

ECOMMERCE CAPSTONE PROJECT

BASIC FLOW/ SCENARIOS:

1. Customer Management

- Add new customers, Update customer information, Retrieve customer details.

2. Product Management:

- Users can view a list of available products, add, and delete products

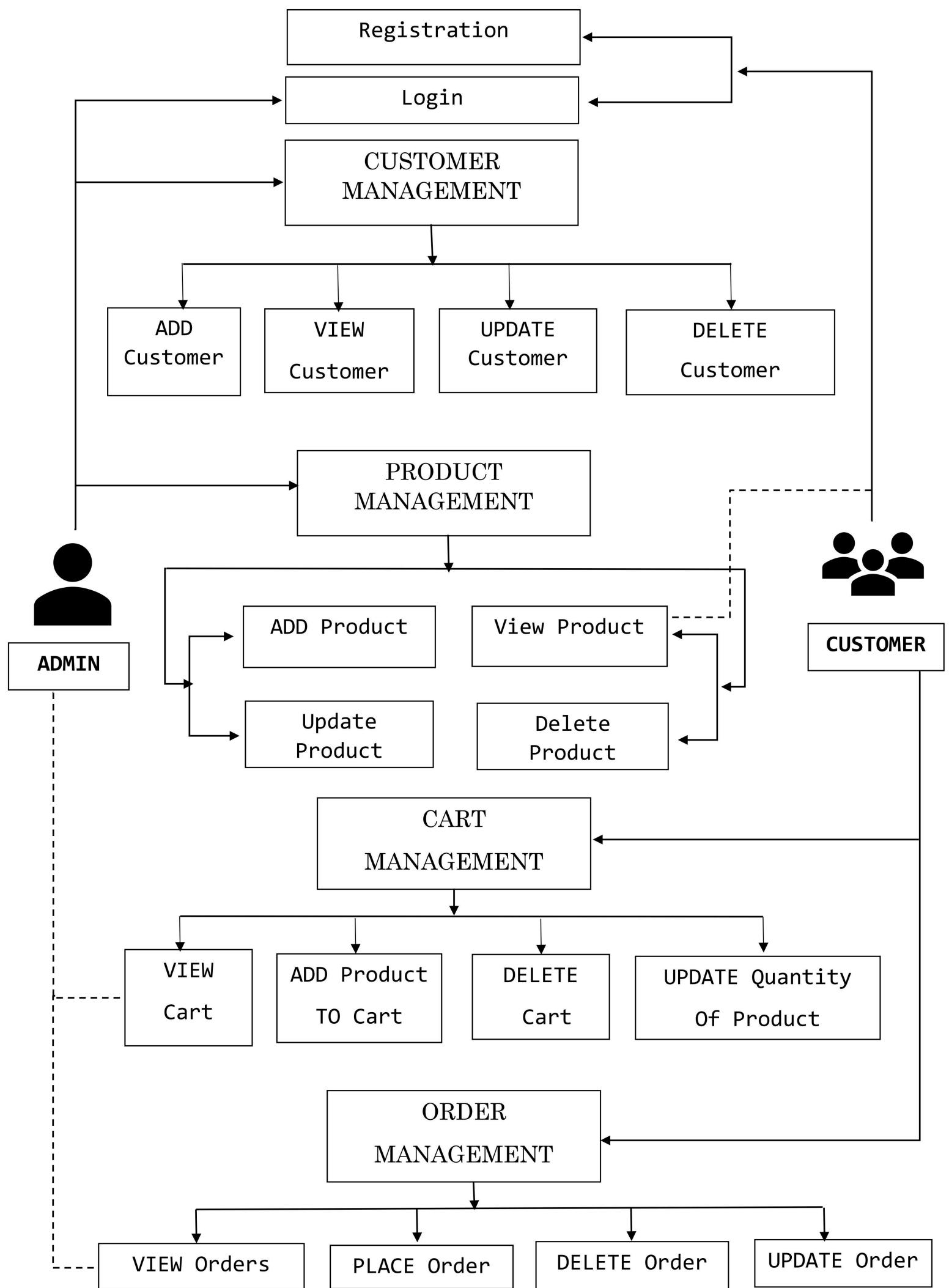
3. Cart Management:

- Users can add and remove products to their shopping cart

4. Order Management:

- Users can place orders, which include product details, quantities, and shipping information.
- The order total is calculated based on the cart contents.

- ❖ All the requirements as per the basic flow/scenario provided has been met.
- ❖ Below is the use case diagram for the E-commerce Capstone project



Console Output

Both the eureka server application and the ecommerce capstone project application have been successfully executed without any errors.

The screenshot shows the Spring Tool Suite interface with the following details:

- Project Explorer**: Shows the project structure with several sub-projects like "drugs-springboot-project", "ecommerce-capstone-project", and "ecommerce-eureka-server".
- EcommerceEurekaServerApplication.java**: The main application class defined as follows:

```
1 package com.wipro;
2
3 import org.springframework.boot.SpringApplication;
4
5 @SpringBootApplication
6 @EnableEurekaServer
7 public class EcommerceEurekaServerApplication {
8
9 }
```

- Console**: Displays the application logs for "ecommerce-eureka-server - EcommerceEurekaServerApplication [Spring Boot App]". The logs show the application starting up, initializing Eureka, and registering with the Eureka server.

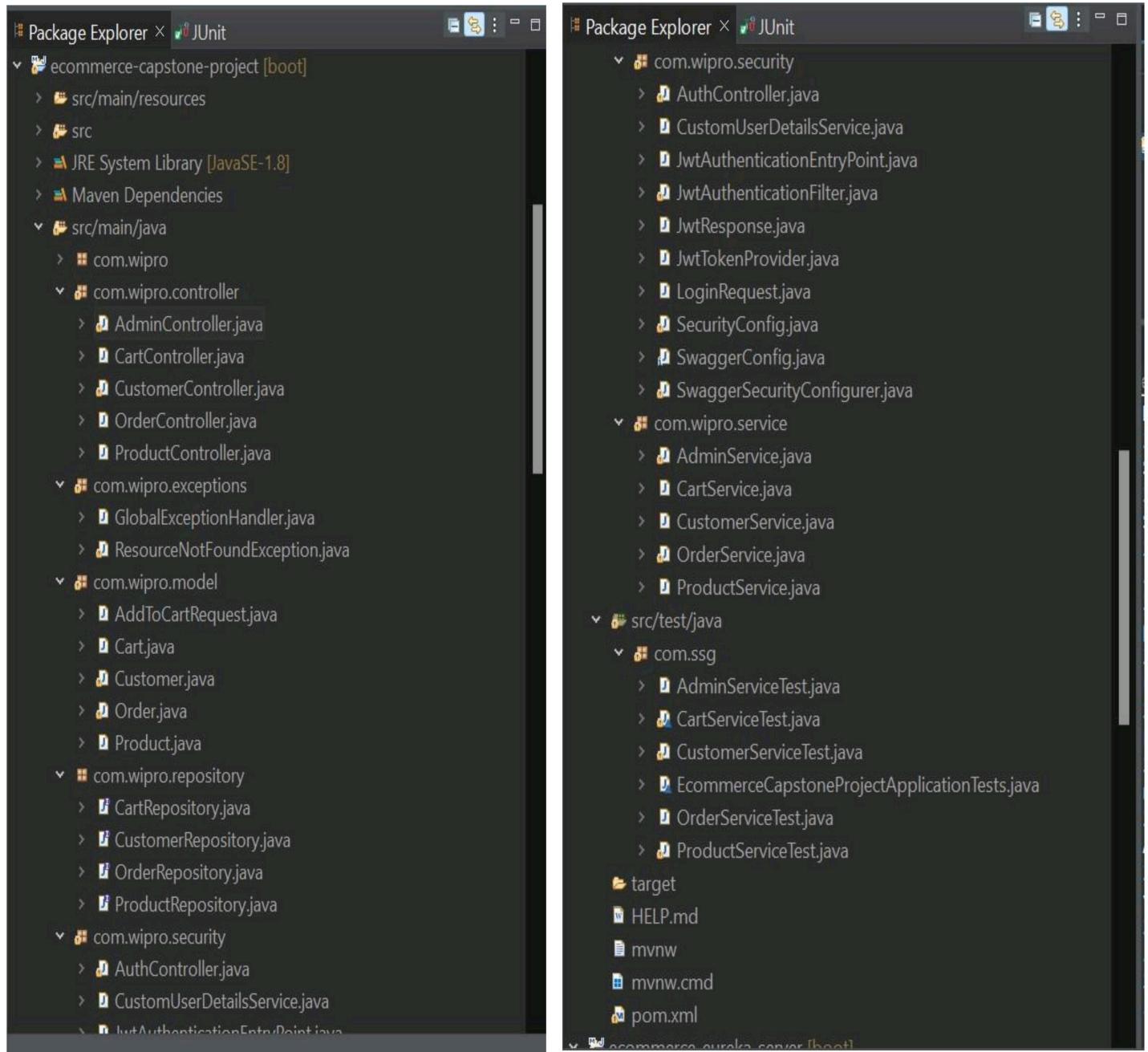
```
false
2024-09-02 13:28:45.478 INFO 14652 --- [           Thread-9] o.s.c.n.e.server.EurekaServerBootstrap : Initialized
server context
2024-09-02 13:28:45.478 INFO 14652 --- [           Thread-9] c.n.e.r.PeerAwareInstanceRegistryImpl : Got 1 instances
from neighboring DS node
2024-09-02 13:28:45.478 INFO 14652 --- [           Thread-9] c.n.e.r.PeerAwareInstanceRegistryImpl : Renew threshold
is: 1
2024-09-02 13:28:45.478 INFO 14652 --- [           Thread-9] c.n.e.r.PeerAwareInstanceRegistryImpl : Changing status
to UP
2024-09-02 13:28:45.488 INFO 14652 --- [           Thread-9] e.s.EurekaServerInitializerConfiguration : Started Eureka
Server
2024-09-02 13:28:45.507 INFO 14652 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started
on port(s): 8761 (http) with context path ''
2024-09-02 13:28:45.507 INFO 14652 --- [           main] s.c.n.e.s.EurekaAutoServiceRegistration : Updating port
to 8761
2024-09-02 13:28:45.523 INFO 14652 --- [           main] c.w.EcommerceEurekaServerApplication : Started
EcommerceEurekaServerApplication in 5.002 seconds (JVM running for 6.242)
2024-09-02 13:28:46.318 INFO 14652 --- [on(4)-127.0.0.1] o.a.c.c.C[Tomcat].[localhost].[/] : Initializing
Spring DispatcherServlet 'dispatcherServlet'
2024-09-02 13:28:46.318 INFO 14652 --- [on(4)-127.0.0.1] o.s.web.servlet.DispatcherServlet : Initializing
Servlet 'dispatcherServlet'
2024-09-02 13:28:46.320 INFO 14652 --- [on(4)-127.0.0.1] o.s.web.servlet.DispatcherServlet : Completed
initialization in 1 ms
```

The screenshot shows the Spring Tool Suite interface with the following details:

- Project Explorer:** Shows the project structure for "ecommerce-capstone-project".
- AdminController.java:** The current file being edited, containing Java code for a REST controller.
- CartController.java:** Another file in the project.
- Console:** A terminal window showing the application logs for "E-commerce Capstone Project".

```
ecommerce-capstone-project - E-commerce Capstone Project Application [Spring Boot App] C:\Users\sohan\Downloads\Spring-Tool-Suite-4-4.23.1.RELEASE-e4.32.0-win
2024-09-02 13:32:39.381 INFO 4000 --- [main] com.netflix.discovery.DiscoveryClient : Discovery Client initialized at timestamp 1725264159380 with initial instances count: 0
2024-09-02 13:32:39.384 INFO 4000 --- [main] o.s.c.n.e.s.EurekaServiceRegistry : Registering application E-COMMERCE-CAPSTONE-PROJECT with eureka with status UP
2024-09-02 13:32:39.385 INFO 4000 --- [main] com.netflix.discovery.DiscoveryClient : Saw local status change event StatusChangeEvent [timestamp=1725264159385, current=UP, previous=STARTING]
2024-09-02 13:32:39.387 INFO 4000 --- [noReplicator-0] com.netflix.discovery.DiscoveryClient : DiscoveryClient_E-COMMERCE-CAPSTONE-PROJECT/LaptopsG65:ecommerce-capstone-project: registering service...
2024-09-02 13:32:39.424 INFO 4000 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2024-09-02 13:32:39.425 INFO 4000 --- [main] s.c.n.e.s.EurekaAutoServiceRegistration : Updating port to 8080
2024-09-02 13:32:39.446 INFO 4000 --- [main] c.w.E-commerceCapstoneProjectApplication : Started E-commerce Capstone Project Application in 11.598 seconds (JVM running for 12.725)
2024-09-02 13:32:39.581 INFO 4000 --- [noReplicator-0] com.netflix.discovery.DiscoveryClient : DiscoveryClient_E-COMMERCE-CAPSTONE-PROJECT/LaptopsG65:ecommerce-capstone-project - registration status: 204
2024-09-02 13:32:39.992 INFO 4000 --- [on(4)-127.0.0.1] o.a.c.c.[Tomcat].[localhost].[] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2024-09-02 13:32:39.992 INFO 4000 --- [on(4)-127.0.0.1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2024-09-02 13:32:39.995 INFO 4000 --- [on(4)-127.0.0.1] o.s.web.servlet.DispatcherServlet : Completed initialization in 3 ms
```

Package Explorer Screenshots



Eureka Discovery Service

The Capstone project is successfully registered on eureka server

The screenshot shows the Spring Eureka web interface. At the top, there's a header with the Eureka logo and navigation links for 'HOME' and 'LAST 1000 SINCE STARTUP'. Below the header, the 'System Status' section displays various metrics: Environment (N/A), Data center (N/A), Current time (2024-09-02T14:42:10 +0530), Uptime (01:13), Lease expiration enabled (true), Renews threshold (3), and Renews (last min) (4). The 'DS Replicas' section shows a single instance at 'localhost'. The 'Instances currently registered with Eureka' table lists one application named 'ECOMMERCE-CAPSTONE-PROJECT' with 1 AMI and 1 availability zone, both marked as 'UP'.

Application	AMIs	Availability Zones	Status
ECOMMERCE-CAPSTONE-PROJECT	n/a (1)	(1)	UP (1) - LaptopSG05.ecommerce-capstone-project

Actuator

The actuator shows the info, health and env of the project successfully

The screenshot shows the Actuator UI with a JSON response for the '/info' endpoint. The response object contains an 'application' key with a nested 'author' key. The 'application' object has a 'description' field set to 'An Actuator for Ecommerce Capstone Project!!!'. The 'author' object has a 'name' field set to 'Sohan Gurav' and a 'bio' field set to 'Ecommerce Website'.

```
{  
  - application: {  
      description: "An Actuator for Ecommerce Capstone Project!!!",  
      - author: {  
          name: "Sohan Gurav",  
          bio: "Ecommerce Website"  
      }  
  }  
}
```

Outputs on Postman:

- Admin is able to login with the predefined credentials and generate jwt token successfully.

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** My Workspace, Collections, Environments, History.
- Request URL:** POST http://localhost:8080/api/auth/login
- Body (raw JSON):**

```
1 {
2   "email": "admin123@company.com",
3   "password": "Sohan@wipro"
4 }
```
- Response Headers:** 200 OK, 789 ms, 720 B
- Response Body (Pretty JSON):**

```
1 {
2   "accessToken": "eyJhbGciOiJIUzUxMiJ9.
3   eyJzdWlIoiJhZG1pbjEyM0Bjb21wYW55LnNbSIisImIhdCI6MTcyNTI2NDI3MCwiZXhwIjoxNzI1MjY3ODcwfQ.
4   XglLtxbIdA90jbRC39Kc0vy_h3u35My05KT3Bk1zHplPwqfQ51CvLyFX9mQe-L_jRae47LrXcMxGoBo_HRUQ",
5   "tokenType": "Bearer"
```

Admin can retrieve, add, update, and delete customer records.

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** My Workspace, Collections, Environments, History.
- Request URL:** GET http://localhost:8080/admin/customers?Authorization=Bearer eyJhbGciOiJIUzUxMiJ9eyJ...
- Response Headers:** 200 OK, 314 ms, 1.73 KB
- Response Body (Pretty JSON):**

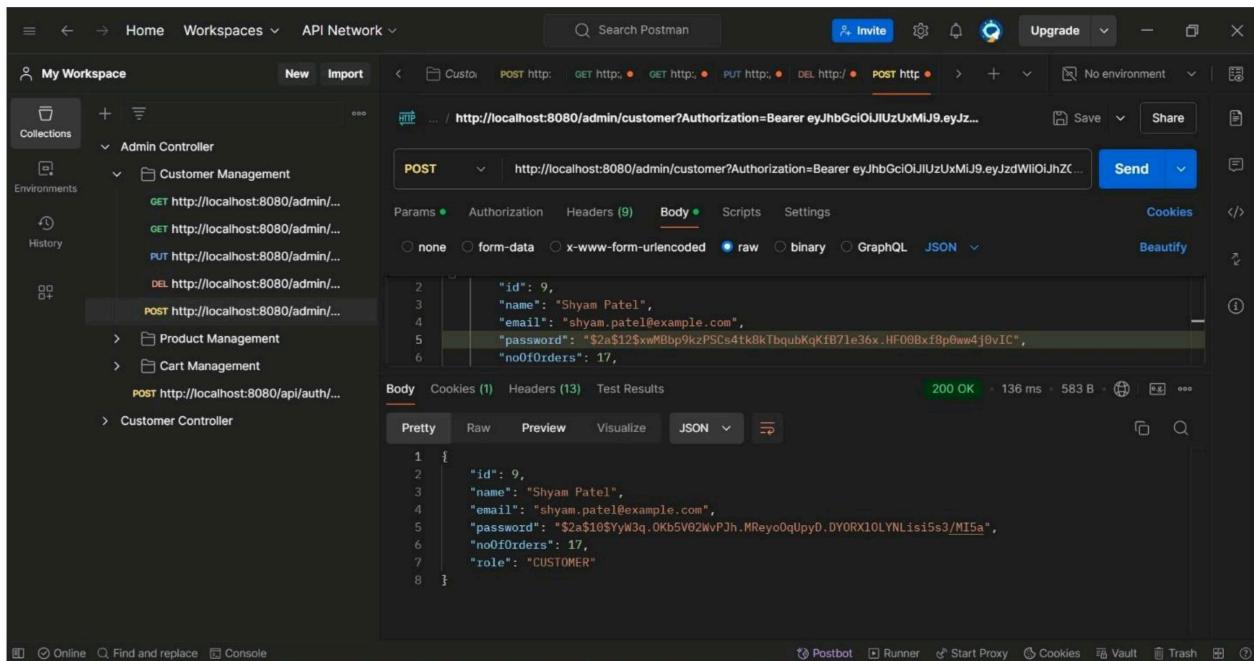
```
3 [
4   {
5     "id": 1,
6     "name": "Mihir Jain",
7     "email": "mihir.jain@xyz.com",
8     "password": "$2a$10$SCNo/0CFuWkUy0Kkvjs49.RIFSoC2sfbi60yeq762UW3EGfRN9Jm",
9     "noOfOrders": 45,
10    "role": "CUSTOMER"
11  },
12  {
13    "id": 2,
14    "name": "Sahil Patel",
15    "email": "sahil.patel@example.com",
16    "password": "$2a$10$JfI2swCPAxWa2yyEzrjEuJib6t/VD/IldqjbD8MN0dnkU9ZAS/ywi",
17    "noOfOrders": 65,
18    "role": "CUSTOMER"
19  }
20 ]
```

My Workspace

POST http://localhost:8080/admin/customer?Authorization=Bearer eyJhbGciOiJIUzUxMiJ9eyJ...
Body (raw) JSON

```
1 {  
2   "id": 9,  
3   "name": "Shyam Patel",  
4   "email": "shyam.patel@example.com",  
5   "password": "$2a$12$xwMBbp9kzPScs4tk8kBqubKqKFb7le36x.HF0Bxf8p0ww4j0vIC",  
6   "noOfOrders": 17,  
7   "role": "CUSTOMER"  
8 }
```

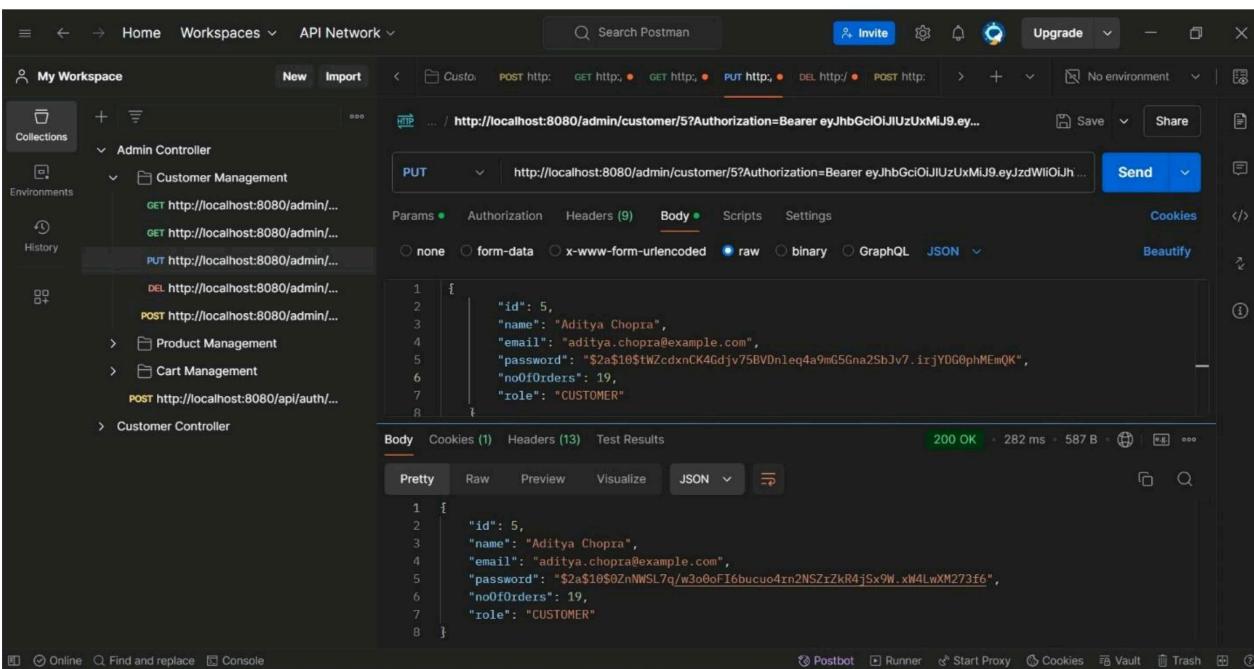
200 OK



PUT http://localhost:8080/admin/customer/5?Authorization=Bearer eyJhbGciOiJIUzUxMiJ9eyJ...
Body (raw) JSON

```
1 {  
2   "id": 5,  
3   "name": "Aditya Chopra",  
4   "email": "aditya.chopra@example.com",  
5   "password": "$2a$10$tWZcdxnCK4gdjv75BVdnLeq4a9mG5Gna2Sbjv7.iTjYDGphMEmQK",  
6   "noOfOrders": 19,  
7   "role": "CUSTOMER"  
8 }
```

200 OK



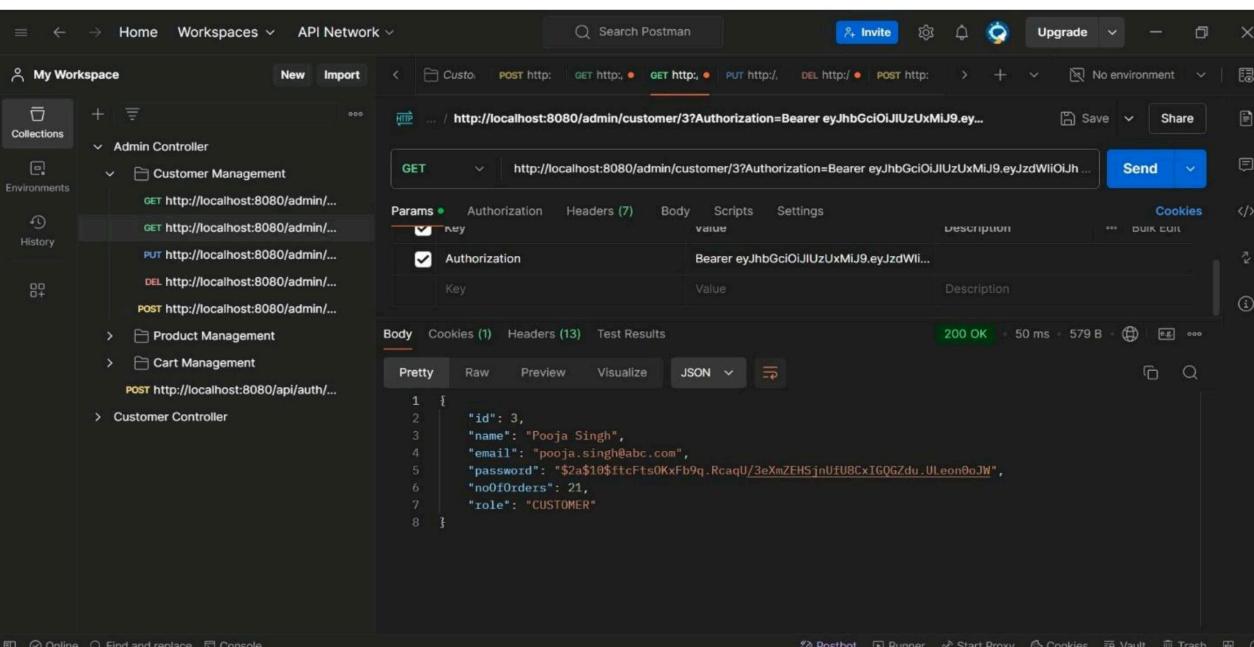
GET http://localhost:8080/admin/customer/3?Authorization=Bearer eyJhbGciOiJIUzUxMiJ9eyJ...
Headers (7)

Key	Value	Description
Authorization	Bearer eyJhbGciOiJIUzUxMiJ9eyJ... Key Value Description	BAKER eyJhbGciOiJIUzUxMiJ9eyJ...

Body (raw) JSON

```
1 {  
2   "id": 3,  
3   "name": "Pooja Singh",  
4   "email": "pooja.singh@abc.com",  
5   "password": "$2a$10$ftcFtsOKxFb9q.RcaQu/3eXmZEHSjnUfU8CxIGQGZdu.ULeon0oJW",  
6   "noOfOrders": 21,  
7   "role": "CUSTOMER"  
8 }
```

200 OK



Similarly, Admin can also retrieve, add, update, and delete a product

Add Product

The screenshot shows the Postman interface with a dark theme. On the left, the 'My Workspace' sidebar lists collections like 'Admin Controller', 'Customer Management', and 'Product Management'. In the main area, a POST request is being made to `http://localhost:8080/products/add?Authorization=Bearer eyJhbGciOiJIUzUxMiJ9eyJzdWI...` . The 'Body' tab is selected, showing a JSON payload:

```
1 {  
2   "id": 57,  
3   "name": "Nike Sports Shoes",  
4   "category": "Footwear",  
5   "price": 3499.0  
6 }
```

The response status is 200 OK, with a response time of 74 ms and a size of 485 B. The response body is identical to the request body.

Retrieve Product

The screenshot shows the Postman interface with a dark theme. The 'My Workspace' sidebar lists various collections. A GET request is being made to `http://localhost:8080/products/all?Authorization=Bearer eyJhbGciOiJIUzUxMiJ9eyJzdWI...` . The 'Body' tab is selected, showing a JSON response:

```
1 [  
2   {  
3     "id": 1,  
4     "name": "One Plus 12R",  
5     "category": "Electronics",  
6     "price": 39999.0  
7   },  
8   {  
9     "id": 2,  
10    "name": "Hp Pavilion 15 Laptop",  
11    "category": "Electronics",  
12    "price": 58990.0  
13  },  
14  {  
15    "id": 3,  
16    "name": "Samsung Galaxy Tab S9 FE",  
17    "category": "Electronics",  
18    "price": 34499.0  
19  }.
```

The response status is 200 OK, with a response time of 61 ms and a size of 2.25 KB. The response body is identical to the request body.

Update Product

The screenshot shows the Postman application interface. In the left sidebar, under 'My Workspace' - 'Admin Controller' - 'Product Management', there is a 'PUT http://localhost:8080/products/update/product/55?Authorization=Bearer eyJhbGciOiJIUzUxMiJ9eyJ...' entry. The main panel displays a 'PUT' request to the same URL. The 'Body' tab is selected, showing a JSON payload:

```
1 {  
2   "id": 57,  
3   "name": "Nike Sports Shoes",  
4   "category": "Footwear",  
5   "price": 2499.0  
6 }
```

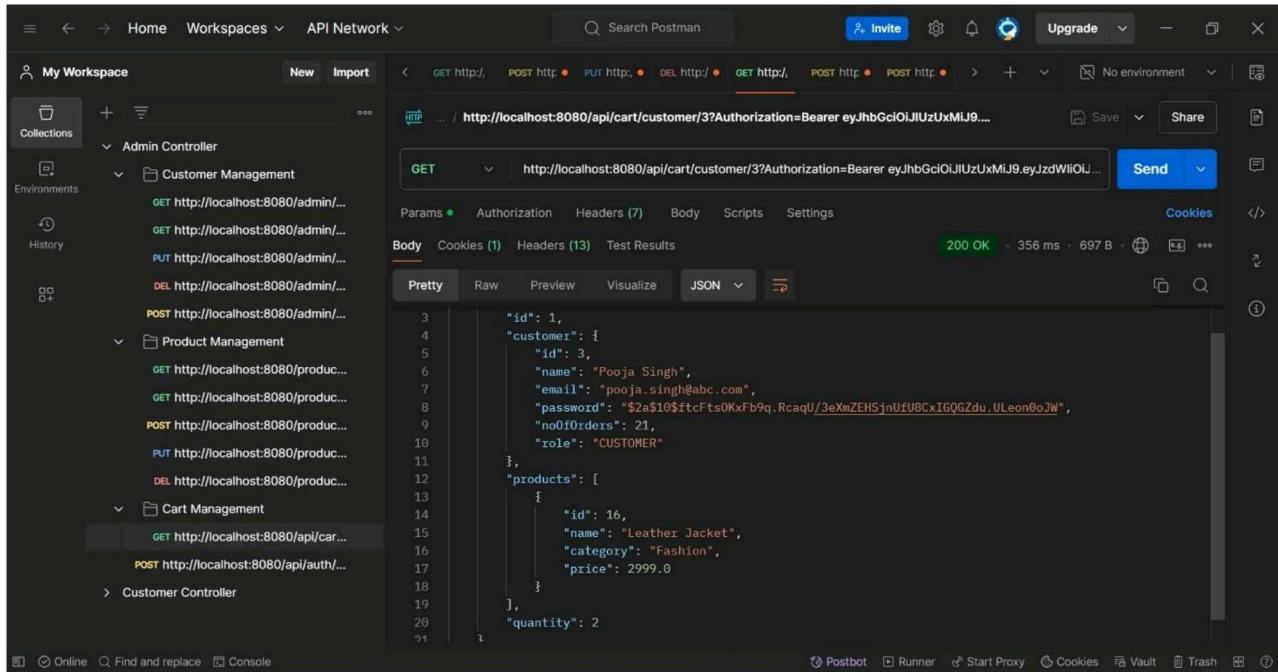
The response section shows a 200 OK status with 26 ms and 485 B. Below the body, there is another identical JSON object.

Delete Product

This screenshot is identical to the one above, showing the Postman interface with a 'PUT' request to update a product. The JSON payload and the resulting 200 OK response are identical to the previous update operation.

View Cart

Admin can also view cart details in which the cart id, customer details, product details, quantity is displayed.



```
3
4     "id": 1,
5     "customer": {
6         "id": 3,
7         "name": "Pooja Singh",
8         "email": "pooja.singh@abc.com",
9         "password": "$2a$10$ftcfTs0KxFb9q.RcaQu/3eXmZEHSjnUfU8CxIGQGZdu.ULeon@oJW",
10        "noOfOrders": 21,
11        "role": "CUSTOMER"
12    },
13    "products": [
14        {
15            "id": 16,
16            "name": "Leather Jacket",
17            "category": "Fashion",
18            "price": 2999.0
19        }
20    ],
21    "quantity": 2
22 }
```

Customer Registration/Login

- Customer can register themselves as well as login and generate the jwt token successfully

The screenshot shows the Postman interface with a dark theme. On the left, the 'My Workspace' sidebar lists collections: 'Admin Controller' and 'Customer Controller'. Under 'Customer Controller', there are three folders: 'Product Details', 'Add To Cart', and 'Order Products'. The 'Product Details' folder contains four items: GET, GET, POST, and PUT requests. The 'Add To Cart' folder contains three items: POST, PUT, and DEL requests. The 'Order Products' folder contains two items: POST and POST requests. The main workspace shows a POST request to 'http://localhost:8080/api/auth/register?name=Rajesh Milind&email=rajesh.milind@abc.com&password=password678rajesh'. The 'Params' tab shows four parameters: 'name' (Rajesh Milind), 'email' (rajesh.milind@abc.com), 'password' (password678rajesh), and 'key'. The 'Body' tab displays the JSON response:

```
1 {
2   "id": 12,
3   "name": "Rajesh Milind",
4   "email": "rajesh.milind@abc.com",
5   "password": "$2a$10$1819eqr0k1Nx26CQAvRT4u13NDT1FiZeHKEP5GPT.tKGpyPB.oB0",
6   "noOfOrders": 0,
7   "role": "CUSTOMER"
8 }
```

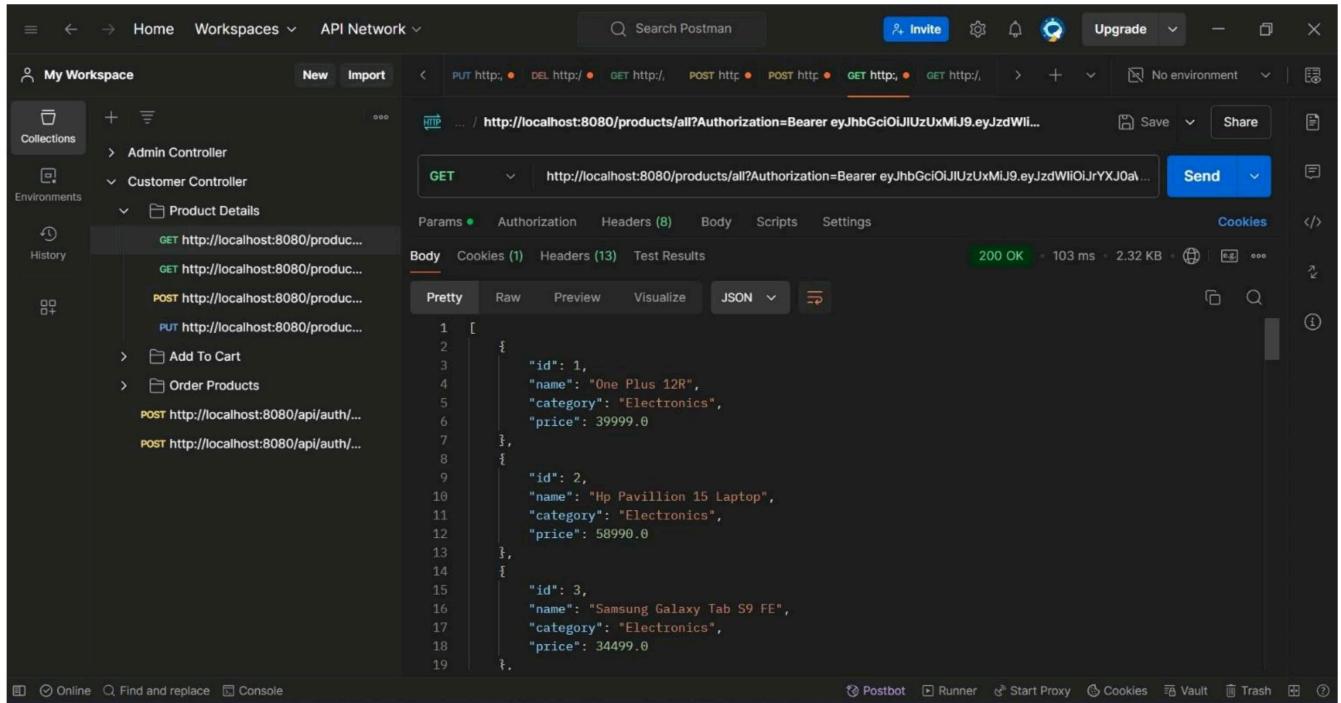
The screenshot shows the Postman interface with a dark theme. The 'My Workspace' sidebar is identical to the previous screenshot. The main workspace shows a POST request to 'http://localhost:8080/api/auth/login'. The 'Body' tab is selected and shows the raw JSON body:

```
1 {
2   "email": "sandeep.verma@abc.com",
3   "password": "password678sandeep"
4 }
```

The response status is 200 OK with a response time of 330 ms and a body size of 646 B. The response JSON is:

```
1 {
2   "accessToken": "eyJhbGciOiJIUzIwMjQ9.eyJzdWIiOiJzY5k2WwlzLcm1hQGfIYy5jb20iLCJpYXQiOjE3MjUyNjYzMzIwMjI2OTg3M30.aKugvc56Kib2qzj19dAuzNih33wm93zfix_NB1hyKLwYKh4XhxHK3DgBh0jVlboLuUi1232t1havbRUJZ5H6Vg",
3   "tokenType": "Bearer"
4 }
```

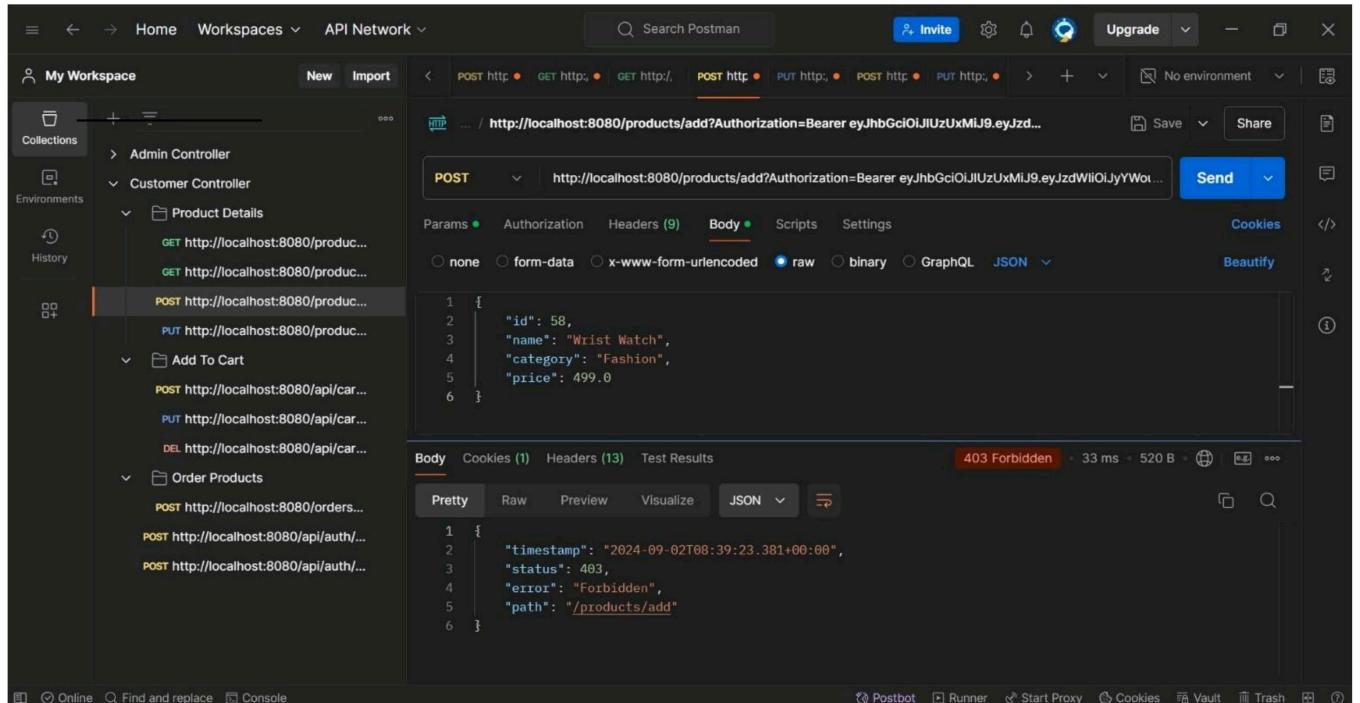
Customers can view all the products.



The screenshot shows the Postman interface with a dark theme. On the left, the 'My Workspace' sidebar lists collections like 'Admin Controller' and 'Customer Controller'. Under 'Customer Controller', there's a 'Product Details' folder containing several requests: 'GET http://localhost:8080/products/all...', 'GET http://localhost:8080/products...', 'POST http://localhost:8080/products...', 'PUT http://localhost:8080/products...', 'Add To Cart', 'Order Products', 'POST http://localhost:8080/api/auth...', and 'POST http://localhost:8080/api/auth/...'. The main panel shows a successful GET request to `http://localhost:8080/products/all?Authorization=Bearer eyJhbGciOiJIUzUxMiJ9eyJzdWl...`. The response body is a JSON array of three products:

```
1 [  
2 {  
3     "id": 1,  
4     "name": "One Plus 12R",  
5     "category": "Electronics",  
6     "price": 39999.0  
7 },  
8 {  
9     "id": 2,  
10    "name": "Hp Pavilion 15 Laptop",  
11    "category": "Electronics",  
12    "price": 58990.0  
13 },  
14 {  
15     "id": 3,  
16     "name": "Samsung Galaxy Tab S9 FE",  
17     "category": "Electronics",  
18     "price": 34499.0  
19 }.
```

Customers cannot add, update or delete product. These operations can be only done by the admin. So, when customer tries to perform this operation, they get a “403 forbidden error”



The screenshot shows the Postman interface with a dark theme. The 'My Workspace' sidebar is similar to the previous one. The main panel shows a failed POST request to `http://localhost:8080/products/add?Authorization=Bearer eyJhbGciOiJIUzUxMiJ9eyJzdWl...`. The request body is a JSON object:

```
1 {  
2     "id": 58,  
3     "name": "Wrist Watch",  
4     "category": "Fashion",  
5     "price": 499.0  
6 }
```

The response status is '403 Forbidden' with a timestamp and error message in the body:

```
1 {  
2     "timestamp": "2024-09-02T08:39:23.381+00:00",  
3     "status": 403,  
4     "error": "Forbidden",  
5     "path": "/products/add"  
6 }
```

The screenshot shows the Postman interface with a collection named 'Customer Controller'. A PUT request is selected, targeting the URL `http://localhost:8080/products/update/product/55?Authorization=Bearer eyJhbGciOiJIUz...` . The request body is set to raw JSON:

```
1 {
2   "id": 55,
3   "name": "Vegan Leather Jacket",
4   "category": "Fashion",
5   "price": 6499.0
6 }
```

The response status is 403 Forbidden, with the following JSON data:

```
1 {
2   "timestamp": "2024-09-02T08:39:54.848+09:00",
3   "status": 403,
4   "error": "Forbidden",
5   "path": "/products/update/product/55"
6 }
```

Add product to cart:

Customers can add product to cart, and when they do they can see the cart id, customer details, and product details along with the quantity.

The screenshot shows the Postman interface with a collection named 'Customer Controller'. A POST request is selected, targeting the URL `http://localhost:8080/api/cart/add-to-cart?Authorization=Bearer eyJhbGciOiJIUz...` . The request body is set to raw JSON:

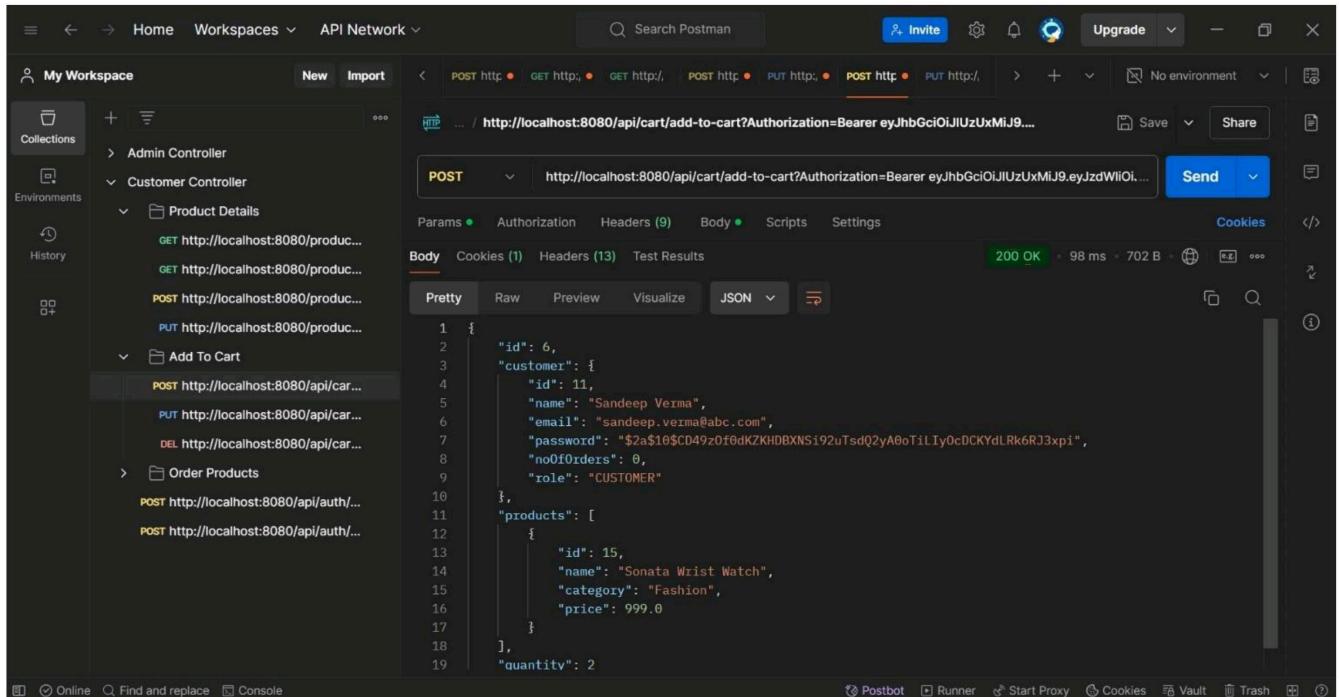
```
1 {
2   "id": 6,
3   "customer": {
4     "id": 11,
5     "name": "Sandeep Verma",
6     "email": "sandeep.verma@abc.com",
7     "password": "$2a$10$CD49z0f0dKZKHDXNS192uTsdQ2yA0oTlTy0cDCKYdLrk6RJ3xpi",
8     "noOfOrders": 0,
9     "role": "CUSTOMER"
10   },
11   "products": [
12     {
13       "id": 15,
14       "name": "Sonata Wrist Watch",
15       "category": "Fashion",
16       "price": 999.0
17     }
18   ],
19   "quantity": 2
20 }
```

The response status is 200 OK, with the following JSON data:

```
1 {
2   "id": 6,
3   "customer": {
4     "id": 11,
5     "name": "Sandeep Verma",
6     "email": "sandeep.verma@abc.com",
7     "password": "$2a$10$CD49z0f0dKZKHDXNS192uTsdQ2yA0oTlTy0cDCKYdLrk6RJ3xpi",
8     "noOfOrders": 0,
9     "role": "CUSTOMER"
10   },
11   "products": [
12     {
13       "id": 15,
14       "name": "Sonata Wrist Watch",
15       "category": "Fashion",
16       "price": 999.0
17     }
18   ],
19   "quantity": 2
20 }
```

Update Quantity

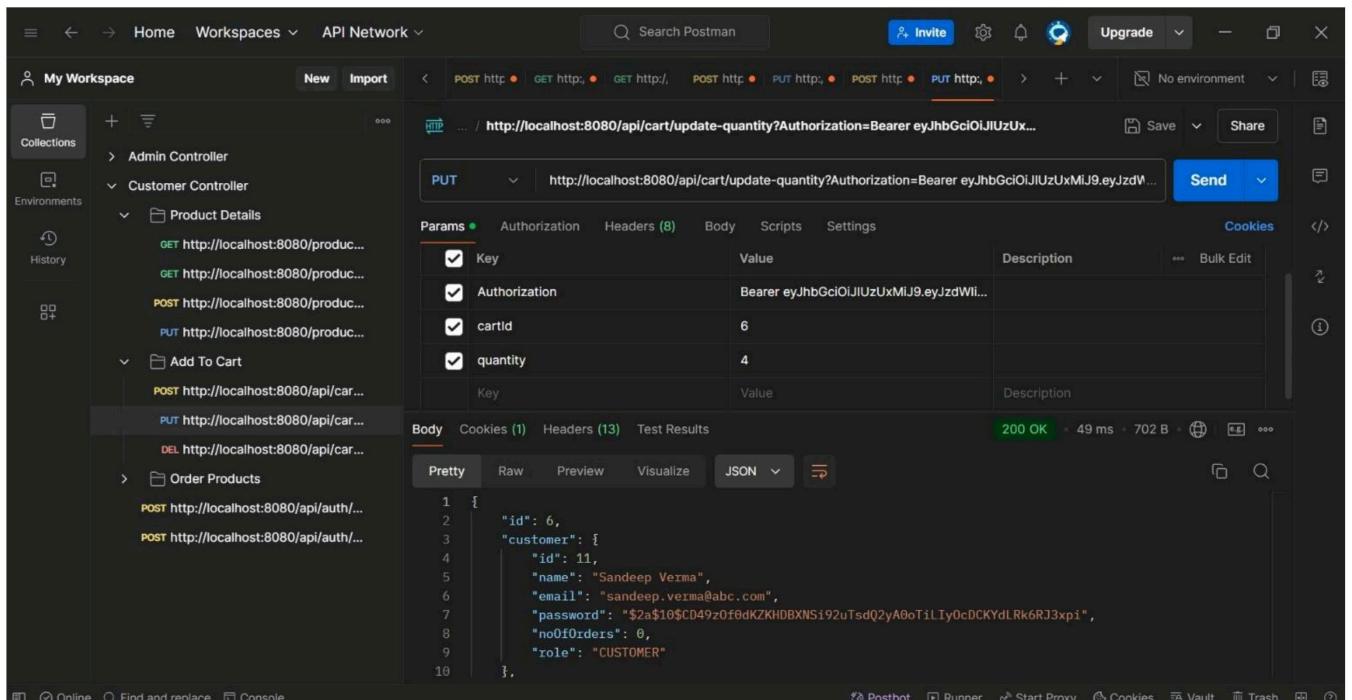
Customers can update the quantity of products as well



The screenshot shows the Postman interface with a collection named "Customer Controller". A POST request is selected with the URL `http://localhost:8080/api/cart/add-to-cart?Authorization=Bearer eyJhbGciOiJIUzUxMiJ9eyJzdWIiOiJ...` . The "Body" tab displays a JSON payload:

```
1 {
2     "id": 6,
3     "customer": {
4         "id": 11,
5         "name": "Sandeep Verma",
6         "email": "sandeep.verma@abc.com",
7         "password": "$2a$10$CD49zOf0dKZKHDBXNSi92uTsdQ2yA0oTiLiY0cDCKYdLRk6RJ3xpI",
8         "noOfOrders": 0,
9         "role": "CUSTOMER"
10    },
11    "products": [
12        {
13            "id": 15,
14            "name": "Sonata Wrist Watch",
15            "category": "Fashion",
16            "price": 999.0
17        }
18    ],
19    "quantity": 2
}
```

In the above image the product quantity was initially 2 which is then updated to 4 in the below image



The screenshot shows the Postman interface with the same collection. A PUT request is selected with the URL `http://localhost:8080/api/cart/update-quantity?Authorization=Bearer eyJhbGciOiJIUzUxMiJ9eyJzdWIiOiJ...` . The "Params" tab shows parameters: Key (checked), Value (empty), Description (empty), and Bulk Edit (checkbox). The "Body" tab displays a JSON payload:

```
1 {
2     "id": 6,
3     "customer": {
4         "id": 11,
5         "name": "Sandeep Verma",
6         "email": "sandeep.verma@abc.com",
7         "password": "$2a$10$CD49zOf0dKZKHDBXNSi92uTsdQ2yA0oTiLiY0cDCKYdLRk6RJ3xpI",
8         "noOfOrders": 0,
9         "role": "CUSTOMER"
10    },
11    "products": [
12        {
13            "id": 15,
14            "name": "Sonata Wrist Watch",
15            "category": "Fashion",
16            "price": 999.0
17        }
18    ],
19    "quantity": 4
}
```

Delete a cart

The screenshot shows the Postman interface with the following details:

- URL:** `http://localhost:8080/api/cart/remove?Authorization=Bearer eyJhbGciOiJIUzUxMiJ9eyJz...`
- Method:** `DELETE`
- Params:** `Authorization` (Value: Bearer eyJhbGciOiJIUzUxMiJ9eyJz...), `cartId` (Value: 6)
- Body:** `Pretty` view shows the response: "Cart item removed successfully."
- Headers:** `Content-Type` is set to `application/json`.
- Status:** `200 OK`, `57 ms`, `443 B`

Using the cart id, customers can place order. Below we can see that when the order is placed, in the output we get the customer details, product details, shipping address and the total amount depending on the quantity.

The screenshot shows the Postman interface with the following details:

- URL:** `http://localhost:8080/orders/4/place?Authorization=Bearer eyJhbGciOiJIUzUxMiJ9eyJz...`
- Method:** `POST`
- Params:** `cartId` (Value: 6), `shippingAddress` (Value: 123/A, Indira Nagar,Bangalore)
- Body:** `Pretty` view shows the response JSON:

```
2   "id": 7,
3   "customer": {
4     "id": 11,
5     "name": "Sandeep Verma",
6     "email": "sandeep.verma@abc.com",
7     "password": "$2a$10$CD49z0f0dKZKHBXNSi92uTsdQ2yA0oTiLiY0cDCKYdLrk6RJ3xpi",
8     "noOfOrders": 0,
9     "role": "CUSTOMER"
10   },
11   "products": [
12     {
13       "id": 15,
14       "name": "Sonata Wrist Watch",
15       "category": "Fashion",
16       "price": 999.0
17     }
18   ],
19   "shippingAddress": "123/A, Indira Nagar,Bangalore",
20   "totalAmount": 3996.0
```

Deleting an order:

The screenshot shows the Postman interface. On the left, the 'My Workspace' sidebar lists collections like 'Admin Controller' and 'Customer Controller'. Under 'Customer Controller', there are sub-folders 'Product Details', 'Add To Cart', and 'Order Products', each containing several API endpoints. The main workspace shows a DELETE request to 'http://localhost:8080/orders/7/delete'. The 'Headers' tab is selected, showing an 'Authorization' header with the value 'Bearer eyJhbGciOiJIUzUxMiJ9eyJzdWI...'. The response section shows a status of 200 OK with a response body containing the message 'Order with cart ID 7 deleted successfully.'

Swagger :

All of the endpoints are successfully working in swagger.

The screenshot shows the Swagger UI interface. At the top, it says 'API Documentation 1.0 OAS3' and 'API for eCommerce website'. Below this, there's a 'Servers' dropdown set to 'http://localhost:8080 - Generated server url' and an 'Authorize' button. The main area is divided into sections: 'product-controller', 'order-controller', and 'cart-controller'. The 'product-controller' section contains five endpoints: a PUT endpoint for '/products/update/product/{id}' (highlighted in orange), a POST endpoint for '/products/add' (highlighted in green), a POST endpoint for '/products/addMultiple', a GET endpoint for '/products/{id}', and a GET endpoint for '/products/all'. The 'order-controller' and 'cart-controller' sections are currently collapsed, indicated by a downward arrow icon.

order-controller

PUT	/orders/{id}/update	✓ 🔒
POST	/orders/{cartId}/place	✓ 🔒
DELETE	/orders/{id}/delete	✓ 🔒

cart-controller

admin-controller

GET	/admin/product/{id}	✓ 🔒
PUT	/admin/product/{id}	✓ 🔒
DELETE	/admin/product/{id}	✓ 🔒
GET	/admin/customer/{id}	✓ 🔒

auth-controller

POST	/api/auth/register	✓ 🔒
POST	/api/auth/login	✓ 🔒

customer-controller

POST	/admin/customers/register	✓ 🔒
------	---------------------------	-----

Schemas

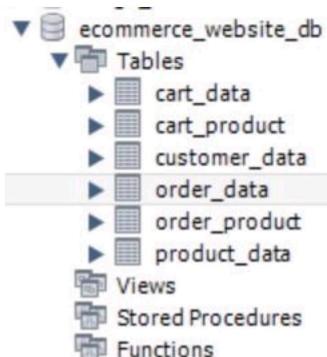
- Product >
- Customer >

Schemas

- Product >
- Customer >
- Order >
- Cart >
- AddToCartRequest >
- LoginRequest >

MySQL Database

Schema



Customer Table

	id	email	name	no_of_orders	password	role
▶	1	mihir.jain@xyz.com	Mihir Jain	45	\$2a\$10\$SCNo/0CFuWKUy0Kkvjs49.RIFSocY2sf...	CUSTOMER
▶	2	sahil.patel@example.com	Sahil Patel	65	\$2a\$10\$jfr2swCPAxWa2yyEzrjEuJl6t/VD/Hd...	CUSTOMER
▶	3	pooja.singh@abc.com	Pooja Singh	21	\$2a\$10\$ftcFtsOKxFb9q.RcaqU/3eXmZEHSjnUf...	CUSTOMER
▶	4	anjali.sharma@abc.com	Anjali Sharma	13	\$2a\$10\$bV.JicdHnRAd.wxG9/C1h.CDjpG8KbiUL...	CUSTOMER
▶	5	aditya.chopra@example.com	Aditya Chopra	19	\$2a\$10\$0ZnNWSL7q/w3o0oFI6bucuo4rn2NSzr...	CUSTOMER
▶	7	sanjay.roy@example.com	Sanjay Roy	17	\$2a\$10\$tFPao6pb9H/QG2wezERgT.H2PVunMP...	CUSTOMER
▶	8	kartik.patel@abc.com	Kartik Patel	0	\$2a\$10\$7S9vZtbzKwtvkwFwMDQtpuKxrYnAgKK...	CUSTOMER
▶	9	raj.sri@xyz.com	Raj Srivastav	0	\$2a\$10\$HQF/7FDZjO7p267zYd0G4O3ck.yJpHh...	CUSTOMER
*	HULL	HULL	HULL	HULL	HULL	HULL

Product Table

	id	category	name	price
▶	1	Electronics	One Plus 12R	39999
▶	2	Electronics	Hp Pavilion 15 Laptop	58990
▶	3	Electronics	Samsung Galaxy Tab S9 FE	34499
▶	4	Electronics	Nothing Watch Pro 2	5499
▶	5	Electronics	Boat Bluetooth Speaker Stone 14W	3199
▶	6	Electronics	Google Pixel 7	32299
▶	7	Footwear	Nike Men Running Shoes	4149
▶	8	Footwear	Red Tape Sneakers	1539
▶	9	Footwear	Mast & Harbour Sneakers	1099
▶	10	Footwear	Crocs	3297

Cart table for product

Result Grid		Filter Rows:
	cart_id	product_id
▶	1	16
	3	2
	3	2
	5	16

Cart table for customer details

Result Grid				Filter Rows:
	id	quantity	customer_id	
▶	1	2	3	
	3	3	5	
	5	2	7	
*	NULL	NULL	NULL	

Order table with product id

Result Grid		Filter Rows:
	order_id	product_id
▶	3	2
	4	12
	5	16
*	NONE	NONE

Order table with customer and shipping address details

Result Grid				Filter Rows:	Edit:	Export
	id	shipping_address	total_amount	customer_id		
▶	1	123/C, Nehru Chowk, Delhi	5998	3		
	2	123/Y, Gandhi Chowk, Ahmedabad	54990	5		
	3	123/Y, Gandhi Chowk, Ahmedabad	329940	5		
	4	123/z, Bangalore	11892	8		
*	5	123/A,Delhi	5998	7		
	NONE	NONE	NONE	NONE		

order_data 1 ×