Angular Auth+Rout protection 20

Session_20_Auth_Routing(authentication-final)
249 FrontEnd- -BackEnd communication
251 Setup FIreBase SDk (Authentication)
253.SingIn
254.Requiring Token (from firebase)
255.Sending a Token
256.Check if a user is Authenticated and display
257.Add a Logout Button
258.Route Protection with guards
260.Improvements Opprotunities

249 FrontEnd- -BackEnd communication

More information on the webtoken
https://jwt.io/introduction/

251 Setup FIreBase SDk (Authentication)

Authentication/Sign-In methods - Email/Password

1. install Firebase package:
- **npm install firebase ---save**
2. Get your Authenticaton Api + domain name from firebase and initiate when the main component is loaded
app.component.ts

```
import { Component, OnInit } from '@angular/core';
import * as firebase from 'firebase';


@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit {
  loadedFeature = 'recipe';
```

```
// You can get it from the Authentication/web setup, firebase
  ngOnInit() {
    firebase.initializeApp({              //After it has been imported, we can use it's funcito
ns
    apiKey: "xxxx",
    authDomain: "xxx",
    });
  }
  onNavigate(feature: string) {
    this.loadedFeature = feature;
  }
}
```

2. Create an authentication service, with a functions passing the necessary parameters to the firebase package.

auth.service.ts

```
import { Router } from '@angular/router';            //Needed for navigation
import * as firebase from 'firebase';                //Needed for the firebase functions
import { Injectable } from '@angular/core';          //Nedded to use the router service

@Injectable()
export class AuthService {
  token: string;

  constructor(private router: Router) {}
  //Basically a simple function which takes two parameters + and pass it to the firebase functi
o
  signupUser(email: string, password: string) {
    //This is a built in function which is returning a promise, which can be catched.
    firebase.auth().createUserWithEmailAndPassword(email, password)
      .catch(
        error => console.log(error)
```

```
    )
}

signinUser(email: string, password: string) {
  firebase.auth().signInWithEmailAndPassword(email, password)
    .then(
      response => {
        this.router.navigate(['/']);
        firebase.auth().currentUser.getToken()
          .then(
            (token: string) => this.token = token
          )
      }
    )
    .catch(
      error => console.log(error)
    );
}

logout() {
  firebase.auth().signOut();
  this.token = null;
}

getToken() {
  firebase.auth().currentUser.getToken()
    .then(
      (token: string) => this.token = token
    );
  return this.token;
}
```

```
  isAuthenticated() {
    return this.token != null;
  }
}
```

3. On **signUp/signIn component.html + component.ts** create a simple click listener *(ngSubmit)* + an event which passes the data from the Form *(#f="ngForm")* to the appropriate functions, *onSignup(f)* which are calling the injsected **auth services**

signUp.component.html

```
<form (ngSubmit)="onSignup(f)" #f="ngForm">
```

signUp.Component.ts

```
  onSignup(form: NgForm) {
    const email = form.value.email;
    const password = form.value.password;
    this.authService.signupUser(email, password);
  }
```

253. SingIn

1. Have a seperate route for the sign in.
app-routing.module.ts

```
  { path: 'signin', component: SigninComponent },
```

2. Have a component with html + Form, so you can collect the data

```
<div class="row">
  <div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-offset-2">
    <form (ngSubmit)="onSignin(f)" #f="ngForm">
      <div class="form-group">
        <label for="email">Mail</label>
        <input type="email" id="email" name="email" ngModel class="form-control">
```

```
      </div>
      <div class="form-group">
        <label for="password">Password</label>
        <input
          type="password"
          id="password"
          name="password"
          ngModel
          class="form-control">
      </div>
      <button class="btn btn-primary" type="submit" [disabled]="!f.valid">Sign In</button>
    </form>
  </div>
</div>
```

2. Inject `Autservice` to the **signin.component.ts +** pass the collected variables from the Form to the autservice functions

```
import { Component, OnInit } from '@angular/core';
import { NgForm } from '@angular/forms';
import { AuthService } from '../auth.service';

@Component({
  selector: 'app-signin',
  templateUrl: './signin.component.html',
  styleUrls: ['./signin.component.css']
})
export class SigninComponent implements OnInit {
  constructor(private authService: AuthService) { }
  ngOnInit() {
  }
  onSignin(form: NgForm) {
    const email = form.value.email;
```

```
    const password = form.value.password;
    this.authService.signinUser(email, password);
  }
}
```

3. Then in the Auth.service.ts after injecting firebase,you can use it's functions and pass the variables from the component

```
import { Router } from '@angular/router';
import * as firebase from 'firebase';
import { Injectable } from '@angular/core';

@Injectable()
export class AuthService {
  token: string;

  constructor(private router: Router) {}

  signupUser(email: string, password: string) {
    //This is a built in function which is returnin a prmise, which can be catched.
    firebase.auth().createUserWithEmailAndPassword(email, password)
      .catch(
        error => console.log(error)
      )
  }
//This is a promise as well, so there is a callback upon completion .then + .catch error handli
ng
  signinUser(email: string, password: string) {
    firebase.auth().signInWithEmailAndPassword(email, password)
      .then(
        response => {   console.log(ressponse);      })
      .catch(
        error => console.log(error)
      );
```

```
  }
}
```

4. The recieved user Token can be seen in the **Application/localStorage** of the web app

```
firebase:authUser:AIzaSyC5alS-cEZFulqPzxt2--MMw1PcqV3XPak:[DEFAULT]
{
   "uid":"id8bvFSMhYVUKxNt6nQowB3WDOU2",
   "displayName":null,
   "photoURL":null,
   "email":"laczor@test.com",
   "emailVerified":false,
   "phoneNumber":null,
   "isAnonymous":false,
   "providerData":[
      {
         "uid":"laczor@test.com",
         "displayName":null,
         "photoURL":null,
         "email":"laczor@test.com",
         "phoneNumber":null,
         "providerId":"password"
      }
   ],
   "apiKey":"xxx",
   "appName":"[DEFAULT]",
   "authDomain":"xxx",
   "stsTokenManager":{
      "apiKey":"xxx",
      "refreshToken":"APWA_krzbLGEMrW401WNA-AuFE5Qxt4GUaXjnqcQW17UUVcGP2OD2xNSKIHuY9nHrbeyDJHw35YL-q5m9K_seSAcla6Dp7-0S6hgKNzw2bcWWRW9dcFBPso4uPfNvxKa6-gjeHBhdpFfd90mVAXwRfZ31PZnzM_Uamw2IGQt3Csoy93b8t-9YZ5mEPD8BrnCGtVvj3iWNfs9riK-uYQyt1dIuATck21_Sw",
```

```
      "accessToken":"eyJhbGciOiJSUzI1NiIsImtpZCI6IjRmMDYzNDJiNDdjYTQ1Zjg0NjM2MTk0NjE5MjNiMDdjYzQ4OTA1N2UifQ.eyJpc3MiOiJodHRwczovL3NlY3VyZXRva2VuLmdvb2dsZS5jb20vdWRlbXktbmctaHR0cC1kZGVkNSIsImF1ZCI6InVkZW15LW5nLWh0dHAtZGRlZDUiLCJhdXRoX3RpbWUiOjE1MTE1MzYzNDgsInVzZXJfaWQiOiJpZDhidkZTTWhZVlVLeE50Nm5Rb3dCM1dET1UyIiwic3ViIjoiaWQ4YnZGU01oWVZVS3hOdDZuUW93QjNXRE9VMiIsImlhdCI6MTUxMTUzNjM0OCwiZXhwIjoxNTExNTM5OTQ4LCJlbWFpbCI6ImxhY3pvckB0ZXN0LmNvbSIsImVtYWlsX3ZlcmlmaWVkIjpmYWxzZSwiZmlyZWJhc2UiOnsiaWRlbnRpdGllcyI6eyJlbWFpbCI6WyJsYWN6b3JAdGVzdC5jb20iXX0sInNpZ25faW5fcHJvdmlkZXIiOiJwYXNzd29yZCJ9fQ.Ykm62LeIfjcKgJLwG3nziYybU6-qZPU6aISuHEqNsAY6I_qHwCrVHHjEFw3UAc_GIqXBNiVkqn8NQ-9UjKU-Aqr_Y9F-o6yXqv9ZLfAbc2BStCONfD9j75S_DM6hrHVN3p4OF4bO5IvH0fTRuGdBXbyLc2f9sMxgR8wr9MgopwuKKV4UCLbPWhtBBKCNEqsstlP9IA_c8kWt6LmOpIG_xdqsmlZLcOMtcZH7_IiDvYsccDVCf-jZtj1_mGR2U3hEdoGFbfgQOx0anzH-ao0jc2Smd95wWuqmfFqwxw1eB0XrohvbCpZu9gGHpsXz0if2NkFFfs0BOhzChpuPTXu_gg",
      "expirationTime":1511539950201
   },
   "redirectEventId":null,
   "lastLoginAt":"1511536348000",
   "createdAt":"1511536348000"
}
```


254. Requiring Token (from firebase)

1. Ensue only authenticated users can modify anything at firebase
Databases/Rules

```
{
  "rules": {
    ".read": "auth!= null",
    ".write": "auth!= null"
  }
}
```

255. Sending a Token

1. Get a token and store it in the **Authservice.ts** as a component variable
2. Whe the **user sign's in,** and we got a response, let's immediately start asking for token, with **getToken(),** and when that is executed, we will store the recieved token at the component's variable
auth.service.ts

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

```
import * as firebase from 'firebase';
import { Injectable } from '@angular/core';

@Injectable()
export class AuthService {
  token: string;

  constructor(private router: Router) {}
//This is a promise as well, so there is a callback upon completion .then + .catch error handli
ng
  signinUser(email: string, password: string) {
    firebase.auth().signInWithEmailAndPassword(email, password)
      .then(
        response => {
                    firebase.auth().currentUser.getToken()
            .then(
              (token: string) => this.token = token
            )
        }
      )
      .catch(
        error => console.log(error)
      );
  }

getToken() {
//This is an async function wich wll return a promise.
    firebase.auth().currentUser.getToken()
      .then(
        (token: string) => this.token = token
      );
    return this.token;
```

```
  }
}
```

3. When the data is request from the server **(Post, Put, Get**) then the token has to be requested, (since we are requesting when the user sign's in + when the request is made, maybe timing delays in the token can be experienced, but for most usecases it is fine.)
Important, the token is parsed as a **queryParameter**

```
https://ng-recipe-book.firebaseio.com/recipes.json?auth= token
```

data-storage.service.ts

```
  storeRecipes() {
    const token = this.authService.getToken();
    return this.http.put('https://ng-recipe-book.firebaseio.com/recipes.json?auth=' + token, th
is.recipeService.getRecipes());
  }
```

256. Check if a user is Authenticated and display

1. check if the user has a token or not in the **auth-service.ts** and return it if it is not null, which means the user is logged in.

```
 isAuthenticated() {
    return this.token != null;
  }
```

2. Inject the **Auth service** in the header.component.ts and use it in it's html, to execute it directly, using with *ngIf template
header.component.ts

```
import { AuthService } from '../auth/auth.service';


export class HeaderComponent {
  constructor(private dataStorageService: DataStorageService,
              private authService: AuthService) {
  }
}
```

Note: - We are using the template function, which will enable us to show the element in condition to a variable
- here the variable, is an **immediately executed injected service function**
header.component.html

```
 <ng-template [ngIf]="!authService.isAuthenticated()">
  <li><a routerLink="/signup">Register</a></li>
  <li><a routerLink="/signin">Login</a></li>
</ng-template>
```

257. Add a Logout Button

1. add a funciton in the **auth.service.ts**
this will clean out the token + the firebase stuff

```
 logout() {
    firebase.auth().signOut();
    this.token = null;
  }
```

2. Add a function *onLogout()* conditionally to the **header.component.html**

```
<a          style="cursor: pointer;"
          (click)="onLogout()"
          *ngIf="authService.isAuthenticated()">Logout</a></li>
```

3. Which will call the injected authService function

```
onLogout() {
    this.authService.logout();
  }
```

258.Route Protection with guards

1. Create ant **auth.guard.service.ts,** by implementing the CanActivate service
2. Call the injected **Autservice** to call **firebase**.

```
import { CanActivate,                    //Built in router check method
 ActivatedRouteSnapshot,                //To check the active router, t
 RouterStateSnapshot } from '@angular/router';            //to check the active route states
import { Injectable } from '@angular/core';                //to use the CanActivate servce

import { AuthService } from './auth.service';

@Injectable()
export class AuthGuard implements CanActivate {

  constructor(private authService: AuthService) {}

  canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot) {
    return this.authService.isAuthenticated();
  }
}
```

3. Add to **app-routing.service.ts**

```
import { RecipeEditComponent } from './recipes/recipe-edit/recipe-edit.component';
import { AuthGuard } from './auth/auth-guard.service';

const appRoutes: Routes = [
  { path: '', redirectTo: '/recipes', pathMatch: 'full' },
  { path: 'recipes', component: RecipesComponent, children: [
    { path: '', component: RecipeStartComponent },
    { path: 'new', component: RecipeEditComponent, canActivate: [AuthGuard] },
    { path: ':id', component: RecipeDetailComponent },
    { path: ':id/edit', component: RecipeEditComponent, canActivate: [AuthGuard] },
  ] },
];
@NgModule({
```

```
  imports: [RouterModule.forRoot(appRoutes)],
  exports: [RouterModule]
})
export class AppRoutingModule {
}
```

4. Include the created `AuthGuard` service in the **app.module.ts**

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { HttpModule } from '@angular/http';

import { AppComponent } from './app.component';
import { AuthService } from './auth/auth.service';
import { AuthGuard } from './auth/auth-guard.service';

@NgModule({
  declarations: [
    AppComponent,
  ],
  imports: [
    BrowserModule,
    FormsModule,
    ReactiveFormsModule,
    HttpModule,
    AppRoutingModule
  ],
  providers: [ShoppingListService, RecipeService, DataStorageService, AuthService, AuthGuard],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

260. Improvements Opprotunities

- Check if a token is present at application startup (check the localStorage manually or use the Firebase SDK to do so - just make sure that you somehow wait for the SDK to finish its initialization)
- Redirect the user if he want to access a protected route (right now, nothing happens) - inject the router and call this.router.navigate(...) to do so
- Redirect the user on logout so that he's not able to stay on pages which are reserved for authenticated users - you can simply inject the router and call this.router.navigate(...) in the logout() method