Capstone Project – Secure Online Job Portal System (SOJPS)

Objective

Build a web secure application that enables employers to post jobs and candidates to apply for those jobs. The system will be implemented using Spring MVC (Spring 5), REST APIs, JWT authentication, and PostgreSQL with Hibernate following a Layered Architecture pattern and using Criteria Queries for dynamic data retrieval.

Technical Requirements

- Spring MVC (Spring 5) – No Spring Boot
- PostgreSQL database integration
- Hibernate ORM
- JWT-based authentication & authorization
- REST API endpoints returning JSON
- AJAX-based frontend interactions to consume REST APIs asynchronously
- Maven for dependency management
- Layered Architecture (Controller → Service → DAO → Entity)
- Criteria Query for dynamic searching/filtering of jobs and applications
- Global Exception Handling for consistent JSON error responses

Core Requirements

User Authentication & Authorization

- JWT-based authentication for all protected endpoints
- Role-based access control (EMPLOYER, CANDIDATE)

CRUD Operations

- Employers can Create, Read, Update, Delete job postings
- Candidates can apply for jobs and view their applications

Validation Requirements

- Job title: Required, min length 3, max length 150
- Job description: Required, max length 1000
- Employer email: Must be unique, valid format
- Candidate email: Must be unique, valid format

- Password: Min 8 characters, must contain uppercase, lowercase, number, and special character
- Application cover letter: Optional, max length 500

Error Handling

- Consistent error response format with HTTP status codes
- Use @ControllerAdvice and @ExceptionHandler for global exception handling

Database Integration

- Use PostgreSQL for persistence

API Requirements

Public Endpoints (No Auth):

- POST /register – Register new user (Employer/Candidate)
- POST /login – Authenticate and return JWT token

Protected Endpoints (JWT Required):

Employer Role:

- POST /jobs – Create a new job posting
- PUT /jobs/{id} – Update a job posting
- DELETE /jobs/{id} – Delete a job posting
- GET /jobs/{id}/applications – View applications for a job

Candidate Role:

- GET /jobs – View all available jobs (with search/filter support)
- POST /jobs/{id}/apply – Apply for a job
- GET /my-applications – View all applications submitted by the candidate

Functional Requirements

User Management

- Two roles: Employer (job poster) and Candidate (job seeker)
- Registration endpoint for new users with role assignment
- Login endpoint with JWT token generation
- Passwords securely hashed before storing (e.g., BCrypt)

Job Management (Employer Role)

- Employers can manage job postings with full CRUD

- Job posting details include: Job ID, Title, Description, Location, Salary Range, Posting Date, Application Deadline

Application Management (Candidate Role)

- Candidates can browse jobs and apply
- Prevent multiple applications to the same job by the same candidate
- Candidates can view their submitted applications

Entities

- User
- Job
- Application

Deliverables Expected

- Fully functional web application fulfilling all core and technical requirements
- ER Diagram and Class Diagrams
- Individual documentation from each participant detailing errors faced during development and the steps taken to resolve them

Evaluation Criteria

Criteria	Weightage
Functionality Implementation	30%
Validation Implementation	15%
JWT Authentication & Authorization	15%
Database Design (ERD, Class Diagram)	10%
Naming Conventions, Module Separation, Comments	10%
Documentation Completeness	10%
New Features Implemented Beyond Requirements	10%