**WEEK 5:**

* **Mandatory Hands-on-**

**Q. Creating Microservices for account and loan :**

**For account service create:**

1. **AccountApplication.java**

package com.cognizant.account;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class AccountApplication {

public static void main(String[] args) {

SpringApplication.run(AccountApplication.class, args);

}

}

1. **AccountController.java**

package com.cognizant.account.controller;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.RestController;

import java.util.Map;

@RestController

public class AccountController {

@GetMapping("/accounts/{number}")

public Map<String, Object> getAccountDetails(@PathVariable String number) {

return Map.of(

"number", number,

"type", "savings",

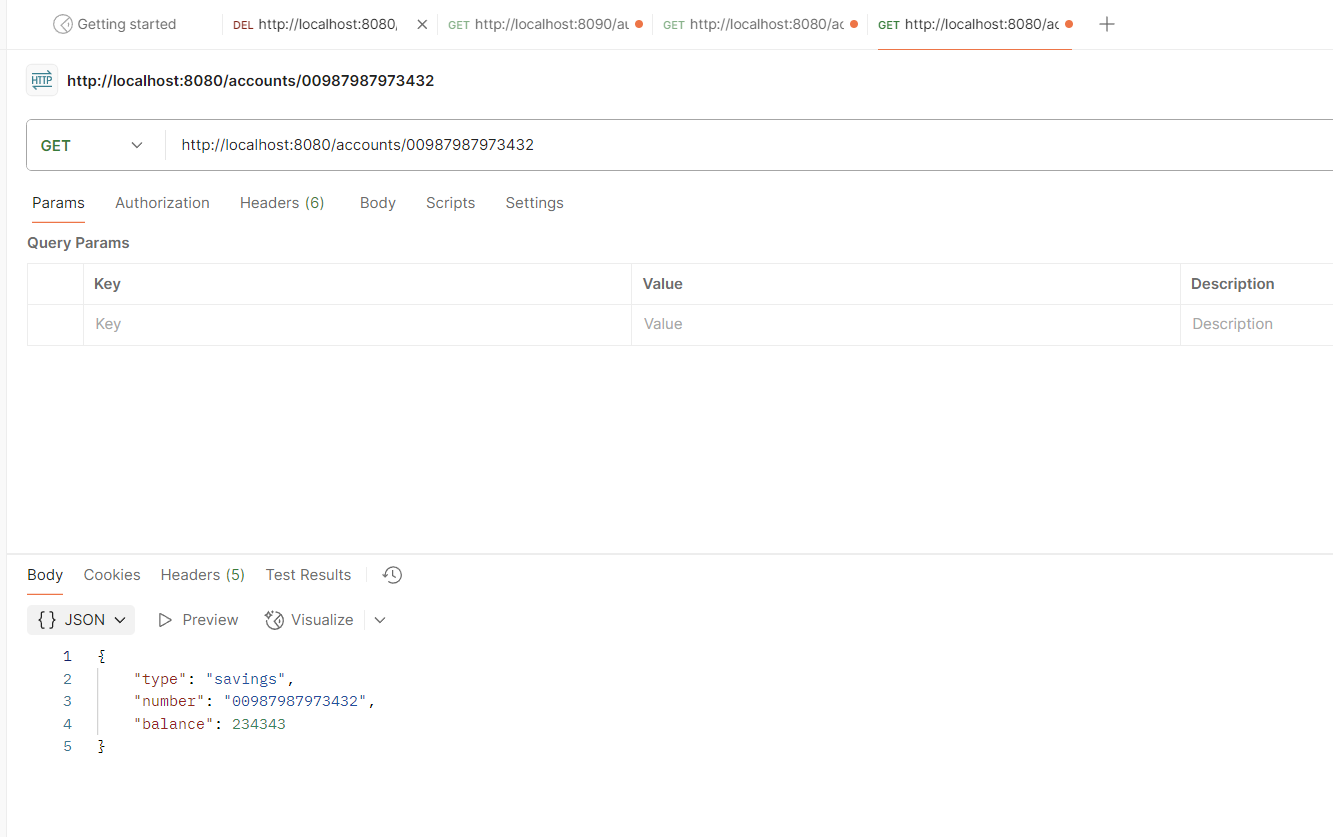
"balance", 234343

);

}

}

* **OUTPUT**:



**For loan service create:**

1. **LoanApplication.java**

package com.cognizant.loan;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class LoanApplication {

public static void main(String[] args) {

SpringApplication.run(LoanApplication.class, args);

}

}

1. **LoanController.java**

package com.cognizant.loan.controller;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.RestController;

import java.util.Map;

@RestController

public class LoanController {

@GetMapping("/loans/{number}")

public Map<String, Object> getLoanDetails(@PathVariable String number) {

return Map.of(

"number", number,

"type", "car",

"loan", 400000,

"emi", 3258,

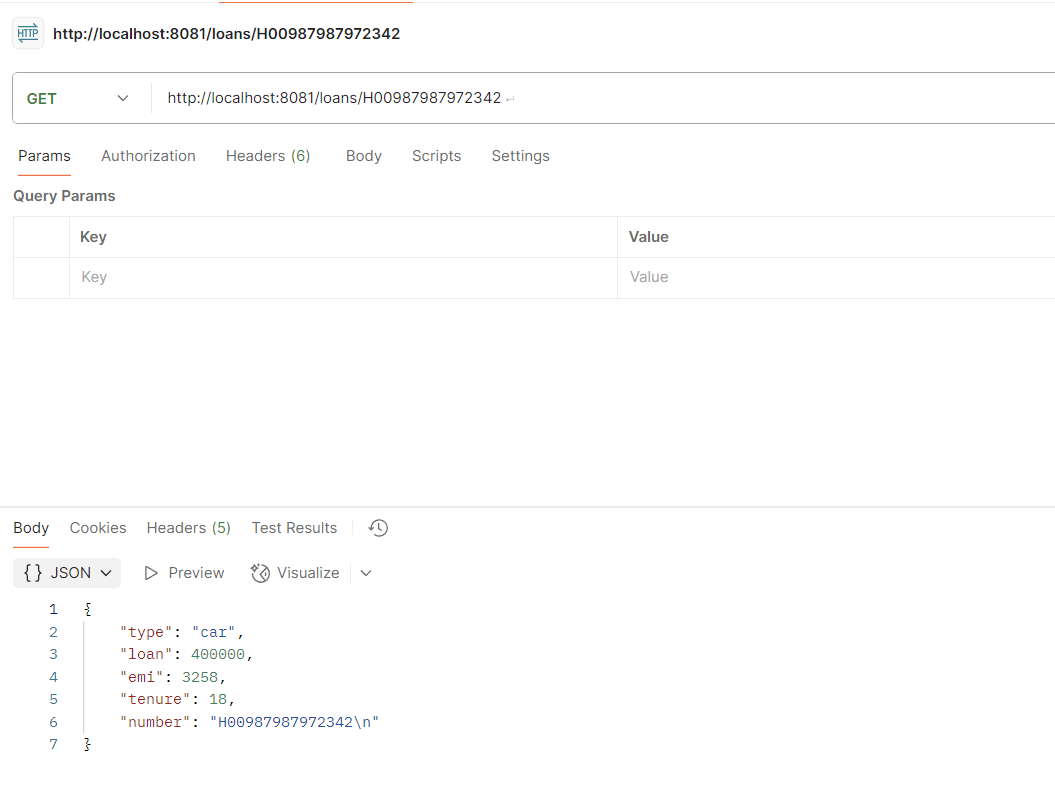
"tenure", 18

);

}

}

* **OUTPUT:**



**Thus,**

**AccountApplication.java** 🡪 Runs on 8080

**LoanApplication.java** 🡪 Runs on 8081

creating two different services.

* **Additional Hands-on-**

**Q. Create Eureka Discovery Server and register microservices-**

**Step 1: Create Eureka Discovery Server**

**1.1 Generate Eureka Discovery Server**

Add dependency:

Spring Cloud Discovery → Eureka Server

**application.properties**

server.port=8761

eureka.client.register-with-eureka=false

eureka.client.fetch-registry=false

logging.level.com.netflix.eureka=OFF

logging.level.com.netflix.discovery=OFF

**In EurekaDiscoveryServerApplication.java:**

@SpringBootApplication

@EnableEurekaServer

public class EurekaDiscoveryServerApplication {

public static void main(String[] args) {

SpringApplication.run(EurekaDiscoveryServerApplication.class, args);

}

}

Run the main class → [**http://localhost:8761**](http://localhost:8761)

**Step 2: Register Account & Loan Microservices**

We already have **Account** and **Loan** services.

Now we make them **Eureka Clients**.

**2.1 Modify Account Microservice**

**a) Update Dependencies**

* + Dependencies:
    - **Spring Web**
    - **Spring Boot DevTools**
    - **Eureka Discovery Client**

**b) Enable Eureka Client**

**In AccountApplication.java:**

@SpringBootApplication

@EnableDiscoveryClient

public class AccountApplication {

public static void main(String[] args) {

SpringApplication.run(AccountApplication.class, args);

}

}

**c) Add Eureka Config**

**Inside src/main/resources/application.properties:**

spring.application.name=account

server.port=8080

eureka.client.service-url.defaultZone=http://localhost:8761/eureka/

**d) Run** [**http://localhost:8761**](http://localhost:8761)

**2.2 Modify Loan Microservice**

* Add Eureka Discovery Client dependency
* @EnableDiscoveryClient in LoanApplication.java
* Update application.properties:

spring.application.name=loan-service

server.port=8081

eureka.client.service-url.defaultZone=http://localhost:8761/eureka/

Run **Loan Service** → Refresh Eureka → LOAN-SERVICE registered

Instances currently registered with Eureka

- ACCOUNT

- LOAN

**Step 3: Create API Gateway + Greet Service**

**3.1 Create greet-service**

1. Dependencies:
   * **Spring Web**
   * **Eureka Discovery Client**

**a) Configure greet-service**

application.properties:

spring.application.name=greet-service

server.port=8082

eureka.client.service-url.defaultZone=http://localhost:8761/eureka/

**b) Create Controller**

@RestController

public class GreetController {

@GetMapping("/greet")

public String greet() {

return "Hello World";

}

}

**c) Enable Eureka Client**

**In GreetServiceApplication.java:**

@SpringBootApplication

@EnableDiscoveryClient

public class GreetServiceApplication {

public static void main(String[] args) {

SpringApplication.run(GreetServiceApplication.class, args);

}

}

Run it → registers as GREET-SERVICE in Eureka.

**3.2 Create Spring Cloud API Gateway**

Dependencies:

* + **Spring Cloud Gateway**
  + **Eureka Discovery Client**

**a) Configure api-gateway**

**application.properties:**

spring.application.name=api-gateway

server.port=9090

eureka.client.service-url.defaultZone=http://localhost:8761/eureka/

spring.cloud.gateway.discovery.locator.enabled=true

spring.cloud.gateway.discovery.locator.lower-case-service-id=true

**b) Enable Eureka Client**

**In ApiGatewayApplication.java:**

@SpringBootApplication

@EnableDiscoveryClient

public class ApiGatewayApplication {

public static void main(String[] args) {

SpringApplication.run(ApiGatewayApplication.class, args);

}

}

**c) Run API Gateway**

It will as API-GATEWAY.

**3.3 Test greet-service via Gateway**

<http://localhost:9090/greet-service/greet>

Hello World