Employ Me Recruitment Agency (Spring Boot + Microservices)

1.0 Introduction

Employee Me Recruitment is a leading recruiter agency and a leading provider of jobs in India. Landing the dream job is a difficult task, especially in today's competitive job market, but Employ Me recruitment agency help the candidate to choose the right job. The candidate needs to register with Employ Me by providing their details, based on the details given by the candidate, the candidate will get openings from various companies via mail.

The database of the recruitment details has increased drastically. So, they need an application to enhance the below task.

1. Register Candidate
2. View the registered Candidate
3. Filter the employee based on experience
4. Remove the selected candidate

The client wishes to have restful webservices using spring boot for the doing the above task. Help them to automate the above process by developing Rest Service using Maven and incorporate microservices for the same.

2.0 Technical Specifications

The provided RecruitmentController which is a RestController, should be created with the below services:

Services:

Request Method	Request Url	Method in Controller	Description
Post	/register	registerCandidate	This service should add the candidate by using the registerCandidate method of the RecruitmentImpl class
Get	/view	viewCandidateBasedonPosition	This service should retrieve the candidates by using the viewCandidateBasedonPosition method of the RecruitmentImpl class.
Get	/filter/{yearsOfExperience}	filterCandidate	This service should retrieve the List of candidates by using the filterCandidate method of the RecruitmentImpl class based

			on the years of experience passed.
delete	/remove	removeCandidate	This service should remove the candidate based on the status by using the removeCandidate method of the RecruitmentImpl class.

The Controller class has the below attribute

IRecruitmentService service;

It should be injected in the Controller via annotation

Microservice Specification:

Spring boot application should contain all the REST services implementation as per the case study specification.

The Spring rest service application should have the application name as "recruitmentapp" in the application.properties.

Register the spring rest service application with the Eureka server which should run in port 8761.

API Gateway Server:

- Implement API Gateway using Cloud API Gateway
- Must contain a routing information for
  - /register URI mapping,
  - /view URI mapping
  - /filter/{yearsOfExperience} URI mapping and
  - /remove URI mapping to the actual implementation running at port 8080.
- The API gate way server (i.e Router) port(8777) must be specified in the "application.properties" file.
- The service must be accessible through API gateway.

Note:

Do not specify absolute URL of the rest service in the Gateway properties file. Rather use service-id to specify the name of the application (Rest service). That is in Gateway, to specify the application name use below property in application.properties

spring.application.name = recruitmentapp

Service Layer

Refer to the IRecruitmentService interface provided as part of the code skeleton. The RecruitmentServiceImpl class which is provided as part of the code skeleton has to realize all the methods in the IRecruitmentService interface.

RecruitmentServiceImpl class should be configured via annotation as Service.

Attributes in the RecruitmentServiceImpl class

RecruitmentServiceImpl	
Attribute Name	**Attribute Type**
candidateList	List<Candidate>

Method specification in service class

Method in Service	Description	Exception
registerCandidate	This method should add the candidate details to the candidateList and return the candidate object. Use SLF4J and log the success message as **"Candidate with id <<candidateId>> registered successfully"**	If the candidate already registered with Employ Me, then throw a user defined exception CandidateAlreadyExistsException with the message **"Candidate already exists with us".** If the candidate email id already exists with Employ Me then throw the exception. Use SLF4J and log this as error message
removeCandidate	This method should remove the candidate based on the status. If the status is recruited then remove all those candidates. This method should return the number of candidates removed. Use SLF4J and log the success message as **"<<xxx>> candidate recruited via Employ Me"**	

filterCandidate	This method accepts the year of experience as parameter. It should iterate the candidateList and return the list of candidates who have that experience. Use SLF4J and log the success message as "**View candidate based on the experience is successfully done**"	
viewCandidateBasedonPosition	This method should return the Map as position applied for as a key and list of candidates who applied for that position as value. Use SLF4J and log the message as "**View candidate based on position is successful**"	

Exception Class

Refer to the CandidateAlreadyExistsException class provided as part of the code skeleton. This class should extend the Exception class. Provide the needed annotation so that it handles HttpStatus as NOT_FOUND

Model Class

Refer to the Candidate class provided as part of the code skeleton.

Attribute Name	**DataType**
candidateId	String
candidateName	String
mobileNumber	String
emailId	String
positionAppliedFor	String
yearsOfExperience	Integer
expectedSalary	double
status	String

Note:

- **Use Lombok to generate getters and setters for the attributes**
- **Do not change the all-argument constructor and no argument constructor provided as part of the code skeleton**
- **Do not change the datatype or the attribute name provided as part of the code skeleton**

Business Validation

Inbuilt Validation

Rule	Message when validation fails
candidateId should not be null	Provide value for candidate Id
emailId should not be null	Provide value for email Id
mobileno should not be null	Provide value for mobile number
expectedSalary should not be negative	Provide value greater than or equal to zero

Perform the above Validations using annotations that are available in javax.validation.constraints package

Custom Validation

Perform Custom validation for the years of experience attribute.

Rule	Message when validation fails
If the applied for has the value as fresher, then the years of experience should be zero	The years of experience should be zero if you applied as a fresher

For doing the custom validation this you are provided with RecruitmentValidator class. This class should be configured via annotation as Component.

Incorporate Exception and validations as ResponseEntity

To incorporate the messages related to Exception and validations (both custom and inbuilt validations) as ResponseEntity, you are provided with the below classes

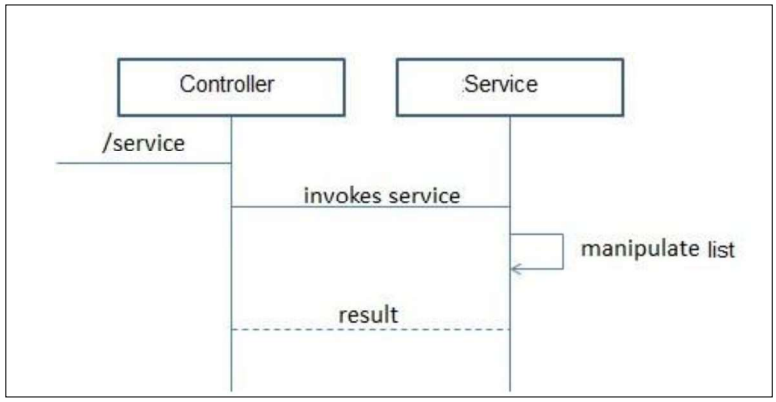
- Make use of ExceptionResponse class that is already provided as part of the code skeleton to send the customized response error message.
- CustomizedResponseEntityExceptionHandler class to handle all the exceptions that has occurred in the application (both user defined and pre-defined)
-

Method Name	Explanation

handleAllExceptions	This method should generally handle all the exceptions
handleNotFoundException	This method should handle and provide customized error message using Exception Response class for user defined exception CandidateAlreadyExistsException
handleMethodArgumentNotValid	This method should handle and provide customized error message using Exception Response class for both pre-defined and user defined validations

Note: In CustomizedResponseEntityExceptionHandler class log all the error messages using SLF4J.

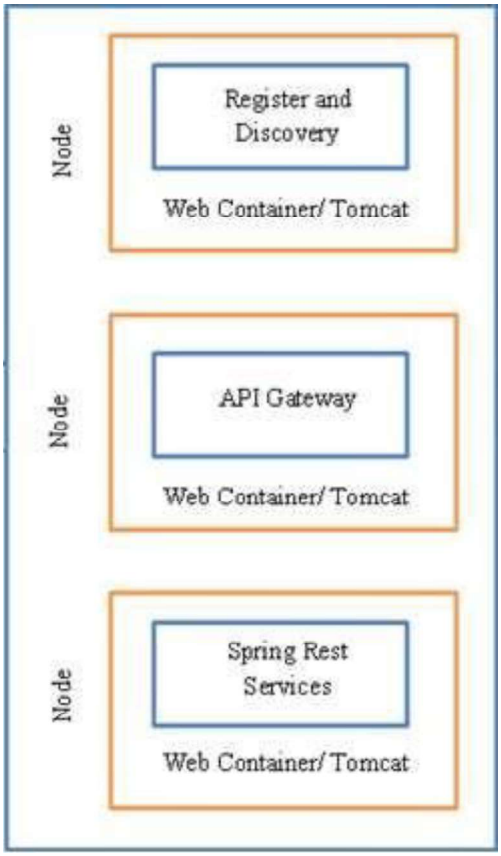
3.0 Process Flow



- Client invokes the required service.
- Controller invokes the method of the RecruitmentServiceImpl.
- RecruitmentServiceImpl performs the service and returns the data back.
- IRecruitmentService has to be injected into the RecruitmentController.
- RecruitmentValidator has to be injected into the RecruitmentController.
- Use appropriate annotations for performing the validations.
- To perform Custom validation, you are provided with a class RecruitmentValidator. Provide the necessary annotation and implement the validate method
- To perform **logging**, you are provided with logger.xml file. Do not change the contents of the logger.xml file provided as part of the code skeleton
- Use Lombok to create getters and setters in Candidate class. Do not alter the constructors provided as part of the code skeleton.
- Configure logging using Slf4j annotation with Lombok in RecruitmentService and CustomizedResponseEntityExceptionHandler class.

- When multiple validation fails, concatenate all messages as a single string and store it in details attribute of ExceptionResponse object.

Architecture diagram



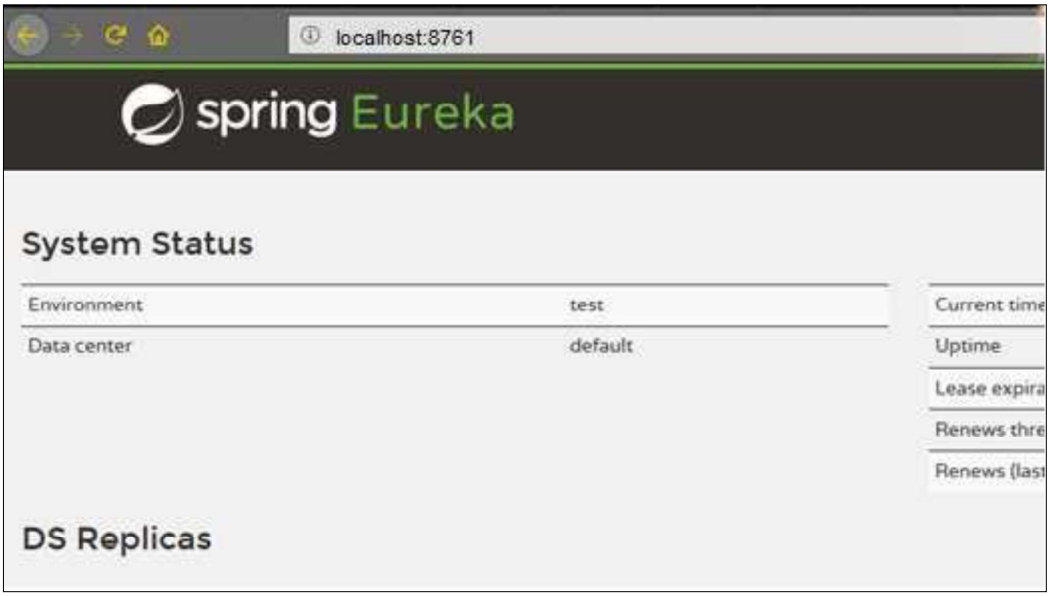
4.0 Microservices implementation Guidelines

For Microservices implementation, we will be implementing 3 projects.

- Spring Cloud (Eureka-Server) for service registry and discovery
- Spring Boot for REST service creation
- Spring Cloud gateway for API Gateway

Project 1:

This project should be the Eureka-Server acting as the registry. Run this server in port 8761



Project 2:

This project should be a spring boot application containing all the REST services implemented as per the requirements stated in the case study.

These services must be registered with Eureka-Server.

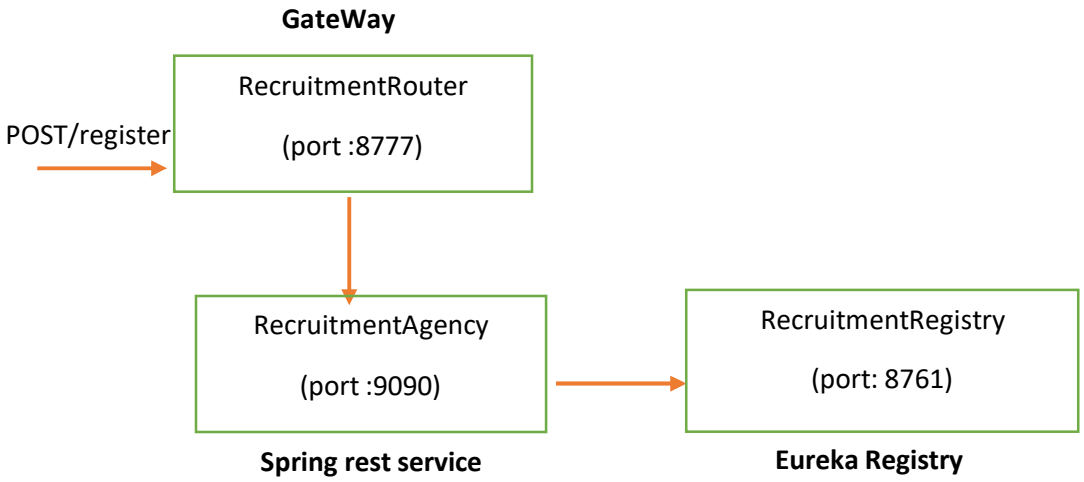
Run the services in port 9090.Services should get automatically registered with the Eureka Registry as shown below

Note:

- **Applicationname should be given as recruitmentapp as specified in the case study.**
- **Eureka Registry should run only in port 8761**
- **You can change application port number i.e. 9090 if required.**

Project 3:

This must be a Spring cloud gateway project that contains the routing implementation to the actual services running in port 9090. Run this in port 8777.



Overall Design Constraints

1) **Do not change the property name given in the application.properties files, you can change the value and you can include additional property if needed.**
2) In the pom.xml you are provided with all the dependencies needed for developing the application.
3) Use the service type and the service names as expected in the specification
4) Adhere to the design specifications mentioned in the case study.
5) Do not change or delete the class/method names or return types which are provided to you as a part of the base code skeleton.
6) Developer will create 3 applications:
    - Spring Boot (Name of the application is RecruitmentAgency)
    - Spring Cloud (Eureka-Server, spring-cloud-starter-eureka-server) (Name of the application is RecruitmentRegistry)
    - Spring Cloud API GateWay(spring-cloud-starter-gateway) (Name of the application is RecruitmentRouter)
7) CORS related issues need to be handled at the application level.
8) Services can be tested using external tools like postman by the associate.
9) Eureka Registry should run only in port 8761. You can change the port number of the application (spring rest) or the Gateway (Router) if required.

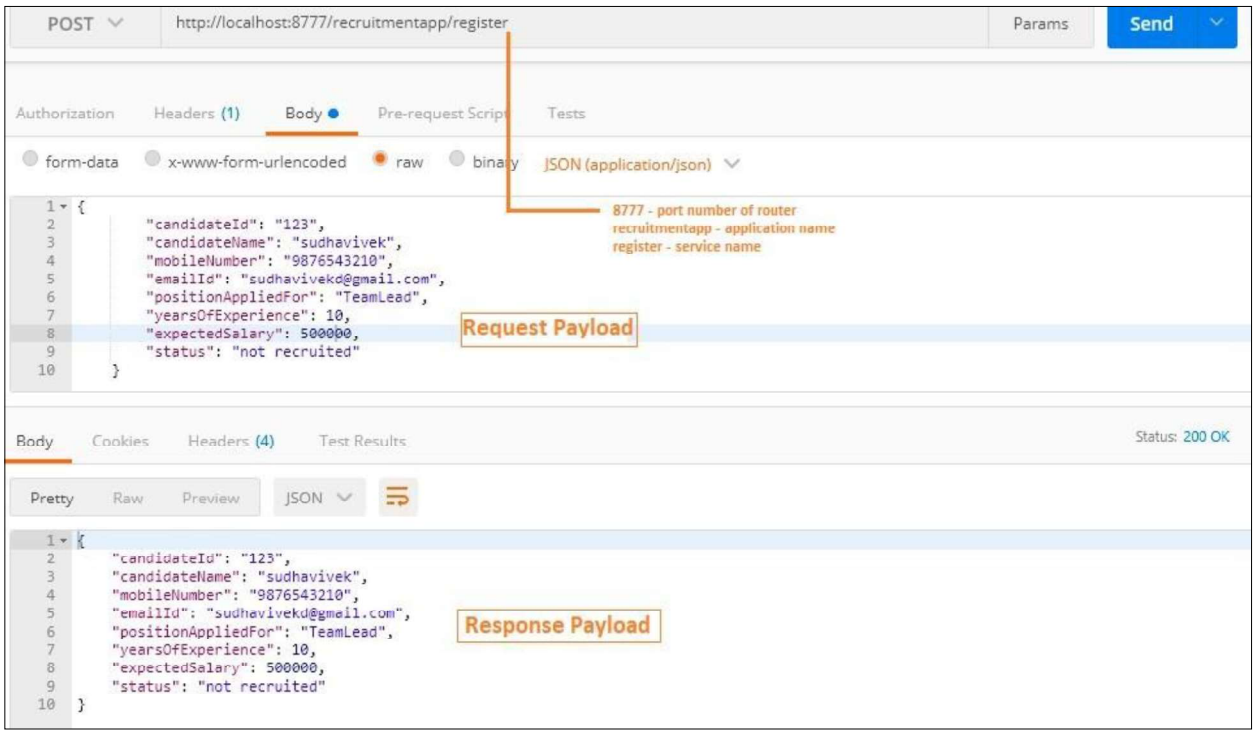
10) Application should be configured in the Gateway(Router) only by using service-id and not by using absolute url. ie in Gateway properties use the below properties to configure the service `spring.application.name = recruitmentapp`

11) **Please make sure that your code does not have any compilation errors while submitting your case study solution.**

12) **Your code will not get evaluated if the code skeleton is altered.**

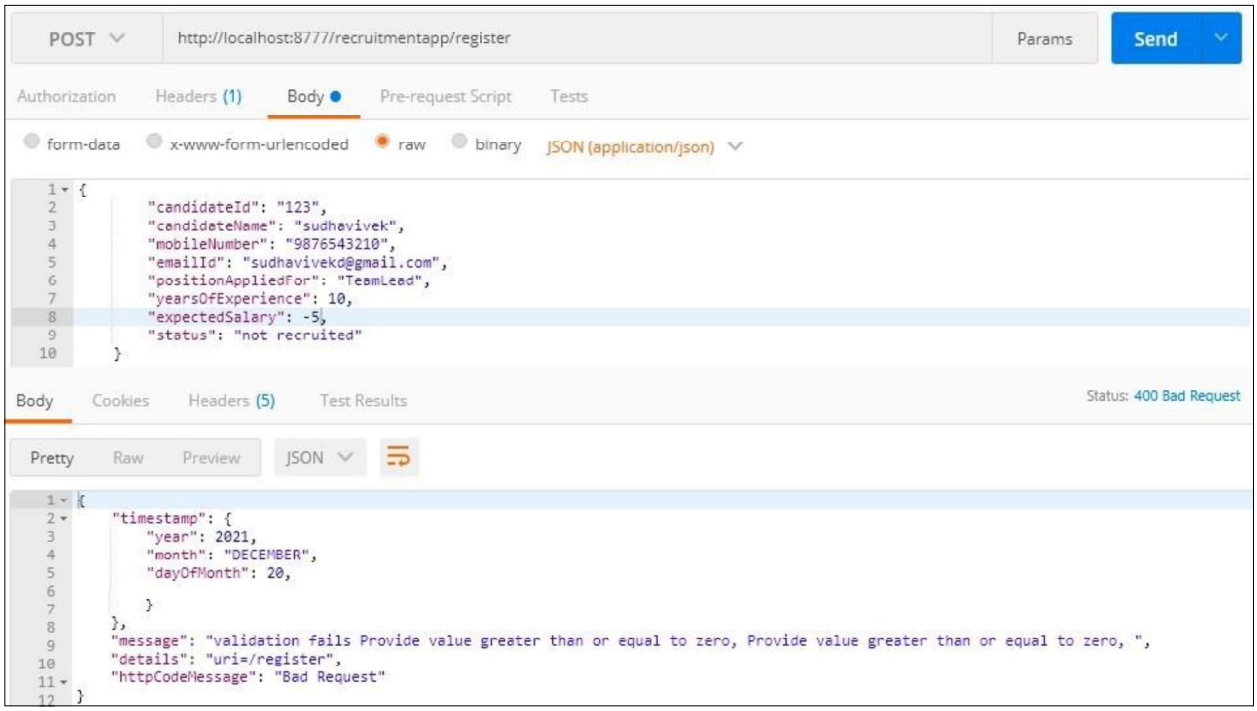
6.0 Output –Sample Screen shots in postman

1. Rest service for /register with valid data
   URL : http://localhost:8777/recruitmentapp/register
   JSON input: {
   "candidateId": "123",
   "candidateName": "sudhavivek",
   "mobileNumber": "9876543210",
   "emailId": "sudhavivekd@gmail.com",
   "positionAppliedFor": "TeamLead",
   "yearsOfExperience": 10,
   "expectedSalary": 500000,
   "status": "not recruited"
   }



2. Rest service for /register with invalid data
   URL : http://localhost:8777/recruitmentapp/register
   JSON input: {
   "candidateId": "123",

"candidateName": "sudhavivek",
"mobileNumber": "9876543210",
"emailId": "sudhavivekd@gmail.com",
"positionAppliedFor": "TeamLead",
"yearsOfExperience": 10,
"expectedSalary": -5,
"status": "not recruited"
}



3. Rest Service for view
URL : http://localhost:8777/recruitmentapp/view

JSON Output:

{

"HR": [

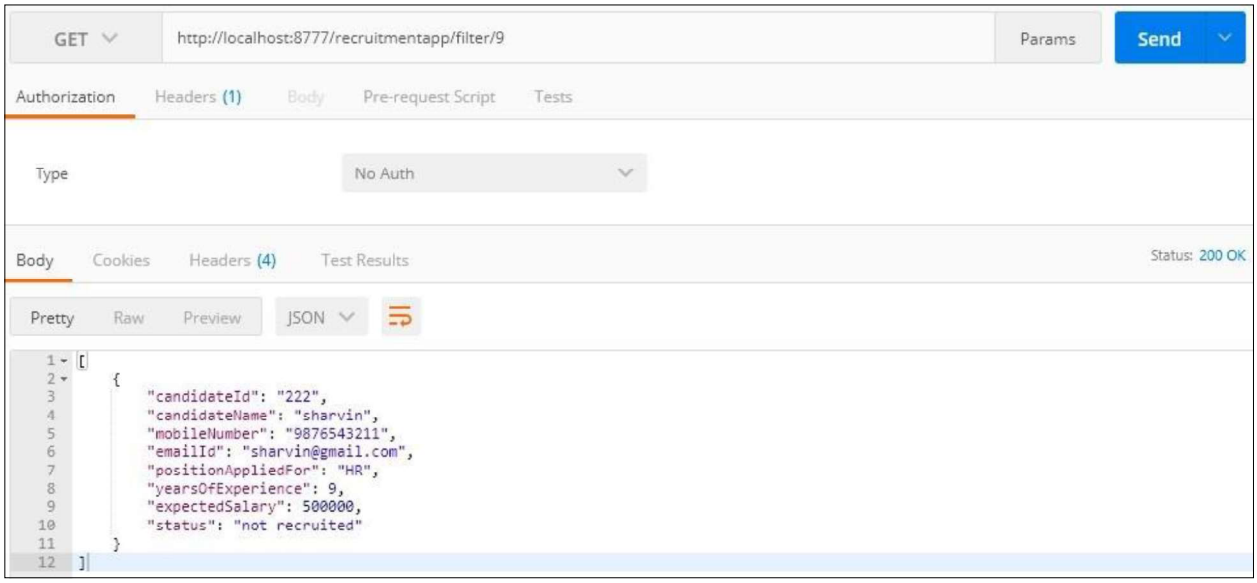
{

"candidateId": "222",

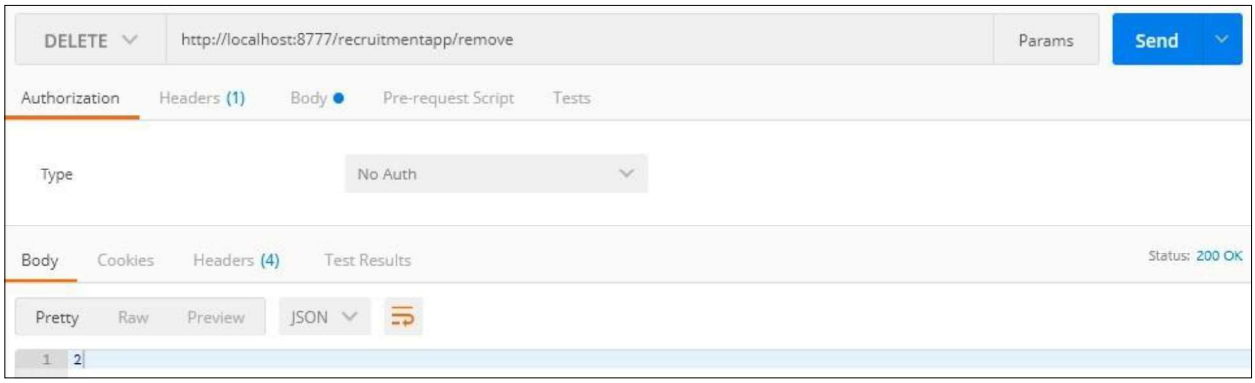
```
      "candidateName": "sharvin",
      "mobileNumber": "9876543211",
      "emailId": "sharvin@gmail.com",
      "positionAppliedFor": "HR",
      "yearsOfExperience": 9,
      "expectedSalary": 500000,
      "status": "not recruited"
    },
    {
      "candidateId": "322",
      "candidateName": "vishahan",
      "mobileNumber": "9876593411",
      "emailId": "vishahan@gmail.com",
      "positionAppliedFor": "HR",
      "yearsOfExperience": 10,
      "expectedSalary": 500000,
      "status": "recruited"
    }
  ],
  "TeamLead": [
    {
      "candidateId": "123",
      "candidateName": "sudhavivek",
      "mobileNumber": "9876543210",
      "emailId": "sudhavivekd@gmail.com",
      "positionAppliedFor": "TeamLead",
      "yearsOfExperience": 10,
      "expectedSalary": 500000,
      "status": "not recruited"
```

```
    }
  ],
  "Manager": [
    {
      "candidateId": "111",
      "candidateName": "sudhac",
      "mobileNumber": "9876543210",
      "emailId": "sudha@gmail.com",
      "positionAppliedFor": "Manager",
      "yearsOfExperience": 10,
      "expectedSalary": 500000,
      "status": "not recruited"
    },
    {
      "candidateId": "444",
      "candidateName": "arav",
      "mobileNumber": "9876593211",
      "emailId": "arav@gmail.com",
      "positionAppliedFor": "Manager",
      "yearsOfExperience": 10,
      "expectedSalary": 500000,
      "status": "recruited"
    }
  ]
}
```

4. Rest Service for filter
   URL: http://localhost:8777/recruitmentapp/filter/9



5. Rest Service for delete
   URL: http://localhost:8777/recruitmentapp/remove



PART -II

INTER MICROSERVICE CONFIGURATION

The information about the agency is provided by the RecruitmentGreeting microservice. The RecruitmentGreeting microservice has the below service method. This service method is present in the GreetingController class

Request Method	Request Url	Method in Controller	Description
Get	/welcome	greeting	This service will display the information about the agency.

The solution for the RecruitmentGreeting is provided as part of the code skeleton, you just need to use this microservice in RecruitmentApplication microservice.

The RecruitmentApplication microservice is responsible for using the welcome service present in the RecruitmentGreeting microservice. The RecruitmentApplication has the below service. This service is present in the ApplicationController which is a RestController.

Request Method	Request Url	Method in Controller	Description
Get	/greet	retreiveInfo	This service will display the information about the restaurant by invoking the welcome service of the RecruitmentGreeting Microservice. Note: Use inter microservice communication – Feign Client

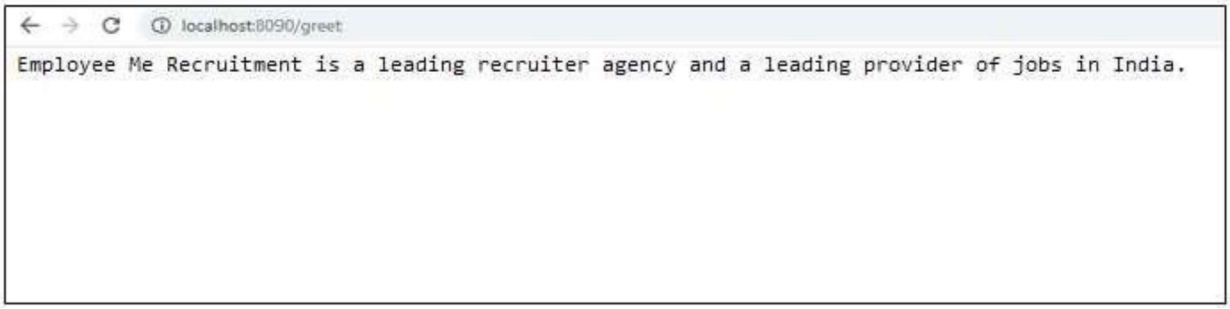
The below proxy interface named ServiceProxy is provided in the RecruitmentApplication Microservice. Use appropriate annotation in this proxy for performing inter microservice communication.

The ServiceProxy should be injected in the ApplicationController. Multiple clients will be accessing the greet service present in the RecruitmentApplication Microservice, so ensure the load balancing is done accordingly using client side load balancer – Ribbon.

When the client invokes the greet service, there can be possibilities where the remote service(welcome service of RecruitmentGreeting microservice) might not be available. Handle this failure by using circuit breaker. Also use fallback method in the circuit breaker to display the appropriate message.

Output –Sample Screen shots in postman

1. **Rest service for /greet**



2. Rest service for /greet when the /welcome service is unavailable in RecruitmentGreeting Microservice

7.0 Deliverables

Use the below code skeleton for implementation. Do not change or delete the class/method names or return types which are provided to you as a part of the code skeleton.


RecruitmentRouter.zip


RecruitmentAgency.zip


RecruitmentRegistry.zip


RecruitmentApplication.zip


Recruitmentgreeting.zip