MICROSERVICES ASSIGNMENT SUBMISSION

- **Pull Docker Images of All Microservices from Docker Hub**

```
docker pull dpcode72/orderapi:1.0
docker pull dpcode72/cartapi:4.0
docker pull dpcode72/productapi:2.0
docker pull dpcode72/authapi:3.0
docker pull dpcode72/apigateway:5.0
```

- **After Pulling Docker images of All Microservices then Please Create a custom Network using command because I used custom network that is (172.20.0.1)**

```
sudo docker network create --driver bridge
nagarroAss
```

- **After this Please you wish to run any of above images of Microservices then Please use below command to run**

```
sudo docker run -p 7000:80 --network nagarroAss
dpcode72/apigateway:5.0
```

- **As you can see above command that is used to run apigateway(Ocelot) Service & So you will wish to run further images then please use below Command**

- **Rest of all command to run the Docker Images like ProductAPI, CartAPI, AuthAPI & OrderAPI**

```
sudo docker run -p 8082:80 --network nagarroAss
dpcode72/productapi:2.0
sudo docker run -p 8081:80 --network nagarroAss
dpcode72/authapi:3.0
```

```
sudo docker run -p 8083:80 --network nagarroAss dpcode72/cartapi:4.0
sudo docker run -p 8084:80 --network nagarroAss dpcode72/orderapi:1.0
```

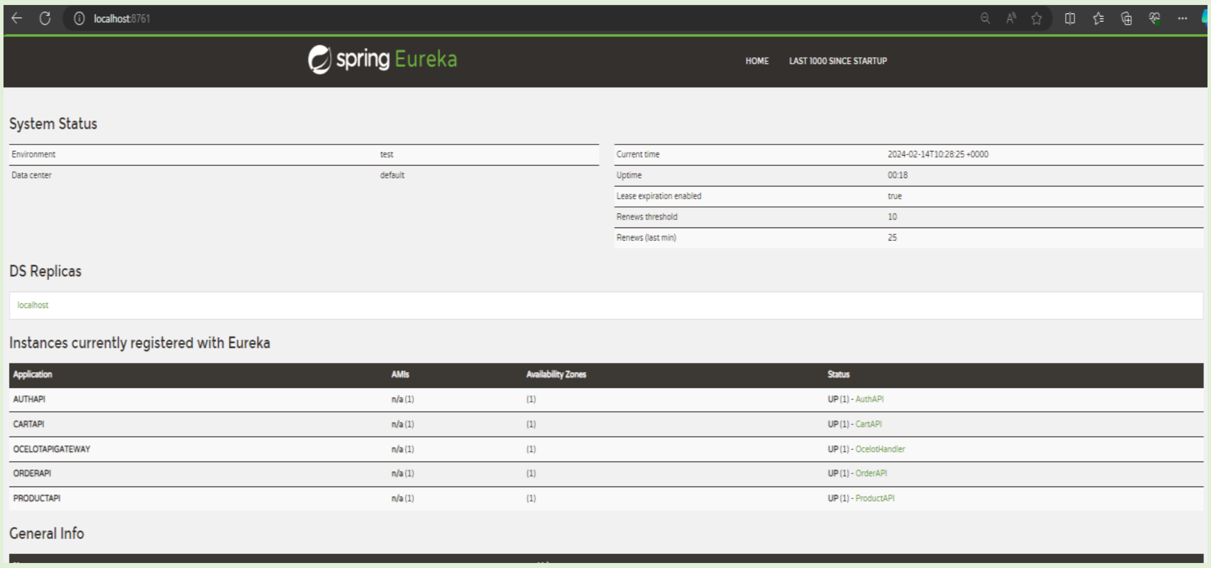
- **Please, Also Run two Commands to let Microservice be registered on EUREKA As well as TO communicate Microservice to each other with help of Message Broken Service (RabitMQ).**

Commands are Below:-

```
//RabbitMQ
docker run -d --hostname rmq --name rabbit-server -p 9000:15672 -p 5672:5672 rabbitmq:3-management
//EUREKA
docker run --publish 8761:8761 steeltoeoss/eureka-server
```

- **After Running all above Commands you can see such interface on <u>Eureka Server</u> Where all Services are register On Eureka As well as you can also see <u>Message Broken Service</u> on Web Browser Queue are generated on name of <u>Consumermessage</u>**

EUREKA UI Where you can all Services are registered.



System Status

Environment	test
Data center	default

Current time	2024-02-14T10:28:25 +0000
Uptime	00:18
Lease expiration enabled	true
Renews threshold	10
Renews (last min)	25

DS Replicas

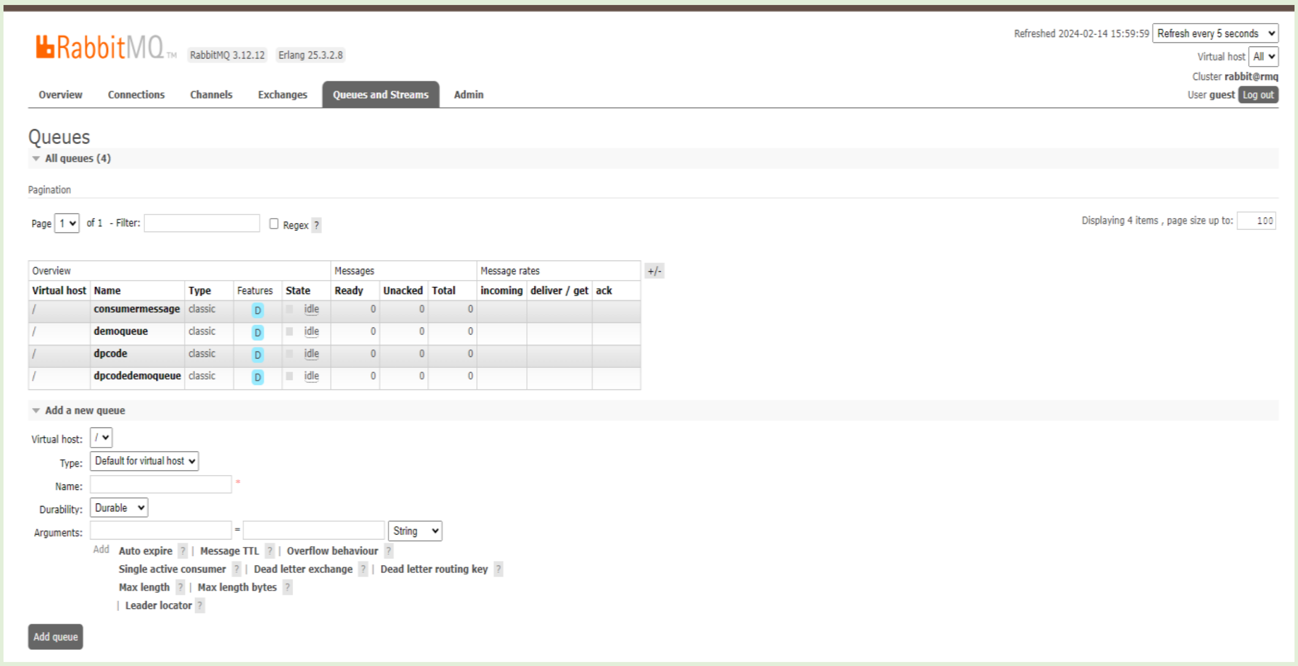
localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
AUTHAPI	n/a (1)	(1)	UP (1) - AuthAPI
CARTAPI	n/a (1)	(1)	UP (1) - CartAPI
OCELOTAPIGATEWAY	n/a (1)	(1)	UP (1) - OcelotHandler
ORDERAPI	n/a (1)	(1)	UP (1) - OrderAPI
PRODUCTAPI	n/a (1)	(1)	UP (1) - ProductAPI

General Info

As you can also see the Message Broken Service(Rabbitmq)



RabbitMQ 3.12.12 Erlang 25.3.2.8

Refreshed 2024-02-14 15:59:59 Refresh every 5 seconds

Virtual host All

Cluster rabbit@rmq

User guest Log out

Overview Connections Channels Exchanges Queues and Streams Admin

Queues

All queues (4)

Pagination

Page 1 of 1 - Filter: Regex ?

Displaying 4 items , page size up to: 100

Overview					Messages			Message rates		
Virtual host	Name	Type	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack
/	consumermessage	classic	D	idle	0	0	0			
/	demoqueue	classic	D	idle	0	0	0			
/	dpcode	classic	D	idle	0	0	0			
/	dpcodedemoqueue	classic	D	idle	0	0	0			

Add a new queue

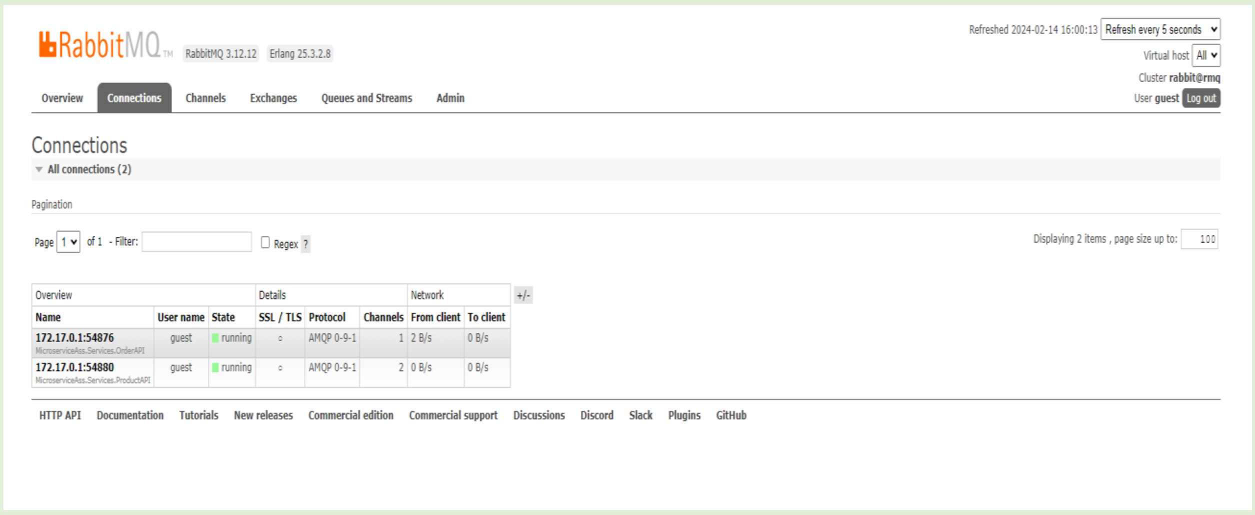
Virtual host:	/
Type:	Default for virtual host
Name:	*
Durability:	Durable
Arguments:	= String

Add Auto expire ? | Message TTL ? | Overflow behaviour ?
Single active consumer ? | Dead letter exchange ? | Dead letter routing key ?
Max length ? | Max length bytes ?
| Leader locator ?

Add queue

For shake of understanding of Rabbitmq I tried to make Service Communicate to each other like ProductAPI Service is communicating with OrderAPI Service

As you can see in below Images



Refreshed 2024-02-14 16:00:13 Refresh every 5 seconds
Virtual host All
Cluster rabbit@rmq
User guest

Connections

All connections (2)

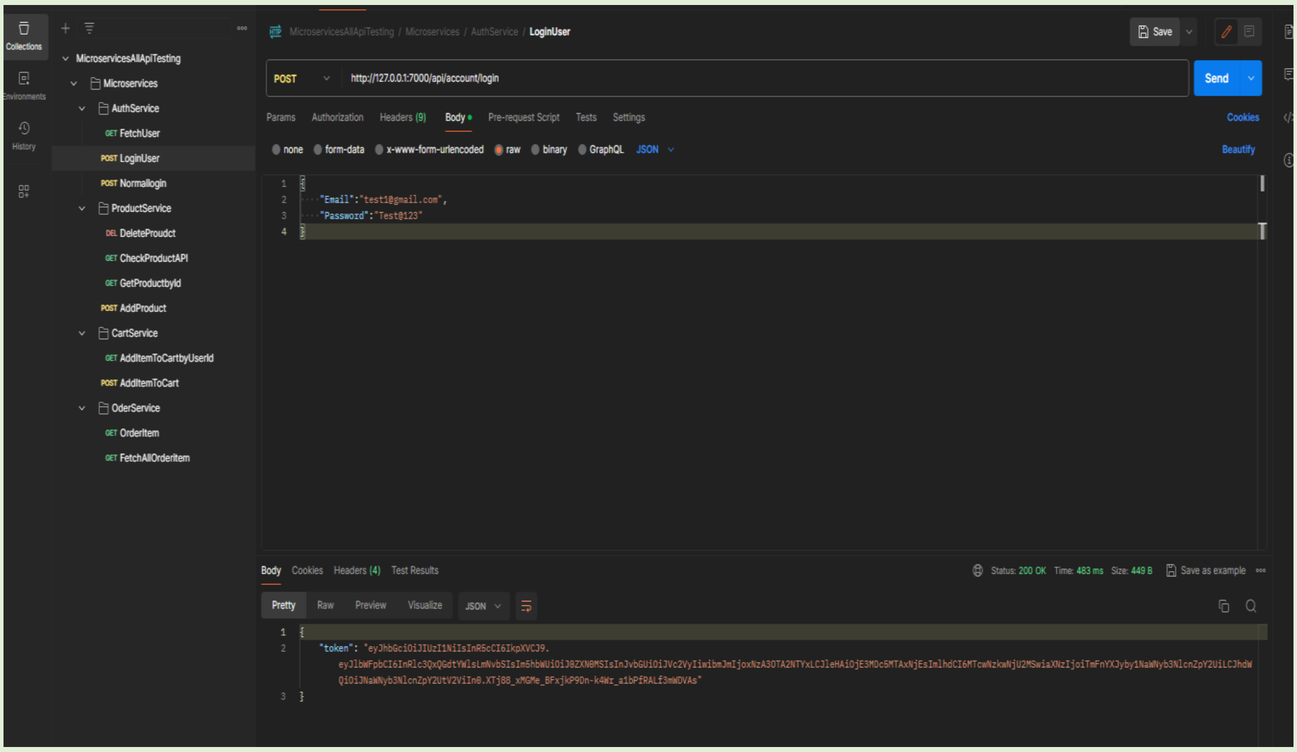
Pagination

Page 1 of 1 - Filter: Regex ?

Displaying 2 items , page size up to: 100

Overview			Details			Network	
Name	User name	State	SSL / TLS	Protocol	Channels	From client	To client
172.17.0.1:54876 MicroserviceAss.Services.OrderAPI	guest	running	○	AMQP 0-9-1	1	2 B/s	0 B/s
172.17.0.1:54880 MicroserviceAss.Services.ProductAPI	guest	running	○	AMQP 0-9-1	2	0 B/s	0 B/s

Import the postman collection named“MicroservicesAllApiTesting” in the postman application.



MicroservicesAllApiTesting / Microservices / AuthService / LoginUser

POST http://127.0.0.1:7000/api/account/login

```
{
    "Email":"test1@gmail.com",
    "Password":"Test@123"
}
```

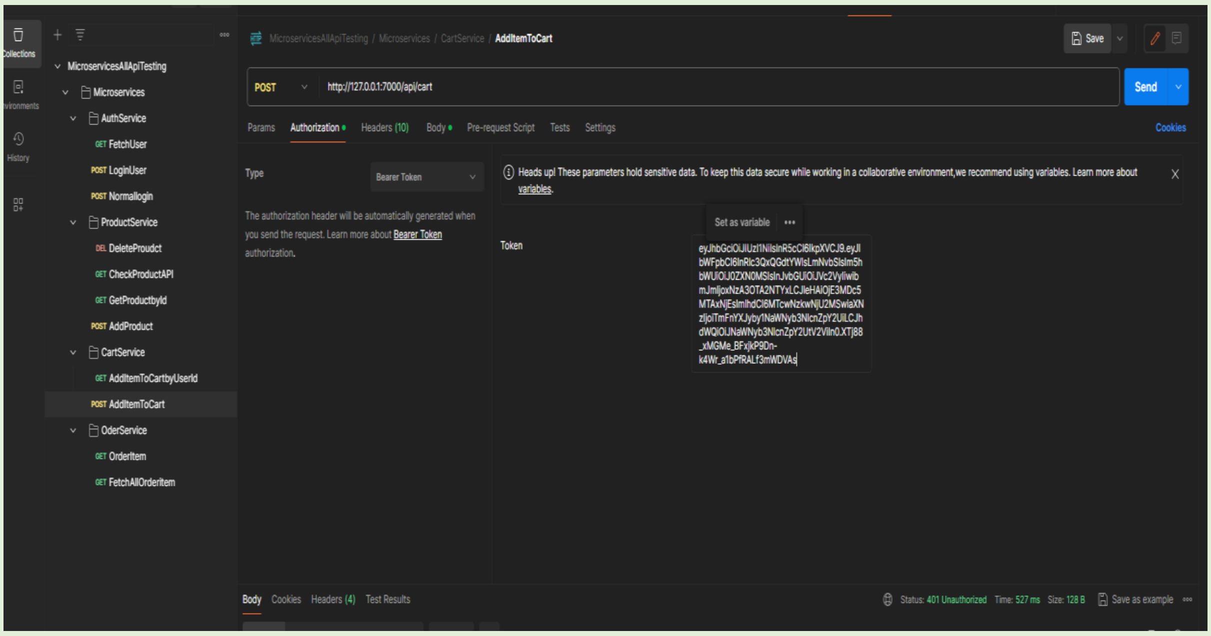
Status: 200 OK Time: 483 ms Size: 449 B

```
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJlbWFpbCI6InRlc3QxQGdtYWlsLmNvbSIsIm5hbWUiOiJ0ZXN0MSIsInJvbGUiOiJVc2VyIiwibmJmIjoxNzA3OTA2NTYxLCJleHAiOjE3MDc5MTAxNjEsImlhdCI6MTcwNzkwNjU2MSwiaXNzIjoiTmFnYXJyby1NaWNyb3NlcnZpY2UiLCJhdW
        QiOiJNaWNyb3NlcnZpY2UtV2ViIn0.XTj88_xMGMe_BFxjkP9Dn-k4Wr_a1bPfRALf3mWDVAs"
}
```

Now, after giving right credentials for login as you can see that user will get jwt token to access the Microservices Service such Proudct

Service, Cart Service & Order Service, but passing Jwt token with every request to get authorized to use service like add item to cart & place order & so on….

As you can see in below images of POSTMAN like how things are working



MicroservicesAllApiTesting / Microservices / CartService / AddItemToCart

POST http://127.0.0.1:7000/api/cart

Params | Authorization | Headers (10) | Body | Pre-request Script | Tests | Settings

Type: Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about Bearer Token authorization.

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment,we recommend using variables. Learn more about variables.

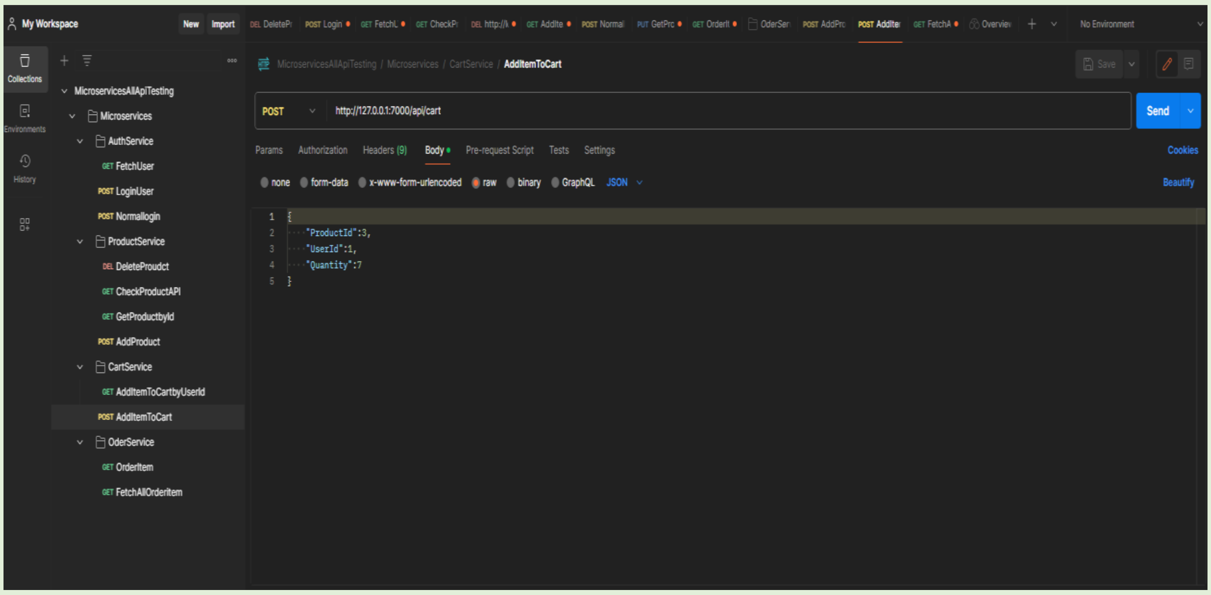
Token:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJI
bWFpbCI6InRlc3QxQGdtYWlsLmNvbSIsIm5h
bWUiOiJ0ZXN0MSIsInJvbGUiOiJVc2VyIiwib
mJmIjoxNzA3OTA2NTYxLCJleHAiOjE3MDc5
MTAxNjEsImlhdCI6MTcwNzkwNjU2MSwiaXN
zIjoiTmFnYXJyby1NaWNyb3NlcnZpY2UiLCJh
dWQiOiJNaWNyb3NlcnZpY2UtV2ViIn0.XTj88
_xMGMe_BFxjkP9Dn-
k4Wr_a1bPfRALf3mWDVAs
```

Body | Cookies | Headers (4) | Test Results

Status: 401 Unauthorized   Time: 527 ms   Size: 128 B

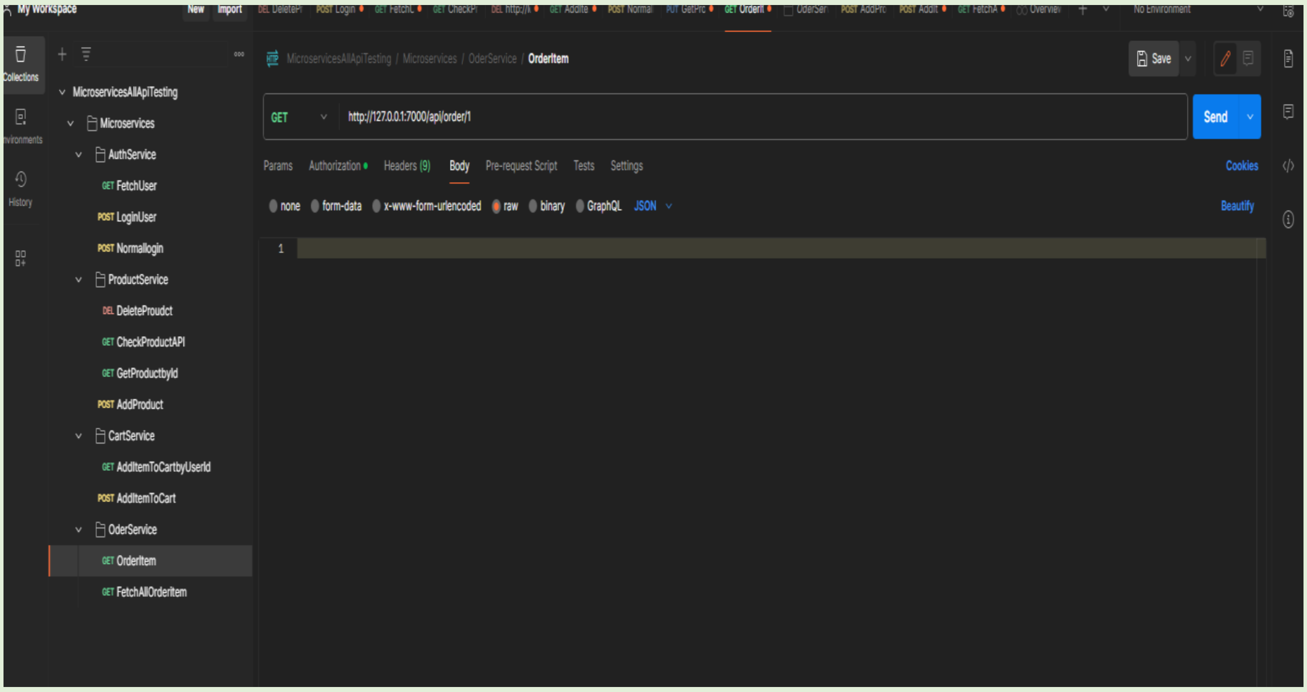
Here you can see that user trying to add item to cart



POST http://127.0.0.1:7000/api/cart

```
{
    "ProductId":3,
    "UserId":1,
    "Quantity":7
}
```

Later user can place order which he/she added to the cart as you can see in below images of POSTMAN like how things are working



GET http://127.0.0.1:7000/api/order/1

For more information such you will like to know like EUREKA, Message Broken Service & My Docker Hub Repository

I am giving reference of all those below please check in case you found something messing or anything

EUREKA DASHBOARD URL
http://localhost:8761/
MESSAGE BROKEN SERVICE URL
http://localhost:9000/#/connections
MY DOCKER HUB REPOSITORY URL
https://hub.docker.com/u/dpcode72