**Week – 05**

**Microservices**

**Creating Microservices for account and loan**

**Explanation (for Account Microservice):**

1. Created Maven Spring Boot project named application in Eclipse with dependencies: spring-boot-starter-web and jjwt (for JWT handling).
2. Created REST Controller
   * A REST endpoint /hello was set up using @RestController and @GetMapping.
   * The controller receives a username as a query parameter and returns a JWT token.
3. The JwtUtil class uses the io.jsonwebtoken library to generate JWT tokens with a secret key and expiration time.
4. application.properties was configured (if needed) with server port and other settings like server.port=8081.
5. Built the project using mvn clean install or Eclipse’s Run button.
6. The output was tested by manually entering the URL (<http://localhost:8081/>) and got a JSON response with a JWT token - to verify token generation worked correctly.

**Code:**

**AccountController.java:**

package com.cognizant.account.controller;

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

import java.util.\*;

*@RestController*

public class AccountController {

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

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

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

account.put("number", number);

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

account.put("balance", 234343);

return account;

}

}

**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);

}

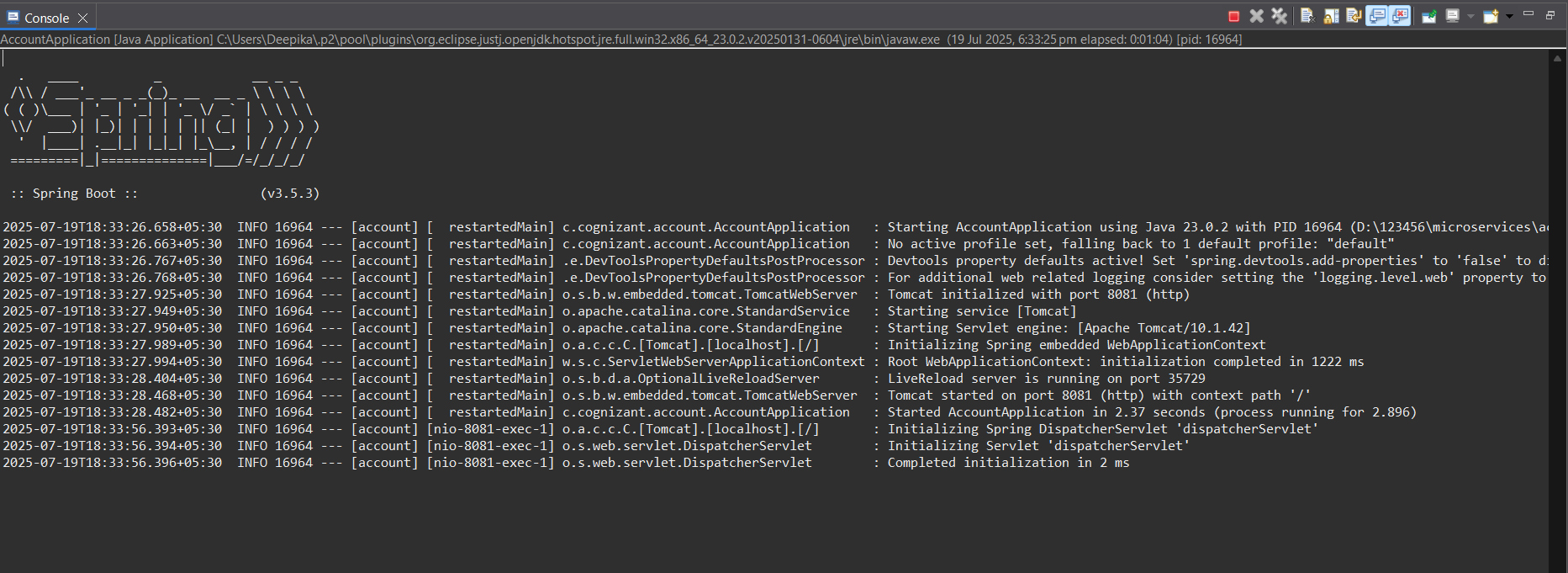
}

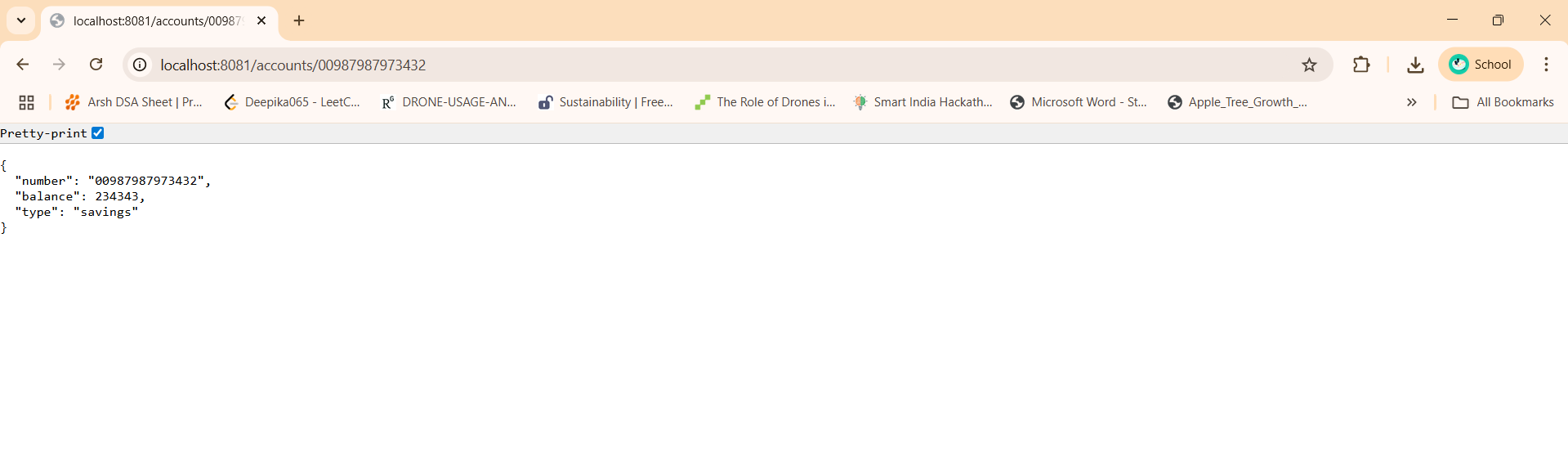
**application.properties:**

spring.application.name=account

server.port=8081

**Output:**

****

****

**Explanation (for Loan Microservice):**

1. Created a second Maven Spring Boot project named loan with spring-boot-starter-web and jjwt dependencies.
2. **Created REST Controller**
   * Defined a secured endpoint /loan using @RestController.
   * Used @RequestHeader to extract the Authorization token from the request header.
3. The same JwtUtil logic was reused to validate the JWT token passed from the application service.
4. **Implemented Token Validation Logic**
   * Inside LoanController, JwtUtil.validateToken(token) is called to check if the token is valid.
5. Made a request to <http://localhost:8082/loan> - If valid, returns loan info. If invalid or expired, returns an appropriate error message.

**Code:**

**LoanController.java:**

package com.cognizant.loan.controller;

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

import java.util.\*;

*@RestController*

public class LoanController {

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

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

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

loan.put("number", number);

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

loan.put("loan", 400000);

loan.put("emi", 3258);

loan.put("tenure", 18);

return loan;

}

}

**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);

}

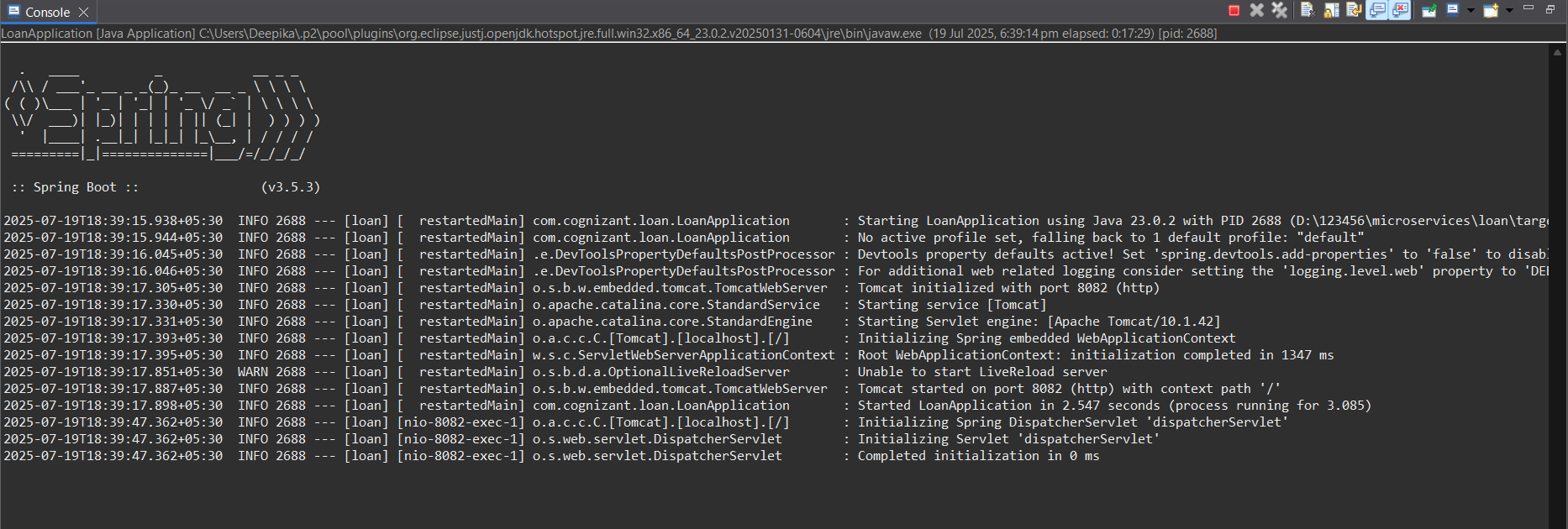
}

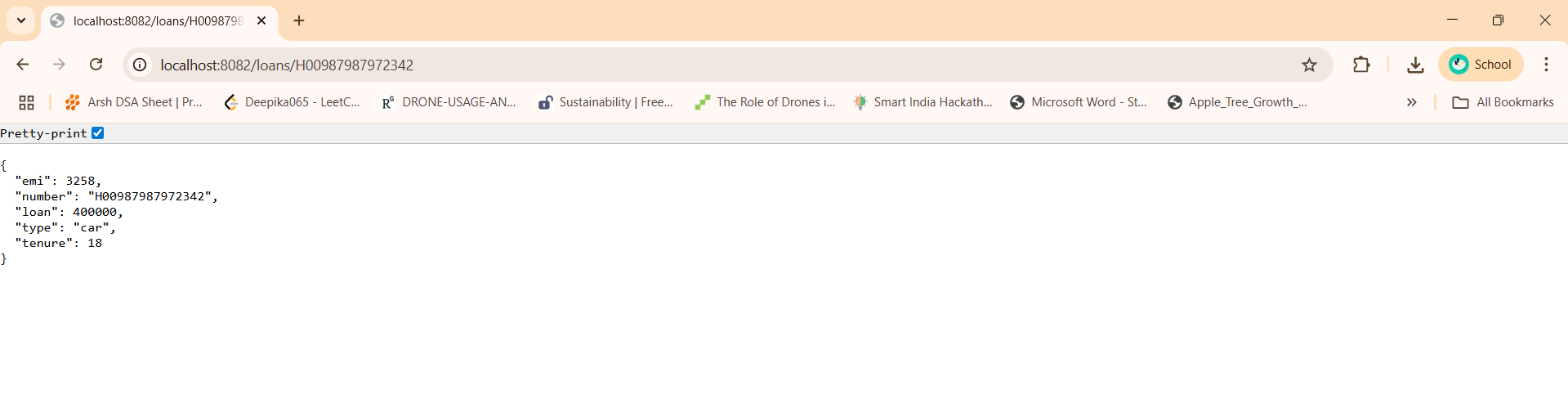
**application.properties:**

spring.application.name=loan

server.port=8082

**Output:**

****

****