



## Computer Engineering Department

### Course Name: Distributed Operating System

### PROJECT 2: Bazar.com: A Multi-tier Online Book Store

*tasked to design Bazar.com - the World's smallest book store.*

Dr.Samer Arandi

Student Name	Student Id
Aya Moeen	12027735
Razan Khammash	12028582





## Create Images:

First we are create 3 images as shown in figure




```
C:\Users\awwad\docker\project1>docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
frontend             latest             6f871c804663       14 hours ago       211MB
catalog              latest             b814f252c830       14 hours ago       211MB
order                latest             f282cc8d3a0c       14 hours ago       211MB
```

The frontend used to write code in the operating system host and test it using PostMan and the frontend put in port 3001.






Catalog the server that used to receive request from frontend just request [search and info] and the catalog put it in port 3002.

Order the server that used to receive request from frontend just [order] and the order server put it in port 3003.

## Tools and library used in project :

-  Docker
-  Nodejs
-  REST Api

### Library

-  Express
-  Axios
-  csv-parser
-  body-parser
-  fs

## Describe the program :

We are designed the project as multitier book store [Bazar.com]. build it as microservice architecture and two tiers front-end [frontend image] accept user request and the backend have two server catalog and order we are implement at as separate microservice .



## How it work:

### Frontend :

The frontend have three operation search, info, and order the user can interact with system using these operation we are used postman to test the system. When have a request then the frontend send it to the server should have this request [search and info go to server catalog] & [order go to server order].

### Backend:

- Catalog server this server or microservice contain the information about book and if available for sale. should receive query and update operation, from frontend receive the [search, info] and then the server response the information related to request receive and update operation received it from order server [to show if have a book available to sale then decrement it or it out of stock ] then update the file or database.
- Order server this server for the order of user support just order operation when receive request then order server send request to catalog server .and receive if the book available or not .

## Design tradeoffs:

### Monolithic or Microservice :

we are choice microservice architecture because allow for better scalability and fault, but it more complex for deployment and communication between service.

### RESTful API:

We are choice RESTful API because it easy, simple in communication and allow better decoupling between components. But it have more overhead.



### *Lightweight frameworks:*

*We are choice lightweight framework nodejs and Express because it reduce complicity of the system, easy for use, simple, easy for maintain and develop. But it lack some feature that exist in heavyweight framework.*

### *Describe possible improvements and extensions:*

- ✚ Database: program use text file for store the data of two server [catalog and order] can use database like SQLite, MonogDB so it improve the scalability and management for data.*
- ✚ Container Orchestration: use tools to deploy the microservice like Kubernetes it improve scalability.*
- ✚ Authentication and Authorization : implement it to enhance security*

### *Cases not work correctly:*

- ✚ Error handling: maybe the program not handle all error, so when enhance the error and fault tolerance it improve robustness.*
- ✚ Concurrent request: maybe have problem of performance issue under heavy load, to enhance it can use caching and load balancing.*



## Output :

First run the three microservice :

```
C:\Windows\System32\cmd.e x + v
Microsoft Windows [Version 10.0.22621.3296]
(c) Microsoft Corporation. All rights reserved.

C:\Users\awwad\docker\project2>docker run -it -v ./home catalog
Server ready at: http://localhost: 3002
The CSV file was written successfully


C:\Windows\System32\cmd.e x + v
REPOSITORY TAG IMAGE ID CREATED SIZE
frontend latest 6f871c804663 14 hours ago 211MB
catalog latest b814f252c830 14 hours ago 211MB
order latest f282cc8d3a0c 14 hours ago 211MB
<none> <none> f5b2b77159ec 2 days ago 432MB
catlog latest 6525d45e0f88 2 days ago 432MB
myubuntu/ubuntu-gcc-nano latest 044b0a6c87d4 3 days ago 297MB
ubuntu latest ca2b0f26964c 2 weeks ago 77.9M

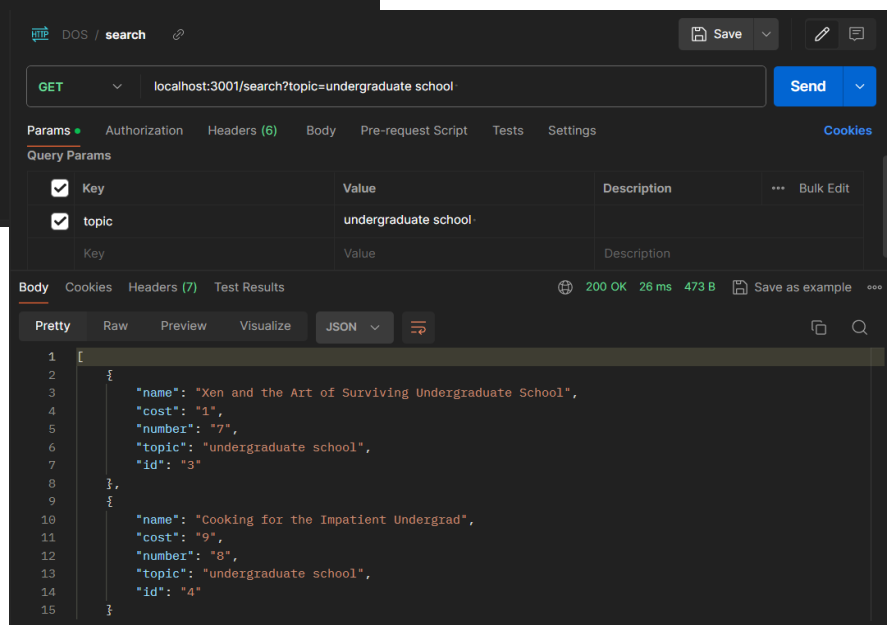
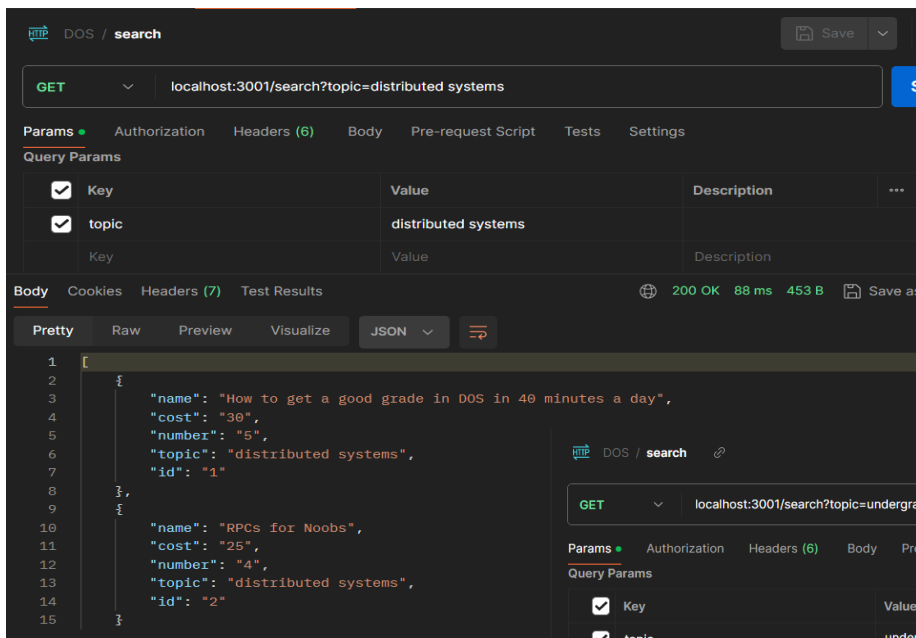
C:\Users\awwad\docker\project1>docker run -it -v ./home frontend
Server ready at: http://localhost: 3001

C:\Windows\System32\cmd.e x + v
Microsoft Windows [Version 10.0.22621.3296]
(c) Microsoft Corporation. All rights reserved.

C:\Users\awwad\docker\project3>docker run -it -v ./home order
Server ready at: http://localhost: 3003
```

Then using postman test it

 GET --- search





## GET --- info

HTTP DOS / info

GET localhost:3001/search/1

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (7) Test Results 200 OK

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "name": "How to get a good grade in DOS in 40 minutes a day",
4     "cost": "30",
5     "number": "5",
6     "topic": "distributed systems",
7     "id": "1"
8   }
9 ]
```

HTTP DOS / info

GET localhost:3001/search/3

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (7) Test Results 200 OK

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "name": "Xen and the Art of Surviving Undergraduate School",
4     "cost": "1",
5     "number": "7",
6     "topic": "undergraduate school",
7     "id": "3"
8   }
9 ]
```

HTTP DOS / info

GET localhost:3001/search/2

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (7) Test Results 200 OK

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "name": "RPCs for Noobs",
4     "cost": "25",
5     "number": "4",
6     "topic": "distributed systems",
7     "id": "2"
8   }
9 ]
```

HTTP DOS / info

GET localhost:3001/search/4

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (7) Test Results 200 OK

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "name": "Cooking for the Impatient Undergrad",
4     "cost": "9",
5     "number": "8",
6     "topic": "undergraduate school",
7     "id": "4"
8   }
9 ]
```



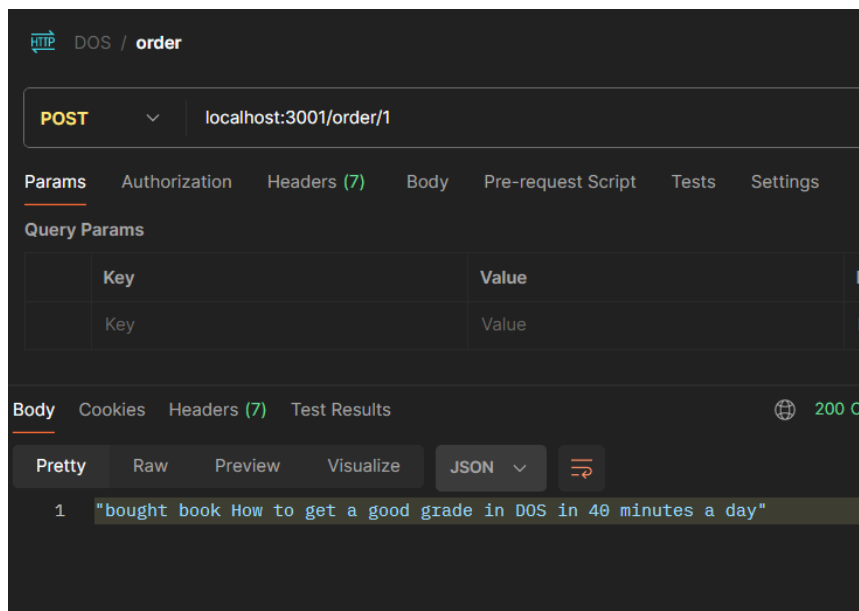
## POST --- order

The file before update

```
Dockerfile BooksFile.csv
1 name,cost,number,topic,id
2 How to get a good grade in DOS in 40 minutes a day,30,5,distributed systems,1
3 RPCs for Noobs,25,4,distributed systems,2
4 Xen and the Art of Surviving Undergraduate School,1,7,undergraduate school ,3
5 Cooking for the Impatient Undergrad,9,8,undergraduate school ,4
6
```

When user need to order the book with id =1

Then the response is :



And the file after update will see the book with id =1 the quantity decrement :

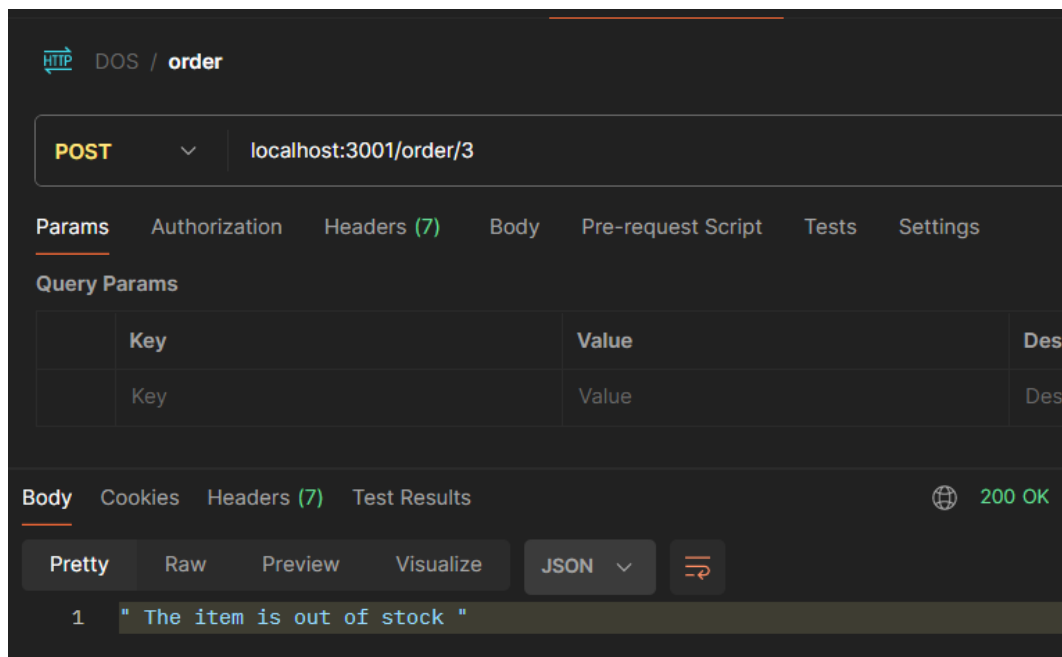
```
Dockerfile BooksFile.csv
1 name,cost,number,topic,id
2 How to get a good grade in DOS in 40 minutes a day,30,4,distributed systems,1
3 RPCs for Noobs,25,4,distributed systems,2
4 Xen and the Art of Surviving Undergraduate School,1,7,undergraduate school ,3
5 Cooking for the Impatient Undergrad,9,8,undergraduate school ,4
6
```



*When try to order the book with id =3 and the book is out of stock*

```
Dockerfile x BooksFile.csv x
1 name,cost,number,topic,id
2 How to get a good grade in DOS in 40 minutes a day,30,4,distributed systems,1
3 RPCs for Noobs,25,4,distributed systems,2
4 Xen and the Art of Surviving Undergraduate School,1,0,undergraduate school ,3
5 Cooking for the Impatient Undergrad,9,8,undergraduate school ,4
6
```

*The response :*



### *Conclusion:*

*We are designed system scalable for mange the book store microservice REST API and lightweight framework as docker so we are achieve flexibility , modularity ,easy development and implementation, scalable.*