Experiment No.8

Date:05.05.25

Aim: Demonstrate implementation of real world problem based on Spring Frameworks

CO Mapping – CO 4

Objective:

1. To develop program based on Spring framework

Code:

Application.properties

spring.application.name=cms  
spring.datasource.url=jdbc:mysql://127.0.0.1:3306/cms?createDatabaseIfNotExist=true  
spring.datasource.username=root  
spring.datasource.password=Athanee@08  
spring.jpa.hibernate.ddl-auto=update  
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect  
api\_doc\_url=http://api.localhost.com  
  
# Server Configuration  
server.port=8080  
  
# JSP Configuration  
spring.mvc.view.prefix=/WEB-INF/jsp/  
spring.mvc.view.suffix=.jsp  
spring.web.resources.static-locations=classpath:/META-INF/resources/,classpath:/resources/,classpath:/static/,classpath:/public/  
  
# Enable logging for Spring MVC  
logging.level.org.springframework.web=DEBUG  
  
# Enable bean definition overriding  
spring.main.allow-bean-definition-overriding=true

cmsApplication.java  
package com.spring.cms;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.boot.builder.SpringApplicationBuilder;  
import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;  
  
@SpringBootApplication  
public class CmsApplication extends SpringBootServletInitializer {  
  
 @Override  
 protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {  
 return application.sources(CmsApplication.class);  
 }  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(CmsApplication.class, args);  
 }  
}

user.java  
package com.spring.cms.model;

import jakarta.persistence.\*;

import jakarta.validation.constraints.Email;

import jakarta.validation.constraints.NotBlank;

import jakarta.validation.constraints.Size;

@Entity

@Table(name = "users")

public class User {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    @NotBlank(message = "Username is required")

    @Size(min = 3, max = 20, message = "Username must be between 3 and 20 characters")

    @Column(unique = true)

    private String username;

    @NotBlank(message = "Email is required")

    @Email(message = "Please provide a valid email")

    @Column(unique = true)

    private String email;

    @NotBlank(message = "Password is required")

    @Size(min = 6, message = "Password must be at least 6 characters")

    private String password;

*// 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 getEmail() {

        return email;

    }

    public void setEmail(String *email*) {

        this.email = email;

    }

    public String getPassword() {

        return password;

    }

    public void setPassword(String *password*) {

        this.password = password;

    }

}

usercontroller.java  
package com.spring.cms.controller;

import com.spring.cms.model.User;

import com.spring.cms.service.UserService;

import jakarta.servlet.http.HttpSession;

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

import org.springframework.http.ResponseEntity;

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

import java.util.HashMap;

import java.util.Map;

@RestController

@RequestMapping("/api/users")

public class UserController {

    @Autowired

    private UserService userService;

    @PostMapping("/register")

    public ResponseEntity<?> registerUser(@RequestBody User *user*) {

        try {

            User registeredUser = userService.registerUser(user);

            return ResponseEntity.ok(registeredUser);

        } catch (RuntimeException *e*) {

            return ResponseEntity.badRequest().body(e.getMessage());

        }

    }

    @PostMapping("/login")

    public ResponseEntity<?> loginUser(@RequestParam String *username*, @RequestParam String *password*, HttpSession *session*) {

        try {

            User user = userService.loginUser(username, password);

*// Store user info in session*

            session.setAttribute("user", user);

*// Create a response object with user data (excluding password)*

            Map<String, Object> response = new HashMap<>();

            response.put("id", user.getId());

            response.put("username", user.getUsername());

            response.put("email", user.getEmail());

            response.put("message", "Login successful");

            return ResponseEntity.ok(response);

        } catch (RuntimeException *e*) {

            Map<String, String> errorResponse = new HashMap<>();

            errorResponse.put("error", e.getMessage());

            return ResponseEntity.badRequest().body(errorResponse);

        }

    }

    @GetMapping("/logout")

    public ResponseEntity<?> logout(HttpSession *session*) {

        session.invalidate();

        return ResponseEntity.ok("Logged out successfully");

    }

}

userservices.java  
package com.spring.cms.service;

import com.spring.cms.model.User;

import com.spring.cms.repository.UserRepository;

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

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.stereotype.Service;

import java.util.Optional;

@Service

public class UserService {

    @Autowired

    private UserRepository userRepository;

    @Autowired

    private PasswordEncoder passwordEncoder;

    public User registerUser(User *user*) {

        if (userRepository.existsByUsername(user.getUsername())) {

            throw new RuntimeException("Username already exists");

        }

        if (userRepository.existsByEmail(user.getEmail())) {

            throw new RuntimeException("Email already exists");

        }

*// Encode the password before saving*

        user.setPassword(passwordEncoder.encode(user.getPassword()));

        return userRepository.save(user);

    }

    public User loginUser(String *username*, String *password*) {

        System.out.println("Attempting login with username/email: " + username);

*// Try to find user by username first*

        Optional<User> userOptional = userRepository.findByUsername(username);

        System.out.println("Found by username: " + userOptional.isPresent());

*// If not found by username, try email*

        if (!userOptional.isPresent()) {

            userOptional = userRepository.findByEmail(username);

            System.out.println("Found by email: " + userOptional.isPresent());

        }

        if (userOptional.isPresent()) {

            User user = userOptional.get();

            System.out.println("User found: " + user.getUsername());

            boolean passwordMatches = passwordEncoder.matches(password, user.getPassword());

            System.out.println("Password matches: " + passwordMatches);

            if (passwordMatches) {

                return user;

            }

        }

        throw new RuntimeException("Invalid username/email or password");

    }

}

userrepository.java  
package com.spring.cms.repository;

import com.spring.cms.model.User;

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

import java.util.Optional;

public interface UserRepository extends JpaRepository<User, Long> {

    Optional<User> findByUsername(String *username*);

    Optional<User> findByEmail(String *email*);

    boolean existsByUsername(String *username*);

    boolean existsByEmail(String *email*);

}

index.jsp  
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>${title}</title>

    <link rel="stylesheet" href="<c:url value='/css/styles.css'/>">

    <style>

        body {

            background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);

            min-height: 100vh;

            display: flex;

            justify-content: center;

            align-items: center;

            margin: 0;

            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

        }

        .container {

            width: 100%;

            max-width: 400px;

            padding: 20px;

        }

        .header {

            text-align: center;

            margin-bottom: 20px;

            color: white;

        }

        .form-container {

            background: white;

            padding: 30px;

            border-radius: 10px;

            box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);

        }

    </style>

</head>

<body>

    <div class="container">

        <div class="header">

            <h1>Job Portal</h1>

            <p>Find your dream job or post new opportunities</p>

        </div>

        <div class="form-container">

            <div class="tabs">

                <button class="tab-btn active" data-tab="login">Login</button>

                <button class="tab-btn" data-tab="register">Register</button>

            </div>

            <div id="login-form" class="form active">

                <h2>Login</h2>

                <form id="loginForm">

                    <div class="form-group">

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

                        <input type="text" id="login-username" name="username" required>

                    </div>

                    <div class="form-group">

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

                        <input type="password" id="login-password" name="password" required>

                    </div>

                    <button type="submit" class="btn">Login</button>

                </form>

            </div>

            <div id="register-form" class="form">

                <h2>Register</h2>

                <form id="registerForm">

                    <div class="form-group">

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

                        <input type="text" id="register-username" name="username" required>

                    </div>

                    <div class="form-group">

                        <label for="register-email">Email</label>

                        <input type="email" id="register-email" name="email" required>

                    </div>

                    <div class="form-group">

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

                        <input type="password" id="register-password" name="password" required>

                    </div>

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

                </form>

            </div>

        </div>

        <div id="message" class="message"></div>

    </div>

    <script src="<c:url value='/js/script.js'/>"></script>

</body>

</html>

job.java  
package com.spring.cms.model;

import jakarta.persistence.\*;

import jakarta.validation.constraints.NotBlank;

import jakarta.validation.constraints.Size;

import java.time.LocalDate;

import java.time.LocalDateTime;

@Entity

@Table(name = "jobs")

public class Job {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    @NotBlank(message = "Job title is required")

    @Size(min = 3, max = 100, message = "Job title must be between 3 and 100 characters")

    private String title;

    @NotBlank(message = "Job description is required")

    @Column(columnDefinition = "TEXT")

    private String description;

    @NotBlank(message = "Job requirements are required")

    @Column(columnDefinition = "TEXT")

    private String requirements;

    private String salary;

    private String location;

    private String companyName;

    private String employmentType; *// Full-time, Part-time, Contract, etc.*

    private LocalDateTime postedDate;

    private LocalDate expiryDate;

*// Default constructor*

    public Job() {

        this.postedDate = LocalDateTime.now();

    }

*// Getters and Setters*

    public Long getId() {

        return id;

    }

    public void setId(Long *id*) {

        this.id = id;

    }

    public String getTitle() {

        return title;

    }

    public void setTitle(String *title*) {

        this.title = title;

    }

    public String getDescription() {

        return description;

    }

    public void setDescription(String *description*) {

        this.description = description;

    }

    public String getRequirements() {

        return requirements;

    }

    public void setRequirements(String *requirements*) {

        this.requirements = requirements;

    }

    public String getSalary() {

        return salary;

    }

    public void setSalary(String *salary*) {

        this.salary = salary;

    }

    public String getLocation() {

        return location;

    }

    public void setLocation(String *location*) {

        this.location = location;

    }

    public String getCompanyName() {

        return companyName;

    }

    public void setCompanyName(String *companyName*) {

        this.companyName = companyName;

    }

    public String getEmploymentType() {

        return employmentType;

    }

    public void setEmploymentType(String *employmentType*) {

        this.employmentType = employmentType;

    }

    public LocalDateTime getPostedDate() {

        return postedDate;

    }

    public void setPostedDate(LocalDateTime *postedDate*) {

        this.postedDate = postedDate;

    }

    public LocalDate getExpiryDate() {

        return expiryDate;

    }

    public void setExpiryDate(LocalDate *expiryDate*) {

        this.expiryDate = expiryDate;

    }

*// Helper method for JSP display*

    public String getFormattedPostedDate() {

        if (postedDate == null) {

            return "";

        }

        return postedDate.getMonth() + " " + postedDate.getDayOfMonth() + ", " + postedDate.getYear();

    }

}

jobcontroller.java  
package com.spring.cms.controller;

import com.spring.cms.model.Job;

import com.spring.cms.service.JobService;

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

import org.springframework.http.ResponseEntity;

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

import java.util.List;

import java.util.Optional;

@RestController

@RequestMapping("/api/jobs")

public class JobController {

    @Autowired

    private JobService jobService;

*// Create a new job*

    @PostMapping

    public ResponseEntity<?> createJob(@RequestBody Job *job*) {

        try {

            Job createdJob = jobService.createJob(job);

            return ResponseEntity.ok(createdJob);

        } catch (Exception *e*) {

            return ResponseEntity.badRequest().body(e.getMessage());

        }

    }

*// Get all jobs*

    @GetMapping

    public ResponseEntity<List<Job>> getAllJobs() {

        List<Job> jobs = jobService.getAllJobs();

        return ResponseEntity.ok(jobs);

    }

*// Get job by ID*

    @GetMapping("/{id}")

    public ResponseEntity<?> getJobById(@PathVariable Long *id*) {

        Optional<Job> job = jobService.getJobById(id);

        if (job.isPresent()) {

            return ResponseEntity.ok(job.get());

        } else {

            return ResponseEntity.notFound().build();

        }

    }

*// Update job*

    @PutMapping("/{id}")

    public ResponseEntity<?> updateJob(@PathVariable Long *id*, @RequestBody Job *job*) {

        try {

            job.setId(id); *// Ensure the ID is set*

            Job updatedJob = jobService.updateJob(job);

            return ResponseEntity.ok(updatedJob);

        } catch (RuntimeException *e*) {

            return ResponseEntity.badRequest().body(e.getMessage());

        }

    }

*// Delete job*

    @DeleteMapping("/{id}")

    public ResponseEntity<?> deleteJob(@PathVariable Long *id*) {

        try {

            jobService.deleteJob(id);

            return ResponseEntity.ok("Job deleted successfully");

        } catch (RuntimeException *e*) {

            return ResponseEntity.badRequest().body(e.getMessage());

        }

    }

*// Search jobs by title*

    @GetMapping("/search/title")

    public ResponseEntity<List<Job>> searchByTitle(@RequestParam String *title*) {

        return ResponseEntity.ok(jobService.searchByTitle(title));

    }

*// Search jobs by location*

    @GetMapping("/search/location")

    public ResponseEntity<List<Job>> searchByLocation(@RequestParam String *location*) {

        return ResponseEntity.ok(jobService.searchByLocation(location));

    }

*// Search jobs by company name*

    @GetMapping("/search/company")

    public ResponseEntity<List<Job>> searchByCompanyName(@RequestParam String *companyName*) {

        return ResponseEntity.ok(jobService.searchByCompanyName(companyName));

    }

*// Filter jobs by employment type*

    @GetMapping("/filter/type")

    public ResponseEntity<List<Job>> filterByEmploymentType(@RequestParam String *type*) {

        return ResponseEntity.ok(jobService.filterByEmploymentType(type));

    }

}

jobservice.java  
package com.spring.cms.service;

import com.spring.cms.model.Job;

import com.spring.cms.repository.JobRepository;

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

import org.springframework.stereotype.Service;

import java.time.LocalDateTime;

import java.util.List;

import java.util.Optional;

@Service

public class JobService {

    @Autowired

    private JobRepository jobRepository;

*// Create a new job*

    public Job createJob(Job *job*) {

        if (job.getPostedDate() == null) {

            job.setPostedDate(LocalDateTime.now());

        }

        return jobRepository.save(job);

    }

*// Get all jobs*

    public List<Job> getAllJobs() {

        return jobRepository.findAll();

    }

*// Get job by ID*

    public Optional<Job> getJobById(Long *id*) {

        return jobRepository.findById(id);

    }

*// Update job*

    public Job updateJob(Job *job*) {

        if (job.getId() == null || !jobRepository.existsById(job.getId())) {

            throw new RuntimeException("Job not found");

        }

        return jobRepository.save(job);

    }

*// Delete job*

    public void deleteJob(Long *id*) {

        if (!jobRepository.existsById(id)) {

            throw new RuntimeException("Job not found");

        }

        jobRepository.deleteById(id);

    }

*// Search jobs by title*

    public List<Job> searchByTitle(String *title*) {

        return jobRepository.findByTitleContainingIgnoreCase(title);

    }

*// Search jobs by location*

    public List<Job> searchByLocation(String *location*) {

        return jobRepository.findByLocationContainingIgnoreCase(location);

    }

*// Search jobs by company name*

    public List<Job> searchByCompanyName(String *companyName*) {

        return jobRepository.findByCompanyNameContainingIgnoreCase(companyName);

    }

*// Filter jobs by employment type*

    public List<Job> filterByEmploymentType(String *employmentType*) {

        return jobRepository.findByEmploymentType(employmentType);

    }

}

jobrepositroy.java  
package com.spring.cms.repository;

import com.spring.cms.model.Job;

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

import java.util.List;

public interface JobRepository extends JpaRepository<Job, Long> {

    List<Job> findByTitleContainingIgnoreCase(String *title*);

    List<Job> findByLocationContainingIgnoreCase(String *location*);

    List<Job> findByCompanyNameContainingIgnoreCase(String *companyName*);

    List<Job> findByEmploymentType(String *employmentType*);

}

post-job.jsp  
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>${title}</title>

    <link rel="stylesheet" href="<c:url value='/css/styles.css'/>">

    <style>

        body {

            background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);

            min-height: 100vh;

            display: flex;

            justify-content: center;

            align-items: center;

            margin: 0;

            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

        }

        .container {

            width: 100%;

            max-width: 800px;

            padding: 20px;

        }

        .dashboard {

            background: white;

            padding: 30px;

            border-radius: 10px;

            box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);

            margin: 0 auto;

        }

        .welcome-header {

            margin-bottom: 20px;

            text-align: center;

            color: #333;

        }

        .back-link {

            display: inline-block;

            margin-bottom: 20px;

            color: #667eea;

            text-decoration: none;

        }

        .back-link:hover {

            text-decoration: underline;

        }

        .form-group {

            margin-bottom: 15px;

        }

        label {

            display: block;

            margin-bottom: 5px;

            color: #555;

            font-weight: 500;

        }

        input, select, textarea {

            width: 100%;

            padding: 10px;

            border: 1px solid #ddd;

            border-radius: 5px;

            font-size: 15px;

        }

        textarea {

            min-height: 120px;

            resize: vertical;

        }

        .btn {

            background: #667eea;

            color: white;

            border: none;

            border-radius: 5px;

            padding: 10px 20px;

            cursor: pointer;

            font-size: 16px;

            margin-top: 10px;

        }

        .btn:hover {

            background: #764ba2;

        }

        .form-row {

            display: flex;

            gap: 15px;

        }

        .form-row .form-group {

            flex: 1;

        }

    </style>

</head>

<body>

    <div class="container">

        <div class="dashboard">

            <a href="<c:url value='/home'/>" class="back-link">← Back to Dashboard</a>

            <h1 class="welcome-header">Post a New Job</h1>

            <form id="postJobForm">

                <div class="form-group">

                    <label for="job-title">Job Title</label>

                    <input type="text" id="job-title" name="title" required>

                </div>

                <div class="form-row">

                    <div class="form-group">

                        <label for="company-name">Company Name</label>

                        <input type="text" id="company-name" name="companyName" required>

                    </div>

                    <div class="form-group">

                        <label for="employment-type">Employment Type</label>

                        <select id="employment-type" name="employmentType" required>

                            <option value="">Select Type</option>

                            <option value="Full-time">Full-time</option>

                            <option value="Part-time">Part-time</option>

                            <option value="Contract">Contract</option>

                            <option value="Internship">Internship</option>

                            <option value="Freelance">Freelance</option>

                        </select>

                    </div>

                </div>

                <div class="form-row">

                    <div class="form-group">

                        <label for="location">Location</label>

                        <input type="text" id="location" name="location" required>

                    </div>

                    <div class="form-group">

                        <label for="salary">Salary/Compensation</label>

                        <input type="text" id="salary" name="salary" placeholder="e.g. $50,000 - $70,000 per year">

                    </div>

                </div>

                <div class="form-group">

                    <label for="description">Job Description</label>

                    <textarea id="description" name="description" required></textarea>

                </div>

                <div class="form-group">

                    <label for="requirements">Job Requirements</label>

                    <textarea id="requirements" name="requirements" required></textarea>

                </div>

                <div class="form-group">

                    <label for="expiry-date">Application Deadline</label>

                    <input type="date" id="expiry-date" name="expiryDate">

                </div>

                <button type="submit" class="btn">Post Job</button>

            </form>

        </div>

    </div>

    <script>

        document.addEventListener('DOMContentLoaded', () => {

            const postJobForm = document.getElementById('postJobForm');

            postJobForm.addEventListener('submit', async (*e*) => {

*e*.preventDefault();

                const formData = {

                    title: document.getElementById('job-title').value,

                    companyName: document.getElementById('company-name').value,

                    employmentType: document.getElementById('employment-type').value,

                    location: document.getElementById('location').value,

                    salary: document.getElementById('salary').value,

                    description: document.getElementById('description').value,

                    requirements: document.getElementById('requirements').value,

                    expiryDate: document.getElementById('expiry-date').value

                };

                try {

                    const response = await fetch('/api/jobs', {

                        method: 'POST',

                        headers: {

                            'Content-Type': 'application/json'

                        },

                        body: JSON.stringify(formData)

                    });

                    if (response.ok) {

                        alert('Job posted successfully!');

                        window.location.href = '/home';

                    } else {

                        const errorData = await response.json();

                        alert(`Error: ${errorData}`);

                    }

                } catch (error) {

                    alert('An error occurred while posting the job.');

                    console.error(error);

                }

            });

        });

    </script>

</body>

</html>

view-job.jsp  
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>${title}</title>

    <link rel="stylesheet" href="<c:url value='/css/styles.css'/>">

    <style>

        body {

            background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);

            min-height: 100vh;

            display: flex;

            justify-content: center;

            align-items: center;

            margin: 0;

            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

        }

        .container {

            width: 100%;

            max-width: 900px;

            padding: 20px;

        }

        .dashboard {

            background: white;

            padding: 30px;

            border-radius: 10px;

            box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);

            margin: 0 auto;

        }

        .welcome-header {

            margin-bottom: 20px;

            text-align: center;

            color: #333;

        }

        .back-link {

            display: inline-block;

            margin-bottom: 20px;

            color: #667eea;

            text-decoration: none;

        }

        .back-link:hover {

            text-decoration: underline;

        }

        .job-list {

            margin-top: 20px;

        }

        .job-card {

            background: #f9f9f9;

            border-radius: 8px;

            padding: 20px;

            margin-bottom: 15px;

            box-shadow: 0 2px 5px rgba(0,0,0,0.05);

            transition: transform 0.2s, box-shadow 0.2s;

        }

        .job-card:hover {

            transform: translateY(-3px);

            box-shadow: 0 5px 15px rgba(0,0,0,0.1);

        }

        .job-title {

            font-size: 1.2rem;

            color: #333;

            margin-bottom: 5px;

            font-weight: 600;

        }

        .job-company {

            font-weight: 500;

            color: #555;

            margin-bottom: 10px;

        }

        .job-details {

            display: flex;

            flex-wrap: wrap;

            gap: 15px;

            margin-bottom: 15px;

            color: #666;

            font-size: 0.9rem;

        }

        .job-actions {

            display: flex;

            gap: 10px;

        }

        .btn {

            background: #667eea;

            color: white;

            border: none;

            border-radius: 5px;

            padding: 8px 16px;

            cursor: pointer;

            text-decoration: none;

            font-size: 0.9rem;

            display: inline-block;

        }

        .btn-secondary {

            background: #6c757d;

        }

        .btn:hover {

            background: #764ba2;

        }

        .btn-secondary:hover {

            background: #5a6268;

        }

        .empty-state {

            text-align: center;

            padding: 40px 0;

            color: #666;

        }

        .search-bar {

            display: flex;

            gap: 10px;

            margin-bottom: 20px;

        }

        .search-bar input {

            flex: 1;

            padding: 10px;

            border: 1px solid #ddd;

            border-radius: 5px;

            font-size: 15px;

        }

    </style>

</head>

<body>

    <div class="container">

        <div class="dashboard">

            <a href="<c:url value='/home'/>" class="back-link">← Back to Dashboard</a>

            <h1 class="welcome-header">All Job Listings</h1>

            <div class="job-list">

                <c:choose>

                    <c:when test="${empty jobs}">

                        <div class="empty-state">

                            <h3>No jobs found</h3>

                            <p>There are no job listings available at the moment.</p>

                            <a href="<c:url value='/jobs/post'/>" class="btn">Post a New Job</a>

                        </div>

                    </c:when>

                    <c:otherwise>

                        <c:forEach var="job" items="${jobs}">

                            <div class="job-card">

                                <div class="job-title">${job.title}</div>

                                <div class="job-company">${job.companyName}</div>

                                <div class="job-details">

                                    <span>📍 ${job.location}</span>

                                    <span>💼 ${job.employmentType}</span>

                                    <c:if test="${not empty job.salary}">

                                        <span>💰 ${job.salary}</span>

                                    </c:if>

                                    <c:if test="${not empty job.formattedPostedDate}">

                                        <span>📅 Posted: ${job.formattedPostedDate}</span>

                                    </c:if>

                                </div>

                                <div class="job-actions">

                                    <a href="#" class="btn" onclick="viewJobDetails(${job.id})">View Details</a>

                                    <a href="<c:url value='/jobs/update/${job.id}'/>" class="btn">Update</a>

                                    <button class="btn btn-secondary" onclick="deleteJob(${job.id})">Delete</button>

                                </div>

                            </div>

                        </c:forEach>

                    </c:otherwise>

                </c:choose>

            </div>

        </div>

    </div>

    <script>

        document.addEventListener('DOMContentLoaded', () => {

*// Empty initialization function*

        });

        function **viewJobDetails**(*jobId*) {

*// This would typically open a modal or navigate to a details page*

            alert('Viewing details for job #' + *jobId*);

*// For a real implementation:*

*// window.location.href = "/jobs/details/" + jobId;*

        }

        function **deleteJob**(*jobId*) {

            if (confirm('Are you sure you want to delete this job posting?')) {

                fetch("/api/jobs/" + *jobId*, {

                    method: 'DELETE'

                })

                .then(*response* => {

                    if (*response*.ok) {

                        alert('Job deleted successfully');

                        location.reload();

                    } else {

                        alert('Error deleting job');

                    }

                })

                .catch(*error* => {

                    console.error('Error:', *error*);

                    alert('An error occurred while deleting the job');

                });

            }

        }

    </script>

</body>

</html>

view-job.jsp  
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>${title}</title>

    <link rel="stylesheet" href="<c:url value='/css/styles.css'/>">

    <style>

        body {

            background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);

            min-height: 100vh;

            display: flex;

            justify-content: center;

            align-items: center;

            margin: 0;

            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

        }

        .container {

            width: 100%;

            max-width: 800px;

            padding: 20px;

        }

        .dashboard {

            background: white;

            padding: 30px;

            border-radius: 10px;

            box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);

            margin: 0 auto;

        }

        .welcome-header {

            margin-bottom: 20px;

            text-align: center;

            color: #333;

        }

        .back-link {

            display: inline-block;

            margin-bottom: 20px;

            color: #667eea;

            text-decoration: none;

        }

        .back-link:hover {

            text-decoration: underline;

        }

        .form-group {

            margin-bottom: 15px;

        }

        label {

            display: block;

            margin-bottom: 5px;

            color: #555;

            font-weight: 500;

        }

        input, select, textarea {

            width: 100%;

            padding: 10px;

            border: 1px solid #ddd;

            border-radius: 5px;

            font-size: 15px;

        }

        textarea {

            min-height: 120px;

            resize: vertical;

        }

        .btn {

            background: #667eea;

            color: white;

            border: none;

            border-radius: 5px;

            padding: 10px 20px;

            cursor: pointer;

            font-size: 16px;

            margin-top: 10px;

        }

        .btn:hover {

            background: #764ba2;

        }

        .form-row {

            display: flex;

            gap: 15px;

        }

        .form-row .form-group {

            flex: 1;

        }

    </style>

</head>

<body>

    <div class="container">

        <div class="dashboard">

            <a href="<c:url value='/jobs/update'/>" class="back-link">← Back to Job Selection</a>

            <h1 class="welcome-header">Update Job</h1>

            <form id="updateJobForm">

                <input type="hidden" id="job-id" value="${job.id}">

                <div class="form-group">

                    <label for="job-title">Job Title</label>

                    <input type="text" id="job-title" name="title" value="${job.title}" required>

                </div>

                <div class="form-row">

                    <div class="form-group">

                        <label for="company-name">Company Name</label>

                        <input type="text" id="company-name" name="companyName" value="${job.companyName}" required>

                    </div>

                    <div class="form-group">

                        <label for="employment-type">Employment Type</label>

                        <select id="employment-type" name="employmentType" required>

                            <option value="">Select Type</option>

                            <option value="Full-time" ${job.employmentType == 'Full-time' ? 'selected' : ''}>Full-time</option>

                            <option value="Part-time" ${job.employmentType == 'Part-time' ? 'selected' : ''}>Part-time</option>

                            <option value="Contract" ${job.employmentType == 'Contract' ? 'selected' : ''}>Contract</option>

                            <option value="Internship" ${job.employmentType == 'Internship' ? 'selected' : ''}>Internship</option>

                            <option value="Freelance" ${job.employmentType == 'Freelance' ? 'selected' : ''}>Freelance</option>

                        </select>

                    </div>

                </div>

                <div class="form-row">

                    <div class="form-group">

                        <label for="location">Location</label>

                        <input type="text" id="location" name="location" value="${job.location}" required>

                    </div>

                    <div class="form-group">

                        <label for="salary">Salary/Compensation</label>

                        <input type="text" id="salary" name="salary" value="${job.salary}" placeholder="e.g. $50,000 - $70,000 per year">

                    </div>

                </div>

                <div class="form-group">

                    <label for="description">Job Description</label>

                    <textarea id="description" name="description" required>${job.description}</textarea>

                </div>

                <div class="form-group">

                    <label for="requirements">Job Requirements</label>

                    <textarea id="requirements" name="requirements" required>${job.requirements}</textarea>

                </div>

                <div class="form-group">

                    <label for="expiry-date">Application Deadline</label>

                    <input type="date" id="expiry-date" name="expiryDate" value="${job.expiryDate}">

                </div>

                <button type="submit" class="btn">Update Job</button>

            </form>

        </div>

    </div>

    <script>

        document.addEventListener('DOMContentLoaded', () => {

            const updateJobForm = document.getElementById('updateJobForm');

            updateJobForm.addEventListener('submit', async (*e*) => {

*e*.preventDefault();

                const jobId = document.getElementById('job-id').value;

                const formData = {

                    title: document.getElementById('job-title').value,

                    companyName: document.getElementById('company-name').value,

                    employmentType: document.getElementById('employment-type').value,

                    location: document.getElementById('location').value,

                    salary: document.getElementById('salary').value,

                    description: document.getElementById('description').value,

                    requirements: document.getElementById('requirements').value,

                    expiryDate: document.getElementById('expiry-date').value

                };

                try {

                    const response = await fetch("/api/jobs/" + jobId, {

                        method: 'PUT',

                        headers: {

                            'Content-Type': 'application/json'

                        },

                        body: JSON.stringify(formData)

                    });

                    if (response.ok) {

                        alert('Job updated successfully!');

                        window.location.href = '/jobs/view';

                    } else {

                        const errorData = await response.json();

                        alert('Error: ' + errorData);

                    }

                } catch (error) {

                    alert('An error occurred while updating the job.');

                    console.error(error);

                }

            });

        });

    </script>

</body>

</html>

delete-job.jsp  
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>${title}</title>

    <link rel="stylesheet" href="<c:url value='/css/styles.css'/>">

    <style>

        body {

            background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);

            min-height: 100vh;

            display: flex;

            justify-content: center;

            align-items: center;

            margin: 0;

            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

        }

        .container {

            width: 100%;

            max-width: 900px;

            padding: 20px;

        }

        .dashboard {

            background: white;

            padding: 30px;

            border-radius: 10px;

            box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);

            margin: 0 auto;

        }

        .welcome-header {

            margin-bottom: 20px;

            text-align: center;

            color: #333;

        }

        .back-link {

            display: inline-block;

            margin-bottom: 20px;

            color: #667eea;

            text-decoration: none;

        }

        .back-link:hover {

            text-decoration: underline;

        }

        .job-list {

            margin-top: 20px;

        }

        .job-card {

            background: #f9f9f9;

            border-radius: 8px;

            padding: 20px;

            margin-bottom: 15px;

            box-shadow: 0 2px 5px rgba(0,0,0,0.05);

            transition: transform 0.2s, box-shadow 0.2s;

        }

        .job-card:hover {

            transform: translateY(-3px);

            box-shadow: 0 5px 15px rgba(0,0,0,0.1);

        }

        .job-title {

            font-size: 1.2rem;

            color: #333;

            margin-bottom: 5px;

            font-weight: 600;

        }

        .job-company {

            font-weight: 500;

            color: #555;

            margin-bottom: 10px;

        }

        .job-details {

            display: flex;

            flex-wrap: wrap;

            gap: 15px;

            margin-bottom: 15px;

            color: #666;

            font-size: 0.9rem;

        }

        .btn {

            background: #667eea;

            color: white;

            border: none;

            border-radius: 5px;

            padding: 8px 16px;

            cursor: pointer;

            text-decoration: none;

            font-size: 0.9rem;

            display: inline-block;

        }

        .btn-danger {

            background: #dc3545;

        }

        .btn:hover {

            background: #764ba2;

        }

        .btn-danger:hover {

            background: #c82333;

        }

        .empty-state {

            text-align: center;

            padding: 40px 0;

            color: #666;

        }

        .search-bar {

            display: flex;

            gap: 10px;

            margin-bottom: 20px;

        }

        .search-bar input {

            flex: 1;

            padding: 10px;

            border: 1px solid #ddd;

            border-radius: 5px;

            font-size: 15px;

        }

        .delete-warning {

            text-align: center;

            padding: 15px;

            background-color: #fff3cd;

            color: #856404;

            border-radius: 5px;

            margin-bottom: 20px;

        }

    </style>

</head>

<body>

    <div class="container">

        <div class="dashboard">

            <a href="<c:url value='/home'/>" class="back-link">← Back to Dashboard</a>

            <h1 class="welcome-header">Delete Job</h1>

            <div class="delete-warning">

                <p><strong>Warning:</strong> Deleting a job is permanent and cannot be undone.</p>

            </div>

            <div class="job-list">

                <c:choose>

                    <c:when test="${empty jobs}">

                        <div class="empty-state">

                            <h3>No jobs found</h3>

                            <p>There are no job listings available to delete.</p>

                            <a href="<c:url value='/jobs/post'/>" class="btn">Post a New Job</a>

                        </div>

                    </c:when>

                    <c:otherwise>

                        <c:forEach var="job" items="${jobs}">

                            <div class="job-card">

                                <div class="job-title">${job.title}</div>

                                <div class="job-company">${job.companyName}</div>

                                <div class="job-details">

                                    <span>📍 ${job.location}</span>

                                    <span>💼 ${job.employmentType}</span>

                                    <c:if test="${not empty job.salary}">

                                        <span>💰 ${job.salary}</span>

                                    </c:if>

                                    <span>📅 Posted: ${job.postedDate}</span>

                                </div>

                                <button class="btn btn-danger" onclick="confirmDelete(${job.id}, '${job.title}')">Delete Job</button>

                            </div>

                        </c:forEach>

                    </c:otherwise>

                </c:choose>

            </div>

        </div>

    </div>

    <script>

        document.addEventListener('DOMContentLoaded', () => {

*// Empty initialization function*

        });

        function **confirmDelete**(*jobId*, *jobTitle*) {

            if (confirm("Are you sure you want to delete the job \"" + *jobTitle* + "\"? This action cannot be undone.")) {

                fetch("/api/jobs/" + *jobId*, {

                    method: 'DELETE'

                })

                .then(*response* => {

                    if (*response*.ok) {

                        alert('Job deleted successfully');

                        location.reload();

                    } else {

                        alert('Error deleting job');

                    }

                })

                .catch(*error* => {

                    console.error('Error:', *error*);

                    alert('An error occurred while deleting the job');

                });

            }

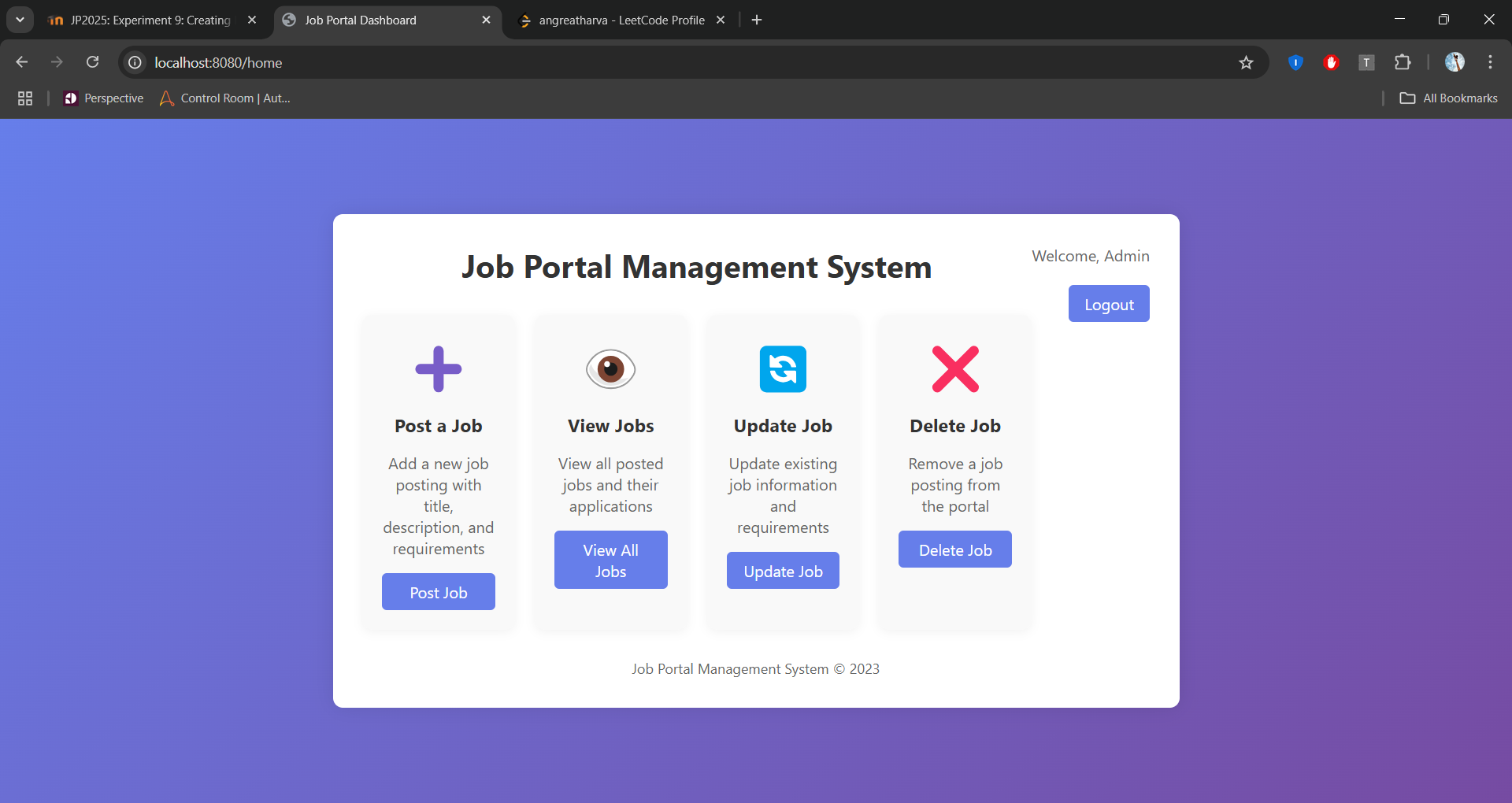
        }

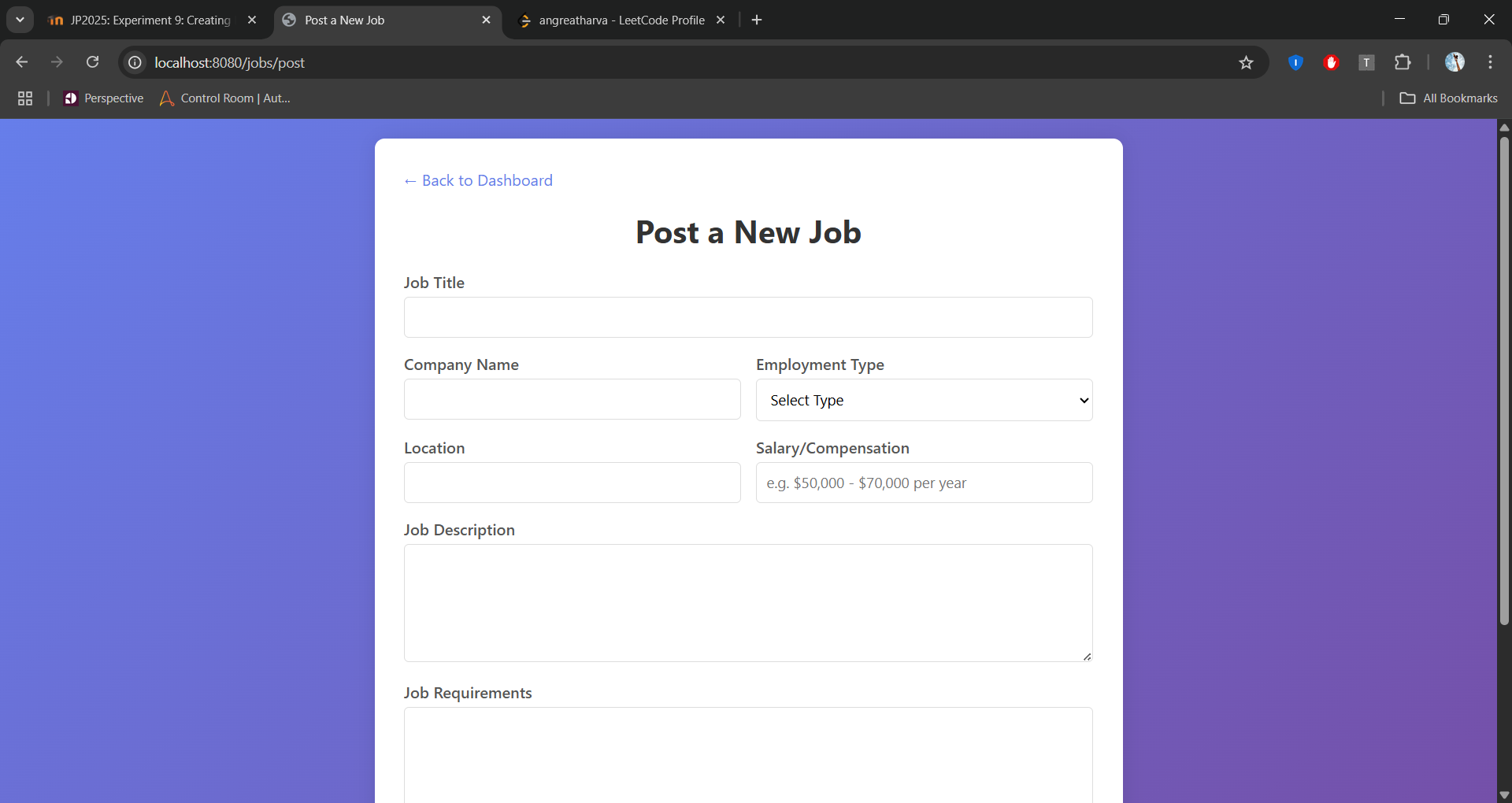
    </script>

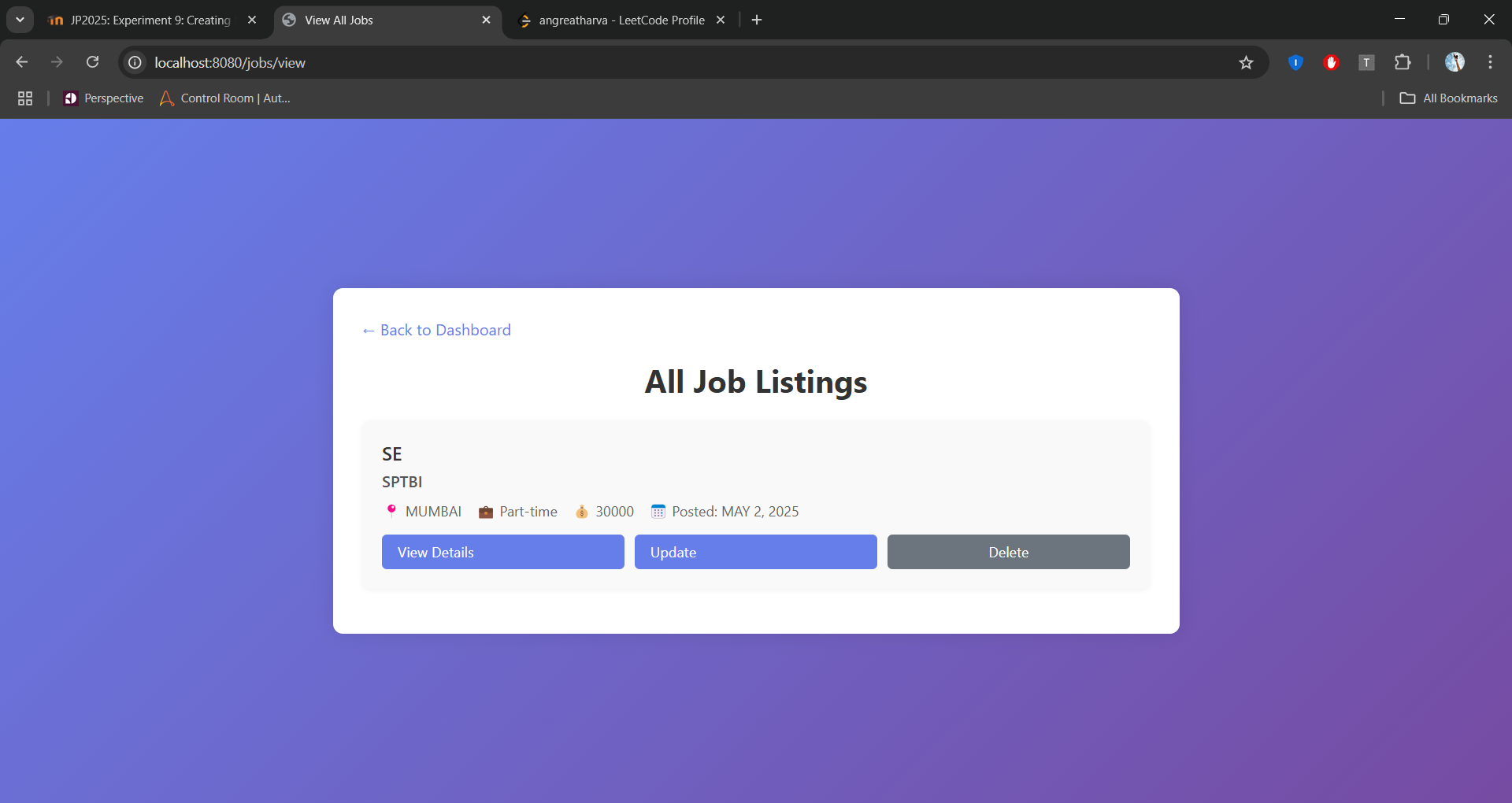
</body>

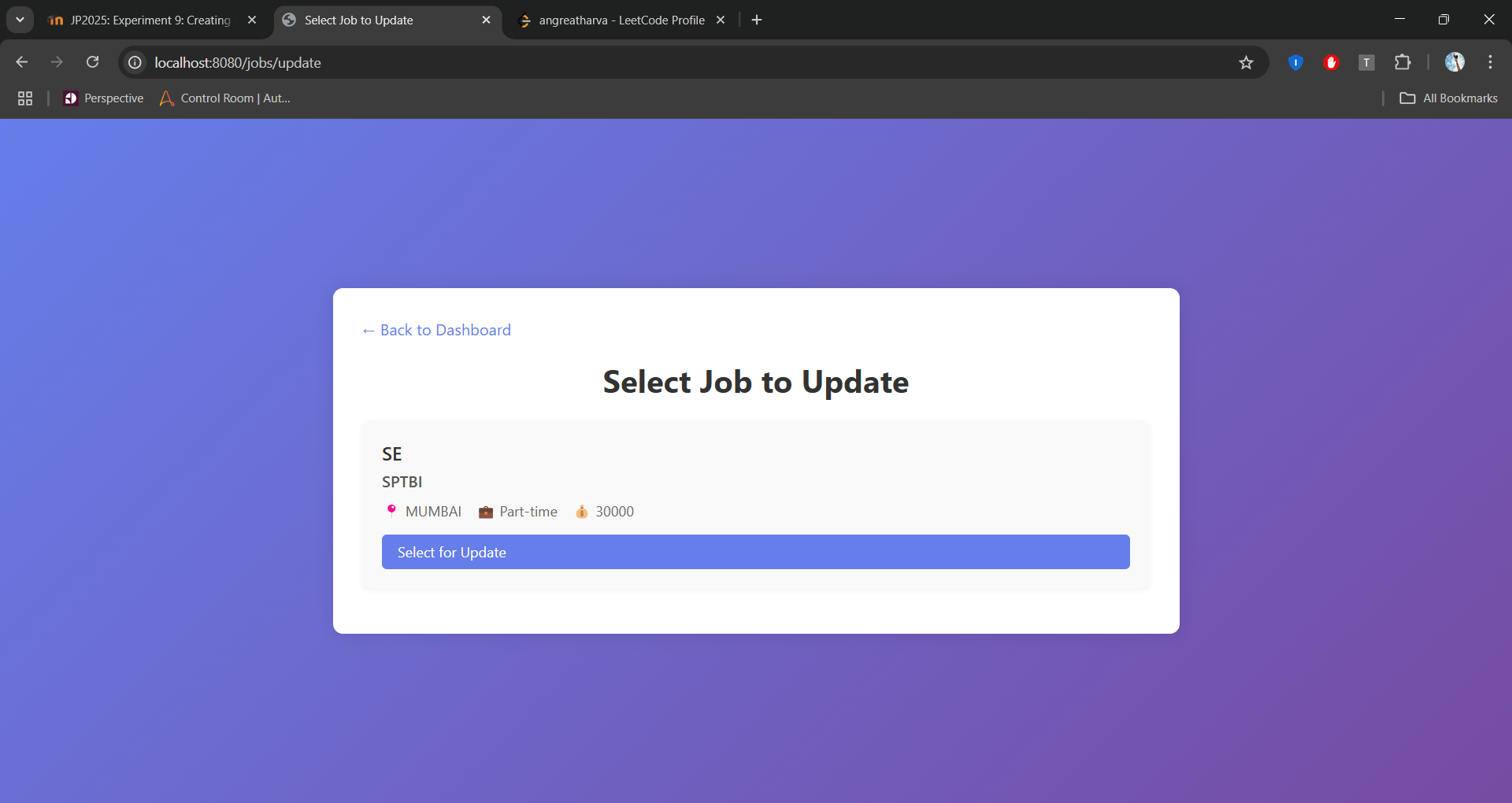
</html>

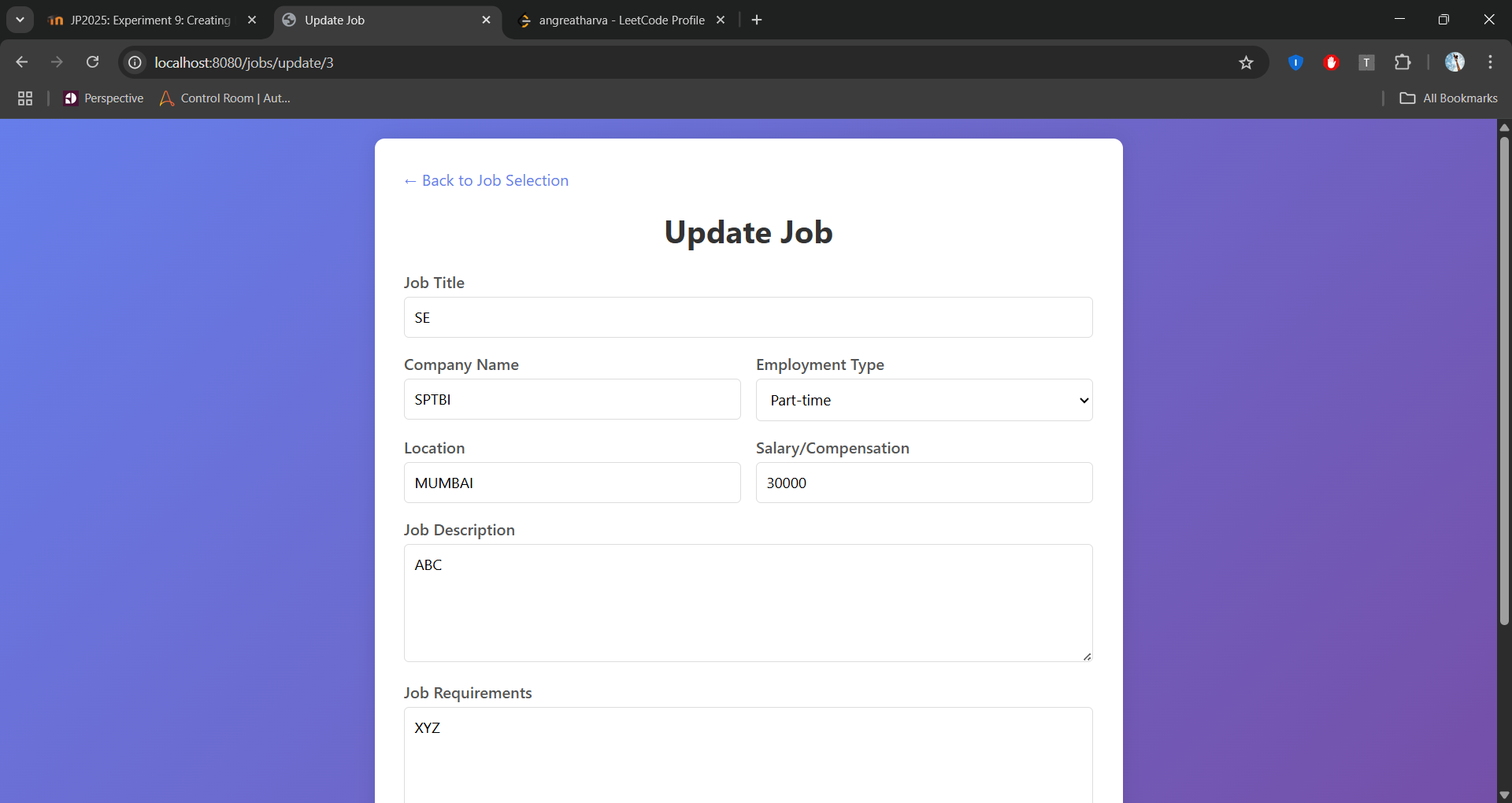
Output:

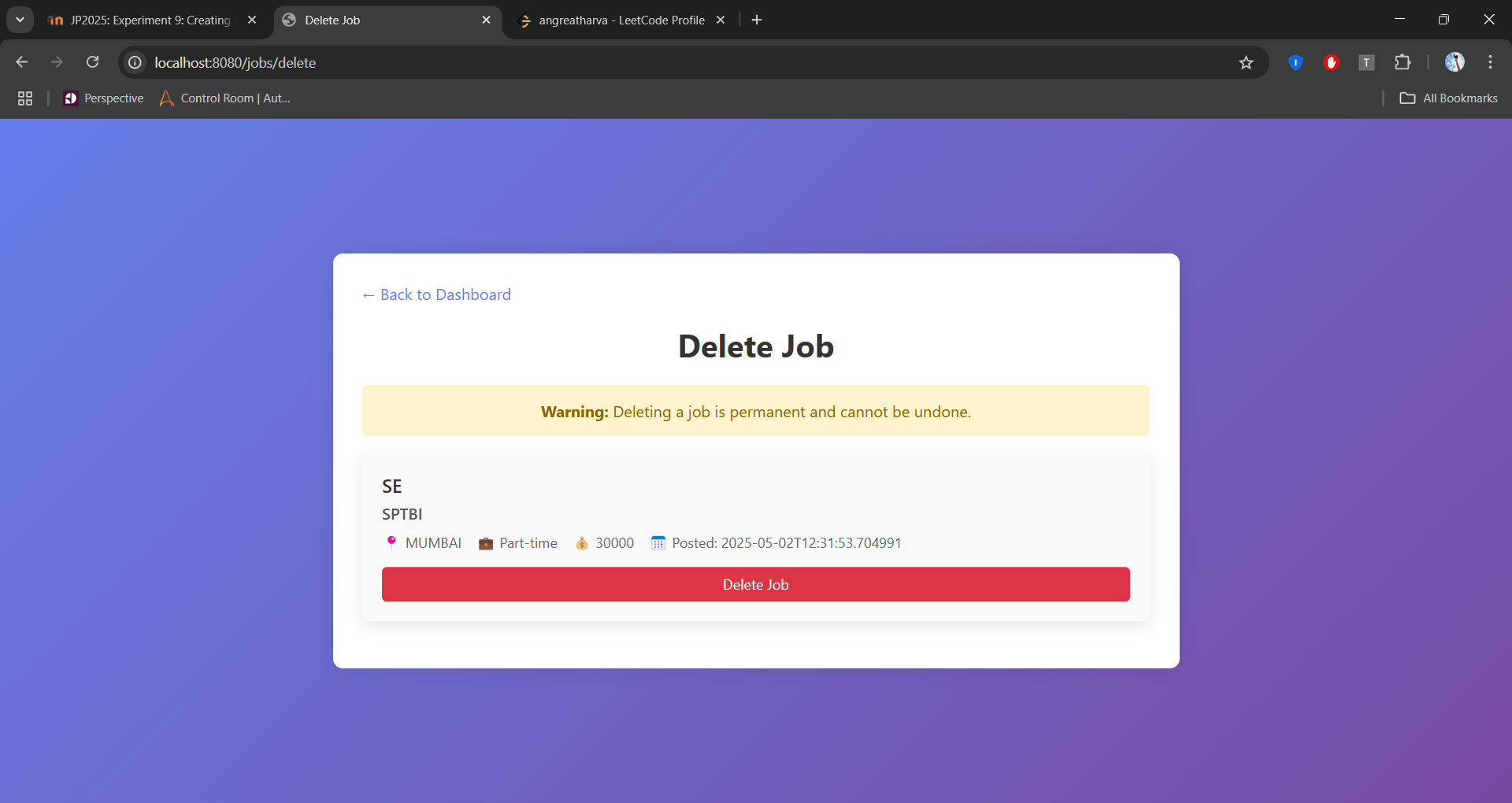












**Observation**: Through these practical sessions, I developed a Spring Boot-based Content Management System (CMS) that provided hands-on experience in modern web application development. The project involved implementing user authentication, job posting functionality, and database operations using Spring Data JPA with MySQL. I learned to structure the application using Spring MVC architecture, separating concerns into controllers, services, and repositories. The implementation included JSP views for the frontend, RESTful APIs for data exchange, and proper security measures using Spring Security. Additionally, I gained experience in configuring application properties, managing dependencies with Maven, and implementing custom filters for request processing. This project enhanced my understanding of full-stack development using Spring Boot, from database integration to frontend implementation.