**Event Management App**

**Source Code :-**

**Signin :**

**Signin.component.html :**

|  |
| --- |
|  |
| < section class = "vh-100 mt-5"> |
| < div class = "container-fluid h-custom"> |
| < p class = "title text-center"> EventManagement Admin Portal</ p> |
| < div class = "row d-flex justify-content-center align-items-center h-100"> |
| < div class = "col-md-9 col-lg-6 col-xl-5"> |
| < img |
| src = " https://static3.depositphotos.com/1005893/265/i/600/depositphotos\_2652934-stock-photo-mini-laptop.jp |
| g " |
| class = "img-fluid" alt = "Sample image"> |
| </ div> |
| < div class = "col-md-8 col-lg-6 col-xl-4 offset-xl-1"> |
| < form action = "dashboard"> |
|  |
| < div class = "divider d-flex align-items-center my-4"> |
| < p class = "text-center fw-bold mx-3 mb-0"> Sign In</ p> |
| </ div> |
|  |
| <!-- Email input --> |
| < div class = "form-outline mb-4"> |
| < input type = "email" id = "email" class = "form-control form-control-lg" |
| placeholder = "Enter a valid email address" /> |
| < label class = "form-label" for = "email"> Email address</ label> |
| </ div> |
|  |
| <!-- Password input --> |
| < div class = "form-outline mb-3"> |
| < input type = "password" id = "password" class = "form-control form-control-lg" |
| placeholder = "Enter password" /> |
| < label class = "form-label" for = "password"> Password</ label> |
| </ div> |
|  |
| < div class = "d-flex justify-content-between align-items-center"> |
| <!-- Checkbox --> |
| < div class = "form-check mb-0"> |
| < input class = "form-check-input me-2" type = "checkbox" value = "" id = "form2Example3" |
| /> |
| < label class = "form-check-label" for = "form2Example3"> |
| Remember me |
| </ label> |
| </ div> |
| < a href = "#!" class = "text-body"> Forgot password?</ a> |
| </ div> |
|  |
| < div class = "text-center text-lg-start mt-4 pt-2"> |
| < button onclick = " submit ()" type = "submit" class = "btn btn-primary btn-lg" |
| style = "padding-left: 2.5rem; padding-right: 2.5rem;"> Login</ button> |
| </ div> |
|  |
| </ form> |
| </ div> |
| </ div> |
| </ div> |
| </ section> |
|  |

**Signin.component.css :**

|  |
| --- |
|  |
| .title { |
| font-size : xx-large ; |
| font-weight : bold ; |
| } |
|  |

**Signin.component.ts :**

|  |
| --- |
|  |
| import { Component , OnInit } from '@angular/core' ; |
| import { FormBuilder , FormGroup , Validators } from '@angular/forms' ; |
|  |
| @ Component ({ |
| selector: 'app-signin' , |
| templateUrl: './signin.component.html' , |
| styleUrls: [ './signin.component.css' ] |
| }) |
| export class SigninComponent implements OnInit { |
|  |
| ngOnInit () {} |
| } |
|  |

**Signin.component.spec.ts :**

|  |
| --- |
|  |
| import { ComponentFixture , TestBed } from '@angular/core/testing' ; |
|  |
| import { SigninComponent } from './signin.component' ; |
|  |
| describe ( 'SigninComponent' , () => { |
| let component : SigninComponent ; |
| let fixture : ComponentFixture< SigninComponent>; |
|  |
| beforeEach ( async () => { |
| await TestBed . configureTestingModule ({ |
| declarations: [ SigninComponent ] |
| }) |
| . compileComponents (); |
|  |
| fixture = TestBed . createComponent ( SigninComponent ); |
| component = fixture . componentInstance ; |
| fixture . detectChanges (); |
| }); |
|  |
| it ( 'should create' , () => { |
| expect ( component ). toBeTruthy (); |
| }); |
| }); |
|  |

**Dashboard :**

**Dashboard.component.html :**

|  |
| --- |
| < nav class = "navbar bg-primary"> |
| < div class = "container-fluid"> |
| < h3 class = "text-white"> Employee Master List</ h3> |
| < div class = "d-flex"> |
| < button class = "btn btn-success" (click) = "showModel=true"> Add Employee</ button> |
| < form action = "signin">< button type = "submit" class = "mx-3 btn |
| btn-warning"> Logout</ button></ form> |
| < button class = "btn btn-sm btn-outline-dark" (click) = "showPassModel=true"> Change |
| Password</ button> |
| </ div> |
| </ div> |
| </ nav> |
|  |
| < div class = "container mt-5"> |
| < table class = "table text-center table-hover mt-5"> |
| < thead> |
| < tr> |
| < th> ID</ th> |
| < th> Firstname</ th> |
| < th> Lastname</ th> |
| < th> Email</ th> |
| < th> Action</ th> |
| </ tr> |
| </ thead> |
| < tbody> |
| < tr \*ngFor = "let row of employeeData"> |

|  |
| --- |
| < td> {{row.id}}</ td> |
| < td> {{row.first\_name}}</ td> |
| < td> {{row.last\_name}}</ td> |
| < td> {{row.email}}</ td> |
| < td> |
| < button type = "button" (click) = "onEdit(row)" (click) = "showModel=true" class = "btn btn-sm |
| btn-warning"> Edit</ button> |
| < button type = "button" (click) = "deleteEmployee(row)" class = " mx-2 btn btn-sm |
| btn-outline-danger"> Delete</ button> |
| </ td> |
| </ tr> |
| </ tbody> |
| </ table> |
| </ div> |
|  |
| <!-- Modal --> |
| < div class = "modal fade" id = "exampleModal" tabindex = "-1" role = "dialog" |
| [ngClass] = "{'show':showModel}"> |
| < div class = "modal-dialog" role = "document"> |
| < div class = "modal-content"> |
| < div class = "modal-header"> |
| < h5 class = "modal-title" id = "exampleModalLabel"> Employee Details</ h5> |
| < button id = "cancel" (click) = "showModel=false"> |
| < i class = "fa-solid fa-circle-xmark"></ i> |
| </ button> |
|  |
| </ div> |
| < div class = "modal-body"> |
| < form [formGroup] = "formValue"> |
| < div class = "form-group"> |
| < label for = "exampleInputEmail1"> Id</ label> |
| < input type = "number" formControlName = "id" class = "form-control" |
| id = "exampleInputEmail1" aria-describedby = "emailHelp" required> |

|  |
| --- |
| </ div> |
| < div class = "form-group"> |
| < label for = "exampleInputEmail1"> Firstname</ label> |
| < input type = "text" formControlName = "firstname" class = "form-control" |
| id = "exampleInputEmail1" aria-describedby = "emailHelp" placeholder = "John..." required> |
| </ div> |
| < div class = "form-group"> |
| < label for = "exampleInputPassword1"> Lastname</ label> |
| < input type = "text" formControlName = "lastname" class = "form-control" |
| id = "exampleInputPassword1" placeholder = "X.." required> |
| </ div> |
| < div class = "form-group"> |
| < label for = "exampleInputPassword1"> Email</ label> |
| < input type = "email" formControlName = "email" class = "form-control" |
| id = "exampleInputPassword1" placeholder = "John@gmail.com" required> |
| </ div> |
| </ form> |
| </ div> |
| < div class = "modal-footer"> |
| < button type = "button" class = "btn btn-secondary" data-dismiss = "modal" |
| (click) = "showModel=false"> Close</ button> |
| < button type = "submit" class = "btn btn-primary" (click) = "postEmployeeDetails()" |
| > Add</ button> |
| < button type = "submit" class = "btn btn-primary" (click) = "updateEmployeeDetails()" |
| > Update</ button> |
| </ div> |
| </ div> |
| </ div> |
| </ div> |
|  |
| <!-- Modal --> |
| < div class = "modal fade" id = "exampleModal" tabindex = "-1" role = "dialog" |
| aria-labelledby = "exampleModalLabel" aria-hidden = "true" [ngClass] = "{'show':showPassModel}"> |

|  |
| --- |
| < div class = "modal-dialog" role = "document"> |
| < div class = "modal-content"> |
| < div class = "modal-header"> |
| < h5 class = "modal-title" id = "exampleModalLabel"> Change Password</ h5> |
| < button type = "button" class = "close" data-dismiss = "modal" aria-label = "Close" |
| (click) = "showPassModel=false"> |
| < span aria-hidden = "true">&times;</ span> |
| </ button> |
| </ div> |
| < div class = "modal-body"> |
| < form action = ""> |
| < div class = "form-group row"> |
| < label for = "staticEmail" class = "col-sm-5 col-form-label"> Current Password</ label> |
| < div class = "col-sm-5"> |
| < input type = "password" class = "form-control" id = "staticEmail" |
| placeholder = "Password"> |
| </ div> |
| </ div> |
| < div class = "form-group row"> |
| < label for = "inputPassword" class = "col-sm-5 col-form-label"> New Password</ label> |
| < div class = "col-sm-5"> |
| < input type = "password" class = "form-control" id = "inputPassword" |
| placeholder = "lkdes@#%12SD"> |
| </ div> |
| </ div> |
| </ form> |
| </ div> |
| < div class = "modal-footer"> |
| < button type = "button" class = "btn btn-secondary" data-dismiss = "modal" |
| (click) = "showPassModel=false"> Close</ button> |
| < button (click) = "changePass()" type = "button" class = "btn btn-primary"> Update |
| Password</ button> |
| </ div> |
| </ div> |
| </ div> |
| </ div> |
|  |

**Dashboard.component.css :**

|  |
| --- |
|  |
| .modal.show { |
| display : block ; |
|  |
| } |
|  |

**Dashboard.component.ts :**

|  |
| --- |
|  |
| import { Component , OnInit } from '@angular/core' ; |
| import { CrudHttpService } from '../crud-http.service' ; |
| import { FormBuilder , FormGroup } from '@angular/forms' ; |
| import { EmployeeModel } from './employeeDashboard-Model' ; |
|  |
| @ Component ({ |
| selector: 'app-dashboard' , |
| templateUrl: './dashboard.component.html' , |
| styleUrls: [ './dashboard.component.css' ] |
| }) |
| export class DashboardComponent implements OnInit { |
|  |
| formValue !: FormGroup ; |
| empModel : EmployeeModel = new EmployeeModel (); |
| employeeData !: any ; |
|  |
| empList : any = []; |
| showModel : any ; |
| showPassModel : any ; |
|  |
| constructor ( private crudHttpService : CrudHttpService , private formBuilder : FormBuilder ){} |

|  |
| --- |
|  |
| ngOnInit () : void { |
| this . getAllEmployeeDetails (); |
| this . formValue = this . formBuilder . group ({ |
| id: [ '' ], |
| firstname: [ '' ], |
| lastname: [ '' ], |
| email: [ '' ] |
| }) |
| } |
|  |
| postEmployeeDetails (){ |
| this . empModel . id = this . formValue . value . id ; |
| this . empModel . first\_name = this . formValue . value . firstname ; |
| this . empModel . last\_name = this . formValue . value . lastname ; |
| this . empModel . email = this . formValue . value . email ; |
|  |
| this . crudHttpService . postEmployee ( this . empModel ). subscribe ( res => { |
| console . log ( res ); |
| alert ( "Employee Added successfully" ); |
| this . formValue . reset (); |
| let ref = document . getElementById ( 'cancel' ) |
| ref ?. click (); |
| this . getAllEmployeeDetails (); |
| }, |
| err => { |
| alert ( "Something Went Wrong" ) |
| }) |
|  |
| } |
|  |
| getAllEmployeeDetails (){ |
| this . crudHttpService . getEmployee (). subscribe ( res => |
| { |
| this . formValue . reset (); |
| this . employeeData = res ; |
| }) |
| } |
|  |
| deleteEmployee ( row : any ){ |
| this . crudHttpService . deleteEmployee ( row . id ). subscribe ( res => { |
| this . getAllEmployeeDetails (); |
| }) |
|  |
| } |
|  |
| onEdit ( row : any ){ |
| this . empModel . id = row . id ; |
| this . formValue . controls [ 'id' ]. setValue ( row . id ); |
| this . formValue . controls [ 'firstname' ]. setValue ( row . first\_name ); |
| this . formValue . controls [ 'lastname' ]. setValue ( row . last\_name ); |
| this . formValue . controls [ 'email' ]. setValue ( row . email ); |
|  |
| } |
| updateEmployeeDetails (){ |
| this . empModel . id = this . formValue . value . id ; |
| this . empModel . first\_name = this . formValue . value . firstname ; |
| this . empModel . last\_name = this . formValue . value . lastname ; |
| this . empModel . email = this . formValue . value . email ; |
| this . crudHttpService . updateEmployee ( this . empModel , this . empModel . id ). subscribe ( res => { |
| this . formValue . reset (); |
| let ref = document . getElementById ( 'cancel' ) |
| ref ?. click (); |
| this . getAllEmployeeDetails (); |
| }) |
|  |
| } |
| changePass (){ |
| alert ( "Password Updated Successfully" ); |
| } |
|  |
| } |
|  |

**Dashboard.component.spec.ts :**

|  |
| --- |
|  |
| import { ComponentFixture , TestBed } from '@angular/core/testing' ; |
|  |
| import { DashboardComponent } from './dashboard.component' ; |
|  |
| describe ( 'DashboardComponent' , () => { |
| let component : DashboardComponent ; |
| let fixture : ComponentFixture< DashboardComponent>; |
|  |
| beforeEach ( async () => { |
| await TestBed . configureTestingModule ({ |
| declarations: [ DashboardComponent ] |
| }) |
| . compileComponents (); |
|  |
| fixture = TestBed . createComponent ( DashboardComponent ); |
| component = fixture . componentInstance ; |
| fixture . detectChanges (); |
| }); |
|  |
| it ( 'should create' , () => { |
| expect ( component ). toBeTruthy (); |
| }); |
| }); |
|  |

**employeeDashboard-Model.ts :**

|  |
| --- |
|  |
| export class EmployeeModel { |
| id : number = 0 ; |
| first\_name : string = '' ; |
| last\_name : string = '' ; |
| email : string = '' ; |
| } |
|  |

**Db.json :-**

|  |
| --- |
|  |
| { |
| "employees" : [ |
| { |
| "id" : 1 , |
| "first\_name" : "Sebastian" , |
| "last\_name" : "Eschweiler" , |
| "email" : "sebastian@codingthesmartway.com" |
| }, |
| { |
| "id" : 2 , |
| "first\_name" : "Steve" , |
| "last\_name" : "Palmer" , |
| "email" : "steve@codingthesmartway.com" |
| }, |
| { |
| "id" : 3 , |
| "first\_name" : "Ann" , |
| "last\_name" : "Smith" , |
| "email" : "ann@codingthesmartway.com" |
| }, |
| { |
| "id" : 4 , |
| "first\_name" : "James" , |
| "last\_name" : "Watt" , |
| "email" : "James@codingthesmartway.com" |
| }, |
| { |
| "id" : 5 , |
| "first\_name" : "Davi" , |
| "last\_name" : "Jones" , |
| "email" : "davi@codingthesmartway.com" |
| }, |
| { |
| "id" : 6 , |
| "first\_name" : "Luther" , |
| "last\_name" : "Chadwick" , |
| "email" : "luther@codingthesmartway.com" |
| } |
| ] |
| } |
|  |

**Package.json :-**

|  |
| --- |
|  |
| { |
| "name" : "event-management-app" , |
| "version" : "0.0.0" , |
| "scripts" : { |
| "ng" : "ng" , |
| "start" : "ng serve" , |
| "build" : "ng build" , |
| "watch" : "ng build --watch --configuration development" , |
| "test" : "ng test" |
| }, |
| "private" : true , |
| "dependencies" : { |
| "@angular/animations" : "^14.2.0" , |
| "@angular/common" : "^14.2.0" , |
| "@angular/compiler" : "^14.2.0" , |
| "@angular/core" : "^14.2.0" , |
| "@angular/forms" : "^14.2.0" , |
| "@angular/platform-browser" : "^14.2.0" , |
| "@angular/platform-browser-dynamic" : "^14.2.0" , |
| "@angular/router" : "^14.2.0" , |
| "@fortawesome/fontawesome-free" : "^6.2.0" , |
| "bootstrap" : "^5.2.1" , |
| "jquery" : "^3.6.1" , |
| "popper.js" : "^1.16.1" , |
| "rxjs" : "~7.5.0" , |
| "tslib" : "^2.3.0" , |
| "zone.js" : "~0.11.4" |
| }, |
| "devDependencies" : { |
| "@angular-devkit/build-angular" : "^14.2.3" , |
| "@angular/cli" : "~14.2.3" , |
| "@angular/compiler-cli" : "^14.2.0" , |
| "@types/jasmine" : "~4.0.0" , |
| "jasmine-core" : "~4.3.0" , |
| "karma" : "~6.4.0" , |
| "karma-chrome-launcher" : "~3.1.0" , |
| "karma-coverage" : "~2.2.0" , |
| "karma-jasmine" : "~5.1.0" , |
| "karma-jasmine-html-reporter" : "~2.0.0" , |
| "typescript" : "~4.7.2" |
| } |
| } |
|  |

**App.component.html :-**

|  |
| --- |
|  |
| < app-signin></ app-signin> |
| < router-outlet></ router-outlet> |
| <!---<app-dashboard></app-dashboard>---> |
|  |

**App.component.ts :-**

|  |
| --- |
|  |
| import { Component } from '@angular/core' ; |
| import { CrudHttpService } from './crud-http.service' ; |
|  |
| @ Component ({ |
| selector: 'app-root' , |
| templateUrl: './app.component.html' , |
| styleUrls: [ './app.component.css' ] |
| }) |
| export class AppComponent { |
| title = 'Event-management-app' ; |
| } |
|  |

**App.component.spec.ts :-**

|  |
| --- |
|  |
| import { TestBed } from '@angular/core/testing' ; |
| import { RouterTestingModule } from '@angular/router/testing' ; |
| import { AppComponent } from './app.component' ; |
|  |
| describe ( 'AppComponent' , () => { |
| beforeEach ( async () => { |
| await TestBed . configureTestingModule ({ |
| imports: [ |
| RouterTestingModule |
| ], |
| declarations: [ |
| AppComponent |
| ], |
| }). compileComponents (); |
| }); |
|  |
| it ( 'should create the app' , () => { |
| const fixture = TestBed . createComponent ( AppComponent ); |
| const app = fixture . componentInstance ; |
| expect ( app ). toBeTruthy (); |
| }); |
|  |
| it ( `should have as title 'Event-management-app'` , () => { |
| const fixture = TestBed . createComponent ( AppComponent ); |
| const app = fixture . componentInstance ; |
| expect ( app . title ). toEqual ( 'Event-management-app' ); |
| }); |
|  |
| it ( 'should render title' , () => { |
| const fixture = TestBed . createComponent ( AppComponent ); |
| fixture . detectChanges (); |
| const compiled = fixture . nativeElement as HTMLElement ; |
| expect ( compiled . querySelector ( '.content |
| span' )?. textContent ). toContain ( 'Event-management-app app is running!' ); |
| }); |
| }); |
|  |

**App.module.ts :-**

|  |
| --- |
|  |
| import { NgModule } from '@angular/core' ; |
| import { BrowserModule } from '@angular/platform-browser' ; |
| import { HttpClientModule } from '@angular/common/http' ; |
| import { AppRoutingModule } from './app-routing.module' ; |
| import { AppComponent } from './app.component' ; |
| import { SigninComponent } from './signin/signin.component' ; |
| import { DashboardComponent } from './dashboard/dashboard.component' ; |
| import { ReactiveFormsModule } from '@angular/forms' ; |
| import { FormsModule } from '@angular/forms' ; |
|  |
| @ NgModule ({ |
| declarations: [ |
| AppComponent , |
| SigninComponent , |
| DashboardComponent |
| ], |
| imports: [ |
| BrowserModule , |
| AppRoutingModule , |
| HttpClientModule , |
| ReactiveFormsModule , |
| FormsModule |
| ], |
| providers: [], |
| bootstrap: [ AppComponent ] |
| }) |
| export class AppModule { } |
|  |

**app-routing.module.ts :-**

|  |
| --- |
|  |
| import { NgModule } from '@angular/core' ; |
| import { RouterModule , Routes } from '@angular/router' ; |
| import { DashboardComponent } from './dashboard/dashboard.component' ; |
| import { SigninComponent } from './signin/signin.component' ; |
|  |
| const routes : Routes = [ |
| { path: 'signin' , component : SigninComponent }, |
| { path: 'dashboard' , component : DashboardComponent } |
| ]; |
|  |
| @ NgModule ({ |
| imports: [ RouterModule . forRoot ( routes )], |
| exports: [ RouterModule ] |
| }) |
| export class AppRoutingModule { } |
| export const routingComponents = [ SigninComponent , DashboardComponent ] |
|  |

**crud-http.service.ts :-**

|  |
| --- |
|  |
| import { HttpClient , HttpErrorResponse , HttpHeaders } from '@angular/common/http' ; |
| import { Injectable } from '@angular/core' ; |
| import { map , catchError } from 'rxjs/operators' ; |
| import { Observable , throwError } from 'rxjs' ; |
|  |
| @ Injectable ({ |
| providedIn: 'root' |
| }) |
| export class CrudHttpService { |
|  |
| apiUrl : string = 'http://localhost:3000/employees' ; |
| headers = new HttpHeaders (). set ( 'Content-Type' , 'application/json' ); |
|  |
| constructor ( private http : HttpClient ) { } |
|  |
| postEmployee ( data : any ){ |
| return this . http . post< any>( this . apiUrl , data ). pipe ( map (( res : any ) => { |
| return res ; |
| })) |
| } |
|  |
| getEmployee (){ |
| return this . http . get< any>( this . apiUrl ). pipe ( map (( res : any ) => { |
| return res ; |
| })) |
| } |
|  |
| updateEmployee ( data : any , id : number ){ |
| return this . http . put< any>( this . apiUrl + '/' + id , data ). pipe ( map (( res : any ) => { |
| return res ; |
| })) |
| } |
|  |
| deleteEmployee ( id : number ){ |
| return this . http . delete< any>( this . apiUrl + '/' + id ). pipe ( map (( res : any ) => |
| { |
| return res ; |
| })) |
| } |
|  |
| // Handle API errors |
| handleError ( error : HttpErrorResponse ) { |
| if ( error . error instanceof ErrorEvent ) { |
| console . error ( 'An error occurred:' , error . error . message ); |
| } else { |
| console . error ( |
| `Backend returned code ${ error . status } , ` + |
| `body was: ${ error . error } ` ); |
| } |
| return throwError ( |
| 'Something bad happened; please try again later.' ); |
| }; |
|  |
| } |
|  |

**Crud-http.service.spec.ts :-**

|  |
| --- |
|  |
| import { TestBed } from '@angular/core/testing' ; |
| import { CrudHttpService } from './crud-http.service' ; |
|  |
| describe ( 'CrudHttpService' , () => { |
| let service : CrudHttpService ; |
|  |
| beforeEach (() => { |
| TestBed . configureTestingModule ({}); |
| service = TestBed . inject ( CrudHttpService ); |
| }); |
|  |
| it ( 'should be created' , () => { |
| expect ( service ). toBeTruthy (); |
| }); |
| }); |
|  |