System Requirements Specification Index

**For**

Professional Consultancy Services

### Version 1.0

### IIHT Pvt. Ltd.

[**fullstack@iiht.com**](mailto:fullstack@iiht.com)

**Table of Contents**

1. [Project Abstract 3](#_bookmark0)
2. [Constraints 4](#_bookmark1)
3. [Introduction to Microservices 4](#_bookmark2)
   1. eureka-naming-server 4

|  |  |  |
| --- | --- | --- |
|  | 3.2 api-gateway | 4 |
| 3.3 employee-service | 4 |
| 3.4 skills-service | 4 |
| 3.5 certificates-service | 5 |
| 4 | Microservices Communication | 5 |
| 5 | Rest Endpoints | 5 |
|  | 5.1 EmployeeRestController | 5 |
|  | 5.2 SkillsRestController | 6 |
|  | 5.3 CertificatesRestController | 7 |
| 6 | Considerations | 8 |
| 7 | Sequence to execute | 16 |
| 8. | Execution Steps to |  |

Follow……………………………………………………………………………………………………………

………8

**Professional Consultancy Services**

## System Requirements Specification

# 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 |
|  |  |
|  |  |
|  |  |

# 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 using appropriate annotations.
* All the database operations must be implemented on entity object only. Use appropriate annotations wherever applicable.
* 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

# System Requirements

* 1. **EUREKA-NAMING-SERVER**

This is a discovery server for all the registered microservices. Following implementations are expected to be done:

1. Configure the Eureka server to run on port: 8761.
2. Configure the Eureka server to deregister itself as Eureka client.
3. Add appropriate annotation to Enable this module to run as Eureka Server.

**You can launch the admin panel of Eureka server in browser preview option.**

* 1. **API-GATEWAY**

This microservice is an api gateway to all the microservices. All the microservices can be accessed by using this common gateway. Following implementations are expected to be done:

1. Configure API Gateway to run on port: 8765.
2. Implement the routes and logging in this api-gateway.
   1. **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. Following implementations are expected to be done:

1. Configure this service to run on port: 8001.
2. You are required to configure 2 feign proxy to fetch (Get Certificate by Employee ID and get Skills by Employee Id)
   1. **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. Following implementations are expected to be done:

1. Configure to run this module on port: 8090
2. You are required to configure feign proxy to fetch : Certificate by Skill name
   1. **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. Following implementations are expected to be done:

1. Configure to run this module on port: 9001
2. **Kindly follow the sequence and run your commands through all the folders separately.**
3. **To build your project use command:**

mvn clean package -Dmaven.test.skip

1. **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

# Fault Tolerance

Configure your skill microservice DB operation for Hysterix circuit breaker and create fallback function in case database is not connected.

1. **Microservices Communication**

Communication among the microservices needs to be achieved by using FeignClient. Feign configuration class is created in the project, but you are required to implement the feign client method. You can check in the proxy package of the microservice.

1. **Rest Endpoints**

Rest Endpoints to be exposed in the controller along with method details for the same to be created

### EmployeeRestController(employee-service)

|  |  |
| --- | --- |
| **URL Exposed (Employee Service)** | **Purpose** |
| 1. /api/employees | Register an Employee |
| /api/employees | Update an Employee |

|  |  |
| --- | --- |
| Http Method | POST |
| PHattrpamMeettehro1d | EPmUTployeeDto |
| PRaertaumrneter 1 | EmployeeDto |
| Return | EmployeeDto |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| /api/employees | | | | |  |  | | --- | --- | | Http Method | GET | | Parameter 1 | - | | Return | List<EmployeeDto > |   Fetches the list of all registered Employees |
|  | Http Method | DELETE |  |
|  | Parameter 1 | Integer (id) |
|  | /aRpeit/uermn ployees/{id} | Boolean |  | Fetches the details of Employee by Id |
|  | Http Method | GET |
|  | Parameter 1 | Integer (id) |
|  | HRet**t**purMn ethod | GEmETployeeDto |  |  |
|  | Parameter 1 | Integer (id) |
|  | /aRpeit/uermn ployees/{id} | List<SkillsDto> |  | Delete an employee information |
|  | | | |
| /api/employees/skills/{id} | | | | Get all the skills belongs to the employee |
| /api/employees/certificates/{id} | | | | Get all the certificates for an employee |

### SkillsRestController (skills-service)

|  |  |
| --- | --- |
| Http Method | GET |
| Parameter 1 | Integer (id) |
| Return | List<CertificatesDto> |

**Purpose**

**URL Exposed (Skills Service)**

/api/skills

|  |  |
| --- | --- |
| Http Method | POST |
| Parameter 1 | SkillsDto |
| Return | SkillsDto |

/api/skills

Fetches all the skills

Update the Skill

for an

Skill

Register a

employee

|  |  |
| --- | --- |
| Http Method | PUT |
| Parameter 1 | SkillsDto |
| Return | SkillsDto |

/api/skills

|  |  |
| --- | --- |
| Http Method | GET |
| Parameter 1 | - |
| Return | List< SkillsDto > |

/api/skills/{id}

|  |  |
| --- | --- |
| Http Method | GET |

/api/skills/{id}

Fetch the details of a Skill

Page6 | 11

Delete a Skill

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Parameter 1 | Integer (id) |  | |
|  | HRet**t**purMn ethod | GSkEiTllsDto |
|  | Parameter 1 | Integer (employeeId) |
|  | Return | List<SkillsDto > |
|  | Http Method | DELETE |
|  | Parameter 1 | Integer (id) |
|  | HRet**t**purMn ethod | GBoETolean |
|  | /aPpair/asmkiellste/{re1mployeeId} | String (name) |  | Fetch the details of all the skills registered under an employee |
|  | Return | List<CertificatesDto > |
|  | | | |
| /api/skills/certificates-by-skill-name/{name} | | | | Fetch the details of all the certificates for a given skill |

### CertificateRestController (certificates-service)

/api/certificates

/api/certificates/{id }

Fetch the details of a certificate

Fetches all the generated certificates

Update an existing certificate

Generate a Certificate

**Purpose**

**URL Exposed (Certificates Service)**

|  |  |
| --- | --- |
| Http Method | POST |
| Parameter 1 | CertificatesDto |
| Return | CertificatesDto |

/api/certificates

|  |  |
| --- | --- |
| Http Method | PUT |
| Parameter 1 | CertificatesDto |
| Return | CertificatesDto |

/api/certificates

|  |  |
| --- | --- |
| Http Method | GET |
| Parameter 1 | - |
| Return | List<CertificatesDto > |

/api/certificates/{id}

|  |  |
| --- | --- |
| Http Method | GET |
| Parameter 1 | Integer(id) |
| Return | List<CertificatesDto > |

Deletes an existing certificate

Page7 | 11

/api/certificates/employees/{employeeId}

Get all the certificates for an

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Http Method | DELETE |  | |
|  | Parameter 1 | Integer(id) |
|  | Return | Boolean |
|  | HPattrpamMeettehro1d | GInEtTeger(employeeId) |  |  |
|  | PRaertaumrneter 1 | SCterrintigfi(csaktilelNsDatmoe) |
|  | Return | List<CertificatesDto> |
| /api/certificates/skills/{skillName} | | | | Get all the certificates for a given skill. |

1. **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 |

1. **Execution Steps to Follow**
2. **All actions like build, compile, running application, running test cases will be through Command Terminal.**
3. **To open the command terminal the test takers, need to go to Application menu (Three horizontal lines at left top) -> Terminal -> New Terminal**
4. **Kindly follow the sequence and run your commands through all the folders separately.**
5. **To build your project use command:**

mvn clean package -Dmaven.test.skip

1. **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

1. **This editor Auto Saves the code**
2. **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.**
3. **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.**
4. **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.**
5. **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.**
6. **Default credentials for MySQL:**
   1. **Username: root**
   2. **Password: pass@word1**
7. **To login to mysql instance: Open new terminal and use following command:**
   1. **sudo systemctl enable mysql**
   2. **sudo systemctl start mysql**
   3. **mysql -u root -p**

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

1. **Mandatory: Before final submission run the following command for each of the module: mvn test**
2. **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.**