

## MỤC TIÊU:

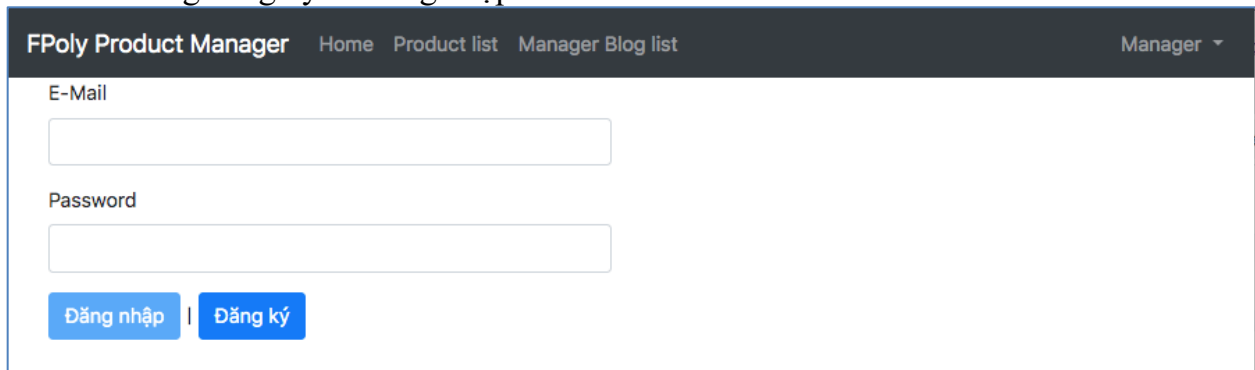
Kết thúc bài thực hành này bạn có khả năng

- ✓ Hiểu và cài đặt Authentcation trong angular
- ✓ Signup, Login và xử lý lỗi Login
- ✓ Lưu trữ User, thêm Token, cài đặt login, logout tự động, phân quyền

## PHẦN I

### Bài 1 (2 điểm)

Thiết kế trang đăng ký và đăng nhập như sau



Hướng dẫn

Tạo component auth và thiết kế giao diện và điều khiển form như sau

```
<div class="row">
  <div class="col-xs-12 col-md-6 col-md-offset-3">
    <form #authForm="ngForm" (ngSubmit)="onSubmit(authForm)">
      <div class="form-group">
        <label for="email">E-Mail</label>
        <input
          type="email"
          id="email"
          class="form-control"
          ngModel
          name="email"
          required
          email
        />
      </div>
      <div class="form-group">
        <label for="password">Password</label>
        <input
```

```

        type="password"
        id="password"
        class="form-control"
        ngModel
        name="password"
        required
        minlength="5"
      />
    </div>
    <div>
      <button
        class="btn btn-primary"
        type="submit"
        [disabled]="!authForm.valid"
      >
        Đăng nhập
      </button>
      |
      <button class="btn btn-primary" type="button">
        Đăng ký
      </button>
    </div>
  </form>
</div>
</div>

```

Sự kiện submit:

```

onSubmit(form: NgForm) {
  console.log(form.value);
  form.reset();
}

```

Thêm route /auth và Xem kết quả:localhost:4200/auth

## Bài 2 (2 điểm)

Xây dựng chức năng đăng ký

The screenshot shows the 'FPoly Product Manager' application interface. At the top, there is a navigation bar with links for 'Home', 'Product list', and 'Manager Blog list', along with a 'Manager' dropdown menu. The main content area displays a form for user authentication. It includes two input fields: 'E-Mail' and 'Password'. Below these fields are two buttons: 'Đăng nhập' (Login) and 'Đăng ký' (Register), separated by a vertical line. The form is styled with a dark header and a light background for the input fields.

Hướng dẫn:

Tạo service có tên authservice: service này làm việc với API (sử dụng API đã học NodeJS)

Thêm hàm:

```
url='http://127.0.0.1:3000/auth/';
signUp(email:string,pass:string){
  return this.http.post(this.url+'register',
    {
      "email": email,
      "password": pass,
      "typeUser":0
    }
  );
}
```

Tại component Auth thực hiện gọi hàm signUp cho sự kiện submit của form

```
onSubmit(form: NgForm) {
  let email=form.value.email;
  let password=form.value.password;
  this.error=null;
  // //Dang ky
  this.authService.signUp(email,password).subscribe(res=>{
    console.log(res);
  },
  error=>{
    console.log(error);
    this.error = error.error.message;
  }
  )
  // //end dang ký

  form.reset();
}
```

Chạy và xem kết quả

### Bài 3 (2 điểm)

Thực hiện đăng nhập và lưu trữ token:

```
onSubmit(form: NgForm) {
  let email=form.value.email;
  let password=form.value.password;
  this.error=null;
  //Dang nhap
  this.authService.signIn(email,password).subscribe(res=>{
    this.authService.setToken(res.accessToken);
    console.log(authService.getToken());
  },
  error=>{
```

```
    console.log(error);  
    this.error = error.error.message;  
  }  
)  
//end dang nhap  
}
```

Authservice bổ sung các hàm sau:

```
export class AuthServiceService {  
  url='http://127.0.0.1:3000/auth';  
  token:string=null;  
  constructor(private http:HttpClient) {}  
  setToken(token:string){  
    this.token=token;  
  }  
  getToken(){  
    return this.token;  
  }  
  signUp(email:string,pass:string){  
    return this.http.post(this.url+'register',  
      {  
        "email": email,  
        "password": pass,  
        "typeUser":0  
      })  
  };  
  signIn(email:string,pass:string){  
    return this.http.post(this.url+'login',  
      {  
        "email": email,  
        "password": pass,  
        "typeUser":0  
      })  
  };  
}
```

Chạy và kiểm tra kết quả

## PHẦN II

### Bài 4 (2 điểm)

Thực hiện các chức năng quản lý bài viết (đọc, thêm, xoá, sửa post) với service cho http request kèm theo header là token để xác thực đã thực hiện đăng nhập vào hệ thống

**Hướng dẫn:**

- Tại các tác vụ get, post, put, delete của post service, thực hiện bổ sung thêm phần header chứa token cho server xác thực JWT để phản hồi kết quả phù hợp
- Code cho service post

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { IPost } from './entities/post';

@Injectable({
  providedIn: 'root'
})
export class PostService {

  url='http://127.0.0.1:3000/blog/posts/';

  constructor(private httpService:HttpClient) { }

  storePost(postData:IPost){
    this.httpService.post(this.url,postData).subscribe(p=>{
      console.log(p);
    })
  }

  getAllPost(){
    return this.httpService.get(this.url,{headers: new HttpHeaders()
    .set('x-access-token', this.authService.getToken())
    });
  }

  deletePost(id:number){
    return this.httpService.delete(this.url+id, {headers: new HttpHeaders()
    .set('x-access-token', this.authService.getToken())
    });
  }
}
```

- Chạy và kiểm tra kết quả xử lý khi nhận submit với console của trình duyệt

## Bài 5 (2 điểm)

Sử dụng Guard:

- Quyết định route dựa trên thời gian hết hạn của token
  - Xây dựng route Guard để xác định quyền có được phép truy cập route hay không, tạo service có tên authGuard và cài đặt như sau

```
export class AuthGuardService implements CanActivate {
  constructor(public auth: AuthServiceService, public router: Router) {}

  canActivate(): boolean {
    if (!this.auth.isAuthenticated()) {
```

```
this.router.navigate(['auth']);  
return false;  
}  
return true;  
}  
}
```

## 2. Kiểm tra và phân quyền User

- Tạo component vùng admin, và route:

```
{  
  path:'admin', component:DashboardComponent,  
  canActivate: [RoleGuard],  
  data: {  
    role: 1  
  }  
}
```

- Xây dựng role Guard để phân quyền, tạo service có tên roleGuard và cài đặt như sau

```
export class RoleGuardService implements CanActivate {  
  
  constructor(private auth: AuthServiceService, private router: Router) {}  
  canActivate(route: ActivatedRouteSnapshot): boolean {  
    //lay role duoc truong tai route  
    const role = route.data.role;  
    const token = this.auth.getToken();  
    // decode the token to get its payload  
    var tokenPayload:any;  
    if(token){  
      tokenPayload = decode(token);  
    }  
    if (  
      !this.auth.isAuthenticated() || tokenPayload.typeUser !== role  
    ) {  
      this.router.navigate(['auth']);  
      return false;  
    }  
    return true;  
  }  
}
```

**\*\*\* Yêu cầu nộp bài:**

SV nén file (*hoặc share thư mục google drive*) bao gồm các yêu cầu đã thực hiện trên, nộp LMS đúng thời gian quy định của giảng viên. KHÔNG NỘP BÀI COI NHƯ KHÔNG CÓ ĐIỂM.

--- Hết ---