

CAPSTONE PROJECT
PRODUCT RETAIL MANAGEMENT
GROUP-E
BATCH-JAVA FULL STACK DEVELOPMENT

Name: Palukuri Chaitanya

E-mail:palukurichaitanya19@gmail.com

Date: September 02,2024

PROBLEM STATEMENT:

- ♦ **A well known retail company wants to strengthen its Middleware by exposing the core logic related to Product Management as Micro services. These middle ware Micro services will be hosted as web app so that all the up/downstream applications can get an access to this for performing business transactions.**
- ♦ **In this project we focus different kind of products avail in stockroom with proper stock values. Products are classified based on product different type of categories. Customer can view those products as well as try to purchase online. But if stock is insufficient then purchase not possible. Otherwise purchase will be occurred and stock is also deducted for this particular product in stockroom. Customer can track the purchased order also by providing proper purchase reference value**

There are 2 Modules/Microservices which have different functionalities

- 1. Customer Module:** A customer can use this module. They can register within system using their personal information and credentials. They can search products from retail house using Id , Name or category. They can purchase product if sufficient stock is available. If they need, they can view their purchased histories and their status. The Customer Micro service has to interact with Product Micro service to get product details before purchasing like stock and price as well as if purchased completed, then update stock of product also using product micro service. Customer Micro service will perform the following functionalities:
 - o A new customer can make registration
 - o A customer can view own purchased product histories
 - o A customer can make online purchase for a product

2. Product Module: Product Micro service is to be called by Customer micro service. This service is used to provide all pre-defined Product details which are already stored in database. Product Micro service will perform the following functionalities:

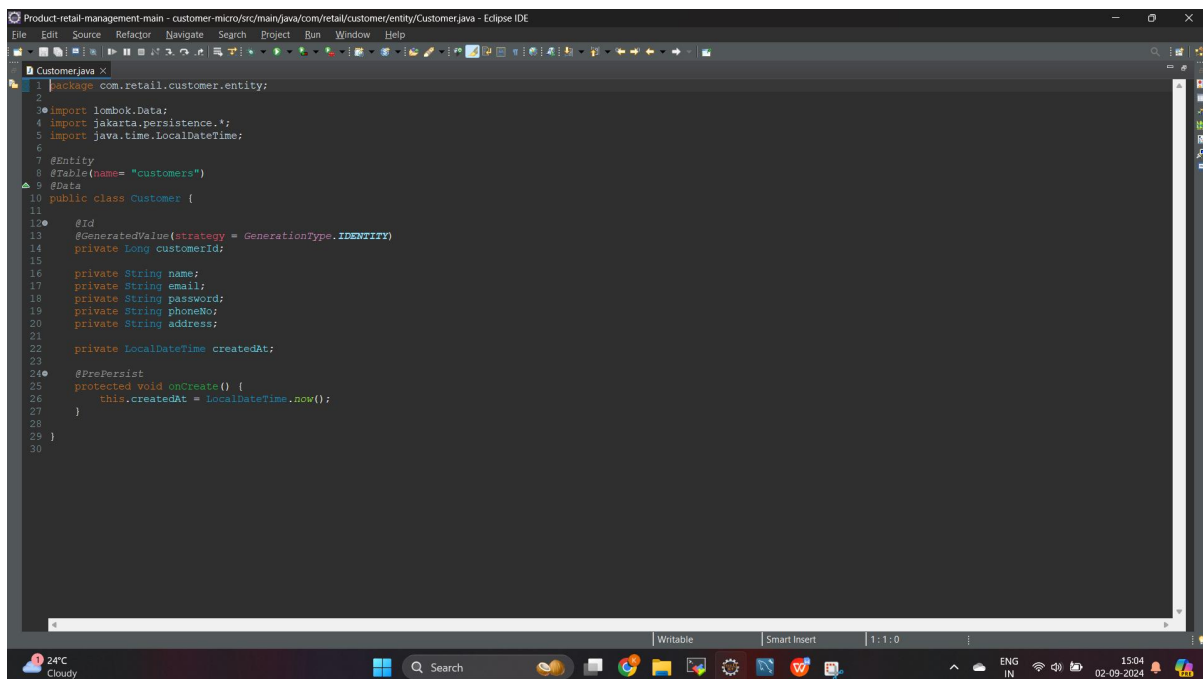
- o View Product details based on different criteria regions like:
- o Based on Product Id
- o Product name
- o Product category
- o Update Product stock based on given id and stock value.

➤ **Customer microservice will have 2 entity class:**

1.Customer:

This table will have following attributes:

Customer.java

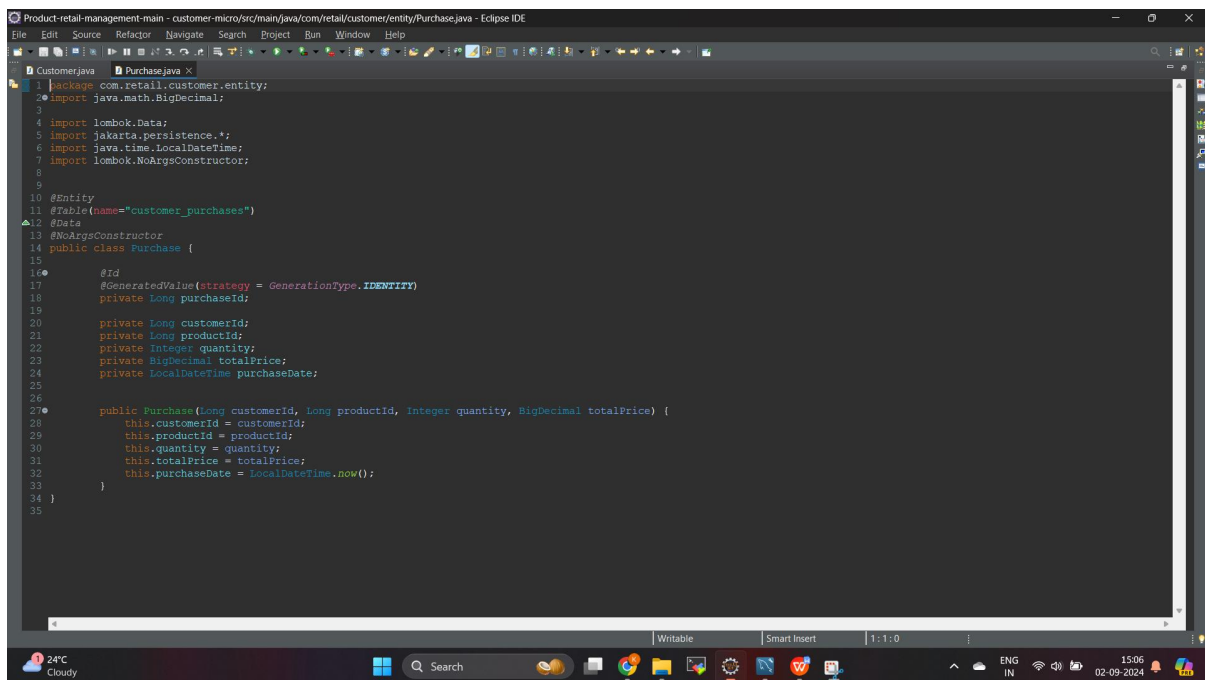


```
1 package com.retail.customer.entity;
2
3 import lombok.Data;
4 import jakarta.persistence.*;
5 import java.time.LocalDateTime;
6
7 @Entity
8 @Table(name = "customers")
9 @Data
10 public class Customer {
11
12     @Id
13     @GeneratedValue(strategy = GenerationType.IDENTITY)
14     private Long customerId;
15
16     private String name;
17     private String email;
18     private String password;
19     private String phoneNo;
20     private String address;
21
22     private LocalDateTime createdAt;
23
24     @PrePersist
25     protected void onCreate() {
26         this.createdAt = LocalDateTime.now();
27     }
28 }
29
30
```

2.Purchase:

This table will have following attributes:

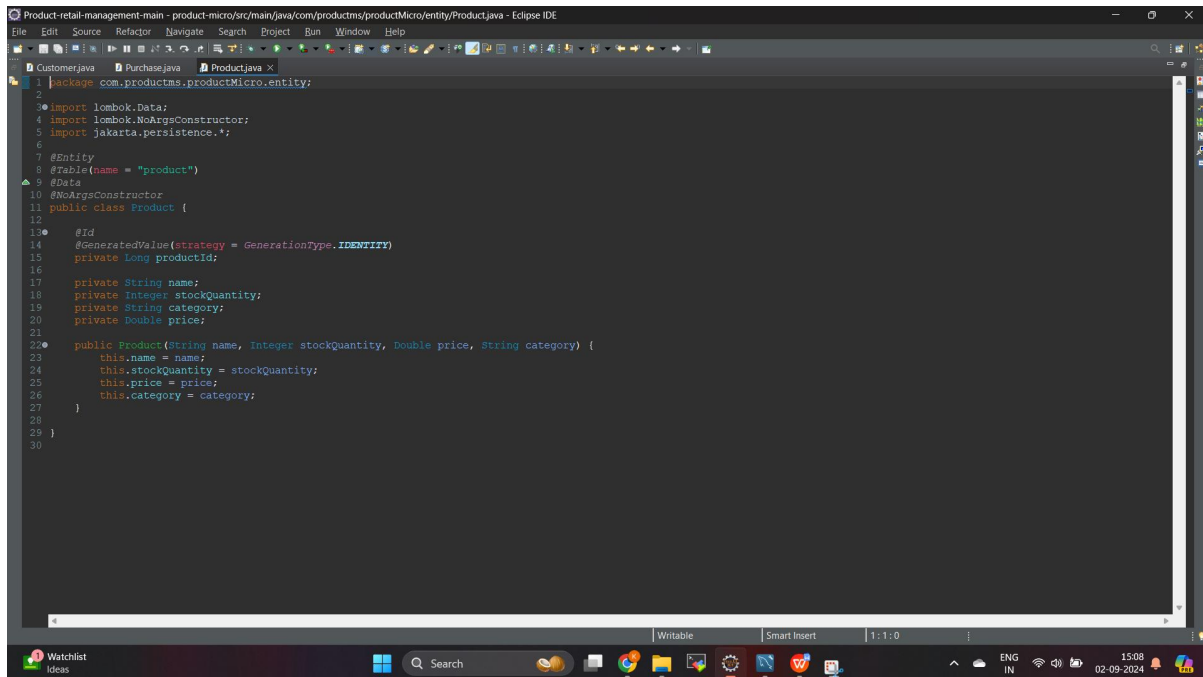
Purchase.java

A screenshot of the Eclipse IDE interface. The main editor window displays the code for Purchase.java. The code includes package declarations, imports for BigDecimal, Lombok annotations, and JPA annotations. It defines a JPA Entity with a table named 'customer_purchases'. The entity has a primary key 'purchaseId' and several other attributes: 'customerId', 'productId', 'quantity', 'totalPrice', and 'purchaseDate'. A constructor is provided that initializes these fields and sets the current date for 'purchaseDate'.

```
1 package com.retail.customer.entity;
2 import java.math.BigDecimal;
3
4 import lombok.Data;
5 import jakarta.persistence.*;
6 import java.time.LocalDateTime;
7 import lombok.NoArgsConstructor;
8
9
10 @Entity
11 @Table(name="customer_purchases")
12 @Data
13 @NoArgsConstructor
14 public class Purchase {
15
16     @Id
17     @GeneratedValue(strategy = GenerationType.IDENTITY)
18     private Long purchaseId;
19
20     private Long customerId;
21     private Long productId;
22     private Integer quantity;
23     private BigDecimal totalPrice;
24     private LocalDateTime purchaseDate;
25
26
27     public Purchase(Long customerId, Long productId, Integer quantity, BigDecimal totalPrice) {
28         this.customerId = customerId;
29         this.productId = productId;
30         this.quantity = quantity;
31         this.totalPrice = totalPrice;
32         this.purchaseDate = LocalDateTime.now();
33     }
34 }
35
```

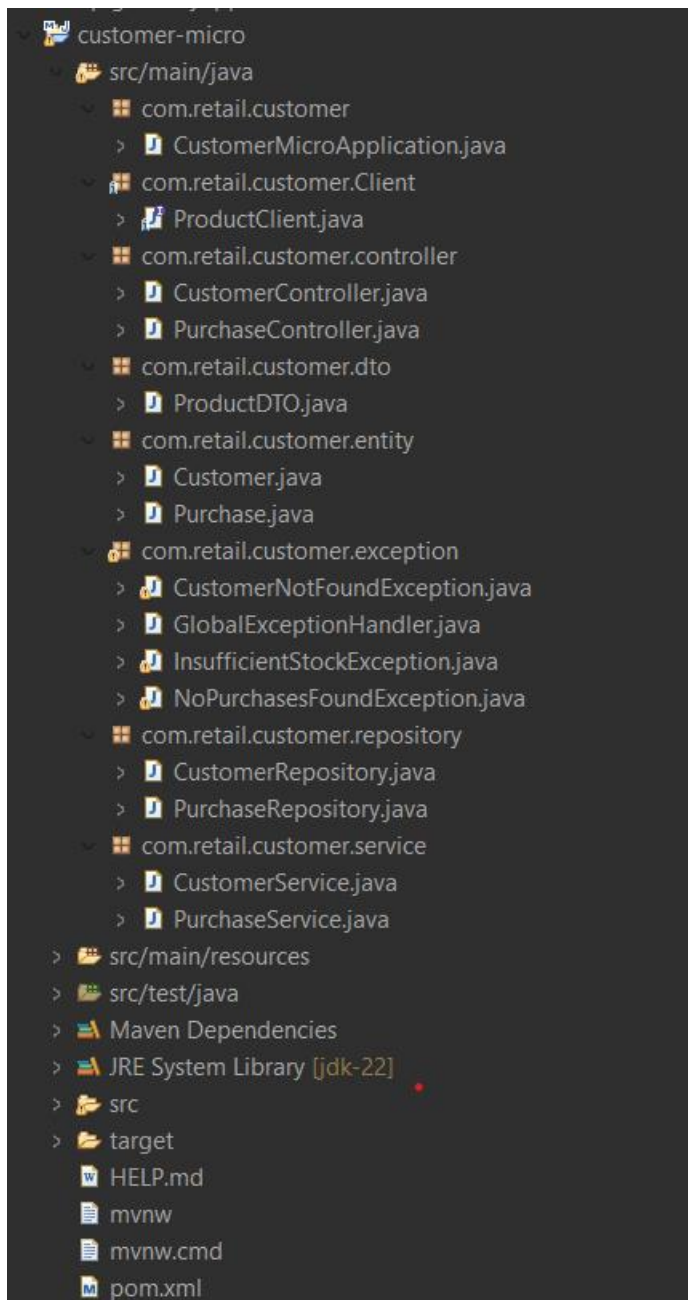
- Product Module will have only 1 entity class:

Product.java

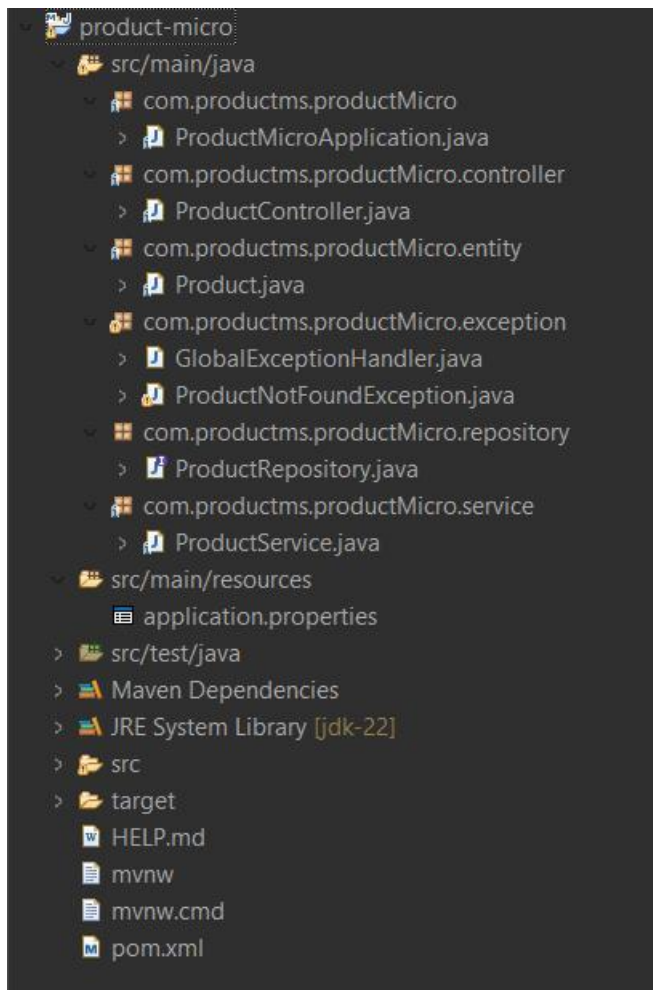


```
1 package com.products.productMicro.entity;
2
3 import lombok.Data;
4 import lombok.NoArgsConstructor;
5 import jakarta.persistence.*;
6
7 @Entity
8 @Table(name = "product")
9 @Data
10 @NoArgsConstructor
11 public class Product {
12
13     @Id
14     @GeneratedValue(strategy = GenerationType.IDENTITY)
15     private Long productId;
16
17     private String name;
18     private Integer stockQuantity;
19     private String category;
20     private Double price;
21
22     public Product(String name, Integer stockQuantity, Double price, String category) {
23         this.name = name;
24         this.stockQuantity = stockQuantity;
25         this.price = price;
26         this.category = category;
27     }
28
29 }
30
```

Customer folder structure:

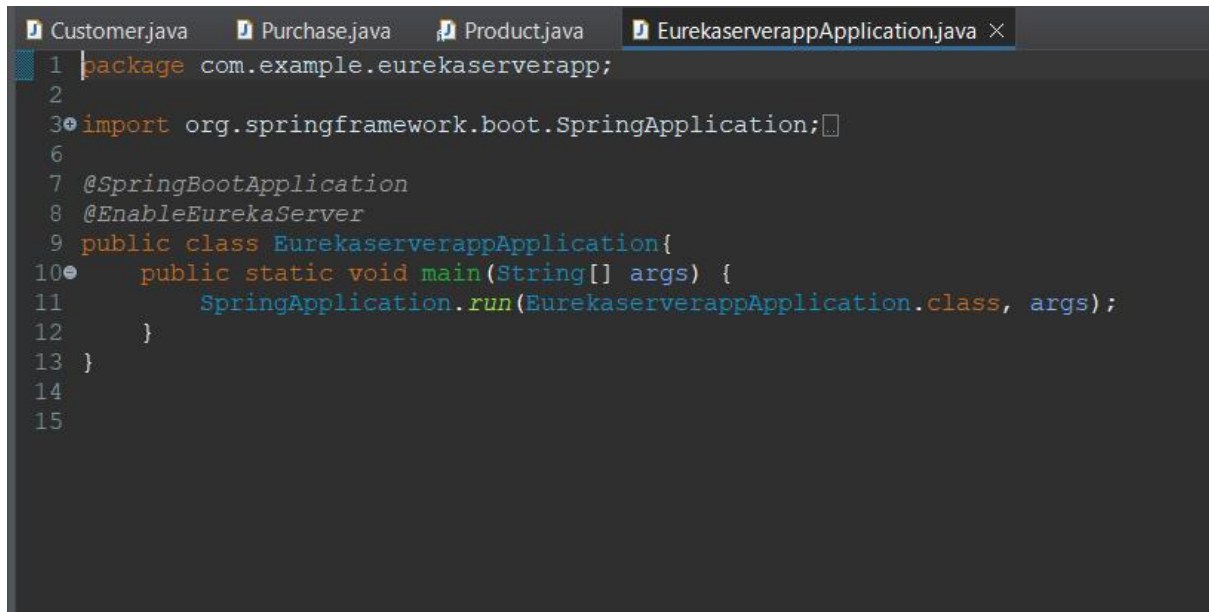


Product folder structure:



Setting up Eureka Server

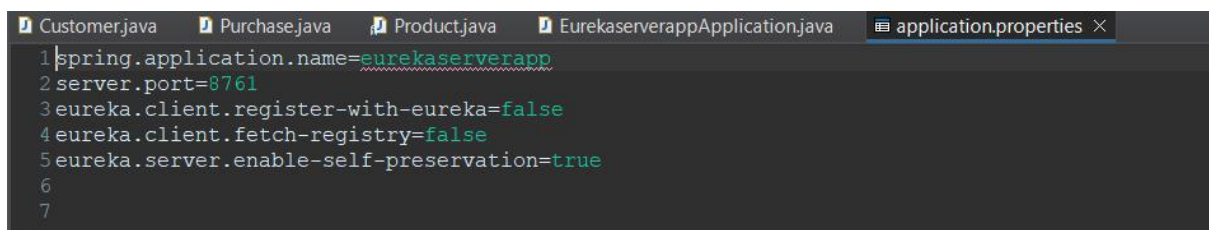
- **Main Application**



The screenshot shows an IDE with four tabs: Customer.java, Purchase.java, Product.java, and EurekaServerAppApplication.java. The EurekaServerAppApplication.java tab is active, displaying the following Java code:

```
1 package com.example.eurekaServerapp;  
2  
3 import org.springframework.boot.SpringApplication;  
4  
5  
6 @SpringBootApplication  
7 @EnableEurekaServer  
8  
9 public class EurekaServerAppApplication {  
10     public static void main(String[] args) {  
11         SpringApplication.run(EurekaServerAppApplication.class, args);  
12     }  
13 }  
14  
15
```

- **Application Properties**

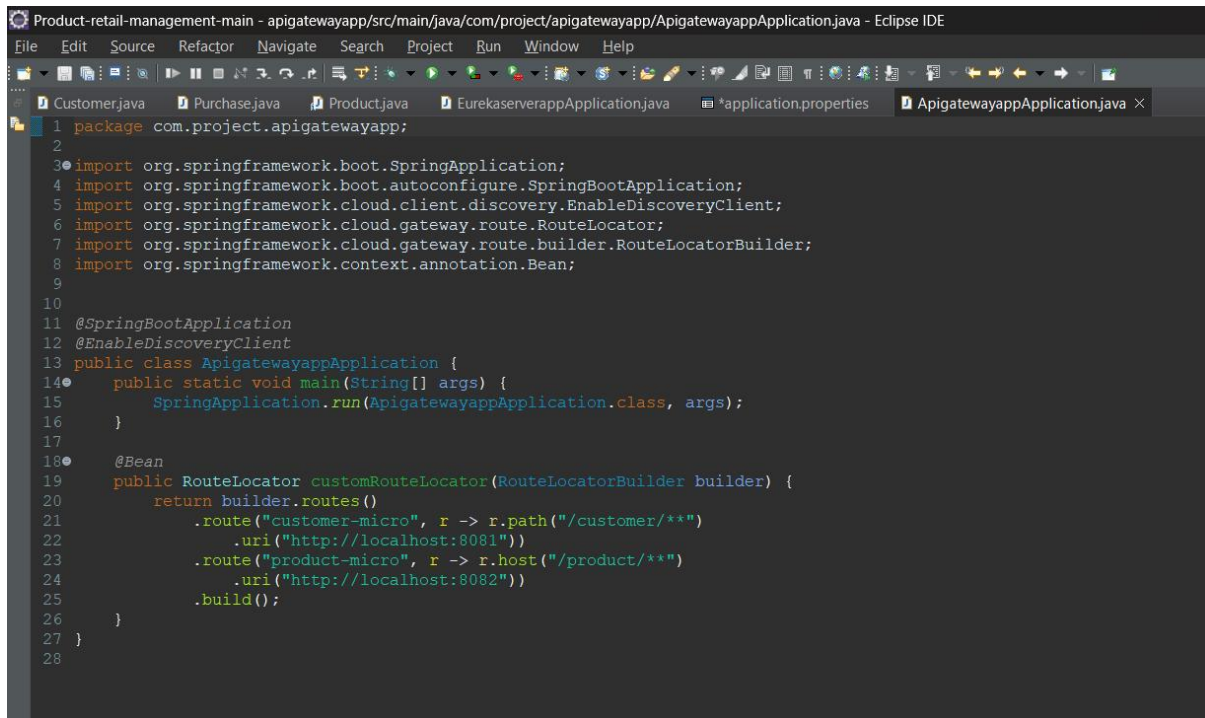


The screenshot shows an IDE with five tabs: Customer.java, Purchase.java, Product.java, EurekaServerAppApplication.java, and application.properties. The application.properties tab is active, displaying the following properties:

```
1 spring.application.name=eurekaServerapp  
2 server.port=8761  
3 eureka.client.register-with-eureka=false  
4 eureka.client.fetch-registry=false  
5 eureka.server.enable-self-preservation=true  
6  
7
```

Setting up API Gateway:

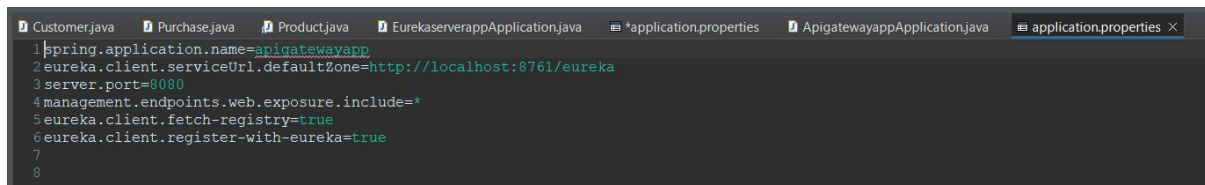
- Main Application



The screenshot shows the Eclipse IDE interface with the file 'ApigatewayappApplication.java' open. The code is as follows:

```
1 package com.project.apigatewayapp;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
6 import org.springframework.cloud.gateway.route.RouteLocator;
7 import org.springframework.cloud.gateway.route.builder.RouteLocatorBuilder;
8 import org.springframework.context.annotation.Bean;
9
10
11 @SpringBootApplication
12 @EnableDiscoveryClient
13 public class ApigatewayappApplication {
14     public static void main(String[] args) {
15         SpringApplication.run(ApigatewayappApplication.class, args);
16     }
17
18     @Bean
19     public RouteLocator customRouteLocator(RouteLocatorBuilder builder) {
20         return builder.routes()
21             .route("customer-micro", r -> r.path("/customer/**")
22                 .uri("http://localhost:8081"))
23             .route("product-micro", r -> r.host("/product/**")
24                 .uri("http://localhost:8082"))
25             .build();
26     }
27 }
28
```

- Application Properties



The screenshot shows the Eclipse IDE interface with the file 'application.properties' open. The properties are as follows:

```
1 spring.application.name=apigatewayapp
2 eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka
3 server.port=8080
4 management.endpoints.web.exposure.include=*
5 eureka.client.fetch-registry=true
6 eureka.client.register-with-eureka=true
7
8
```

Running the Server:

• EurekaServer

```
Markers Properties Servers Data Source Explorer Snippets Terminal Console X
EurekaServerApplication [Java Application] [pid: 10308]
Standard Commons Logging discovery in action with spring-jcl: please remove commons-logging.jar from classpath in order to avoid potential conflicts
Standard Commons Logging discovery in action with spring-jcl: please remove commons-logging.jar from classpath in order to avoid potential conflicts

:: Spring Boot ::                (v3.3.2)

2024-09-02T14:31:38.821+05:30 INFO 10308 --- [eureka-serverapp] [ restartedMain] c.e.e.EurekaServerApplication : Starting EurekaServerApplication
2024-09-02T14:31:38.824+05:30 INFO 10308 --- [eureka-serverapp] [ restartedMain] c.e.e.EurekaServerApplication : No active profile
2024-09-02T14:31:38.880+05:30 INFO 10308 --- [eureka-serverapp] [ restartedMain] e.DevToolsPropertyDefaultsPostProcessor : Devtools property
2024-09-02T14:31:38.881+05:30 INFO 10308 --- [eureka-serverapp] [ restartedMain] e.DevToolsPropertyDefaultsPostProcessor : For additional w
2024-09-02T14:31:39.875+05:30 INFO 10308 --- [eureka-serverapp] [ restartedMain] o.s.cloud.context.scope.GenericScope : BeanFactory id=7
2024-09-02T14:31:40.030+05:30 WARN 10308 --- [eureka-serverapp] [ restartedMain] trationDelegate$BeanPostProcessorChecker : Bean 'org.spring
2024-09-02T14:31:40.033+05:30 WARN 10308 --- [eureka-serverapp] [ restartedMain] trationDelegate$BeanPostProcessorChecker : Bean 'deferringL
2024-09-02T14:31:40.404+05:30 INFO 10308 --- [eureka-serverapp] [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initializ
2024-09-02T14:31:40.417+05:30 INFO 10308 --- [eureka-serverapp] [ restartedMain] o.apache.catalina.core.StandardService : Starting service
2024-09-02T14:31:40.418+05:30 INFO 10308 --- [eureka-serverapp] [ restartedMain] o.apache.catalina.core.StandardEngine : Starting Servlet
2024-09-02T14:31:40.465+05:30 INFO 10308 --- [eureka-serverapp] [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spr
2024-09-02T14:31:40.466+05:30 INFO 10308 --- [eureka-serverapp] [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicat
Standard Commons Logging discovery in action with spring-jcl: please remove commons-logging.jar from classpath in order to avoid potential conflicts
2024-09-02T14:31:41.320+05:30 INFO 10308 --- [eureka-serverapp] [ restartedMain] c.n.d.provider.DiscoveryJerseyProvider : Using JSON encod
2024-09-02T14:31:41.321+05:30 INFO 10308 --- [eureka-serverapp] [ restartedMain] c.n.d.provider.DiscoveryJerseyProvider : Using JSON decod
2024-09-02T14:31:41.513+05:30 INFO 10308 --- [eureka-serverapp] [ restartedMain] c.n.d.provider.DiscoveryJerseyProvider : Using XML encodi
2024-09-02T14:31:41.513+05:30 INFO 10308 --- [eureka-serverapp] [ restartedMain] c.n.d.provider.DiscoveryJerseyProvider : Using XML decodi
2024-09-02T14:31:42.249+05:30 INFO 10308 --- [eureka-serverapp] [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload serve
2024-09-02T14:31:42.305+05:30 WARN 10308 --- [eureka-serverapp] [ restartedMain] igation$LoadBalancerCaffeineWarnLogger : Spring Cloud Load
2024-09-02T14:31:42.325+05:30 INFO 10308 --- [eureka-serverapp] [ restartedMain] o.s.c.n.eureka.InstanceInfoFactory : Setting initial
2024-09-02T14:31:42.343+05:30 INFO 10308 --- [eureka-serverapp] [ restartedMain] com.netflix.discovery.DiscoveryClient : Initializing Eureka
2024-09-02T14:31:42.344+05:30 INFO 10308 --- [eureka-serverapp] [ restartedMain] com.netflix.discovery.DiscoveryClient : Client configure
2024-09-02T14:31:42.345+05:30 INFO 10308 --- [eureka-serverapp] [ restartedMain] com.netflix.discovery.DiscoveryClient : Discovery Client
2024-09-02T14:31:42.381+05:30 INFO 10308 --- [eureka-serverapp] [ restartedMain] c.n.eureka.DefaultEurekaServerContext : Initializing ...
2024-09-02T14:31:42.383+05:30 INFO 10308 --- [eureka-serverapp] [ restartedMain] c.n.eureka.cluster.PeerEurekaNodes : Adding new peer
```

• Customer Application

```
CustomerMicroApplication [Java Application] [pid: 5116]

:: Spring Boot ::                (v3.3.2)

2024-09-02T14:35:56.569+05:30 INFO 5116 --- [customer-micro] [ main] c.c.c.CustomerMicroApplication : Starting CustomerMicroApplication
2024-09-02T14:35:56.571+05:30 DEBUG 5116 --- [customer-micro] [ main] c.c.c.CustomerMicroApplication : Running with Spring
2024-09-02T14:35:56.572+05:30 INFO 5116 --- [customer-micro] [ main] c.c.c.CustomerMicroApplication : No active profile
2024-09-02T14:35:57.437+05:30 INFO 5116 --- [customer-micro] [ main] s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring
2024-09-02T14:35:57.566+05:30 INFO 5116 --- [customer-micro] [ main] s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data
2024-09-02T14:35:57.819+05:30 INFO 5116 --- [customer-micro] [ main] o.s.cloud.context.scope.GenericScope : BeanFactory id=321
2024-09-02T14:35:57.969+05:30 WARN 5116 --- [customer-micro] [ main] trationDelegate$BeanPostProcessorChecker : Bean 'org.springframework
2024-09-02T14:35:57.969+05:30 WARN 5116 --- [customer-micro] [ main] trationDelegate$BeanPostProcessorChecker : Bean 'deferringLoad
2024-09-02T14:35:58.189+05:30 INFO 5116 --- [customer-micro] [ main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized
2024-09-02T14:35:58.202+05:30 INFO 5116 --- [customer-micro] [ main] o.apache.catalina.core.StandardService : Starting service [
2024-09-02T14:35:58.202+05:30 INFO 5116 --- [customer-micro] [ main] o.apache.catalina.core.StandardEngine : Starting Servlet e
2024-09-02T14:35:58.271+05:30 INFO 5116 --- [customer-micro] [ main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring
2024-09-02T14:35:58.272+05:30 INFO 5116 --- [customer-micro] [ main] w.s.c.ServletWebServerApplicationContext : Root WebApplication
2024-09-02T14:35:58.442+05:30 INFO 5116 --- [customer-micro] [ main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Process
2024-09-02T14:35:58.493+05:30 INFO 5116 --- [customer-micro] [ main] org.hibernate.Version : HHH000412: Hibernate
2024-09-02T14:35:58.521+05:30 INFO 5116 --- [customer-micro] [ main] o.h.c.internal.RegionFactoryInitiator : HHH000026: Second-
2024-09-02T14:35:58.794+05:30 INFO 5116 --- [customer-micro] [ main] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver
2024-09-02T14:35:58.817+05:30 INFO 5116 --- [customer-micro] [ main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Sta
2024-09-02T14:35:59.113+05:30 INFO 5116 --- [customer-micro] [ main] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Add
2024-09-02T14:35:59.115+05:30 INFO 5116 --- [customer-micro] [ main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Sta
2024-09-02T14:35:59.855+05:30 INFO 5116 --- [customer-micro] [ main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000489: No JTA
2024-09-02T14:35:59.895+05:30 INFO 5116 --- [customer-micro] [ main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA En
2024-09-02T14:36:00.365+05:30 WARN 5116 --- [customer-micro] [ main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in
2024-09-02T14:36:00.443+05:30 INFO 5116 --- [customer-micro] [ main] o.s.c.openfeign.FeignClientFactoryBean : For 'product-micro
2024-09-02T14:36:00.958+05:30 INFO 5116 --- [customer-micro] [ main] DiscoveryClientOptionalArgsConfiguration : Eureka HTTP Client
2024-09-02T14:36:00.984+05:30 WARN 5116 --- [customer-micro] [ main] igation$LoadBalancerCaffeineWarnLogger : Spring Cloud LoadB
2024-09-02T14:36:01.010+05:30 INFO 5116 --- [customer-micro] [ main] o.s.c.n.eureka.InstanceInfoFactory : Setting initial in
```



```

ProductMicroApplication [Java Application] [pid: 4092]

:: Spring Boot ::

(v3.3.2)

2024-09-02T14:36:35.019+05:30 INFO 4092 --- [product-micro] [
2024-09-02T14:36:35.021+05:30 DEBUG 4092 --- [product-micro] [
2024-09-02T14:36:35.023+05:30 INFO 4092 --- [product-micro] [
2024-09-02T14:36:35.853+05:30 INFO 4092 --- [product-micro] [
2024-09-02T14:36:35.975+05:30 INFO 4092 --- [product-micro] [
2024-09-02T14:36:36.204+05:30 INFO 4092 --- [product-micro] [
2024-09-02T14:36:36.344+05:30 WARN 4092 --- [product-micro] [
2024-09-02T14:36:36.347+05:30 WARN 4092 --- [product-micro] [
2024-09-02T14:36:36.594+05:30 INFO 4092 --- [product-micro] [
2024-09-02T14:36:36.594+05:30 INFO 4092 --- [product-micro] [
2024-09-02T14:36:36.594+05:30 INFO 4092 --- [product-micro] [
2024-09-02T14:36:36.600+05:30 INFO 4092 --- [product-micro] [
2024-09-02T14:36:36.661+05:30 INFO 4092 --- [product-micro] [
2024-09-02T14:36:36.819+05:30 INFO 4092 --- [product-micro] [
2024-09-02T14:36:36.868+05:30 INFO 4092 --- [product-micro] [
2024-09-02T14:36:36.899+05:30 INFO 4092 --- [product-micro] [
2024-09-02T14:36:37.162+05:30 INFO 4092 --- [product-micro] [
2024-09-02T14:36:37.188+05:30 INFO 4092 --- [product-micro] [
2024-09-02T14:36:37.471+05:30 INFO 4092 --- [product-micro] [
2024-09-02T14:36:37.473+05:30 INFO 4092 --- [product-micro] [
2024-09-02T14:36:38.217+05:30 INFO 4092 --- [product-micro] [
Hibernate: create table product (product_id bigint not null auto_increment, category varchar(255), name varchar(255), price float(53), stock
2024-09-02T14:36:38.274+05:30 INFO 4092 --- [product-micro] [
2024-09-02T14:36:38.683+05:30 WARN 4092 --- [product-micro] [
2024-09-02T14:36:39.251+05:30 INFO 4092 --- [product-micro] [
2024-09-02T14:36:39.294+05:30 WARN 4092 --- [product-micro] [
2024-09-02T14:36:39.320+05:30 INFO 4092 --- [product-micro] [
main] c.p.p.ProductMicroApplication : Starting ProductMic
main] c.p.p.ProductMicroApplication : Running with Spring
main] c.p.p.ProductMicroApplication : No active profile s
main] s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Sprin
main] s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Dat
main] o.s.cloud.context.scope.GenericScope : BeanFactory id=fle0
main] trationDelegate$BeanPostProcessorChecker : Bean 'org.springframework
main] trationDelegate$BeanPostProcessorChecker : Bean 'defferingload
main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized
main] o.apache.catalina.core.StandardService : Starting service [T
main] o.apache.catalina.core.StandardEngine : Starting Servlet en
main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring
main] w.s.c.ServletWebServerApplicationContext : Root WebApplication
main] o.hibernate.jpa.internal.util.LogHelper : HHH0000204: Processi
main] org.hibernate.Version : HHH0000412: Hibernat
main] o.h.c.internal.RegionFactoryInitiator : HHH0000026: Second-l
main] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver s
main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Star
main] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Adde
main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Star
main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH0000489: No JTA p
main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA Ent
main] jpa.BaseConfigurationsJpaWebConfiguration : spring.jpa.open-in
main] DiscoveryClientOptionalArgsConfiguration : Eureka HTTP Client
main] igation$LoadBalancerCacheWarnLogger : Spring cloud LoadBa
main] o.s.c.n.eureka.InstanceInfoFactory : Setting initial ins

```

[illegible]

- Application being hosted on Eureka Server

The screenshot shows the Spring Eureka server interface in a web browser. The browser tabs include 'Spring Initializr', 'Eureka', and two instances of 'Swagger UI'. The address bar shows 'localhost:8761'. The Eureka dashboard has a dark header with the 'spring Eureka' logo and navigation links for 'HOME' and 'LAST 1000 SINCE STARTUP'.

System Status

Environment	test	Current time	2024-09-02T14:37:04 +0530
Data center	default	Uptime	00:05
		Lease expiration enabled	false
		Renews threshold	6
		Renews (last min)	2

EMERGENCY! EUREKA MAY BE INCORRECTLY CLAIMING INSTANCES ARE UP WHEN THEY'RE NOT. RENEWALS ARE LESSER THAN THRESHOLD AND HENCE THE INSTANCES ARE NOT BEING EXPIRED JUST TO BE SAFE.

DS Replicas

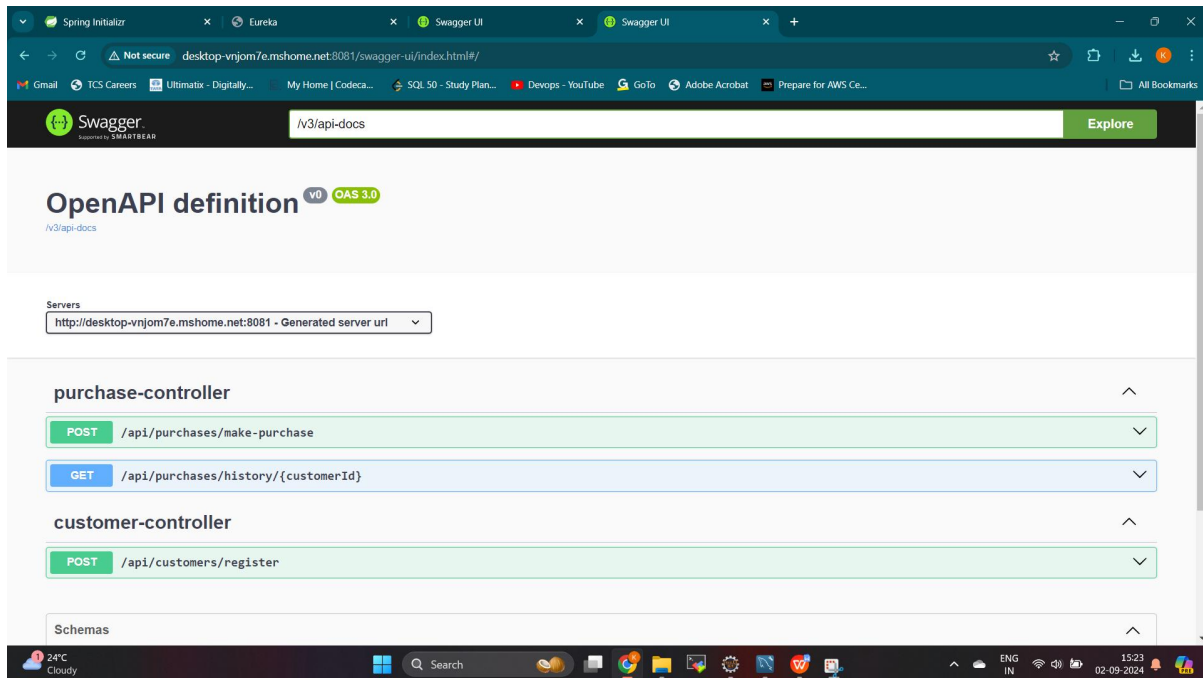
localhost

Instances currently registered with Eureka

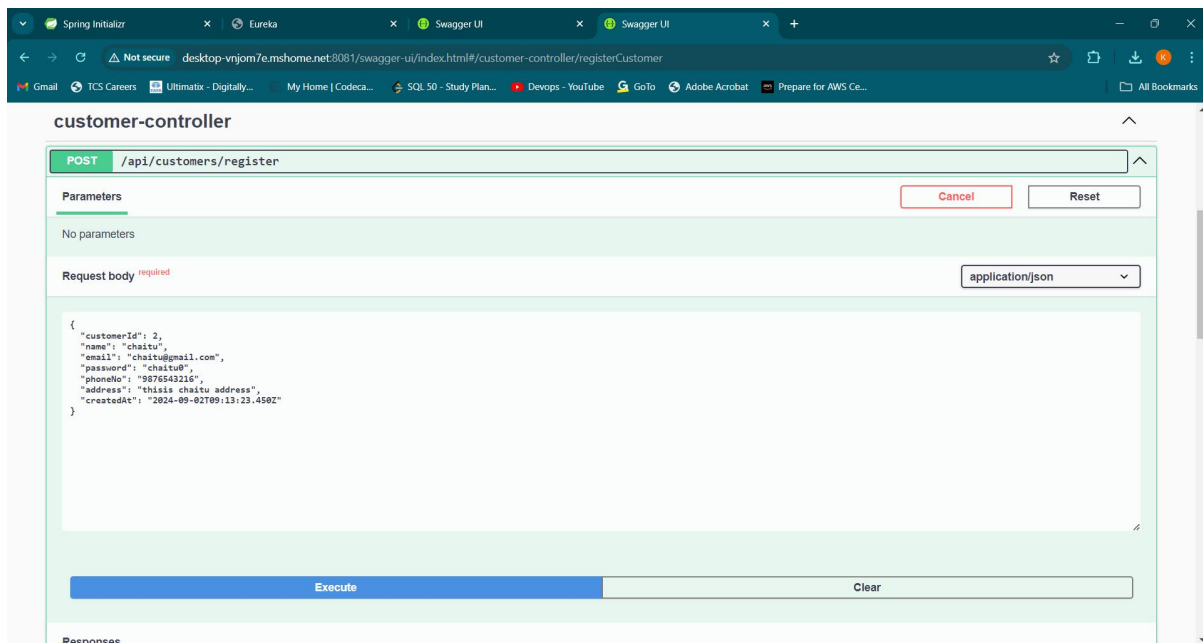
Application	AMIs	Availability Zones	Status
APIGATEWAYAPP	n/a (1)	(1)	UP (1) - DESKTOP-VNJQM7E.mshome.net:apigatewayapp:8080
CUSTOMER-MICRO	n/a (1)	(1)	UP (1) - DESKTOP-VNJQM7E.mshome.net:customer-micro:8081
PRODUCT-MICRO	n/a (1)	(1)	UP (1) - DESKTOP-VNJQM7E.mshome.net:product-micro:8082

The Windows taskbar at the bottom shows the system clock as 15:20 on 02-09-2024, with language set to ENG IN.

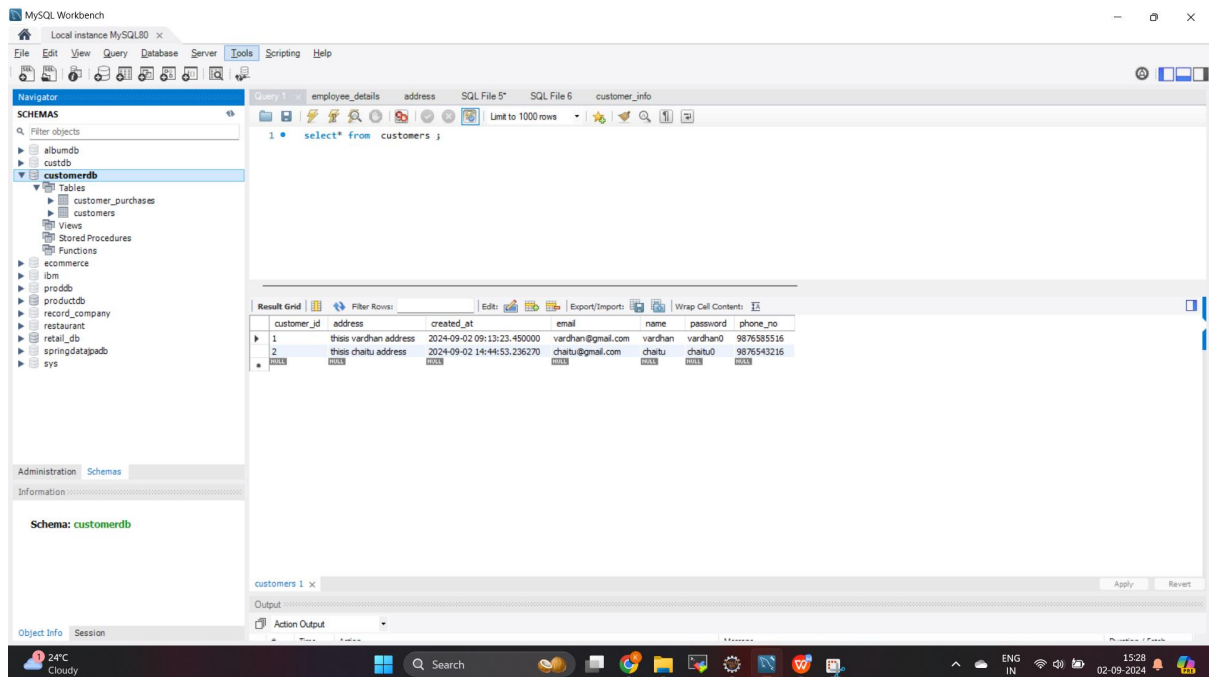
• Customer microservice Swagger



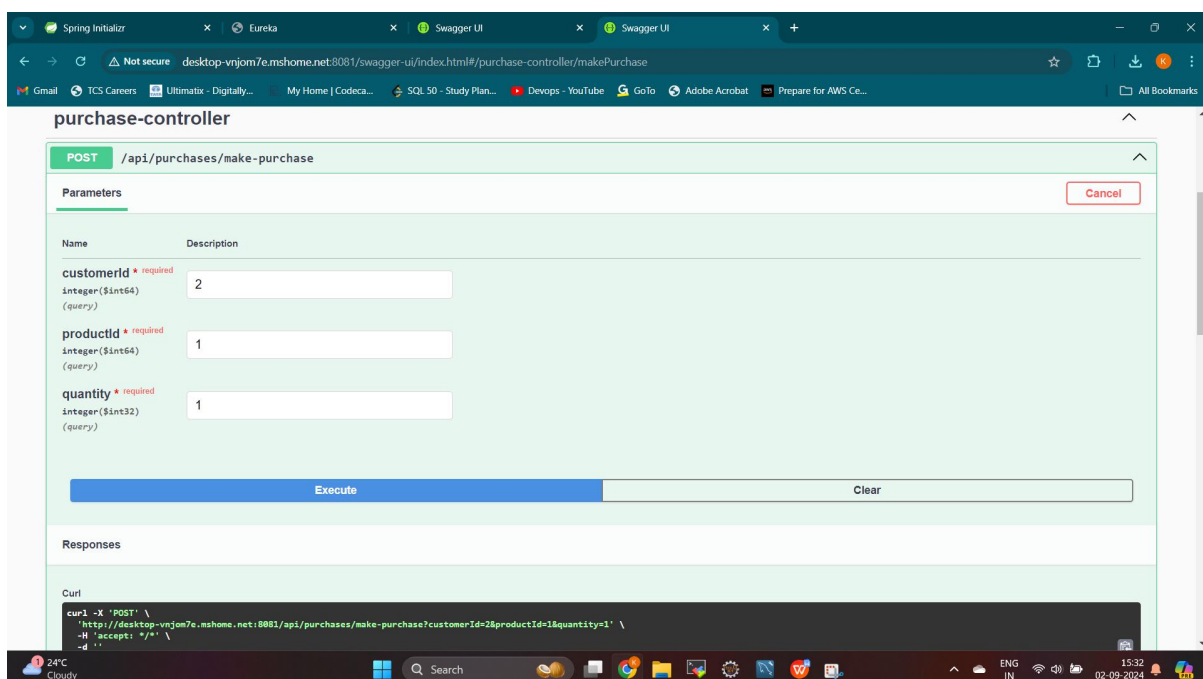
• Customer Register



Customer Table



- Customer can purchase based on customer id , Product id and quantity that he wants to purchase



- Purchase Table

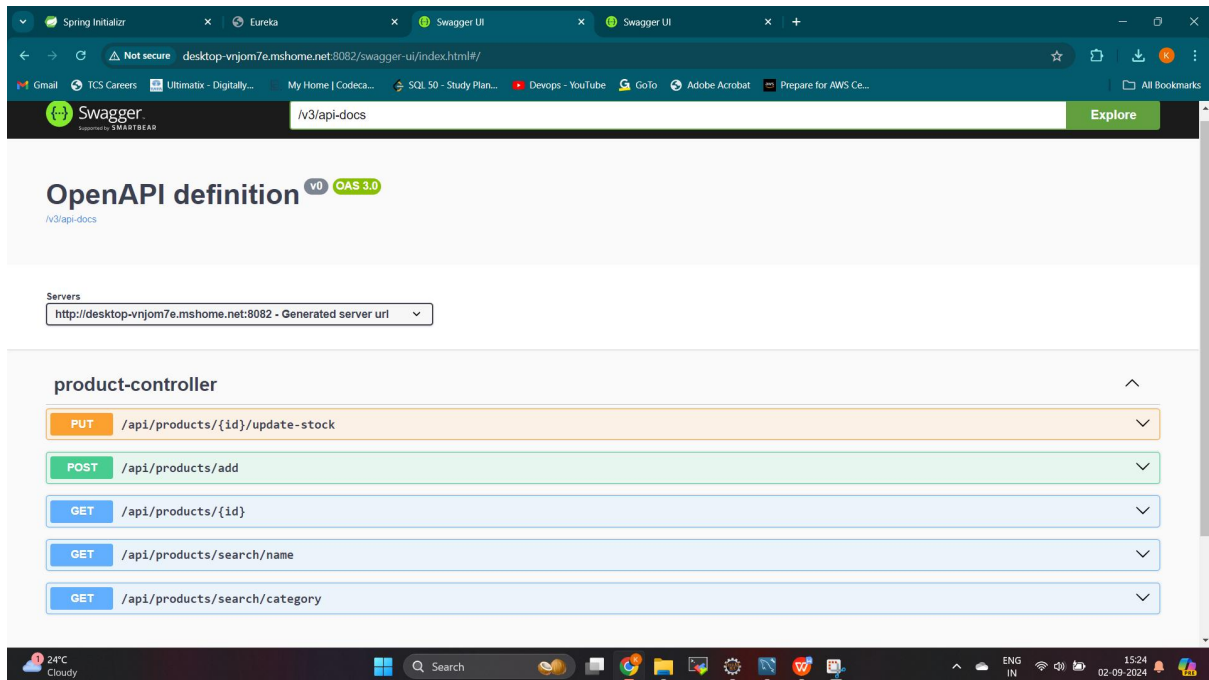
The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays a tree view of databases, with 'customerdb' expanded to show the 'customer_purchases' table. Below this, the 'Table: customer_purchases' section lists its columns: purchase_id (bigint, AI, PK), customer_id (bigint), product_id (bigint), purchase_date (datetime(6)), quantity (int), and total_price (decimal(38,2)). The main pane shows a SQL query: `SELECT * FROM customerdb.customer_purchases;` and a 'Result Grid' containing three rows of purchase data.

purchase_id	customer_id	product_id	purchase_date	quantity	total_price
1	1	2	2024-09-02 14:45:16.252425	2	200.00
2	2	1	2024-09-02 14:45:34.134372	1	120.00
3	2	1	2024-09-02 14:45:37.488615	1	120.00

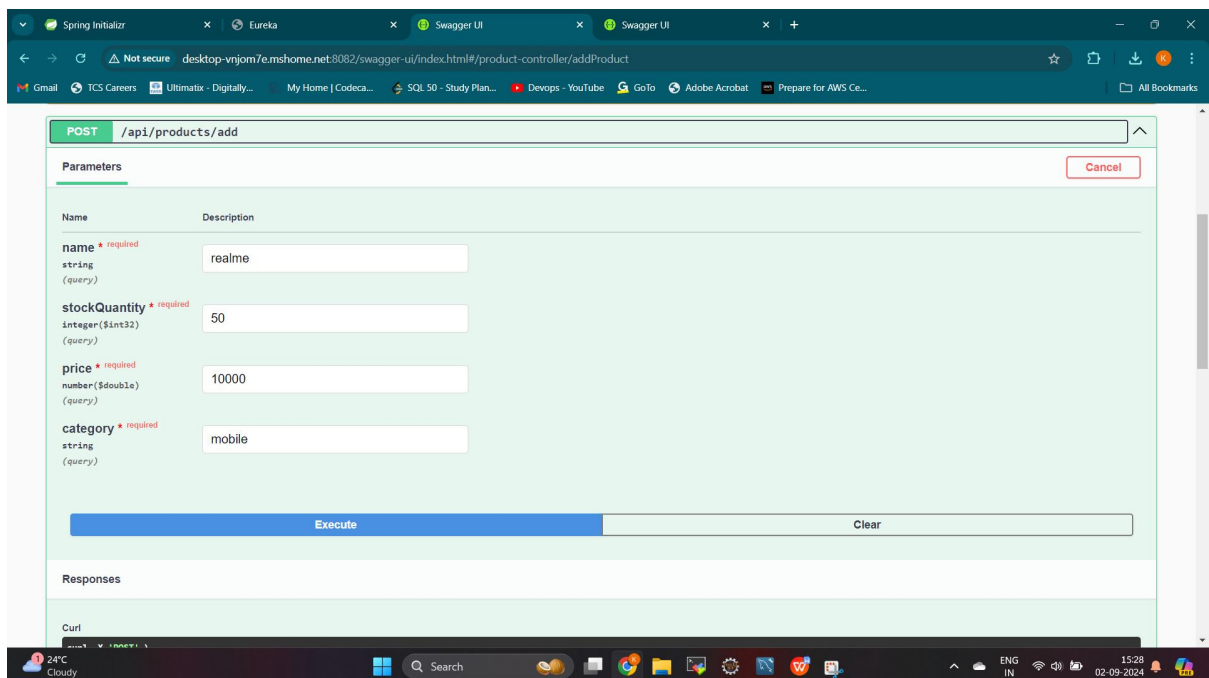
- Get List of purchases made by Customer by Customer Id

The screenshot shows the Swagger UI for a REST API. The endpoint `GET /api/purchases/history/{customerId}` is selected. The 'Parameters' section shows a required path parameter `customerId` of type `integer($int64)` with the value `2` entered. Below the parameters, there are 'Execute' and 'Clear' buttons. The 'Responses' section shows a '200' status code with a 'Response body' field. The 'Curl' section displays the generated curl command: `curl -X 'GET' -H 'accept: */*' 'http://desktop-vnjom7e.mshome.net:8081/api/purchases/history/2'`. The 'Request URL' section shows the full URL: `http://desktop-vnjom7e.mshome.net:8081/api/purchases/history/2`.

• Product MicroService Swagger



• Add Product in Product Database



• Product Table

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays a tree view of databases, with 'productdb' selected. Below it, the 'Table: product' is detailed with columns: product_id (bigint, PK), category (varchar(255)), name (varchar(255)), price (double), and stock_quantity (int). The main query window shows a SQL query: `SELECT * FROM productdb.product;`. The 'Result Grid' pane displays the following data:

product_id	category	name	price	stock_quantity
1	vehicle	car-mirror	120	18
2	vehicle	bike-mirror	100	28
3	vehicle	helmet	200	30
4	mobile	realme	10000	50

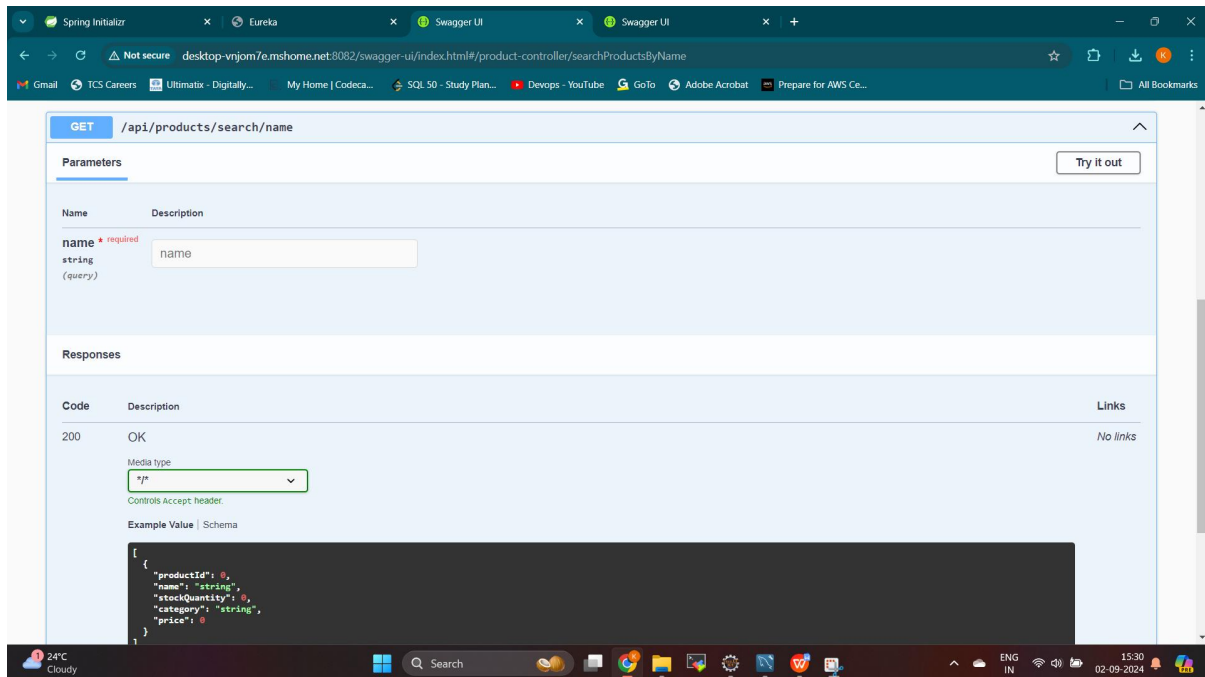
• Get Product by Id

The screenshot shows the Swagger UI interface for a REST API. The endpoint `GET /api/products/{id}` is selected. The 'Parameters' section shows a required integer parameter `id` with the value `9`. The 'Execute' button is highlighted. Below, the 'Responses' section shows a 404 status code with the message 'Error: response status is 404'. The 'Curl' section displays the following command:

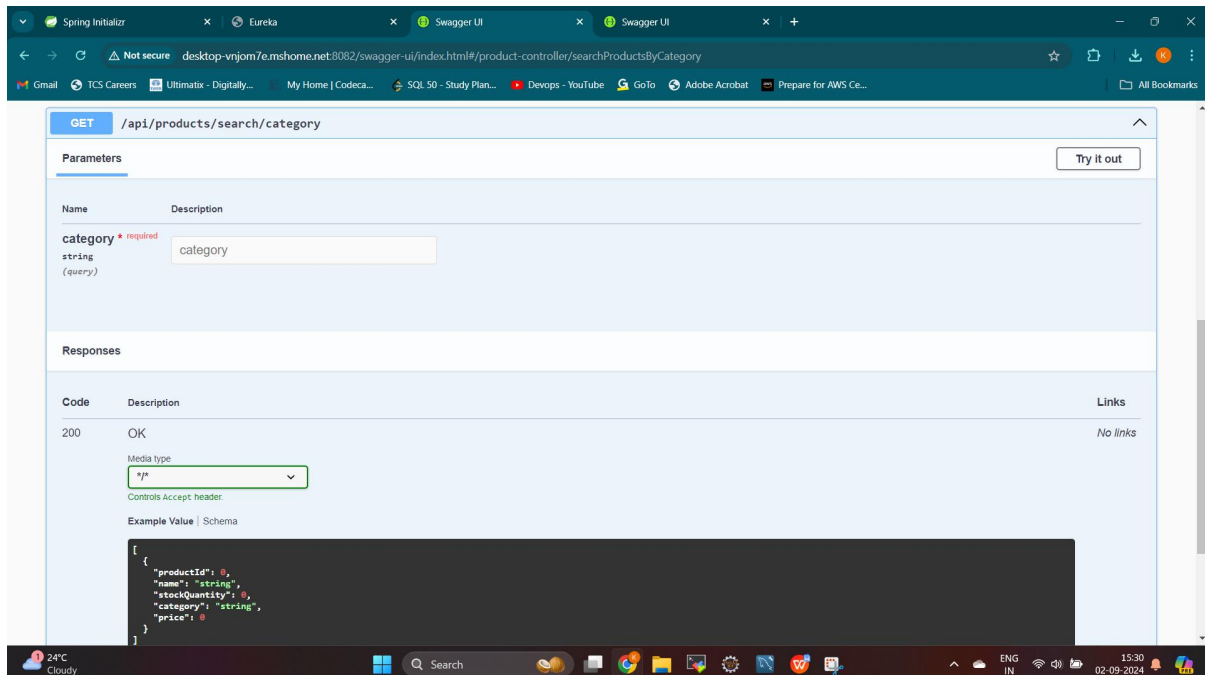
```
curl -X 'GET' \
'http://desktop-vnjon7e.mshome.net:8082/api/products/9' \
-H 'accept: */*'
```

The 'Request URL' is `http://desktop-vnjon7e.mshome.net:8082/api/products/9`.

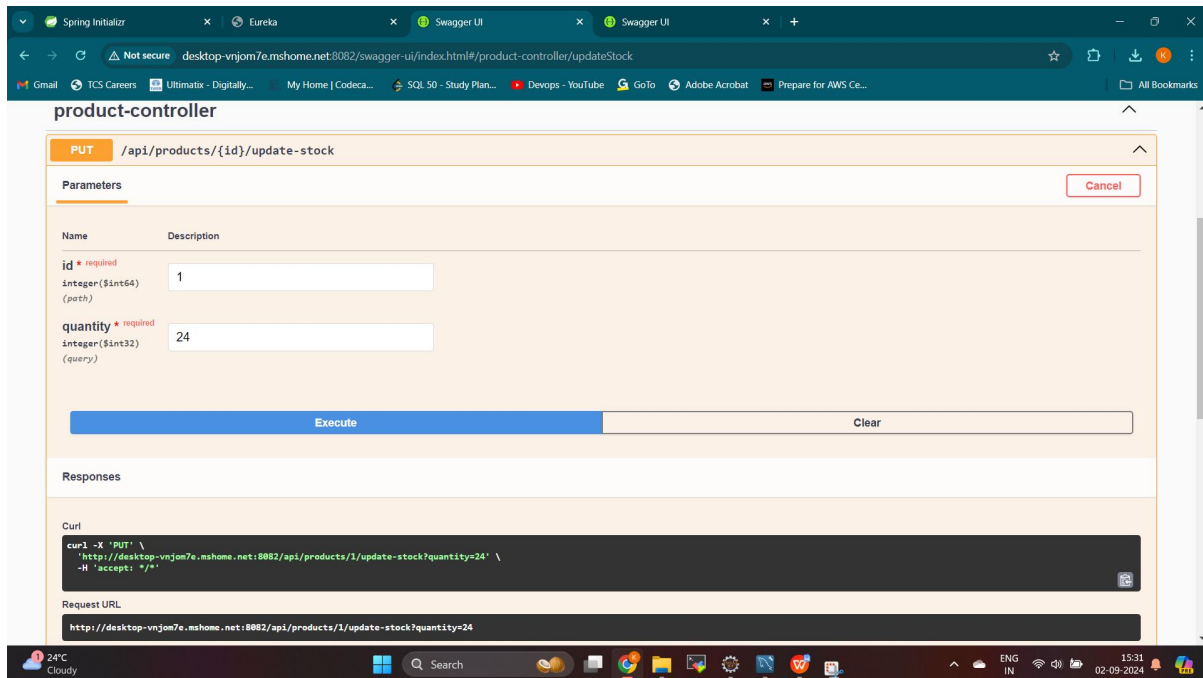
- **Get Product via name**



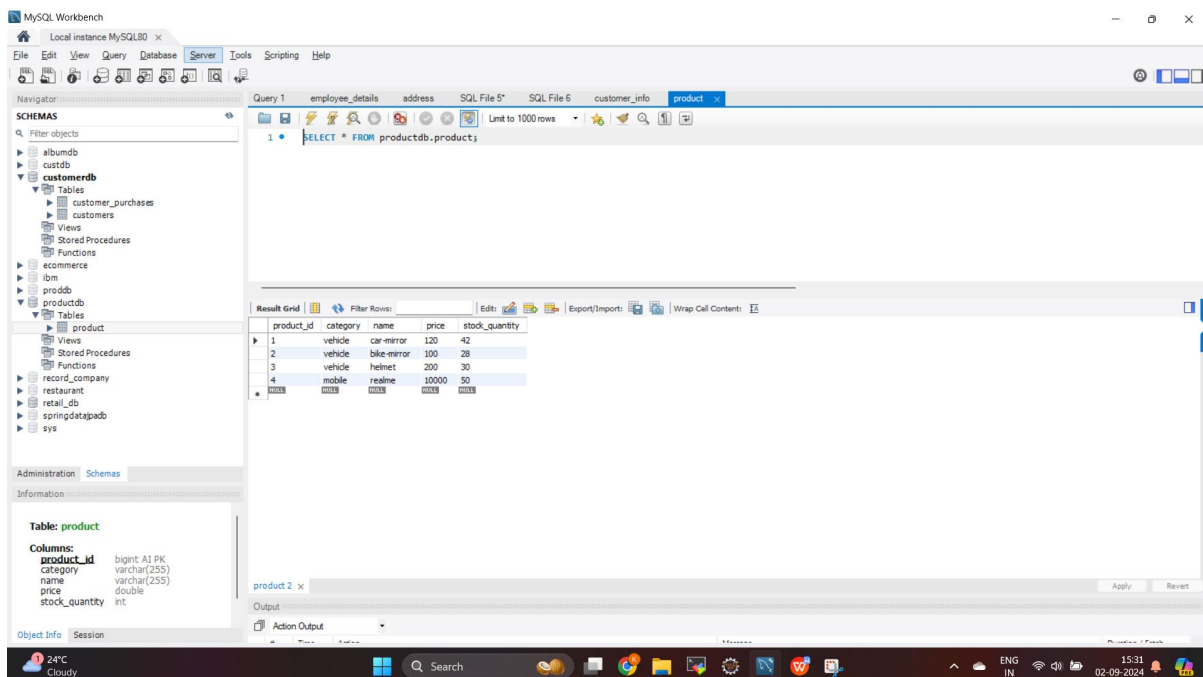
- **Get list Of Product via category**



- Add more quantity in stocks



- Reflects in Database



- Customer can register and purchase available products.
- Customer can also view purchase history.
- Product can be registered and updated in terms of stock quantity.
- Products are fetched by id, name, category.

We implemented all the use cases.....

THANK YOU

====*