

Spring Boot Assessment

Ettaboina Ganesh
61095792

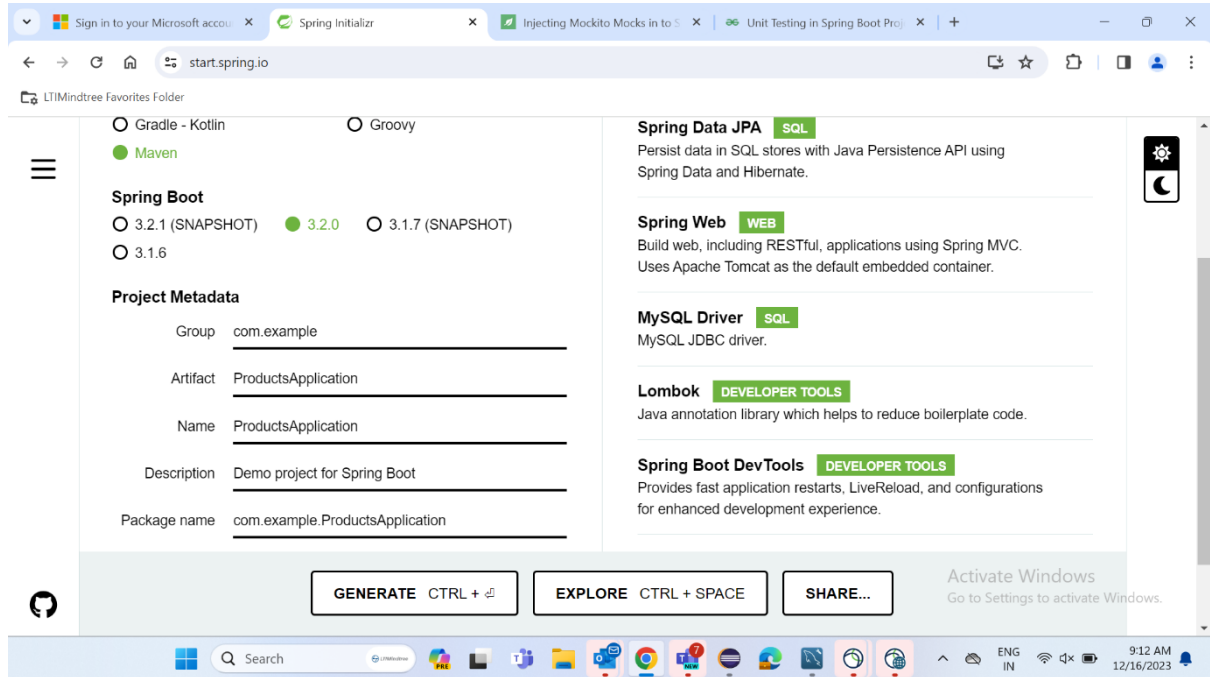
Document Version / Details: Ver. 2.1/ 02-Jan-2023

Scenario: Create a REST API for resource Product. The Product Entity has following fields:

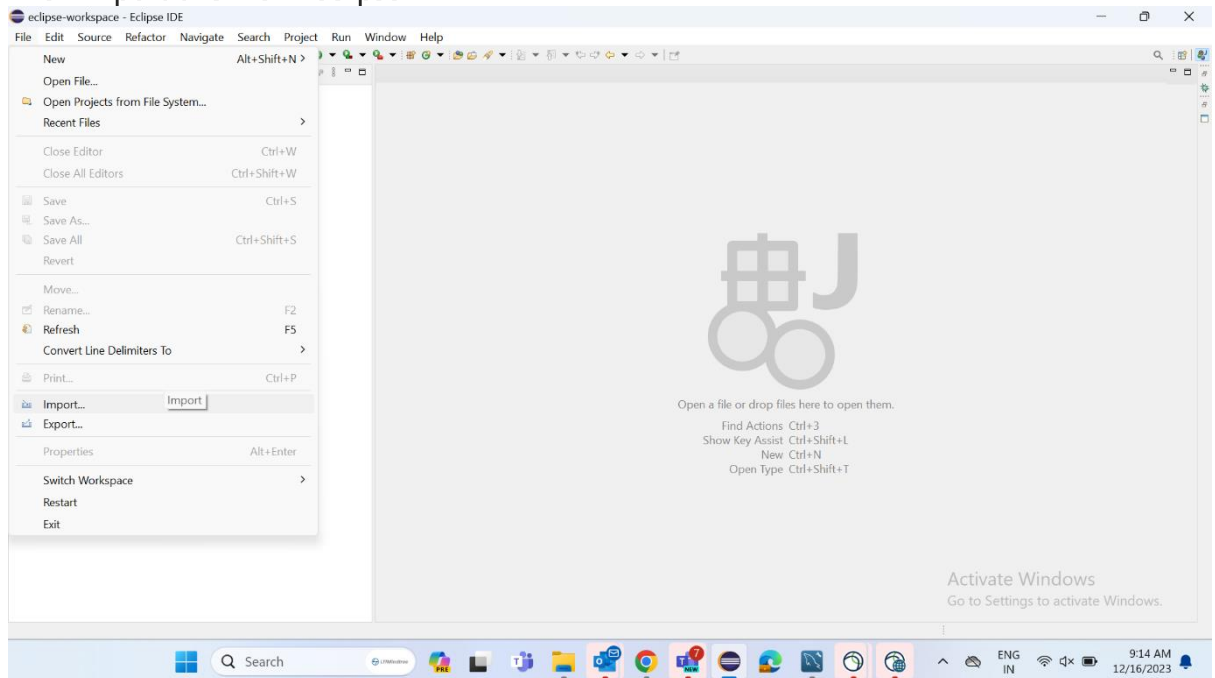
1. Unique ID
2. Name
3. Description
4. Price
5. ManufacturingDate.

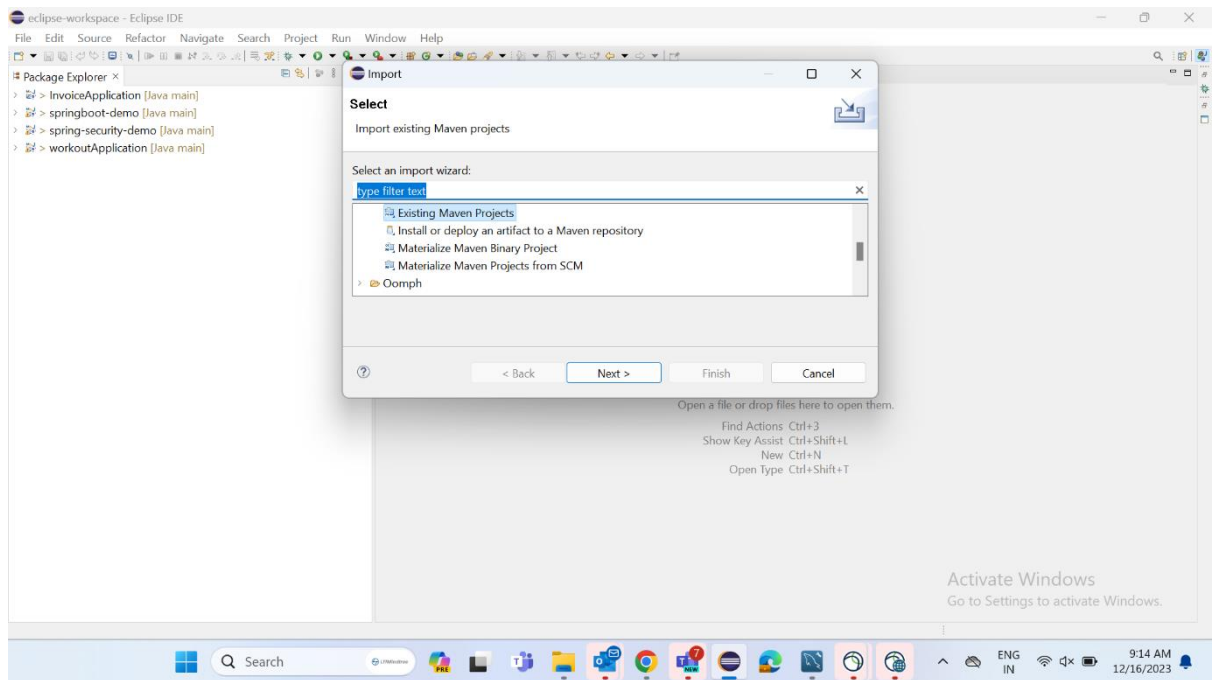
Steps:

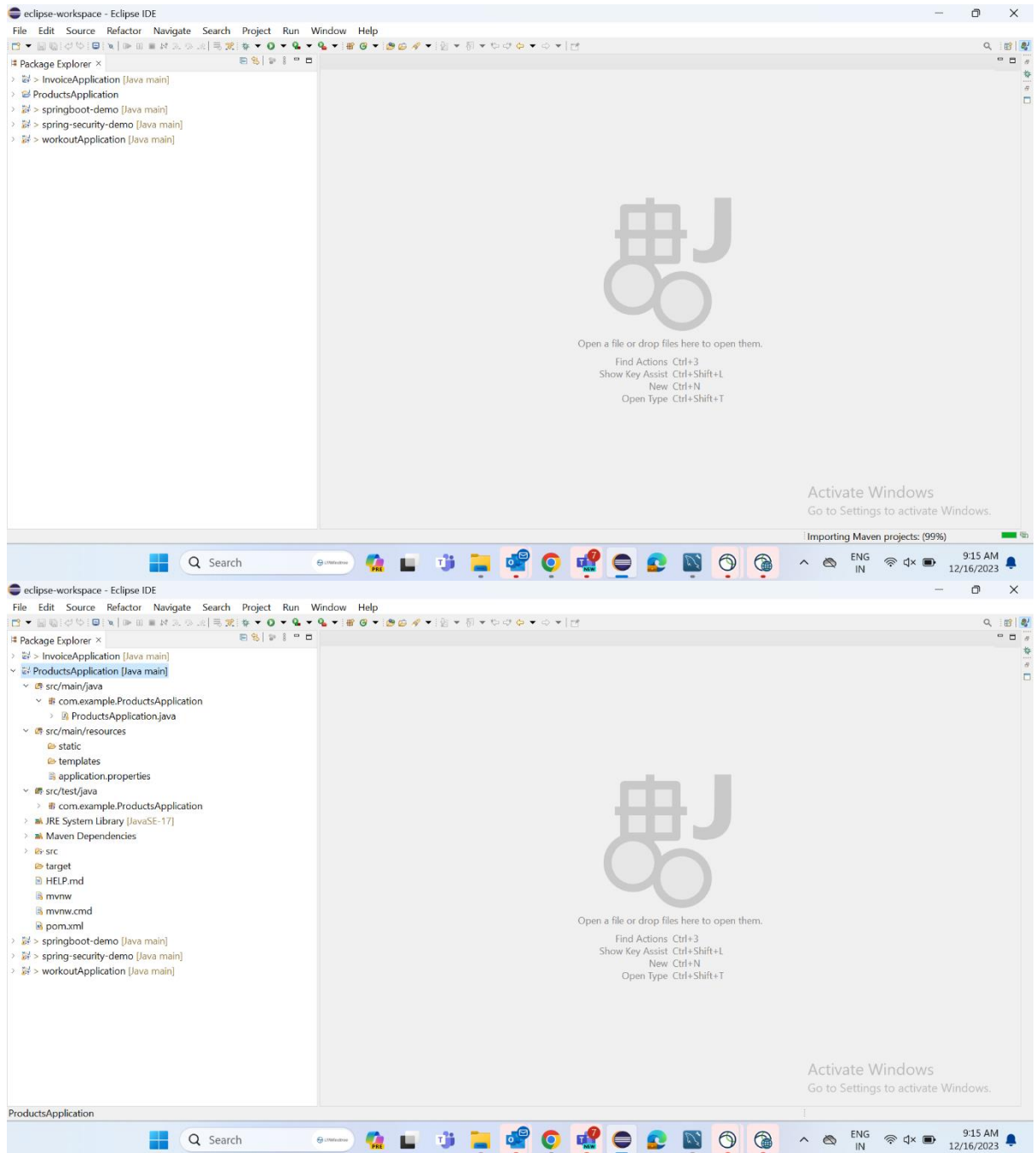
1.Download Spring initializer file



2. Then import this file in eclipse







Pom.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
```

```
<version>3.2.0</version>
<relativePath/> <!-- lookup parent from repository -->
</parent>
<groupId>com.example</groupId>
<artifactId>ProductsApplication</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>ProductsApplication</name>
<description>Demo project for Spring Boot</description>
<properties>
  <java.version>17</java.version>
</properties>
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
  </dependency>
  <dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <scope>runtime</scope>
  </dependency>
  <dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-starter-webmvc-
ui</artifactId>
    <version>2.1.0</version>
  </dependency>
  <dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>
<groupId>org.mockito</groupId>
```

```
<artifactId>mockito-junit-jupiter</artifactId>
<scope>test</scope>
</dependency>

</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
      <configuration>
        <excludes>
          <exclude>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
          </exclude>
        </excludes>
      </configuration>
    </plugin>
  </plugins>
</build>

</project>
```

Application.properties:

```
server.port = 3000

spring.datasource.url=jdbc:mysql://localhost:3306/productsdb
spring.datasource.username=root
spring.datasource.password=Ganesh@123

spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.
MySQL8Dialect
```

Products.java code:

```
package com.example.ProductsApplication.Entities;

import java.sql.Date;
import java.time.LocalDate;
```

```
import com.fasterxml.jackson.annotation.JsonFormat;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class Products {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    public int id;
    String name;
    String description;
    public double price;
    @JsonFormat(pattern="yyyy-MM-dd")
    Date manufacturingDate;
    // LocalDate manufacturingDate;

    public Products() {
    //     super();
    }

    public Products(int id, String name, String description,
double price, Date manufacturingDate) {
        super();
        id = id;
        this.name = name;
        this.description = description;
        this.price = price;
        this.manufacturingDate = manufacturingDate;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        id = id;
    }

    public String getName() {
```

```
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public Date getManufacturingDate() {
        return manufacturingDate;
    }

    public void setManufacturingDate(Date manufacturingDate)
    {
        this.manufacturingDate = manufacturingDate;
    }
}
```

ProductRepository.java code:

```
package com.example.ProductsApplication.Repositories;

import java.sql.Date;
import java.util.List;
import org.springframework.data.jpa.repository.JpaRepository;
```



```
import com.example.ProductsApplication.Entities.Products;

public interface ProductRepository extends
JpaRepository<Products, Integer> {

    List<Products> findByName(String name);

    List<Products> findByDescriptionContaining(String
description);

    List<Products> findByPriceGreaterThan(double price);

    List<Products> findByManufacturingDateEquals(Date mfd);

    List<Products> findByName();

}
```

ProductController.java code:

```
package com.example.ProductsApplication.Controllers;

import java.sql.Date;
//import java.time.LocalDate;
import java.util.List;
import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestController;

import com.example.ProductsApplication.Entities.Products;
import
com.example.ProductsApplication.Repositories.ProductRepository
;
```

```
@RestController
@RequestMapping("/Products")
public class ProductController {

    @Autowired
    ProductRepository productRepository;
    @PostMapping("/AddProducts")
    public ResponseEntity<String> addProducts(@RequestBody
Products products) {
        ResponseEntity response=null;
        try {
            productRepository.save(products);
            response= new ResponseEntity<String>(
                "Product added",HttpStatus.CREATED);
        }
        catch(Exception e){
            response=new ResponseEntity<String>("Adding
products failed",HttpStatus.INTERNAL_SERVER_ERROR);
        }
        return response;
    }
    @GetMapping("/getProducts")
    @ResponseStatus(HttpStatus.OK)
    public List<Products> getProducts(){

        return productRepository.findAll();
    }

    //put

    @PutMapping("/productsUpdate/{id}")
    public ResponseEntity<String>
updateProduct(@PathVariable("id") int id, @RequestBody
Products products) {

        ResponseEntity<String> response = null;

        Optional<Products> productid =
productRepository.findById(id);

        if (productid.isPresent()) {
            products.id=id;
            productRepository.save(products);

            response= new ResponseEntity<String>(
                "Product Updated",HttpStatus.CREATED);
            return response;
        }
    }
}
```

```
        } else
            response = new ResponseEntity<>("Product not
Found",HttpStatus.NOT_FOUND);
        return response;
    }

    //delete
    @DeleteMapping("/productDelete/{id}")
    public ResponseEntity<String>
deleteProduct(@PathVariable("id") int id, @RequestBody
Products products) {

        ResponseEntity<String> response = null;

        Optional<Products> productid =
productRepository.findById(id);

        if (productid.isPresent()) {
            productRepository.deleteById(id);

            response= new ResponseEntity<String>("Product is deleted",HttpStatus.OK);
            return response;
        } else
            response = new ResponseEntity<>("Product not
Found",HttpStatus.NOT_FOUND);
        return response;
    }

    //findMethods.....
    @GetMapping("/findbyname")
    @ResponseStatus(HttpStatus.OK)
    public ResponseEntity<List<Products>>
getProductByName(@RequestParam("name") String name) {
        List<Products>
product=productRepository.findByName(name);
        ResponseEntity<List<Products>> response=null;
        if(product.isEmpty()) {
            response =new
ResponseEntity<List<Products>>(HttpStatus.NOT_FOUND);
        return response;
    }

    else {
        response =new
ResponseEntity<List<Products>>(product,HttpStatus.OK);
        return response;
    }
}
```

```
    }

    @GetMapping("/Descriptioncontains")
    @ResponseStatus(HttpStatus.OK)
    public ResponseEntity<List<Products>>
getDescriptionContaining(@RequestParam("description") String
description) {
    List<Products>
product=productRepository.findByDescriptionContaining(descript
ion);
    ResponseEntity<List<Products>> response=null;
    if(product.isEmpty()) {
        response =new
ResponseEntity<List<Products>>(HttpStatus.NOT_FOUND);
        return response;
    }

    else {
        response =new
ResponseEntity<List<Products>>(product,HttpStatus.OK);
        return response;
    }
}

    @GetMapping("/pricegreaterthan")
    @ResponseStatus(HttpStatus.OK)
    public ResponseEntity<List<Products>>
getPrice(@RequestParam("price") double price) {
    List<Products>
product=productRepository.findByPriceGreaterThan(price);
    ResponseEntity<List<Products>> response=null;
    if(product.isEmpty()) {
        response =new
ResponseEntity<List<Products>>(HttpStatus.NOT_FOUND);
        return response;
    }

    else {
        response =new
ResponseEntity<List<Products>>(product,HttpStatus.OK);
        return response;
    }
}

    @GetMapping("/samemanufacturedate")
    @ResponseStatus(HttpStatus.OK)
    public ResponseEntity<List<Products>>
getSameDate(@RequestParam("mfd") Date mfd) {
    List<Products>
product=productRepository.findByManufacturingDateEquals(mfd);
```

```
        ResponseEntity<List<Products>> response=null;
        if(product.isEmpty()) {
            response =new
ResponseEntity<List<Products>> (HttpStatus.NOT_FOUND) ;
            return response;
        }

        else {
            response =new
ResponseEntity<List<Products>> (product,HttpStatus.OK) ;
            return response;
        }
    }

    @GetMapping("/orderbyname")
    // @ResponseStatus(HttpStatus.OK)
    public ResponseEntity<List<Products>>
getEmployeeByNameByOrder() {
        List<Products> product=productRepository.
findOrderByByName();
        ResponseEntity<List<Products>> response=null;
        if(product.isEmpty()) {
            response =new
ResponseEntity<List<Products>> (HttpStatus.NOT_FOUND) ;
            return response;
        }

        else {
            response =new
ResponseEntity<List<Products>> (product,HttpStatus.OK) ;
            return response;
        }
    }
}
```

ProductionApplication.java code:

```
package com.example.ProductsApplication;

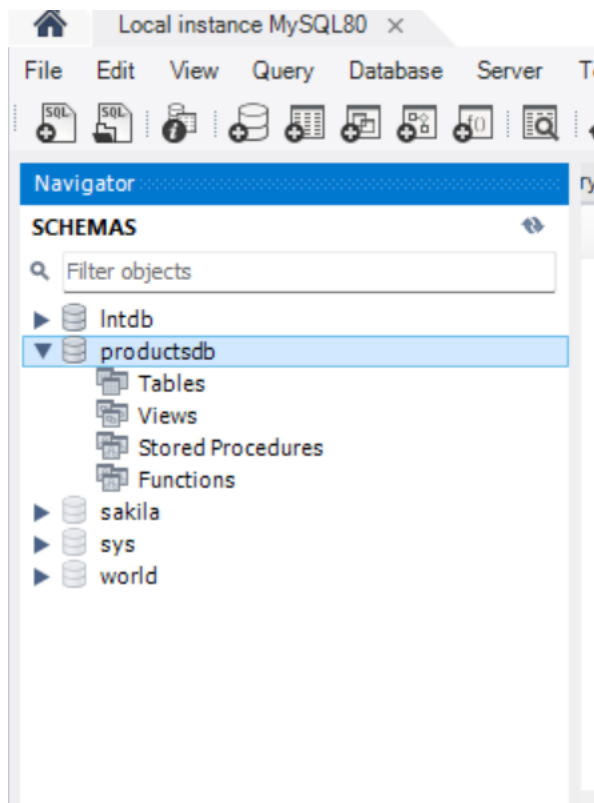
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class ProductsApplication {

    public static void main(String[] args) {
```

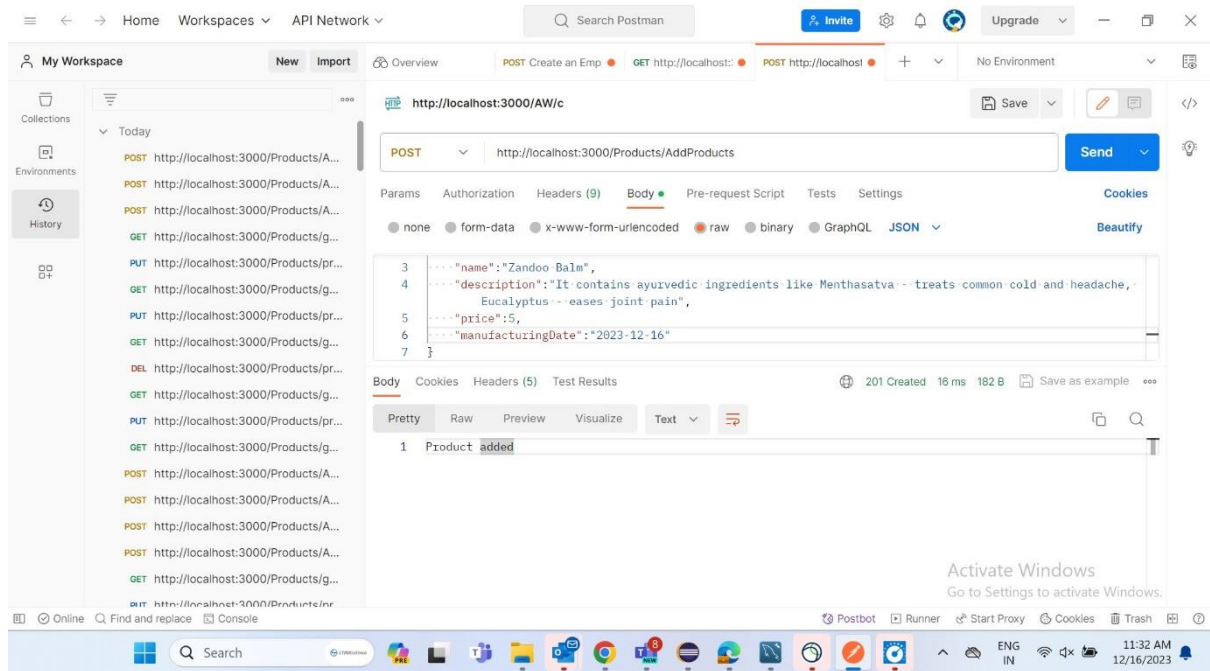
```
SpringApplication.run(ProductsApplication.class,  
args);  
}  
}
```

DataBase:

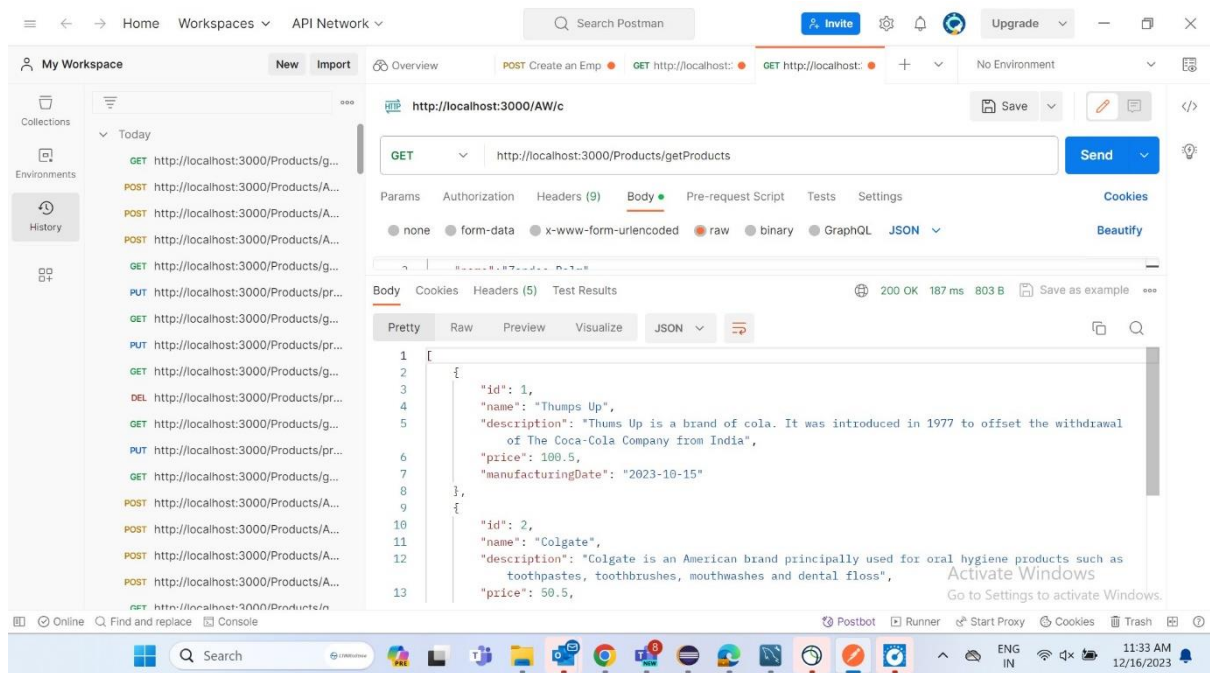


Crud Operations:

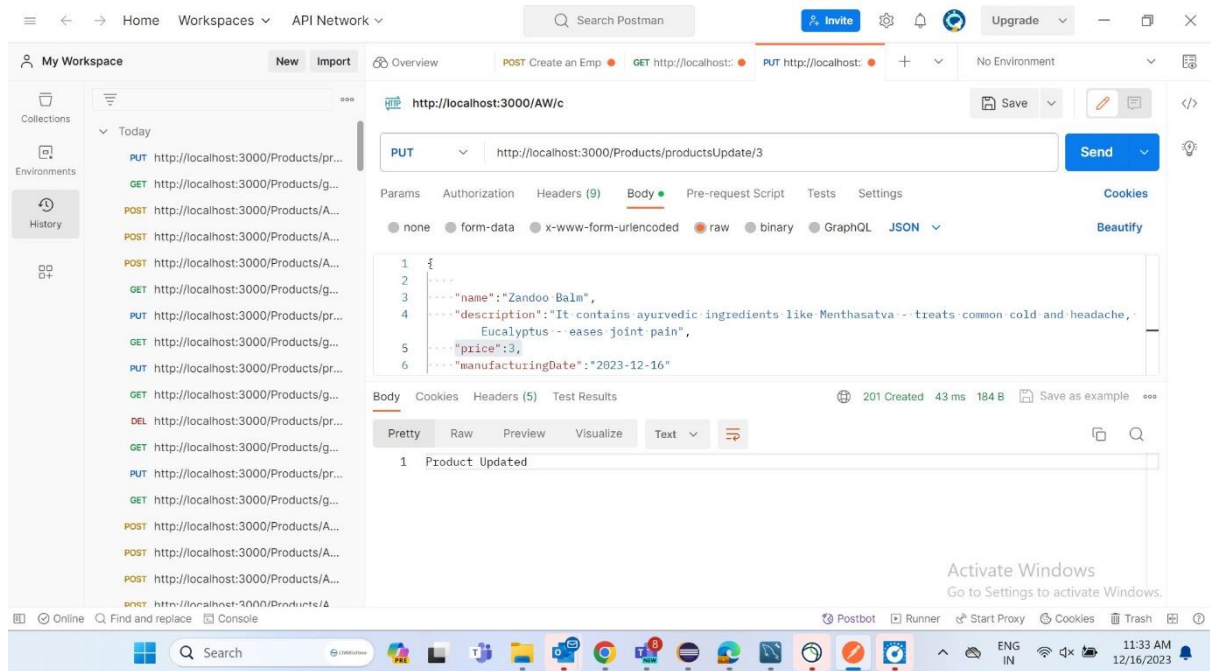
Create-**Post**: <https://localhost:3000/Products/AddProducts>



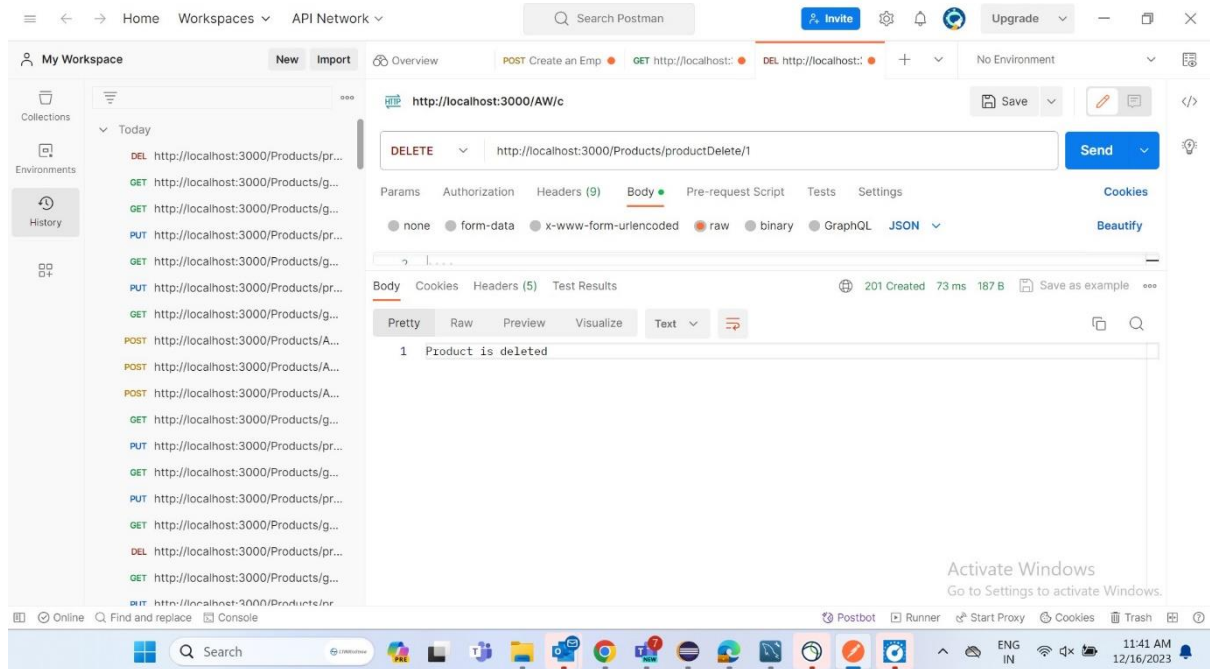
Get: <https://localhost:3000/Products/getProducts>



Update: <https://localhost:3000/Products/updateProducts/3>

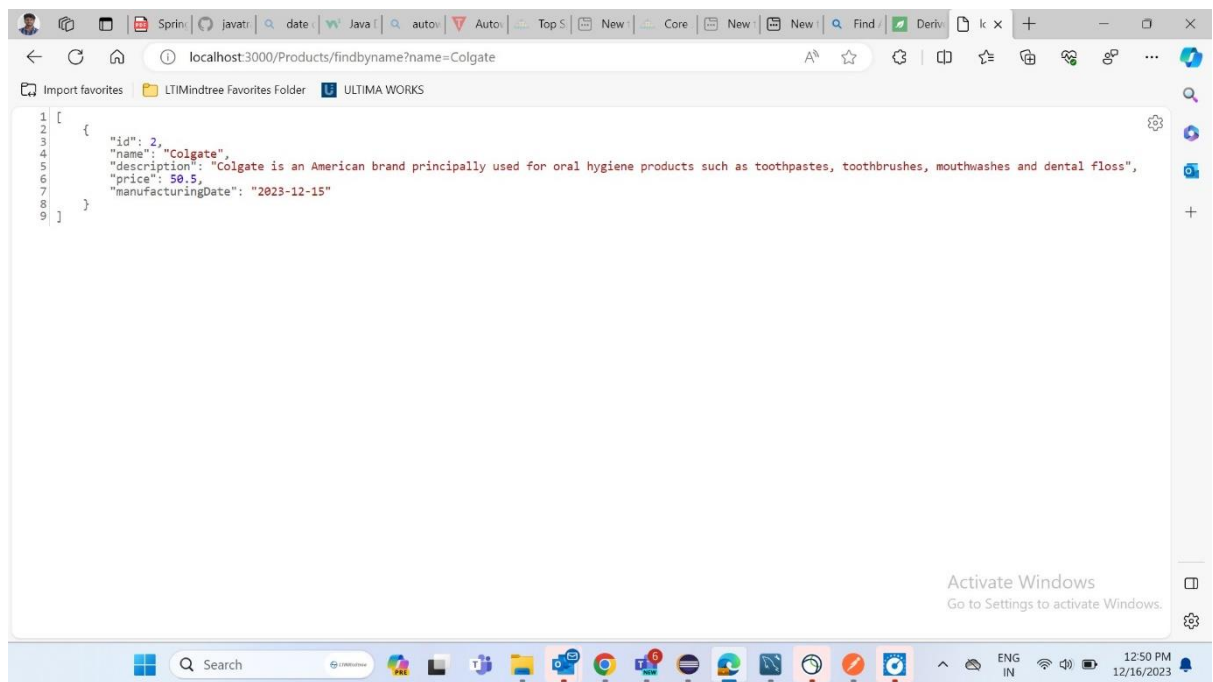


Delete: <https://localhost:3000/Products/productDelete/1>

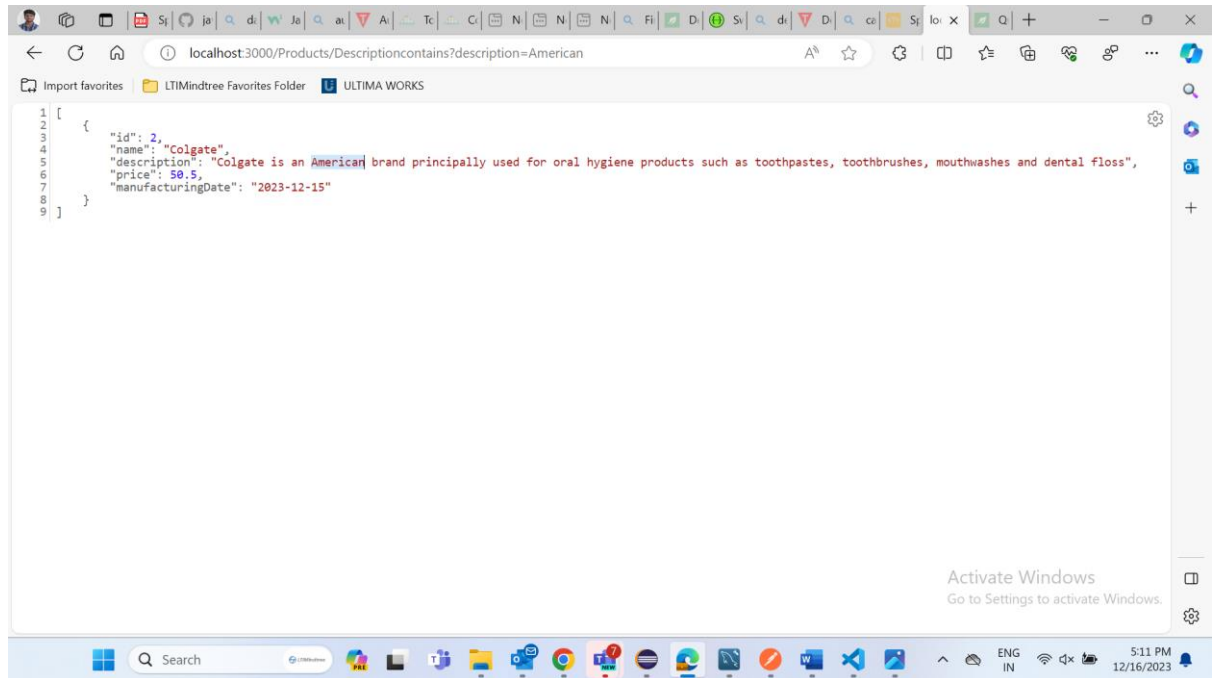


Find Operations:

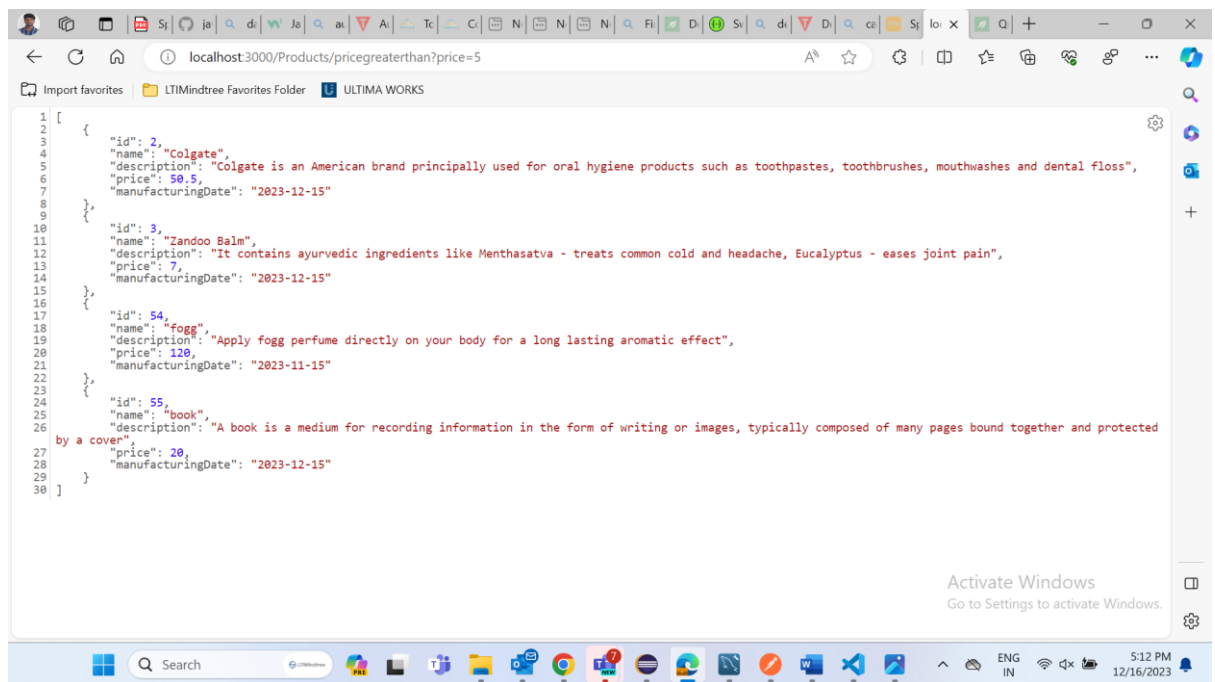
1. Find a product by its name



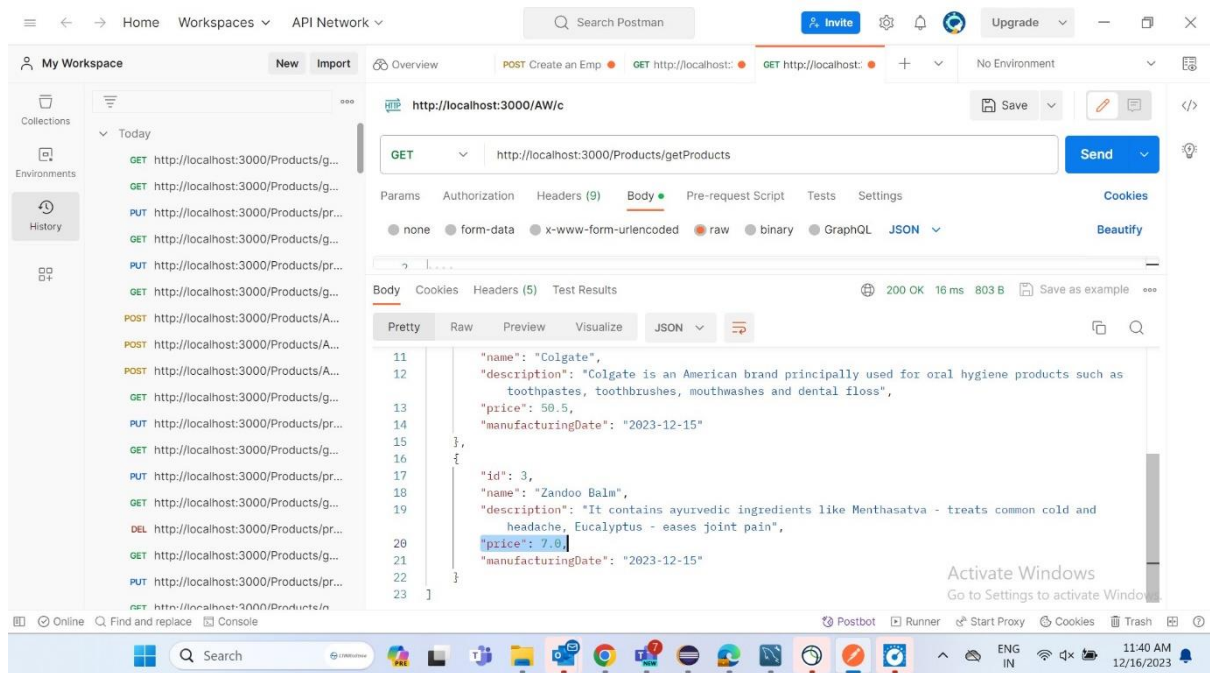
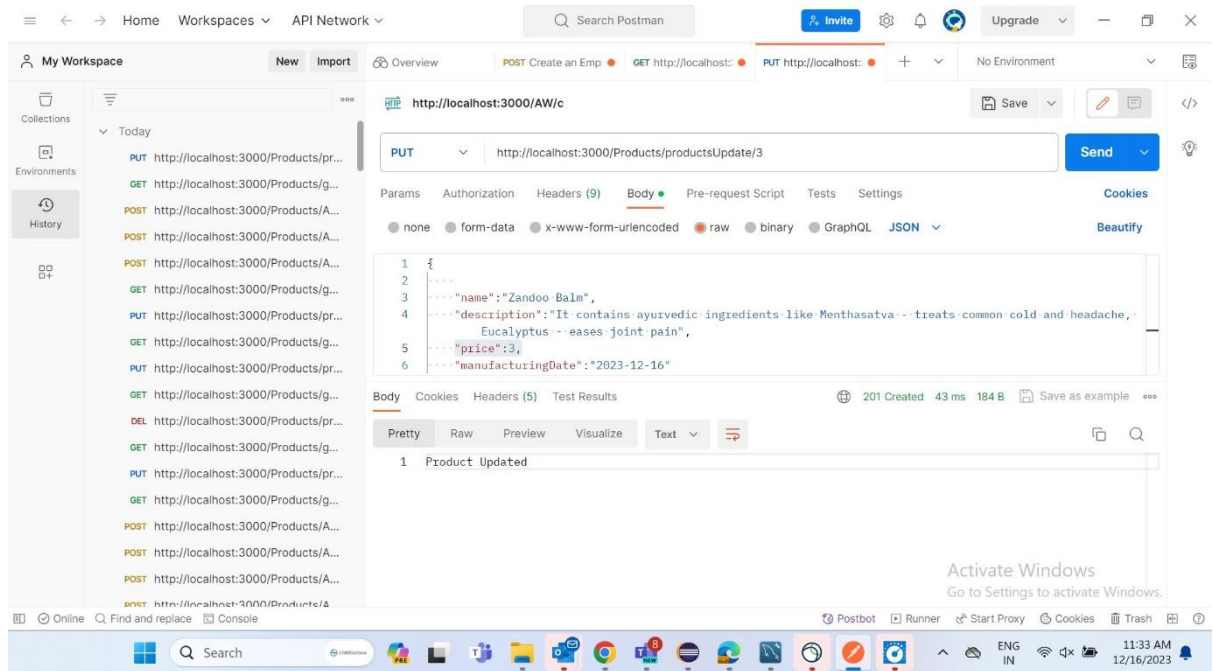
2. Search for a text contained in description



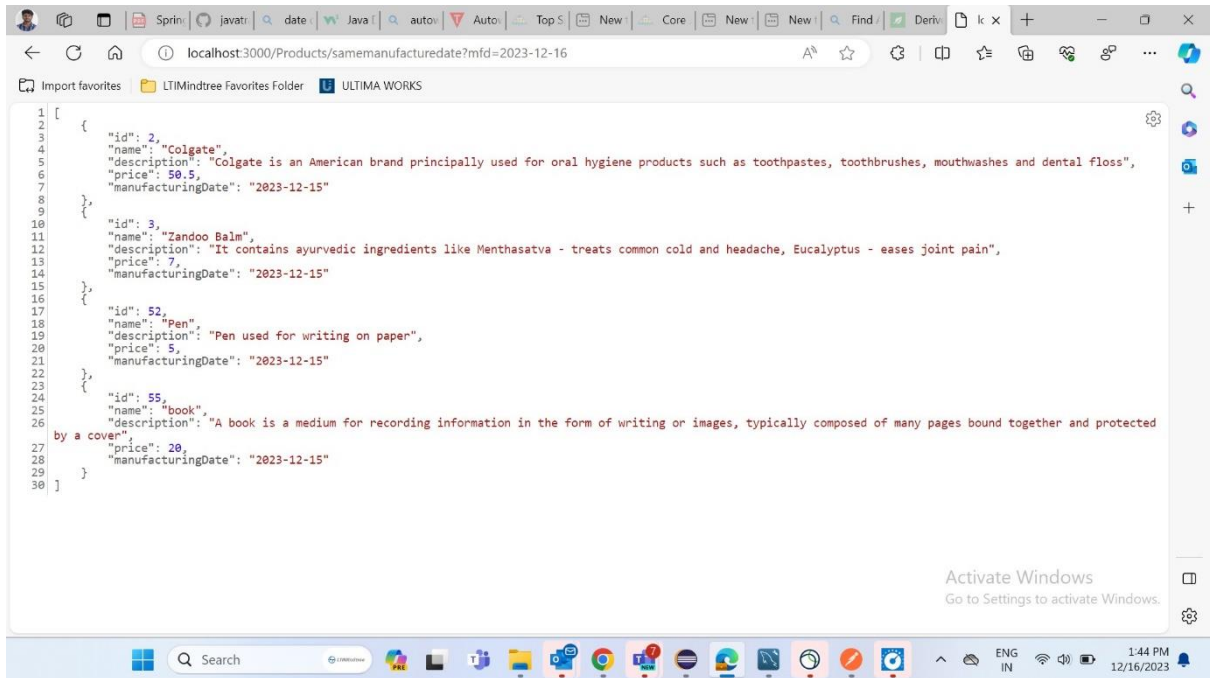
3. Products with price greater than a value.



4. Change the price of product

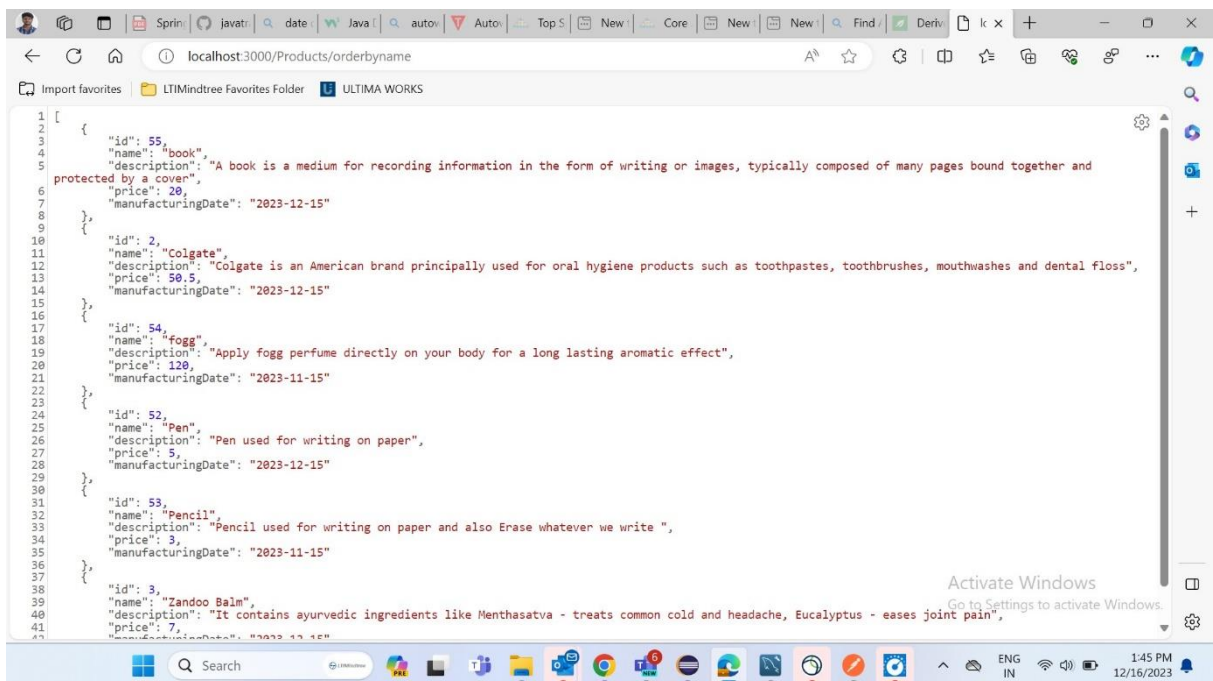


5. Find All Products by same Manufacturing Date



```
1 [
2   {
3     "id": 2,
4     "name": "Colgate",
5     "description": "Colgate is an American brand principally used for oral hygiene products such as toothpastes, toothbrushes, mouthwashes and dental floss",
6     "price": 50.5,
7     "manufacturingDate": "2023-12-15"
8   },
9   {
10    "id": 3,
11    "name": "Zandoo Balm",
12    "description": "It contains ayurvedic ingredients like Menthasatva - treats common cold and headache, Eucalyptus - eases joint pain",
13    "price": 7,
14    "manufacturingDate": "2023-12-15"
15  },
16  {
17    "id": 52,
18    "name": "Pen",
19    "description": "Pen used for writing on paper",
20    "price": 5,
21    "manufacturingDate": "2023-12-15"
22  },
23  {
24    "id": 55,
25    "name": "book",
26    "description": "A book is a medium for recording information in the form of writing or images, typically composed of many pages bound together and protected
27    by a cover",
28    "price": 20,
29    "manufacturingDate": "2023-12-15"
30  }
31 ]
```

6. Display products sorted by Name

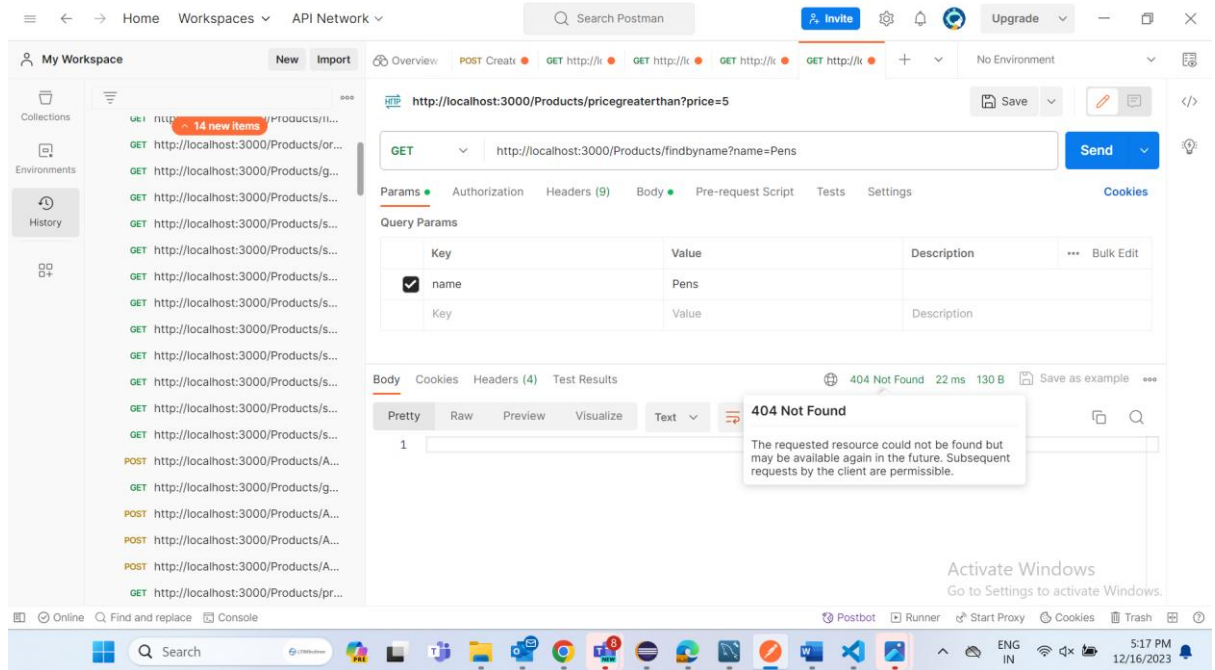


```
1 [
2   {
3     "id": 55,
4     "name": "book",
5     "description": "A book is a medium for recording information in the form of writing or images, typically composed of many pages bound together and
6     protected by a cover",
7     "price": 20,
8     "manufacturingDate": "2023-12-15"
9   },
10  {
11    "id": 2,
12    "name": "Colgate",
13    "description": "Colgate is an American brand principally used for oral hygiene products such as toothpastes, toothbrushes, mouthwashes and dental floss",
14    "price": 50.5,
15    "manufacturingDate": "2023-12-15"
16  },
17  {
18    "id": 54,
19    "name": "fogg",
20    "description": "Apply fogg perfume directly on your body for a long lasting aromatic effect",
21    "price": 120,
22    "manufacturingDate": "2023-11-15"
23  },
24  {
25    "id": 52,
26    "name": "Pen",
27    "description": "Pen used for writing on paper",
28    "price": 5,
29    "manufacturingDate": "2023-12-15"
30  },
31  {
32    "id": 53,
33    "name": "Pencil",
34    "description": "Pencil used for writing on paper and also Erase whatever we write ",
35    "price": 3,
36    "manufacturingDate": "2023-11-15"
37  },
38  {
39    "id": 3,
40    "name": "Zandoo Balm",
41    "description": "It contains ayurvedic ingredients like Menthasatva - treats common cold and headache, Eucalyptus - eases joint pain",
42    "price": 7,
43    "manufacturingDate": "2023-12-15"
44  }
45 ]
```

The REST API must fulfil following requirements:

1. Use appropriate status codes :For every operation I used status codes

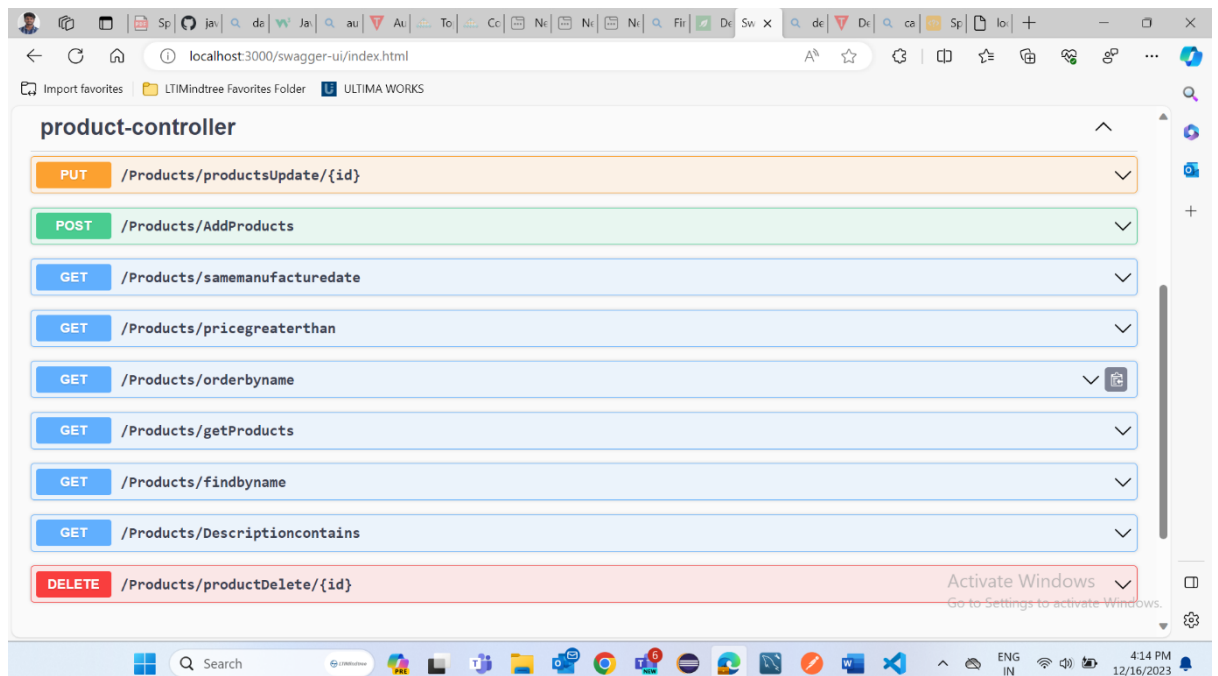
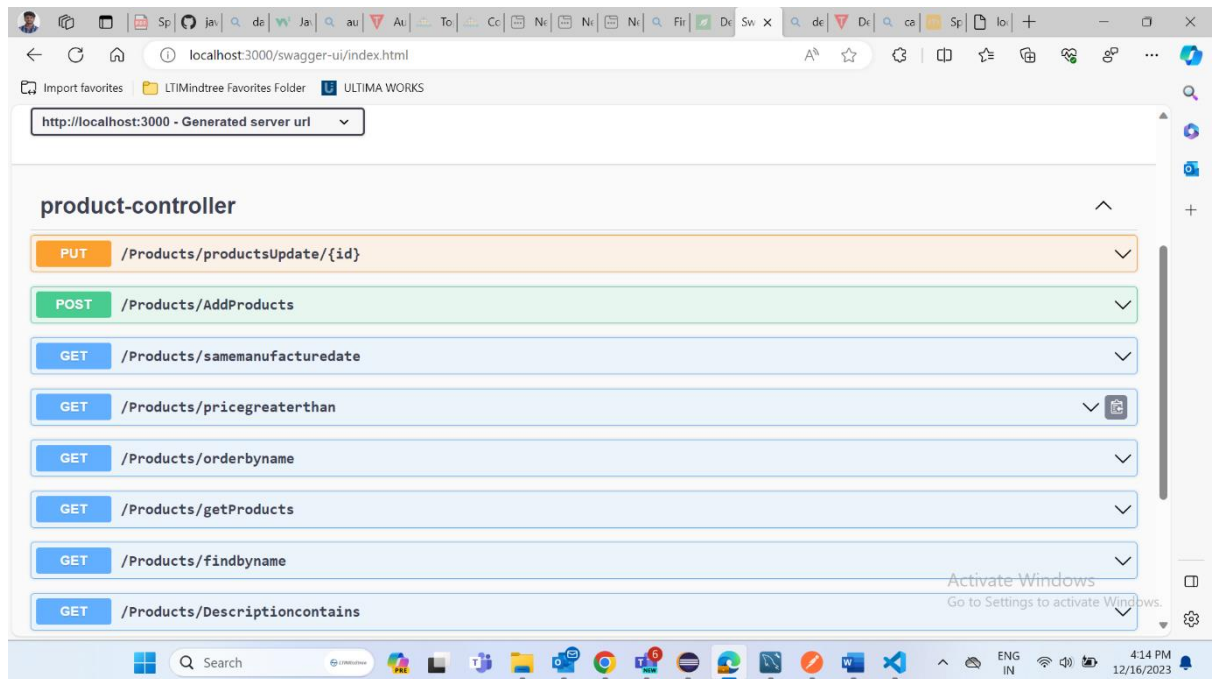
Example: when I search for particular name, if it found it will give results else it will show Not Found Status

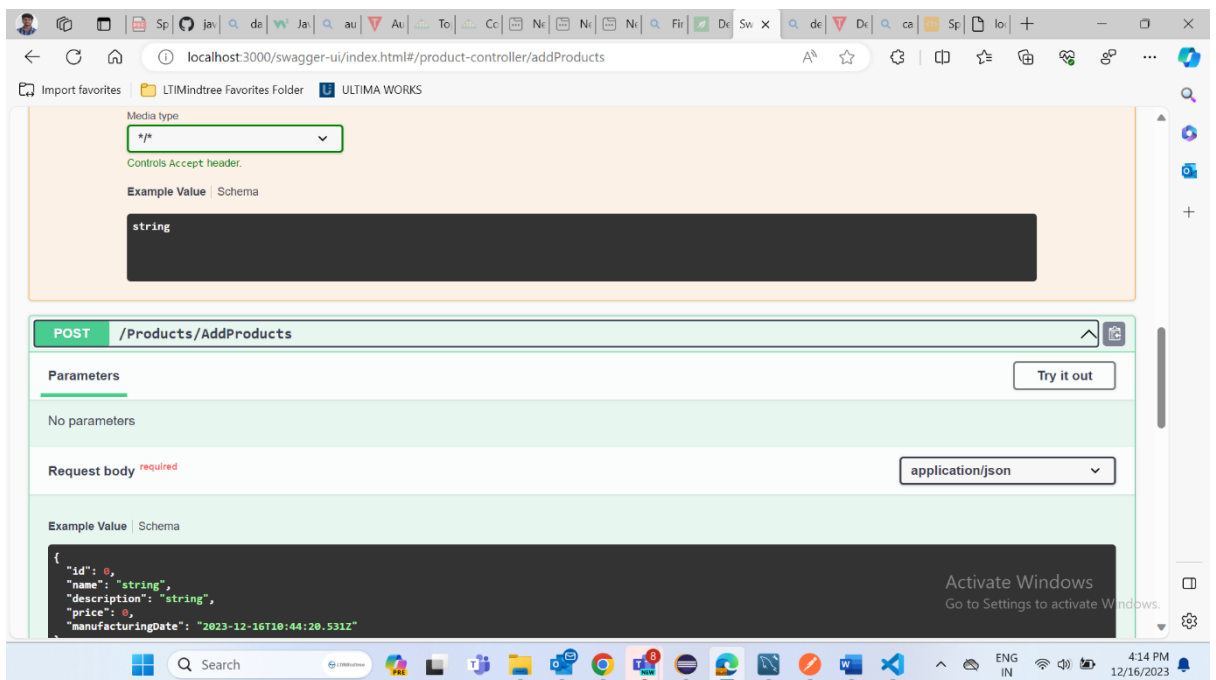
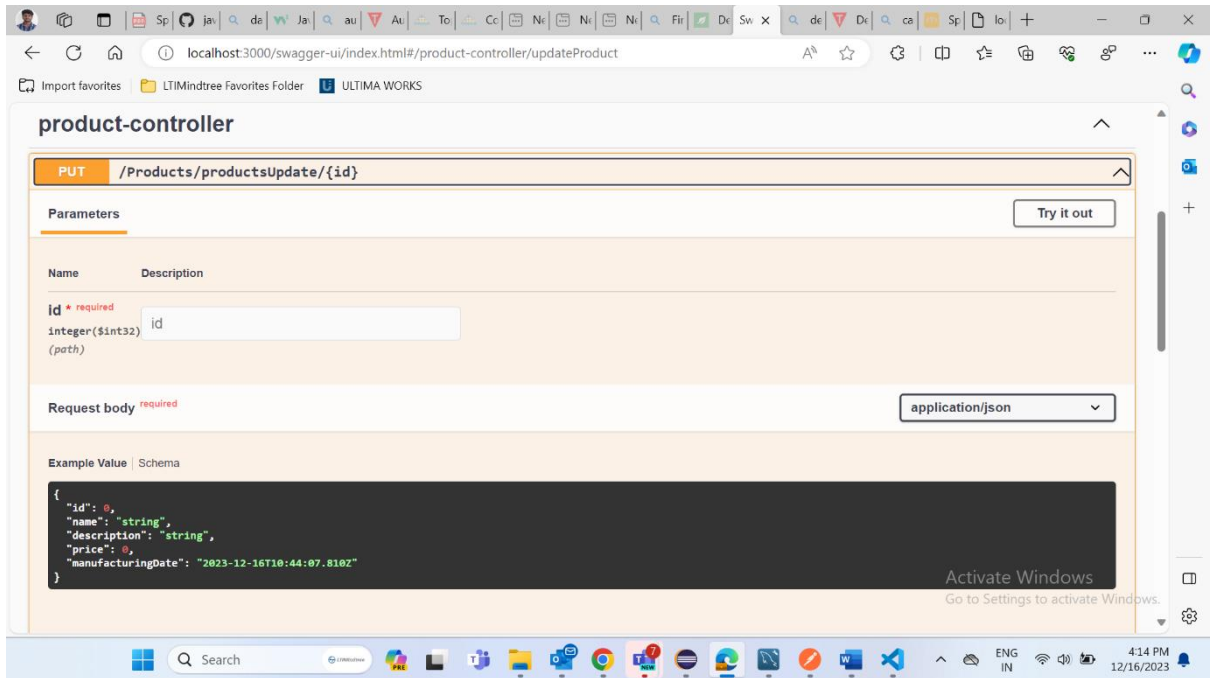


2. Use exception handling

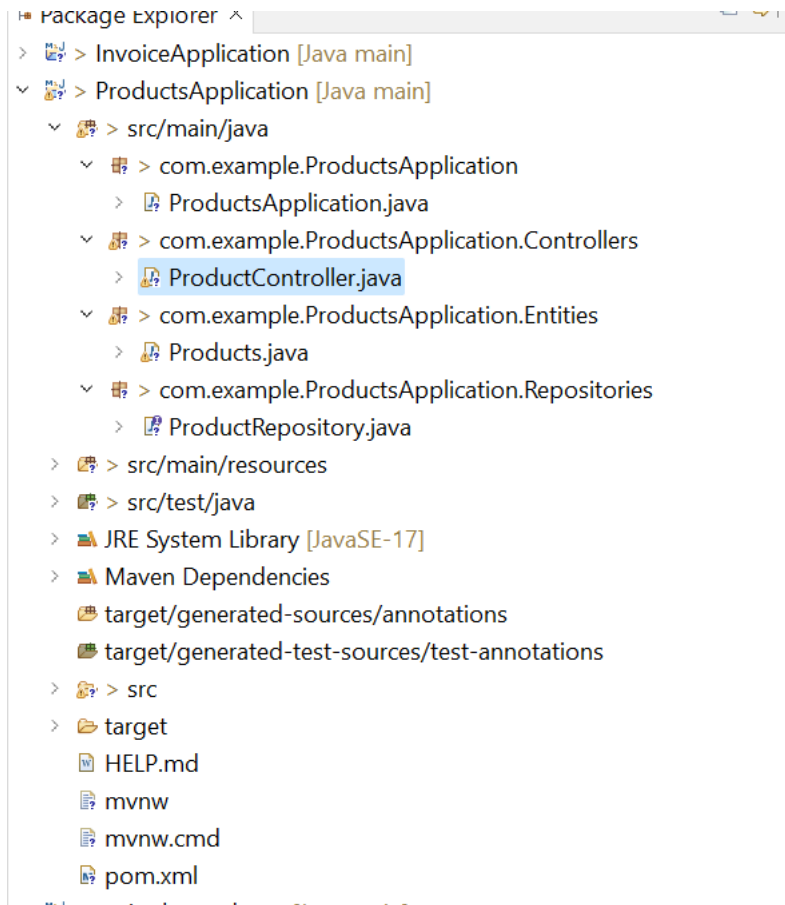
```
public ResponseEntity<String> addProducts(@RequestBody Products products) {  
    ResponseEntity response=null;  
    try {  
        productRepository.save(products);  
        response= new ResponseEntity<String>(  
            "Product added",HttpStatus.CREATED);  
    }  
    catch(Exception e){  
        response=new ResponseEntity<String>("Adding products failed",HttpSta  
    }  
    return response;  
}  
@GetMapping("/getProducts")  
@ResponseStatus(HttpStatus.OK)  
public List<Products> getProducts() {  
    return productRepository.findAll();  
}
```

3. Must expose documentation using Swagger





5 Use separate packages for Controllers, Entities, Repositories etc



**Let's get to the
future, faster.
Together.**

