**Capstone Project: Aadhar Application**

***Developed By: Rishu Kumar***

***Source Code:***

package com.maadhar;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.autoconfigure.domain.EntityScan;

import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

@SpringBootApplication

@EntityScan("com.maadhar.pojo")

@EnableJpaRepositories("com.maadhar.repo")

public class MAadharApplication {

public static void main(String[] args) {

SpringApplication.run(MAadharApplication.class, args);

}

}

package com.maadhar;

import org.springframework.boot.builder.SpringApplicationBuilder;

import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;

public class ServletInitializer extends SpringBootServletInitializer {

@Override

protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {

return application.sources(MAadharApplication.class);

}

}

package com.maadhar.controller;

import java.time.LocalDate;

import java.util.ArrayList;

import java.util.List;

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

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

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.maadhar.pojo.AadharCard;

import com.maadhar.pojo.AdminUser;

import com.maadhar.pojo.Citizen;

import com.maadhar.service.AadharCardService;

import com.maadhar.service.AdminService;

import com.maadhar.service.CitizenService;

import com.maadhar.service.WaitingCitizensEmailService;

@RestController

@CrossOrigin(origins = "\*")

@RequestMapping("/admin")

public class AdminController {

@Autowired

AdminService adminService;

@Autowired

CitizenService citizenService;

@Autowired

WaitingCitizensEmailService waitingCitizenService;

@Autowired

AadharCardService aadharService;

//Admin Operations

//REGISTER

@PostMapping("/")

public String registerAdmin(@RequestBody AdminUser adminUser) {

AdminUser testAdminUser=adminService.getAdminByEmail(adminUser.getEmail());

if (testAdminUser!=null) return "0";

if (adminService.addAdmin(adminUser) != null)

return "1";

else

return "0";

}

//GET ADMIN

@GetMapping("/{email}")

public AdminUser getAdminByEmail(@PathVariable String email) {

return adminService.getAdminByEmail(email);

}

//UPDATE ADMIN DETAILS

@PutMapping("/{email}")

public AdminUser updateAdminUser(@PathVariable String email, @RequestBody AdminUser adminUser) {

AdminUser testAdminUser= adminService.getAdminByEmail(email);

if (testAdminUser==null) return null;

Long id=testAdminUser.getId();

adminUser.setId(id);

return(adminService.editAdmin(adminUser));

}

//ADMIN LOGIN

@PutMapping("/login/{email}/{password}")

public String adminLogIn(@PathVariable String email, @PathVariable String password) {

System.out.println(email+" "+password);

AdminUser testAdminUser=adminService.getAdminByEmailAndPassword(email,password);

if (testAdminUser==null)

return "0";

testAdminUser.setIsLoggedIn(true);

adminService.editAdmin(testAdminUser);

return "1";

}

//ADMIN LOGOUT

@PutMapping("logout/{email}")

public String adminLogOut(@PathVariable String email) {

AdminUser testAdminUser=adminService.getAdminByEmail(email);

if (testAdminUser==null)

return "Oops!";

testAdminUser.setIsLoggedIn(false);

adminService.editAdmin(testAdminUser);

return "Logged out!";

}

// CITIZEN USER OPERATIONS

@GetMapping("/waiting")

public List<Citizen> getWaitingCitizens(){

List<String> waitingCitizensEmail= waitingCitizenService.getWaitingCitizensEmail();

List<Citizen> waitingCitizens=new ArrayList<>();

for(String email:waitingCitizensEmail) {

waitingCitizens.add(citizenService.getCitizenByEmail(email));

}

return waitingCitizens;

}

@PutMapping("/approve/{email}")

public String approveCitizenAadhar(@PathVariable String email) {

Citizen citizen=citizenService.getCitizenByEmail(email);

if(citizen==null)

return "Citizen not found!";

String aadharNumber=aadharService.randomAadharNumberGenerator();

while(isAadharNumberExists(aadharNumber)) {

aadharNumber=aadharService.randomAadharNumberGenerator();

}

AadharCard aadharCard=new AadharCard();

aadharCard.setId(aadharNumber);

aadharCard.setIssueDate(LocalDate.now());

aadharCard.setStatus("issued");

aadharCard.setCitizen(citizen);

aadharService.addAadhar(aadharCard);

citizen.setStatus("approved");

citizenService.editCitizen(citizen);

waitingCitizenService.removeWaitingCitizenEmail(email);

return "Aadhar Number Allocated... Aadhar Number ="+aadharNumber+", Citizen Name ="+citizen.getName();

}

//METHOD TO CHECK IF AADHAR NUMBER HAS ALREADY BEEN ALLOCATED TO A CITIZEN

public boolean isAadharNumberExists(String aadharNumber) {

Citizen testCitizen=citizenService.getCitizenByAadharNumber(aadharNumber);

return(testCitizen!=null);

}

@PutMapping("/reject/{email}")

public String rejectCitizenAadhar(@PathVariable String email) {

Citizen citizen=citizenService.getCitizenByEmail(email);

if(citizen==null)

return "Citizen not found!";

citizen.setStatus("rejected");

waitingCitizenService.removeWaitingCitizenEmail(email);

return "Aadhar request rejected";

}

@PutMapping("/notAlive/{email}")

public String updateDeadCitizen(@PathVariable String email) {

Citizen citizen=citizenService.getCitizenByEmail(email);

if(citizen==null)

return "0";

citizen.setIsAlive(false);

return "1";

}

}

package com.maadhar.controller;

import java.util.List;

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

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

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.maadhar.pojo.Citizen;

import com.maadhar.pojo.WaitingCitizensEmail;

import com.maadhar.service.CitizenService;

import com.maadhar.service.WaitingCitizensEmailService;

@RestController

@CrossOrigin(origins = "\*")

@RequestMapping("/citizen")

public class CitizenController {

@Autowired

CitizenService service;

@Autowired

WaitingCitizensEmailService waitingCitizenService;

@PostMapping("/")

public String registerCitizen(@RequestBody Citizen citizen) {

Citizen testCitizen1 = service.getCitizenByEmail(citizen.getEmailId());

Citizen testCitizen2 = service.getCitizenByMobile(citizen.getMobileNo());

if (testCitizen1!= null)

return "e1";

else if (testCitizen2 != null)

return "e2";

if (service.addCitizen(citizen) != null) {

WaitingCitizensEmail waitingCitizenEmail=new WaitingCitizensEmail();

waitingCitizenEmail.setEmailId(citizen.getEmailId());

if (waitingCitizenService.addWaitingCitizenEmail(waitingCitizenEmail))

return "1";

else return "e";

}else

return "e";

}

@GetMapping("/")

public List<Citizen> getAllCitizens() {

return service.getAllCitizens();

}

@GetMapping("/email/{email}")

public Citizen getCitizenByEmail(@PathVariable String email) {

return service.getCitizenByEmail(email);

}

@GetMapping("/mobile/{mobile}")

public Citizen getCitizenByMobile(@PathVariable String mobile) {

return service.getCitizenByMobile(mobile);

}

@PutMapping("/{email}/{mobile}")

public Citizen updateCitizen(@PathVariable String email, @PathVariable String mobile,

@RequestBody Citizen citizen) {

Citizen testCitizen = service.getCitizenByEmailIdAndMobile(email, mobile);

if(testCitizen==null) return null;

Long id=testCitizen.getId();

citizen.setId(id);

return(service.editCitizen(citizen));

}

@PutMapping("/login/{email}/{password}")

public String citizenLogIn(@PathVariable String email, @PathVariable String password) {

Citizen testCitizen=service.getCitizenByEmailIdAndMobile(email,password);

//Mobile number acts as Password

if (testCitizen==null)

return "0";

testCitizen.setIsLoggedIn(true);

service.editCitizen(testCitizen);

return "1";

}

@GetMapping("/aadharStatus/{email}/{password}")

public String checkAadharStatus(@PathVariable String email, @PathVariable String password) {

Citizen testCitizen=service.getCitizenByEmailIdAndMobile(email,password);

if(testCitizen==null)

return "invalid credentials";

switch(testCitizen.getStatus()) {

case "pending":default: return "Your aadhar status is still pending... Check back later";

case "approved": return "Your aadhar card has been approved.\r Your Aadhar Number : "+testCitizen.getAadharCard().getId();

case "rejected": return "Your aadhar application has been rejected... Please contact nearest Aadhar Center";

}

}

@PutMapping("/logout/{email}")

public String citizenLogOut(@PathVariable String email) {

Citizen testCitizen=service.getCitizenByEmail(email);

if (testCitizen==null)

return "Oops!";

testCitizen.setIsLoggedIn(false);

service.editCitizen(testCitizen);

return "Logged out!";

}

}

package com.maadhar.pojo;

import java.time.LocalDate;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.OneToOne;

import lombok.Data;

@Entity

@Data

public class AadharCard {

@Id

private String id;

@OneToOne

@JoinColumn(name="citizen\_id")

private Citizen citizen;

private LocalDate issueDate;

private String status;

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

public Citizen getCitizen() {

return citizen;

}

public void setCitizen(Citizen citizen) {

this.citizen = citizen;

}

public LocalDate getIssueDate() {

return issueDate;

}

public void setIssueDate(LocalDate issueDate) {

this.issueDate = issueDate;

}

public String getStatus() {

return status;

}

public void setStatus(String status) {

this.status = status;

}

}

**package** com.maadhar.pojo;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** lombok.Data;

@Entity

@Data

**public** **class** AdminUser {

@Id

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

**private** Long id;

**private** String adminName;

**private** String email;

**private** String password;

**private** Boolean isLoggedIn=**false**;

**public** Long getId() {

**return** id;

}

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

**this**.id = id;

}

**public** String getAdminName() {

**return** adminName;

}

**public** **void** setAdminName(String adminName) {

**this**.adminName = adminName;

}

**public** String getEmail() {

**return** email;

}

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

**this**.email = email;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

**public** Boolean getIsLoggedIn() {

**return** isLoggedIn;

}

**public** **void** setIsLoggedIn(Boolean isLoggedIn) {

**this**.isLoggedIn = isLoggedIn;

}

}

package com.maadhar.pojo;

import java.time.LocalDate;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.OneToOne;

import com.fasterxml.jackson.annotation.JsonIgnore;

import lombok.Data;

@Entity

@Data

public class Citizen {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private LocalDate dob;

private String address;

private String emailId;

private String mobileNo;

private String gender;

@JsonIgnore

@OneToOne(mappedBy = "citizen")

private AadharCard aadharCard;

private String status="pending";

private Boolean isLoggedIn=false;

private Boolean isAlive=true;

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public LocalDate getDob() {

return dob;

}

public void setDob(LocalDate dob) {

this.dob = dob;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

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 AadharCard getAadharCard() {

return aadharCard;

}

public void setAadharCard(AadharCard aadharCard) {

this.aadharCard = aadharCard;

}

public String getStatus() {

return status;

}

public void setStatus(String status) {

this.status = status;

}

public Boolean getIsLoggedIn() {

return isLoggedIn;

}

public void setIsLoggedIn(Boolean isLoggedIn) {

this.isLoggedIn = isLoggedIn;

}

public Boolean getIsAlive() {

return isAlive;

}

public void setIsAlive(Boolean isAlive) {

this.isAlive = isAlive;

}

}

package com.maadhar.pojo;

import javax.persistence.Entity;

import javax.persistence.Id;

import lombok.Data;

@Entity

@Data

public class WaitingCitizensEmail {

@Id

private String emailId;

@Override

public String toString() {

return emailId;

}

public String getEmailId() {

return emailId;

}

public void setEmailId(String emailId) {

this.emailId = emailId;

}

}

package com.maadhar.repo;

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

import com.maadhar.pojo.AadharCard;

public interface AadharCardRepo extends JpaRepository<AadharCard, Long>{

}

**package** com.maadhar.repo;

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

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

**import** com.maadhar.pojo.AdminUser;

**public** **interface** AdminRepo **extends** JpaRepository<AdminUser, Long> {

**public** AdminUser findByEmail(String email);

@Query("SELECT admin FROM AdminUser admin WHERE admin.email=?1 AND admin.password=?2")

**public** AdminUser findByEmailAndPassword(String email, String password);

}

package com.maadhar.repo;

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

import org.springframework.data.jpa.repository.Query;

import com.maadhar.pojo.Citizen;

public interface CitizenRepo extends JpaRepository<Citizen, Long> {

public Citizen getByEmailId(String emailId);

public Citizen getByMobileNo(String mobileNo);

@Query("SELECT user FROM Citizen user WHERE user.emailId=?1 AND user.mobileNo=?2")

public Citizen getByEmailIdAndMobile(String email, String mobile);

@Query("SELECT user FROM Citizen user JOIN user.aadharCard ua WHERE ua.id=?1")

public Citizen getByAadharNumber(String aadharNumber);

}

package com.maadhar.repo;

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

import com.maadhar.pojo.WaitingCitizensEmail;

public interface WaitingCitizensEmailRepo extends JpaRepository<WaitingCitizensEmail, String> {

}

package com.maadhar.service;

import java.util.Random;

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

import org.springframework.stereotype.Service;

import com.maadhar.pojo.AadharCard;

import com.maadhar.repo.AadharCardRepo;

@Service

public class AadharCardService {

@Autowired

AadharCardRepo repo;

public String randomAadharNumberGenerator() {

Random random=new Random();

long aadharNumber=100\_000\_000\_000L+(random.nextLong()%100\_000\_000\_000L);

return String.format("%012d", aadharNumber);

}

public void addAadhar(AadharCard aadharCard) {

repo.save(aadharCard);

}

}

package com.maadhar.service;

import java.util.List;

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

import org.springframework.stereotype.Service;

import com.maadhar.pojo.AdminUser;

import com.maadhar.repo.AdminRepo;

@Service

public class AdminService {

@Autowired

AdminRepo repo;

public AdminUser addAdmin(AdminUser adminUser) {

return repo.save(adminUser);

}

public List<AdminUser> getAllAdmin() {

return repo.findAll();

}

public AdminUser getAdminByEmail(String email) {

return repo.findByEmail(email);

}

public AdminUser editAdmin(AdminUser adminUser) {

return repo.save(adminUser);

}

public AdminUser getAdminByEmailAndPassword(String email, String password) {

return repo.findByEmailAndPassword(email,password);

}

}

package com.maadhar.service;

import java.util.List;

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

import org.springframework.stereotype.Service;

import com.maadhar.pojo.Citizen;

import com.maadhar.repo.CitizenRepo;

@Service

public class CitizenService {

@Autowired

CitizenRepo repo;

public Citizen getCitizenByEmail(String emailId) {

return repo.getByEmailId(emailId);

}

public Citizen getCitizenByMobile(String mobileNo) {

return repo.getByMobileNo(mobileNo);

}

public Citizen addCitizen(Citizen citizen) {

return repo.save(citizen);

}

public List<Citizen> getAllCitizens() {

return repo.findAll();

}

public Citizen getCitizenByEmailIdAndMobile(String email, String mobile) {

return repo.getByEmailIdAndMobile(email,mobile);

}

public Citizen editCitizen(Citizen citizen) {

return repo.save(citizen);

}

public Citizen getCitizenByAadharNumber(String aadharNumber) {

return repo.getByAadharNumber(aadharNumber);

}

}

package com.maadhar.service;

import java.util.ArrayList;

import java.util.List;

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

import org.springframework.stereotype.Service;

import com.maadhar.pojo.WaitingCitizensEmail;

import com.maadhar.repo.WaitingCitizensEmailRepo;

@Service

public class WaitingCitizensEmailService {

@Autowired

WaitingCitizensEmailRepo repo;

public boolean addWaitingCitizenEmail(WaitingCitizensEmail waitingCitizenEmail) {

if(repo.save(waitingCitizenEmail)!=null)

return true;

return false;

}

public List<String> getWaitingCitizensEmail() {

List<WaitingCitizensEmail>waitingList= repo.findAll();

List<String> waitingEmailList=new ArrayList<>();

for(WaitingCitizensEmail email: waitingList) {

waitingEmailList.add(email.toString());

System.out.println(email.toString());

}

return waitingEmailList;

}

public void removeWaitingCitizenEmail(String email) {

repo.deleteById(email);

}

}

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.maadhar</groupId>

<artifactId>mAadhar</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>war</packaging>

<name>mAadhar</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>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

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

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

<scope>provided</scope>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

<configuration>

<excludes>

<exclude>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</exclude>

</excludes>

</configuration>

</plugin>

</plugins>

</build>

</project>

Application.properties

server.port=8088

spring.jpa.hibernate.ddl-auto=update

spring.jpa.hibernate.dialect=org.hibernate.dialect.MySQLDialect

spring.jpa.show-sql = true

#spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.driverClassName=com.mysql.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:3306/aadharcard

spring.datasource.username=root

spring.datasource.password=Rishu@170955

Testing.java

package com.maadhar.MaadharTesting;

import static org.testng.Assert.assertEquals;

import java.io.File;

import java.io.IOException;

import java.util.concurrent.TimeUnit;

import org.apache.commons.io.FileUtils;

import org.openqa.selenium.By;

import org.openqa.selenium.OutputType;

import org.openqa.selenium.TakesScreenshot;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.testng.Reporter;

import org.testng.annotations.AfterTest;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Parameters;

import org.testng.annotations.Test;

public class AadharTesting {

private int countOfScreenshots = 0;

private WebDriver webDriver;

@Parameters({ "driverPath" })

@BeforeTest

public void setup(String driverPath) {

System.setProperty("webdriver.chrome.driver", driverPath);

webDriver = new ChromeDriver();

webDriver.manage().window().maximize();

webDriver.manage().timeouts().implicitlyWait(10000, TimeUnit.MILLISECONDS);

}

@Parameters({ "websiteURL", "fileLoc" })

@Test(priority = 1)

public void HomePageTest(String websiteURL, String fileLoc) throws IOException {

long startTime = System.currentTimeMillis();

webDriver.get(websiteURL);

long endTime = System.currentTimeMillis();

Reporter.log("Page load time: " + (endTime - startTime) + " milliseconds");

Reporter.log("Reached Home page ");

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

}

@Parameters({ "fileLoc" })

@Test(priority = 2)

public void UserRegsitrationTestInvalid(String fileLoc) throws InterruptedException, IOException {

webDriver.findElement(By.linkText("Register")).click();

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

Reporter.log("Reached Registration Page");

webDriver.findElement(By.xpath("/html/body/app-root/app-register/div/div/button[2]")).click();

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

Reporter.log("Using Existing email/mobile...");

webDriver.findElement(By.xpath("/html/body/app-root/app-register/div/div/div/form/div[1]/input")).sendKeys("Gaurav kumar");

webDriver.findElement(By.xpath("/html/body/app-root/app-register/div/div/div/form/div[2]/input")).sendKeys("gk@test.com");

webDriver.findElement(By.xpath("/html/body/app-root/app-register/div/div/div/form/div[3]/input")).sendKeys("9874658382");

webDriver.findElement(By.xpath("/html/body/app-root/app-register/div/div/div/form/div[4]/textarea")).sendKeys("123, New Delhi");

webDriver.findElement(By.xpath("/html/body/app-root/app-register/div/div/div/form/div[5]/select/option[1]")).click();

webDriver.findElement(By.xpath("/html/body/app-root/app-register/div/div/div/form/div[6]/input")).sendKeys("19085000");

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

webDriver.findElement(By.xpath("/html/body/app-root/app-register/div/div/div/form/button")).click();

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

Reporter.log("Citizen not Registered");

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

webDriver.findElement(By.linkText("Home")).click();

Reporter.log("Reached Home..");

}

@Parameters({ "fileLoc" })

@Test(priority = 3)

public void UserRegsitrationTestValid(String fileLoc) throws InterruptedException, IOException {

webDriver.findElement(By.linkText("Register")).click();

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

Reporter.log("Reached Registration Page");

webDriver.findElement(By.xpath("/html/body/app-root/app-register/div/div/button[2]")).click();

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

webDriver.findElement(By.xpath("/html/body/app-root/app-register/div/div/div/form/div[1]/input")).sendKeys("amit");

webDriver.findElement(By.xpath("/html/body/app-root/app-register/div/div/div/form/div[2]/input")).sendKeys("am@test.com");

webDriver.findElement(By.xpath("/html/body/app-root/app-register/div/div/div/form/div[3]/input")).sendKeys("9555568890");

webDriver.findElement(By.xpath("/html/body/app-root/app-register/div/div/div/form/div[4]/textarea")).sendKeys("1233, New Mumbai");

webDriver.findElement(By.xpath("/html/body/app-root/app-register/div/div/div/form/div[5]/select/option[1]")).click();

webDriver.findElement(By.xpath("/html/body/app-root/app-register/div/div/div/form/div[6]/input")).sendKeys("28012004");

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

webDriver.findElement(By.xpath("/html/body/app-root/app-register/div/div/div/form/button")).click();

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

String check=webDriver.findElement(By.xpath("/html/body/app-root/app-register/div/div[1]/h1")).getText();

assertEquals(check,"Registered Successfully! Login to see your Aadhar status");

Reporter.log("Citizen Successfully Registered");

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

webDriver.findElement(By.linkText("Home")).click();

Reporter.log("Reached Home..");

}

@Parameters({ "fileLoc" })

@Test(priority = 4)

public void AdminRegsitrationTestValid(String fileLoc) throws InterruptedException, IOException {

webDriver.findElement(By.linkText("Register")).click();

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

Reporter.log("Reached Registration Page");

webDriver.findElement(By.xpath("/html/body/app-root/app-register/div/div/button[1]")).click();

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

webDriver.findElement(By.xpath("/html/body/app-root/app-register/div/div/div/form/div[1]/input")).sendKeys("Admin6");

webDriver.findElement(By.xpath("/html/body/app-root/app-register/div/div/div/form/div[2]/input")).sendKeys("adm6@test.com");

webDriver.findElement(By.xpath("/html/body/app-root/app-register/div/div/div/form/div[3]/input")).sendKeys("Ad\_aab523");

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

webDriver.findElement(By.xpath("/html/body/app-root/app-register/div/div/div/form/button")).click();

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

String check=webDriver.findElement(By.xpath("/html/body/app-root/app-register/div/div[1]/h1")).getText();

assertEquals(check,"Registered Successfully!");

Reporter.log("Admin Successfully Registered");

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

webDriver.findElement(By.linkText("Home")).click();

Reporter.log("Reached Home..");

}

@Parameters({ "fileLoc" })

@Test(priority = 5)

public void AdminLoginTest(String fileLoc) throws InterruptedException, IOException {

webDriver.findElement(By.linkText("Login")).click();

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

Reporter.log("Reached Login Page");

webDriver.findElement(By.xpath("/html/body/app-root/app-login/div/div/button[1]")).click();

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

webDriver.findElement(By.xpath("//\*[@id=\"email\"]")).sendKeys("adm2@test.com");

webDriver.findElement(By.xpath("//\*[@id=\"password\"]")).sendKeys("Ad\_aad123");

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

webDriver.findElement(By.xpath("/html/body/app-root/app-admin/div/div/form/button")).click();

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

Reporter.log("Logged in successfully!");

webDriver.findElement(By.xpath("/html/body/app-root/app-admin/div/div[2]/div/button[1]")).click();

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

webDriver.findElement(By.xpath("//html/body/app-root/app-admin/div/div[2]/table/tbody/tr[1]/td[8]/button")).click();

Reporter.log("Approving Citizen for Aadhar... Generating Aadhar Number");

webDriver.findElement(By.xpath("/html/body/app-root/app-admin/div/div[2]/div/button[1]")).click();

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

webDriver.findElement(By.xpath("//html/body/app-root/app-admin/div/div[2]/table/tbody/tr[1]/td[8]/button")).click();

Reporter.log("Rejecting Citizen for Aadhar...");

webDriver.findElement(By.xpath("/html/body/app-root/app-admin/div/div[2]/div/button[2]")).click();

Thread.sleep(500);

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

webDriver.findElement(By.xpath("//\*[@id=\"email\"]")).sendKeys("va@test.com");

webDriver.findElement(By.xpath("/html/body/app-root/app-admin/div/div[2]/div/form/button")).click();

Reporter.log("Marking Dead Citizen");

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

Reporter.log("Logging out");

webDriver.findElement(By.xpath("/html/body/app-root/app-admin/div/div[1]/div[2]/button")).click();

}

@Parameters({ "fileLoc" })

@Test(priority = 6)

public void CitizenLoginTest1(String fileLoc) throws InterruptedException, IOException {

webDriver.findElement(By.linkText("Home")).click();

webDriver.findElement(By.linkText("Login")).click();

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

Reporter.log("Reached Login Page");

webDriver.findElement(By.xpath("/html/body/app-root/app-login/div/div/button[2]")).click();

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

webDriver.findElement(By.xpath("//\*[@id=\"email\"]")).sendKeys("su@test.com");

webDriver.findElement(By.xpath("//\*[@id=\"password\"]")).sendKeys("9773526401");

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

webDriver.findElement(By.xpath("/html/body/app-root/app-citizen/div/div/form/button")).click();

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

Reporter.log("Logged in successfully!");

webDriver.findElement(By.xpath("/html/body/app-root/app-citizen/div/table/tbody/tr/td[9]/button")).click();

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

Reporter.log("Viewing Status");

webDriver.findElement(By.xpath("/html/body/app-root/app-citizen/div/div/button")).click();

Reporter.log("Logging out");

}

@Parameters({ "fileLoc" })

@Test(priority =7)

public void CitizenLoginTest2(String fileLoc) throws InterruptedException, IOException {

webDriver.findElement(By.linkText("Login")).click();

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

Reporter.log("Reached Login Page");

webDriver.findElement(By.xpath("/html/body/app-root/app-login/div/div/button[2]")).click();

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

webDriver.findElement(By.xpath("//\*[@id=\"email\"]")).sendKeys("sh@test.com");

webDriver.findElement(By.xpath("//\*[@id=\"password\"]")).sendKeys("1235567890");

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

webDriver.findElement(By.xpath("/html/body/app-root/app-citizen/div/div/form/button")).click();

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

Reporter.log("Logged in successfully!");

webDriver.findElement(By.xpath("/html/body/app-root/app-citizen/div/table/tbody/tr/td[9]/button")).click();

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

Reporter.log("Viewing Status");

webDriver.findElement(By.xpath("/html/body/app-root/app-citizen/div/div/button")).click();

Reporter.log("Logging out");

}

@Parameters({ "fileLoc" })

@Test(priority = 8)

public void CitizenLoginTest3(String fileLoc) throws InterruptedException, IOException {

webDriver.findElement(By.linkText("Login")).click();

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

Reporter.log("Reached Login Page");

webDriver.findElement(By.xpath("/html/body/app-root/app-login/div/div/button[2]")).click();

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

webDriver.findElement(By.xpath("//\*[@id=\"email\"]")).sendKeys("si@test.com");

webDriver.findElement(By.xpath("//\*[@id=\"password\"]")).sendKeys("1235568890");

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

webDriver.findElement(By.xpath("/html/body/app-root/app-citizen/div/div/form/button")).click();

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

Reporter.log("Logged in successfully!");

webDriver.findElement(By.xpath("/html/body/app-root/app-citizen/div/table/tbody/tr/td[9]/button")).click();

Thread.sleep(500);

takeScreenshot(webDriver, countOfScreenshots++, fileLoc);

Reporter.log("Viewing Status");

webDriver.findElement(By.xpath("/html/body/app-root/app-citizen/div/div/button")).click();

Reporter.log("Logging out");

}

@AfterTest

public void tearDown() {

webDriver.quit();

}

public static void takeScreenshot(WebDriver wd, int count, String fileLoc) throws IOException {

File file = ((TakesScreenshot) wd).getScreenshotAs(OutputType.FILE);

FileUtils.copyFile(file, new File(fileLoc +"testimg\_"+ count + ".png"));

}

}

Pom.xml

<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 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.maadhar</groupId>

<artifactId>MaadharTesting</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>MaadharTesting</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<dependency>

<groupId>mysql</groupId>

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

<version>8.0.30</version>

</dependency>

<!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>3.1.0</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>io.github.bonigarcia</groupId>

<artifactId>webdrivermanager</artifactId>

<version>5.6.2</version>

</dependency>

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>3.141.59</version>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.1</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/org.testng/testng -->

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>7.5.1</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

Front end:

<!DOCTYPE html>

<html lang="en">

<head>

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

</head>

<body>

<div class="container-fluid">

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

    <a class="navbar-brand" routerLink="">

      <img src="../assets/images/aadhar\_logo.jpg" alt="aadhar\_logo" width="30" height="30" class="d-inline-block align-text-top">

      M-Aadhar

    </a>

    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNavAltMarkup" aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle navigation">

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

    </button>

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

      <div class="navbar-nav ms-auto">

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

        <a class="nav-link" routerLink="/login">Login</a>

        <a class="nav-link" routerLink="/register">Register</a>

      </div>

    </div>

  </nav>

</div>

<router-outlet></router-outlet>

<

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

</body>

</html>

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

import { RouterTestingModule } from '@angular/router/testing';

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

describe('AppComponent', () => {

  beforeEach(() => TestBed.configureTestingModule({

    imports: [RouterTestingModule],

    declarations: [AppComponent]

  }));

  it('should create the app', () => {

    const fixture = TestBed.createComponent(AppComponent);

    const app = fixture.componentInstance;

    expect(app).toBeTruthy();

  });

  it(`should have as title 'mAadhar-Application'`, () => {

    const fixture = TestBed.createComponent(AppComponent);

    const app = fixture.componentInstance;

    expect(app.title).toEqual('mAadhar-Application');

  });

  it('should render title', () => {

    const fixture = TestBed.createComponent(AppComponent);

    fixture.detectChanges();

    const compiled = fixture.nativeElement as HTMLElement;

    expect(compiled.querySelector('.content span')?.textContent).toContain('mAadhar-Application app is running!');

  });

});

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

@Component({

  selector: 'app-root',

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

  styleUrl: './app.component.css'

})

export class AppComponent {

  title = 'maadharapp';

}

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

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

import { AppRoutingModule } from './app-routing.module';

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

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

import { LoginComponent } from './login/login.component';

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

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

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

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

import { AadharService } from './aadhar.service';

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

import { CitizenComponent } from './citizen/citizen.component';

const routes: Routes=[

  {

    "path":"",

    "component":HomeComponent

  },

  {

    "path":"home",

    "component":HomeComponent

  },

  {

    "path":"login",

    "component":LoginComponent

  },

  {

    "path":"register",

    "component":RegisterComponent

  },

  {

    "path":"citizen",

    "component":CitizenComponent

  },

  {

    "path":"admin",

    "component":AdminComponent

  }

]

@NgModule({

  declarations: [

    AppComponent,

    HomeComponent,

    LoginComponent,

    RegisterComponent,

    AdminComponent,

    CitizenComponent

  ],

  imports: [

    BrowserModule,

    AppRoutingModule,

    FormsModule,

    HttpClientModule,

    ReactiveFormsModule,

    RouterModule.forRoot(routes)

  ],

  providers: [AadharService],

  bootstrap: [AppComponent]

})

export class AppModule { }

<div style="height: 100vh">

    <span \*ngIf="showDashboard; then Dashboard; else LoginForm"></span>

    <ng-template #Dashboard>

      <div style="display: flex; justify-content: flex-end">

        <div>

          <div> <button type="submit" class="btn btn-success" (click)="home()">

            Main Page

          </button></div>

        </div>

        <div> <button type="submit" class="btn btn-danger" (click)="logout()">

          Logout

        </button></div>

      </div>

      <h1 style="text-align: center">Welcome,{{ admin.adminName }}!</h1>

      <div \*ngIf="message2=='Citizen Marked as not Alive'" ><h2 style="color: red;text-align: center;">{{message2}}</h2></div>

      <div \*ngIf="btn\_area">

        <div class="d-grid gap-4 col-3 mx-auto">

          <button

            class="btn btn-success"

            type="button"

            (click)="showWaitingCitizens()"

          >

            View Waiting Citizens

          </button>

          <button

            class="btn btn-success"

            type="button"

            (click)="showDeadCitizens()"

          >

            Update Dead Citizens

          </button>

        </div>

      </div>

      <div \*ngIf="wait\_area">

        <h2 style="text-align: center">List of Waiting Citizens</h2>

        <table class="table table-info">

          <thead>

            <tr>

              <th>Citizen ID</th>

              <th>Name</th>

              <th>Date of Birth</th>

              <th>Email</th>

              <th>Mobile No</th>

              <th>Address</th>

              <th>Gender</th>

              <th>Approve</th>

              <th>Reject</th>

            </tr>

          </thead>

          <tbody>

            <tr \*ngFor="let waitingCitizen of waitingCitizens">

              <td>{{ waitingCitizen.id }}</td>

              <td>{{ waitingCitizen.name }}</td>

              <td>{{ waitingCitizen.dob | date : "dd-MMM-YYYY" }}</td>

              <td>{{ waitingCitizen.emailId }}</td>

              <td>{{ waitingCitizen.mobileNo }}</td>

              <td>{{ waitingCitizen.address }}</td>

              <td>{{ waitingCitizen.gender }}</td>

              <td>

                <button

                  type="submit"

                  class="btn btn-success"

                  (click)="approve(waitingCitizen.emailId)"

                >

                  Approve

                </button>

              </td>

              <td>

                <button

                  type="submit"

                  class="btn btn-danger"

                  (click)="reject(waitingCitizen.emailId)"

                >

                  Reject

                </button>

              </td>

            </tr>

          </tbody>

        </table>

      </div>

      <div \*ngIf="notAlive\_area">

        <h2 style="text-align: center">Update Dead Citizens</h2>

        <div \*ngIf="message2=='Invalid email!'"><h2 style="text-align: center;color: red;">{{message2}}</h2></div>

        <div class="col-5">

          <div class="mb-3">

            <h3>Enter email ID of Dead Citizen</h3>

          </div>

          <form>

            <div class="col-sm-8">

              Email

              <input

                type="email"

                class="form-control"

                name="email"

                id="email"

                [(ngModel)]="emailToRem"

                required

              />

            </div>

            <button type="submit" class="btn btn-danger" (click)="update()">

              Mark as Dead

            </button>

          </form>

        </div>

      </div>

    </ng-template>

    <ng-template #LoginForm>

      <h1 style="line-height: 2em; text-align: center">Login as Admin</h1>

      <h4 style="text-align: center; color: red">{{ message }}</h4>

      <div class="col-sm-3" style="margin: auto">

        <form name="regForm">

          <div class="form-group">

            <label>Email</label>

            <input

              type="email"

              class="form-control"

              name="email"

              id="email"

              [(ngModel)]="admin.email"

              required

            />

          </div>

          <div class="form-group">

            Password

            <input

              type="password"

              class="form-control"

              name="password"

              id="password"

              [(ngModel)]="admin.password"

              required

            />

          </div>

          <button type="submit" class="btn btn-dark" (click)="loginAsAdmin()">

            Login

          </button>

        </form>

      </div>

    </ng-template>

  </div>

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

import { AadharService } from '../aadhar.service';

import { AdminUser } from '../classes';

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

@Component({

  selector: 'app-admin',

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

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

})

export class AdminComponent implements OnInit {

  admin: AdminUser = new AdminUser();

  constructor(private service: AadharService, private router: Router) { }

  message = '';

  message2 = '';

  emailToRem = '';

  waitingCitizens: any = {}

  deadCitizens: any = {}

  showDashboard = false;

  btn\_area = true;

  wait\_area = false;

  notAlive\_area = false;

  ngOnInit(): void {

    this.btn\_area = true;

    this.wait\_area = false;

    this.notAlive\_area = false;

    if (this.showDashboard == true)

      this.loginAsAdmin();

  }

  public home() {

    this.message = '';

    this.message2 = '';

    this.emailToRem = '';

    this.showDashboard = true;

    this.btn\_area = true;

    this.wait\_area = false;

    this.notAlive\_area = false;

  }

  public showWaitingCitizens() {

    this.btn\_area = false;

    this.wait\_area = true;

    this.notAlive\_area = false;

    this.service.getWaitingList().subscribe(response => this.waitingCitizens = response)

  }

  public showDeadCitizens() {

    this.btn\_area = false;

    this.wait\_area = false;

    this.notAlive\_area = true;

    this.service.getWaitingList().subscribe(response => this.deadCitizens = response)

  }

  public loginAsAdmin() {

    this.service.checkAdminLoginCreds(this.admin.email, this.admin.password).subscribe(

      respose => {

        console.log(respose);

        if (respose == '1') {

          this.service.getAdmin(this.admin.email).subscribe(response2 => this.admin = response2)

          this.showDashboard = true;

          this.btn\_area = true;

          this.wait\_area = false;

          this.notAlive\_area = false;

        }

        else {

          this.message = "Invalid Credentials!";

        }

      }

    )

  }

  public approve(email: string) {

    console.log(email);

    this.service.approve(email).subscribe();

    this.ngOnInit();

  }

  public reject(email: string) {

    this.service.reject(email).subscribe();

    this.ngOnInit();

  }

  public logout() {

    this.service.adminLogout(this.admin.email).subscribe();

    this.router.navigate(['']);

  }

  public update() {

    this.service.updateDeadCitizen(this.emailToRem).subscribe(

      respose => {

        console.log(respose);

        if (respose == '1') {

          this.showDashboard = true;

          this.message2 = 'Citizen Marked as not Alive'

          this.btn\_area = true;

          this.wait\_area = false;

          this.notAlive\_area = false;

        }

        else {

          this.message2 = "Invalid email!";

        }

      }

    )

  }

}

<div style="height: 100vh">

    <span \*ngIf="showDashboard; then Dashboard; else LoginForm"></span>

    <ng-template #Dashboard>

      <div style="display: flex; justify-content: flex-end " >

        <button type="submit" class="btn btn-danger" (click)="logout()">

          Logout

        </button>

      </div>

      <h1 style="text-align: center">Welcome,{{ citizen.name }}!</h1>

      <table class="table table-info">

        <thead>

          <tr>

            <th>Citizen ID</th>

            <th>Name</th>

            <th>Date of Birth</th>

            <th>Email</th>

            <th>Mobile No</th>

            <th>Address</th>

            <th>Gender</th>

            <th>Aadhar Status</th>

            <th>More Info</th>

          </tr>

        </thead>

        <tbody>

          <tr>

            <td>{{ citizen.id }}</td>

            <td>{{ citizen.name }}</td>

            <td>{{ citizen.dob | date : "dd-MMM-YYYY" }}</td>

            <td>{{ citizen.emailId }}</td>

            <td>{{ citizen.mobileNo }}</td>

            <td>{{ citizen.address }}</td>

            <td>{{ citizen.gender }}</td>

            <td>{{ citizen.status }}</td>

            <td>

              <button

                type="submit"

                class="btn btn-dark"

                (click)="getAadharStatus(citizen.emailId, citizen.mobileNo)"

              >

                Click Here

              </button>

            </td>

          </tr>

        </tbody>

      </table>

      <div \*ngIf="aadharStatus != ''">

        <h1 style="color: green; text-align: center">{{ aadharStatus }}</h1>

      </div>

    </ng-template>

    <ng-template #LoginForm>

      <h1 style="line-height: 2em; text-align: center">Login as Citizen</h1>

      <h4 style="text-align: center; color: red">{{ message }}</h4>

      <div class="col-sm-3" style="margin: auto">

        <form name="regForm">

          <div class="form-group">

            <label>Email</label>

            <input

              type="email"

              class="form-control"

              name="email"

              id="email"

              [(ngModel)]="citizen.emailId"

              required

            />

          </div>

          <div class="form-group">

            Password(Your mobile no.)

            <input

              type="password"

              class="form-control"

              name="password"

              id="password"

              [(ngModel)]="citizen.mobileNo"

              required

            />

          </div>

          <button type="submit" class="btn btn-dark" (click)="loginAsCitizen()">

            Login

          </button>

        </form>

      </div>

    </ng-template>

  </div>

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

import { AadharService } from '../aadhar.service';

import { Citizen } from '../classes';

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

@Component({

  selector: 'app-citizen',

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

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

})

export class CitizenComponent implements OnInit{

constructor(private service:AadharService,private router:Router){}

message='';

aadharStatus='';

citizen:Citizen=new Citizen();

showDashboard = false;

ngOnInit(): void {

}

public loginAsCitizen() {

  this.service.checkCitizenLoginCreds(this.citizen.emailId, this.citizen.mobileNo).subscribe(

    respose => {

      console.log(respose);

      if (respose == '1') {

        this.service.getCitizen(this.citizen.emailId).subscribe(response2 => this.citizen = response2)

        this.showDashboard = true;

      }

      else {

        this.message = "Invalid Credentials!";

      }

    }

  )

}

public logout(){

  this.service.citizenLogout(this.citizen.emailId).subscribe();

  this.router.navigate(['']);

}

public getAadharStatus(emailId:string,password:string){

  this.service.getAadharStatus(emailId,password).subscribe(response=>this.aadharStatus=response)

  console.log(this.aadharStatus);

}

}

<div style="height: 100vh">

    <div

      id="carouselExampleControls"

      class="carousel slide"

      data-bs-ride="carousel"

    >

      <div class="carousel-inner" >

        <div class="carousel-item active" style="height: 100vh" >

          <img

            src="assets/images/carousel\_1.jpg"

            class="d-block w-100"

            style="height: 95vh; width: 100vw;"

          />

        </div>

        <!-- <div class="carousel-item">

          <img

            src="assets/images/carousel\_2.jpg"

            class="d-block w-100"

            alt="..."

          /> -->

        <!-- </div>

        <div class="carousel-item">

          <img

            src="assets/images/carousel\_3.jpeg"

            class="d-block w-100"

            alt="..."

          />

        </div> -->

      <!-- </div> -->

      <!-- <button

        class="carousel-control-prev"

        type="button"

        data-bs-target="#carouselExampleControls"

        data-bs-slide="prev"

      >

        <span class="carousel-control-prev-icon" aria-hidden="true"></span>

        <span class="visually-hidden">Previous</span>

      </button>

      <button

        class="carousel-control-next"

        type="button"

        data-bs-target="#carouselExampleControls"

        data-bs-slide="next"

      >

        <span class="carousel-control-next-icon" aria-hidden="true"></span>

        <span class="visually-hidden">Next</span>

      </button> -->

    </div>

  </div>

import { ComponentFixture, TestBed } from '@angular/core/testing';

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

describe('HomeComponent', () => {

  let component: HomeComponent;

  let fixture: ComponentFixture<HomeComponent>;

  beforeEach(() => {

    TestBed.configureTestingModule({

      declarations: [HomeComponent]

    });

    fixture = TestBed.createComponent(HomeComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

<div style="height: 100vh;">

    <h1 style="line-height:2em; text-align: center;">Select appropriate login</h1>

    <div class="d-grid gap-4 col-3 mx-auto " style="text-align: center;">

        <button class="btn btn-success" type="button" (click)="adminForm()" style="margin-right: 10px;">Login as Admin</button>

        <button class="btn btn-success" type="button" (click)="citizenForm()">Login as Citizen</button>

      </div>

    </div>

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { LoginComponent } from './login.component';

describe('LoginComponent', () => {

  let component: LoginComponent;

  let fixture: ComponentFixture<LoginComponent>;

  beforeEach(() => {

    TestBed.configureTestingModule({

      declarations: [LoginComponent]

    });

    fixture = TestBed.createComponent(LoginComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

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

import { AdminUser, Citizen } from '../classes';

import { AadharService } from '../aadhar.service';

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

@Component({

  selector: 'app-login',

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

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

})

export class LoginComponent implements OnInit{

  showOptions=true;

  showAdminForm=false;

  showCitizenForm=false;

  message="";

  admin: AdminUser=new AdminUser();

  citizen: Citizen=new Citizen();

  constructor(private service:AadharService, private router:Router ) {}

  ngOnInit(): void {

  }

  public adminForm():void{

    this.router.navigate(['/admin']);

  }

  public citizenForm():void{

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

  }

}

<div style="height: 100vh">

    <span \*ngIf="showRegHome; then RegHome; else RegForm"></span>

    <ng-template #RegHome>

      <div \*ngIf="message != ''">

        <h1 style="color: green; text-align: center">{{ message }}</h1>

      </div>

      <h1 style="line-height: 2em; text-align: center">

        Select appropriate Registration

      </h1>

      <div class="d-grid gap-4 col-3 mx-auto">

        <button class="btn btn-success" type="button" (click)="showsAdminForm()">

          Register as Admin

        </button>

        <button

          class="btn btn-success"

          type="button"

          (click)="showsCitizenForm()"

        >

          Register as Citizen

        </button>

      </div>

    </ng-template>

    <ng-template #RegForm>

      <div \*ngIf="showAdminForm">

        <div \*ngIf="message != ''">

          <h1 style="color: red; text-align: center">{{ message }}</h1>

        </div>

        <!-- Admin Registration -->

        <div class="col-sm-7" style="margin: auto">

          <h1 style="text-align: center;">Admin Registration</h1>

          <form (ngSubmit)="registerAdmin()" [formGroup]="adminForm">

            <div class="form-group">

              <label>Name</label>

              <input

                type="text"

                class="form-control"

                formControlName="adminName"

                required

              />

            </div>

            <div class="form-group">

              <label>Email</label>

              <input

                type="email"

                class="form-control"

                formControlName="email"

                required

              />

              <div \*ngIf="adminForm.get('email')?.errors?.['required']">

                <span style="color: red">\*Email is required</span>

              </div>

              <div \*ngIf="adminForm.get('email')?.errors?.['email']">

                <span style="color: red">\*Invalid email format</span>

              </div>

            </div>

            <div class="form-group">

              <label>Password</label>

              <input

                type="password"

                class="form-control"

                formControlName="password"

                required

                minlength="6"

              />

              <div \*ngIf="adminForm.get('password')?.errors?.['required']">

                <span style="color: red">\* Password is required</span>

              </div>

              <div \*ngIf="adminForm.get('password')?.errors?.['minlength']">

                <span style="color: red"

                  >\* Password should be at least 6 characters long</span

                >

              </div>

              <div \*ngIf="adminForm.get('password')?.errors?.['invalidPassword']">

                <span style="color: red"

                  >\* Password should contain at least 1 upper case, 1 lower case, 1

                  number, and 1 special character</span

                >

              </div>

            </div>

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

          </form>

        </div>

      </div>

      <div \*ngIf="showCitizenForm">

        <!-- Citizen Registration -->

        <div class="col-sm-6" style="margin: auto">

          <h1 style="text-align: center;">Citizen Registration</h1>

          <form (ngSubmit)="registerCitizen()" [formGroup]="citizenForm">

            <div class="form-group">

              <label>Name</label>

              <input

                type="text"

                class="form-control"

                formControlName="name"

                required

              />

            </div>

            <div class="form-group">

              <label>Email</label>

              <input

                type="email"

                class="form-control"

                formControlName="emailId"

                required

              />

              <div \*ngIf="citizenForm.get('emailId')?.errors?.['required']">

                <span style="color: red">\* Email is required</span>

              </div>

              <div \*ngIf="citizenForm.get('emailId')?.errors?.['email']">

                <span style="color: red">\* Invalid email format</span>

              </div>

            </div>

            <div class="form-group">

              <label>Mobile</label>

              <input

                type="tel"

                class="form-control"

                formControlName="mobileNo"

                required

                pattern="[0-9]{10}"

              />

              <div \*ngIf="citizenForm.get('mobileNo')?.errors?.['required']">

                <span style="color: red">\* Mobile number is required</span>

              </div>

              <div \*ngIf="citizenForm.get('mobileNo')?.errors?.['pattern']">

                <span style="color: red"

                  >\* Mobile number should be 10 digits long and contain only

                  digits</span

                >

              </div>

            </div>

            <div class="form-group">

              <label>Address</label>

              <textarea

                class="form-control"

                formControlName="address"

                required

              ></textarea>

            </div>

            <div class="form-group">

              <label>Gender</label>

              <select class="form-control" formControlName="gender" required>

                <option value="Male">Male</option>

                <option value="Female">Female</option>

                <option value="Others">Others</option>

              </select>

            </div>

            <div class="form-group">

              <label>Date of Birth</label>

              <input

                type="date"

                class="form-control"

                formControlName="dob"

                required

              />

            </div>

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

          </form>

        </div>

      </div>

    </ng-template>

  </div>

import { ComponentFixture, TestBed } from '@angular/core/testing';

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

describe('RegisterComponent', () => {

  let component: RegisterComponent;

  let fixture: ComponentFixture<RegisterComponent>;

  beforeEach(() => {

    TestBed.configureTestingModule({

      declarations: [RegisterComponent]

    });

    fixture = TestBed.createComponent(RegisterComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

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

import { AadharService } from '../aadhar.service';

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

function passwordValidator(control: AbstractControl): { [key: string]: boolean } | null {

  const value = control.value;

  const hasUpper = /[A-Z]/.test(value);

  const hasLower = /[a-z]/.test(value);

  const hasNumber = /\d/.test(value);

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

  if (hasUpper && hasLower && hasNumber && hasSpecial) {

    return null;

  } else {

    return { 'invalidPassword': true };

  }

}

@Component({

  selector: 'app-register',

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

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

})

export class RegisterComponent {

  adminForm: FormGroup;

  citizenForm: FormGroup;

  admin = {

    adminName: '',

    email: '',

    password: ''

  };

  citizen = {

    name: '',

    emailId: '',

    mobileNo: '',

    address: '',

    gender: 'M',

    dob: ''

  };

  constructor(private service:AadharService, private fb: FormBuilder){

    this.adminForm = this.fb.group({

      adminName: ['', Validators.required],

      email: ['', [Validators.required, Validators.email]],

      password: ['', [Validators.required, Validators.minLength(6), passwordValidator]]

    });

    this.citizenForm = this.fb.group({

      name: ['', Validators.required],

      emailId: ['', [Validators.required, Validators.email]],

      mobileNo: ['', [Validators.required, Validators.pattern('[0-9]{10}')]],

      address: ['', Validators.required],

      gender: ['M', Validators.required],

      dob: ['', Validators.required]

    });

  }

  showRegHome=true;

  showAdminForm=false;

  showCitizenForm=false;

  message='';

  showsAdminForm(){

    this.showRegHome=false;

    this.showAdminForm=true;

    this.showCitizenForm=false;

  }

  showsCitizenForm(){

    this.showRegHome=false;

    this.showAdminForm=false;

    this.showCitizenForm=true;

  }

  registerAdmin(){

    console.log('Admin Form Data:', this.adminForm.value);

    this.service.registerAdmin(this.adminForm.value).subscribe(

      response=>{

        if(response=='1'){

          this.message="Registered Successfully!"

          this.showRegHome=true;

          this.showAdminForm=false;

          this.showCitizenForm=false;

        }

        else{

          this.message='Admin already exists with this email ID!'

        }

    });

  }

  registerCitizen(){

    console.log('Citizen Form Data:', this.citizenForm.value);

    this.service.registerCitizen(this.citizenForm.value).subscribe(

      response=>{

        if(response=='1'){

          this.message="Registered Successfully! Login to see your Aadhar status"

          this.showRegHome=true;

          this.showAdminForm=false;

          this.showCitizenForm=false;

        }

        else if(response=='e1'){

          this.message='Citizen already exists with this email ID!'

        }

        else if(response=='e2'){

          this.message='Citizen already exists with this Mobile Number!'

        }

        else{

          this.message='Oops, something went wrong! Try again.'

        }

    });

  }

}