Angular Authentication: SignIn template

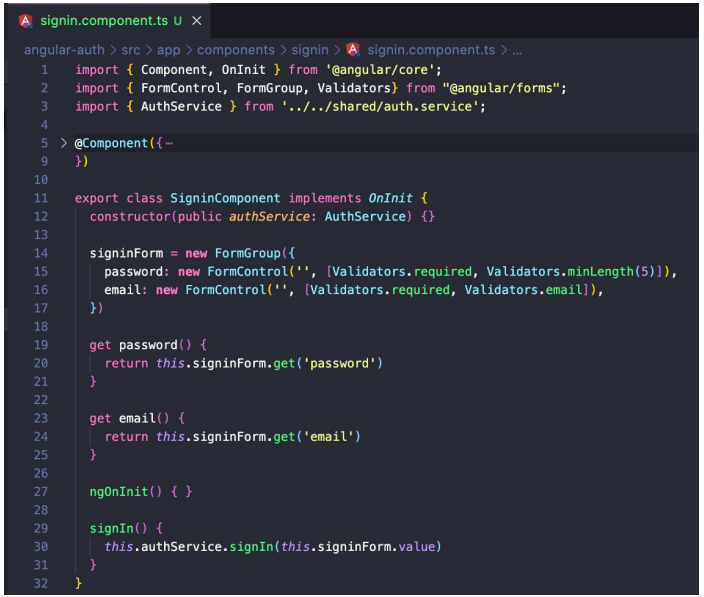
signin.component.html U

angular-auth > src > app > components > signin > signin.component.html > div.auth-wrapper > form.form-signin

```
<div class="auth-wrapper">
  <form class="form-signin" [formGroup]="signinForm" (ngSubmit)="signIn()">
      <h3 class="h3 mb-3 font-weight-normal text-center">Please sign in</h3>
      <div class="form-group mt-3">
          <label>Email address</label>
          <input type="email" class="form-control" formControlName="email" placeholder="Enter email" required>
          <span style="color: red;"
              *ngIf="email && email.touched && email.invalid">
              Email is required. Must input email type
          </span>
      </div>
      <div class="form-group mt-3">
          <label>Password</label>
          <input type="password" class="form-control" formControlName="password" placeholder="Password" required>
          <span style="color: red;"
              *ngIf="password && password.touched && password.invalid">
              Password is required. Min length is 5
          </span>
      </div>
      <button type="submit" class="btn btn-block btn-primary mt-3">Sign in</button>
  </form>
</div>
```

Buat template form sign-in

Angular Authentication: SignIn class component



Dalam file **signin.component.ts**,

1. Import authservice dan class-class yang dibutuhkan
2. Buatlah signInForm yang mempunyai password dan email
3. Getter password dan email untuk menampilkan validasi di templaye
4. Method **signIn()** akan memanggil method signin dalam **authservice**, dan mengirim parameter value dari signinForm

Angular Authentication: SignIn Service

```
import { catchError, map } from 'rxjs/operators';
import { HttpClient, HttpHeaders,
         HttpErrorResponse } from '@angular/common/http';
import { Router } from '@angular/router';
```

```
endpoint: string = 'http://localhost:4000/api';
headers = new HttpHeaders().set('Content-Type', 'application/json');
currentUser: {name: string, email: string, _id: string} = {name: '', email: '', _id: ''}
```

```
constructor(private http: HttpClient, private router: Router) {}
```

```
// Sign-in
signIn(user: User) {
  return this.http.post<any>(`${this.endpoint}/signin`, user)
    .subscribe((res: any) => {
      localStorage.setItem('access_token', res.token)
      this.getUserProfile(res._id).subscribe((res:any) => {
        this.currentUser = res;
        this.router.navigate(['user-profile/' + res.msg._id]);
      })
    })
}
```

auth.service.ts

Dalam file **auth.service.ts**,

1. Import map, httpHeaders dan router..
2. Buat properti **currentUser** dan **headers** yang berisi. **HttpHeaders**
3. Tambahkan/Inject di parameter constructor router yang bertipe **Router**
4. **SignIn** method menerima parameter user, yang akan dikirim ke server method POST dengan path /signin.
   Didalam method **subscribe**, set token dari response ke access_token dalam localstorage.

   Kemudian, panggil lagi method **getUserProfile** yang mem-fetch user profile.
   Navigasi akan menuju ‘user-profile/id’ bila login sukses

Angular Authentication: SignIn > getUserProfile

```
// User profile
getUserProfile(id:any): Observable<any> {
  let api = `${this.endpoint}/user-profile/${id}`;
  return this.http.get(api, { headers: this.headers })
  .pipe(
    map((res: any) => {
      return res || {}
    }),
    catchError(this.handleError)
  )
}
```

auth.service.ts

Masih Dalam file **auth.service.ts**,
Buatlah method **getUserProfile** yang menerima parameter id.
Hit ke endpoint server /user-profile/id dengan method get,
yang membawa **headers**.

Sampai tahap ini, request ke user-profile masih unauthorized, karena belum mengirim access token ke server.

Selanjutnya, kita akan men-set access-token dalam headers dengan HTTPInterceptors.

Angular Authentication: HttpInterceptor - Set header token

```
getToken() {
  return localStorage.getItem('access_token');
}
```

auth-interceptor.service.ts U

angular-auth > src > app > shared > auth-interceptor.service.ts > ...

```
import { Injectable } from "@angular/core";
import { HttpInterceptor, HttpRequest, HttpHandler } from "@angular/common/http";
import { AuthService } from "./auth.service";

@Injectable({
    providedIn: 'root'
  })

export class AuthInterceptorService implements HttpInterceptor {
    constructor(private authService: AuthService) { }

    intercept(req: HttpRequest<any>, next: HttpHandler) {
        const authToken = this.authService.getToken();
        req = req.clone({
            setHeaders: {
                Authorization: "Bearer " + authToken
            }
        });
        return next.handle(req);
    }
}
```

▾ **Request Headers** View source

Accept: application/json, text/plain, */*

Accept-Encoding: gzip, deflate, br

Accept-Language: en-US,en;q=0.9,id;q=0.8,ms;q=0.7

Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6ImFkbWluQG1haWwuY29tI
IjoiNjE1YzIzOTlmZGUzMTU0MGZjYTczM2JmIiwiaWF0IjoxNjMzNDMxMjY0LCJleHAiOjE2MzM0MzQ4NjR9.OI_sY
83H4tR-hDrQ51mmEsQNCv2M7z68

Connection: keep-alive

Dalam file **auth.service.ts,** buat method getToken yang mengambil access_token dari localStorage.

HttpInterceptor akan menyisipkan proses sebelum request dilakukan

Buat 1 service baru, yang didalamnya:

1. meng-import HttpInterceptor, HttpRequest, HttpHandler dan AuthService.
2. Method **intercept** menerima 2 params req httprequest dan next httphandler.
3. Buat variable authToken yang didalamnya memanggil method **getToken** dari service authservice.
4. Method **req** akan men-set headers Authorization dengan **authToken** yang diambil sebelumnya
5. **next** httpHandler akan meneruskan request selanjutnya.

Angular Authentication: User Profile class Component

user-profile.component.ts U

angular-auth > src > app > components > user-profile > user-profile.compone

```ts
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { AuthService } from '../../shared/auth.service';

@Component({…
})

export class UserProfileComponent implements OnInit {
  currentUser: {
    name: string,
    email: string,
    _id: string
  } = {name: '', email: '', _id: ''}

  constructor(
    public authService: AuthService,
    private actRoute: ActivatedRoute
  ) {}


  ngOnInit() {
   this.setCurrentUser()
  }

  setCurrentUser() {
    let id = this.actRoute.snapshot.paramMap.get('id');

    if (!this.authService.currentUser._id) {
      this.authService.getUserProfile(id).subscribe(res => {
        this.currentUser = res.msg;
      })
    } else {
      const {name, email, _id} = this.authService.currentUser
      this.currentUser = {name, email, _id}
    }
  }
}
```

Dalam user profile, di ngOnInit, buatlah 1 properti currentUser yang akan diset isinya di hooks ngOnInit.

Service **getUserProfile** akan di-fetch jika currentUser dalam AuthService belum tersedia.

Angular Authentication: User Profile template

user-profile.component.html U

angular-auth > src > app > components > user-profile > user-profile.component.ht

```
<div class="row">
    <div class="inner-main">
        <h2 class="mb-4">User Profile</h2>
        <p><strong>Name:</strong> {{this.currentUser.name}}</p>
        <p><strong>Email:</strong> {{this.currentUser.email}}</p>
    </div>
</div>
```

Buatlah template user profile yang menampilkan nama dan email dari user yang sudah login.

Halaman atau path User Profile bisa diakses sebelum dan setelah login.

Namun, Jika ingin halaman user profile diakses hanya setelah login, kita harus memproteksi route dengan canActive.

Angular Authentication: Routes Guard dengan canActive

```
get isLoggedIn(): boolean {
  let authToken = localStorage.getItem('access_token');
  return (authToken !== null) ? true : false;
}
```

auth.service.ts

Buatlah satu getter isLoggedIn yang akan me-return true/false dengan mengecek ada tidaknya access_tokeen di localstorage.

canActive merupakan salah satu route guards yang disediakan Angular. DIgunakan untuk mencegah akses user yang tidak ter-otorisasi, ke dalam routing tertentu.

```
❯ ng g guard shared/auth
? Which interfaces would you like to implement? CanActivate
CREATE src/app/shared/auth.guard.spec.ts (331 bytes)
CREATE src/app/shared/auth.guard.ts (457 bytes)
```

Untuk menambahkan guard, gunakan command:

`ng g guard nama_guard`

Angular Authentication: Routes Guard dengan canActive

auth.guard.ts U

angular-auth > src > app > shared > auth.guard.ts > ...

```
1  import { Injectable } from '@angular/core';
2  import { ActivatedRouteSnapshot, CanActivate, RouterStateSnapshot, UrlTree, Router } from '@angular/router';
3  import { Observable } from 'rxjs';
4  import { AuthService } from './auth.service';
5
6  @Injectable({
7    providedIn: 'root'
8  })
9  export class AuthGuard implements CanActivate {
10   constructor(
11     public authService: AuthService,
12     public router: Router
13   ) { }
14
15   canActivate(
16     route: ActivatedRouteSnapshot,
17     state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {
18       if (!this.authService.isLoggedIn) {
19         window.alert("Access not allowed!");
20         this.router.navigate(['login'])
21       }
22       return true;
23   }
24 }
```

Setelah men-generate auth.guard, maka import module-module yang dibutuhkan. Dan didalam method **canActivate,** navigasikan ke path login, apabila belum login. (pengecekan dengan memanggil getter isLogged yang dibuat sebelumnya)

Angular Authentication: Daftarkan Routes Guard

app-routing.module.ts M

angular-auth > src > app > app-routing.module.ts > ...

```
1  import { NgModule } from '@angular/core';
2  import { RouterModule, Routes } from '@angular/router';
3  import { SigninComponent } from './components/signin/signin.component';
4  import { SignupComponent } from './components/signup/signup.component';
5  import { UserProfileComponent } from './components/user-profile/user-profile.component';
6
7  import { AuthGuard } from './shared/auth.guard';
8
9  const routes: Routes = [
10   { path: '', redirectTo: '/login', pathMatch: 'full' },
11   { path: 'login', component: SigninComponent },
12   { path: 'register', component: SignupComponent },
13   { path: 'user-profile/:id', component: UserProfileComponent, canActivate: [AuthGuard] }
14 ];
15
16 @NgModule({
17   imports: [RouterModule.forRoot(routes)],
18   exports: [RouterModule]
19 })
20 export class AppRoutingModule { }
```

Cantumkan AuthGuard di path yang ingin dijaga aksesnya. Contoh di atas mencantumkan AuthGuard yang dibuat sebelumnya di path user-profile.

Maka dengan demikian, path /user-profile hanya bisa diakses bila user sudah login.