SpringemployeeProjectApplication.java:

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringemployeeProjectApplication {

public static void main(String[] args) {

SpringApplication.run(SpringemployeeProjectApplication.class, args);

}

}

SecurityConfig.java:

package com.example.demo.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.web.SecurityFilterChain;

@Configuration

public class SecurityConfig {

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http.csrf().disable()

.authorizeHttpRequests(auth -> auth

.requestMatchers(

"/home", "/register", "/logout", "/login", "/send-otp", "/verify-otp",

"/userlogin", "/download-dsc/\*\*", "/update-auth", "/userlist",

"/css/\*\*", "/js/\*\*", "/images/\*\*", "/userlogin", "/verify-dsc", "/userdashboard", "/user/send-otp", "/user/verify-otp"

).permitAll()

.anyRequest().authenticated()

)

.formLogin(login -> login

.loginPage("/login")

.defaultSuccessUrl("/userlist", true)

.permitAll()

)

.logout(logout -> logout.logoutSuccessUrl("/home"));

return http.build();

}

}

LoginController.java:

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

**import com.example.demo.dto.OtpLogin;**

**import com.example.demo.model.Admin;**

**import com.example.demo.model.User;**

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

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

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

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

**import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;**

**import org.springframework.stereotype.Controller;**

**import org.springframework.ui.Model;**

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

**import java.util.\*;**

**@Controller**

**public class LoginController {**

**@Autowired**

**private AdminRepository adminRepo;**

**@Autowired**

**private UserRepository userRepo;**

**@Autowired**

**private EmailService emailService;**

**// Temporary OTP store (email -> OTP)**

**private Map<String, String> otpStore = new HashMap<>();**

**// Show login page with OTP form**

**@GetMapping("/login")**

**public String showLoginPage(Model model) {**

**model.addAttribute("otpLogin", new OtpLogin());**

**return "login";**

**}**

**// Handle "Generate OTP" submission**

**@PostMapping("/send-otp")**

**public String sendOtp(@ModelAttribute OtpLogin otpLogin, Model model) {**

**Admin admin = adminRepo.findByUsername(otpLogin.getUsername());**

**BCryptPasswordEncoder encoder = new BCryptPasswordEncoder();**

**if (admin == null || !encoder.matches(otpLogin.getPassword(), admin.getPassword())**

**|| !admin.getEmail().equals(otpLogin.getEmail())) {**

**model.addAttribute("error", "Invalid credentials");**

**return "login";**

**}**

**// Generate and store OTP**

**String generatedOtp = String.format("%06d", new Random().nextInt(999999));**

**otpStore.put(otpLogin.getEmail(), generatedOtp);**

**// Send OTP via email**

**emailService.sendOtp(otpLogin.getEmail(), generatedOtp);**

**model.addAttribute("otpLogin", otpLogin);**

**model.addAttribute("otpSent", true);**

**// Do not display OTP on screen anymore**

**return "login";**

**}**

**// Handle OTP verification**

**@PostMapping("/verify-otp")**

**public String verifyOtp(@ModelAttribute OtpLogin otpLogin, Model model) {**

**String correctOtp = otpStore.get(otpLogin.getEmail());**

**if (correctOtp != null && correctOtp.equals(otpLogin.getOtp())) {**

**List<User> users = (List<User>) userRepo.findAll();**

**model.addAttribute("users", users);**

**otpStore.remove(otpLogin.getEmail());**

**return "userlist";**

**} else {**

**model.addAttribute("error", "Invalid OTP");**

**model.addAttribute("otpSent", true);**

**model.addAttribute("otpLogin", otpLogin);**

**return "login";**

**}**

**}**

**@PostMapping("/update-auth")**

**public String updateAuthModes(@RequestParam List<Long> userIds,**

**@RequestParam Map<String, String> allParams,**

**Model model) {**

**for (Long userId : userIds) {**

**String paramName = "authMode\_" + userId;**

**String selectedMode = allParams.getOrDefault(paramName, "NONE");**

**userRepo.findById(userId).ifPresent(u -> {**

**u.setAuthMode(selectedMode);**

**userRepo.save(u);**

**});**

**}**

**model.addAttribute("users", userRepo.findAll());**

**model.addAttribute("success", true);**

**return "userlist";**

**}**

**// Optional logout redirect**

**@GetMapping("/logout")**

**public String logout() {**

**return "redirect:/home";**

**}**

**}**

UserController.java:  
package com.example.demo.controller;

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

import org.springframework.core.io.Resource;

import org.springframework.core.io.UrlResource;

import org.springframework.http.HttpHeaders;

import org.springframework.http.ResponseEntity;

import org.springframework.mail.SimpleMailMessage;

import org.springframework.mail.javamail.JavaMailSender;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

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

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

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

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

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

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

import org.springframework.web.multipart.MultipartFile;

import com.example.demo.model.User;

import com.example.demo.repository.UserRepository;

import jakarta.servlet.http.HttpSession;

import java.io.FileInputStream;

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.nio.file.StandardCopyOption;

import java.security.KeyStore;

import java.security.PublicKey;

import java.security.cert.Certificate;

import java.security.cert.CertificateFactory;

import java.util.Base64;

import java.util.Random;

@Controller

public class UserController {

@Autowired

private JavaMailSender mailSender;

@Autowired

private UserRepository userRepository;

@GetMapping("/home")

public String showHomePage() {

return "home";

}

@GetMapping("/userlogin")

public String showUserLoginForm() {

return "userlogin";

}

/\* @PostMapping("/userlogin")

public String processUserLogin(@RequestParam String email, @RequestParam String phone, Model model) {

User user = userRepository.findByEmail(email);

if (user != null && user.getPhone().equals(phone)) {

model.addAttribute("user", user);

return "success"; // You can create a separate user dashboard if needed

} else {

model.addAttribute("error", "Invalid email or phone");

return "userlogin";

}

}\*/

@GetMapping("/register")

public String showForm(Model model) {

model.addAttribute("user", new User());

return "register";

}

@PostMapping("/register")

public String registerUser(@ModelAttribute User user,

@RequestParam("publicKeyFile") MultipartFile file) throws Exception {

// Save file

String savePath = "src/main/resources/static/publickeys/" + file.getOriginalFilename();

Path path = Paths.get(savePath);

Files.createDirectories(path.getParent());

file.transferTo(path.toFile());

user.setDscPath("/publickeys/" + file.getOriginalFilename());

// Extract public key from certificate

CertificateFactory cf = CertificateFactory.getInstance("X.509");

Certificate cert = cf.generateCertificate(file.getInputStream());

PublicKey publicKey = cert.getPublicKey();

String publicKeyBase64 = Base64.getEncoder().encodeToString(publicKey.getEncoded());

user.setPublicKey(publicKeyBase64);

userRepository.save(user);

return "redirect:/userdashboard";

}

@GetMapping("/download-dsc/{filename:.+}")

@ResponseBody

public ResponseEntity<Resource> downloadDSC(@PathVariable String filename) throws IOException {

Path filePath = Paths.get("src/main/resources/static/dscs").resolve(filename).normalize();

Resource resource = new UrlResource(filePath.toUri());

if (!resource.exists()) {

return ResponseEntity.notFound().build();

}

return ResponseEntity.ok()

.header(HttpHeaders.CONTENT\_DISPOSITION, "attachment; filename=\"" + resource.getFilename() + "\"")

.body(resource);

}

@PostMapping("/userlogin")

public String handleUserLogin(@RequestParam String fullName,

@RequestParam String password,

Model model,

HttpSession session) {

User user = userRepository.findByFullName(fullName);

if (user == null || !user.getPassword().equals(password)) {

model.addAttribute("error", "Invalid credentials");

return "userlogin";

}

session.setAttribute("loggedInUser", user);

String authMode = user.getAuthMode();

if ("OTP".equalsIgnoreCase(authMode)) {

return "redirect:/user/send-otp";

} else if ("DSC".equalsIgnoreCase(authMode)) {

return "redirect:/verify-dsc";

} else {

model.addAttribute("user", user);

return "userdashboard";

}

}

@GetMapping("/user/send-otp")

public String sendOtpToUser(HttpSession session, Model model) {

User user = (User) session.getAttribute("loggedInUser");

if (user == null) return "redirect:/userlogin";

String otp = String.valueOf(new Random().nextInt(900000) + 100000);

session.setAttribute("userOtp", otp);

try {

SimpleMailMessage message = new SimpleMailMessage();

message.setTo(user.getEmail());

message.setSubject("Your OTP for Login");

message.setText("Your OTP is: " + otp);

message.setFrom("youremail@gmail.com"); // must match configured email

mailSender.send(message);

model.addAttribute("info", "OTP sent to your email.");

} catch (Exception e) {

model.addAttribute("error", "Failed to send OTP: " + e.getMessage());

return "userlogin";

}

return "user-otp";

}

@PostMapping("/user/verify-otp")

public String verifyUserOtp(@RequestParam String otp,

HttpSession session,

Model model) {

String sessionOtp = (String) session.getAttribute("userOtp");

User user = (User) session.getAttribute("loggedInUser");

if (user == null) return "redirect:/userlogin";

if (sessionOtp != null && sessionOtp.equals(otp)) {

model.addAttribute("user", user);

return "userdashboard";

} else {

model.addAttribute("error", "Invalid OTP. Try again.");

return "user-otp";

}

}

@GetMapping("/verify-dsc")

public String showDscUploadPage(HttpSession session) {

User user = (User) session.getAttribute("loggedInUser");

if (user == null) return "redirect:/userlogin";

return "verify-dsc";

}

@PostMapping("/verify-dsc")

public String verifyDsc(@RequestParam MultipartFile dscFile,

@RequestParam String keyPassword,

HttpSession session,

Model model) throws Exception {

User user = (User) session.getAttribute("loggedInUser");

if (user == null) return "redirect:/userlogin";

// Save uploaded DSC temporarily

Path tempFile = Files.createTempFile("uploaded-", ".jks");

dscFile.transferTo(tempFile.toFile());

boolean match = false;

try (FileInputStream fis = new FileInputStream(tempFile.toFile())) {

KeyStore ks = KeyStore.getInstance("JKS");

ks.load(fis, keyPassword.toCharArray());

String alias = ks.aliases().nextElement();

Certificate cert = ks.getCertificate(alias);

PublicKey uploadedKey = cert.getPublicKey();

String uploadedBase64 = Base64.getEncoder().encodeToString(uploadedKey.getEncoded());

match = uploadedBase64.equals(user.getPublicKey());

} catch (Exception e) {

model.addAttribute("error", "Invalid or corrupt keystore.");

return "verify-dsc";

} finally {

// Ensure temp file is deleted

Files.deleteIfExists(tempFile);

}

if (match) {

model.addAttribute("user", user);

return "userdashboard";

} else {

model.addAttribute("error", "DSC public key does not match.");

return "verify-dsc";

}

}

}

OtpLogin.java:

**package** com.example.demo.dto;

**public** **class** OtpLogin {

**private** String username;

**private** String password;

**private** String email;

**private** String otp;

// 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;

}

**public** String getEmail() {

**return** email;

}

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

**this**.email = email;

}

**public** String getOtp() {

**return** otp;

}

**public** **void** setOtp(String otp) {

**this**.otp = otp;

}

}

Admin.java:  
**package** com.example.demo.model;

**import** jakarta.persistence.\*;

@Entity

@Table(name = "admin")

**public** **class** Admin {

@Id

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

**private** Long id;

@Column(name = "username", nullable = **false**, unique = **true**)

**private** String username;

@Column(name = "password", nullable = **false**)

**private** String password;

@Column(name = "email", nullable = **false**, unique = **true**)

**private** String email;

// Getters and Setters

**public** Long getId() {

**return** id;

}

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

**this**.id = id;

}

**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;

}

**public** String getEmail() {

**return** email;

}

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

**this**.email = email;

}

@Override

**public** String toString() {

**return** "Admin{" +

"id=" + id +

", username='" + username + '\'' +

", email='" + email + '\'' +

'}';

}

}

User.java:  
**package** com.example.demo.model;

**import** jakarta.persistence.\*;

@Entity

@Table(name = "user")

**public** **class** User {

@Id

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

@Column(name = "id")

**private** Long id;

@Column(name = "full\_name", nullable = **false**)

**private** String fullName;

@Column(name = "email")

**private** String email;

@Column(name = "phone")

**private** String phone;

@Column(name = "organization")

**private** String organization;

@Column(name = "organization\_unit")

**private** String organizationUnit;

@Column(name = "country")

**private** String country;

@Column(name = "password")

**private** String password;

@Column(name = "public\_key", columnDefinition = "TEXT")

**private** String publicKey; // Base64-encoded public key string

@Column(name = "dsc\_path")

**private** String dscPath; // Path to uploaded public key file (optional for audit)

@Column(name = "auth\_mode")

**private** String authMode = "OTP"; // default

// Getters and setters

**public** Long getId() {

**return** id;

}

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

**this**.id = id;

}

**public** String getFullName() {

**return** fullName;

}

**public** **void** setFullName(String fullName) {

**this**.fullName = fullName;

}

**public** String getEmail() {

**return** email;

}

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

**this**.email = email;

}

**public** String getPhone() {

**return** phone;

}

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

**this**.phone = phone;

}

**public** String getOrganization() {

**return** organization;

}

**public** **void** setOrganization(String organization) {

**this**.organization = organization;

}

**public** String getPassword() {

**return** password;

}

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

**this**.password = password;

}

**public** String getDscPath() {

**return** dscPath;

}

**public** **void** setDscPath(String dscPath) {

**this**.dscPath = dscPath;

}

**public** String getAuthMode() {

**return** authMode;

}

**public** **void** setAuthMode(String authMode) {

**this**.authMode = authMode;

}

**public** String getOrganizationUnit() {

**return** organizationUnit;

}

**public** **void** setOrganizationUnit(String organizationUnit) {

**this**.organizationUnit = organizationUnit;

}

**public** String getCountry() {

**return** country;

}

**public** **void** setCountry(String country) {

**this**.country = country;

}

**public** String getPublicKey() {

**return** publicKey;

}

**public** **void** setPublicKey(String publicKey) {

**this**.publicKey = publicKey;

}

}

AdminRepository.java:

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

**import com.example.demo.model.Admin;**

**import org.springframework.data.repository.CrudRepository;**

**public interface AdminRepository extends CrudRepository<Admin, Long> {**

**Admin findByUsername(String username);**

**Admin findByEmail(String email);**

**}**

UserRepository.java:

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

**import java.util.Optional;**

**import org.springframework.data.repository.CrudRepository;**

**import com.example.demo.model.User;**

**public interface UserRepository extends CrudRepository<User, Long> {**

**User findByEmail(String email);**

**Optional<User> findById(Long id);**

**User findByFullName(String fullName);**

**}**

**EmailService.java:**

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

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

**import org.springframework.mail.SimpleMailMessage;**

**import org.springframework.mail.javamail.JavaMailSender;**

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

**@Service**

**public class EmailService {**

**@Autowired**

**private JavaMailSender mailSender;**

**public void sendOtp(String to, String otp) {**

**SimpleMailMessage message = new SimpleMailMessage();**

**message.setTo(to);**

**message.setSubject("Your OTP Code");**

**message.setText("Your OTP is: " + otp);**

**mailSender.send(message);**

**}**

**}**

**KeyToolUtil.java:**

**package com.example.demo.util;**

**import java.io.File;**

**import java.io.InputStream;**

**import java.util.concurrent.TimeUnit;**

**public class KeyToolUtil {**

**public static String generateDSC(String alias, String password, String dname, String outputDir) throws Exception {**

**String filename = alias + ".jks";**

**File dir = new File(outputDir);**

**if (!dir.exists()) dir.mkdirs();**

**String keystorePath = outputDir + "/" + filename;**

**String keytoolPath = "\"C:\\Program Files\\Java\\jdk-17\\bin\\keytool.exe\"";**

**String command = String.format(**

**"%s -genkeypair -alias \"%s\" -keyalg RSA -keystore \"%s\" -storepass \"%s\" -keypass \"%s\" -validity 365 -keysize 2048 -dname \"%s\" -storetype JKS -noprompt",**

**keytoolPath, alias, keystorePath, password, password, dname**

**);**

**System.out.println("Running keytool command: " + command);**

**Process process = Runtime.getRuntime().exec(command);**

**boolean finished = process.waitFor(5, TimeUnit.SECONDS);**

**if (!finished) {**

**throw new RuntimeException("Keytool process timed out");**

**}**

**InputStream errorStream = process.getErrorStream();**

**String errorOutput = new String(errorStream.readAllBytes());**

**System.err.println("Keytool error: " + errorOutput);**

**return keystorePath;**

**}**

**}**

**home.html:**

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<title>Home</title>

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

<style>

body {

background-color: #f8f9fa;

}

.home-container {

margin-top: 100px;

max-width: 500px;

margin-left: auto;

margin-right: auto;

text-align: center;

padding: 40px;

background: #ffffff;

border-radius: 10px;

box-shadow: 0px 0px 15px rgba(0,0,0,0.1);

}

.home-container h1 {

margin-bottom: 30px;

}

.home-container a {

margin: 10px;

}

</style>

</head>

<body>

<div class="home-container">

<h1>Welcome</h1>

<a href="/register" class="btn btn-primary">Register</a>

<a href="/userlogin" class="btn btn-success">User Login</a>

<a href="/login" class="btn btn-info">Admin Login</a>

</div>

</body>

</html>

login.html:  
<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<title>Admin OTP Login</title>

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

</head>

<body>

<div class="container mt-5">

<h2>Admin OTP Login</h2>

<!-- Step 1: Initial login form -->

<form th:action="@{/send-otp}" th:object="${otpLogin}" method="post" th:if="${otpSent == null}">

<div class="mb-3">

<label>Username:</label>

<input type="text" th:field="\*{username}" class="form-control" required />

</div>

<div class="mb-3">

<label>Password:</label>

<input type="password" th:field="\*{password}" class="form-control" required />

</div>

<div class="mb-3">

<label>Email:</label>

<input type="email" th:field="\*{email}" class="form-control" required />

</div>

<div class="mb-3 text-danger" th:if="${error}">

<p th:text="${error}"></p>

</div>

<button class="btn btn-primary">Generate OTP</button>

</form>

<!-- Step 2: OTP verification form -->

<form th:action="@{/verify-otp}" th:object="${otpLogin}" method="post" th:if="${otpSent}">

<input type="hidden" th:field="\*{username}" />

<input type="hidden" th:field="\*{password}" />

<input type="hidden" th:field="\*{email}" />

<div class="mb-3">

<label>Enter OTP:</label>

<input type="text" th:field="\*{otp}" class="form-control" required />

</div>

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

Your OTP is: <span th:text="${generatedOtp}"></span>

</div> -->

<div class="text-danger" th:if="${error}">

<p th:text="${error}"></p>

</div>

<button class="btn btn-success">Verify OTP</button>

</form>

<!-- Step 3: Regenerate OTP button -->

<form th:action="@{/send-otp}" th:object="${otpLogin}" method="post" th:if="${otpSent}">

<input type="hidden" th:field="\*{username}" />

<input type="hidden" th:field="\*{password}" />

<input type="hidden" th:field="\*{email}" />

<button class="btn btn-warning mt-3">Regenerate OTP</button>

</form>

</div>

</body>

</html><!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<title>Admin OTP Login</title>

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

</head>

<body>

<div class="container mt-5">

<h2>Admin OTP Login</h2>

<!-- Step 1: Initial login form -->

<form th:action="@{/send-otp}" th:object="${otpLogin}" method="post" th:if="${otpSent == null}">

<div class="mb-3">

<label>Username:</label>

<input type="text" th:field="\*{username}" class="form-control" required />

</div>

<div class="mb-3">

<label>Password:</label>

<input type="password" th:field="\*{password}" class="form-control" required />

</div>

<div class="mb-3">

<label>Email:</label>

<input type="email" th:field="\*{email}" class="form-control" required />

</div>

<div class="mb-3 text-danger" th:if="${error}">

<p th:text="${error}"></p>

</div>

<button class="btn btn-primary">Generate OTP</button>

</form>

<!-- Step 2: OTP verification form -->

<form th:action="@{/verify-otp}" th:object="${otpLogin}" method="post" th:if="${otpSent}">

<input type="hidden" th:field="\*{username}" />

<input type="hidden" th:field="\*{password}" />

<input type="hidden" th:field="\*{email}" />

<div class="mb-3">

<label>Enter OTP:</label>

<input type="text" th:field="\*{otp}" class="form-control" required />

</div>

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

Your OTP is: <span th:text="${generatedOtp}"></span>

</div> -->

<div class="text-danger" th:if="${error}">

<p th:text="${error}"></p>

</div>

<button class="btn btn-success">Verify OTP</button>

</form>

<!-- Step 3: Regenerate OTP button -->

<form th:action="@{/send-otp}" th:object="${otpLogin}" method="post" th:if="${otpSent}">

<input type="hidden" th:field="\*{username}" />

<input type="hidden" th:field="\*{password}" />

<input type="hidden" th:field="\*{email}" />

<button class="btn btn-warning mt-3">Regenerate OTP</button>

</form>

</div>

</body>

</html>

register.html:

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<meta charset="UTF-8">

<title>Registration Portal</title>

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

<style>

body {

background-color: #f8f9fa;

}

.header {

background-color: #003366;

color: white;

padding: 20px;

text-align: center;

border-radius: 0 0 10px 10px;

}

.container {

margin-top: 40px;

max-width: 600px;

background-color: white;

padding: 30px;

border-radius: 10px;

box-shadow: 0 0 10px rgba(0,0,0,0.1);

}

.form-label {

font-weight: 500;

}

</style>

</head>

<body>

<div class="header">

<h2>Registration Portal</h2>

</div>

<div class="container">

<form th:action="@{/register}" method="post" enctype="multipart/form-data" th:object="${user}">

<div class="mb-3">

<label for="fullName" class="form-label">Full Name</label>

<input type="text" class="form-control" id="fullName" th:field="\*{fullName}" required>

</div>

<div class="mb-3">

<label for="email" class="form-label">Email address</label>

<input type="email" class="form-control" id="email" th:field="\*{email}" required>

</div>

<div class="mb-3">

<label for="phone" class="form-label">Phone Number</label>

<input type="text" class="form-control" id="phone" th:field="\*{phone}" required>

</div>

<div class="mb-3">

<label for="organization" class="form-label">Organization</label>

<input type="text" class="form-control" id="organization" th:field="\*{organization}" required>

</div>

<div class="mb-3">

<label for="organizationUnit" class="form-label">Organization Unit</label>

<input type="text" class="form-control" id="organizationUnit" th:field="\*{organizationUnit}" required>

</div>

<div class="mb-3">

<label for="country" class="form-label">Country Code (e.g., IN)</label>

<input type="text" class="form-control" id="country" th:field="\*{country}" required>

</div>

<div class="mb-3">

<label for="password" class="form-label">Account Password</label>

<input type="password" class="form-control" id="password" th:field="\*{password}" required>

</div>

<div class="mb-3">

<label for="publicKeyFile" class="form-label">Upload Public Key File (.jks)</label>

<input type="file" class="form-control" id="publicKeyFile" name="publicKeyFile" accept=".jks" required>

</div>

<div class="d-grid gap-2">

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

</div>

</form>

</div>

</body>

</html>

success.html:

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<title>Success</title>

<style>

body {

font-family: Arial, sans-serif;

background: #e2f0d9;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

}

.message-box {

background: #fff;

padding: 30px;

border-radius: 8px;

box-shadow: 0 2px 10px rgba(0,0,0,0.1);

text-align: center;

}

a {

text-decoration: none;

color: #007bff;

display: inline-block;

margin-top: 20px;

}

</style>

</head>

<body>

<div class="message-box">

<h2>Registration successful!</h2>

<a th:href="@{/register}">Register another user</a>

</div>

</body>

</html>

userdashboard.html:

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<title>User Dashboard</title>

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

</head>

<body>

<div class="container mt-5">

<h2>Welcome, <span th:text="${user.fullName}"></span></h2>

<p><strong>Email:</strong> <span th:text="${user.email}"></span></p>

<p><strong>Phone:</strong> <span th:text="${user.phone}"></span></p>

<p><strong>Organization:</strong> <span th:text="${user.organization}"></span></p>

<p><strong>Organization Unit:</strong> <span th:text="${user.organizationUnit}"></span></p>

<p><strong>Country:</strong> <span th:text="${user.country}"></span></p>

<!-- <div th:if="${user.dscPath}">

<a th:href="@{'/dscs/' + ${#strings.substring(user.dscPath, user.dscPath.lastIndexOf('/') + 1)}}" class="btn btn-success">Download DSC</a>

</div> -->

<a th:href="@{'/download-dsc/' + ${user.fullName.replaceAll(' ', '\_')} + '.jks'}" class="btn btn-success">Download DSC</a>

<a href="/home" class="btn btn-secondary mt-3">Back to Home</a>

</div>

</body>

</html>

userlist.html:

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<title>All Users</title>

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

</head>

<body>

<div class="container mt-5">

<h2>Registered Users</h2>

<a th:href="@{/logout}" class="btn btn-secondary mb-3">Logout</a>

<form th:action="@{/update-auth}" method="post">

<table class="table table-bordered">

<thead>

<tr>

<th>Full Name</th>

<th>Phone</th>

<th>Email</th>

<th>Organization</th>

<th>Organization Unit</th>

<th>Country</th>

<th>Auth Mode</th>

</tr>

</thead>

<tbody>

<tr th:each="user : ${users}">

<td th:text="${user.fullName}"></td>

<td th:text="${user.phone}"></td>

<td th:text="${user.email}"></td>

<td th:text="${user.organization}"></td>

<td th:text="${user.organizationUnit}"></td>

<td th:text="${user.country}"></td>

<td>

<input type="hidden" name="userIds" th:value="${user.id}" />

<label><input type="radio" th:name="'authMode\_' + ${user.id}" value="OTP"

th:checked="${user.authMode == 'OTP'}" /> OTP</label>

<label><input type="radio" th:name="'authMode\_' + ${user.id}" value="DSC"

th:checked="${user.authMode == 'DSC'}" /> DSC</label>

<label><input type="radio" th:name="'authMode\_' + ${user.id}" value="NONE"

th:checked="${user.authMode == 'NONE'}" /> None</label>

</td>

</tr>

</tbody>

</table>

<button type="submit" class="btn btn-primary">Update Authentication Modes</button>

</form>

<div th:if="${success}" class="alert alert-success mt-3">

Authentication modes updated successfully!

</div>

<script>

**function** toggleExclusive(clicked) {

**const** rowIndex = clicked.id.split('-')[1];

**const** otherType = clicked.id.startsWith("otp") ? "dsc" : "otp";

**const** other = document.getElementById(otherType + "-" + rowIndex);

if (clicked.checked) {

other.checked = false;

}

}

</script>

</div>

</body>

</html>

userlogin.html:

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<title>User Login</title>

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

</head>

<body>

<div class="container mt-5">

<h2>User Login</h2>

<form th:action="@{/userlogin}" method="post" autocomplete="off">

<div class="mb-3">

<label>Full Name:</label>

<input type="text" name="fullName" class="form-control" required autocomplete="off" />

</div>

<div class="mb-3">

<label>Password:</label>

<input type="password" name="password" class="form-control" required autocomplete="new-password" />

</div>

<div th:if="${error}" class="text-danger">

<p th:text="${error}"></p>

</div>

<button class="btn btn-primary">Login</button>

</form>

</div>

</body>

</html>

user-otp.html:

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<title>Verify OTP</title>

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

</head>

<body>

<div class="container mt-5">

<h3>User OTP Verification</h3>

<form th:action="@{/user/verify-otp}" method="post">

<label>Enter OTP sent to your email:</label>

<input type="text" name="otp" class="form-control mb-3" required />

<div th:if="${error}" class="text-danger" th:text="${error}"></div>

<div th:if="${info}" class="text-success" th:text="${info}"></div>

<button class="btn btn-primary">Verify</button>

</form>

</div>

</body>

</html>

verify-dsc.html:

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<title>Verify DSC</title>

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

</head>

<body>

<div class="container mt-5">

<h3>Upload Your DSC (.jks)</h3>

<form action="/verify-dsc" method="POST" enctype="multipart/form-data">

<label>Upload Your DSC (.jks)</label>

<input type="file" name="dscFile" accept=".jks" required>

<label>Key Password</label>

<input type="password" name="keyPassword" required>

<button type="submit">Verify</button>

</form>

</div>

</body>

</html>

application.properties:

spring.datasource.url=jdbc:mysql://localhost:3306/registration6?useSSL=false

spring.datasource.username=root

spring.datasource.password=Welcome@123

spring.jpa.hibernate.ddl-auto=update

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

spring.jpa.show-sql=true

spring.thymeleaf.cache=false

spring.thymeleaf.prefix=classpath:/templates/

server.port=8081

# Gmail SMTP settings

spring.mail.host=smtp.gmail.com

spring.mail.port=587

spring.mail.username=lipikasmtp@gmail.com

spring.mail.password=kdtoawrcwaeunwyz

spring.mail.properties.mail.smtp.auth=true

spring.mail.properties.mail.smtp.starttls.enable=true

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>3.4.5</version>

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

</parent>

<groupId>com.example</groupId>

<artifactId>springemployee\_project</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>springemployee\_project</name>

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

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

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

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<!-- Thymeleaf for HTML templates -->

<dependency>

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

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

<scope>test</scope>

</dependency>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>