

## Kitchen story

This document contains sections for:

- [Description of the project](#)
- [Core concepts used in project](#)
- [Flow of the Application.](#)
- [Demonstrating the product capabilities, appearance, and user interactions.](#)

### **Description of the project**

Kitchen Story is an e-commerce portal that lets people shop basic food items on their website. The website needs to have the following features:

- A search form in the home page to allow entry of the food items to be purchased by the customer.
- Based on item details entered, it will show available food items with price.
- Once a person selects an item to purchase, they will be redirected to the list of available items. In the next page, they are shown the complete breakout of the order and details of the payment to be made in the payment gateway. When payment is done, they are shown a confirmation page with details of the order.

### **Project Overview:**

- Kitchen Story is a company that manufactures and sells kitchen need products. They have a walk-in store, and now, they wish to launch their e-commerce portal KitchenStory.com. They decided to hire a Backend Developer to develop a prototype of the application.
- This project focuses on developing a REST API on the backend. It will be tested by using POSTMAN. There is **no frontend requirement** for this project. You should use dummy data for users, products, and order history

**The administrator can:**

**The admin should be able to change his password if he wants, he should be able to:**

- Manage the products in the store including categorizing them
- Browse the list of users who have signed up and be able to search users
- See purchase reports filtered by date and category

## Core concepts used in the Project

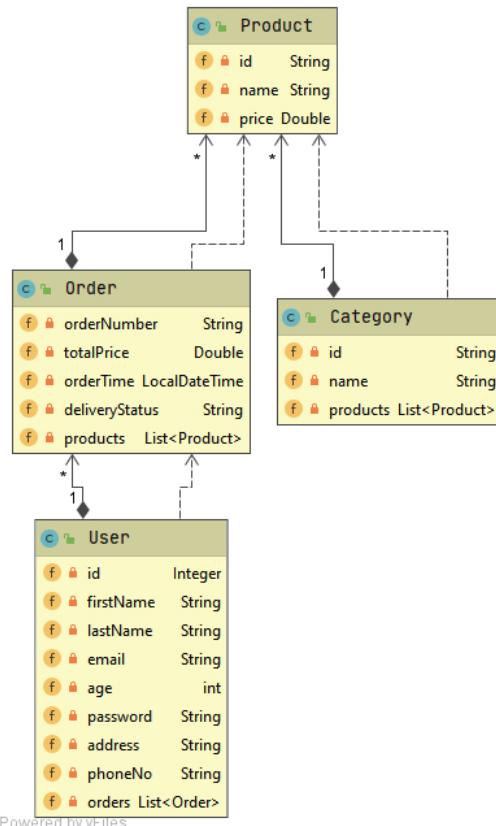
IDE: Spring Tool Suite

Framework: Spring Boot

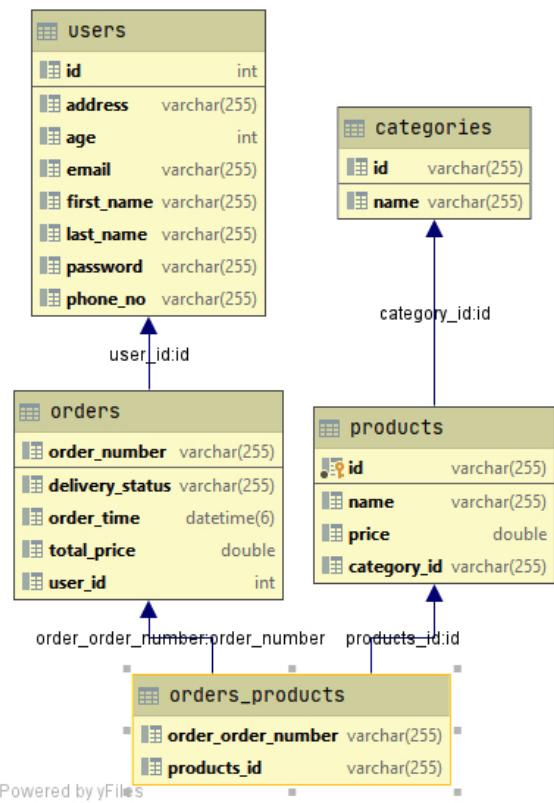
Programming Language: Java

Database: MySQL

## Class Diagram



## Entities diagram



## Database Connectivity

```

app.component.html index.html grocery.css *application.properties X
1 server.port=9094
2 #mysql properties
3 spring.datasource.url=jdbc:mysql://localhost:3306/kitchenStory
4 spring.datasource.username=root
5 spring.datasource.password=Sruthi@1990
6 spring.sql.init.mode=always
7 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
8 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
9
10 spring.jpa.defer-datasource-initialization=true
11 spring.jpa.show-sql=true
12 spring.jpa.hibernate.ddl-auto=update
13
14
15

```

**Database Connectivity:** Create the application.properties file to setup connectivity to the database

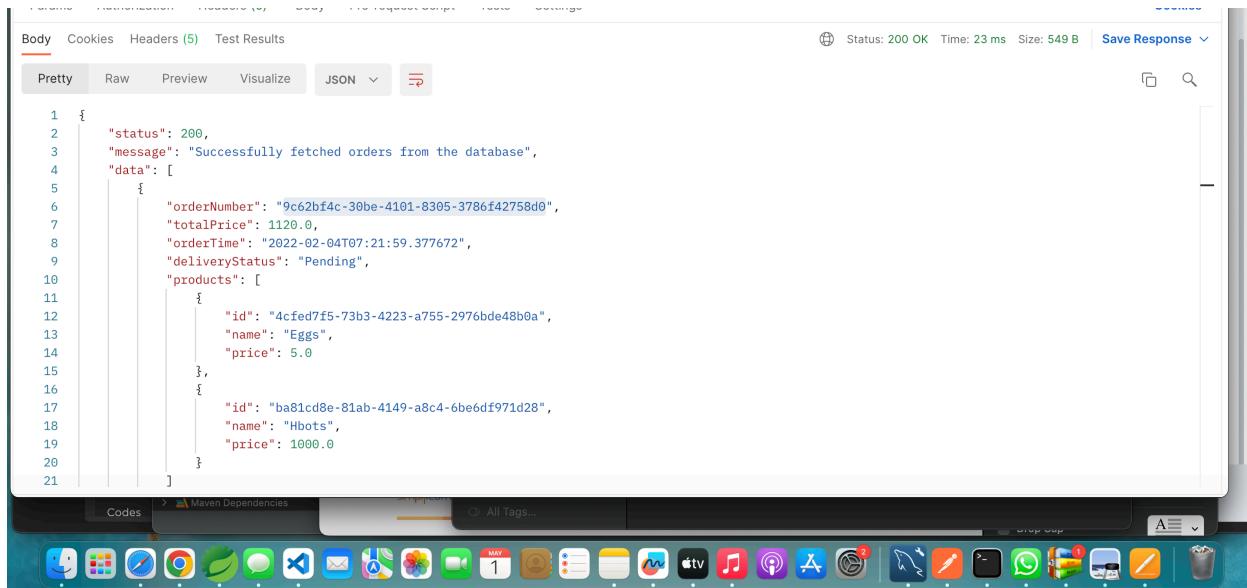
### Used Dependencies :

```
<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-thymeleaf</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.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>
```

## Some ScreenShots

### Order reading

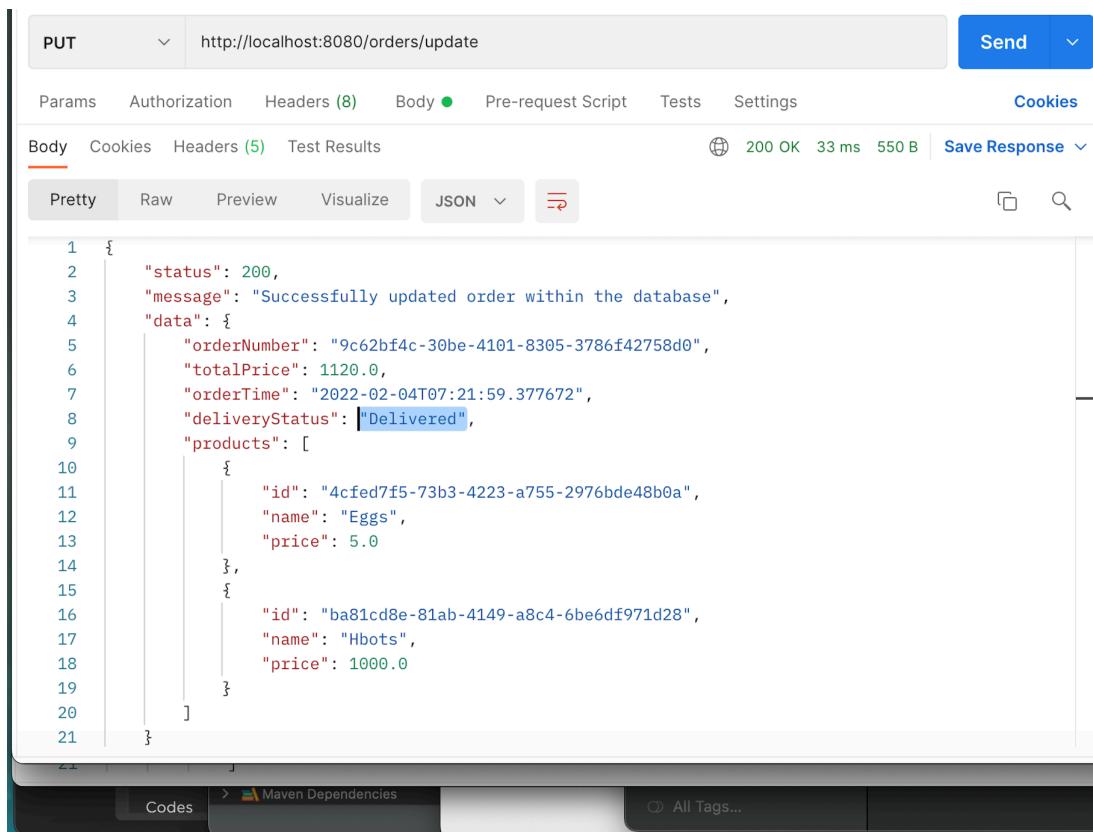


The screenshot shows the Postman interface with a successful response for reading orders. The response body is a JSON object containing the status, message, and data. The data array contains two objects, each representing an order with its number, total price, order time, delivery status, and products.

```

1 {
2     "status": 200,
3     "message": "Successfully fetched orders from the database",
4     "data": [
5         {
6             "orderNumber": "9c62bf4c-30be-4101-8305-3786f42758d0",
7             "totalPrice": 1120.0,
8             "orderTime": "2022-02-04T07:21:59.377672",
9             "deliveryStatus": "Pending",
10            "products": [
11                {
12                    "id": "4cfed7f5-73b3-4223-a755-2976bde48b0a",
13                    "name": "Eggs",
14                    "price": 5.0
15                },
16                {
17                    "id": "ba81cd8e-81ab-4149-a8c4-6be6df971d28",
18                    "name": "Hbots",
19                    "price": 1000.0
20                }
21            ]
22        }
23    ]
24 }
  
```

### Order updating

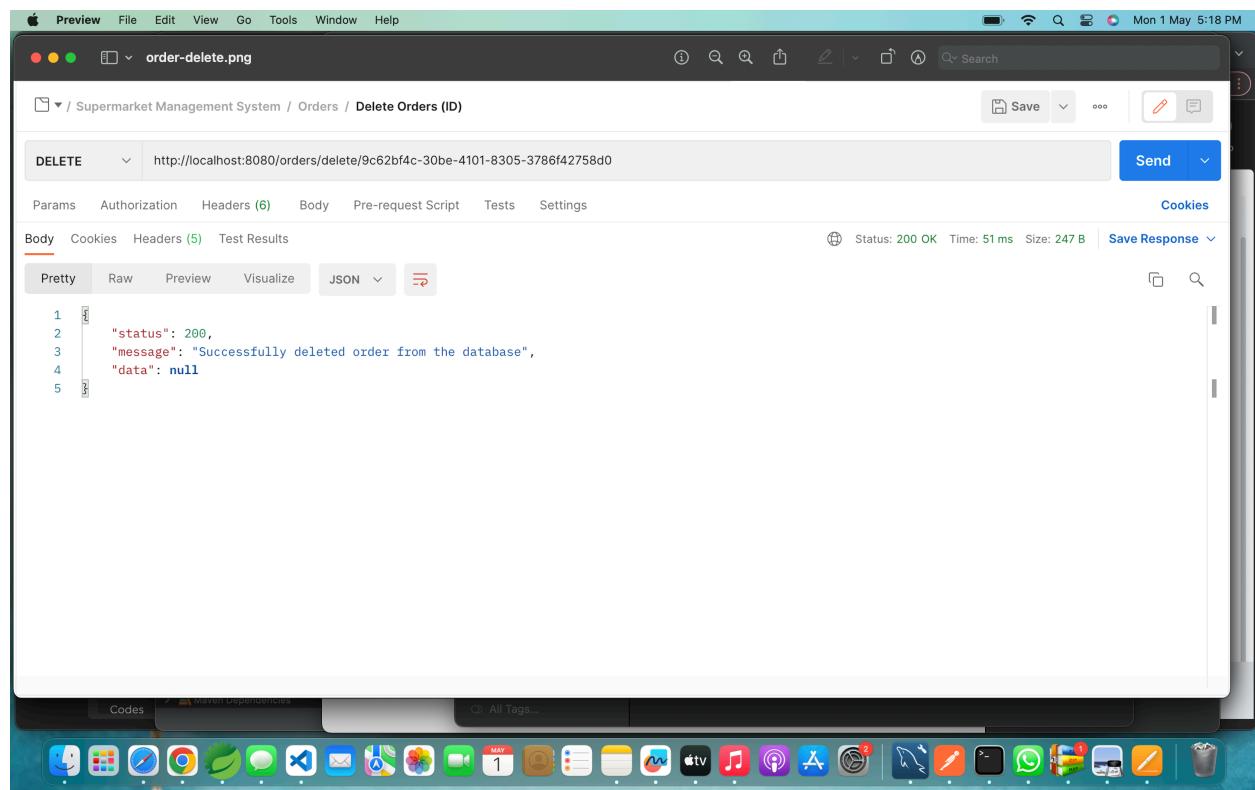


The screenshot shows the Postman interface with a successful PUT request for updating an order. The response body is a JSON object indicating the status, message, and updated data. The delivery status has been changed to 'Delivered'.

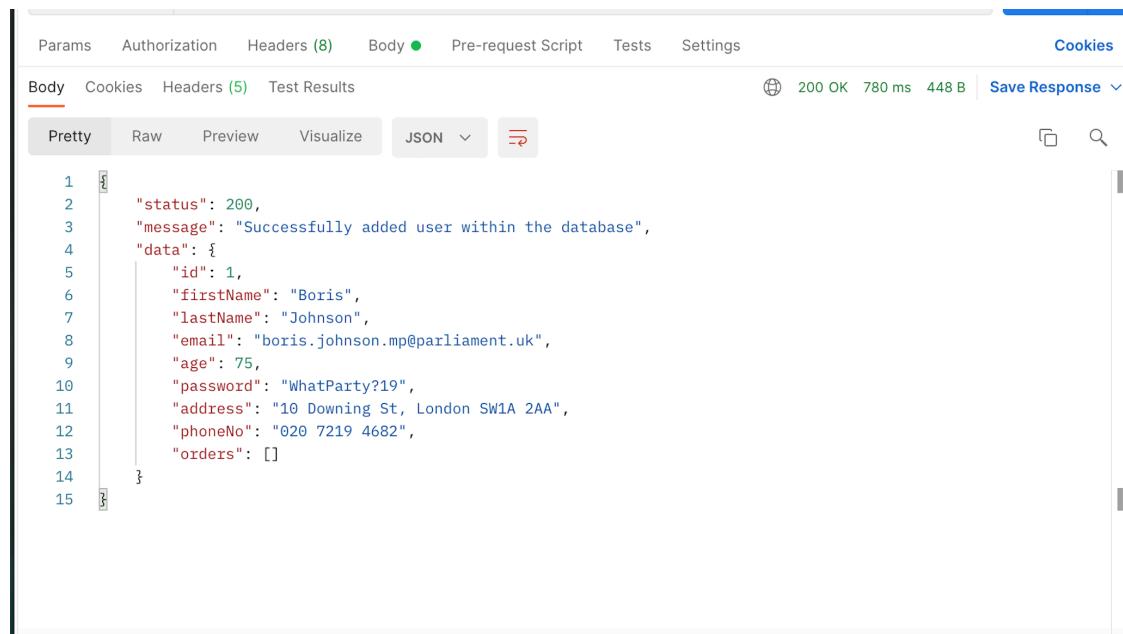
```

1 {
2     "status": 200,
3     "message": "Successfully updated order within the database",
4     "data": {
5         "orderNumber": "9c62bf4c-30be-4101-8305-3786f42758d0",
6         "totalPrice": 1120.0,
7         "orderTime": "2022-02-04T07:21:59.377672",
8         "deliveryStatus": "Delivered",
9         "products": [
10             {
11                 "id": "4cfed7f5-73b3-4223-a755-2976bde48b0a",
12                 "name": "Eggs",
13                 "price": 5.0
14             },
15             {
16                 "id": "ba81cd8e-81ab-4149-a8c4-6be6df971d28",
17                 "name": "Hbots",
18                 "price": 1000.0
19             }
20         ]
21     }
22 }
  
```

## Order deleting



## creating user



## creating user

The screenshot shows a POST request in Postman. The request body is a JSON object:

```
1 ...{  
2     "name": "Eggs",  
3     "price": 5  
4 }
```

The response status is 200 OK, and the response body is:

```
1 {  
2     "status": 200,  
3     "message": "Successfully added product within the database",  
4     "data": {  
5         "id": "aadf4d61-d2b3-4d5b-9eda-aaa1bf9a57c8",  
6         "name": "Eggs",  
7         "price": 5.0  
8     }  
9 }
```

**All code for this project can be found at the following link:**

**[https://github.com/sruthisree001/kitchen\\_Story.git](https://github.com/sruthisree001/kitchen_Story.git)**