Table of Contents

[Current Resources with Changes 2](#_Toc115134949)

[New Resources Added 2](#_Toc115134950)

[ng2-file-upload 2](#_Toc115134951)

[/core/Modules.Shared.module.ts 2](#_Toc115134952)

[Models 2](#_Toc115134953)

[/core/models/userTokenDto.model.ts 2](#_Toc115134954)

[/core/constants/app-constants.ts 2](#_Toc115134955)

[/core/inetrceptors/jwt.interceptor.ts 2](#_Toc115134956)

[Services 3](#_Toc115134957)

[/core/services/account.service.ts 3](#_Toc115134958)

[/core/services/api-url.service.ts 4](#_Toc115134959)

[/core/services/member.service.ts 4](#_Toc115134960)

[Use Bearer from Constants 4](#_Toc115134961)

[Make Photo Active and Delete Methods 4](#_Toc115134962)

[styles.css 5](#_Toc115134963)

[Components 5](#_Toc115134964)

[/site/nav/nav-component 5](#_Toc115134965)

[nav.component.css 5](#_Toc115134966)

[nav.component.html 5](#_Toc115134967)

[/site/members/photo-editor 5](#_Toc115134968)

[/core/modules/shared.module.ts 6](#_Toc115134969)

[photo-editor.component.css 6](#_Toc115134970)

[photo-editor.component.ts 6](#_Toc115134971)

[photo-editor.component.html 8](#_Toc115134972)

[/site/members/member-edit 10](#_Toc115134973)

[member-edit.component.html 10](#_Toc115134974)

# Current Resources with Changes

1. Ng2-file-upload
2. /core/modules/shared.module.ts
3. /core/inetrceptors/jwt.interceptor.ts
4. /site/members/member-edit
5. /core/services/api-url.services
6. /core/services/members.service
7. /core/models/userTokenDto.model
8. /core/services/account.service
9. /site/nav/nav-component
10. styles.css

# New Resources Added

1. /core/constants/app-constants
2. /site/members/photo-editor

# ng2-file-upload

install the package ng2-file-upload

* npm install ng2-file-upload

## /core/Modules.Shared.module.ts

Go to shared module add to both imports and exports array

import { FileUploadModule } from 'ng2-file-upload';

and then add following to both imports and exports arrays

FileUploadModule

# Models

## /core/models/userTokenDto.model.ts

Add the mainPhotoUrl that is being returned by the api

export class UserTokenDto {

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

}

# /core/constants/app-constants.ts

export class AppConstants {

    public static readonly Bearer = 'Bearer ';

}

# /core/inetrceptors/jwt.interceptor.ts

Convert the bearer key word to be used from the constant

import { AppConstants } from '../constants/app-constants';

Use

if (currentUser!) {

      request = request.clone({

        setHeaders: {

          Authorization: `${AppConstants.Bearer}${currentUser.token}`

        }

      });

    }

# Services

## /core/services/account.service.ts

Clean up the service

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

import { map, ReplaySubject } from 'rxjs';

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

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

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

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

import { LoginDto } from '../models/loginDto.model';

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

import { SiteRegisterDto } from '../models/siteRegisterDto.model';

@Injectable({

  providedIn: 'root'

})

export class AccountService {

  //create an observable to store the user in

  //replaySubject is a special type of observable.

  //replaySubject is kind of buffer object, it will store the values and any time a subscriber subscribes to it, it will emit the last value inside it

  private currentUserSource = new ReplaySubject<UserTokenDto>(1);

  currentUser$ = this.currentUserSource.asObservable();

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

  //fire current user

  private fireCurrentUser(user: UserTokenDto) {

    this.currentUserSource.next(user);

  }

  //login the user. receives LoginDto and returns UserTokenDto

  login(loginDto: LoginDto) {

    var url = this.apiUrlService.accountLogin;

    if(environment.displayConsoleLog) console.log(`AccountService LoginUrl: ${url}`);

    //persist the user in the local storage

    return this.httpClientService

      .post<UserTokenDto>(url, loginDto)

      .pipe(

        map((respone: UserTokenDto) => {

          const user = respone;

          if(environment.displayConsoleLog) console.log(user);

          if (user) this.setAndFireCurrentUser(user);

          return user;

        })

      );

  }

  setAndFireCurrentUser(user: UserTokenDto) {

    //store the user in local storage

    this.localStorageService.setItem(this.localStorageService.\_keyUser, user);

    this.fireCurrentUser(user);

  }

  getAndFireCurrentUser() {

    const user: UserTokenDto = this.localStorageService.getItem(this.localStorageService.\_keyUser);

    this.fireCurrentUser(user);

  }

  logout() {

    //remove the user from local storage

    this.localStorageService.removeItem(this.localStorageService.\_keyUser);

    this.fireCurrentUser(null!);

  }

  register(registerDto: SiteRegisterDto) {

    var url = this.apiUrlService.accountRegisterUser;

    if(environment.displayConsoleLog)

      console.log(`AccountService RegisterUrl: ${url}`);

    return this.httpClientService

      .post<UserTokenDto>(url, registerDto)

      .pipe(

        map((respone: UserTokenDto) => {

          const user = respone;

          if (user) this.setAndFireCurrentUser(user);

          return user;

        })

      );

  }

}

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

Update users controllers urls

    //users controller urls

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

  userByIdReplace = "[id]";

  userByNameReplace = "[name]";

  userByGuIdReplace = "[guid]";

  userPhotoIdReplace = "[photoId]";

  userUpdate = `${this.usersBaseUrl}`;

  usersAll = `${this.usersBaseUrl}`;

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

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

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

  userAddPhoto = `${this.usersBaseUrl}add/photo`;

  userSetPhotoMain = `${this.usersBaseUrl}set/photo/${this.userPhotoIdReplace}/main`;//replace [photoId] with the photoId

  userDeletePhoto =  `${this.usersBaseUrl}delete/${this.userPhotoIdReplace}/photo`;//replace [photoId] with the photoId

## /core/services/member.service.ts

### Use Bearer from Constants

import { AppConstants } from '../constants/app-constants';

and use

    this.httpOptions = {

      headers: new HttpHeaders({

        Authorization: AppConstants.Bearer + localStorageService.getLoggedinToken

      })

    };

### Make Photo Active and Delete Methods

  /\*\*

 \* A PUT method to make the photo main

 \* @param photoId photo id to make active

 \*

 \* @returns returns nocontent

 \*/

  setMainPhoto(photoId: number) {

    let url = this.apiUrlService.userSetPhotoMain;

    url = url.replace(this.apiUrlService.userPhotoIdReplace, photoId.toString());

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

    //pass an empty object since it is a put request

    return this.httpClientService.put(url, {});

  }

  /\*\*

 \* A DELETE method to delete the photo

 \* @param photoId photo id to delete

 \*

 \* @returns returns Ok

 \*/

  deletePhoto(photoId: number) {

    let url = this.apiUrlService.userDeletePhoto;

    url = url.replace(this.apiUrlService.userPhotoIdReplace, photoId.toString());

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

    return this.httpClientService.delete(url);

  }

# styles.css

increase the margin from 65px to 75px.

.main-container {

    margin-top: 75px;

    margin-bottom: 65px; /\*with footer\*/

}

# Components

## /site/nav/nav-component

Display the users image in the nac

### nav.component.css

img {

    border-radius: 50%;

    max-height: 40px;

    border: 2px solid #FF8C00;

    display: inline;

}

### nav.component.html

Add the image inside the button

<button type="button" class="btn btn-info">

   <img src="{{ userInfo.mainPhotoUrl }}" alt="{{ userInfo.userName | titlecase }}" class="me-2">

   Welcome {{userInfo.userName | titlecase}}</button>

A picture containing text, screenshot, display

Description automatically generated

## /site/members/photo-editor

Create a component and add it to declarations array of the shared module

* ng g c /site/members/photo-editor --module=/core/modules/shared --skip-tests

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

Photo editor component will get added to the shared module

### photo-editor.component.css

hr{

    border: 1px dashed black;

    background-color: #fff;

}

img.img-thumbnail{

    height: 100px;

    min-width: 100px !important;

    margin-bottom: 2px;

}

.nv-file-over{

    border: dotted 3px #FF4500;

}

/\*Remove "no files selected text next to the buttons" \*/

input[type=file]{

    color: transparent;

}

### photo-editor.component.ts

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

import { Subscription, take } from 'rxjs';

import { FileUploader } from 'ng2-file-upload';

import { ToastrService } from 'ngx-toastr';

import { AppConstants } from '../../../core/constants/app-constants';

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

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

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

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

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

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

import { PhotoDto } from '../../../core/models/photoDto.model';

@Component({

  selector: 'app-photo-editor',

  templateUrl: './photo-editor.component.html',

  styleUrls: ['./photo-editor.component.css']

})

export class PhotoEditorComponent implements OnInit, OnDestroy {

  @Input() member!: userDto ;

  //bg2-file-uplaod

  uploader!: FileUploader;

  hasBaseDropZoneOver = false;

  currentUser!: UserTokenDto;

  setPhotoActiveSubscription!: Subscription;

  deletePhotoSubscriptoion!: Subscription;

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

    //no need to unsubscribe since we are taking 1 only

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

  }

  ngOnInit(): void {

    this.initilizeUploader();

  }

  ngOnDestroy(): void {

    //unsubscribe

    if (this.setPhotoActiveSubscription) this.setPhotoActiveSubscription.unsubscribe();

    if (this.deletePhotoSubscriptoion) this.deletePhotoSubscriptoion.unsubscribe();

  }

  formatBytes(bytes:number, decimals?: number) {

    if (bytes == 0) return '0 Bytes';

    const k = 1024,

      dm = decimals || 2,

      sizes = ['Bytes', 'KB', 'MB', 'GB', 'TB', 'PB', 'EB', 'ZB', 'YB'],

      i = Math.floor(Math.log(bytes) / Math.log(k));

    return parseFloat((bytes / Math.pow(k, i)).toFixed(dm)) + ' ' + sizes[i];

  }

  //to set the drop zone, name must be exact

  fileOverBase(e: any) {

    this.hasBaseDropZoneOver = e;

  }

  initilizeUploader() {

    if (environment.displayConsoleLog) {

      console.log("\*\*\*\*\*\*\*PhotoEditor Start\*\*\*\*\*\*\*\*\*\*\*")

      console.log(`AddPhots Url: ${this.apiUrlService.userAddPhoto}`);

      console.log(`authtoken: ${AppConstants.Bearer}${this.currentUser.token}`);

      console.log("\*\*\*\*\*\*\*PhotoEditor End\*\*\*\*\*\*\*\*\*\*\*")

    }

    const maxFileSize = 10 \* 1024 \* 1024; //10 mb image only

    //set up. provide url, authToken

    this.uploader = new FileUploader({

      url: this.apiUrlService.userAddPhoto,

      authToken: `${AppConstants.Bearer}${this.currentUser.token}`,

      isHTML5: true,

      allowedFileType: ['image'], //jpeg, png, gif etc are allowed

      removeAfterUpload: true, //remove the file from drop zone

      autoUpload: false, //user will need to click the button

      maxFileSize: maxFileSize,

    });

    //when the file is not added display error message

    this.uploader.onWhenAddingFileFailed = (item, filter) => {

      switch (filter.name) {

        case 'fileSize': {

          const fileSizeError = `${item.name} is ${this.formatBytes(item.size)}. Max allowed is ${this.formatBytes(maxFileSize)}`;

          this.toastrService.error(fileSizeError, `Error: ${filter.name}`);

          break;

        }

        default: {

          this.toastrService.error(`Error trying to upload file ${item.name}`, `Error: ${filter.name}`);

          break;

        }

      }

    }

    //we are using bearer token. Otherwise will need to adjust the cors configuration and allow credentials to go with file

    this.uploader.onAfterAddingFile = (file) => {

      file.withCredentials = false;

    }

    //push the photo to the members.photos array after successful upload

    //it is making the users/add/photo call which is returning the added photo. Photo is in the response

    this.uploader.onSuccessItem = (item, response, status, headers) => {

      if (response) {

        const photo = <PhotoDto>JSON.parse(response);

        this.member.photos.push(photo);

      }

    }

  }

  onSetMainPhoto(photo: PhotoDto) {

    this.setPhotoActiveSubscription = this.memberService.setMainPhoto(photo.id).subscribe({

      next: () => {

        //make the new photo current to show for the nav button

        this.currentUser.mainPhotoUrl = photo.url;

        //update the local storage and fire it so that the left nav can update the image

        this.accountService.setAndFireCurrentUser(this.currentUser);

        //update member

        this.member.photoUrl = photo.url;

        //go through member photos and mark the current one as not main and mark the new as main

        this.member.photos.forEach(p => {

          //make the current photo as true, the rest as false

            p.isMain = (p.id === photo.id);

        });

      },

      error: e => {},

      complete: () => {}

    });

  }

  onDeletePhoto(photo: PhotoDto) {

    this.deletePhotoSubscriptoion = this.memberService.deletePhoto(photo.id).subscribe({

      next: () => {

        //remove the photo from the members

        this.member.photos = this.member.photos.filter(x => x.id !== photo.id);

      },

      error: e => { },

      complete: () => {}

    });

  }

}

### photo-editor.component.html

<div class="row">

    <!--display current user images-->

    <div class="col-2" \*ngFor="let photo of member.photos">

        <img src="{{ photo.url }}" alt="{{member.displayName}} photo URL {{ photo.url }}" class="img-thumbnail p-1">

        <div class="text-center mt-1">

            <ng-container \*ngIf="photo.isMain">

                <button class="btn btn-sm btn-success active" [disabled]="photo.isMain" title="Main Image"><i class="fa fa-star"></i></button>

            </ng-container>

            <ng-container \*ngIf="!photo.isMain">

                <button class="btn btn-sm btn-outline-success" title="Make Main" (click)="onSetMainPhoto(photo)"><i class="fa fa-check"></i></button>

            </ng-container>

            &nbsp;

            <button class="btn btn-sm btn-danger" title="Delete" [disabled]="photo.isMain" (click)="onDeletePhoto(photo)"><i class="fa fa-trash"></i></button>

        </div>

    </div>

</div>

<hr class="mt-3 mb-3" />

<!--copied the html from https://valor-software.com/ng2-file-upload/ and then fixed it per our need-->

<div class="row">

    <div class="col-md-3">

        <h3>Add Photos</h3>

        <div ng2FileDrop

             [ngClass]="{'nv-file-over': hasBaseDropZoneOver}"

             (fileOver)="fileOverBase($event)"

             [uploader]="uploader"

             class="card bg-faded p-3 text-center mb-3 my-drop-zone">

             <i class="fa fa-upload fa-3x"></i>

            Drop photos here

        </div>

        Multiple

        <input type="file" ng2FileSelect [uploader]="uploader" multiple /><br/>

        Single

        <input type="file" ng2FileSelect [uploader]="uploader" />

    </div>

    <!--only show uploader when the file has been selected-->

    <div class="col-md-9" style="margin-bottom: 40px" \*ngIf="uploader?.queue?.length">

        <h3>Upload queue</h3>

        <p>Queue length: {{ uploader?.queue?.length }}</p>

        <table class="table">

            <thead>

            <tr>

                <th width="50%">Name</th>

                <th>Size</th>

                <th>Progress</th>

                <th>Status</th>

                <th>Actions</th>

            </tr>

            </thead>

            <tbody>

            <tr \*ngFor="let item of uploader.queue">

                <td><strong>{{ item?.file?.name }}</strong></td>

                <td \*ngIf="uploader.options.isHTML5" nowrap>{{ item?.file?.size/1024/1024 | number:'.2' }} MB</td>

                <td \*ngIf="uploader.options.isHTML5">

                    <div class="progress" style="margin-bottom: 0;">

                        <div class="progress-bar" role="progressbar" [ngStyle]="{ 'width': item.progress + '%' }"></div>

                    </div>

                </td>

                <td class="text-center">

                    <span \*ngIf="item.isSuccess"><i class="glyphicon glyphicon-ok"></i></span>

                    <span \*ngIf="item.isCancel"><i class="glyphicon glyphicon-ban-circle"></i></span>

                    <span \*ngIf="item.isError"><i class="glyphicon glyphicon-remove"></i></span>

                </td>

                <td nowrap>

                    <!--

                    <button type="button" class="btn btn-success btn-xs"

                            (click)="item.upload()" [disabled]="item.isReady || item.isUploading || item.isSuccess">

                        <span class="glyphicon glyphicon-upload"></span> Upload

                    </button>

                    <button type="button" class="btn btn-warning btn-xs"

                            (click)="item.cancel()" [disabled]="!item.isUploading">

                        <span class="glyphicon glyphicon-ban-circle"></span> Cancel

                    </button>

                    <button type="button" class="btn btn-danger btn-xs"

                            (click)="item.remove()">

                        <span class="glyphicon glyphicon-trash"></span> Remove

                    </button>

                    -->

                </td>

            </tr>

            </tbody>

        </table>

        <div>

            <div>

                Queue progress:

                <div class="progress" style="">

                    <div class="progress-bar" role="progressbar" [ngStyle]="{ 'width': uploader.progress + '%' }"></div>

                </div>

            </div>

            <button type="button" class="btn btn-success btn-s"

                    (click)="uploader.uploadAll()" [disabled]="!uploader.getNotUploadedItems().length">

                <span class="fa fa-upload"></span> Upload all

            </button>

            <button type="button" class="btn btn-warning btn-s"

                    (click)="uploader.cancelAll()" [disabled]="!uploader.isUploading">

                <span class="fa fa-ban"></span> Cancel all

            </button>

            <button type="button" class="btn btn-danger btn-s"

                    (click)="uploader.clearQueue()" [disabled]="!uploader.queue.length">

                <span class="fa fa-trash"></span> Remove all

            </button>

        </div>

    </div>

</div>

## /site/members/member-edit

### member-edit.component.html

Add the app-photo-editor to the edit photos tab

            <tab heading="Edit Photos">

                <app-photo-editor [member]="member"></app-photo-editor>

            </tab>