Table of Contents

[Current Resources with Changes 3](#_Toc114699360)

[New Resources Added 3](#_Toc114699361)

[Ngx-spinner 3](#_Toc114699362)

[/core/modules/shared.module.ts 3](#_Toc114699363)

[angular.json 3](#_Toc114699364)

[Models 3](#_Toc114699365)

[/core/models/userTokenDto.ts 3](#_Toc114699366)

[Services 4](#_Toc114699367)

[/core/services/local-storage.service.ts 4](#_Toc114699368)

[/core/services/spinner-service.ts 4](#_Toc114699369)

[/core/services/api-url.service.ts 4](#_Toc114699370)

[/core/services/members.service.ts 4](#_Toc114699371)

[Interceptor 7](#_Toc114699372)

[/core/interceptors/spinner.interceptor.ts 7](#_Toc114699373)

[app.module.ts 7](#_Toc114699374)

[Components 8](#_Toc114699375)

[app.component.html 8](#_Toc114699376)

[/site/members/members-list 8](#_Toc114699377)

[members-list.component.ts 8](#_Toc114699378)

[members-list.component.html 8](#_Toc114699379)

[/site/members/member-edit 9](#_Toc114699380)

[app-routing.module.ts 9](#_Toc114699381)

[member-edit.component.ts 9](#_Toc114699382)

[member-edit.component.html 10](#_Toc114699383)

[member-edit.compoent.css 11](#_Toc114699384)

[/site/nav/nav 12](#_Toc114699385)

[nav.component.html 12](#_Toc114699386)

[Guards 12](#_Toc114699387)

[CanDeactiavte 12](#_Toc114699388)

[/core/guards/prevent-unsaved-changes.guard.ts 12](#_Toc114699389)

[app-routing.module.ts 12](#_Toc114699390)

# Current Resources with Changes

1. app-routing.module.ts
2. app.module.ts
3. app.component.html
4. /site/nav/nav.component.html
5. /core/models/userTokenDto.mode.ts
6. /core/services/local-storage.service.ts
7. /core/services/api-url.service.ts
8. /core/services/members.service.ts
9. /core/modules/sahed.module.ts
10. /site/members/members-list

# New Resources Added

1. /site/members/member-edit
2. /core/guards/prevent-unsaved-changes.guard.ts
3. ngx-spinner
4. /core/services/spinner.service.ts
5. /core/interceptors/spinner.interceptor.ts

# Ngx-spinner

Install the ngx-spinner. Since the app is built with Angular 13, the appropriate version of it is 13.1.1. <https://www.npmjs.com/package/ngx-spinner#installation>

* npm add [ngx-spinner@13.1.1](mailto:ngx-spinner@13.1.1)

If you see an error @angular/cdk/schematics missing for spinner then install following first

* npm install @angular/cdk

If it is not added to the app.module.ts then add it. In fact, we will add it to the shared module.

## /core/modules/shared.module.ts

import { NgxSpinnerModule } from 'ngx-spinner';

and then add to both import and export array

NgxSpinnerModule

## angular.json

add the css, you can add multiple css if you want to. Just provide the type accordingly. See [SpinnerService](#_/core/services/spinner-service.ts) below.

              "./node\_modules/ngx-spinner/animations/ball-scale-multiple.css",

              "./node\_modules/ngx-spinner/animations/line-scale-party.css",

# Models

## /core/models/userTokenDto.ts

Addd a guId property to the dto

export class UserTokenDto {

    constructor(public userName: string = "", public guId: string = "", public token: string = "") { }

}

**Important**: logout and log back in

# Services

## /core/services/local-storage.service.ts

Add a helper property to get the guid as well

  getLoggedinUserGuid: string = this.getLoggedInUser?.guId;

## /core/services/spinner-service.ts

* ng g s /core/services/spinner --skip-tests

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

import { NgxSpinnerService } from 'ngx-spinner';

@Injectable({

  providedIn: 'root'

})

export class SpinnerService {

  busyRequestCount: number = 0;

  constructor(private spinnerService: NgxSpinnerService) { }

  //there are multiple type of spinner avaialble, check https://www.npmjs.com/package/ngx-spinner

  show() {

    this.busyRequestCount++;

    this.spinnerService.show(undefined, {

      type: 'line-scale-party',

      bdColor: 'rgba(255,255,255,0)',

      color: '#333333'

    });

  }

  hide() {

    this.busyRequestCount--;

    if (this.busyRequestCount <= 0) {

      this.busyRequestCount = 0;

      this.spinnerService.hide();

    }

  }

}

## /core/services/api-url.service.ts

Add a new url for updating the member

//users controller urls

  private usersBaseUrl = `${this.apiBaseUrl}users/`;

  userUpdate = `${this.usersBaseUrl}`;

  usersAll = `${this.usersBaseUrl}`;

  userByGuIdReplace = "[guid]";

  userByGuId = `${this.usersBaseUrl}${this.userByGuIdReplace}/guid`; //replace [guid] with the id of the user

  userByIdReplace = "[id]";

  userById = `${this.usersBaseUrl}${this.userByIdReplace}/id`; //replace [id] with the id of the user

  userByNameReplace = "[name]";

  userByName = `${this.usersBaseUrl}${this.userByNameReplace}/name`; //replace [name] with the userName of the user

## /core/services/members.service.ts

Full member service is here since this has now state management

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

import { ApiUrlService } from './api-url.service';

import { HttpClientService } from './http-client.service';

import { LocalStorageService } from './local-storage.service';

import { environment } from 'src/environments/environment';

import { userDto } from '../models/userDto.model';

import { HttpHeaders } from '@angular/common/http';

import { map, Observable, of } from 'rxjs';

@Injectable({

  providedIn: 'root'

})

export class MembersService {

  //to pass the auth token to the api, later will use interceptor

  private httpOptions;

  //state mangement. We dont want to pull the user info every time

  members: userDto[] = [];

  constructor(private apiUrlService: ApiUrlService, private httpClientService: HttpClientService, private localStorageService: LocalStorageService) {

    //to pass the auth token to the api, later will user interceptor

    //not using this any more, using the interceptor

    this.httpOptions = {

      headers: new HttpHeaders({

        Authorization: 'Bearer ' + localStorageService.getLoggedinToken

      })

    };

  }

  /\*\*

   \* A GET method

   \* @returns returns Observable userDto[]

   \*/

  getMembers(): Observable<userDto[]> {

    //state mangement. when we habe members then return that

    if (this.members.length > 0) {

      //return observable

      return of(this.members);

    }

    const url = this.apiUrlService.usersAll;

    if (environment.displayConsoleLog) console.log(`Users allUrl: ${url}`);

    //users end point is protected by authentication so need to send the token as well

    //check jwt interceptor for details

    /\*

    return this.httpClientService

      .get<userDto[]>(url, this.httpOptions);

    \*/

    return this.httpClientService.get<userDto[]>(url).pipe(

        map(members => {

          //set the members for state management and return as well

          this.members = members;

          return members;

        })

      );

    }

  /\*\*

   \* A GET method

   \* @param id id to fetch

   \* @returns returns Observable userDto

   \*/

  getMemberByGuId(guid: string): Observable<userDto>{

    //state management. When the member is in members array then return from there other wise get from api

    if (this.members.length > 0) {

      const member = this.members.find(x => x.guId === guid);

      if (member !== undefined) {

        return of(member);

      }

    }

    let url = this.apiUrlService.userByGuId;

    //replace [guid] with the guid in the url

    url = url.replace(this.apiUrlService.userByGuIdReplace, guid.toString());

    if (environment.displayConsoleLog) console.log(`Users by id url: ${url}`)

    //users end point is protected by authentication so need to send the token as well

    //check jwt interceptor for details

    /\*

    return this.httpClientService

      .get<userDto>(url, this.httpOptions);

      \*/

    return this.httpClientService.get<userDto>(url);

  }

  /\*\*

   \* A GET method

   \* @param id id to fetch

   \* @returns returns Observable userDto

   \*/

  getMemberById(id: number): Observable<userDto>{

    //state management. When the member is in members array then return from there other wise get from api

    if (this.members.length > 0) {

      const member = this.members.find(x => x.id === id);

      if (member !== undefined) {

        return of(member);

      }

    }

    let url = this.apiUrlService.userById;

    //replace [id] with the id in the url

    url = url.replace(this.apiUrlService.userByIdReplace, id.toString());

    if (environment.displayConsoleLog) console.log(`Users by id url: ${url}`);

    //users end point is protected by authentication so need to send the token as well

    //check jwt interceptor for details

    /\*

    return this.httpClientService

      .get<userDto>(url, this.httpOptions);

      \*/

      return this.httpClientService.get<userDto>(url);

  }

  /\*\*

   \* A GET method

   \* @param userName userName to fetch

   \* @returns returns Observable userDto

   \*/

  getMemberByUserName(userName: string): Observable<userDto>{

    //state management. When the member is in members array then return from there other wise get from api

    if (this.members.length > 0) {

      const member = this.members.find(x => x.userName === userName);

      if (member !== undefined) {

        return of(member);

      }

    }

    let url = this.apiUrlService.userByName;

    //replace [name] with the userName in the url

    url = url.replace(this.apiUrlService.userByNameReplace, userName);

    if (environment.displayConsoleLog) console.log(`Users by userName url: ${url}`);

    //users end point is protected by authentication so need to send the token as well

    //check jwt interceptor for details

    /\*return this.httpClientService

      .get<userDto>(url, this.httpOptions);

    \*/

      return this.httpClientService.get<userDto>(url);

  }

  /\*\*

   \* A PUT method

   \* @param member member to update.

   \* Only updating introduction, lookingFor, interests, city, country so still can send the full member model

   \* @returns returns Observable noContent 204

   \*/

  updateMember(member: userDto) {

    let url = this.apiUrlService.userUpdate;

    if (environment.displayConsoleLog) console.log(`User update url: ${url}`);

    return this.httpClientService.put(url, member).pipe(

      map(() => {

        const index = this.members.indexOf(member);

        this.members[index] = member;

      })

    );

  }

}

# Interceptor

## /core/interceptors/spinner.interceptor.ts

* ng g interceptor /core/interceptors/spinner --skip-tests

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

import {

  HttpRequest,

  HttpHandler,

  HttpEvent,

  HttpInterceptor

} from '@angular/common/http';

import { delay, finalize, Observable } from 'rxjs';

/\*

calls the spinner service

root component has the spinner html

\*/

import { SpinnerService } from '../services/spinner.service';

@Injectable()

export class SpinnerInterceptor implements HttpInterceptor {

  constructor(private spinnerService: SpinnerService) {}

  intercept(request: HttpRequest<unknown>, next: HttpHandler): Observable<HttpEvent<unknown>> {

    //call the spinner when the request is made

    this.spinnerService.show();

    //when the request is complete then hide it so tap into next

    return next.handle(request).pipe(

      //fake delay for showing the spinner. real work application no delay is needed

      delay(1000), //one second

      finalize(() => {

        this.spinnerService.hide();

      })

    );

  }

}

Check the [app.compoent.html](#_app.component.html) for th spinner html

### app.module.ts

Add to the app.module.ts providers array

{ provide: HTTP\_INTERCEPTORS, useClass: SpinnerInterceptor, multi: true },

# Components

## app.component.html

Display the spinner

<ngx-spinner>

    <h3>Loading...</h3>

</ngx-spinner>

<app-nav [title]="title"></app-nav>

<div class="container main-container" >

    <!--add the router outlet-->

    <!--<app-site-home [title]="title"></app-site-home>-->

    <router-outlet></router-outlet>

</div>

<footer><small><span class="title">{{ title }}</span> WebApiURL: <span class="api-url">{{webApiUrl}}</span></small></footer>

## /site/members/members-list

* Setting up members$ as observable
* And using of async pipe in the html

### members-list.component.ts

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

import { Observable } from 'rxjs';

import { userDto } from '../../../core/models/userDto.model';

import { MembersService } from '../../../core/services/members.service';

@Component({

  selector: 'app-member-list',

  templateUrl: './member-list.component.html',

  styleUrls: ['./member-list.component.css']

})

export class MemberListComponent implements OnInit {

  //showing async operator in html

  //members: userDto[] = [];

  members$!: Observable<userDto[]>;

  constructor(private membersService: MembersService) { }

  ngOnInit(): void {

    //this.loadMembers();

    this.members$ = this.membersService.getMembers();

  }

  /\*

  loadMembers() {

    this.membersService.getMembers().subscribe({

      next: (members : userDto[]) => {

        this.members = members;

        console.log(members);

      }, error: e => {

        //error will get displayed by the error interceptor

      }, complete: () => {}

    });

  }

  \*/

}

### members-list.component.html

<div class="row">

    <!--

    members$ is now observable so using async pipe

    <div \*ngFor="let member of members" class="col-sm-4 col-md-2 col-12">

        <app-member-card [member]="member"></app-member-card>

    </div>

    -->

    <div \*ngFor="let member of members$ | async" class="col-sm-4 col-md-2 col-12">

        <app-member-card [member]="member"></app-member-card>

    </div>

</div>

## /site/members/member-edit

Create a new member edit component by using command

* ng g c /site/members/member-edit --module=/core/modules/shared.module.ts --skip-tests

It will add the declarations to the share module.

### app-routing.module.ts

Add the route for the members edit page

children: [

      { path: 'members/list', component: MemberListComponent },

      { path: 'members/detail/:guid/:name', component: MemberDetailComponent },

      { path: 'members/edit', component: MemberEditComponent },

      { path: 'lists', component: ListsComponent },

      { path: 'messages', component: MessagesComponent },

    ]

### member-edit.component.ts

Important: must logout and then log back in

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

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

import { ToastrService } from 'ngx-toastr';

import { Subscription, take } from 'rxjs';

import { userDto } from '../../../core/models/userDto.model';

import { UserTokenDto } from '../../../core/models/userTokenDto.model';

import { AccountService } from '../../../core/services/account.service';

import { MembersService } from '../../../core/services/members.service';

@Component({

  selector: 'app-member-edit',

  templateUrl: './member-edit.component.html',

  styleUrls: ['./member-edit.component.css']

})

export class MemberEditComponent implements OnInit {

  @ViewChild('editForm') editForm!: NgForm;

  member: userDto = <userDto>{};

  user: UserTokenDto = <UserTokenDto>{};

  memberUpdateSubscription!:Subscription;

  //inside the angular user wil get propted when the user will move to another route without saving the changes.

  //when the browser is closed or gone to another website then do the following

  @HostListener('window:beforeunload', ['$event']) unloadNotification($event: any) {

    if (this.editForm.dirty)

      $event.returnValue = true;

  }

  memberSubscription!: Subscription;

  constructor(private accountService: AccountService, private memberService: MembersService, private toastrService: ToastrService) {

    //subscribe to the logged in user. due to take(1) dont need to unsubscribe

    this.accountService.currentUser$.pipe(take(1)).subscribe(user => this.user = user);

  }

  ngOnInit(): void {

    this.loadMember();

  }

  loadMember() {

    this.memberSubscription = this.memberService.getMemberByGuId(this.user.guId).subscribe({

      next: (member: userDto) => {

        this.member = member;

      },

      error: e => {},

      complete: () => { }

    });

  }

  onUpdateMember() {

    this.memberUpdateSubscription = this.memberService.updateMember(this.member).subscribe({

      next: () => {

        //it is not reurning any thing

        this.toastrService.success('Profile updated successfully');

        this.editForm.reset(this.member);

      },

      error: e => { },

      complete: () => {}

    });

  }

}

### member-edit.component.html

<div class="row" \*ngIf="member">

    <div class="col-4">

        <h1>{{ member.displayName }}'s profile</h1>

    </div>

    <div class="col-8">

        <!--only display this when the form is direty-->

        <div class="alert alert-info" \*ngIf="editForm.dirty">

            <strong>Information: </strong> You have made changes. Any unsaved changes will be lost

        </div>

    </div>

    <!--left side quick info start-->

    <div class="col-4">

        <div class="card">

            <div class="card-header">

                <div class="d-grid gap-2">

                    <button form="editForm" type="sumit" [disabled]="!editForm.dirty" class="btn btn-success">Save Changes</button>

                </div>

              </div>

            <img src="{{ member.photoUrl || './assets/user.png' }}" alt="{{ member.displayName}}" class="card-img-top img-thumbnail">

            <div class="card-body">

                <div>

                    <strong>Location:</strong>

                    <p>{{ member.city }}, {{ member.country }}</p>

                </div>

                <div>

                    <strong>Age:</strong>

                    <p>{{ member.age }}</p>

                </div>

                <div>

                    <strong>Last Active:</strong>

                    <p>{{ member.lastActive }}</p>

                </div>

                <div>

                    <strong>Member Since:</strong>

                    <p>{{ member.createdOn }}</p>

                </div>

            </div>

            <div class="card-footer">

                <div class="d-grid gap-2">

                    <button form="editForm" type="sumit" [disabled]="!editForm.dirty" class="btn btn-success">Save Changes</button>

                </div>

            </div>

        </div>

    </div>

    <!--left side quick info end-->

    <!--right side detail starts-->

    <div class="col-8">

        <tabset class="member-tabset">

            <tab heading="About {{ member.displayName }}">

                <form #editForm="ngForm" id="editForm" (ngSubmit)="onUpdateMember()">

                    <h4>Description</h4>

                    <textarea [(ngModel)]="member.introduction" name="introduction" id="introduction" rows="6" class="form-control"></textarea>

                    <h4 class="mt-2">Looking for</h4>

                    <textarea [(ngModel)]="member.lookingFor" name="lookingFor" id="lookingFor" rows="6" class="form-control"></textarea>

                    <h4 class="mt-2">Interests</h4>

                    <textarea [(ngModel)]="member.interests" name="interests" id="interests" rows="6" class="form-control"></textarea>

                    <h4 class="mt-2">Location Details: </h4>

                    <div class="mt-2 row">

                        <label for="city" class="col-2 col-form-label">City: </label>

                        <div class="col-sm-10">

                        <input [(ngModel)]="member.city" type="text" name="city" id="city" class="form-control mx-2">

                        </div>

                    </div>

                    <div class="mt-2 row">

                        <label for="country" class="col-2 col-form-label">country: </label>

                        <div class="col-sm-10">

                        <input [(ngModel)]="member.country" type="text" name="country" id="country" class="form-control mx-2">

                        </div>

                    </div>

                </form>

            </tab>

            <tab heading="Edit Photos">

                <h4>Interests</h4>

                <p>Photo edit will go here</p>

            </tab>

        </tabset>

    </div>

    <!--right side detail ends-->

</div>

### member-edit.compoent.css

.img-thumbnail{

    margin: 25px;

    width: 85%;

    height: 85%;

}

.card-body{

    padding: 0 25px;

}

.card-footer{

    padding: 10px 15px;

    background-color: #fff;

    border-top: none;

}

## /site/nav/nav

### nav.component.html

In the drop down we have edit profile so will add the link here

li role="menuitem"><a class="dropdown-item" routerLink="/members/edit" (click)="onNavBarItemClickCloseNavBar()">Edit Profile</a></li>

# Guards

## CanDeactiavte

### /core/guards/prevent-unsaved-changes.guard.ts

Use the following command

* ng g guard /core/guards/prevent-unsaved-changes --skip-tests

and then select “CanDeactiavte”

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

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

import { MemberEditComponent } from '../../site/members/member-edit/member-edit.component';

@Injectable({

  providedIn: 'root'

})

export class PreventUnsavedChangesGuard implements CanDeactivate<unknown> {

  canDeactivate(component: MemberEditComponent): boolean {

    if (component.editForm.dirty)

    {

      return confirm('Are you sure you want to continue? Any unsaved changes will be lost.');

    }

    return true;

  }

}

### app-routing.module.ts

apply the CanDeactiavte guard to members edit route

{ path: 'members/edit', component: MemberEditComponent, canDeactivate: [PreventUnsavedChangesGuard] },