System Requirements Specification

Index

For

Job Recruitment Platform

Version 1.0

TABLE OF CONTENTS

JOB RECRUITMENT PLATFORM

System Requirements Specification

You need to consume APIs exposed by Backend application in Angular to make application work as FULLSTACK

BACKEND-SPRING BOOT RESTFUL APPLICATION

1 PROJECT ABSTRACT

The **Job Recruitment Platform** is a FullStack Application with a backend implemented using Spring Boot with a MySQL database and a frontend developed using Angular. It enables users to manage various aspects of job details and facilitates job search for the job seekers.

Following is the requirement specifications:

	Job Recruitment Platform
Modules	
1	Job
Event Module Functionalities	
1	Create an Job
2	Update the existing Job details
3	Get the Job by Id
4	Get all Jobs
5	Delete an Job
6	Search for Job by Job Title
7	Search for Job by Company
8	Search for Job by Location

2 Assumptions, Dependencies, Risks / Constraints

2.1 JOB CONSTRAINTS

- When fetching a Job by ID, if the job ID does not exist, the operation should throw a custom exception.
- When updating a Job, if the job ID does not exist, the operation should throw a custom exception.
- When removing a Job, if the job ID does not exist, the operation should throw a custom exception.
- When searching for a Job using other parameters excluding by job title, company and location, the operation should throw a custom exception.

Common Constraints

- For all rest endpoints receiving @RequestBody, validation check must be done and must throw custom exception if data is invalid
- All the business validations must be implemented in dto classes only.
- All the database operations must be implemented on entity object only
- Do not change, add, remove any existing methods in service layer
- In Repository interfaces, custom methods can be added as per requirements.
- All RestEndpoint methods and Exception Handlers must return data wrapped in **ResponseEntity**

3 Business Validations

- Job Title is not null, min 3 and max 50 characters.
- Description of the job is not null.
- Company name is not null.
- Location of the job is not null.
- Employment Type is not null.
- Salary Range is optional.
- Skills Required is optional.
- Qualifications are optional.
- Contact Email is not null, should be a valid email address and max 200 characters.

4 REST ENDPOINTS

Rest End-points to be exposed in the controller along with method details for the same to be created

4.1 JOBCONTROLLER

URL Exposed		Purpose
1. /jobs		Fetches all the jobs
Http Method	GET	
Parameter	-	
Return	List<Jobs>	
2. /jobs		Add a new Job
Http Method	POST	
Parameter 1	Job	
Return	Job	
3. /jobs/{id}		Delete job with given job id
Http Method	DELETE	
Parameter 1	Long (id)	
Return	Boolean	
4. /jobs/{id}		Fetches the job with the given id
Http Method	GET	
Parameter 1	Long (id)	
Return	Job	
5. /jobs/{id}		Updates existing job
Http Method	PUT	
Parameter 1	Long (id)	
Parameter 2	Job	
Return	Job	
6. /jobs/search?jobTitle={jobTitle}		Search the job with the given job title
Http Method	GET	
Parameter 1	String (jobTitle)	
Return	List<Jobs>	
7. /jobs/search?company={company}		Search the job with the given company
Http Method	GET	
Parameter 1	String (company)	
Return	List<Jobs>	

8. /jobs/search?location={location}		Search the job with the given location
Http Method	GET	
Parameter 1	String (location)	
Return	List<Jobs>	

5 TEMPLATE CODE STRUCTURE

5.1 PACKAGE: COM.JOBRECRUIT

Resources

JobRecruitApplication (Class)	This is the Spring Boot starter class of the application.	Already Implemented

5.2 PACKAGE: COM.JOBRECRUIT.REPOSITORY

Resources

Class/Interface	Description	Status
JobDAO (interface)	• Repository interface exposing CRUD functionality for Job Entity. • You can go ahead and add any custom methods as per requirements.	Partially implemented.

5.3 PACKAGE: COM.JOBRECRUIT.SERVICE

Resources

Class/Interface	Description	Status
JobService (interface)	• Interface to expose method signatures for job related functionality. • Do not modify, add or delete any method.	Already implemented.

5.4 PACKAGE: COM.JOBRECRUIT.SERVICE.IMPL

Resources

Class/Interface	Description	Status
JobServiceImpl (class)	• Implements JobService. • Contains template method implementation. • Need to provide implementation for job related functionalities. • Do not modify, add or delete any method signature	To be implemented.

5.5 PACKAGE: COM.JOBRECRUIT.CONTROLLER

Resources

Class/Interface	Description	Status
JobController (Class)	• Controller class to expose all rest-endpoints for job related activities. • May also contain local exception handler methods	To be implemented

5.6 PACKAGE: COM.JOBRECRUIT.DTO

Resources

Class/Interface	Description	Status
JobDTO (Class)	Use appropriate annotations from the Java Bean Validation API for validating attributes of this class.	Partially implemented.

5.7 PACKAGE: COM.JOBRECRUIT.ENTITY

Resources

Class/Interface	Description	Status
Job (Class)	• This class is partially implemented. • Annotate this class with proper annotation to declare it as an entity class with **jobId** as primary key. • Map this class with an **job table**. • Generate the **jobId** using the IDENTITY strategy	Partially implemented.

5.8 PACKAGE: COM.JOBRECRUIT.EXCEPTION

Resources

Class/Interface	Description	Status
ErrorResponse (Class)	• Object of this class is supposed to be returned in case of exception through exception handlers	Already implemented.
ResourceNotFoundException (Class)	• Custom Exception to be thrown when trying to fetch or delete the job info which does not exist. • Need to create Exception Handler for same wherever needed (local or global)	Already implemented.
GlobalExceptionHandler (Class)	• RestControllerAdvice Class for defining global exception handlers. • Contains Exception Handler for **InvalidDataException** class. • Use this as a reference for creating exception handler for other custom exception classes.	Already implemented.

6 CONSIDERATIONS

A. There is no roles in this application
B. You can perform the following possible action

Job

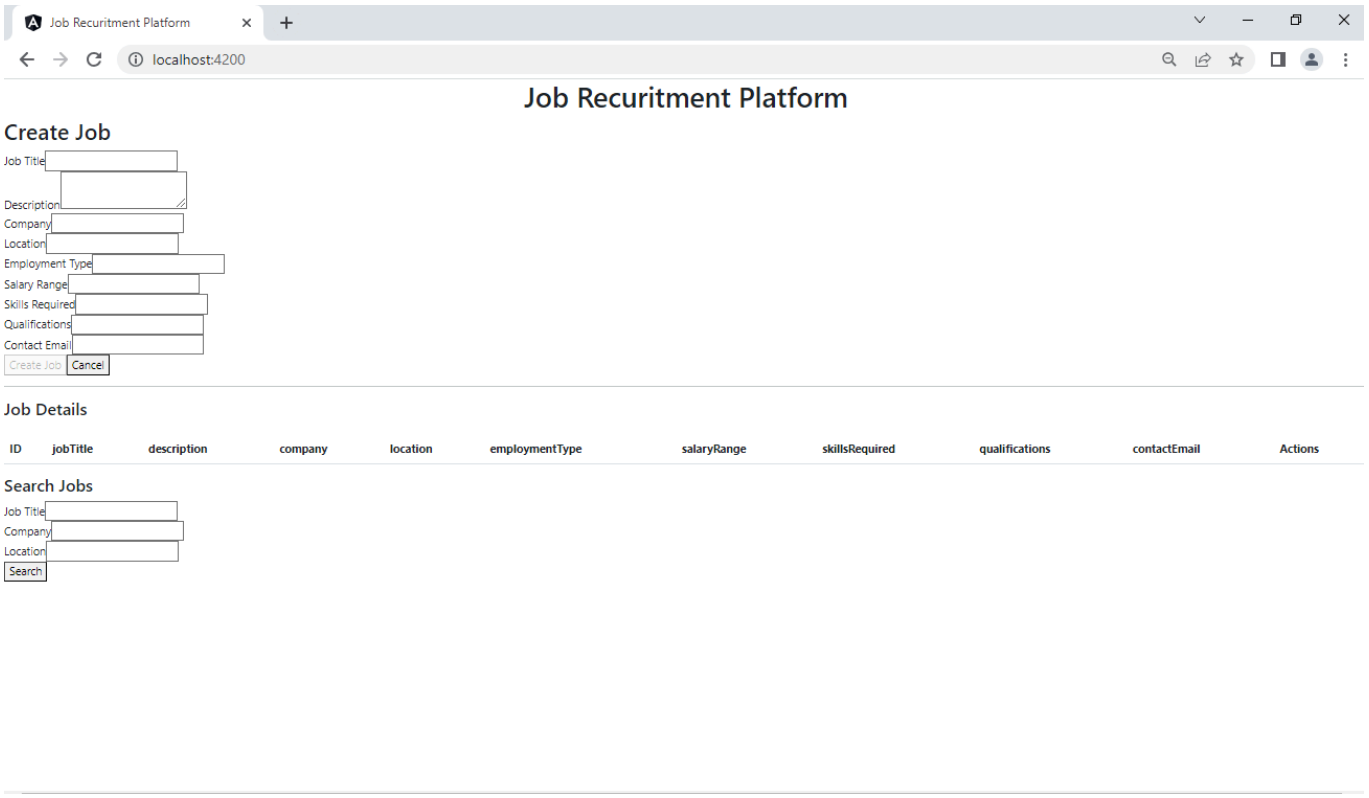
FRONTEND-ANGULAR SPA

1 PROBLEM STATEMENT

Job Recruitment Platform is SPA (Single Page Application), it allows to manage the jobs with functionalities to create a new job, update an existing job, get detailed information, and search for a job.

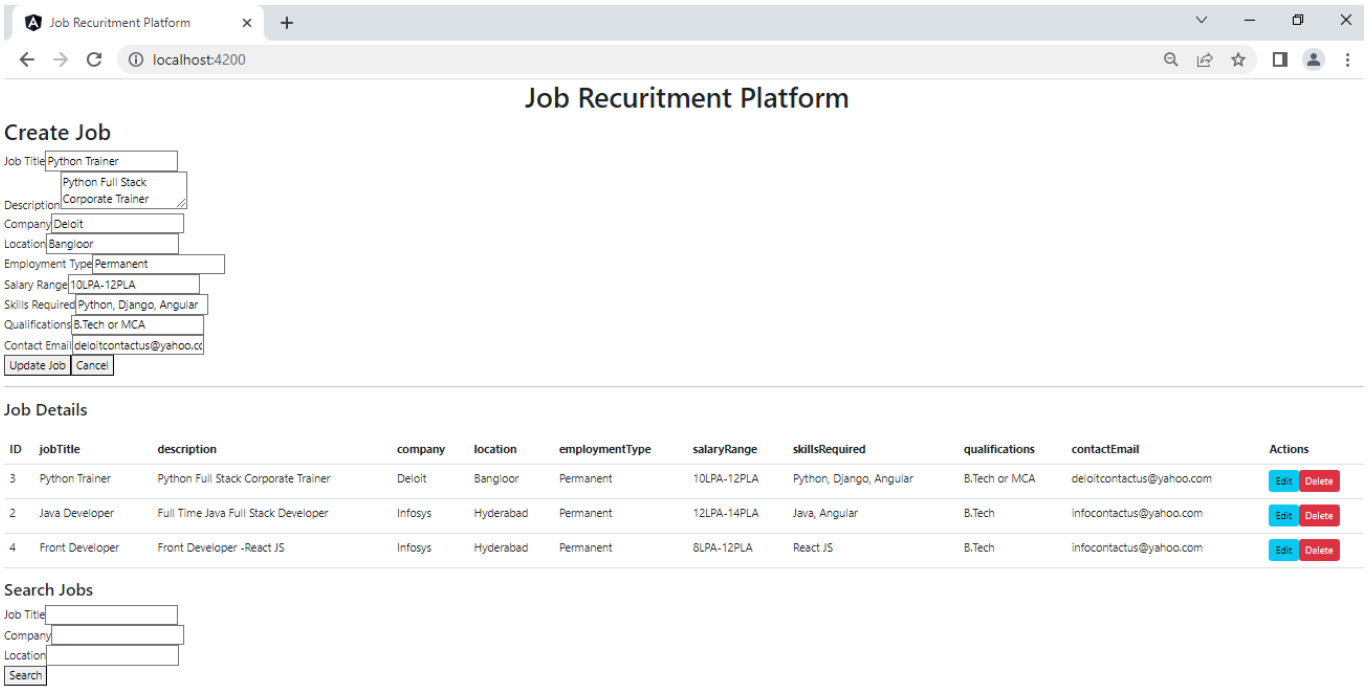
2. PROPOSED JOB RECRUITMENT PLATFORM WIREFRAME

2. 1 HOME PAGE



AFTER CREATING THE JOBS:

ON CLICK EDIT BUTTON:



3. BUSINESS-REQUIREMENT:

As an application developer, develop the Job Recruitment Platform App (Single Page App) with below guidelines:

User Story #	User Story Name	User Story
US_01	Home Page	As a user I should be able to visit the home page as the default page. Where I can see a form to create or update the job, a list of all jobs with options to edit and delete any job and at last there should be a form to search any job on its job title, company or location criteria.
US_01	Home Page	As a user I should be able to see the homepage and perform all operations: Acceptance criteria: 1. As a user I should be able to furnish the following details at the time of creating the job. 1.1 Job Title 1.2 Description 1.3 Company 1.4 Location 1.5 Employment Type 1.6 Salary Range 1.7 Skills Required 1.8 Qualifications 1.9 Contact Email 2. Create Job button should be disabled until all fields are validated. 3. Update Job button should be displayed when you click on the Edit button. 4. Update Job button should be disabled until all fields are validated. 5. On click cancel button, should reset the form. 6. Job Title field min length is 3 and max length 50. 7. All fields are mandatory except salary range, skills required, and qualifications fields. If any other field is missing or if any constraint is not satisfied then must show a message.

		8. Job Form control names should be case sensitive and they should be like as follows: jobTitle description company location employmentType salaryRange skillsRequired qualifications contactEmail 9. Search form fields are case sensitive and they should be like as follows: jobTitle company location These 3 fields are not mandatory in the search form.

7 EXECUTION STEPS TO FOLLOW FOR BACKEND

1. **All actions like build, compile, running application, running test cases will be through Command Terminal.**
2. **To open the command terminal the test takers need to go to the Application menu (Three horizontal lines at left top) -> Terminal -> New Terminal.**
3. **cd into your backend project folder**
4. **To build your project use command:**
   mvn clean package -Dmaven.test.skip
5. **To launch your application, move into the target folder (cd target). Run the following command to run the application:**
   java -jar <your application jar file name>
6. **This editor Auto Saves the code.**
7. **If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use CTRL+Shift+B-command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.**
8. **These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.**
9. **To test any Restful application, the last option on the left panel of IDE, you can find ThunderClient, which is the lightweight equivalent of POSTMAN.**
10. **To test any UI based application the second last option on the left panel of IDE, you can find Browser Preview, where you can launch the application.**
11. **Default credentials for MySQL:**
    a. **Username: root**
    b. **Password: pass@word1**

12. To login to mysql instance: Open new terminal and use following command:
    a. sudo systemctl enable mysql
    b. sudo systemctl start mysql

        NOTE: After typing any of the above commands you might encounter any warnings.

        >> Please note that this warning is expected and can be disregarded. Proceed to the next step.

    c. mysql -u root -p

        The last command will ask for password which is ‘pass@word1’

13. Mandatory: Before final submission run the following command:
    mvn test

14. You need to use CTRL+Shift+B - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.

8 EXECUTION STEPS TO FOLLOW FOR FRONTEND

1. **All actions like build, compile, running application, running test cases will be through Command Terminal.**
2. **To open the command terminal the test takers, need to go to**

   Application menu (Three horizontal lines at left top) -> Terminal ->New Terminal.
3. **This is a web-based application, to run the application on a browser, use the internal browser in the environment.**
4. **You can follow series of command to setup Angular environment once you are in your project-name folder:**
   a. **npm install -> Will install all dependencies -> takes 10 to 15 min**
   b. **npm run start -> To compile and deploy the project in browser. You can press <Ctrl> key while clicking on localhost:4200 to open project in browser -> takes 2 to 3 min**
   c. **npm run test -> to run all test cases. It is mandatory to run this command before submission of workspace -> takes 5 to 6 min**
5. **You need to use CTRL+Shift+B - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.**