**BACKEND SOURCE CODE**

AadharBackendApplication.java

package com.example.capstone;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class AadharBackendApplication {

public static void main(String[] args) {

SpringApplication.run(AadharBackendApplication.class, args);

}

AadharApiException.java

package com.example.capstone.AadharRestApi.Controller;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.ControllerAdvice;

import org.springframework.web.bind.annotation.ExceptionHandler;

import com.examples.capstone.AadharRestApi.Exception.AdminNotFoundException;

import com.examples.capstone.AadharRestApi.Exception.CitizenNotFoundException;

import com.examples.capstone.AadharRestApi.Exception.UserNotFoundException;

@ControllerAdvice

public class AadharApiException {

@ExceptionHandler(value=AdminNotFoundException.class)

public ResponseEntity<Object> handleException(AdminNotFoundException ex) {

return new ResponseEntity<Object>(ex.getMessage(), HttpStatus.NOT\_FOUND);

}

@ExceptionHandler(value=UserNotFoundException.class)

public ResponseEntity<Object> handleException(UserNotFoundException ex) {

return new ResponseEntity<Object>(ex.getMessage(), HttpStatus.NOT\_FOUND);

}

@ExceptionHandler(value=CitizenNotFoundException.class)

public ResponseEntity<Object> handleException(CitizenNotFoundException ex) {

return new ResponseEntity<Object>(ex.getMessage(), HttpStatus.NOT\_FOUND);

}

}

}

RestApiController.java

package com.example.capstone.AadharRestApi.Controller;

import java.util.Optional;

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

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.CrossOrigin;

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.RestController;

import com.example.capstone.AadharRestApi.Entity.Admin;

import com.example.capstone.AadharRestApi.Entity.Citizens;

import com.example.capstone.AadharRestApi.Entity.LoginRequest;

import com.example.capstone.AadharRestApi.Entity.User;

import com.example.capstone.AadharRestApi.Repository.AdminRepo;

import com.example.capstone.AadharRestApi.Repository.CitizensRepo;

import com.example.capstone.AadharRestApi.Repository.UserRepo;

import com.examples.capstone.AadharRestApi.Exception.AdminNotFoundException;

import com.examples.capstone.AadharRestApi.Exception.CitizenNotFoundException;

import com.examples.capstone.AadharRestApi.Exception.UserNotFoundException;

import com.example.capstone.AadharRestApi.Entity.SuccessResponse;

@RestController

public class RestApiController {

@Autowired

private AdminRepo adminrepo;

@Autowired

private UserRepo userrepo;

@Autowired

private CitizensRepo citiRepo;

// Login endpoint

@CrossOrigin(origins = "http://localhost:4200")

@PostMapping("/login")

public ResponseEntity<String> login(@RequestBody LoginRequest loginRequest) {

// Retrieve the username and password from the request body

String username = loginRequest.getUsername();

String password = loginRequest.getPassword();

// Here, you can implement your authentication logic, such as checking if the user exists in the database

// and verifying the password. For simplicity, we'll use a hardcoded username and password.

if ("admin".equals(username) && "Pass@word123".equals(password)) {

// If authentication is successful, return a success response

return ResponseEntity.ok().body("Login Successful");

} else {

// If authentication fails, return an error response

return ResponseEntity.badRequest().body("Invalid credentials");

}}

// Admin

@CrossOrigin(origins = "http://localhost:4200")

@GetMapping("/admin")

public Iterable<Admin> getAdmin() {

return adminrepo.findAll();

}

@CrossOrigin(origins = "http://localhost:4200")

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

public Admin getAdmin(@PathVariable("id") Integer id) {

Optional<Admin> optAdmin = adminrepo.findById(id);

if (optAdmin.isEmpty()) {

throw new AdminNotFoundException(id);

}

return optAdmin.get();

}

@CrossOrigin(origins = "http://localhost:4200")

@PostMapping("/admin")

public Admin create(@RequestBody Admin admin) {

return adminrepo.save(admin);

}

@CrossOrigin(origins = "http://localhost:4200")

@PutMapping("/admin")

public Admin update(@RequestBody Admin admin) {

return adminrepo.save(admin);

}

@CrossOrigin(origins = "http://localhost:4200")

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

public void deleteadmin(@PathVariable("id") Integer id) {

adminrepo.deleteById(id);

}

// User

@CrossOrigin(origins = "http://localhost:4200")

@GetMapping("/user")

public Iterable<User> getUser() {

return userrepo.findAll();

}

@CrossOrigin(origins = "http://localhost:4200")

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

public User getUser(@PathVariable("id") Integer id) {

Optional<User> optUser = userrepo.findById(id);

if (optUser.isEmpty()) {

throw new UserNotFoundException(id);

}

return optUser.get();

}

@CrossOrigin(origins = "http://localhost:4200")

@PostMapping("/user")

public User create(@RequestBody User user) {

return userrepo.save(user);

}

@CrossOrigin(origins = "http://localhost:4200")

@PutMapping("/user")

public User update(@RequestBody User user) {

return userrepo.save(user);

}

@CrossOrigin(origins = "http://localhost:4200")

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

public void deleteuser(@PathVariable("id") Integer id) {

userrepo.deleteById(id);

}

// Citizens

@CrossOrigin(origins = "http://localhost:4200")

@GetMapping("/citizens")

public Iterable<Citizens> getCitizens() {

return citiRepo.findAll();

}

@CrossOrigin(origins = "http://localhost:4200")

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

public Citizens getCiti(@PathVariable("id") Integer id) {

Optional<Citizens> optCiti = citiRepo.findById(id);

if (optCiti.isEmpty()) {

throw new CitizenNotFoundException(id);

}

return optCiti.get();

}

// Registration endpoint with check for existing citizen

@CrossOrigin(origins = "http://localhost:4200")

@PostMapping("/citizens")

public ResponseEntity<?> create(@RequestBody Citizens citi) {

// Check if a citizen with the same email and name already exists

boolean citizenExists = citiRepo.existsByEmailidAndName(citi.getEmailid(), citi.getName());

if (citizenExists) {

return ResponseEntity.badRequest().body("User with the same email and name already exists!");

}

// If no existing citizen, proceed with registration

return ResponseEntity.ok(citiRepo.save(citi));

}

// Update Adhar number endpoint

@CrossOrigin(origins = "http://localhost:4200")

@PostMapping("/citizens/{id}/adhar")

public ResponseEntity<?> addAdharNumber(@PathVariable("id") Integer id, @RequestBody String adharNumber) {

Optional<Citizens> optCiti = citiRepo.findById(id);

if (optCiti.isEmpty()) {

return ResponseEntity.notFound().build();

}

Citizens citi = optCiti.get();

citi.setAdharNumber(adharNumber);

citiRepo.save(citi);

SuccessResponse response = new SuccessResponse("Adhar number added successfully!");

return ResponseEntity.ok(response);

}

@CrossOrigin(origins = "http://localhost:4200")

@PutMapping("/citizens")

public Citizens update(@RequestBody Citizens citi) {

return citiRepo.save(citi);

}

@CrossOrigin(origins = "http://localhost:4200")

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

public void deleteciti(@PathVariable("id") Integer id) {

citiRepo.deleteById(id);

}

}

Admin.java

package com.example.capstone.AadharRestApi.Entity;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Entity

public class Admin {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String admin;

private String password;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getAdmin() {

return admin;

}

public void setAdmin(String admin) {

this.admin = admin;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

@Override

public String toString() {

return "Admin [id=" + id + ", admin=" + admin + ", password=" + password + "]";

}}

Citizens.java

package com.example.capstone.AadharRestApi.Entity;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Entity

public class Citizens {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "citizenId", nullable = false)

private int citizenId;

private String name;

private String dob;

private String emailid;

private String mobileno;

private String gender;

private String address;

@Column(name = "adhar\_number")

private String adharNumber;

public int getCitizenId() {

return citizenId;

}

public void setCitizenId(int citizenId) {

this.citizenId = citizenId;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getDob() {

return dob;

}

public void setDob(String dob) {

this.dob = dob;

}

public String getEmailid() {

return emailid;

}

public void setEmailid(String emailid) {

this.emailid = emailid;

}

public String getMobileno() {

return mobileno;

}

public void setMobileno(String mobileno) {

this.mobileno = mobileno;

}

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 getAdharNumber() {

return adharNumber;

}

public void setAdharNumber(String adharNumber) {

this.adharNumber = adharNumber;

}

@Override

public String toString() {

return "Citizens [citizenId=" + citizenId + ", name=" + name + ", dob=" + dob + ", emailid=" + emailid

+ ",mobileno=" + mobileno + ", gender=" + gender + ", address=" + address

+ ", adharNumber=" + adharNumber + "]";

}}

LoginRequest

package com.example.capstone.AadharRestApi.Entity;

public class LoginRequest {

private String username;

private String password;

// Add constructors, getters, and setters here

// Constructor

public LoginRequest() {

}

// Getters and Setters

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

}

SuccessResponse.java

package com.example.capstone.AadharRestApi.Entity;

public class SuccessResponse {

private String message;

public SuccessResponse(String message) {

this.message = message;

}

public String getMessage() {

return message;

}

public void setMessage(String message) {

this.message = message;

}

}

User.java

package com.example.capstone.AadharRestApi.Entity;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

@Entity

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String emailId;

private String password;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getEmail() {

return emailId;

}

public void setEmail(String email) {

this.emailId = email;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

@Override

public String toString() {

return "User [id=" + id + ", email=" + emailId + ", password=" + password + "]";

}}

AdminRepo.java

package com.example.capstone.AadharRestApi.Repository;

import org.springframework.data.repository.CrudRepository;

import com.example.capstone.AadharRestApi.Entity.Admin;

public interface AdminRepo extends CrudRepository<Admin, Integer>{

}

CitizensRepo.java

package com.example.capstone.AadharRestApi.Repository;

import org.springframework.data.repository.CrudRepository;

import com.example.capstone.AadharRestApi.Entity.Citizens;

public interface CitizensRepo extends CrudRepository<Citizens, Integer> {

// Custom method to check if a citizen with the given email and name exists

boolean existsByEmailidAndName(String emailid, String name);

}

UserRepo.java

package com.example.capstone.AadharRestApi.Repository;

import org.springframework.data.repository.CrudRepository;

import com.example.capstone.AadharRestApi.Entity.User;

public interface UserRepo extends CrudRepository<User, Integer>{

}

AdminNotFoundException.java

package com.examples.capstone.AadharRestApi.Exception;

public class AdminNotFoundException extends RuntimeException{

public AdminNotFoundException(int id) {

super("Admin with id " + id + " is not Found.Pls Give another id!!");

}

}

CitizenNotFoundException.java

package com.examples.capstone.AadharRestApi.Exception;

public class CitizenNotFoundException extends RuntimeException{

public CitizenNotFoundException(int id) {

super("Citizen with id " + id + " is not Found.Pls Give another id!!");

}

}

UserNotFoundException.java

package com.examples.capstone.AadharRestApi.Exception;

public class UserNotFoundException extends RuntimeException{

public UserNotFoundException(int id) {

super("User with id " + id + " is not Found.Pls Give another id!!");

}

}

Application.properties

server.port=11000

spring.datasource.url=jdbc:mysql://127.0.0.1:3306/capstone

spring.datasource.username=root

spring.datasource.password=Root

spring.jpa.properties.hibernate.show\_sql=true

spring.jpa.properties.hibernate.format\_sql=true

spring.jpa.hibernate.ddl-auto=update

logging.level.org.hibernate=INFO

logging.level.org.hibernate.SQL=INFO

logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE

logging.level.org.springframework=INFO

logging.level.org.apache=INFO

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>2.7.17</version>

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

</parent>

<groupId>com.example</groupId>

<artifactId>AadharBackend</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>AadharBackend</name>

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

<properties>

<java.version>1.8</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.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

<groupId>mysql</groupId>

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

<version>8.0.27</version> <!-- Use the appropriate version -->

</dependency>

<dependency>

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

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

</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>

Sql code

create database capstone;

use capstone;

CREATE TABLE admin (

id INT PRIMARY KEY AUTO\_INCREMENT,

admin VARCHAR(255) NOT NULL,

password VARCHAR(255) NOT NULL

);

CREATE TABLE citizens (

citizenId INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(255) NOT NULL,

dob DATE NOT NULL,

emailid VARCHAR(255) NOT NULL,

mobileno VARCHAR(20) NOT NULL,

gender VARCHAR(10) NOT NULL,

address VARCHAR(500) NOT NULL

);

CREATE TABLE user (

id INT PRIMARY KEY AUTO\_INCREMENT,

emailId VARCHAR(255) NOT NULL,

password VARCHAR(255) NOT NULL

);

ALTER TABLE User MODIFY COLUMN emailId VARCHAR(255) NULL;

ALTER TABLE citizens RENAME COLUMN citizenId TO citizen\_id;

ALTER TABLE citizens DROP COLUMN citizen\_id;

ALTER TABLE citizens CHANGE citizenId citizen\_id INT AUTO\_INCREMENT;

ALTER TABLE citizens

ADD adharNumber VARCHAR(50)

**FRONTEND CODE**

Admin.component.html

<div class="image">

  <br />

  <br />

  <div class="container d-flex justify-content-center">

    <form

      [formGroup]="loginForm"

      (ngSubmit)="onSubmit()"

      class="w-50 p-4"

      style="color: rgb(3, 11, 51)"

    >

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

      <div class="row">

        <div class="form-group">

          <label for="username">Admin name</label>

          <input

            type="text"

            id="admin"

            class="form-control"

            formControlName="admin"

          />

          <span

            style="color: red"

            class="small"

            \*ngIf="

              !loginForm.get('admin').valid && loginForm.get('admin').touched

            "

            >Please enter a valid admin name</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

            style="color: red"

            class="small"

            \*ngIf="

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

              loginForm.get('password').touched

            "

            >Please enter a valid password</span

          >

          <br />

          <span

            style="color: red"

            class="small"

            \*ngIf="loginForm.get('password').hasError('passwordStrength')"

          >

            {{ loginForm.get("password").errors["passwordStrength"] }}

          </span>

        </div>

      </div>

      <br />

      <button type="submit" class="btn btn-outline-danger">Login</button>

    </form>

  </div>

</div>

Admin.component.ts

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

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

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

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

import { PasswordStrengthValidator } from "./password-strength.validators"

@Component({

  selector: 'app-admin',

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

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

})

export class AdminComponent implements OnInit {

  public loginForm!: FormGroup;

  url="http://localhost:11000/admin";

  submitted=false;

  constructor(private formBuilder: FormBuilder,private http: HttpClient , private router:Router) { }

  ngOnInit(): void {

    this.loginForm=this.formBuilder.group({

      admin : ['',Validators.required],

      password : ['',Validators.compose([

        Validators.required, PasswordStrengthValidator])]

  })

  }

  onSubmit() {

    this.submitted = true;

    this.login();

  }

  login() {

    this.http.post<any>(this.url, {

      admin: this.loginForm.value.admin,

      password: this.loginForm.value.password

    }).subscribe(res => {

      alert("Login Success,Click Ok to continue!!");

      this.loginForm.reset();

      this.router.navigate(['adminhome']);

    }, err => {

      alert("Try Again!!");

      this.loginForm.reset();

    });

  }

  }

Password-strength.validators

import { AbstractControl, ValidationErrors } from "@angular/forms"

export const PasswordStrengthValidator = function (control: AbstractControl): ValidationErrors | null {

  let value: string = control.value || '';

  if (!value) {

    return null

  }

  let upperCaseCharacters = /[A-Z]+/g

  if (upperCaseCharacters.test(value) === false) {

    return { passwordStrength: `Upper case required` };

  }

  let lowerCaseCharacters = /[a-z]+/g

  if (lowerCaseCharacters.test(value) === false) {

    return { passwordStrength: `lower case required` };

  }

  let numberCharacters = /[0-9]+/g

  if (numberCharacters.test(value) === false) {

    return { passwordStrength: `number required` };

  }

  let specialCharacters = /[!@#$%^&\*()\_+\-=\[\]{};':"\\|,.<>\/?]+/

  if (specialCharacters.test(value) === false) {

    return { passwordStrength: `Special char required` };

  }

  return null;

}

Adminhome.component.html

<div class="image">

  <div>

    <h1 class="d-flex justify-content-center" style="color: crimson">

      AADHAR APPLICATION

    </h1>

  </div>

  <div class="container">

    <button class="btn btn-outline-danger mt-5" (click)="onFetchUsers()">

      View All

    </button>

    <div class="row m-3">

      <div class="col">

        <ul class="list-group" \*ngIf="fetchedUsers.length >= 1">

          <li class="list-group-item" \*ngFor="let user of fetchedUsers">

            <p><b>Aadhar id:</b>&nbsp;{{ user.citizenId }}</p>

            <p><b>Name&nbsp;:</b>&nbsp;{{ user.name }}</p>

            <p><b>Date Of Birth:</b>&nbsp;{{ user.dob }}</p>

            <p><b>Email:</b>&nbsp;{{ user.emailid }}</p>

            <p><b>Address:</b>&nbsp;{{ user.address }}</p>

            <p><b>Mobile No:</b>&nbsp;{{ user.mobileno }}</p>

            <p><b>Gender:</b>&nbsp;{{ user.gender }}</p>

     <!-- Add an input field to enter the Adhar number -->

     <div class="form-group">

      <label for="adharNumber">Adhar Number</label>

      <input

        type="text"

        id="adharNumber"

        class="form-control"

        [(ngModel)]="adharNumber"

        name="adharNumber"

      />

    </div>

    <!-- Add "Issue Adhar" button for each user -->

    <button

      class="btn btn-sm btn-primary me-2"

      (click)="onSubmitIssueAdhar(user.citizenId)"

    >

      Issue Adhar

    </button>

            <button

              class="btn btn-sm btn-danger"

              (click)="onDeleteUser(user.citizenId)"

            >

              Delete

            </button>

          </li>

        </ul>

      </div>

    </div>

  </div>

</div>

Admin.component.ts

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

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

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

import { map } from 'rxjs/operators';

import { ApiService } from '../service/api.service';

import { UserData } from './userdata.model';

@Component({

  selector: 'app-adminhome',

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

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

})

export class AdminhomeComponent implements OnInit {

  fetchedUsers: UserData[] = [];

  backendurl = 'http://localhost:11000/citizens';

  updateurl = 'http:localhost:11000/citizens/adhar';

  adharNumber: string; // New property to store the Adhar number

  constructor(

    private http: HttpClient,

    private formBuilder: FormBuilder,

    private api: ApiService

  ) {}

  ngOnInit(): void {

  }

  onFetchUsers() {

    this.fetchUsers();

  }

  fetchUsers() {

    this.http

      .get(this.backendurl)

      .pipe(

        map((responseData) => {

          const userArray: UserData[] = [];

          for (const key in responseData) {

            var x = { ...responseData[key] };

            userArray.push(x);

          }

          return userArray;

        })

      )

      .subscribe((users) => {

        this.fetchedUsers = users;

      });

  }

  onDeleteUser(id: number) {

    this.http.delete(this.backendurl + '/' + id).subscribe((response) => {

      console.log('User deleted: ' + response);

      alert('Closing adhar due to death!');

      location.reload();

    });

  }

  onSubmitIssueAdhar(citizenId: number) {

    console.log('Submitting Adhar number for citizenId:', citizenId);

    console.log('Adhar number:', this.adharNumber);

    // You can access the Adhar number from this.adharNumber

    const adharNumber = this.adharNumber;

    // Make a POST request to your backend API to add the Adhar number

    this.http.post(`${this.backendurl}/${citizenId}/adhar`, adharNumber).subscribe(

        (response: any) => {

            console.log('Response from backend:', response);

            console.log('Adhar number added:', response.message); // Access the success message

            alert('Adhar number added successfully!');

            location.reload();

        },

        (error) => {

            console.error('Error adding Adhar number:', error);

            alert('Failed to add Adhar number. Please try again.');

        }

    );

}

}

Home.component.html

<div class="image">

  <div class="container">

    <h1 class="d-flex justify-content-center" style="color: crimson">

      AADHAR APPLICATION

    </h1>

    <a

      type="button"

      class="d-inline btn btn-outline-danger"

      routerLink="/userlogin"

    >

      User Login</a

    >&nbsp;&nbsp;New User?<a routerLink="/register">Register Here</a>

    <br />

    <br />

    <a

      type="button"

      class="d-inline btn btn-outline-danger"

      routerLink="/adminlogin"

    >

      Admin Login

    </a>

  </div>

</div>

Home.component.ts

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

@Component({

  selector: 'app-home',

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

  styleUrls: ['./home.component.css']

})

export class HomeComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {

  }

}

Issue.adhar.component.html

<div class="container mt-5">

    <h3>Issue Aadhar</h3>

    <form [formGroup]="issueAdharForm" (submit)="onSubmitAdharForm()">

      <div class="mb-3">

        <label for="adharNumber" class="form-label">Aadhar Number</label>

        <input

          type="text"

          class="form-control"

          id="adharNumber"

          formControlName="adharNumber"

        />

      </div>

      <button type="submit" class="btn btn-primary">Submit</button>

    </form>

  </div>

Issue-adhar.component.ts

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

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

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

@Component({

  selector: 'app-issue-adhar',

  templateUrl: './issue-adhar.component.html',

  styleUrls: ['./issue-adhar.component.css'],

})

export class IssueAdharComponent implements OnInit {

  issueAdharForm: FormGroup;

  constructor(private formBuilder: FormBuilder, private router: Router) {

    this.issueAdharForm = this.formBuilder.group({

      adharNumber: [''],

    });

  }

  ngOnInit(): void {}

  onSubmitAdharForm() {

    // Get the adhar number from the form

    const adharNumber = this.issueAdharForm.get('adharNumber')?.value;

    // Do whatever you need to do with the adhar number, e.g., store it in the backend

    // After processing, navigate back to the admin page

    this.router.navigate(['/admin']);

  }

}

Register.component.html

<div class="image">

  <br />

  <h3 style="color: rgba(42, 2, 71, 0.84)" class="text-center">

    Registration Page

  </h3>

  <div class="container mt-3 d-flex justify-content-center">

    <form #f="ngForm" class="w-50 p-4" style="color: rgb(120, 55, 5)">

      <div class="row mb-2">

        <div class="form-group">

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

          <input

            type="text"

            id="name"

            class="form-control"

            name="name"

            required

            ngModel

            #name="ngModel"

          />

        </div>

      </div>

      <div class="row mb-2">

        <div class="form-group">

          <label for="dob">Date Of Birth</label>

          <input

            type="text"

            id="dob"

            class="form-control"

            name="dob"

            ngModel

            required

            dob

            #dob="ngModel"

          />

          <span

            id="err-dob"

            style="color: red"

            class="small"

            \*ngIf="!dob.valid && dob.touched"

            >Please enter a valid dob</span

          >

        </div>

      </div>

      <div class="row mb-2">

        <div class="form-group">

          <label for="emailid">Email</label>

          <input

            type="text"

            id="emailid"

            class="form-control"

            name="emailid"

            ngModel

            required

            emailid

            #emailid="ngModel"

          />

          <span

            id="err-email"

            style="color: red"

            class="small"

            \*ngIf="!emailid.valid && emailid.touched"

            >Please enter a valid email</span

          >

        </div>

      </div>

      <div class="row mb-2">

        <div class="form-group">

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

          <textarea

            type="text"

            id="address"

            class="form-control"

            name="address"

            ngModel

            required

            address

            #address="ngModel"

          ></textarea>

          <span

            id="err-address"

            style="color: red"

            class="small"

            \*ngIf="!address.valid && address.touched"

            >Please enter a valid address</span

          >

        </div>

      </div>

      <div class="row mb-2">

        <div class="form-group">

          <label for="Phoneno">Mobile No:</label>

          <input

            type="text"

            id="mobileno"

            class="form-control"

            name="mobileno"

            ngModel

            required

            mobileno

            #mobileno="ngModel"

          />

          <span

            id="err-mobile"

            style="color: red"

            class="small"

            \*ngIf="!mobileno.valid && mobileno.touched"

            >Please enter a valid Mobile Number</span

          >

        </div>

      </div>

      <div class="row mb-2">

        <div class="form-group">

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

          <input

            type="text"

            id="gender"

            class="form-control"

            name="gender"

            ngModel

            required

            gender

            #gender="ngModel"

          />

          <span

            id="err-gender"

            style="color: red"

            class="small"

            \*ngIf="!gender.valid && gender.touched"

            >Please enter a valid Gender</span

          >

        </div>

      </div>

      <button

        class="btn btn-outline-danger m-2"

        [disabled]="!f.valid"

        (click)="onAddCitizen(f.value, f)"

        id="register"

      >

        Register

      </button>

    </form>

  </div>

</div>

Register.component.ts

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

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

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

@Component({

  selector: 'app-register',

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

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

})

export class RegisterComponent implements OnInit {

  url='http://localhost:11000/citizens';

  constructor(private http:HttpClient) { }

  ngOnInit(): void {

  }

  onAddCitizen(citizenData: { name: string, dob: string, emailid: string, gender: string, mobileno: string, address: string }, form: NgForm) {

    // Check if the user with the same email and name already exists

    this.http.get(this.url, { params: { emailid: citizenData.emailid, name: citizenData.name } })

      .subscribe((responseData: any[]) => {

        if (responseData.length > 0) {

          alert("User with the same email and name already exists!");

        } else {

          // If user doesn't exist, proceed with registration

          this.http.post(this.url, citizenData).subscribe(() => {

            console.log("Registered Successfully!!");

            alert("Registered Successfully!!");

            form.reset();

          }, err => {

            alert("Something Happened!!");

          });

        }

      }, err => {

        alert("Something Happened!!");

      });

  }

}

Api.service.ts

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

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

import { map } from 'rxjs/operators';

@Injectable({

  providedIn: 'root'

})

export class ApiService {

  constructor(private http:HttpClient) { }

  UpdateUser(data:any,id:number){

    return this.http.put<any>('http://localhost:11000/citizens',data)

    .pipe(map((res:any)=>{

      return res;

    }))

  }

}

User.component.html

<div class="image">

  <br />

  <br />

  <div class="container d-flex justify-content-center">

    <form

      [formGroup]="loginForm"

      (ngSubmit)="onSubmit()"

      class="w-50 p-4"

      style="color: rgb(3, 11, 51)"

    >

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

      <div class="row">

        <div class="form-group">

          <label for="citizenId">Id</label>

          <br />

          <input

            type="number"

            id="citizen\_id"

            class="form-control"

            formControlName="citizenId"

          />

        </div>

      </div>

      <div class="row">

        <div class="form-group">

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

          <br />

          <span class="small" style="color: darkgreen"

            >Your mobile number is your password</span

          >

          <input

            type="password"

            id="password"

            class="form-control"

            formControlName="mobileno"

          />

          <span

            style="color: red"

            class="small"

            \*ngIf="

              !loginForm.get('mobileno').valid &&

              loginForm.get('mobileno').touched

            "

            >Please enter a valid password</span

          >

        </div>

      </div>

      <br />

      <button type="submit" class="btn btn-outline-danger">Login</button>

    </form>

  </div>

</div>

User.component.ts

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

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

import { FormBuilder, 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 {

  public loginForm!: FormGroup;

  url="http://localhost:11000/citizens";

  submitted=false;

  constructor(private formBuilder: FormBuilder,private http: HttpClient , private router:Router) { }

  ngOnInit(): void {

    this.loginForm=this.formBuilder.group({

      citizenId : ['',Validators.required],

     mobileno : ['',Validators.required]

  })

  }

  onSubmit() {

    this.submitted = true;

    this.login();

  }

  login(){

    this.http.get<any>(this.url).subscribe(res=>{

      const log = res.find((a:any)=>{

        return a.citizenId === this.loginForm.value.citizenId && a.mobileno === this.loginForm.value.mobileno

      });

      if(log){

        alert("Login Success,Click Ok to continue!!");

        this.loginForm.reset();

        this.router.navigate(['userhome']);

      }else{

        alert("Try Again!!");

      }

    },err=>{

        alert("Something Went Wrong!!");

        this.loginForm.reset();

    })

  }

}

Userhome.component.html

<div class="image">

  <div class="container">

    <div class="h1 d-flex justify-content-center">AADHAR APPLICATION</div>

    <div class="col card d-flex justify-content-center" style="width: 25rem">

      <div class="card-body">

        <h5 class="card-title" style="color: red">Your Details</h5>

        <div class="card-text" \*ngFor="let user of fetchedUsers">

          <p><b>Aadhar id:</b>&nbsp;{{ user.citizenId }}</p>

          <p><b>Name&nbsp;:</b>&nbsp;{{ user.name }}</p>

          <p><b>Date Of Birth:</b>&nbsp;{{ user.dob }}</p>

          <p><b>Email:</b>&nbsp;{{ user.emailid }}</p>

          <p><b>Address:</b>&nbsp;{{ user.address }}</p>

          <p><b>Mobile No:</b>&nbsp;{{ user.mobileno }}</p>

          <p><b>Gender:</b>&nbsp;{{ user.gender }}</p>

          <p><b>Adhar Number:</b>&nbsp;{{ user.adharNumber }}</p> <!-- Display the Adhar number here -->

          <button

            type="button"

            class="btn btn-outline-primary"

            data-bs-toggle="modal"

            data-bs-target="#exampleModal"

            (click)="onEdit(user)"

          >

            Update</button

          >&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

          <a

            type="button"

            class="btn btn-outline-primary align-right"

            routerLink="/home"

            >Logout</a

          >

        </div>

      </div>

    </div>

  </div>

  <div class="footer bg-secondary d-flex align-items-end"></div>

</div>

<!-- Modal -->

<div

  class="modal fade"

  id="exampleModal"

  tabindex="-1"

  aria-labelledby="exampleModalLabel"

  aria-hidden="true"

>

  <div class="modal-dialog">

    <div class="modal-content">

      <div class="modal-header">

        <h5 class="modal-title" id="exampleModalLabel">Update User</h5>

        <button

          type="button"

          class="btn-close"

          data-bs-dismiss="modal"

          aria-label="Close"

        ></button>

      </div>

      <div class="modal-body">

        <form [formGroup]="loginForm" style="color: rgb(120, 55, 5)">

          <div class="row mb-2">

            <div class="form-group">

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

              <input

                type="text"

                id="name"

                class="form-control"

                name="name"

                required

                formControlName="name"

              />

            </div>

          </div>

          <div class="row mb-2">

            <div class="form-group">

              <label for="dob">Date Of Birth</label>

              <input

                type="text"

                id="dob"

                class="form-control"

                name="dob"

                formControlName="dob"

                required

              />

            </div>

          </div>

          <div class="row mb-2">

            <div class="form-group">

              <label for="emailid">Email</label>

              <input

                type="text"

                id="emailid"

                class="form-control"

                name="emailid"

                formControlName="emailid"

                required

              />

            </div>

          </div>

          <div class="row mb-2">

            <div class="form-group">

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

              <textarea

                type="text"

                id="address"

                class="form-control"

                name="address"

                required

                formControlName="address"

              ></textarea>

            </div>

          </div>

          <div class="row mb-2">

            <div class="form-group">

              <label for="Phoneno">Mobile No:</label>

              <input

                type="text"

                id="mobileno"

                class="form-control"

                name="mobileno"

                required

                formControlName="mobileno"

              />

            </div>

          </div>

          <div class="row mb-2">

            <div class="form-group">

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

              <input

                type="text"

                id="gender"

                class="form-control"

                name="gender"

                required

                formControlName="gender"

              />

            </div>

          </div>

        </form>

        <div class="modal-footer">

          <button

            type="button"

            class="btn btn-secondary"

            data-bs-dismiss="modal"

          >

            Close

          </button>

          <button type="button" (click)="updateUser()" class="btn btn-primary">

            Save changes

          </button>

        </div>

      </div>

    </div>

  </div>

</div>

Userhome.component.ts

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

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

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

import { map } from 'rxjs/operators';

import { ApiService } from '../service/api.service';

import { UserModel } from './user.model';

import { UserData } from './userdata.model';

@Component({

  selector: 'app-userhome',

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

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

})

export class UserhomeComponent implements OnInit {

  userModelObj:UserModel=new UserModel();

  public loginForm!: FormGroup;

  fetchedUsers: UserData[] = [];

  backendurl = 'http://localhost:11000/citizens';

  constructor(private http:HttpClient,private formBuilder:FormBuilder,private api:ApiService) {}

  ngOnInit(): void {

    this.loginForm=this.formBuilder.group({

      name : [''],

      dob : [''],

      emailid : [''],

      mobileno : [''],

      address : [''],

      gender : [''],

  })

    this.fetchUsers();

  }

  fetchUsers() {

    this.http

    .get<UserData[]>(this.backendurl)

      .pipe(

        map((responseData) => {

          const userArray: UserData[] = [];

          for (const key in responseData) {

            var x = { ...responseData[key] };

            userArray.push(x);

          }

          return userArray;

        })

      )

      .subscribe((users) => {

        this.fetchedUsers = users;

      });

  }

  onEdit(user:any){

   this.userModelObj.citizenId=user.citizenId;

   this.loginForm.controls['name'].setValue(user.name);

   this.loginForm.controls['dob'].setValue(user.dob);

   this.loginForm.controls['emailid'].setValue(user.emailid);

   this.loginForm.controls['mobileno'].setValue(user.mobileno);

   this.loginForm.controls['address'].setValue(user.address);

   this.loginForm.controls['gender'].setValue(user.gender);

  }

  updateUser(){

    this.userModelObj.name=this.loginForm.value.name;

    this.userModelObj.emailid=this.loginForm.value.emailid;

    this.userModelObj.gender=this.loginForm.value.gender;

    this.userModelObj.address=this.loginForm.value.address;

    this.userModelObj.mobileno=this.loginForm.value.mobileno;

    this.userModelObj.dob=this.loginForm.value.dob;

    this.api.UpdateUser(this.userModelObj,this.userModelObj.citizenId)

    .subscribe(res=>{

      alert("Updated SuccessFully!!");

      this.loginForm.reset();

      this.fetchUsers();

    },err=>{

      alert("Something Went Wrong:/")

    })

  }

}

Userdata.model.ts

export interface UserData{

    citizenId:number,

    name:string,

    dob:string,

    mobileno:string,

    address:string,

    emailid:string,

    gender:string,

    adharNumber: string;

}

App.component.html

<router-outlet></router-outlet>

App.component.ts

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

@Component({

  selector: 'app-root',

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

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  title = 'AadharApp';

}

App.module.ts

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

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

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

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

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

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

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

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

import { RegisterComponent } from './register/register.component';

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

import { AdminhomeComponent } from './adminhome/adminhome.component';

import { UserhomeComponent } from './userhome/userhome.component';

import { IssueAdharComponent } from './issue-adhar/issue-adhar.component';

const appRoutes:Routes=[

  { path: '', redirectTo: 'home' , pathMatch:'full' },

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

  { path: 'home', component: HomeComponent },

  { path : 'register' , component: RegisterComponent},

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

  { path : 'adminhome' , component: AdminhomeComponent},

  { path : 'userhome' , component: UserhomeComponent}

]

@NgModule({

  declarations: [

    AppComponent,

    AdminComponent,

    HomeComponent,

    RegisterComponent,

    UserComponent,

    IssueAdharComponent,

    AdminhomeComponent,

    UserhomeComponent,

    IssueAdharComponent

  ],

  imports: [

    BrowserModule,FormsModule,

    RouterModule.forRoot(appRoutes),

    ReactiveFormsModule,

    HttpClientModule

  ],

  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }

Index.html

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8" />

    <title>AadharApp</title>

    <base href="/" />

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

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

    <style>

      body {

        background-image: url("");

        background-repeat: no-repeat;

        background-attachment: fixed;

        background-size: cover;

      }

    </style>

  </head>

  <body>

    <app-root></app-root>

  </body>

</html>

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));