EDUBRIDGE

COURSE

Certified Java Full Stack Professional

PROJECT NAME

INVENTORY MANAGEMENT SYSTEM

Under the Guidence of

Trainer Mrs. Indrakka Mali Mam

Presented By

Ms.Deepika S

Ms.Lokeshwari S

Mrs.Renugadevi R

Ms.Shipra Mahato

Mrs.Sri Vidhya C

AGENDA

- Introduction
- Objectives
- Modules Description
- CRUD Operation
- Annotation
- Screenshot
- Conclusion

INTRODUCTION

Inventory management System

- An inventory management system is the process by which you track your goods throughout your entire supply chain, from purchasing to production to end sales. In our project we can implement the shopkeeper module maintain the product details and store all the information of customer on the database.

Functions of Inventory Management System

- Increase productivity: An inventory management system enhances the productivity of the company to a greater extent. Avoid stock-outs and over-stocks: An inventory management system assists companies with avoiding over-stocks and stock-out scenarios. So, the company never stops functioning.
- Manage quality and Inspect products: This system is responsible for quality inspection and management of the product.

OBJECTIVES

- It provides “better and efficient” service”.
- Faster way to get details about the product and customer information.
- Provide facility for proper monitoring and reduce paper work.
- All details will be available on a click.

System View

- The Inventory management system will be automated the traditional system.
- There is no need to use paper and pen. Checking a product details is very easy.
- Adding new customer record is very easy. Creating, Updating, or Deleting a product and customer field is very to implement using some certain methods.

MODULE DESCRIPTION

Shopkeeper Module

- In this class we implement the information of product details, and create the join column to combine the customer class to view the details of product to buy the customer. Shopkeeper class using variables are total stock, product id, product name, product type, product brand, product quality, product price, current stock and total amount.
- Details of product will be stored on the database will be viewed using the postman get the details of stock description.
- Using GET method to view the all details of customer details.

MODULE DESCRIPTION

Customer Module

- In Customer Class used to define the customer details and product brought by customer.
- Custom variables are customer id, customer name, customer mobile number and customer address are used on this class connection to shopkeeper class.
- Fetch the details of product brought by customer using foreign key reference get the product details of customer

CRUD OPERTION

- CRUD stands for Create, Read/Retrieve, Update and Delete and these are the four basic operations that we perform on persistence storage.CRUD is data-oriented and the standardized use of HTTP methods.
- So, standard CRUD Operations is as follows:
  - POST: Creates a new resource
  - GET: Reads/Retrieve a resource
  - PUT: Updates an existing resource
  - DELETE: Deletes a resource

As the name suggests

• CREATE Operation: Performs the INSERT statement to create a new record. READ Operation: Reads table records based on the input parameter. UPDATE Operation: Executes an update statement on the table. It is based on the input parameter. DELETE Operation: Deletes a specified row in the table. It is also based on the input parameter

ANNOTATIONS

1.**@entity**: The @Entity annotation specifies that the class is an entity and is mapped to a database table.

2.**@Table**: annotation specifies the name of the database table to be used for mapping.

3. **@Id** : annotation used for the primary key in database

4.**@NotNull** : content is not null on the table.

5.**@NotBlank** : used to content is not empty on the table

6.**@length**: used to specify the length of content placed on the table.

ANNOTATION

7.**@Service**: We mark beans with @Service to indicate that they're holding the business logic.

8. **@Repository**: To catch persistence-specific exceptions and re-throw them as one of spring’s unified unchecked exceptions.

9**. @Autowired**: The spring framework enables automatic dependency injection

10**.@RestController**:This annotation is used at the class level and allows the class to handle the requests made by the client.

ANNOTATION

11.**@GetMapping**: is a specialized version of @RequestMapping annotation that acts as a shortcut for @RequestMapping(method = RequestMethod.GET).

12. **@PostMapping**: The methods in the @Controller annotated classes handle the HTTP POST requests matched with given URI expression.

13. **@DeleteMapping**: maps HTTP DELETE requests onto specific handler methods.

14. **@RequestBody**: annotation maps the HttpRequest body to a transfer or domain object, enabling automatic deserialization of the inbound HTTP Request body onto a Java object.

ANNOTATIONS

15**. @OneToMany**: A one-to-many relationship between two entities is defined by using the @OneToMany annotation in Spring Data JPA.

16. **@generatedValue**: Marking a field with the annotation specifies that a value will be automatically generated for that field.

17. **@SequenceGenerator** :The defines a primary key generator that may be referenced by name when a generator element is specified for the GeneratedValue annotation.

18**.@JoinColumn**: Used to combine the foreign key column on this table.

ANNOTATIONS

19.**@SpringBootApplication** :annotation is used to mark a configuration class that declares one or more @Bean methods and also triggers auto-configuration .

20.**@ComponentScan** annotation: is used with the @Configuration annotation to tell Spring the packages to scan for annotated components.

21.**@ResponseStatus**: marks a method or exception class with the status code and reason message that should be returned.

ANNOTATIONS

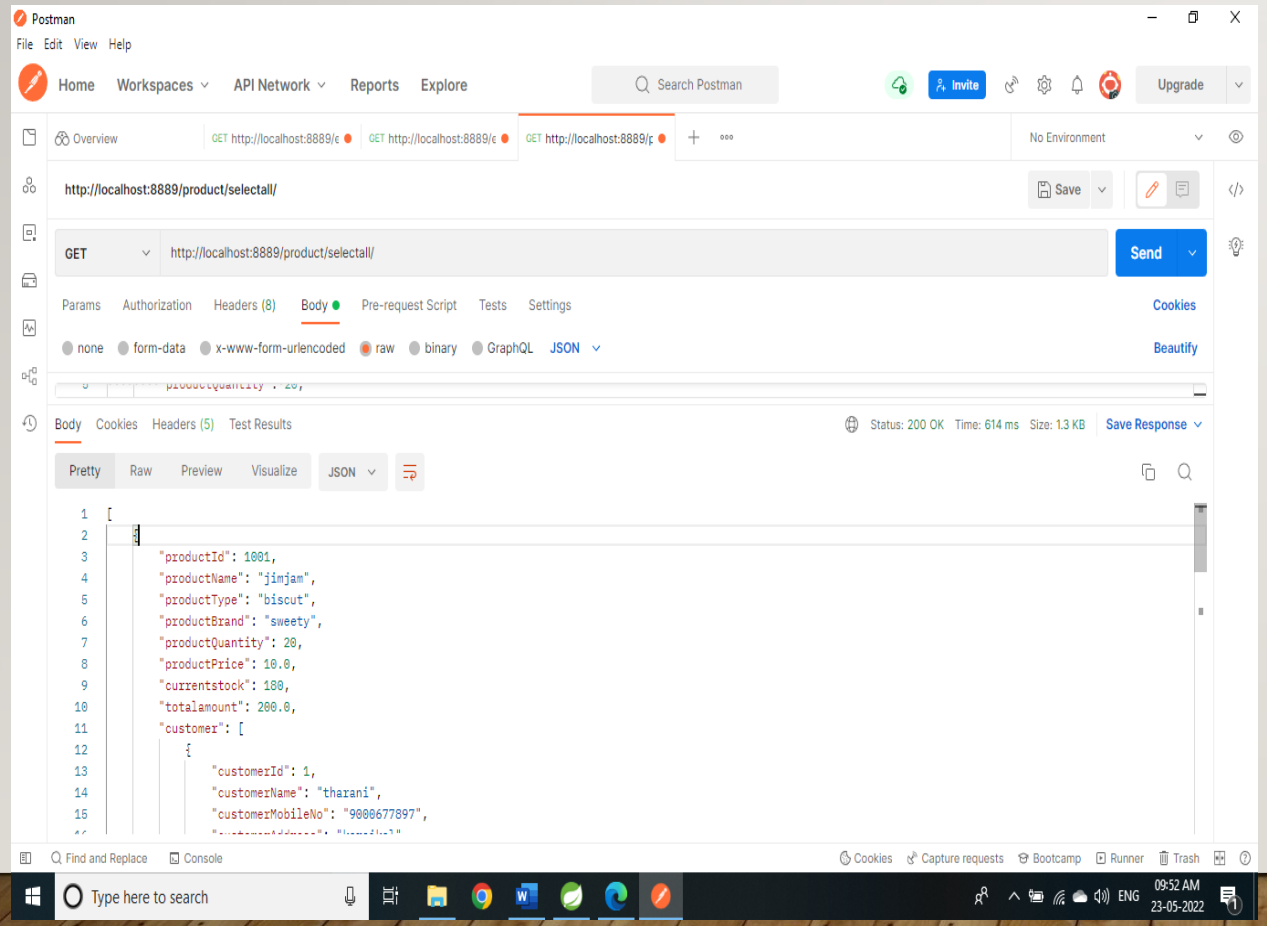
22.**@ExceptionHandler**: The annotation is used to annotate the method(s) in the controller class for handling the exceptions raised during the execution of the controller methods.

23.**@Column**: Annotation is used to change the table name on database.

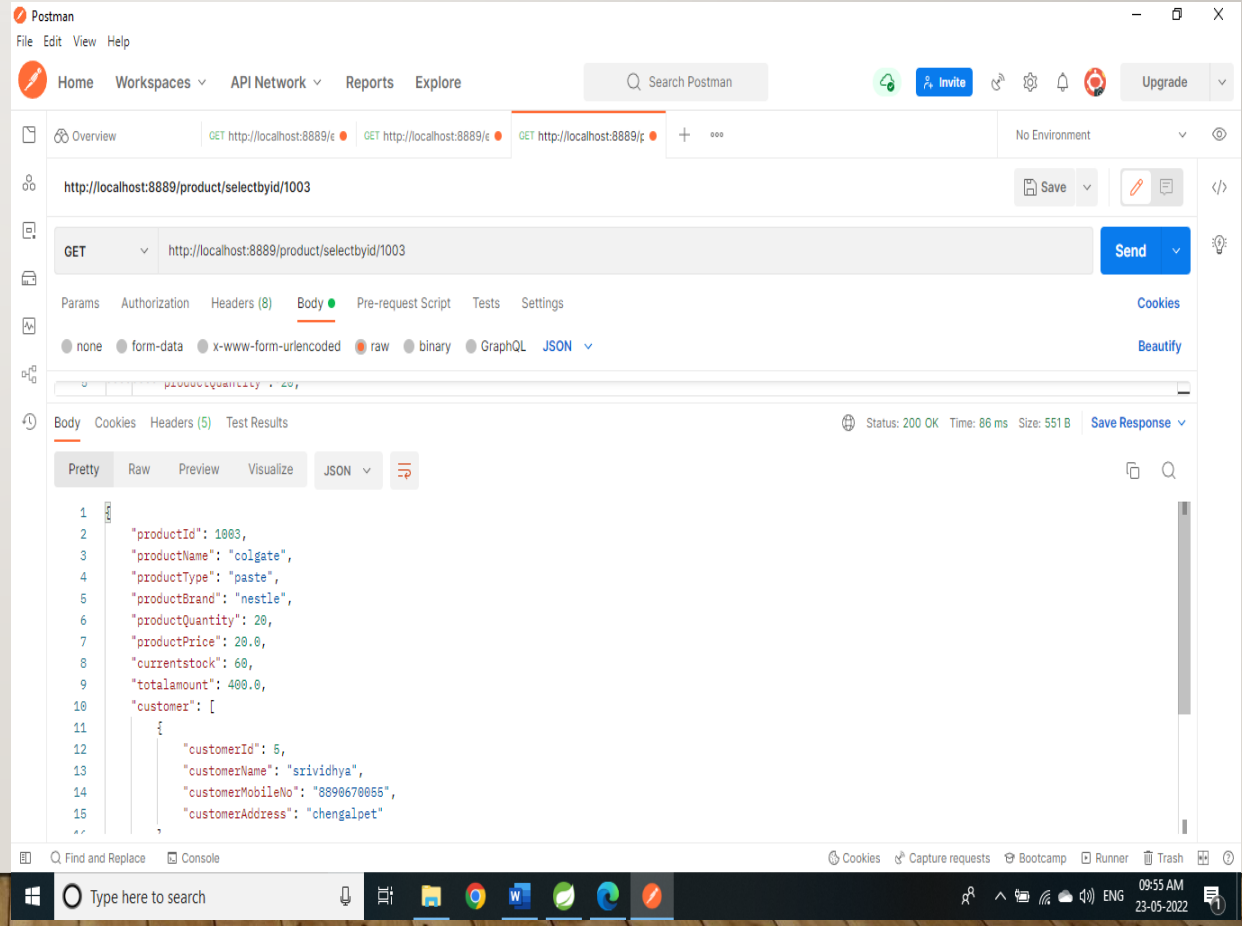
24.**@PathVariable**: Spring annotation which indicates that a method parameter should be bound to a URI template variable.

SCREENSHOT

Get: select all products
http://localhost:8889/product/selectall/

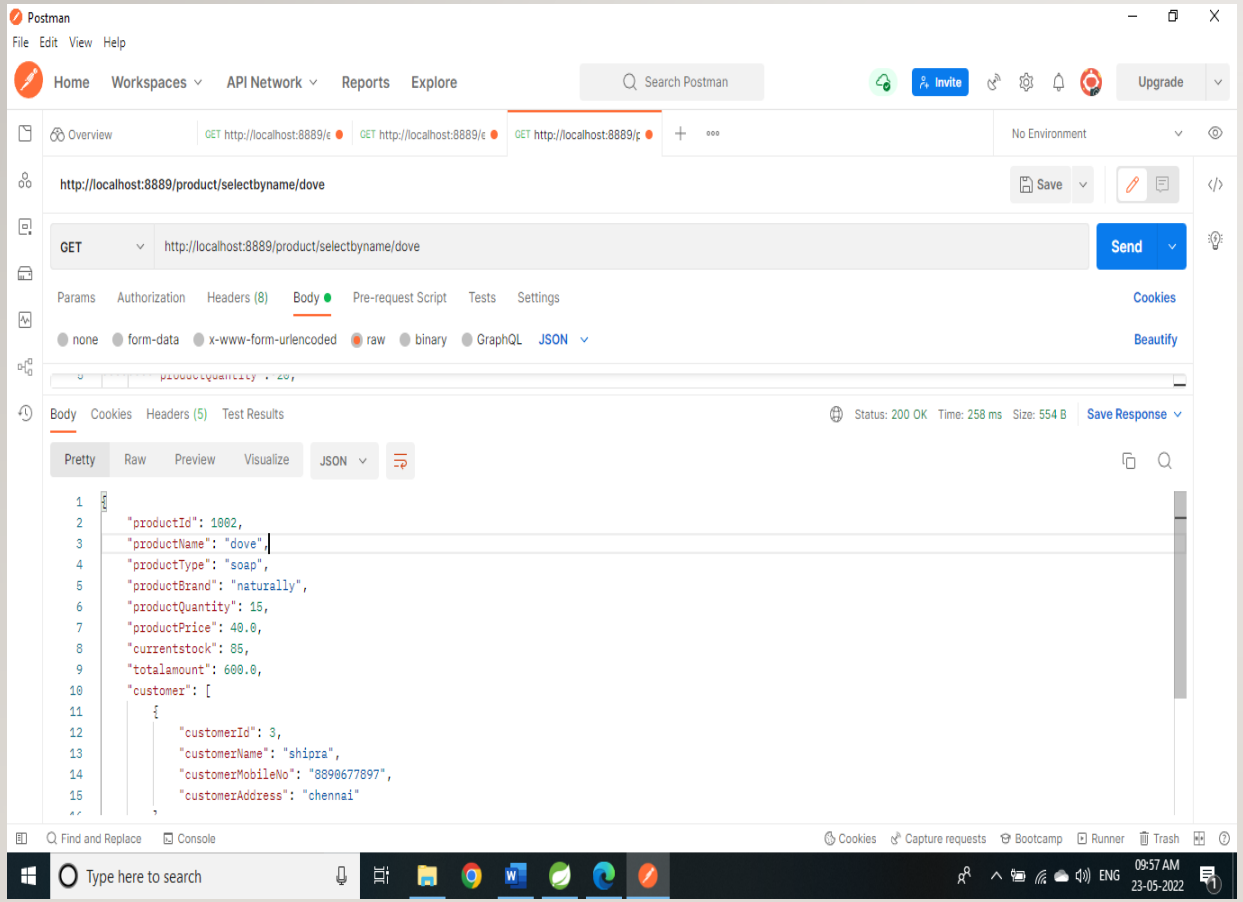


Get: select product by id:
http://localhost:8889/product/selectbyid/1003

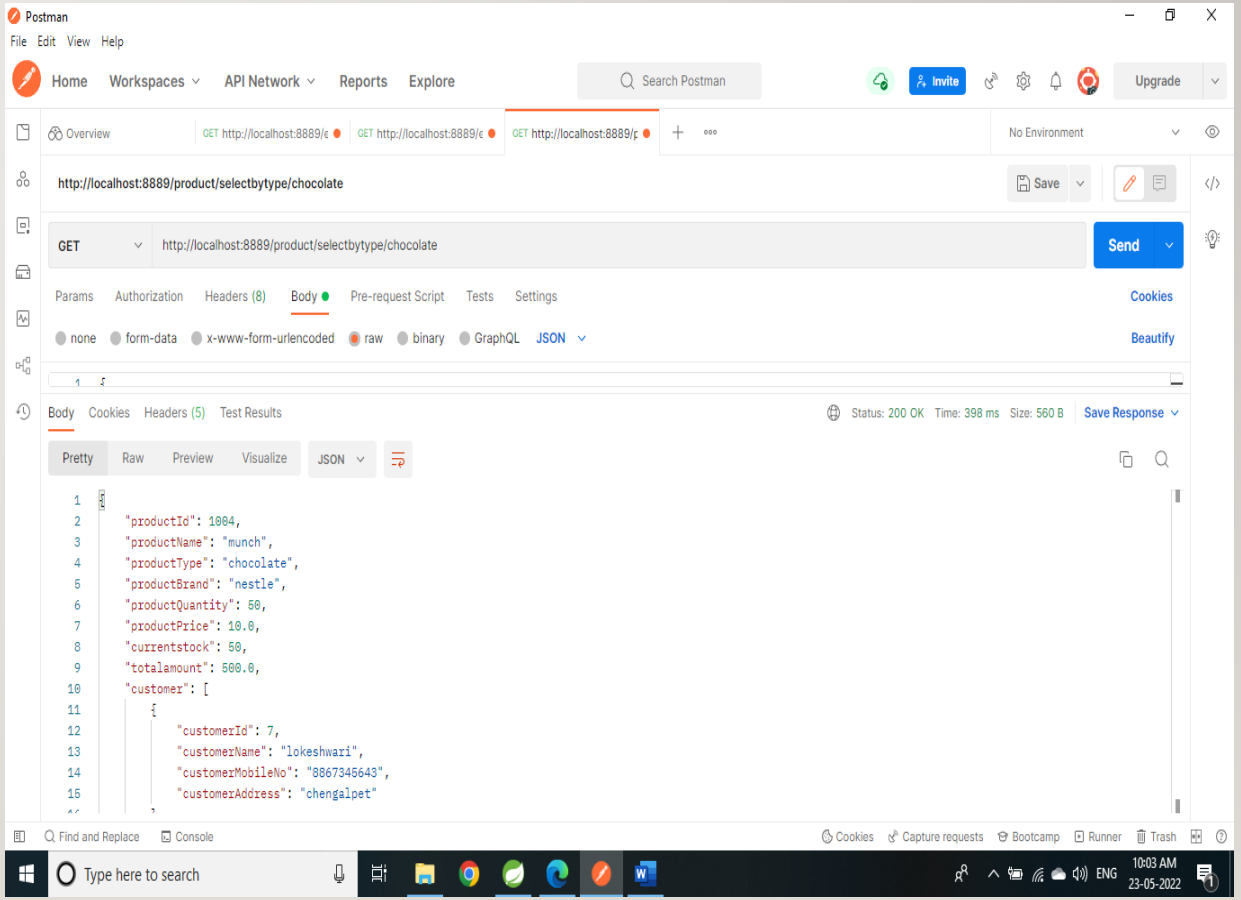


SCREENSHOT

Get: select product by name
http://localhost:8889/product/selectbyname/dove

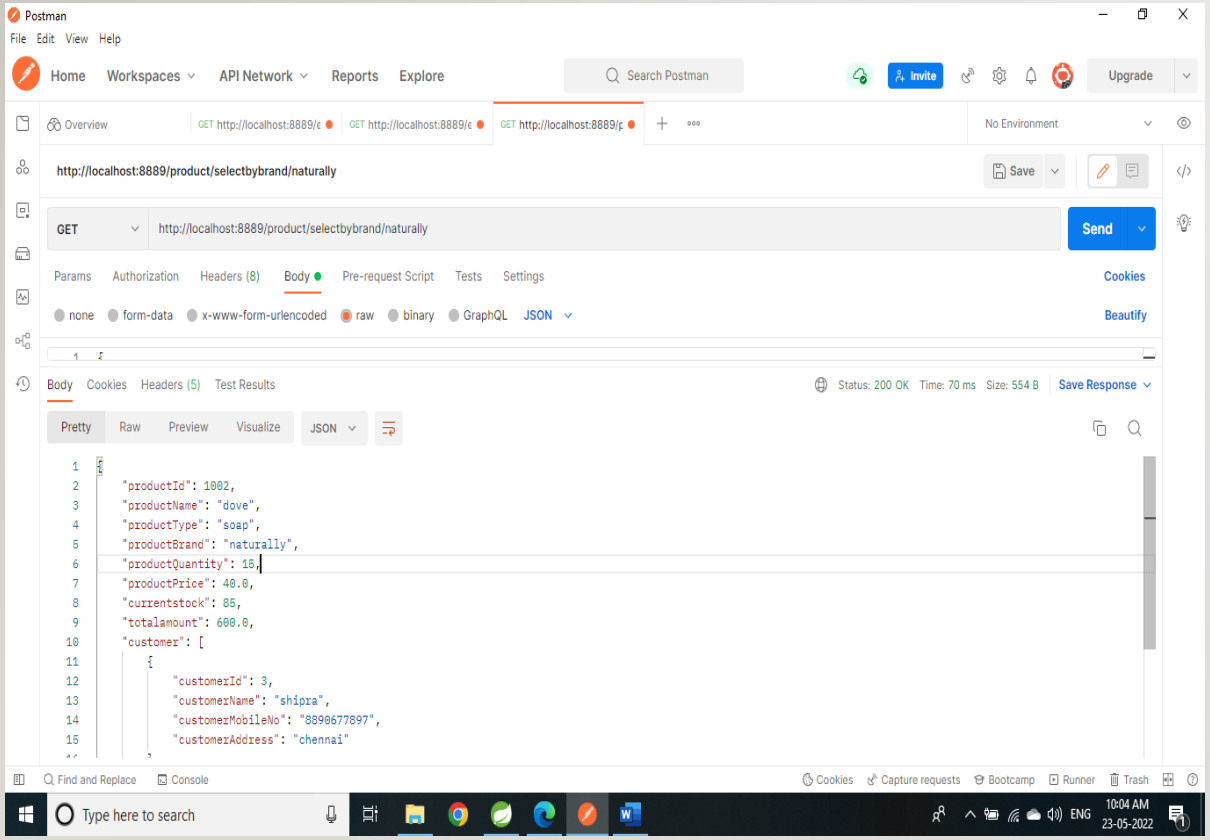


Select product by type: Get:
http://localhost:8889/product/selectbytype/chocolate

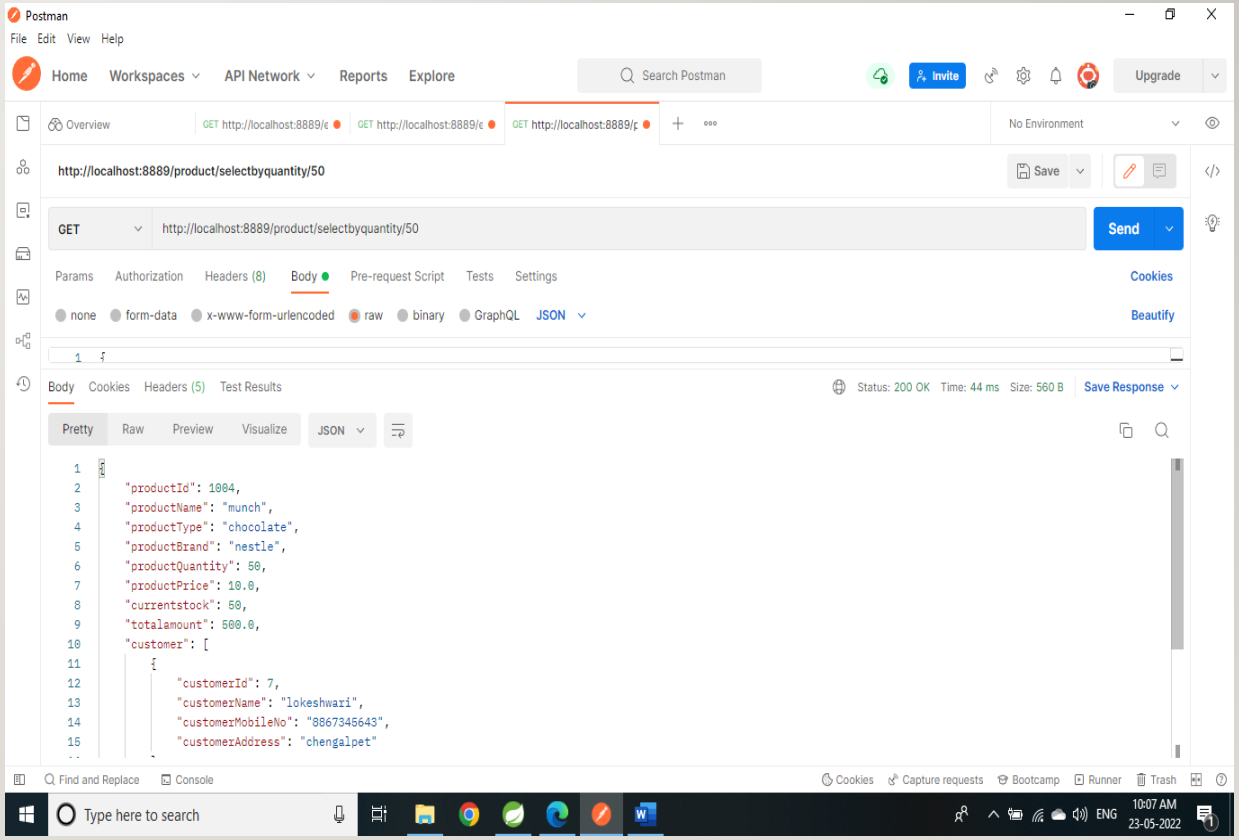


SCREENSHOT

Select product by brand:
http://localhost:8889/product/selectbybrand/naturally

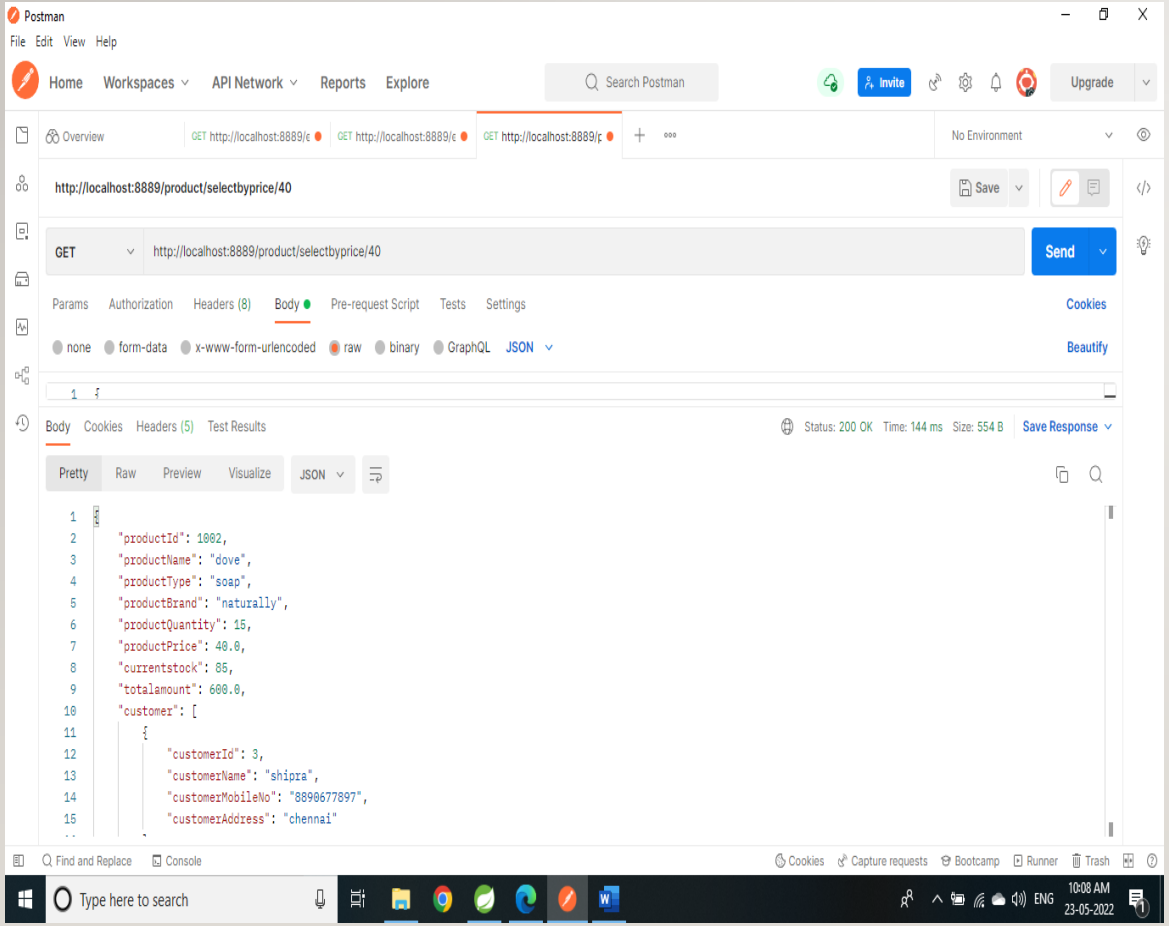


Select product by quantity:
http://localhost:8889/product/selectbyquantity/50

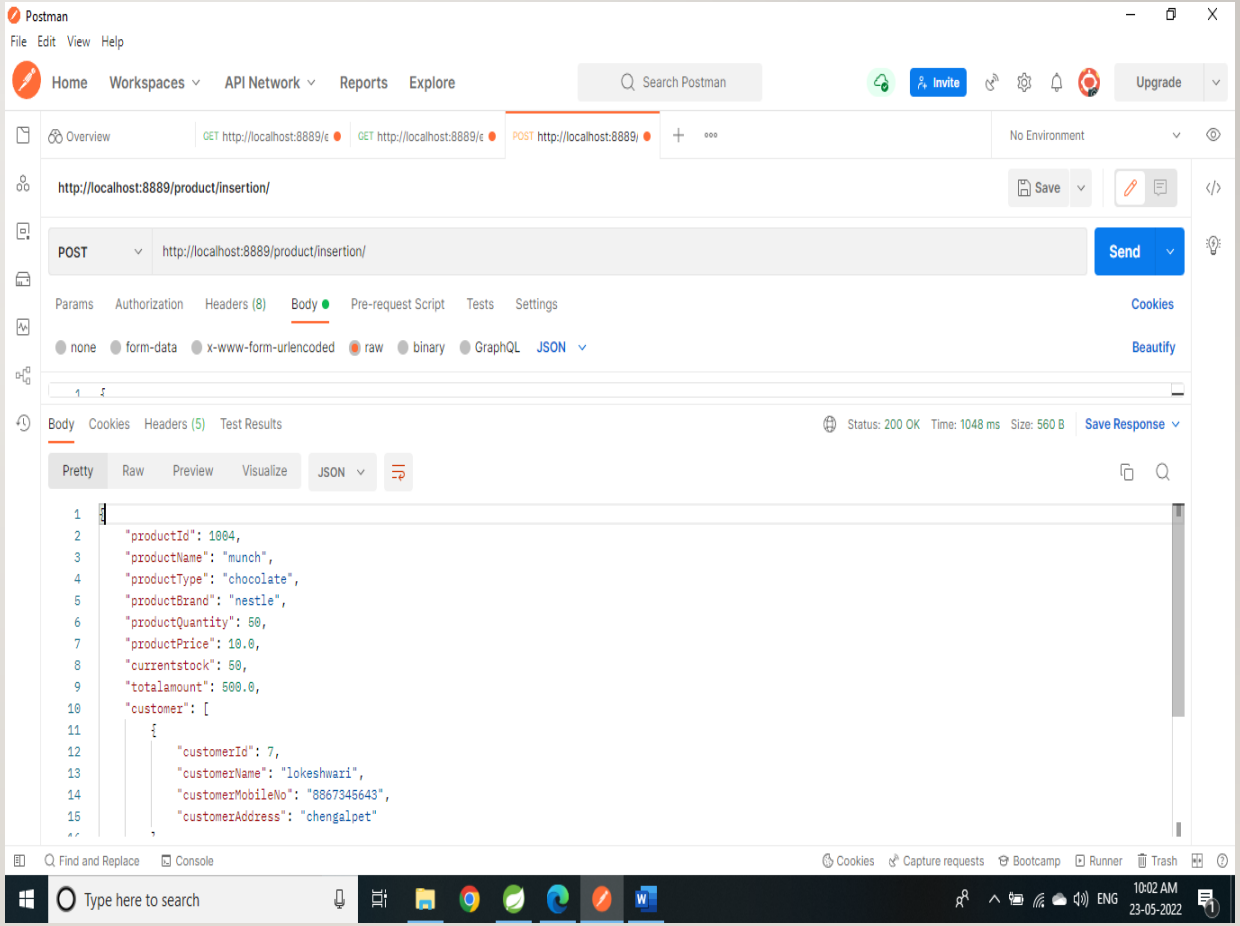


SCREENSHOT

Select by price:
http://localhost:8889/product/selectbyprice/40

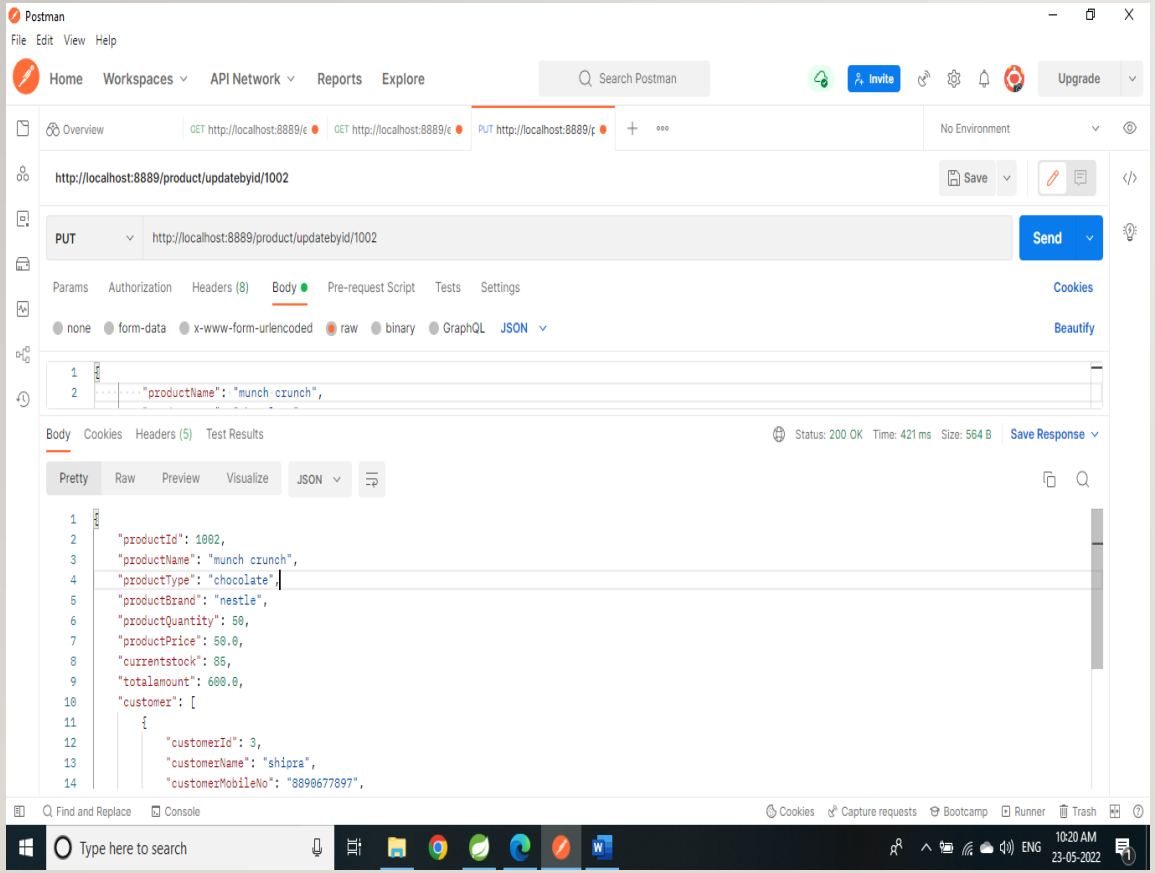


Post: insertion
http://localhost:8889/product/insertion/

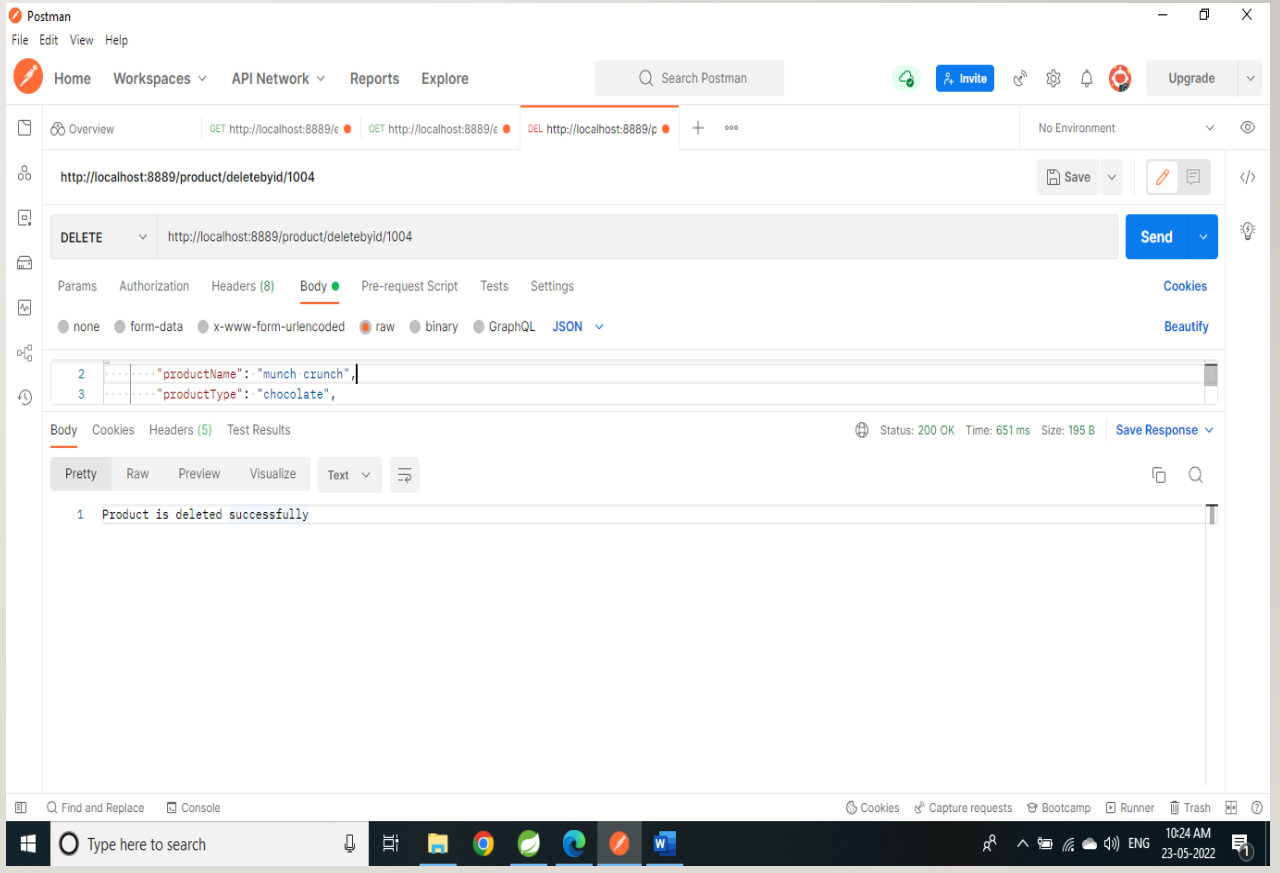


SCREENSHOT

Update product by id:
Put: http://localhost:8889/product/updatebyid/1002



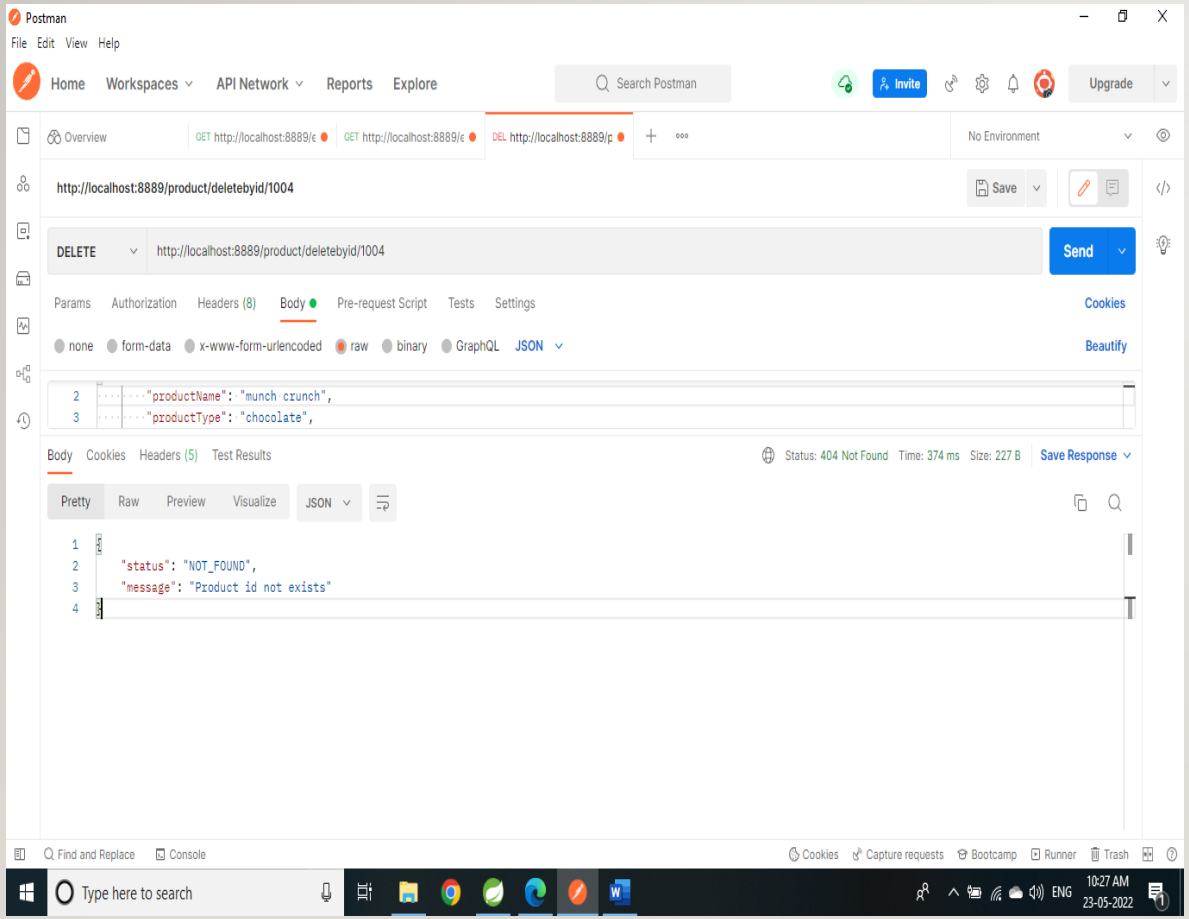
Delete product by id:
http://localhost:8889/product/deletebyid/1004



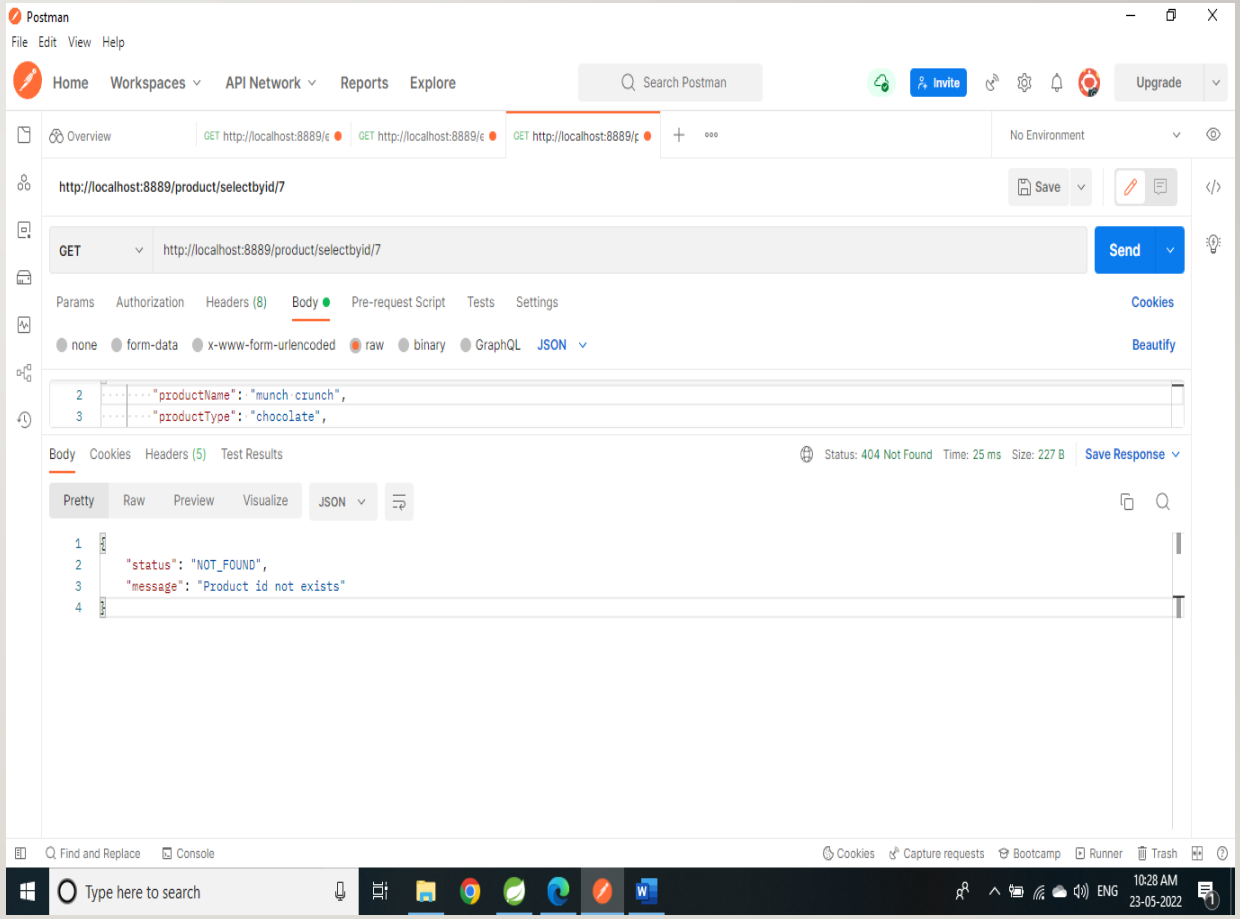
SCREENSHOT

Not found display:

Delete: http://localhost:8889/product/deletebyid/1004

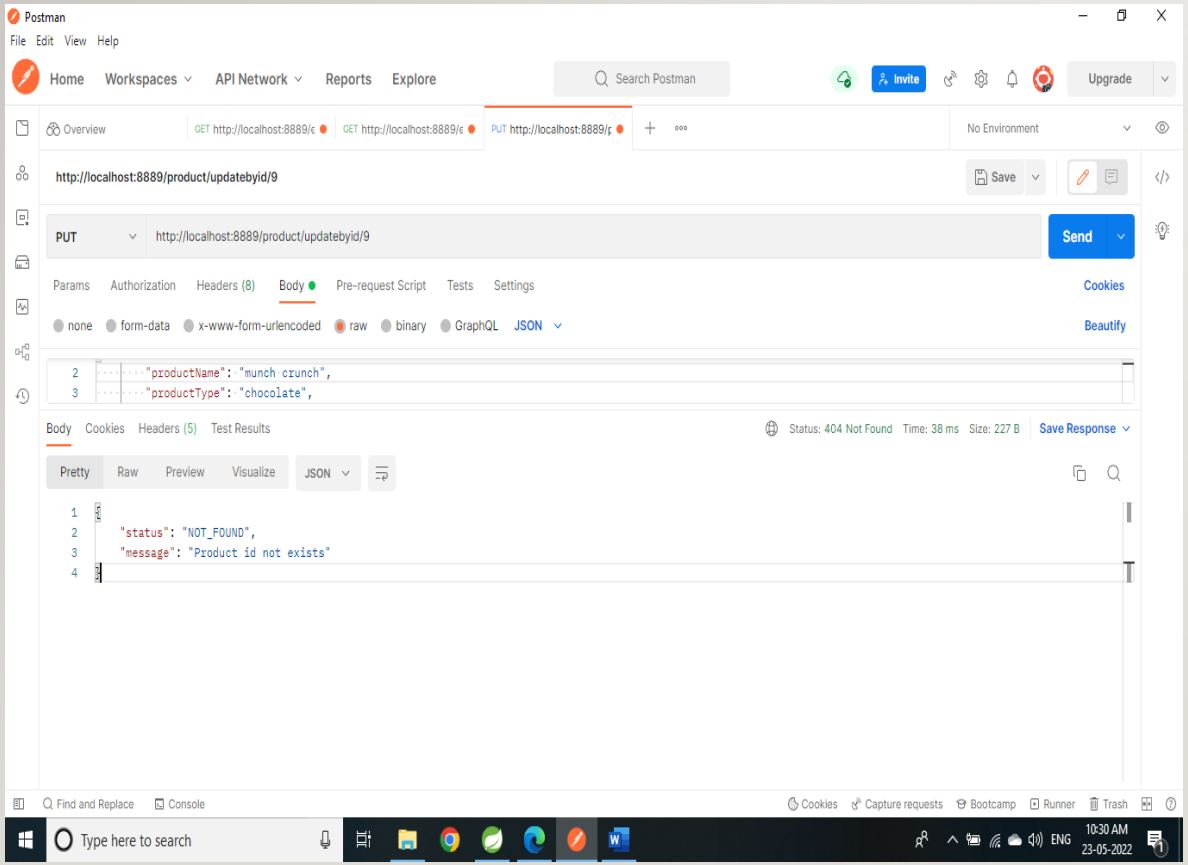


Get : http://localhost:8889/product/selectbyid/7

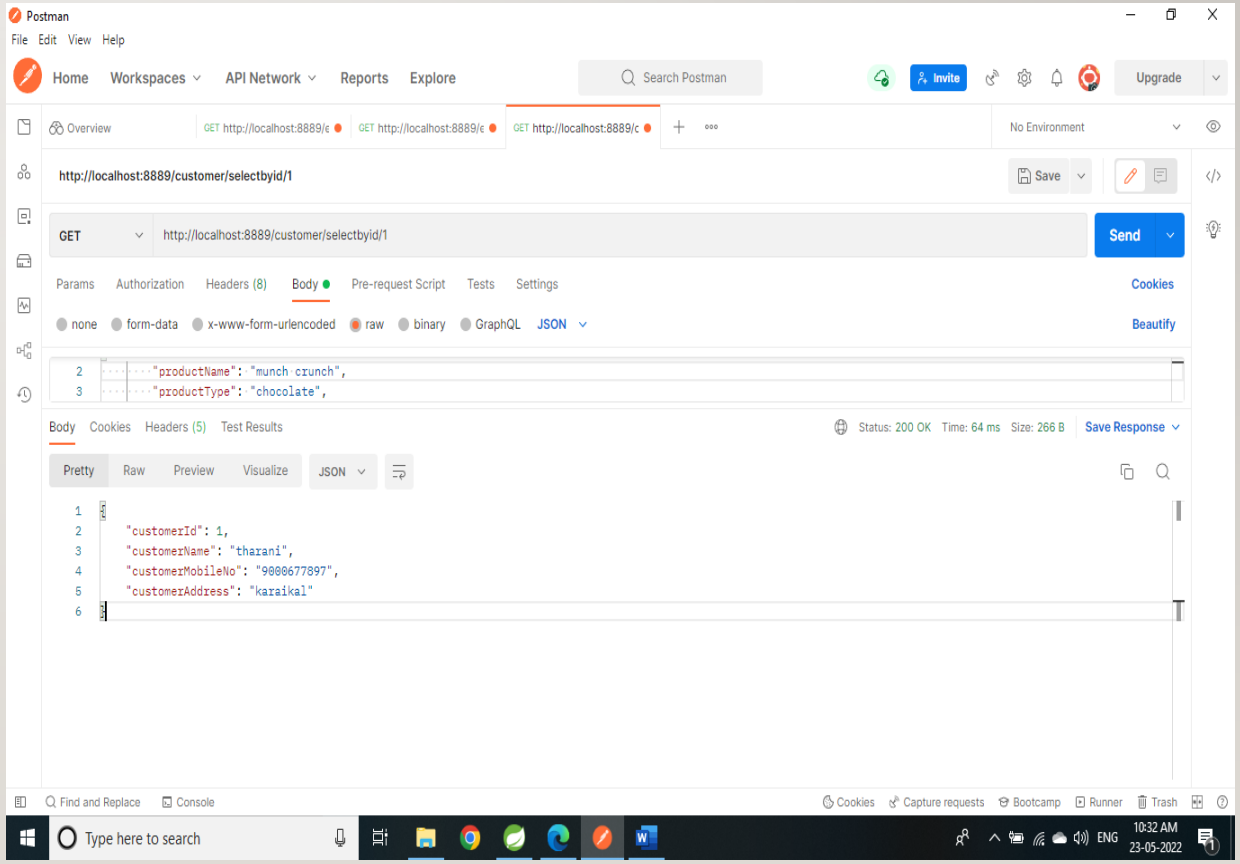


SCREENSHOT

Update: http://localhost:8889/product/updatebyid/9

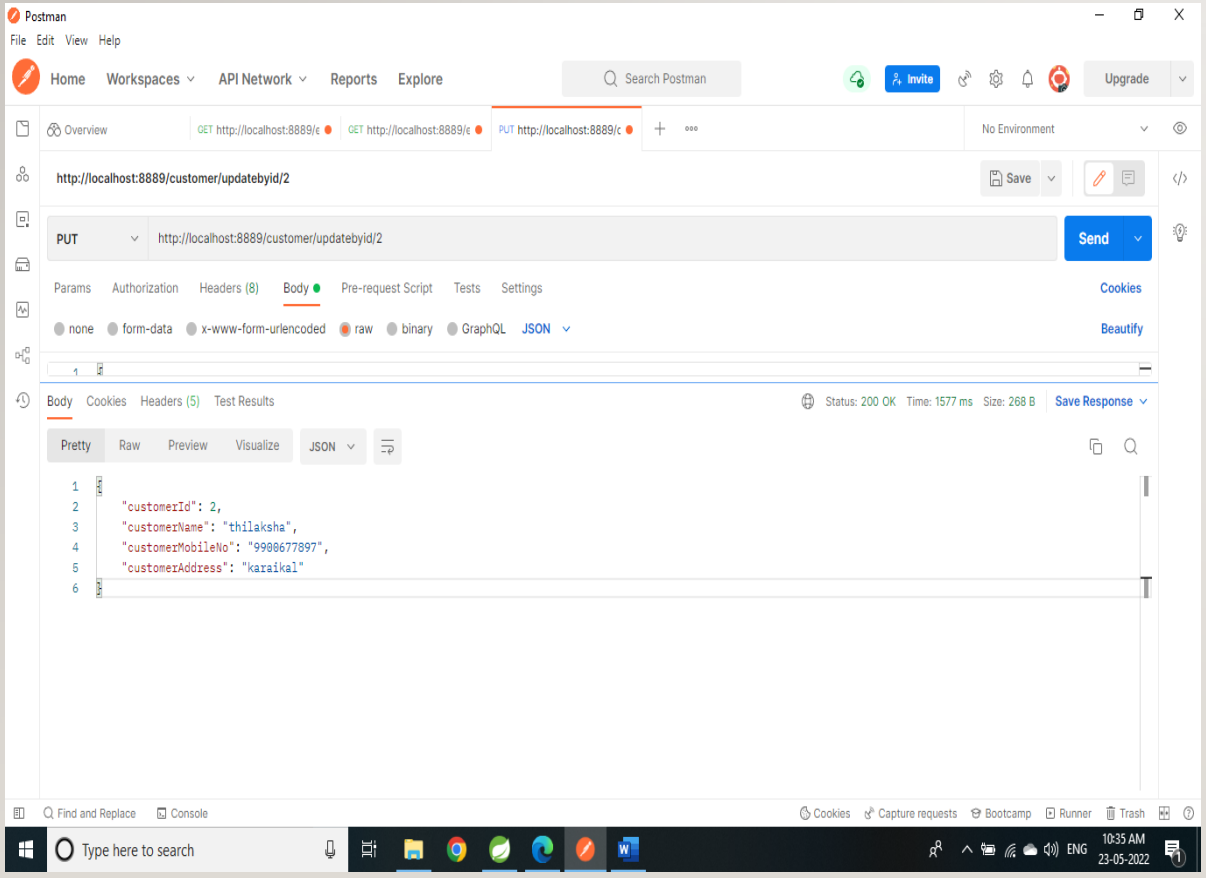


Customer screen shot:
Get: http://localhost:8889/customer/selectbyid/1

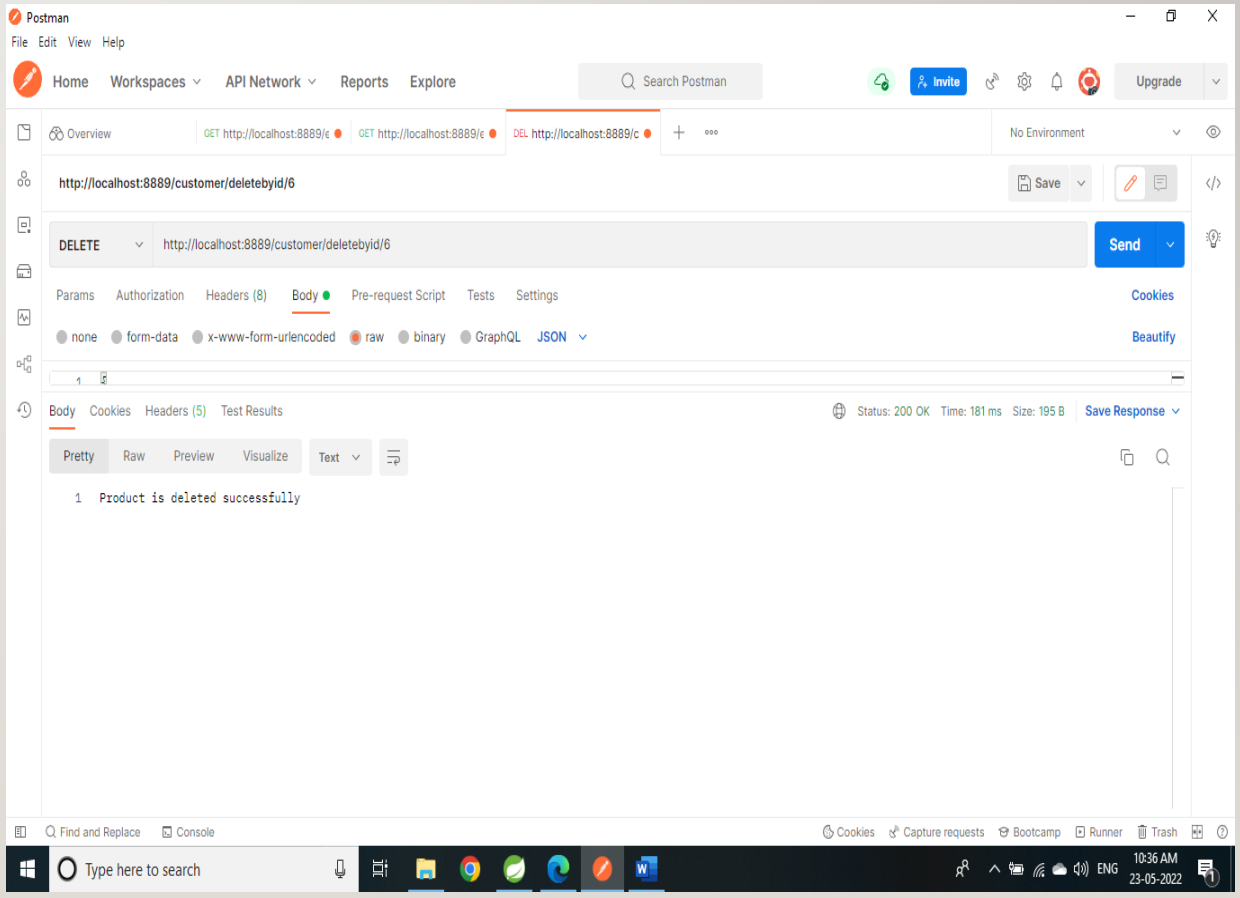


SCREENSHOT

Update by id: http://localhost:8889/customer/updatebyid/2

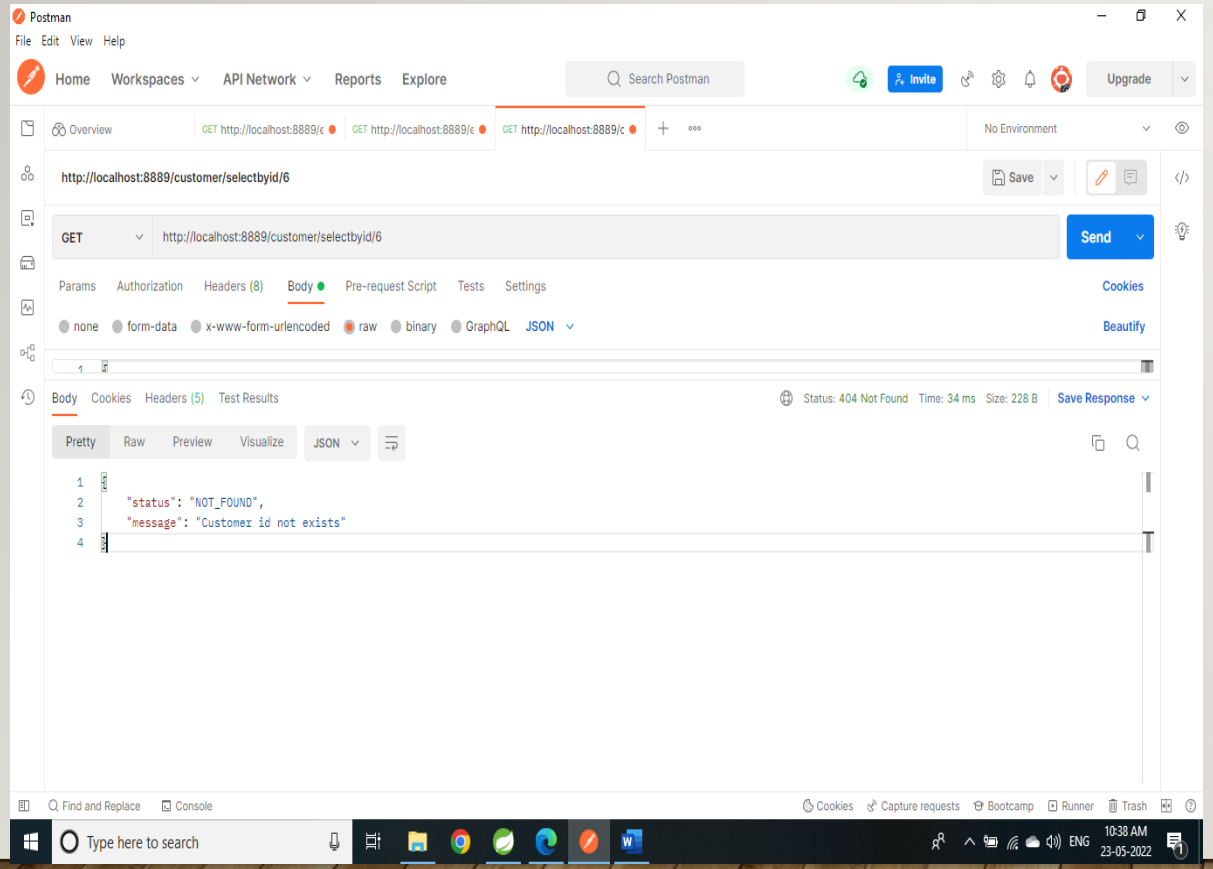


Delete by id: http://localhost:8889/customer/deletebyid/6

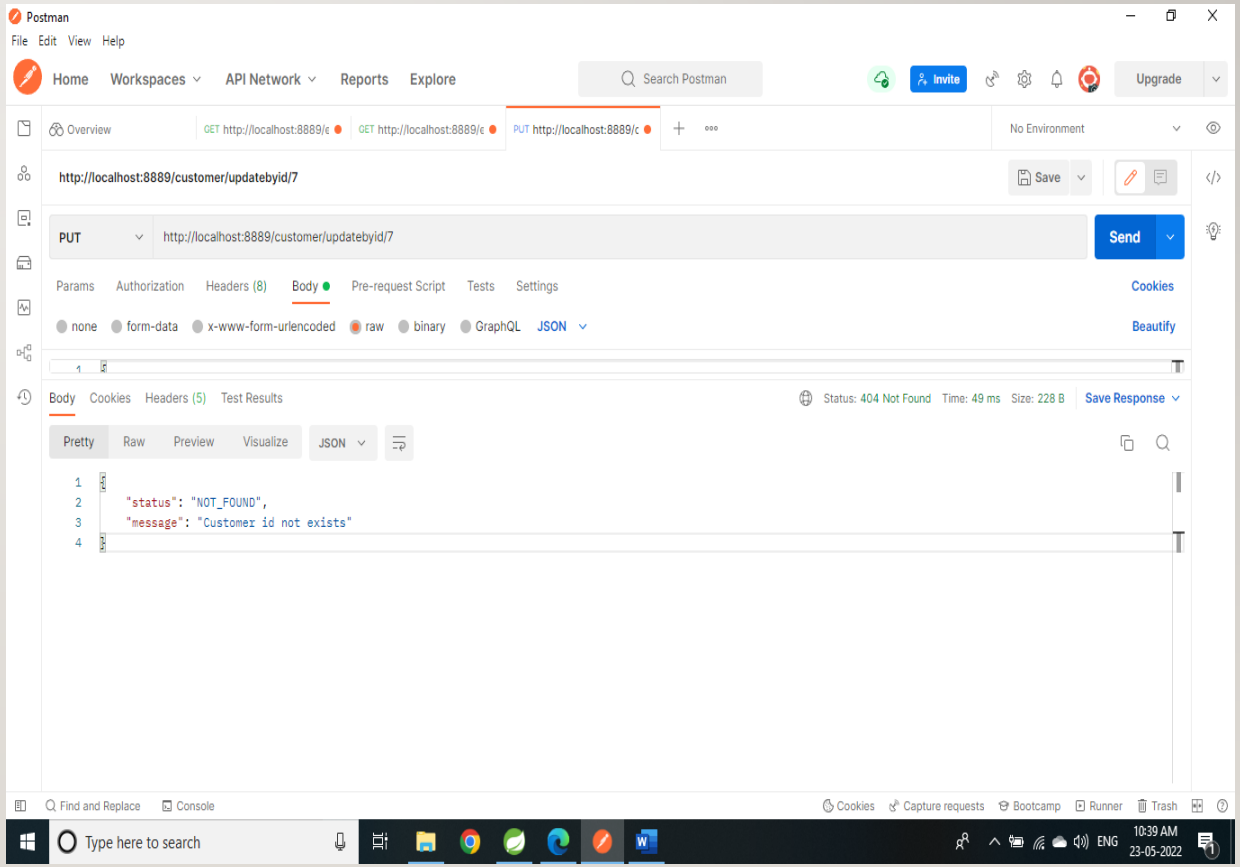


SCREENSHOT

Select by id: http://localhost:8889/customer/selectbyid/6

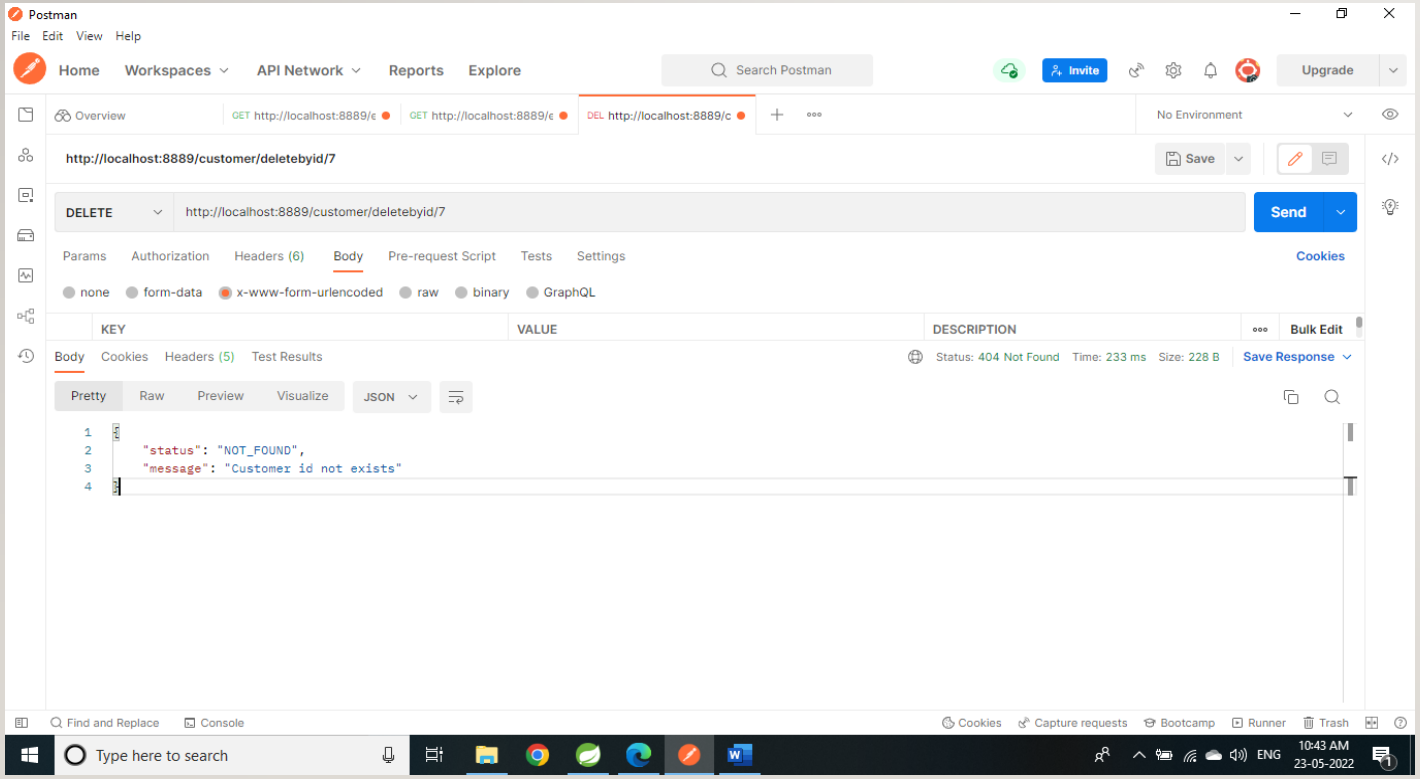


Update: http://localhost:8889/customer/updatebyid/7



SCREENSHOT

Delete: http://localhost:8889/customer/deletebyid/7



Input format:

```
{"productName": "jimjam",
      "productType": "biscuit",
      "productBrand": "sweety",
      "productQuantity": 20,
      "productPrice": 10.0,
   "customer": [ {
        "customerName": "tharani",
           "customerMobileNo": "9000677897",
           "customerAddress": "karaikal"
        }, {
            "customerName": "thilaksha",
           "customerMobileNo": "9900677897",
           "customerAddress": "karaikal"
        }
   ] }
```

SCREENSHOT

Shopkeeper_table

```
mysql> desc shopkeeper_table;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| product_id      | int         | NO   | PRI | NULL    |       |
| currentstock    | int         | NO   |     | NULL    |       |
| product_brand   | varchar(255)| NO   |     | NULL    |       |
| product_name    | varchar(255)| YES  |     | NULL    |       |
| product_price   | double      | NO   |     | NULL    |       |
| product_quantity| int         | NO   |     | NULL    |       |
| product_type    | varchar(255)| YES  |     | NULL    |       |
| totalamount     | double      | NO   |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
8 rows in set (0.20 sec)
```

Customer_table

MySQL 8.0 Command Line Client

```
7 rows in set (4.98 sec)

mysql> desc customer_table;
+------------------+-------------+------+-----+---------+-------+
| Field            | Type        | Null | Key | Default | Extra |
+------------------+-------------+------+-----+---------+-------+
| customer_id      | int         | NO   | PRI | NULL    |       |
| customer_address | varchar(255)| NO   |     | NULL    |       |
| customer_mobile_no| varchar(13)| YES  |     | NULL    |       |
| customer_name    | varchar(255)| YES  |     | NULL    |       |
| product_id       | int         | YES  | MUL | NULL    |       |
+------------------+-------------+------+-----+---------+-------+
5 rows in set (1.42 sec)
```

CONCLUSION

In Inventory management system we implement using spring boot to build the program in very easy way and using postman to run the program and database to store all the information of the product and customer details. Fetching details are very easy on this system.

THANK YOU