**WEEK-5**

**MICROSERVICES**

**Superset ID: 6419740**

**Creating Microservices for account and loan**

**CODE:**

**AccountController.java:**

package com.cognizant.account.controller;

import org.springframework.web.bind.annotation.\*;

import java.util.\*;

@RestController

@RequestMapping("/accounts")

public class AccountController {

@GetMapping("/{number}")

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

Map<String, Object> response = new HashMap<>();

response.put("number", number);

response.put("type", "savings");

response.put("balance", 234343);

return response;

}

}

**LoanController.java:**

package com.cognizant.loan.controller;

import org.springframework.web.bind.annotation.\*;

import java.util.\*;

@RestController

@RequestMapping("/loans")

public class LoanController {

@GetMapping("/{number}")

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

Map<String, Object> response = new HashMap<>();

response.put("number", number);

response.put("type", "car");

response.put("loan", 400000);

response.put("emi", 3258);

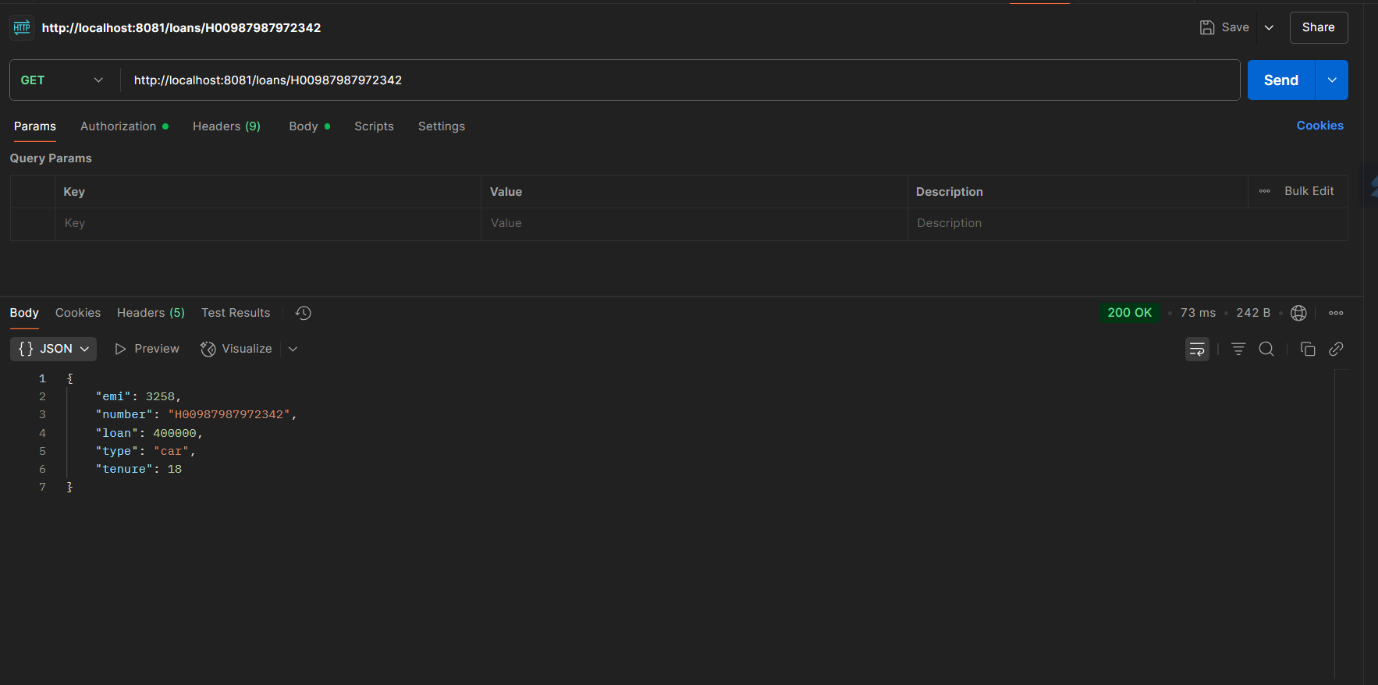
response.put("tenure", 18);

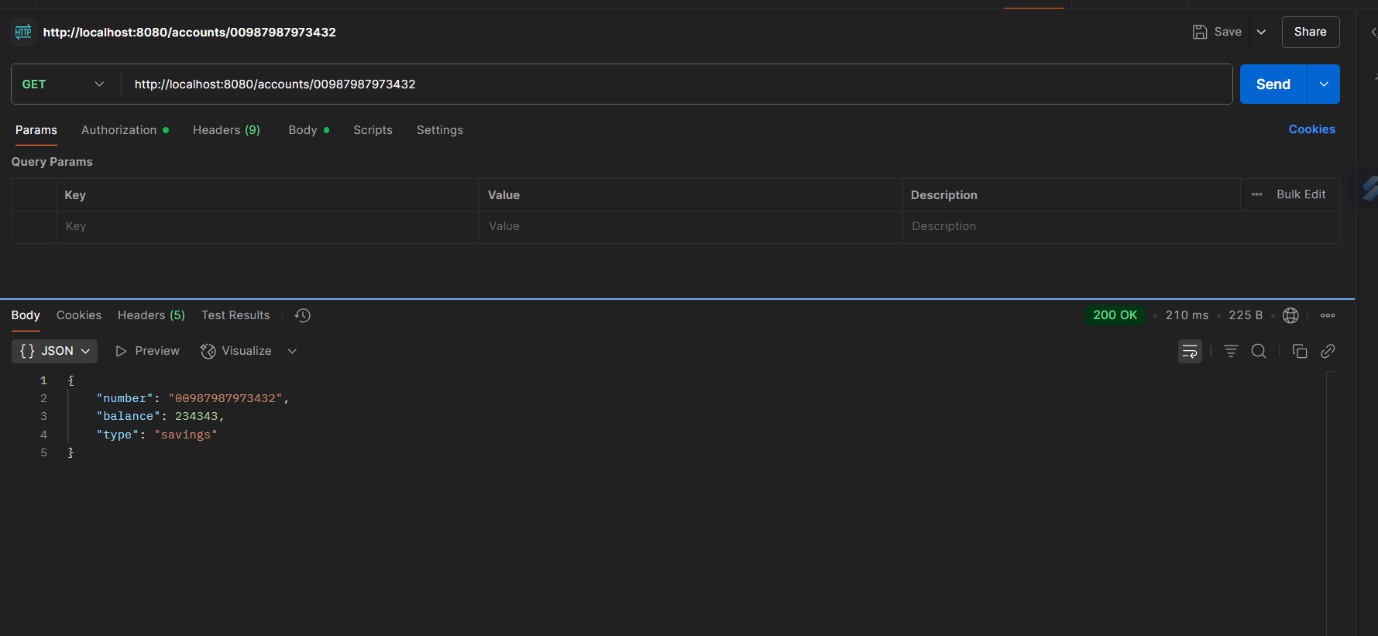
return response;

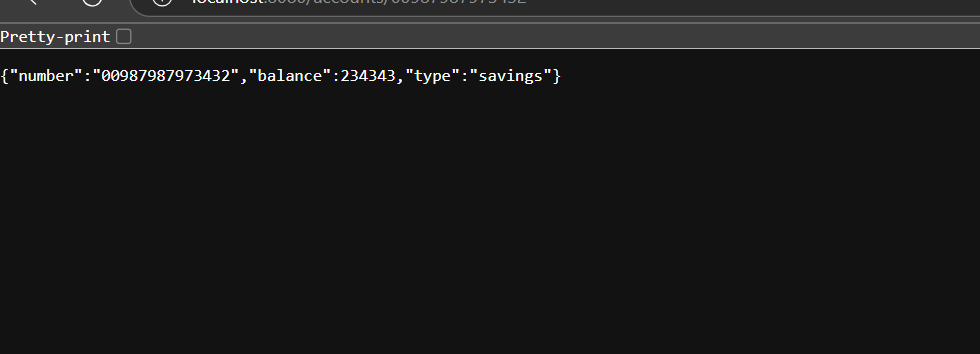
}

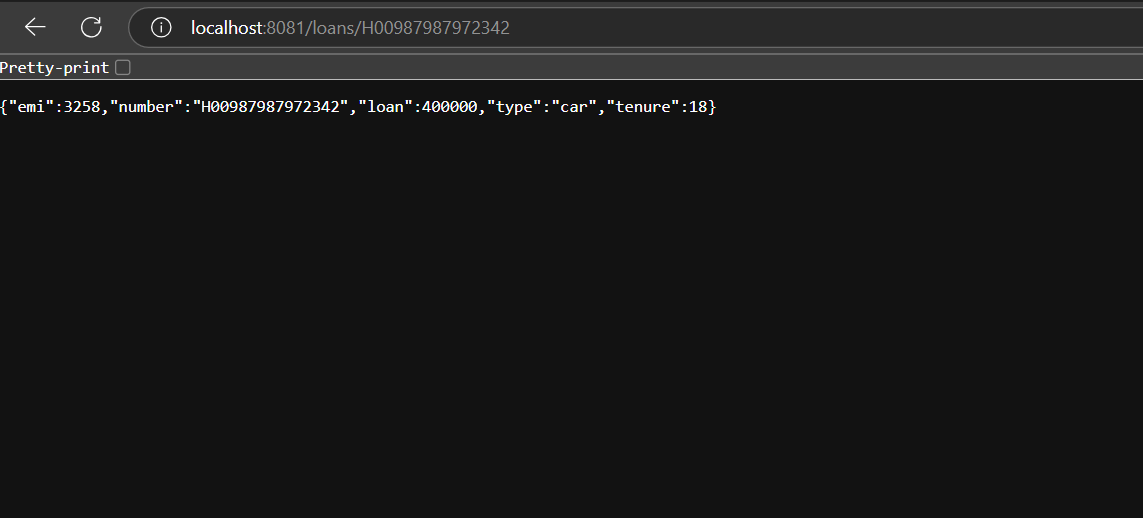
}

**OUTPUT:**

****

****

****

****

**Create Eureka Discovery Server and register microservices**

**CODE:**

**EurekaDiscoveryServerApplication.java:**

package com.cognizant.eureka\_discovery\_server;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@SpringBootApplication

@EnableEurekaServer

public class EurekaDiscoveryServerApplication {

    public static void main(String[] args) {

        SpringApplication.run(EurekaDiscoveryServerApplication.class, args);

    }

}

**AccountController.java:**

package com.cognizant.account.controller;

import org.springframework.web.bind.annotation.\*;

import java.util.\*;

@RestController

@RequestMapping("/accounts")

public class AccountController {

    @GetMapping("/{number}")

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

        Map<String, Object> response = new HashMap<>();

        response.put("number", number);

        response.put("type", "savings");

        response.put("balance", 234343);

        return response;

    }

}

**LoanController.java:**

package com.cognizant.loan.controller;

import org.springframework.web.bind.annotation.\*;

import java.util.\*;

@RestController

@RequestMapping("/loans")

public class LoanController {

    @GetMapping("/{number}")

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

        Map<String, Object> response = new HashMap<>();

        response.put("number", number);

        response.put("type", "car");

        response.put("loan", 400000);

        response.put("emi", 3258);

        response.put("tenure", 18);

        return response;

    }

}

GreetController.java:

package com.cognizant.greet\_service.controller;

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

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

@RestController

public class GreetController {

    @GetMapping("/greet")

    public String greet() {

        return "Hello World";

    }

}

**LogFilter.java:**

package com.cognizant.api\_gateway.filter;

import org.springframework.cloud.gateway.filter.GlobalFilter;

import org.springframework.cloud.gateway.filter.GatewayFilterChain;

import org.springframework.core.Ordered;

import org.springframework.stereotype.Component;

import org.springframework.web.server.ServerWebExchange;

import reactor.core.publisher.Mono;

@Component

public class LogFilter implements GlobalFilter, Ordered {

    @Override

    public Mono<Void> filter(ServerWebExchange exchange, GatewayFilterChain chain) {

        System.out.println("Incoming request: " + exchange.getRequest().getURI());

        return chain.filter(exchange);

    }

    @Override

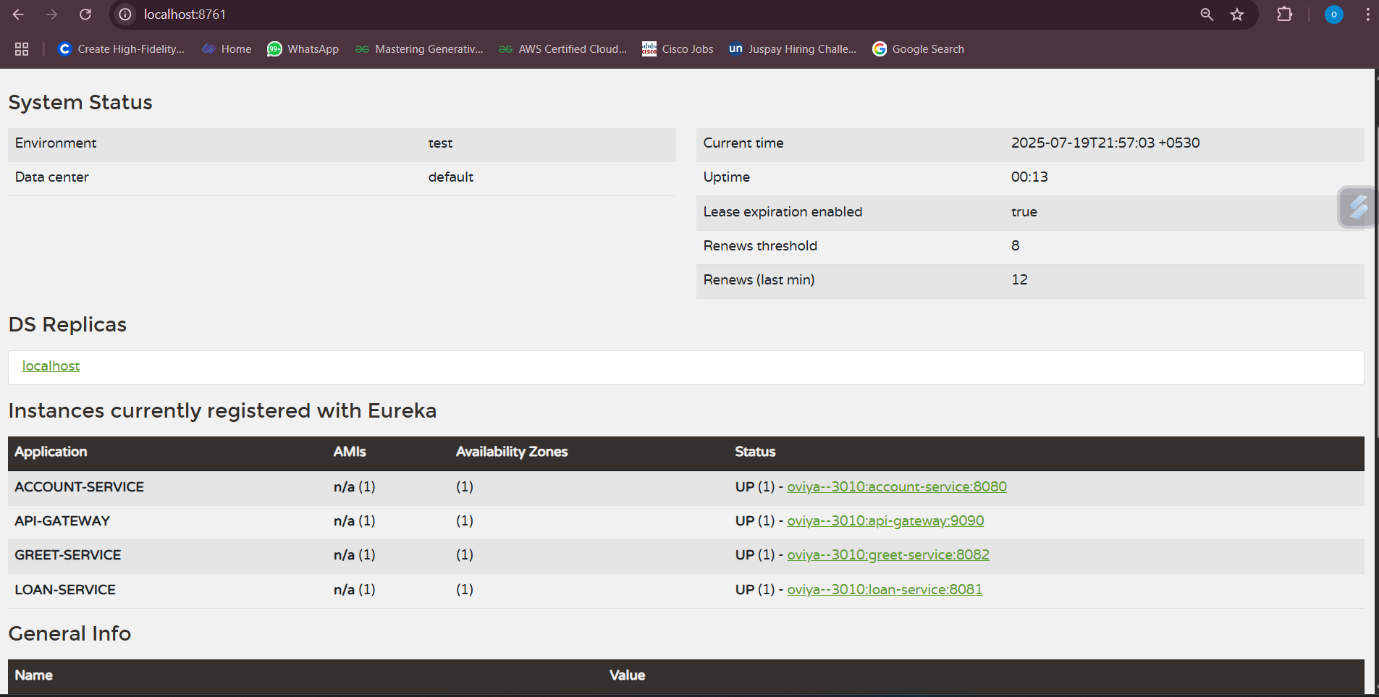
    public int getOrder() {

        return -1; // highest priority

    }

}

**OUTPUT:**

****