Working With Authentication

Authentication and authorization are important for creating secure and user-friendly applications in Angular. Authentication is the process of verifying a user's identity, while authorization is the process of identifying the level of access a user has.

The sample project we are building with Angular 18 and featured user authentication & authorization flows, protected routes, etc.

Just to keep things simple, I am considering logins in two ways,

1. Admin/User Oriented app, only authenticated users can go in. So, login should appear first and then based on Authentication-Authorization navigating to other components. We mostly manage using routing.
2. Website with Login as an optional in app. Few links are accessible anonymously and few will be activated based on Authentication.

Here we are going to start with first method in which we will be using Hard Coded / Static username and password. Let's create an Angular application, I named it MyAuthDemoApp

> ng new MyAuthDemoApp

Open application in Visual Studio Code and open terminal into it. Now install bootstrap as usual:

..\MyAuthDemoApp> npm i bootstrap

Go to angular.json and make changes to make bootstrap available to entire application.

```
"styles": ["node_modules/bootstrap/dist/css/bootstrap.css",
              "src/styles.css"
            ],
            "scripts":
["node_modules/bootstrap/dist/js/bootstrap.js"]
          },
```

Go to the app folder add a new folder named components. Use right-click on it and use Open in integrated terminal. Add few components.

>……\MyAuthAppDemo\src\app\components> ng g c addEmp

>……\MyAuthAppDemo\src\app\components> ng g c listEmp

>……\MyAuthAppDemo\src\app\components> ng g c showEmpById

>……\MyAuthAppDemo\src\app\components> ng g c editEmp

>……\MyAuthAppDemo\src\app\components> ng g c deleteEmp

We will be using these in Authentication-Authorization. You may create them as in previous example to perform crud operations or add login feature to an existing application having few components.

As we decided to go with first method, Login component need to appear first. Let's create login component first

>……\MyAuthAppDemo\src\app\components> ng g c login

Now define routes in app.routes.ts file.

```
import { Routes } from '@angular/router';
import { LoginComponent } from './components/login/login.component';

export const routes: Routes = [
    //defualt route
    {
        path: '',
        redirectTo: 'login',
        pathMatch: 'full'
    },
    {
        path: 'login',
        component: LoginComponent
    },
];
```

Test it as application is loaded it will go to Login Component.

Now create one component that will appear only after logging in. Name it layout.

>……\MyAuthAppDemo\src\app\components> ng g c layout

Here in this example, I am adding a NavBar.

Go to layout.component.html

```
<nav class="navbar navbar-expand-sm navbar-dark bg-dark">
  <div class="container-fluid">
    <a class="navbar-brand" href="javascript:void(0)">Logo</a>
    <button
      class="navbar-toggler"
      type="button"
      data-bs-toggle="collapse"
      data-bs-target="#mynavbar"
    >
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="mynavbar">
      <ul class="navbar-nav me-auto">
        <li class="nav-item">
          <a class="nav-link" routerLink="add-emp">AddNew</a>
        </li>
```

```
        <li class="nav-item">
          <a class="nav-link" routerLink="edit-emp">Modify</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" routerLink="delete">DeleteEmp</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" routerLink="list-emp"> AllEmps</a>
        </li>
      </ul>
      <form class="d-flex">
        <span class="text-white">LoggedUser</span>  
        <button class="btn btn-primary"
type="button">LogOff</button>
      </form>
    </div>
  </div>
</nav>

<div class="container">
  <router-outlet></router-outlet>
</div>
```

Also add respective imports in layout.component.ts

```
import { Component } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { Router, RouterLink, RouterOutlet } from '@angular/router';
@Component({
  selector: 'app-layout',
  standalone: true,
  imports: [RouterOutlet, RouterLink, FormsModule],
  templateUrl: './layout.component.html',
  styleUrl: './layout.component.css'
})
export class LayoutComponent {
}
```

Add routes as children of Layout. Currently I am adding all my link of navbar.

Modify app.routes.ts

```
import { Routes } from '@angular/router';
```

```
import { LoginComponent } from './components/login/login.component';
import { LayoutComponent } from
'./components/layout/layout.component';
import { AddEmpComponent } from './components/add-emp/add-
emp.component';
import { ShowEmpByIdComponent } from './components/show-emp-by-
id/show-emp-by-id.component';
import { ListEmpComponent } from './components/list-emp/list-
emp.component';
import { DeleteEmpComponent } from './components/delete-emp/delete-
emp.component';
import { EditEmpComponent } from './components/edit-emp/edit-
emp.component';

export const routes: Routes = [
    //defualt route
    {
        path: '',
        redirectTo: 'login',
        pathMatch: 'full'
    },
    {
        path: 'login',
        component: LoginComponent
    },
    {
        path: '',
        component: LayoutComponent,
        children: [
            {
                path: 'add-emp',
                component: AddEmpComponent
            },
            {
                path: 'edit-emp',
                component: EditEmpComponent
            },
            {
                path: 'delete-emp',
                component: DeleteEmpComponent
            },
```

```
            {
                path: 'list-emp',
                component: ListEmpComponent
            },
            {
                path: 'show-emp-by-id',
                component: ShowEmpByIdComponent
            },
        ]
    }
];
```

Now let's design Login Form. As usual you can use some template from Codepen or any other website. Choose template which uses only HTML and CSS. I selected one and using here.

Before design, we need object respective to login form.

Go to login.component.ts and add userObj

```
userObj: any = {
    EmailId:'',
    Password:''
  };
```

Go to login.component.html and bind fields.

Let's add styles of template first in login.component.css

```
.parent {
    font-family: 'Arial', sans-serif;
    height: 100vh;
    background: linear-gradient(135deg, #ff6f61, #ffbc67);
    display: flex;
    justify-content: center;
    align-items: center;
}

.container-login {
    background: rgba(255, 255, 255, 0.9);
    padding: 2rem;
    border-radius: 10px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    animation: fadeIn 1.5s ease-in-out;
}

.signin-form {
```

```
    width: 300px;
}

h2 {
    text-align: center;
    margin-bottom: 1.5rem;
    font-size: 2rem;
    color: #333;
    font-weight: bold;
}

.input-group {
    margin-bottom: 1rem;
}

.input-group label {
    display: block;
    font-size: 1rem;
    color: #333;
    margin-bottom: 0.5rem;
}

.input-group input {
    width: 100%;
    padding: 0.5rem;
    border: 1px solid #ddd;
    border-radius: 5px;
    font-size: 1rem;
    transition: border 0.3s;
}

.input-group input:focus {
    border: 1px solid #ff6f61;
    outline: none;
}

button {
    width: 100%;
    padding: 0.75rem;
    border: none;
    border-radius: 5px;
```

```
    background: #ff6f61;
    color: white;
    font-size: 1.25rem;
    cursor: pointer;
    transition: background 0.3s, transform 0.3s;
}

button:hover {
    background: #ff3e2d;
    transform: scale(1.05);
}

@keyframes fadeIn {
    from {
        opacity: 0;
        transform: translateY(-10px);
    }
    to {
        opacity: 1;
        transform: translateY(0);
    }
}
```

Login.component.html

```
<div class="parent">
    <div class="container-login">
        <div class="signin-form">
            <h2>Sign In</h2>
            <form>
                <div class="input-group">
                    <label for="username">Username</label>
                    <input [(ngModel)]="userObj.EmailId" type="text"
id="username" name="EmailId" required>
                </div>
                <div class="input-group">
                    <label for="password">Password</label>
                    <input
[(ngModel)]="userObj.Password"  type="password" id="password"
name="Password" required>
```

```
                </div>
                <button type="button" (click)="onLogin()">Sign
In</button>
            </form>
        </div>
    </div>
</div>
```

Add event to button so on click it will login.

Go to login.component.html and add click event binding to a method.

```
<button type="button" (click)="onLogin()">Sign In</button>
```

Go to login.component.ts and add method onLogin.

Before create router instance

```
 router = inject(Router)
```

Now add method (complete .TS as of now for Reference)

```
import { Component, inject } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { Router } from '@angular/router';

@Component({
  selector: 'app-login',
  standalone: true,
  imports: [FormsModule],
  templateUrl: './login.component.html',
  styleUrl: './login.component.css'
})
export class LoginComponent {
  userObj: any = {
    EmailId: '',
    Password: ''
  };
  router = inject(Router)
  onLogin() {
    if (this.userObj.EmailId == "admin" && this.userObj.Password ==
"1234") {
      alert("login Success");
      localStorage.setItem('loginUser', this.userObj.EmailId)
      this.router.navigateByUrl('add-emp')
    } else {
      alert('Wrong Credentials')
```

```
    }
  }
}
```

Test application.

Now add LogOff button functionality also.

Go to layout.html and add onClick to LogOff button

```
<form class="d-flex">
        <span class="text-white">LoggedUser</span>
        <button class="btn btn-primary" type="button"
(click)="logOff()">
          LogOff
        </button>
      </form>
```

Add logoff code in layout.component.ts file.

```
 logOff() {
    localStorage.removeItem('loginUser');
    this.router.navigateByUrl('login');
  }
```

Now I want to display the logged in user name of the NavBar

Go to layout.component.ts and add constructor with on property

```
loggedUserData: any;
  constructor() {
    const loggedData = localStorage.getItem("loginUser");
    if (loggedData != null) {
      this.loggedUserData = loggedData;
    }
  }
```

Change layout.component.html

```
<form class="d-flex">
        <span class="text-white">Hello {{ loggedUserData }}
</span>  
        <button class="btn btn-primary" type="button"
(click)="logOff()">
          LogOff
        </button>
      </form>
```

Test Application.

☸✠End of Chapter✠☸