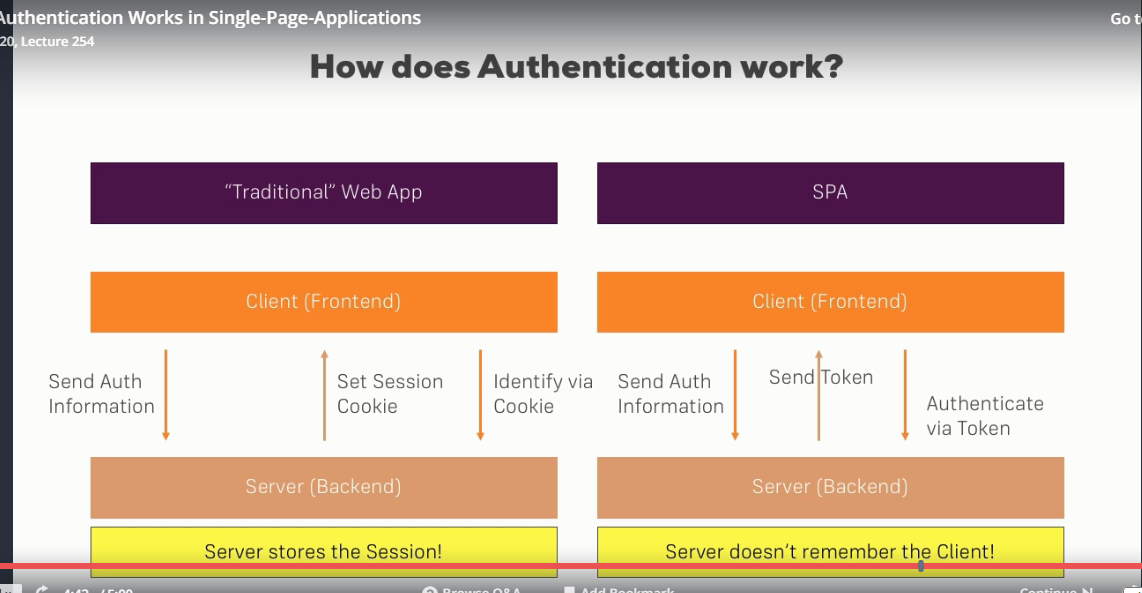
**Section 20 Authentication and Route Protection in Angular Apps**

**Section 20: Lecture 253//Module Introduction**

1. A big chunk of the authentication, as of now we will put the authentication in the firebase application.

**Section 20: Lecture 254//How Authentication Works in Single-Page-Applications**

1. Here we will understand, how does Authentication work. Here we must understand difference between a full stack app and a single page application.
2. In the traditional app we have a very strong connection between the front end and backend i.e. the client and server. In the end the client will receive a finished HTML view from the server which will be displayed at client side.
3. Once the Authorization information is sent from the client to the server – a session gets created on the server. Here, there is a very close connection between the server and the client with the server rendering the views. Here the server is also able to remember the client because it is communicating with the client. Now, once the session is created at the server side the client will get the session cookie.
4. Now, the server can receive the ID of the session through the cookie.
5. For the single page application, we don’t have that strong connection between the front end and backend.
6. We might reach backend time to time but that is not guaranteed – many requests and we must put that in the quotation marks. For example, if a list must be brought and the list is already available at the server side then the client will not at all reach to the server.
7. In the SPA the server will not remember who the client is but still we will send the authentication. If it is not valid then the client will not at all create session at server. Still we must give server something to tell that the client is logged in.
8. In this scenario the server will send us a token that we will send with the request. The server can validate this token.
9. We authenticate via this token on future requests. The token was created on the sever so the sever is easily able to validate the token.
10. 

**Section 20: Lecture 255//More about JWT**

1. Want to learn about the Token which is exchanged?
2. The following page should be helpful: <https://jwt.io/> - specifically, the introduction: <https://jwt.io/introduction/>

**Section 20: Lecture 256//Creating a signup Page and Route**

1. Here we will create 2 new components i.e. sign-in and sign-up page.
2. header.component.html
3. <nav class="navbar navbar-default">
4. <div class="container-fluid">
5. <div class="navbar-header">
6. <a routerLink="/" class="navbar-brand">Recipe Book</a>
7. </div>
8. <div class="collapse navbar-collapse">
9. <ul class="nav navbar-nav">
10. <li routerLinkActive="active"><a routerLink="/recipes">Recipes</a></li>
11. <li routerLinkActive="active"><a routerLink="/shopping-list">Shopping List</a></li>
12. </ul>
13. <ul class="nav navbar-nav navbar-right">
14. <li><a routerLink="/signup">Register</a></li>
15. <li class="dropdown" appDropdown>
16. <a style="cursor: pointer;" class="dropdown-toggle" role="button">Manage <span class="caret"></span></a>
17. <ul class="dropdown-menu">
18. <li><a style="cursor: pointer;" (click)="onSaveData()">Save Data</a></li>
19. <li><a style="cursor: pointer;" (click)="onFetchData()">Fetch Data</a></li>
20. </ul>
21. </li>
22. </ul>
23. </div>
24. </div>
25. </nav>
26. signup.component.html:
27. <div class="row">
28. <div class="col-xs-12 col-sm-10 col-md-8 col-sm-offset-1 col-md-offset-2">
29. <form (ngSubmit)="onSignUp(f)" #f="ngForm">
30. <div class="form-group">
31. <label for="email">Mail</label>
32. <input type="email" id="email" name="email" ngModel class="form-control">
33. </div>
34. <div class="form-group">
35. <label for="password">Password</label>
36. <input type="password" id="password" name="password" ngModel class="form-control">
37. </div>
38. <button class="btn btn-primary" type="submit">Sign Up</button>
39. </form>
40. </div>
41. </div>

4. signup.component.ts

import { Component, OnInit } from '@angular/core';

import { NgForm } from '@angular/forms';

@Component({

selector: 'app-signup',

templateUrl: './signup.component.html',

styleUrls: ['./signup.component.css']

})

export class SignupComponent implements OnInit {

constructor() { }

ngOnInit() {

}

onSignUp(form: NgForm){

const email = form.value.email;

const password = form.value.password;

}

}

5. app-routing.module.ts:

import { NgModule } from '@angular/core';

import { Routes, RouterModule } from '@angular/router';

import { RecipesComponent } from './recipes/recipes.component';

import { ShoppingListComponent } from './shopping-list/shopping-list.component';

import { RecipeStartComponent } from './recipes/recipe-start/recipe-start.component';

import { RecipeDetailComponent } from './recipes/recipe-detail/recipe-detail.component';

import { RecipeEditComponent } from './recipes/recipe-edit/recipe-edit.component';

import { SignupComponent } from './auth/signup/signup.component';

const appRoutes: Routes = [

{ path: '', redirectTo: '/recipes', pathMatch: 'full' },

{ path: 'recipes', component: RecipesComponent, children: [

{ path: '', component: RecipeStartComponent },

{ path: 'new', component: RecipeEditComponent },

{ path: ':id', component: RecipeDetailComponent },

{ path: ':id/edit', component: RecipeEditComponent },

] },

{ path: 'shopping-list', component: ShoppingListComponent },

{ path: 'signup', component: SignupComponent }

];

@NgModule({

imports: [RouterModule.forRoot(appRoutes)],

exports: [RouterModule]

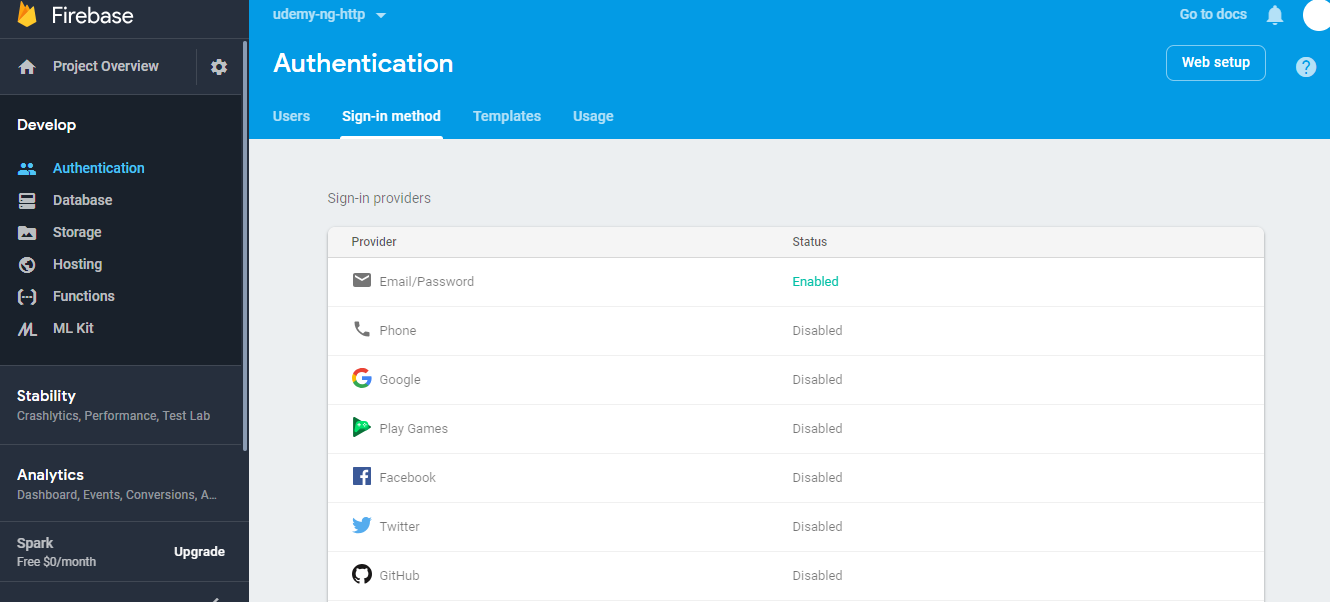
})

export class AppRoutingModule {

}

**Section 20: Lecture 257//Setting up the Firebase SDK**

1. In firebase we will setup the Authentication Area.
2. We have set up the email authentication in the firebase, as shown below:



1. To send the things for the authentication, we will actually use the firebase SDK. The reason is there is no great rest API from firebase side for this. So we will use SDK in our app.
2. To install firebase SDK in our app we will run below line in the terminal:

npm install –save firebase

1. This will also give us some of the typescript libraries that we can use directly in out typescript code. To add it we need some kind of extra service.
2. Now, we will add new service in the Auth file.
3. Here, we will also configure firebase at the app.component.ts in our app.
4. app.component.ts:
5. import { Component, OnInit } from '@angular/core';
6. import \* as firebase from 'firebase';
7. @Component({
8. selector: 'app-root',
9. templateUrl: './app.component.html',
10. styleUrls: ['./app.component.css']
11. })
12. export class AppComponent implements OnInit {
13. loadedFeature = 'recipe';
14. ngOnInit(){
15. firebase.initializeApp({
16. apiKey: "AIzaSyCvBTYzM\_J8EWWM4yFcwWPsaGqx7vtHRsY",
17. authDomain: "udemy-ng-http-81cda.firebaseapp.com"
18. });
19. }
20. onNavigate(feature: string) {
21. this.loadedFeature = feature;
22. }
23. }

9. auth.service.ts:

export class AuthService{

signupUser(email: string, password: string){

}

}

**Section 20: Lecture 258// Signing Users Up**

1. Let’s now allow the user to sign up
2. auth.service.ts:
3. import \* as firebase from 'firebase';
4. export class AuthService{
5. signupUser(email: string, password: string){
6. firebase.auth().createUserWithEmailAndPassword(email, password)
7. .catch(
8. error => console.log(error)
9. )
10. }
11. }
12. app.module.ts:
13. import { BrowserModule } from '@angular/platform-browser';
14. import { NgModule } from '@angular/core';
15. import { FormsModule, ReactiveFormsModule } from '@angular/forms';
16. import { HttpModule } from '@angular/http';
17. import { AppComponent } from './app.component';
18. import { HeaderComponent } from './header/header.component';
19. import { RecipesComponent } from './recipes/recipes.component';
20. import { RecipeListComponent } from './recipes/recipe-list/recipe-list.component';
21. import { RecipeDetailComponent } from './recipes/recipe-detail/recipe-detail.component';
22. import { RecipeItemComponent } from './recipes/recipe-list/recipe-item/recipe-item.component';
23. import { ShoppingListComponent } from './shopping-list/shopping-list.component';
24. import { ShoppingEditComponent } from './shopping-list/shopping-edit/shopping-edit.component';
25. import { DropdownDirective } from './shared/dropdown.directive';
26. import { ShoppingListService } from './shopping-list/shopping-list.service';
27. import { AppRoutingModule } from './app-routing.module';
28. import { RecipeStartComponent } from './recipes/recipe-start/recipe-start.component';
29. import { RecipeEditComponent } from './recipes/recipe-edit/recipe-edit.component';
30. import { RecipeService } from './recipes/recipe.service';
31. import { DataStorageService } from './shared/data-storage.service';
32. import { SignupComponent } from './auth/signup/signup.component';
33. import { SigninComponent } from './auth/signin/signin.component';
34. import { AuthService } from './auth/auth.service';
35. @NgModule({
36. declarations: [
37. AppComponent,
38. HeaderComponent,
39. RecipesComponent,
40. RecipeListComponent,
41. RecipeDetailComponent,
42. RecipeItemComponent,
43. ShoppingListComponent,
44. ShoppingEditComponent,
45. DropdownDirective,
46. RecipeStartComponent,
47. RecipeEditComponent,
48. SignupComponent,
49. SigninComponent
50. ],
51. imports: [
52. BrowserModule,
53. FormsModule,
54. ReactiveFormsModule,
55. HttpModule,
56. AppRoutingModule
57. ],
58. providers: [ShoppingListService, RecipeService, DataStorageService, AuthService],
59. bootstrap: [AppComponent]
60. })
61. export class AppModule { }

4. signup.component.ts:

import { Component, OnInit } from '@angular/core';

import { NgForm } from '@angular/forms';

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

@Component({

selector: 'app-signup',

templateUrl: './signup.component.html',

styleUrls: ['./signup.component.css']

})

export class SignupComponent implements OnInit {

constructor(private authService: AuthService) { }

ngOnInit() {

}

onSignUp(form: NgForm){

const email = form.value.email;

const password = form.value.password;

this.authService.signupUser(email, password);

}

}

**Section 20: Lecture 259 //Signing Users in**

1. When we use firebase with SDK, then the token to sign in will be created by itself by the SDK.
2. Header.component.html:
3. <nav class="navbar navbar-default">
4. <div class="container-fluid">
5. <div class="navbar-header">
6. <a routerLink="/" class="navbar-brand">Recipe Book</a>
7. </div>
8. <div class="collapse navbar-collapse">
9. <ul class="nav navbar-nav">
10. <li routerLinkActive="active"><a routerLink="/recipes">Recipes</a></li>
11. <li routerLinkActive="active"><a routerLink="/shopping-list">Shopping List</a></li>
12. </ul>
13. <ul class="nav navbar-nav navbar-right">
14. <li><a routerLink="/signup">Register</a></li>
15. <li><a routerLink="/signin">Login</a></li>
16. <li class="dropdown" appDropdown>
17. <a style="cursor: pointer;" class="dropdown-toggle" role="button">Manage <span class="caret"></span></a>
18. <ul class="dropdown-menu">
19. <li><a style="cursor: pointer;" (click)="onSaveData()">Save Data</a></li>
20. <li><a style="cursor: pointer;" (click)="onFetchData()">Fetch Data</a></li>
21. </ul>
22. </li>
23. </ul>
24. </div>
25. </div>
26. </nav>

3. signin.component.html:

<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">Sign In</button>

</form>

</div>

</div>

4. signin.componnet.ts  
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);

}

}

5. app-routing.module.ts:

import { NgModule } from '@angular/core';

import { Routes, RouterModule } from '@angular/router';

import { RecipesComponent } from './recipes/recipes.component';

import { ShoppingListComponent } from './shopping-list/shopping-list.component';

import { RecipeStartComponent } from './recipes/recipe-start/recipe-start.component';

import { RecipeDetailComponent } from './recipes/recipe-detail/recipe-detail.component';

import { RecipeEditComponent } from './recipes/recipe-edit/recipe-edit.component';

import { SignupComponent } from './auth/signup/signup.component';

import { SigninComponent } from './auth/signin/signin.component';

const appRoutes: Routes = [

{ path: '', redirectTo: '/recipes', pathMatch: 'full' },

{ path: 'recipes', component: RecipesComponent, children: [

{ path: '', component: RecipeStartComponent },

{ path: 'new', component: RecipeEditComponent },

{ path: ':id', component: RecipeDetailComponent },

{ path: ':id/edit', component: RecipeEditComponent },

] },

{ path: 'shopping-list', component: ShoppingListComponent },

{ path: 'signup', component: SignupComponent },

{ path: 'signin', component: SigninComponent }

];

@NgModule({

imports: [RouterModule.forRoot(appRoutes)],

exports: [RouterModule]

})

export class AppRoutingModule {

}

6. auth.service.ts:

import \* as firebase from 'firebase';

import { concat } from 'rxjs';

export class AuthService{

signupUser(email: string, password: string){

firebase.auth().createUserWithEmailAndPassword(email, password)

.catch(

error => console.log(error)

)

}

signinUser(email: string, password: string){

firebase.auth().signInWithEmailAndPassword(email, password)

.then(

response => console.log(response)

)

.catch(

error => console.log(error)

);

}

}

**Section 20: Lecture 260 //Requiring a Token (on the Backend)**

1. In the last lecture we learnt how to send our credentials and how to get back sign in info - and also get the token which gets stored automatically. Now, we want to use this token.
2. In true sense we will provision app in such a way that saving and fetching data is only possible if send the token to the backend.
3. So, on the backend I want to check if the user us authenticated/ or if the user is not authenticated we will deny access. First step is that we must configure the rules on the backend.
4. Change firebase database rules to:

{

"rules": {

".read": "auth != null",

".write": "auth != null"

}

}

1. Till now we have not sent the token with the request, we have just stored it, so we need to send with the future request which will make sure that only authorized user can login.
2. To do this we would need to change the way in which we send the requests.
3. Right now, we are not sending any token with the request and that must be changed.
4. In next lecture we will work on the auth-service a little bit more so that we can get access to that token.

**Section 20: Lecture 261//Sending the Token**

1. We will add a method to the auth.service.ts to get access to the Token. The below line will give use token Asynchronously

firebase.auth().currentUser.getIdToken();

1. So, here the firebase checks if the token is still valid, if it has expired then it will create a new token for us.
2. data-storage.service.ts:
3. import { Injectable } from '@angular/core';
4. import { Http, Response } from '@angular/http';
5. import 'rxjs/Rx';
6. import { RecipeService } from '../recipes/recipe.service';
7. import { Recipe } from '../recipes/recipe.model';
8. import { AuthService } from '../auth/auth.service';
9. import { tokenName } from '@angular/compiler';
10. @Injectable()
11. export class DataStorageService {
12. constructor(private http: Http,
13. private recipeService: RecipeService,
14. private authService: AuthService
15. ) {}
16. storeRecipes() {
17. const token = this.authService.getToken();
18. return this.http.put('https://udemy-ng-http-81cda.firebaseio.com/recipes.json?auth='+token, this.recipeService.getRecipes());
19. }
20. getRecipes() {
21. const token = this.authService.getToken();
22. this.http.get('https://udemy-ng-http-81cda.firebaseio.com/recipes.json?auth='+token)
23. .map(
24. (response: Response) => {
25. const recipes: Recipe[] = response.json();
26. for (let recipe of recipes) {
27. if (!recipe['ingredients']) {
28. recipe['ingredients'] = [];
29. }
30. }
31. return recipes;
32. }
33. )
34. .subscribe(
35. (recipes: Recipe[]) => {
36. this.recipeService.setRecipes(recipes);
37. }
38. );
39. }
40. }

4. auth.service.ts:

import \* as firebase from 'firebase';

import { concat } from 'rxjs';

export class AuthService{

token: string;

signupUser(email: string, password: string){

firebase.auth().createUserWithEmailAndPassword(email, password)

.catch(

error => console.log(error)

)

}

signinUser(email: string, password: string){

firebase.auth().signInWithEmailAndPassword(email, password)

.then(

response => {

firebase.auth().currentUser.getIdToken()

.then(

(token: string)=> this.token = token

)

}

)

.catch(

error => console.log(error)

);

}

getToken(){

firebase.auth().currentUser.getIdToken()

.then(

(token: string)=> this.token = token

);

return this.token;

}

}

**Section 20: Lecture 262 //Checking and Using Authentication Status**

1. Now, our basic authentication is working, here we will make sure that once we are logged in we will only see the items which are required to be visible to us.
2. In auth.service.ts we will add a method which will tell us whether we are authenticated or not.
3. auth.service.ts:
4. import \* as firebase from 'firebase';
5. import { concat } from 'rxjs';
6. export class AuthService{
7. token: string;
8. signupUser(email: string, password: string){
9. firebase.auth().createUserWithEmailAndPassword(email, password)
10. .catch(
11. error => console.log(error)
12. )
13. }
14. signinUser(email: string, password: string){
15. firebase.auth().signInWithEmailAndPassword(email, password)
16. .then(
17. response => {
18. firebase.auth().currentUser.getIdToken()
19. .then(
20. (token: string)=> this.token = token
21. )
22. }
23. )
24. .catch(
25. error => console.log(error)
26. );
27. }
28. getToken(){
29. firebase.auth().currentUser.getIdToken()
30. .then(
31. (token: string)=> this.token = token
32. );
33. return this.token;
34. }
35. isAuthenticated(){
36. return this.token != null;
38. }
39. }

4. header.component.ts:

import { Component } from '@angular/core';

import { Response } from '@angular/http';

import { DataStorageService } from '../shared/data-storage.service';

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

@Component({

selector: 'app-header',

templateUrl: './header.component.html'

})

export class HeaderComponent {

constructor(private dataStorageService: DataStorageService,

private autheService: AuthService

) {}

onSaveData() {

this.dataStorageService.storeRecipes()

.subscribe(

(response: Response) => {

console.log(response);

}

);

}

onFetchData() {

this.dataStorageService.getRecipes();

}

}

5. header.component.html:

<nav class="navbar navbar-default">

<div class="container-fluid">

<div class="navbar-header">

<a routerLink="/" class="navbar-brand">Recipe Book</a>

</div>

<div class="collapse navbar-collapse">

<ul class="nav navbar-nav">

<li routerLinkActive="active"><a routerLink="/recipes">Recipes</a></li>

<li routerLinkActive="active"><a routerLink="/shopping-list">Shopping List</a></li>

</ul>

<ul class="nav navbar-nav navbar-right">

<ng-template [ngIf]="!authService.isAuthenticated()">

<li><a routerLink="/signup">Register</a></li>

<li><a routerLink="/signin">Login</a></li>

</ng-template>

<li class="dropdown" appDropdown \*ngIf="!authService.isAuthenticated()">

<a style="cursor: pointer;" class="dropdown-toggle" role="button">Manage <span class="caret"></span></a>

<ul class="dropdown-menu">

<li><a style="cursor: pointer;" (click)="onSaveData()">Save Data</a></li>

<li><a style="cursor: pointer;" (click)="onFetchData()">Fetch Data</a></li>

</ul>

</li>

</ul>

</div>

</div>

</nav>

**Section 20: Lecture 263//Adding a Logout Button**

1. Let’s add a logout button
2. header.component.html:
3. <nav class="navbar navbar-default">
4. <div class="container-fluid">
5. <div class="navbar-header">
6. <a routerLink="/" class="navbar-brand">Recipe Book</a>
7. </div>
8. <div class="collapse navbar-collapse">
9. <ul class="nav navbar-nav">
10. <li routerLinkActive="active"><a routerLink="/recipes">Recipes</a></li>
11. <li routerLinkActive="active"><a routerLink="/shopping-list">Shopping List</a></li>
12. </ul>
13. <ul class="nav navbar-nav navbar-right">
14. <ng-template [ngIf]="!authService.isAuthenticated()">
15. <li><a routerLink="/signup">Register</a></li>
16. <li><a routerLink="/signin">Login</a></li>
17. </ng-template>
18. <li><a style="cursor: pointer;" (click)="onLogout" \*ngIf="authservice.isAuthenticated()">Logout</a></li>
19. <li class="dropdown" appDropdown \*ngIf="!authService.isAuthenticated()">
20. <a style="cursor: pointer;" class="dropdown-toggle" role="button">Manage <span class="caret"></span></a>
21. <ul class="dropdown-menu">
22. <li><a style="cursor: pointer;" (click)="onSaveData()">Save Data</a></li>
23. <li><a style="cursor: pointer;" (click)="onFetchData()">Fetch Data</a></li>
24. </ul>
25. </li>
26. </ul>
27. </div>
28. </div>
29. </nav>

3. auth.service.ts:

import \* as firebase from 'firebase';

import { concat } from 'rxjs';

export class AuthService{

token: string;

signupUser(email: string, password: string){

firebase.auth().createUserWithEmailAndPassword(email, password)

.catch(

error => console.log(error)

)

}

signinUser(email: string, password: string){

firebase.auth().signInWithEmailAndPassword(email, password)

.then(

response => {

firebase.auth().currentUser.getIdToken()

.then(

(token: string)=> this.token = token

)

}

)

.catch(

error => console.log(error)

);

}

logout(){

firebase.auth().signOut();

this.token = null;

}

getToken(){

firebase.auth().currentUser.getIdToken()

.then(

(token: string)=> this.token = token

);

return this.token;

}

isAuthenticated(){

return this.token != null;

}

}

4. header.component.ts:

import { Component } from '@angular/core';

import { Response } from '@angular/http';

import { DataStorageService } from '../shared/data-storage.service';

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

@Component({

selector: 'app-header',

templateUrl: './header.component.html'

})

export class HeaderComponent {

constructor(private dataStorageService: DataStorageService,

private autheService: AuthService

) {}

onSaveData() {

this.dataStorageService.storeRecipes()

.subscribe(

(response: Response) => {

console.log(response);

}

);

}

onFetchData() {

this.dataStorageService.getRecipes();

}

onLogout(){

this.autheService.logout();

}

}

**Section 20: Lecture 264//Route Protection and Redirection Example**

1. Let’s fix the routing in our application. If we sign in we want to redirect the user here, but we will only redirect if the sign in was successful
2. In the auth.service.ts we will inject the router.
3. App.module.ts:
4. import { BrowserModule } from '@angular/platform-browser';
5. import { NgModule } from '@angular/core';
6. import { FormsModule, ReactiveFormsModule } from '@angular/forms';
7. import { HttpModule } from '@angular/http';
8. import { AppComponent } from './app.component';
9. import { HeaderComponent } from './header/header.component';
10. import { RecipesComponent } from './recipes/recipes.component';
11. import { RecipeListComponent } from './recipes/recipe-list/recipe-list.component';
12. import { RecipeDetailComponent } from './recipes/recipe-detail/recipe-detail.component';
13. import { RecipeItemComponent } from './recipes/recipe-list/recipe-item/recipe-item.component';
14. import { ShoppingListComponent } from './shopping-list/shopping-list.component';
15. import { ShoppingEditComponent } from './shopping-list/shopping-edit/shopping-edit.component';
16. import { DropdownDirective } from './shared/dropdown.directive';
17. import { ShoppingListService } from './shopping-list/shopping-list.service';
18. import { AppRoutingModule } from './app-routing.module';
19. import { RecipeStartComponent } from './recipes/recipe-start/recipe-start.component';
20. import { RecipeEditComponent } from './recipes/recipe-edit/recipe-edit.component';
21. import { RecipeService } from './recipes/recipe.service';
22. import { DataStorageService } from './shared/data-storage.service';
23. import { SignupComponent } from './auth/signup/signup.component';
24. import { SigninComponent } from './auth/signin/signin.component';
25. import { AuthService } from './auth/auth.service';
26. import { AuthGuard } from './auth/auth-guard.service';
27. @NgModule({
28. declarations: [
29. AppComponent,
30. HeaderComponent,
31. RecipesComponent,
32. RecipeListComponent,
33. RecipeDetailComponent,
34. RecipeItemComponent,
35. ShoppingListComponent,
36. ShoppingEditComponent,
37. DropdownDirective,
38. RecipeStartComponent,
39. RecipeEditComponent,
40. SignupComponent,
41. SigninComponent
42. ],
43. imports: [
44. BrowserModule,
45. FormsModule,
46. ReactiveFormsModule,
47. HttpModule,
48. AppRoutingModule
49. ],
50. providers: [ShoppingListService, RecipeService, DataStorageService, AuthService, AuthGuard],
51. bootstrap: [AppComponent]
52. })
53. export class AppModule { }

4. app-routing.module.ts

import { NgModule } from '@angular/core';

import { Routes, RouterModule } from '@angular/router';

import { RecipesComponent } from './recipes/recipes.component';

import { ShoppingListComponent } from './shopping-list/shopping-list.component';

import { RecipeStartComponent } from './recipes/recipe-start/recipe-start.component';

import { RecipeDetailComponent } from './recipes/recipe-detail/recipe-detail.component';

import { RecipeEditComponent } from './recipes/recipe-edit/recipe-edit.component';

import { SignupComponent } from './auth/signup/signup.component';

import { SigninComponent } from './auth/signin/signin.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] },

] },

{ path: 'shopping-list', component: ShoppingListComponent },

{ path: 'signup', component: SignupComponent },

{ path: 'signin', component: SigninComponent }

];

@NgModule({

imports: [RouterModule.forRoot(appRoutes)],

exports: [RouterModule]

})

export class AppRoutingModule {

}

5. auth.service.ts:

import \* as firebase from 'firebase';

import { concat } from 'rxjs';

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

import { Injectable } from '@angular/core';

@Injectable()

export class AuthService{

token: string;

constructor(private router: Router ){

}

signupUser(email: string, password: string){

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.getIdToken()

.then(

(token: string)=> this.token = token

)

}

)

.catch(

error => console.log(error)

);

}

logout(){

firebase.auth().signOut();

this.token = null;

}

getToken(){

firebase.auth().currentUser.getIdToken()

.then(

(token: string)=> this.token = token

);

return this.token;

}

isAuthenticated(){

return this.token != null;

}

}

6. auth-guard.service.ts:

import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot } from "@angular/router";

import { Injectable } from "@angular/core";

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

@Injectable()

export class AuthGuard implements CanActivate{

constructor(private authService: AuthService){}

canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot){

return this.authService.isAuthenticated();

}

}

**Section 20: Lecture 265//Wrap Up**

**Section 20: Lecture 266//Possible Improvements**

You can of course improve this app even more. Some ideas:

* 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

============================END OF SECTION==========================