**SourceCodes:**

**myaadhar.java:**

**package** com.example.demo.entity;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

@Entity

**public** **class** myaadhar {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

**private** **int** id;

**private** String fullname;

**private** String gender;

**private** String address;

**private** String dob;

**private** **int** phone;

**private** String email;

**private** String imageUrl;

**public** String getImageUrl() {

**return** imageUrl;

}

**public** **void** setImageUrl(String imageUrl) {

**this**.imageUrl = imageUrl;

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getFullname() {

**return** fullname;

}

**public** **void** setFullname(String fullname) {

**this**.fullname = fullname;

}

**public** String getGender() {

**return** gender;

}

**public** **void** setGender(String gender) {

**this**.gender = gender;

}

**public** String getAddress() {

**return** address;

}

**public** **void** setAddress(String address) {

**this**.address = address;

}

**public** String getDob() {

**return** dob;

}

**public** **void** setDob(String dob) {

**this**.dob = dob;

}

**public** **int** getPhone() {

**return** phone;

}

**public** **void** setPhone(**int** phone) {

**this**.phone = phone;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

@Override

**public** String toString() {

**return** "myaadhar [id=" + id + ", fullname=" + fullname + ", gender=" + gender + ", address=" + address

+ ", dob=" + dob + ", phone=" + phone + ", email=" + email + ", imageUrl=" + imageUrl + "]";

}

}

**MyAadharRepository:**

**package** com.example.demo.repository;

**import** java.util.Optional;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** com.example.demo.entity.myaadhar;

**public** **interface** MyAadharRepository **extends** JpaRepository<myaadhar, Integer> {

Optional<myaadhar> findById(**int** id);

}

**AppService.java**

**package** com.example.demo.service;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** com.example.demo.entity.myaadhar;

**import** com.example.demo.exceptions.UserNotFoundException;

**import** com.example.demo.repository.MyAadharRepository;

@Service

**public** **class** AppService {

**private** **final** MyAadharRepository repo;

@Autowired

**public** AppService(MyAadharRepository repo) {

**this**.repo = repo;

}

**public** List<myaadhar> findallRecords(){

**return** repo.findAll();

}

**public** myaadhar add(myaadhar x) {

**return** repo.save(x);

}

**public** myaadhar update(myaadhar x) {

**return** repo.save(x);

}

**public** myaadhar findmyaadharById(**int** id) {

**return** repo.findById(id)

.orElseThrow(() -> **new** UserNotFoundException("User by id " + id + " not found"));

}

**public** **void** delete(**int** id) {

repo.deleteById(id);

}

}

**UserNotFoundException.java**:

**package** com.example.demo.exceptions;

**public** **class** UserNotFoundException **extends** RuntimeException {

**public** UserNotFoundException(String message) {

**super**(message);

}

}

**AppController.java:**

**package** com.example.demo.controller;

**import** java.util.List;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.DeleteMapping;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.PostMapping;

**import** org.springframework.web.bind.annotation.PutMapping;

**import** org.springframework.web.bind.annotation.RequestBody;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

**import** com.example.demo.entity.myaadhar;

**import** com.example.demo.service.AppService;

@RestController

@RequestMapping("/aadhar")

**public** **class** AppController {

**private** **final** AppService appservice;

**public** AppController(AppService appservice) {

**this**.appservice=appservice;

}

@GetMapping("/all")

**public** ResponseEntity<List<myaadhar>> getrecords(){

List<myaadhar> records=appservice.findallRecords();

**return** **new** ResponseEntity<>(records,HttpStatus.***OK***);

}

@GetMapping("/find/{id}")

**public** ResponseEntity<myaadhar> getaadharById(@PathVariable("id") **int** id){

myaadhar x=appservice.findmyaadharById(id);

**return** **new** ResponseEntity<>(x,HttpStatus.***OK***);

}

@PostMapping("/add")

ResponseEntity<myaadhar> addRequest(@RequestBody myaadhar x){

myaadhar y=appservice.add(x);

**return** **new** ResponseEntity<>(y,HttpStatus.***CREATED***);

}

@PutMapping("/update")

ResponseEntity<myaadhar> updateRequest(@RequestBody myaadhar x){

myaadhar y=appservice.update(x);

**return** **new** ResponseEntity<>(y,HttpStatus.***OK***);

}

@DeleteMapping("/delete/{id}")

ResponseEntity<?> deleteRequest(@PathVariable("id") **int** id){

appservice.delete(id);

**return** **new** ResponseEntity<>(HttpStatus.***OK***);

}

**Pom.xml**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<project xmlns=*"http://maven.apache.org/POM/4.0.0"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.7.4</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<groupId>com.example</groupId>

<artifactId>MyApp</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>MyApp</name>

<description>Demo project for Spring Boot</description>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.apache.tomcat.embed</groupId>

<artifactId>tomcat-embed-jasper</artifactId>

<version>9.0.44</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-tomcat</artifactId>

<version>2.4.4</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

<dependency>

<groupId>taglibs</groupId>

<artifactId>standard</artifactId>

<version>1.1.2</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

Index.html:

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8" />

    <title>FrontendApp</title>

    <base href="/" />

    <meta name="viewport" content="width=device-width, initial-scale=1" />

    <link rel="icon" type="image/x-icon" href="favicon.ico" />

  </head>

  <body>

    <app-root></app-root>

    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"></script>

    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.6/umd/popper.min.js"></script>

    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/js/bootstrap.min.js"></script>

  </body>

</html>

Test.ts

// This file is required by karma.conf.js and loads recursively all the .spec and framework files

import 'zone.js/testing';

import { getTestBed } from '@angular/core/testing';

import {

  BrowserDynamicTestingModule,

  platformBrowserDynamicTesting

} from '@angular/platform-browser-dynamic/testing';

declare const require: {

  context(path: string, deep?: boolean, filter?: RegExp): {

    <T>(id: string): T;

    keys(): string[];

  };

};

// First, initialize the Angular testing environment.

getTestBed().initTestEnvironment(

  BrowserDynamicTestingModule,

  platformBrowserDynamicTesting(),

);

// Then we find all the tests.

const context = require.context('./', true, /\.spec\.ts$/);

// And load the modules.

context.keys().map(context);

main.ts

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

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';

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

if (environment.production) {

  enableProdMode();

}

platformBrowserDynamic().bootstrapModule(AppModule)

  .catch(err => console.error(err));

App.component.html

<!-- <app-myaadhar></app-myaadhar> -->

<!-- <app-admin></app-admin> -->

<!-- <app-myaadhar></app-myaadhar> -->

<!-- <app-user></app-user> -->

<!-- <app-myaadhar2></app-myaadhar2> -->

<!-- <app-home></app-home> -->

<div class="container">

  <div class="row">

    <div class="col">

      <ul class="nav nav-tabs">

        <li

          role="presentation"

          class="nav-item"

          routerLinkActive="active"

          [routerLinkActiveOptions]="{ exact: true }"

        >

          <a class="nav-link" routerLink="/">Home</a>

        </li>

        <li role="presentation" class="nav-item" routerLinkActive="active">

          <a class="nav-link" routerLink="admin"></a>

        </li>

        <li role="presentation" class="nav-item" routerLinkActive="active">

          <a class="nav-link" routerLink="user"></a>

        </li>

        <li role="presentation" class="nav-item" routerLinkActive="active">

          <a class="nav-link" routerLink="myaadhar"></a>

        </li>

        <li role="presentation" class="nav-item" routerLinkActive="active">

          <a class="nav-link" routerLink="myaadhar2"></a>

        </li>

      </ul>

    </div>

  </div>

  <div class="row">

    <div class="col">

      <router-outlet></router-outlet>

    </div>

  </div>

</div>

App.module.ts

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

import { BrowserModule } from '@angular/platform-browser';

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

import { AppComponent } from './app.component';

import { AppService } from './myaadhar/myaadhar.service';

import { FormsModule, ReactiveFormsModule } from '@angular/forms';

import { MyAadharComponent } from './myaadhar/myaadhar.component';

import { AdminComponent } from './admin/admin.component';

import { HomeComponent } from './home/home.component';

import { UserComponent } from './user/user.component';

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

import { MyAadhar2Component } from './myaadhar2/myaadhar2.component';

const appRoutes: Routes = [

  { path: '', component: HomeComponent },

  { path: 'admin', component: AdminComponent },

  { path: 'user', component: UserComponent },

  { path: 'myaadhar', component: MyAadharComponent },

  { path: 'myaadhar2', component: MyAadhar2Component },

];

@NgModule({

  declarations: [

    AppComponent,

    MyAadharComponent,

    AdminComponent,

    HomeComponent,

    UserComponent,

    MyAadhar2Component

  ],

  imports: [

    BrowserModule,

    HttpClientModule,

    FormsModule,

    ReactiveFormsModule,

    RouterModule.forRoot(appRoutes)

  ],

  providers: [AppService],

  bootstrap: [AppComponent]

})

export class AppModule { }

app.component.ts

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

@Component({

  selector: 'app-root',

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

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  title = 'form-tutorial';

}

Admin.component.html

<div class="form-container">

<h2 class="text-center">Admin Login</h2>

<div

  class="container mx-200 mt-10 border justify-center"

  style="width: 400px; background-color: rgb(100, 198, 196)"

>

  <h4 class="text-center">Login Page</h4>

  <form [formGroup]="signupForm" (ngSubmit)="onSubmit()" class="form-center">

    <div class="row">

      <div class="form-group">

        <label for="username">Username</label>

        <input

          type="text"

          id="username"

          class="form-control"

          formControlName="username"

        />

        <span

          class="small"

          \*ngIf="

            !signupForm.get('username').valid &&

            signupForm.get('username').touched

          "

          >Please enter a valid username</span

        >

      </div>

    </div>

    <div class="row">

      <div class="form-group">

        <label for="password">Password</label>

        <input

          type="password"

          id="password"

          class="form-control"

          formControlName="password"

        />

        <span

          class="small"

          \*ngIf="

            !signupForm.get('password').valid &&

            signupForm.get('password').touched

          "

          >Incorrect Password!</span

        >

      </div>

    </div>

    <button

      class="btn btn-primary m-2"

      type="submit"

      [disabled]="!signupForm.valid"

    >

      Submit

    </button>

  </form>

</div>

</div>

Admin.component.css

input.ng-invalid.ng-touched {

    border: 2px solid red;

  }

  .form-container{

    text-align: center;

    border: 1px solid black;

    padding: 1%;

    background-color: aliceblue;

}

Admin.component.ts

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

import { FormControl, FormGroup, Validators } from '@angular/forms';

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

@Component({

  selector: 'app-admin',

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

  styleUrls: ['./admin.component.css'],

})

export class AdminComponent implements OnInit {

  constructor(private router: Router) {}

  signupForm: FormGroup;

  ngOnInit(): void {

    this.signupForm = new FormGroup({

      username: new FormControl(null, Validators.required),

      password: new FormControl(null, [Validators.required]),

    });

  }

  onSubmit() {

    console.log(this.signupForm.value);

    this.router.navigate(['/myaadhar']);

  }

}

[Home.component.html](http://Home.component.html)

<div class="form-container">

<h2 class="text-center">Welcome to mAadhar </h2>

<p class="text-center"><b> Login to Continue!!</b></p>

<div class="container mt-50 text-center">

  <button class="btn btn-info mt-5" (click)="gotoAdmin()">Admin LogIn</button

  ><br />

  <button class="btn btn-success mt-5" (click)="gotoCitizen()">

    Citizen LogIn</button

  ><br />

  <button class="btn btn-primary mt-5" (click)="gotoRegister()">

    Register New Citizen

  </button>

</div>

</div>

[Home.component.css](http://Home.component.css)

hr.new5 {

    border: 8px solid #6D214F;

    border-radius: 5px;

  }

  .form-container{

    text-align: center;

    border: 1px solid black;

    padding: 1%;

    background-color: aliceblue;

}

[Home.component.ts](http://Home.component.ts)

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

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

@Component({

  selector: 'app-home',

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

  styleUrls: ['./home.component.css']

})

export class HomeComponent implements OnInit {

  constructor(private router: Router) {}

  ngOnInit(): void {

  }

  gotoAdmin() {

    this.router.navigate(['/admin']);

  }

  gotoCitizen() {

    this.router.navigate(['/user']);

  }

  gotoRegister(){

    this.router.navigate(['/myaadhar2']);

  }

}

User.component.html

<h1 class="text-center">Please Login</h1>

<div

  class="container mx-200 mt-10 border justify-center"

  style="width: 400px; background-color: darkgrey"

>

  <h4 class="text-center">Login Page</h4>

  <form [formGroup]="signupForm" (ngSubmit)="onSubmit()" class="form-center">

    <div class="row">

      <div class="form-group">

        <label for="username">Username</label>

        <input

          type="text"

          id="username"

          class="form-control"

          formControlName="username"

        />

        <span

          class="small"

          \*ngIf="

            !signupForm.get('username').valid &&

            signupForm.get('username').touched

          "

          >Please enter a valid username</span

        >

      </div>

    </div>

    <div class="row">

      <div class="form-group">

        <span><i>Enter Mobile number as Password</i></span

        ><br />

        <label for="password">Password</label>

        <input

          type="text"

          id="password"

          class="form-control"

          formControlName="password"

        />

        <span

          class="small"

          \*ngIf="

            !signupForm.get('password').valid &&

            signupForm.get('password').touched

          "

          >Incorrest Password!</span

        >

      </div>

    </div>

    <button

      class="btn btn-primary m-2"

      type="submit"

      [disabled]="!signupForm.valid"

    >

      Submit

    </button>

  </form>

</div>

User.component.css

input.ng-invalid.ng-touched {

    border: 2px solid red;

  }

User.component.ts

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

import { FormControl, FormGroup, Validators } from '@angular/forms';

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

@Component({

  selector: 'app-user',

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

  styleUrls: ['./user.component.css']

})

export class UserComponent implements OnInit {

  constructor(private router: Router) {}

  signupForm: FormGroup;

  ngOnInit(): void {

    this.signupForm = new FormGroup({

      username: new FormControl(null, Validators.required),

      password: new FormControl(null, [Validators.required]),

    });

  }

  onSubmit() {

    console.log(this.signupForm.value);

    this.router.navigate(['/myaadhar2']);

  }

}

Myaadhar.component.html

<h1 class="text-center">

  <i><u>Admin Portal</u></i>

</h1>

<nav class="navbar navbar-expand-lg navbar-dark bg-dark">

  <a class="navbar-brand" style="color: white">AadharApplication</a>

  <button

    class="navbar-toggler"

    type="button"

    data-toggle="collapse"

    data-target="#navbarColor02"

    aria-controls="navbarColor02"

    aria-expanded="false"

    aria-label="Toggle navigation"

  >

    <span class="navbar-toggler-icon"></span>

  </button>

  <div class="collapse navbar-collapse" id="navbarColor02">

    <!-- <ul class="navbar-nav mr-auto">

      <li class="nav-item active">

        <a class="nav-link" (click)="onOpenModal(null, 'add')"

          >New Citizen Register <span class="sr-only">(current)</span></a

        >

      </li>

    </ul> -->

    <form class="form-inline my-2 my-lg-0">

      <input

        type="search"

        (ngModelChange)="searchCitizen(key.value)"

        #key="ngModel"

        ngModel

        name="key"

        id="searchName"

        class="form-control mr-sm-2"

        placeholder="Search Citizen..."

        required

      />

    </form>

  </div>

</nav>

<div class="container" id="main-container">

  <div class="row">

    <div \*ngFor="let record of records" class="col-md-6 col-xl-3">

      <div class="card m-b-30">

        <div class="card-body row">

          <div class="col-6">

            <a href=""

              ><img

                src="{{ record?.imageUrl }}"

                alt=""

                class="img-fluid rounded-circle w-60"

            /></a>

          </div>

          <div class="col-6 card-title align-self-center mb-0">

            <h5>{{ record?.fullname }}</h5>

            <p class="m-0">{{ record?.gender }}</p>

          </div>

        </div>

        <ul class="list-group list-group-flush">

          <li class="list-group-item">

            <i class="fa fa-phone float-right"></i>DOB :

            {{ record?.dob }}

          </li>

          <li class="list-group-item">

            <i class="fa fa-phone float-right"></i>Address :

            {{ record?.address }}

          </li>

          <li class="list-group-item">

            <i class="fa fa-envelope float-right"></i>{{ record?.email }}

          </li>

          <li class="list-group-item">

            <i class="fa fa-phone float-right"></i>Phone : {{ record?.phone }}

          </li>

        </ul>

        <div class="card-body">

          <div class="float-right btn-group btn-group-sm">

            <a

              (click)="onOpenModal(record, 'edit')"

              class="btn btn-primary tooltips"

              data-placement="top"

              data-original-title="Edit"

              ><i class="fa fa-pencil">Edit</i>

            </a>

            <a

              (click)="onOpenModal(record, 'delete')"

              class="btn btn-secondary tooltips"

              data-placement="top"

              data-original-title="Delete"

              ><i class="fa fa-times">Delete</i></a

            >

          </div>

        </div>

      </div>

    </div>

  </div>

  <!-- Add Employee Modal -->

  <div

    class="modal fade"

    id="addCitizenModal"

    tabindex="-1"

    role="dialog"

    aria-labelledby="addEmployeeModalLabel"

    aria-hidden="true"

  >

    <div class="modal-dialog" role="document">

      <div class="modal-content">

        <div class="modal-header">

          <h5 class="modal-title" id="addEmployeeModalLabel">

            New Citizen Register

          </h5>

          <button

            type="button"

            class="close"

            data-dismiss="modal"

            aria-label="Close"

          >

            <span aria-hidden="true">&times;</span>

          </button>

        </div>

        <div class="modal-body">

          <form #addForm="ngForm" (ngSubmit)="onAddCitizen(addForm)">

            <div class="form-group">

              <label for="fullname">Name</label>

              <input

                type="text"

                ngModel

                name="fullname"

                class="form-control"

                id="fullname"

                placeholder="Name"

                required

              />

            </div>

            <div class="form-group">

              <label for="dob">D.O.B</label>

              <input

                type="date"

                ngModel

                name="dob"

                class="form-control"

                id="dob"

                placeholder="dob"

                required

              />

            </div>

            <div class="form-group">

              <label for="email">Email Address</label>

              <input

                type="email"

                ngModel

                name="email"

                class="form-control"

                id="email"

                placeholder="Email"

                required

              />

            </div>

            <div class="form-group">

              <label for="address">Home Address</label>

              <input

                type="text"

                ngModel

                name="address"

                class="form-control"

                id="address"

                placeholder="Address"

                required

              />

            </div>

            <div class="form-group">

              <label for="gender">Gender</label>

              <input

                type="text"

                ngModel

                name="gender"

                class="form-control"

                id="gender"

                placeholder="Gender"

                required

              />

            </div>

            <div class="form-group">

              <label for="phone">Phone</label>

              <input

                type="text"

                ngModel

                name="phone"

                class="form-control"

                id="phone"

                placeholder="Phone"

                required

              />

            </div>

            <div class="form-group">

              <label for="phone">Image URL</label>

              <input

                type="text"

                ngModel

                name="imageUrl"

                class="form-control"

                id="imageUrl"

                placeholder="Image URL"

                required

              />

            </div>

            <div class="modal-footer">

              <button

                type="button"

                id="add-record-form"

                class="btn btn-secondary"

                data-dismiss="modal"

              >

                Close

              </button>

              <button

                [disabled]="addForm.invalid"

                type="submit"

                class="btn btn-primary"

              >

                Save changes

              </button>

            </div>

          </form>

        </div>

      </div>

    </div>

  </div>

  <!-- Edit Modal -->

  <div

    class="modal fade"

    id="updateCitizenModal"

    tabindex="-1"

    role="dialog"

    aria-labelledby="employeeEditModalLabel"

    aria-hidden="true"

  >

    <div class="modal-dialog" role="document">

      <div class="modal-content">

        <div class="modal-header">

          <h5 class="modal-title" id="updateEmployeeModalLabel">

            Edit Citizen {{ editRecord?.fullname }}

          </h5>

          <button

            type="button"

            class="close"

            data-dismiss="modal"

            aria-label="Close"

          >

            <span aria-hidden="true">&times;</span>

          </button>

        </div>

        <div class="modal-body">

          <form #editForm="ngForm">

            <div class="form-group">

              <label for="fullname">Name</label>

              <input

                type="text"

                ngModel="{{ editRecord?.fullname }}"

                name="fullname"

                class="form-control"

                id="fullname"

                aria-describedby="emailHelp"

                placeholder="Name"

              />

            </div>

            <div class="form-group">

              <label for="gender">Gender</label>

              <input

                type="text"

                ngModel="{{ editRecord?.gender }}"

                name="gender"

                class="form-control"

                id="gender"

                aria-describedby="emailHelp"

                placeholder="Gender"

              />

            </div>

            <input

              type="hidden"

              ngModel="{{ editRecord?.id }}"

              name="id"

              class="form-control"

              id="id"

              placeholder="Email"

            />

            <div class="form-group">

              <label for="email">Email Address</label>

              <input

                type="email"

                ngModel="{{ editRecord?.email }}"

                name="email"

                class="form-control"

                id="email"

                placeholder="Email"

              />

            </div>

            <div class="form-group">

              <label for="dob">D.O.B</label>

              <input

                type="date"

                ngModel="{{ editRecord?.dob }}"

                name="dob"

                class="form-control"

                id="dob"

                placeholder="DOB"

              />

            </div>

            <div class="form-group">

              <label for="address">Home Address</label>

              <input

                type="text"

                ngModel="{{ editRecord?.address }}"

                name="address"

                class="form-control"

                id="address"

                placeholder="Address"

              />

            </div>

            <div class="form-group">

              <label for="phone">Phone</label>

              <input

                type="text"

                ngModel="{{ editRecord?.phone }}"

                name="phone"

                class="form-control"

                id="phone"

                name="phone"

                placeholder="Phone"

              />

            </div>

            <div class="form-group">

              <label for="phone">Image URL</label>

              <input

                type="text"

                ngModel="{{ editRecord?.imageUrl }}"

                name="imageUrl"

                class="form-control"

                id="imageUrl"

                placeholder="Image URL"

              />

            </div>

            <div class="modal-footer">

              <button

                type="button"

                id=""

                data-dismiss="modal"

                class="btn btn-secondary"

              >

                Close

              </button>

              <button

                (click)="onUpdateCitizen(editForm.value)"

                data-dismiss="modal"

                class="btn btn-primary"

              >

                Save changes

              </button>

            </div>

          </form>

        </div>

      </div>

    </div>

  </div>

  <!-- Delete Modal -->

  <div

    class="modal fade"

    id="deleteCitizenModal"

    tabindex="-1"

    role="dialog"

    aria-labelledby="deleteModelLabel"

    aria-hidden="true"

  >

    <div class="modal-dialog" role="document">

      <div class="modal-content">

        <div class="modal-header">

          <h5 class="modal-title" id="deleteModelLabel">Delete Citizen</h5>

          <button

            type="button"

            class="close"

            data-dismiss="modal"

            aria-label="Close"

          >

            <span aria-hidden="true">&times;</span>

          </button>

        </div>

        <div class="modal-body">

          <p>

            Are you sure you want to delete

            {{ deleteRecord?.fullname }}?

          </p>

          <div class="modal-footer">

            <button

              type="button"

              class="btn btn-secondary"

              data-dismiss="modal"

            >

              No

            </button>

            <button

              (click)="onDeleteCitizen(deleteRecord?.id)"

              class="btn btn-danger"

              data-dismiss="modal"

            >

              Yes

            </button>

          </div>

        </div>

      </div>

    </div>

  </div>

</div>

<!-- Notification for no employees -->

<div \*ngIf="records?.length == 0" class="col-lg-12 col-md-12 col-xl-12">

  <div class="alert alert-info" role="alert">

    <h4 class="alert-heading">NO EMPLOYEES!</h4>

    <p>No Employees were found.</p>

  </div>

</div>

Myaadhar.component.ts

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

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

import { myaadhar } from "./myaadhar";

import { AppService } from './myaadhar.service';

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

@Component({

  selector: 'app-myaadhar',

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

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

})

export class MyAadharComponent implements OnInit{

  public records: myaadhar[];

  public editRecord: myaadhar;

  public deleteRecord: myaadhar;

  constructor(private appService: AppService){ }

  ngOnInit(): void {

    this.get();

  }

  public get(): void {

    this.appService.get().subscribe(

      (response: myaadhar[]) => {

        this.records=response;

        console.log(this.records);

      },

        (error: HttpErrorResponse) => {

          alert (error.message);

        }

        );

  }

  public onAddCitizen(addForm: NgForm): void {

    document.getElementById('add-record-form').click();

    this.appService.add(addForm.value).subscribe(

      (response: myaadhar) => {

       console.log(response);

       this.get();

       addForm.reset();

      },

      (error: HttpErrorResponse) => {

        alert(error.message);

        addForm.reset();

      }

    );

  }

  public onUpdateCitizen(record: myaadhar): void {

    this.appService.update(record).subscribe(

      (response: myaadhar) => {

       console.log(response);

       this.get();

      },

      (error: HttpErrorResponse) => {

        alert(error.message);

      }

    );

  }

  public searchCitizen(key: string): void {

    console.log(key);

    const results: myaadhar[] = [];

    for (const record of this.records) {

      if (record.fullname.toLowerCase().indexOf(key.toLowerCase()) !== -1

      || record.email.toLowerCase().indexOf(key.toLowerCase()) !== -1

      || record.dob.toLowerCase().indexOf(key.toLowerCase()) !== -1

      || record.address.toLowerCase().indexOf(key.toLowerCase()) !== -1) {

        results.push(record);

      }

    }

    this.records = results;

    if (results.length === 0 || !key) {

      this.get();

    }

  }

  public onDeleteCitizen(id: number): void {

    this.appService.delete(id).subscribe(

      (response: void) => {

       console.log(response);

       this.get();

      },

      (error: HttpErrorResponse) => {

        alert(error.message);

      }

    );

  }

  public onOpenModal(record: myaadhar, mode: string): void {

    const container = document.getElementById('main-container');

    const button = document.createElement('button');

    button.type = 'button';

    button.style.display = 'none';

    button.setAttribute('data-toggle', 'modal');

    if (mode === 'add') {

      button.setAttribute('data-target', '#addCitizenModal');

    }

    if (mode === 'edit') {

      this.editRecord = record;

      button.setAttribute('data-target', '#updateCitizenModal');

    }

    if (mode === 'delete') {

      this.deleteRecord=record;

      button.setAttribute('data-target', '#deleteCitizenModal');

    }

    container.appendChild(button);

    button.click();

  }

}

Myaadhar2.component.html

<h1 class="text-center">

  <p>Registe Yourself</p>

</h1>

<nav class="navbar navbar-expand-lg navbar-dark bg-dark">

  <a class="navbar-brand" style="color: white">AadharApplication</a>

  <button

    class="navbar-toggler"

    type="button"

    data-toggle="collapse"

    data-target="#navbarColor02"

    aria-controls="navbarColor02"

    aria-expanded="false"

    aria-label="Toggle navigation"

  >

    <span class="navbar-toggler-icon"></span>

  </button>

  <div class="collapse navbar-collapse" id="navbarColor02">

    <ul class="navbar-nav mr-auto">

      <li class="nav-item active">

        <a class="nav-link" (click)="onOpenModal(null, 'add')"

          >New Citizen Register <span class="sr-only">(current)</span></a

        >

      </li>

    </ul>

    <form class="form-inline my-2 my-lg-0">

      <input

        type="search"

        (ngModelChange)="searchCitizen(key.value)"

        #key="ngModel"

        ngModel

        name="key"

        id="searchName"

        class="form-control mr-sm-2"

        placeholder="Enter your Name.."

        required

      />

    </form>

  </div>

</nav>

<!---->

<div class="container" id="main-container">

  <div class="row">

    <div \*ngFor="let record of records" class="col-md-6 col-xl-3">

      <div class="card m-b-30">

        <div class="card-body row">

          <div class="col-6">

            <a href=""

              ><img

                src="{{ record?.imageUrl }}"

                alt=""

                class="img-fluid rounded-circle w-60"

            /></a>

          </div>

          <div class="col-6 card-title align-self-center mb-0">

            <h5>{{ record?.fullname }}</h5>

            <p class="m-0">{{ record?.gender }}</p>

          </div>

        </div>

        <ul class="list-group list-group-flush">

          <li class="list-group-item">

            <i class="fa fa-phone float-right"></i>DOB :

            {{ record?.dob }}

          </li>

          <li class="list-group-item">

            <i class="fa fa-phone float-right"></i>Address :

            {{ record?.address }}

          </li>

          <li class="list-group-item">

            <i class="fa fa-envelope float-right"></i>{{ record?.email }}

          </li>

          <li class="list-group-item">

            <i class="fa fa-phone float-right"></i>Phone : {{ record?.phone }}

          </li>

        </ul>

        <div class="card-body">

          <div class="float-right btn-group btn-group-sm">

            <a

              (click)="onOpenModal(record, 'edit')"

              class="btn btn-primary tooltips"

              data-placement="top"

              data-original-title="Edit"

              ><i class="fa fa-pencil">Edit</i>

            </a>

            <a

              (click)="OnDeleteRequest()"

              class="btn btn-secondary tooltips"

              data-placement="top"

              data-original-title="Delete"

              ><i class="fa fa-times">Delete</i></a

            >

          </div>

        </div>

      </div>

    </div>

  </div>

  <!-- Add Employee Modal -->

  <div

    class="modal fade"

    id="addCitizenModal"

    tabindex="-1"

    role="dialog"

    aria-labelledby="addEmployeeModalLabel"

    aria-hidden="true"

  >

    <div class="modal-dialog" role="document">

      <div class="modal-content">

        <div class="modal-header">

          <h5 class="modal-title" id="addEmployeeModalLabel">

            New Citizen Register

          </h5>

          <button

            type="button"

            class="close"

            data-dismiss="modal"

            aria-label="Close"

          >

            <span aria-hidden="true">&times;</span>

          </button>

        </div>

        <div class="modal-body">

          <form #addForm="ngForm" (ngSubmit)="onAddCitizen(addForm)">

            <div class="form-group">

              <label for="fullname">Name</label>

              <input

                type="text"

                ngModel

                name="fullname"

                class="form-control"

                id="fullname"

                placeholder="Name"

                required

              />

            </div>

            <div class="form-group">

              <label for="dob">D.O.B</label>

              <input

                type="date"

                ngModel

                name="dob"

                class="form-control"

                id="dob"

                placeholder="dob"

                required

              />

            </div>

            <div class="form-group">

              <label for="email">Email Address</label>

              <input

                type="email"

                ngModel

                name="email"

                class="form-control"

                id="email"

                placeholder="Email"

                required

              />

            </div>

            <div class="form-group">

              <label for="address">Home Address</label>

              <input

                type="text"

                ngModel

                name="address"

                class="form-control"

                id="address"

                placeholder="Address"

                required

              />

            </div>

            <div class="form-group">

              <label for="gender">Gender</label>

              <input

                type="text"

                ngModel

                name="gender"

                class="form-control"

                id="gender"

                placeholder="Gender"

                required

              />

            </div>

            <div class="form-group">

              <label for="phone">Phone</label>

              <input

                type="text"

                ngModel

                name="phone"

                class="form-control"

                id="phone"

                placeholder="Phone"

                required

              />

            </div>

            <div class="form-group">

              <label for="phone">Image URL</label>

              <input

                type="text"

                ngModel

                name="imageUrl"

                class="form-control"

                id="imageUrl"

                placeholder="Image URL"

                required

              />

            </div>

            <div class="modal-footer">

              <button

                type="button"

                id="add-record-form"

                class="btn btn-secondary"

                data-dismiss="modal"

              >

                Close

              </button>

              <button

                [disabled]="addForm.invalid"

                type="submit"

                class="btn btn-primary"

              >

                Save changes

              </button>

            </div>

          </form>

        </div>

      </div>

    </div>

  </div>

  <div

    class="modal fade"

    id="updateCitizenModal"

    tabindex="-1"

    role="dialog"

    aria-labelledby="employeeEditModalLabel"

    aria-hidden="true"

  >

    <div class="modal-dialog" role="document">

      <div class="modal-content">

        <div class="modal-header">

          <h5 class="modal-title" id="updateEmployeeModalLabel">

            Edit Citizen {{ editRecord?.fullname }}

          </h5>

          <button

            type="button"

            class="close"

            data-dismiss="modal"

            aria-label="Close"

          >

            <span aria-hidden="true">&times;</span>

          </button>

        </div>

        <div class="modal-body">

          <form #editForm="ngForm">

            <div class="form-group">

              <label for="fullname">Name</label>

              <input

                type="text"

                ngModel="{{ editRecord?.fullname }}"

                name="fullname"

                class="form-control"

                id="fullname"

                aria-describedby="emailHelp"

                placeholder="Name"

              />

            </div>

            <div class="form-group">

              <label for="gender">Gender</label>

              <input

                type="text"

                ngModel="{{ editRecord?.gender }}"

                name="gender"

                class="form-control"

                id="gender"

                aria-describedby="emailHelp"

                placeholder="Gender"

              />

            </div>

            <input

              type="hidden"

              ngModel="{{ editRecord?.id }}"

              name="id"

              class="form-control"

              id="id"

              placeholder="Email"

            />

            <div class="form-group">

              <label for="email">Email Address</label>

              <input

                type="email"

                ngModel="{{ editRecord?.email }}"

                name="email"

                class="form-control"

                id="email"

                placeholder="Email"

              />

            </div>

            <div class="form-group">

              <label for="dob">D.O.B</label>

              <input

                type="date"

                ngModel="{{ editRecord?.dob }}"

                name="dob"

                class="form-control"

                id="dob"

                placeholder="DOB"

              />

            </div>

            <div class="form-group">

              <label for="address">Home Address</label>

              <input

                type="text"

                ngModel="{{ editRecord?.address }}"

                name="address"

                class="form-control"

                id="address"

                placeholder="Address"

              />

            </div>

            <div class="form-group">

              <label for="phone">Phone</label>

              <input

                type="text"

                ngModel="{{ editRecord?.phone }}"

                name="phone"

                class="form-control"

                id="phone"

                name="phone"

                placeholder="Phone"

              />

            </div>

            <div class="form-group">

              <label for="phone">Image URL</label>

              <input

                type="text"

                ngModel="{{ editRecord?.imageUrl }}"

                name="imageUrl"

                class="form-control"

                id="imageUrl"

                placeholder="Image URL"

              />

            </div>

            <div class="modal-footer">

              <button

                type="button"

                id=""

                data-dismiss="modal"

                class="btn btn-secondary"

              >

                Close

              </button>

              <button

                (click)="OnUpdateRequest()"

                data-dismiss="modal"

                class="btn btn-primary"

              >

                Save changes

              </button>

            </div>

          </form>

        </div>

      </div>

    </div>

  </div>

</div>

Myaadhar2.component.ts:

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

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

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

import { myaadhar } from "./myaadhar";

import { AppService } from './myaadhar.service';

@Component({

  selector: 'app-myaadhar2',

  templateUrl: './myaadhar2.component.html',

  styleUrls: ['./myaadhar2.component.css']

})

export class MyAadhar2Component implements OnInit{

  public records: myaadhar[];

  public editRecord: myaadhar;

  public deleteRecord: myaadhar;

  constructor(private appService: AppService){ }

  ngOnInit(): void {

    this.get();

  }

  public get(): void {

    this.appService.get().subscribe(

      (response: myaadhar[]) => {

        this.records=response;

        console.log(this.records);

      },

        (error: HttpErrorResponse) => {

          alert (error.message);

        }

        );

  }

  public searchCitizen(key: string): void {

    console.log(key);

    const results: myaadhar[] = [];

    for (const record of this.records) {

      if (record.fullname.toLowerCase().indexOf(key.toLowerCase()) !== -1

      || record.email.toLowerCase().indexOf(key.toLowerCase()) !== -1

      || record.dob.toLowerCase().indexOf(key.toLowerCase()) !== -1

      || record.address.toLowerCase().indexOf(key.toLowerCase()) !== -1) {

        results.push(record);

      }

    }

    this.records = results;

    if (results.length === 0 || !key) {

      this.get();

    }

  }

  public getElementById(id: number): void {

    this.appService.findmyaadharById(id).subscribe(

      (response: myaadhar) => {

        console.log(response);

        // this.get();

       },

       (error: HttpErrorResponse) => {

         alert(error.message);

       }

    )

  }

  public onUpdateCitizen(record: myaadhar): void {

    this.appService.update(record).subscribe(

      (response: myaadhar) => {

       console.log(response);

       this.get();

      },

      (error: HttpErrorResponse) => {

        alert(error.message);

      }

    );

  }

  OnUpdateRequest(){

    alert("Message for Aadhar Update Sent Successfully!");

  }

  OnDeleteRequest(){

    alert("Message for Aadhar delete Sent Successfully!");

  }

  public onOpenModal(record: myaadhar, mode: string): void {

    const container = document.getElementById('main-container');

    const button = document.createElement('button');

    button.type = 'button';

    button.style.display = 'none';

    button.setAttribute('data-toggle', 'modal');

    if (mode === 'add') {

      button.setAttribute('data-target', '#addCitizenModal');

    }

    if (mode === 'edit') {

      this.editRecord = record;

      button.setAttribute('data-target', '#updateCitizenModal');

    }

    if (mode === 'delete') {

      this.deleteRecord=record;

      button.setAttribute('data-target', '#deleteCitizenModal');

    }

    container.appendChild(button);

    button.click();

  }

  public onAddCitizen(addForm: NgForm): void {

    document.getElementById('add-record-form').click();

    this.appService.add(addForm.value).subscribe(

      (response: myaadhar) => {

       console.log(response);

       this.get();

       addForm.reset();

      },

      (error: HttpErrorResponse) => {

        alert(error.message);

        addForm.reset();

      }

    );

  }

}

App.module.ts

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

import { BrowserModule } from '@angular/platform-browser';

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

import { AppComponent } from './app.component';

import { AppService } from './myaadhar/myaadhar.service';

import { FormsModule, ReactiveFormsModule } from '@angular/forms';

import { MyAadharComponent } from './myaadhar/myaadhar.component';

import { AdminComponent } from './admin/admin.component';

import { HomeComponent } from './home/home.component';

import { UserComponent } from './user/user.component';

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

import { MyAadhar2Component } from './myaadhar2/myaadhar2.component';

const appRoutes: Routes = [

  { path: '', component: HomeComponent },

  { path: 'admin', component: AdminComponent },

  { path: 'user', component: UserComponent },

  { path: 'myaadhar', component: MyAadharComponent },

  { path: 'myaadhar2', component: MyAadhar2Component },

];

@NgModule({

  declarations: [

    AppComponent,

    MyAadharComponent,

    AdminComponent,

    HomeComponent,

    UserComponent,

    MyAadhar2Component

  ],

  imports: [

    BrowserModule,

    HttpClientModule,

    FormsModule,

    ReactiveFormsModule,

    RouterModule.forRoot(appRoutes)

  ],

  providers: [AppService],

  bootstrap: [AppComponent]

})

export class AppModule { }

myaadhar.service.ts

import { HttpClient } from "@angular/common/http";

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

import { Observable } from "rxjs";

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

import { myaadhar } from "./myaadhar";

@Injectable({

    providedIn: 'root'

})

export class AppService{

    private apiServerUrl = environment.apiBaseUrl;

    constructor(private http: HttpClient){ }

   public get():Observable<myaadhar[]>{

    return this.http.get<myaadhar[]>(`${this.apiServerUrl}/aadhar/all`);

   }

    public findmyaadharById(id:number): Observable<myaadhar>{

        return this.http.get<myaadhar>(`${this.apiServerUrl}/aadhar/find/${id}`);

    }

    public add(x:myaadhar): Observable<myaadhar>{

        return this.http.post<myaadhar>(`${this.apiServerUrl}/aadhar/add`,x);

    }

    public update(x:myaadhar): Observable<myaadhar>{

        return this.http.put<myaadhar>(`${this.apiServerUrl}/aadhar/update`,x);

    }

    public delete(id:number): Observable<void>{

        return this.http.delete<void>(`${this.apiServerUrl}/aadhar/delete/${id}`);

    }

}

Test.ts

// This file is required by karma.conf.js and loads recursively all the .spec and framework files

import 'zone.js/testing';

import { getTestBed } from '@angular/core/testing';

import {

  BrowserDynamicTestingModule,

  platformBrowserDynamicTesting

} from '@angular/platform-browser-dynamic/testing';

declare const require: {

  context(path: string, deep?: boolean, filter?: RegExp): {

    <T>(id: string): T;

    keys(): string[];

  };

};

// First, initialize the Angular testing environment.

getTestBed().initTestEnvironment(

  BrowserDynamicTestingModule,

  platformBrowserDynamicTesting(),

);

// Then we find all the tests.

const context = require.context('./', true, /\.spec\.ts$/);

// And load the modules.

context.keys().map(context);

angular.json

{

  "$schema": "./node\_modules/@angular/cli/lib/config/schema.json",

  "cli": {

    "analytics": false

  },

  "version": 1,

  "newProjectRoot": "projects",

  "projects": {

    "FrontendApp": {

      "projectType": "application",

      "schematics": {},

      "root": "",

      "sourceRoot": "src",

      "prefix": "app",

      "architect": {

        "build": {

          "builder": "@angular-devkit/build-angular:browser",

          "options": {

            "outputPath": "dist/frontend-app",

            "index": "src/index.html",

            "main": "src/main.ts",

            "polyfills": "src/polyfills.ts",

            "tsConfig": "tsconfig.app.json",

            "assets": [

              "src/favicon.ico",

              "src/assets"

            ],

            "styles": [

              "src/styles.css",

              "node\_modules/bootstrap/dist/css/bootstrap.min.css"

            ],

            "scripts": []

          },

          "configurations": {

            "production": {

              "budgets": [

                {

                  "type": "initial",

                  "maximumWarning": "2mb",

                  "maximumError": "5mb"

                },

                {

                  "type": "anyComponentStyle",

                  "maximumWarning": "6kb",

                  "maximumError": "10kb"

                }

              ],

              "fileReplacements": [

                {

                  "replace": "src/environments/environment.ts",

                  "with": "src/environments/environment.prod.ts"

                }

              ],

              "outputHashing": "all"

            },

            "development": {

              "buildOptimizer": false,

              "optimization": false,

              "vendorChunk": true,

              "extractLicenses": false,

              "sourceMap": true,

              "namedChunks": true

            }

          },

          "defaultConfiguration": "production"

        },

        "serve": {

          "builder": "@angular-devkit/build-angular:dev-server",

          "configurations": {

            "production": {

              "browserTarget": "FrontendApp:build:production"

            },

            "development": {

              "browserTarget": "FrontendApp:build:development"

            }

          },

          "defaultConfiguration": "development"

        },

        "extract-i18n": {

          "builder": "@angular-devkit/build-angular:extract-i18n",

          "options": {

            "browserTarget": "FrontendApp:build"

          }

        },

        "test": {

          "builder": "@angular-devkit/build-angular:karma",

          "options": {

            "main": "src/test.ts",

            "polyfills": "src/polyfills.ts",

            "tsConfig": "tsconfig.spec.json",

            "karmaConfig": "karma.conf.js",

            "assets": [

              "src/favicon.ico",

              "src/assets"

            ],

            "styles": [

              "src/styles.css"

            ],

            "scripts": []

          }

        }

      }

    }

  },

  "defaultProject": "FrontendApp"

}

Karma.cong.js

// Karma configuration file, see link for more information

// https://karma-runner.github.io/1.0/config/configuration-file.html

module.exports = function (config) {

  config.set({

    basePath: '',

    frameworks: ['jasmine', '@angular-devkit/build-angular'],

    plugins: [

      require('karma-jasmine'),

      require('karma-chrome-launcher'),

      require('karma-jasmine-html-reporter'),

      require('karma-coverage'),

      require('@angular-devkit/build-angular/plugins/karma')

    ],

    client: {

      jasmine: {

        // you can add configuration options for Jasmine here

        // the possible options are listed at https://jasmine.github.io/api/edge/Configuration.html

        // for example, you can disable the random execution with `random: false`

        // or set a specific seed with `seed: 4321`

      },

      clearContext: false // leave Jasmine Spec Runner output visible in browser

    },

    jasmineHtmlReporter: {

      suppressAll: true // removes the duplicated traces

    },

    coverageReporter: {

      dir: require('path').join(\_\_dirname, './coverage/frontend-app'),

      subdir: '.',

      reporters: [

        { type: 'html' },

        { type: 'text-summary' }

      ]

    },

    reporters: ['progress', 'kjhtml'],

    port: 9876,

    colors: true,

    logLevel: config.LOG\_INFO,

    autoWatch: true,

    browsers: ['Chrome'],

    singleRun: false,

    restartOnFileChange: true

  });

};

Package.json

{

  "name": "frontend-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": "~13.3.0",

    "@angular/common": "~13.3.0",

    "@angular/compiler": "~13.3.0",

    "@angular/core": "~13.3.0",

    "@angular/forms": "~13.3.0",

    "@angular/platform-browser": "~13.3.0",

    "@angular/platform-browser-dynamic": "~13.3.0",

    "@angular/router": "~13.3.0",

    "bootstrap": "^4.6.2",

    "rxjs": "^7.5.7",

    "tslib": "^2.3.0",

    "zone.js": "~0.11.4"

  },

  "devDependencies": {

    "@angular-devkit/build-angular": "~13.3.9",

    "@angular/cli": "~13.3.9",

    "@angular/compiler-cli": "~13.3.0",

    "@types/jasmine": "~3.10.0",

    "@types/node": "^12.11.1",

    "jasmine-core": "~4.0.0",

    "karma": "~6.3.0",

    "karma-chrome-launcher": "~3.1.0",

    "karma-coverage": "~2.1.0",

    "karma-jasmine": "~4.0.0",

    "karma-jasmine-html-reporter": "~1.7.0",

    "typescript": "~4.6.2"

  }

}

Tsconfig.json

{

  "compileOnSave": false,

  "compilerOptions": {

    "baseUrl": "./",

    "outDir": "./dist/out-tsc",

    "sourceMap": true,

    "declaration": false,

    "downlevelIteration": true,

    "experimentalDecorators": true,

    "moduleResolution": "node",

    "importHelpers": true,

    "target": "es2017",

    "module": "es2020",

    "lib": [

      "es2020",

      "dom"

    ]

  },

  "angularCompilerOptions": {

    "enableI18nLegacyMessageIdFormat": false

  }

}