System Requirements Specification Index

For

Professional Consultancy Services

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

TABLE OF CONTENTS

1 PROJECT ABSTRACT

Professional Consultancy Services (PCS) is a business consultancy that has established itself as a renowned service provider of a wide range of business services to its clients. PCS is planning to provide business- and employment-oriented service through a skill mapping application that operates via online recruiting website. This application is having spring boot microservices (employee, skills and certificates) which are using different databases and communicating to each other.

Following is the requirement specifications:

	Professional Consultancy Services
Microservices	
1	Employee-service
2	Skills-service
3	Certificates-service
Employee Microservice	
1	Register Employee
2	Update an Employee
3	Delete existing employee
4	Get the details of an employee
5	Get the details of all the employees
6	Get all the skills for an employee
7	Get all the certificates for an employee
Skills Microservice	
1	Create a skill for an employee
2	Update skill information
3	Get skill by Id
4	Fetch all skills
5	Delete skill
6	Get all skills for an employee
7	Get all certificates generated for the given skill
Certificates Microservice	
1	Generate a certificate for an employee
2	Update certificate information

3	Get certificate by Id
4	Fetch all certificates
5	Delete the certificate
6	Get all certificates for an employee
7	Get all certificates generated for the given skill

2 CONSTRAINTS

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 INTRODUCTION TO MICROSERVICES

3.1 EUREKA-NAMING-SERVER

This is a discovery server for all the registered microservices. This is fully implemented.

3.2 API-GATEWAY

This microservice is an api gateway to all the microservices. All the microservices can be accessed by using this common gateway. All the routes are already implemented in this api-gateway. All the rest endpoints can be accessed by this common gateway.

3.3 EMPLOYEE-SERVICE

The employee microservice is used to perform all the operations related to the employee. In this microservice , you have to write the logic for EmployeeServiceImpl.java and EmployeeRestController.java classes. Rest all the classes and interfaces are fully implemented. All the dependencies are already configured.

3.4 SKILLS-SERVICE

The skills microservice is used to perform all the operations related to employee skills. In this microservice, you have to write the logic for SkillsServiceImpl.java and SkillsRestController.java classes. Rest all the classes and interfaces are fully implemented. All the dependencies are already configured.

3.5 CERTIFICATES-SERVICE

The certificates microservice is used to perform all the operations related to employee certificates. In this microservice, you have to write the logic for CertificatesServiceImpl.java and CertificatesRestController.java classes. Rest all the classes and interfaces are fully implemented. All the dependencies are also already configured.

4 MICROSERVICES COMMUNICATION

The communication among the microservices is achieved by using the FeignClient. It is already configured in all the microservices. You can check in the proxy package of the microservice.

5 REST ENDPOINTS

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

5.1 EMPLOYEERESTCONTROLLER(EMPLOYEE-SERVICE)

URL Exposed (Employee Service)		Purpose
1. /api/employees		Register an Employee
Http Method	POST	
Parameter 1	EmployeeDto	
Return	EmployeeDto	
/api/employees		Update an Employee
Http Method	PUT	
Parameter 1	EmployeeDto	
Return	EmployeeDto	
/api/employees		Fetches the list of all registered Employees
Http Method	GET	
Parameter 1	-	
Return	List<EmployeeDto >	
/api/employees/{id}		Fetches the details of Employee by Id
Http Method	GET	
Parameter 1	Integer (id)	
Return	EmployeeDto	
/api/employees/{id}		Delete an employee information
Http Method	DELETE	

Parameter 1	Integer (id)	
Return	Boolean	
/api/employees/skills/{id}		Get all the skills belongs to the employee
Http Method	GET	
Parameter 1	Integer (id)	
Return	List<SkillsDto>	
/api/employees/certificates/{id}		Get all the certificates for an employee
Http Method	GET	
Parameter 1	Integer (id)	
Return	List<CertificatesDto>	

5.2 SkillsRestController (skills-service)

URL Exposed (Skills Service)		Purpose
/api/skills		Register a Skill for an employee
Http Method	POST	
Parameter 1	SkillsDto	
Return	SkillsDto	
/api/skills		Update the Skill
Http Method	PUT	
Parameter 1	SkillsDto	
Return	SkillsDto	
/api/skills		Fetches all the skills
Http Method	GET	
Parameter 1	-	
Return	List< SkillsDto >	
/api/skills/{id}		Fetch the details of a Skill
Http Method	GET	
Parameter 1	Integer (id)	
Return	SkillsDto	
/api/skills/{id}		Delete a Skill
Http Method	DELETE	
Parameter 1	Integer (id)	
Return	Boolean	
/api/skills/{employeeId}		Fetch the details of all the skills registered under an employee
Http Method	GET	
Parameter 1	Integer (employeeId)	
Return	List<SkillsDto >	
/api/skills/certificates-by-skill-name/{name}		Fetch the details of all the certificates for a given skill
Http Method	GET	
Parameter 1	String (name)	
Return	List<CertificatesDto >	

5.3 CERTIFICATERESTCONTROLLER (CERTIFICATES-SERVICE)

URL Exposed (Certificates Service)		Purpose
/api/certificates		Generate a Certificate
Http Method	POST	
Parameter 1	CertificatesDto	
Return	CertificatesDto	
/api/certificates		Update an existing certificate
Http Method	PUT	
Parameter 1	CertificatesDto	
Return	CertificatesDto	
/api/certificates		Fetches all the generated certificates
Http Method	GET	
Parameter 1	-	
Return	List<CertificatesDto >	
/api/certificates/{id}		Fetch the details of a certificate
Http Method	GET	
Parameter 1	Integer(id)	
Return	List<CertificatesDto >	
/api/certificates/{id }		Deletes an existing certificate
Http Method	DELETE	
Parameter 1	Integer(id)	
Return	Boolean	
/api/certificates/employees/{employeeId}		Get all the certificates for an employees
Http Method	GET	
Parameter 1	Integer(employeeId)	
Return	CertificatesDto	
/api/certificates/skills/{skillName}		Get all the certificates for a given skill.
Http Method	GET	
Parameter 1	String(skillName)	
Return	List<CertificatesDto>	

6 CONSIDERATIONS

A. You can perform the following 5 possible actions

Employee Actions
Skills Actions
Certificates Actions
Discover all the services by Eureka
Use API Gateway to access all the microservices

7 SEQUENCE TO EXECUTE

The sequence has to be followed for step 8 for every microservice are given below:

- eureka-naming-server
- api-gateway
- employee-service
- skills-service
- certificates-service

**Strictly follow the above sequence to follow step number 8.

8 EXECUTION STEPS TO FOLLOW

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. **Kindly follow the sequence and run your commands through all the folders separately.**
4. **To build your project and run test cases use command:**

   mvn clean package
5. **To launch your application, move into the target folder (cd target). Run the following command to run the application:**

   java -jar <jar-name>-0.0.1-SNAPSHOT.jar
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**
11. **To login to mysql instance: Open new terminal and use following command:**
    a. **sudo systemctl enable mysql**
    b. **sudo systemctl start mysql**
    c. **mysql -u root -p**
    The last command will ask for password which is 'pass@word1'
12. **Mandatory: Before final submission run the following command:**
    mvn test
13. **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.**