**Week5\_Creating Microservices for account and loan**

### Phase 1: Setting Up the Development Environment

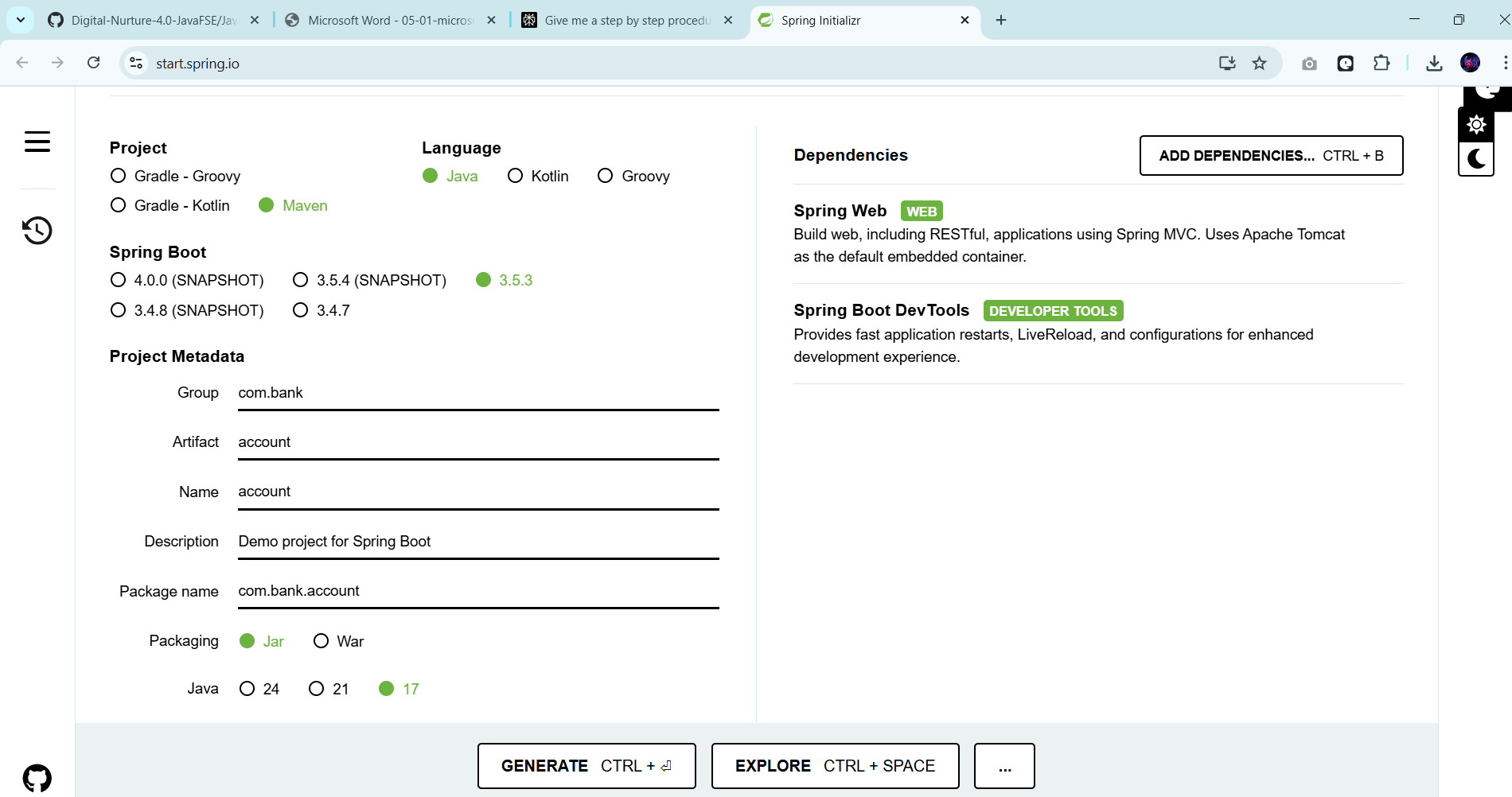
* This is a one-time setup to prepare your machine for Java and Spring development.
* Eclipse IDE for Enterprise Java and Web Developers: Installed to serve as the main coding environment.

### Phase 2: Building the account Microservice (Port 8080)

This service provides a dummy response for account information.

#### Step 1: Project Generation

* The project was created using Spring Initializr (start.spring.io) with the following settings:
* Group: com.cognizant
* Artifact: account
* Dependencies: Spring Web, Spring Boot DevTools



#### Step 2: Code Implementation

The following two Java classes were created inside the com.cognizant.account package.

**1. Account.java (Data Model)** - This class defines the structure for the account data.

// File: account/src/main/java/com/cognizant/account/Account.java

package com.cognizant.account;

public class Account {

private String number;

private String type;

private double balance;

public Account(String number, String type, double balance) {

this.number = number;

this.type = type;

this.balance = balance;

}

// Getters and setters...

public String getNumber() { return number; }

public void setNumber(String number) { this.number = number; }

public String getType() { return type; }

public void setType(String type) { this.type = type; }

public double getBalance() { return balance; }

public void setBalance(double balance) { this.balance = balance; }

}

**2. AccountController.java (API Endpoint) -** This class handles web requests and returns the dummy account data.

//File:account/src/main/java/com/cognizant/account/AccountController.jav

package com.cognizant.account;

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

@RestControllerpublic class AccountController {

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

public Account getAccountDetails(@PathVariable String number) {

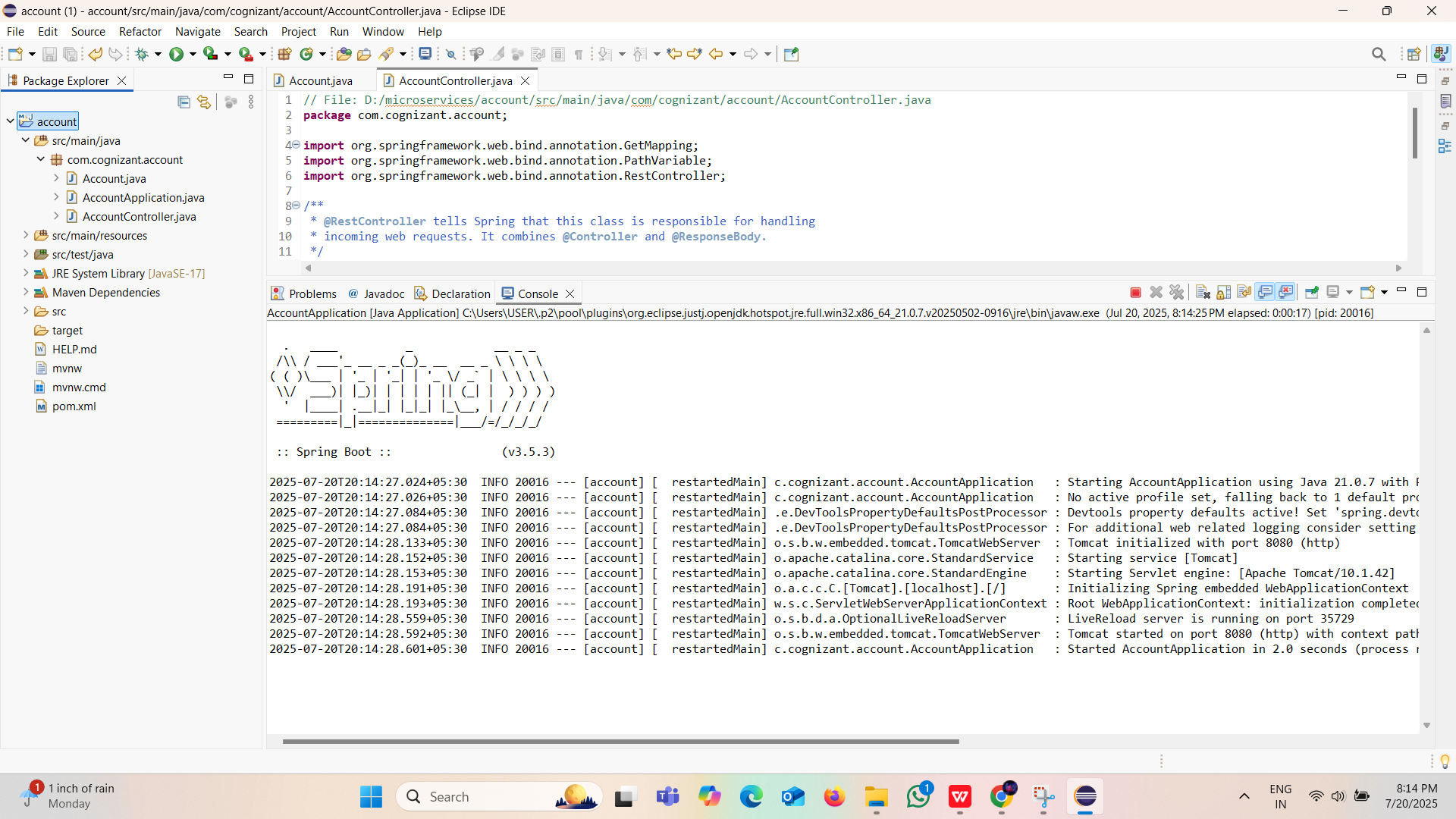
return new Account("00987987973432", "savings", 234343);

}

}

#### Step 3: Configuration

* The configuration file account/src/main/resources/application.properties was left empty. This causes Spring Boot to use the default web server port, which is 8080.

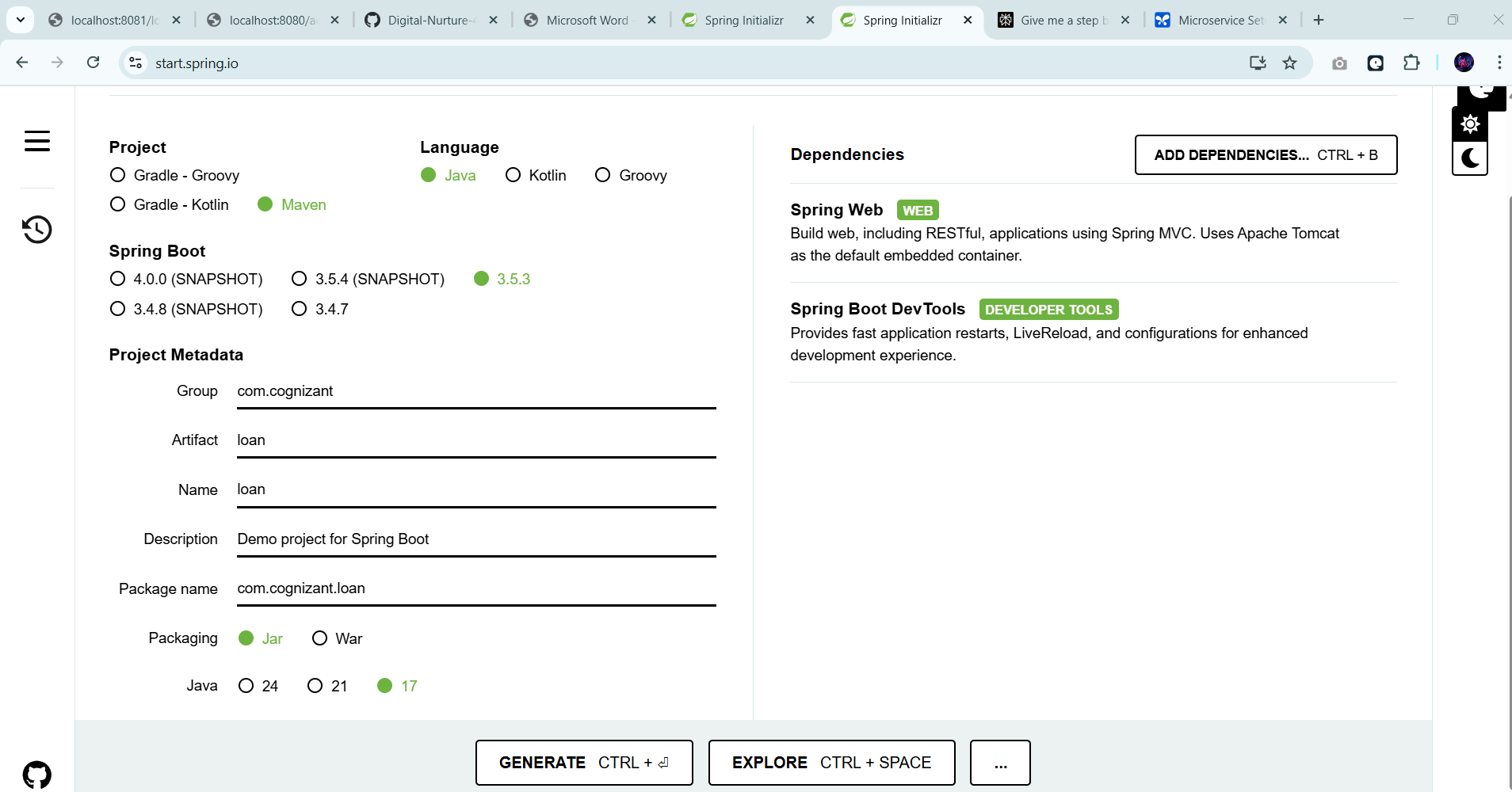


### Phase 3: Building the loan Microservice (Port 8081)

This service provides a dummy response for loan information and is configured to avoid conflicting with the account service.

#### Step 1: Project Generation

* A new project was created using Spring Initializr with these settings:
* Group: com.cognizant
* Artifact: loan
* Dependencies: Spring Web, Spring Boot DevTools



#### Step 2: Code Implementation

The following two Java classes were created inside the com.cognizant.loan package.

**1. Loan.java (Data Model) -** This class defines the structure for the loan data.

// File: loan/src/main/java/com/cognizant/loan/Loan.java

package com.cognizant.loan;

public class Loan {

private String number;

private String type;

private long loan;

private long emi;

private int tenure;

public Loan(String number, String type, long loan, long emi, int tenure) {

this.number = number;

this.type = type;

this.loan = loan;

this.emi = emi;

this.tenure = tenure;

}

// Getters and setters...

public String getNumber() { return number; }

public void setNumber(String number) { this.number = number; }

public String getType() { return type; }

public void setType(String type) { this.type = type; }

public long getLoan() { return loan; }

public void setLoan(long loan) { this.loan = loan; }

public long getEmi() { return emi; }

public void setEmi(long emi) { this.emi = emi; }

public int getTenure() { return tenure; }

public void setTenure(int tenure) { this.tenure = tenure; }

}

**2. LoanController.java (API Endpoint) -** This class handles web requests and returns the dummy loan data.

// File: loan/src/main/java/com/cognizant/loan/LoanController.java

package com.cognizant.loan;

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

@RestControllerpublic class LoanController {

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

public Loan getLoanDetails(@PathVariable String number) {

return new Loan("H00987987972342", "car", 400000, 3258, 18);

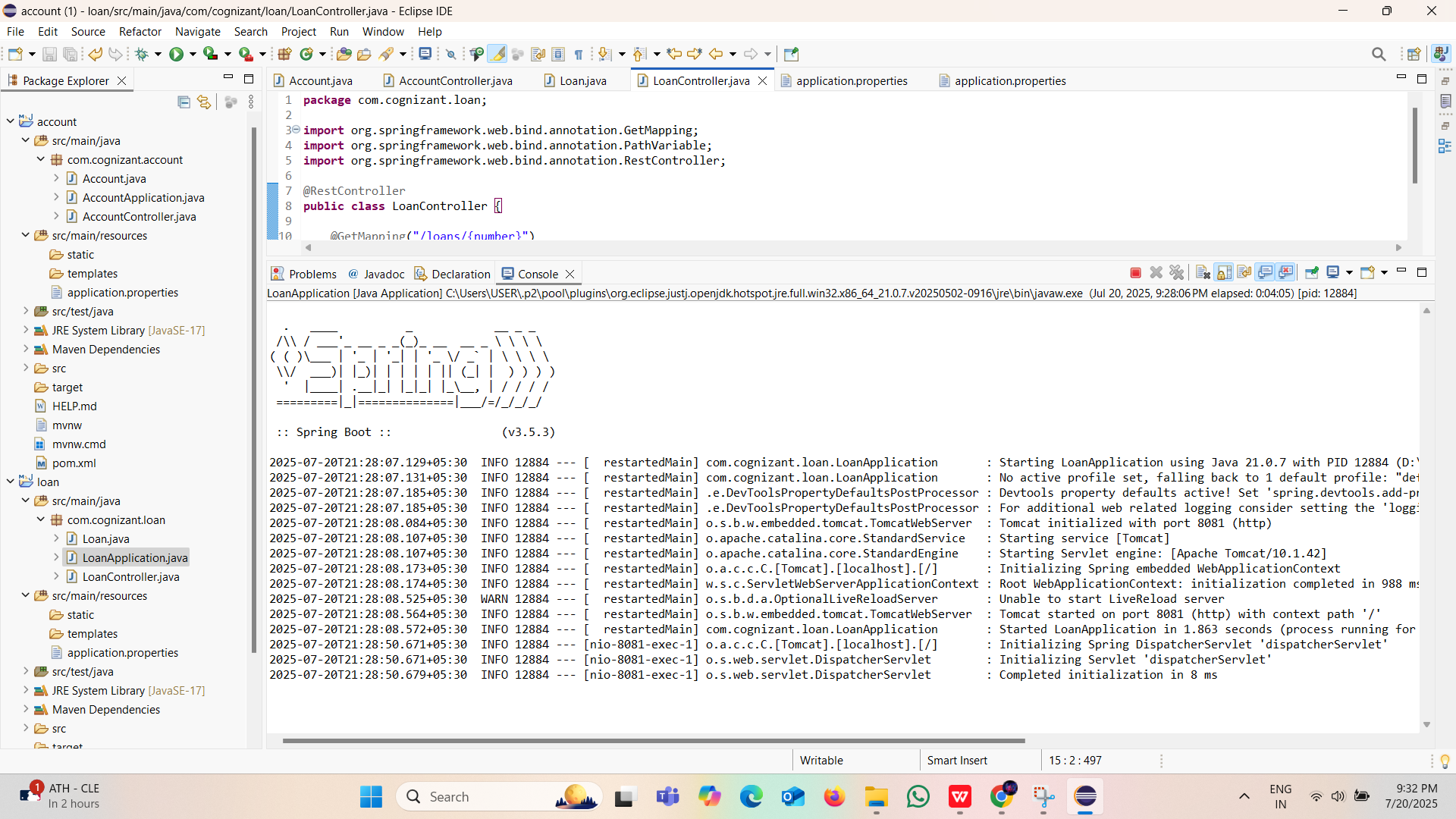
}

}

#### Step 3: Configuration (The Crucial Fix)

To prevent a port conflict with the account service, the configuration file loan/src/main/resources/application.properties was modified to contain the following line:

server.port=8081 # This property forces the loan service to start on port 8081.



### **Phase 4: Final Execution and Testing**

This phase ensures both applications are run correctly and are accessible.

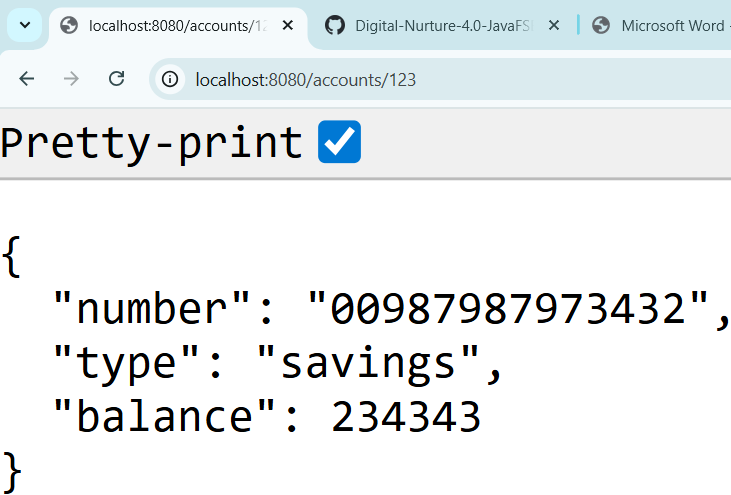
#### Step 1: Running the Applications

* The previous error (SpringBootTestAotProcessor exception) was resolved by running each application correctly:
* Run Account Service: Right-click the AccountApplication.java file -> Run As -> Spring Boot App.
* Run Loan Service: Right-click the LoanApplication.java file -> Run As -> Spring Boot App.

#### Step 2: Verifying the Services

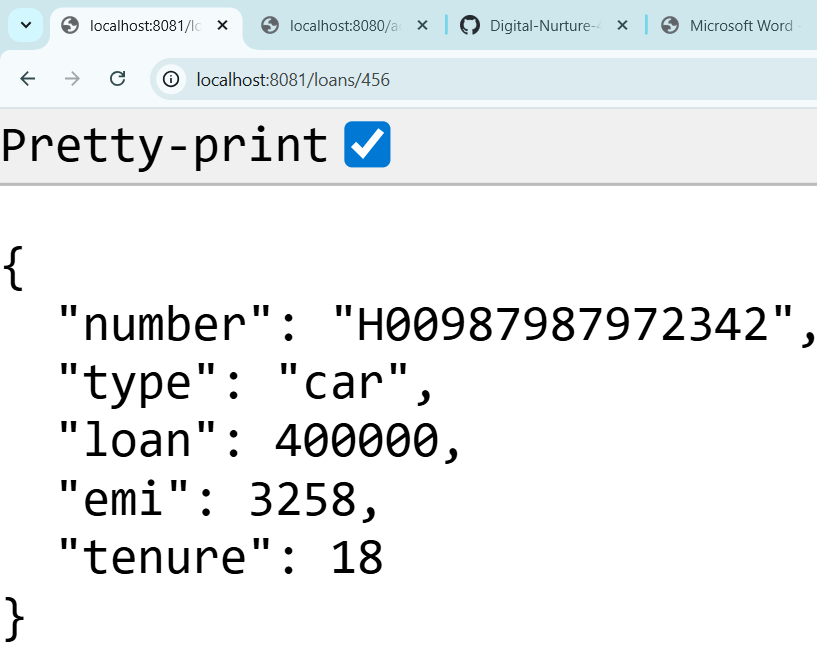
* With both applications running, they can be tested in a web browser:
* Account Service Test:

URL:***http://localhost:8080/accounts/123***

******

* Loan Service Test:

URL: ***http://localhost:8081/loans/456***



### Final Result

You have successfully built and deployed two independent microservices.

The Account Microservice is running on port 8080.

The Loan Microservice is running on port 8081.

Both services are operating simultaneously without conflict, and their separate console logs can be monitored in Eclipse using the "Display Selected Console" button.