course Code: CSC 4182 Course Title: Advanced Programming In Web Technologies

# Dept. of Computer Science Faculty of Science and Technology

<b>Lecture No:</b>	1	Week No:	14	Semester:	
Lecturer:	Sazzad Hossain; sazzad@aiub.edu				

### Lecture Outline



- ✓ Tailwind CSS
- √ Key Features of Tailwind
- ✓ Installation
- ✓ Configure Project
- ✓ Using TailWind
- ✓ Tailwind Documentation
- ✓ Tailwind Resources





 Cookies are small pieces of data that are stored on the client-side (typically in the user's browser) and are often used for session management, user authentication, and other data storage purposes.

## NestJs Cookie Configuration



- **Secure**: This option determines whether the cookie **should only** be sent over **secure connections (HTTPS)**. When set to false, the cookie can also be sent over non-secure (HTTP) connections.
- httpOnly: When this option is set to true, the cookie is inaccessible to JavaScript running in the browser.
- maxAge: This option specifies the maximum age of the cookie in milliseconds.

# Cross-Origin Resource Sharing



 CORS (Cross-Origin Resource Sharing is a security feature implemented by web browsers that controls whether a web page running at one origin (domain) is allowed to request resources from a server at a different origin (domain).

```
app.enableCors({
          origin: true,
          methods: 'GET,HEAD,PUT,PATCH,POST,DELETE,OPTIONS',
          credentials: true,
     });
```

# Cross-Origin Resource Sharing



**Origin**: This specifies the allowed origins for cross-origin requests. The **true** value here means that any origin is allowed to make requests to the server.

**Methods**: This specifies the HTTP methods that are allowed in the cross-origin requests. In the provided code, it allows the common HTTP methods like GET, HEAD, PUT, PATCH, POST, DELETE, and OPTIONS.

**Credentials**: This indicates whether the server should include any cookies, HTTP authentication, or client-side SSL certificates in crossorigin requests. When set to true, it means that the browser's crossorigin request includes credentials (like cookies) in the request.

## Axios with Credentials Option



- withCredentials option is used to control whether the browser should send cookies or HTTP authentication along with crossorigin requests.
- When withCredentials is set to false (the default), Axios will not send cookies or authentication headers with cross-origin requests.
- When withCredentials is set to true, Axios will include cookies and authentication headers in cross-origin requests.

```
const response = await axios.post(process.env.NEXT_PUBLIC_API_ENDPOINT + '/admin/signin/',
    {
       email,
       password,
    },
    {
       headers: { 'Content-Type': 'application/x-www-form-urlencoded' },
       withCredentials: true
    }
}:
```

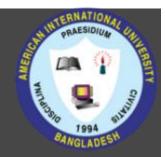


**useContext** hook is part of the React Hooks API and is used for **consuming values from the React context**. Context provides a way to **pass data through the component tree** without having to pass props down manually at every level.

Context.Provider component work together with useContext to provide a way to share state or data across components without having to pass props manually through each level of the component tree.



```
// MyContext.js
import React, { createContext, useContext } from 'react';
// Create a context
const MyContext = createContext();
// Create a provider component
const MyProvider = ({ children }) => {
  const sharedData = 'Hello from Context';
  return <MyContext.Provider
value={sharedData}>{children}</MyContext.Provider>;
};
// Custom hook to consume the context
const useMyContext = () => useContext(MyContext);
export { MyProvider, useMyContext };
```



```
// _app.js
import React from 'react';
import { MyProvider } from './MyContext';
function MyApp({ Component, pageProps }) {
 return (
    <MyProvider>
      <Component {...pageProps} />
    </MyProvider>
export default MyApp;
```



```
// AnyComponent.js
import React from 'react';
import { useMyContext } from './MyContext';
function AnyComponent() {
  const sharedData = useMyContext();
  return {sharedData};
}
export default AnyComponent;
```

#### References



- 1. W3Schools Online Web Tutorials, URL: <a href="http://www.w3schools.com">http://www.w3schools.com</a>
- 2. Next.js, URL: <a href="https://nextjs.org/">https://nextjs.org/</a>
- 3. Mozilla Developer Networks, URL: <a href="https://developer.mozilla.org/">https://developer.mozilla.org/</a>



### Thank You!