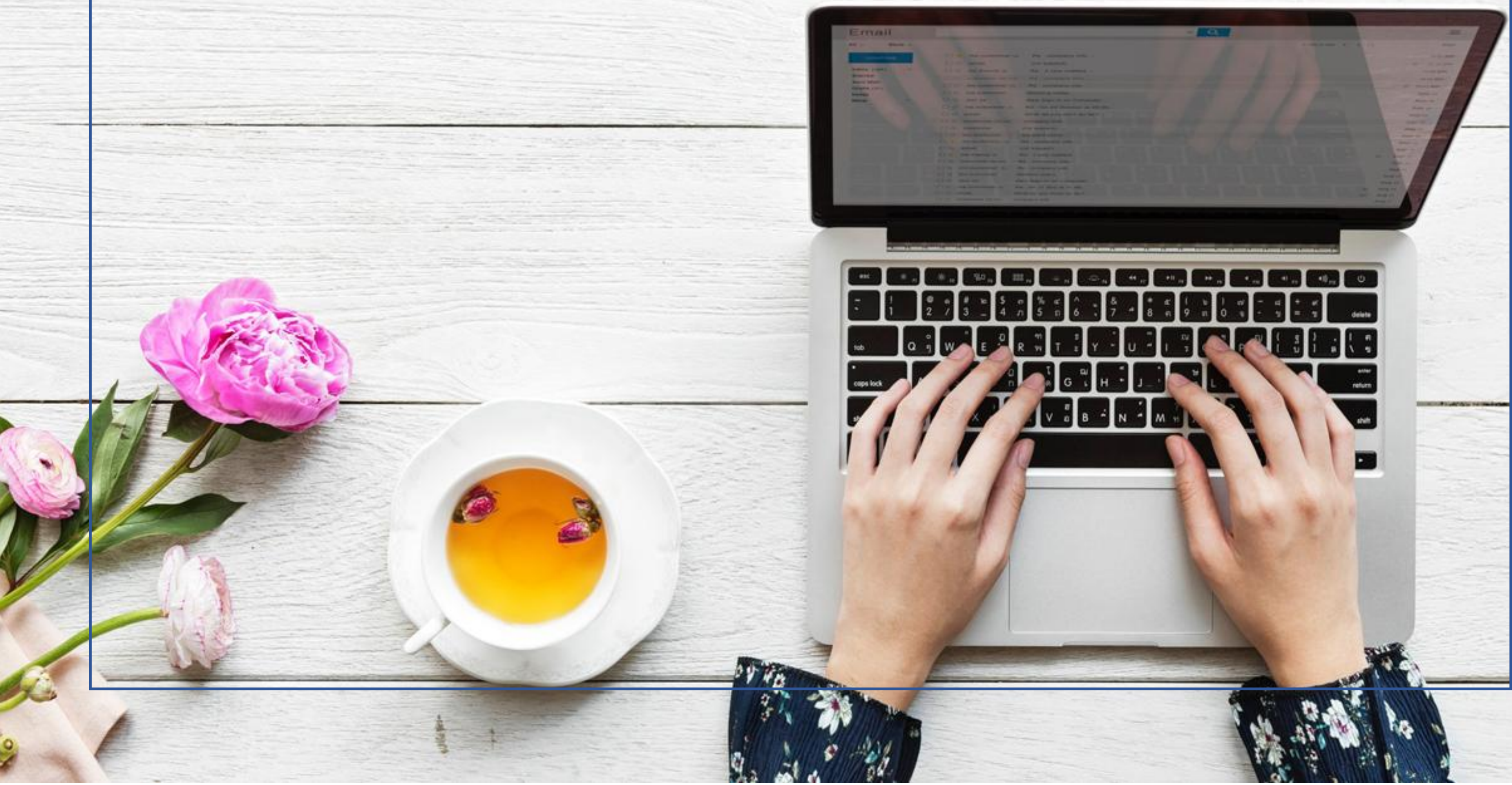
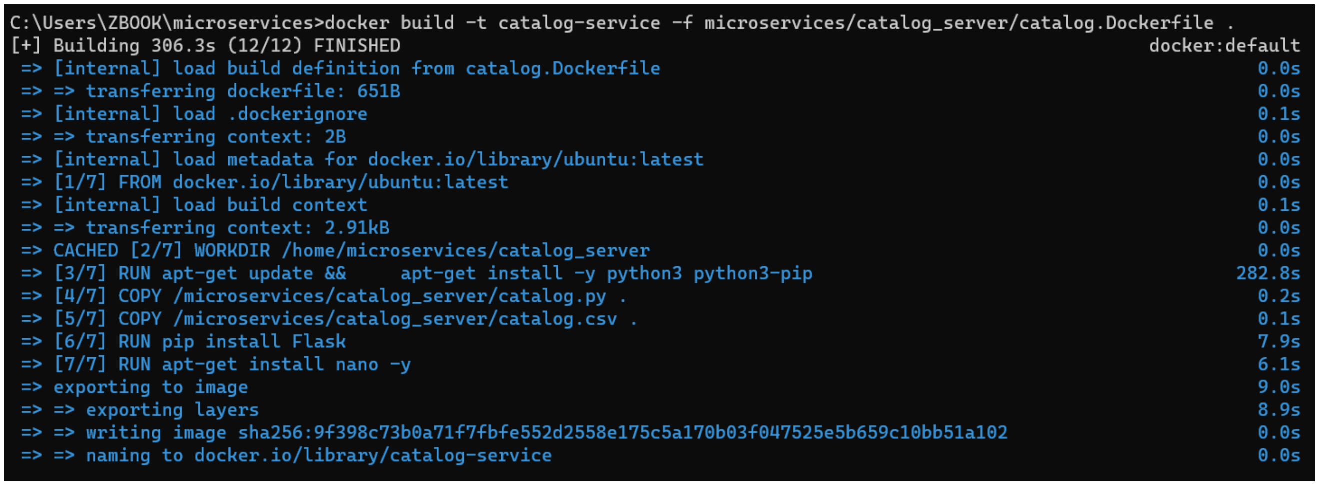
Output generated for Bazar.com project

Lina Qurom
11924435



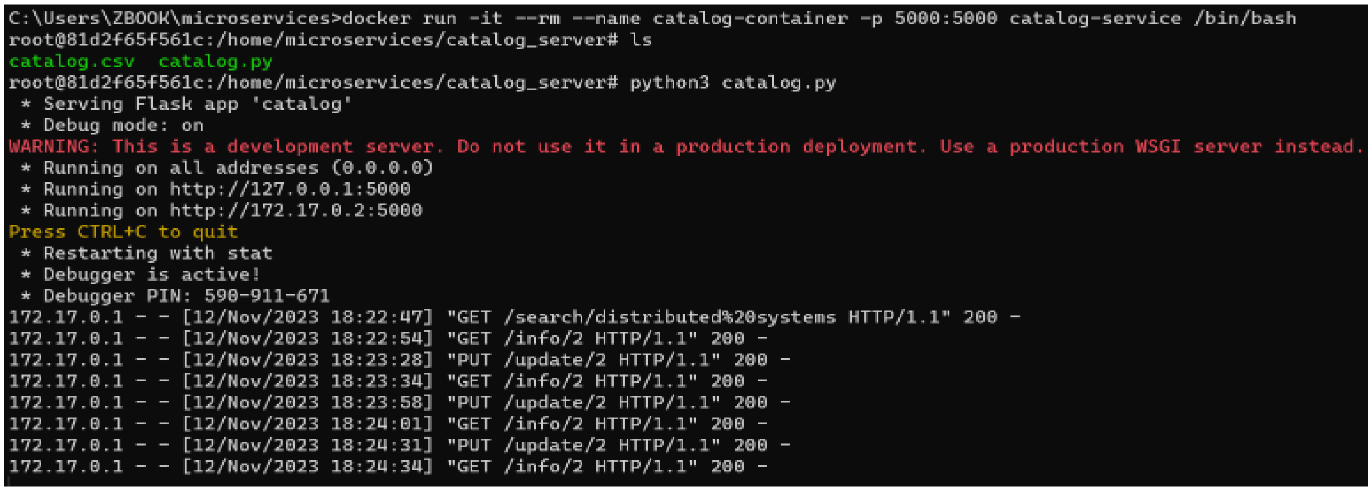
How to Use It With Docker:

- Building catalog, order, and frontend services image:
  1. docker build -t catalog-service -f microservices/catalog_server/catalog.Dockerfile .
  2. docker build -t order-service -f microservices/order_server/order.Dockerfile .
  3. docker build -t frontend-service -f microservices/ frontend _server/ frontend.Dockerfile .



```
C:\Users\ZBOOK\microservices>docker build -t catalog-service -f microservices/catalog_server/catalog.Dockerfile .
[+] Building 306.3s (12/12) FINISHED                                                              docker:default
 => [internal] load build definition from catalog.Dockerfile                                                0.0s
 => => transferring dockerfile: 651B                                                                        0.0s
 => [internal] load .dockerignore                                                                           0.1s
 => => transferring context: 2B                                                                             0.0s
 => [internal] load metadata for docker.io/library/ubuntu:latest                                            0.0s
 => [1/7] FROM docker.io/library/ubuntu:latest                                                              0.0s
 => [internal] load build context                                                                           0.1s
 => => transferring context: 2.91kB                                                                         0.0s
 => CACHED [2/7] WORKDIR /home/microservices/catalog_server                                                 0.0s
 => [3/7] RUN apt-get update &&     apt-get install -y python3 python3-pip                                282.8s
 => [4/7] COPY /microservices/catalog_server/catalog.py .                                                   0.2s
 => [5/7] COPY /microservices/catalog_server/catalog.csv .                                                  0.1s
 => [6/7] RUN pip install Flask                                                                             7.9s
 => [7/7] RUN apt-get install nano -y                                                                       6.1s
 => exporting to image                                                                                      9.0s
 => => exporting layers                                                                                     8.9s
 => => writing image sha256:9f398c73b0a71f7fbfe552d2558e175c5a170b03f047525e5b659c10bb51a102                0.0s
 => => naming to docker.io/library/catalog-service                                                          0.0s
```

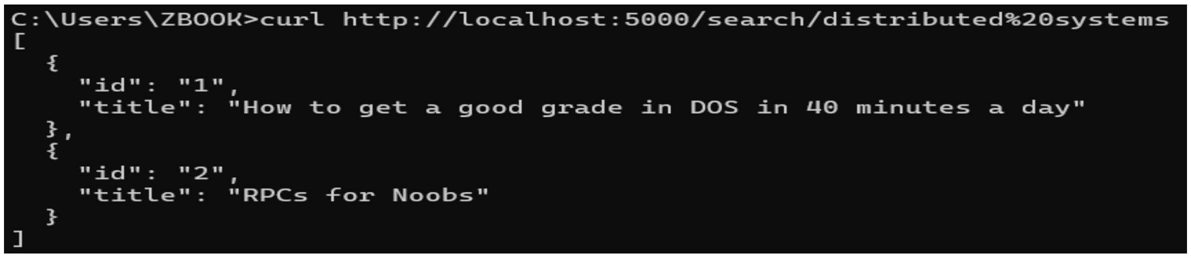
- Then, run the containers and run the python files, the servers are working now. Also note the status of the server when executing requests.



```
C:\Users\ZBOOK\microservices>docker run -it --rm --name catalog-container -p 5000:5000 catalog-service /bin/bash
root@81d2f65f561c:/home/microservices/catalog_server# ls
catalog.csv  catalog.py
root@81d2f65f561c:/home/microservices/catalog_server# python3 catalog.py
 * Serving Flask app 'catalog'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.17.0.2:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 590-911-671
172.17.0.1 - - [12/Nov/2023 18:22:47] "GET /search/distributed%20systems HTTP/1.1" 200 -
172.17.0.1 - - [12/Nov/2023 18:22:54] "GET /info/2 HTTP/1.1" 200 -
172.17.0.1 - - [12/Nov/2023 18:23:28] "PUT /update/2 HTTP/1.1" 200 -
172.17.0.1 - - [12/Nov/2023 18:23:34] "GET /info/2 HTTP/1.1" 200 -
172.17.0.1 - - [12/Nov/2023 18:23:58] "PUT /update/2 HTTP/1.1" 200 -
172.17.0.1 - - [12/Nov/2023 18:24:01] "GET /info/2 HTTP/1.1" 200 -
172.17.0.1 - - [12/Nov/2023 18:24:31] "PUT /update/2 HTTP/1.1" 200 -
172.17.0.1 - - [12/Nov/2023 18:24:34] "GET /info/2 HTTP/1.1" 200 -
```

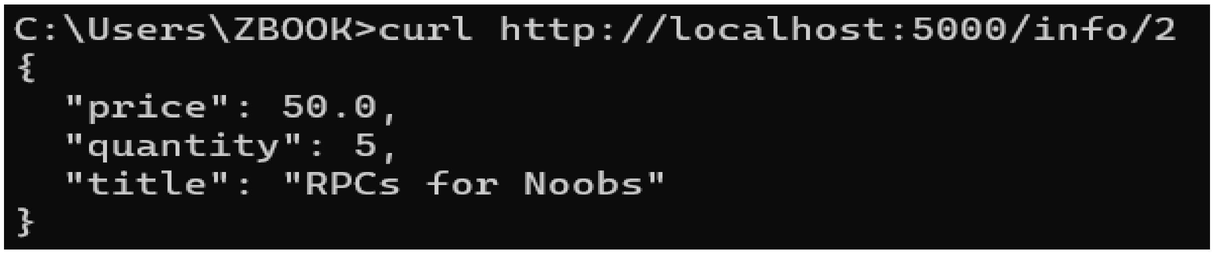
- In the above, I run the server without mounting, to make mounting to be the changes that occur on the container show in the host files also by these commands:
  1. docker run -v .:/microservices/catalog_server/ -p 5000:5000 catalog-service
  2. docker run -v .:/microservices/order_server/ -p 5001:5001 order-service
  3. docker run -v .:/microservices/frontend_server/ -p 5001:5001 frontend –service

- Now, we can run curl command to test the catalog microservice separately:
  1. curl http://localhost:5000/search/distributed%20systems



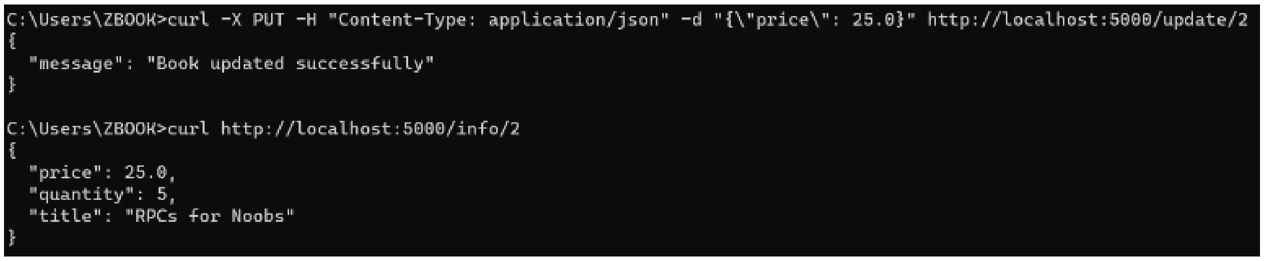
```
C:\Users\ZBOOK>curl http://localhost:5000/search/distributed%20systems
[
  {
    "id": "1",
    "title": "How to get a good grade in DOS in 40 minutes a day"
  },
  {
    "id": "2",
    "title": "RPCs for Noobs"
  }
]
```

  2. curl http://localhost:5000/info/2



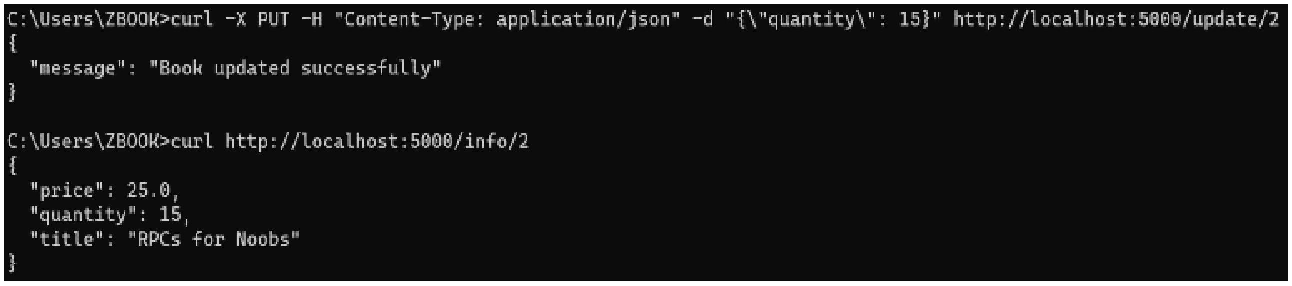
```
C:\Users\ZBOOK>curl http://localhost:5000/info/2
{
  "price": 50.0,
  "quantity": 5,
  "title": "RPCs for Noobs"
}
```

  3. I did three ways for update request:
     a. Update price:
     b. curl -X PUT -H "Content-Type: application/json" -d "{\"price\": 25.0}" http://localhost:5000/update/2



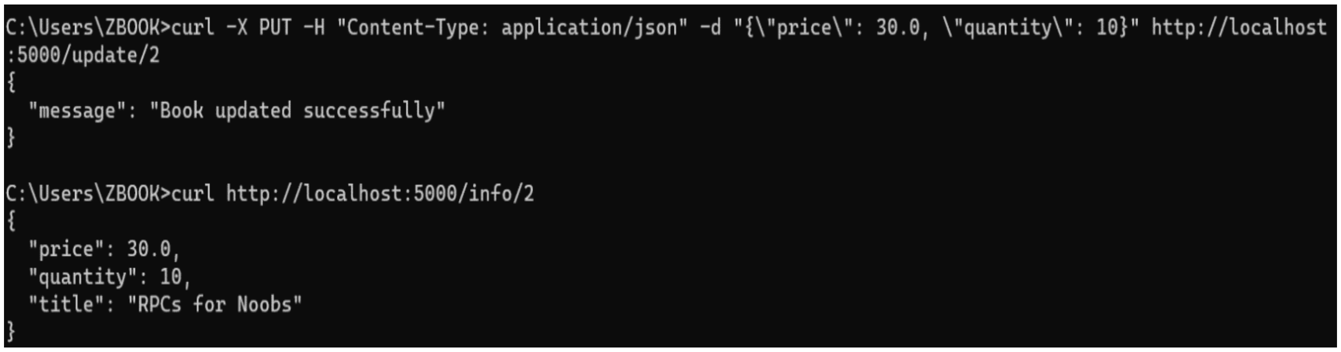
```
C:\Users\ZBOOK>curl -X PUT -H "Content-Type: application/json" -d "{\"price\": 25.0}" http://localhost:5000/update/2
{
  "message": "Book updated successfully"
}

C:\Users\ZBOOK>curl http://localhost:5000/info/2
{
  "price": 25.0,
  "quantity": 5,
  "title": "RPCs for Noobs"
}
```

     c. Update quantity:
        curl -X PUT -H "Content-Type: application/json" -d "{\"quantity\": 15}" http://localhost:5000/update/2



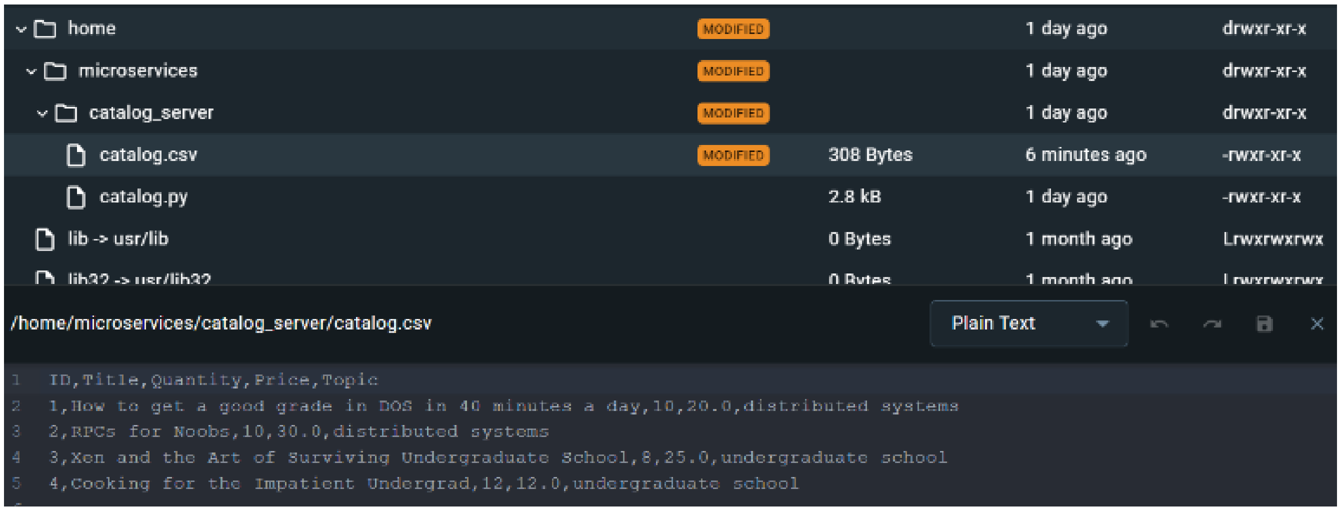
```
C:\Users\ZBOOK>curl -X PUT -H "Content-Type: application/json" -d "{\"quantity\": 15}" http://localhost:5000/update/2
{
  "message": "Book updated successfully"
}

C:\Users\ZBOOK>curl http://localhost:5000/info/2
{
  "price": 25.0,
  "quantity": 15,
  "title": "RPCs for Noobs"
}
```

d. Update price and quantity:
curl -X PUT -H "Content-Type: application/json" -d "{\"price\": 25.0, \"quantity\": 15}" http://localhost:5000/update/2



```
C:\Users\ZBOOK>curl -X PUT -H "Content-Type: application/json" -d "{\"price\": 30.0, \"quantity\": 10}" http://localhost
:5000/update/2
{
  "message": "Book updated successfully"
}

C:\Users\ZBOOK>curl http://localhost:5000/info/2
{
  "price": 30.0,
  "quantity": 10,
  "title": "RPCs for Noobs"
}
```

We can see the changes in the csv file:

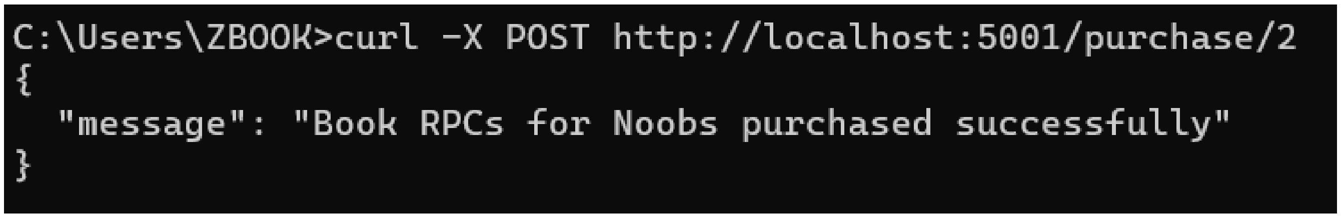


```
home                MODIFIED                    1 day ago       drwxr-xr-x
  microservices     MODIFIED                    1 day ago       drwxr-xr-x
    catalog_server  MODIFIED                    1 day ago       drwxr-xr-x
      catalog.csv   MODIFIED   308 Bytes        6 minutes ago   -rwxr-xr-x
      catalog.py               2.8 kB           1 day ago       -rwxr-xr-x
lib -> usr/lib                 0 Bytes          1 month ago     Lrwxrwxrwx
lib32 -> usr/lib32             0 Bytes          1 month ago     Lrwxrwxrwx
```

/home/microservices/catalog_server/catalog.csv   Plain Text

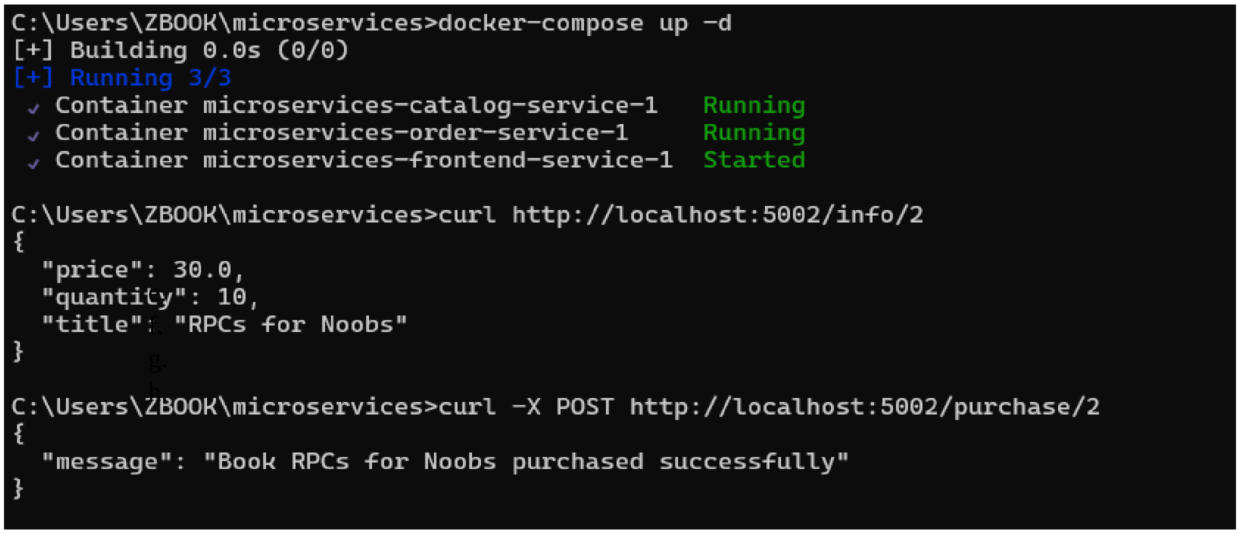
```
ID,Title,Quantity,Price,Topic
1,How to get a good grade in DOS in 40 minutes a day,10,20.0,distributed systems
2,RPCs for Noobs,10,30.0,distributed systems
3,Xen and the Art of Surviving Undergraduate School,8,25.0,undergraduate school
4,Cooking for the Impatient Undergrad,12,12.0,undergraduate school
```

- Testing order microservice:
  curl -X POST http://localhost:5001/purchase/2

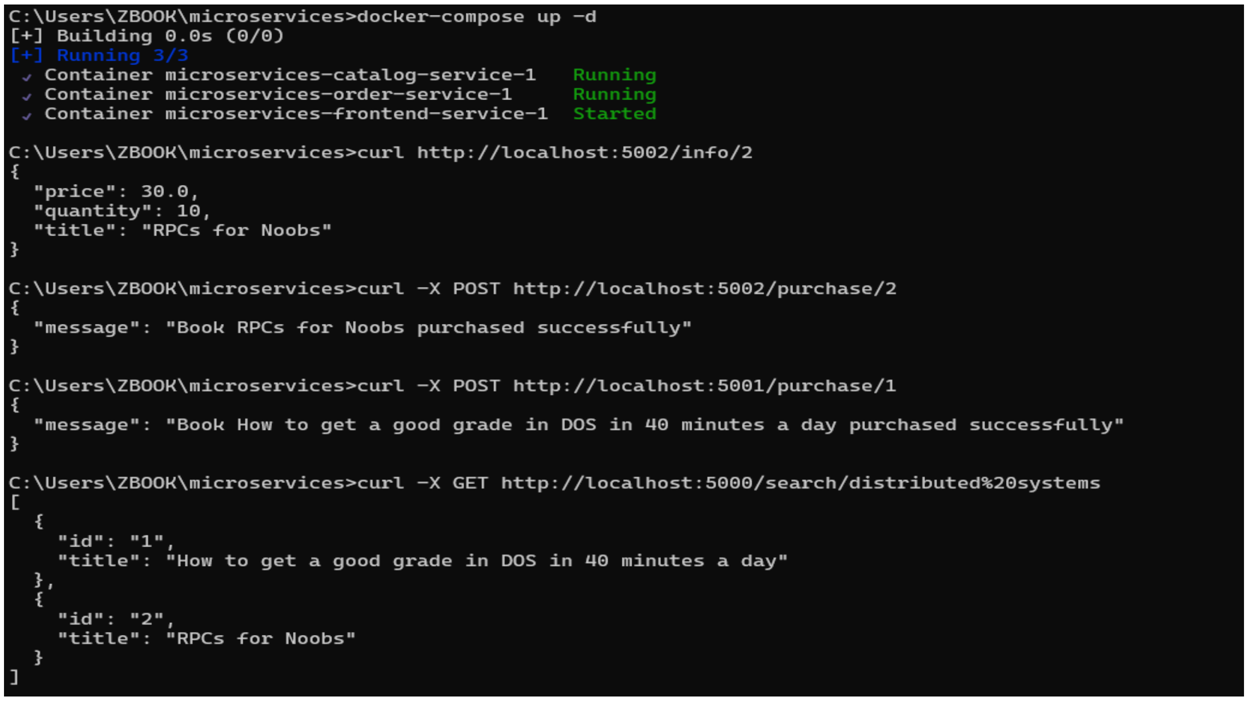


```
C:\Users\ZBOOK>curl -X POST http://localhost:5001/purchase/2
{
  "message": "Book RPCs for Noobs purchased successfully"
}
```

Instead of running each microservice separately, I tried running all the microservices by docker-compose.yaml file, the result is below, as you see, the servers work correctly.

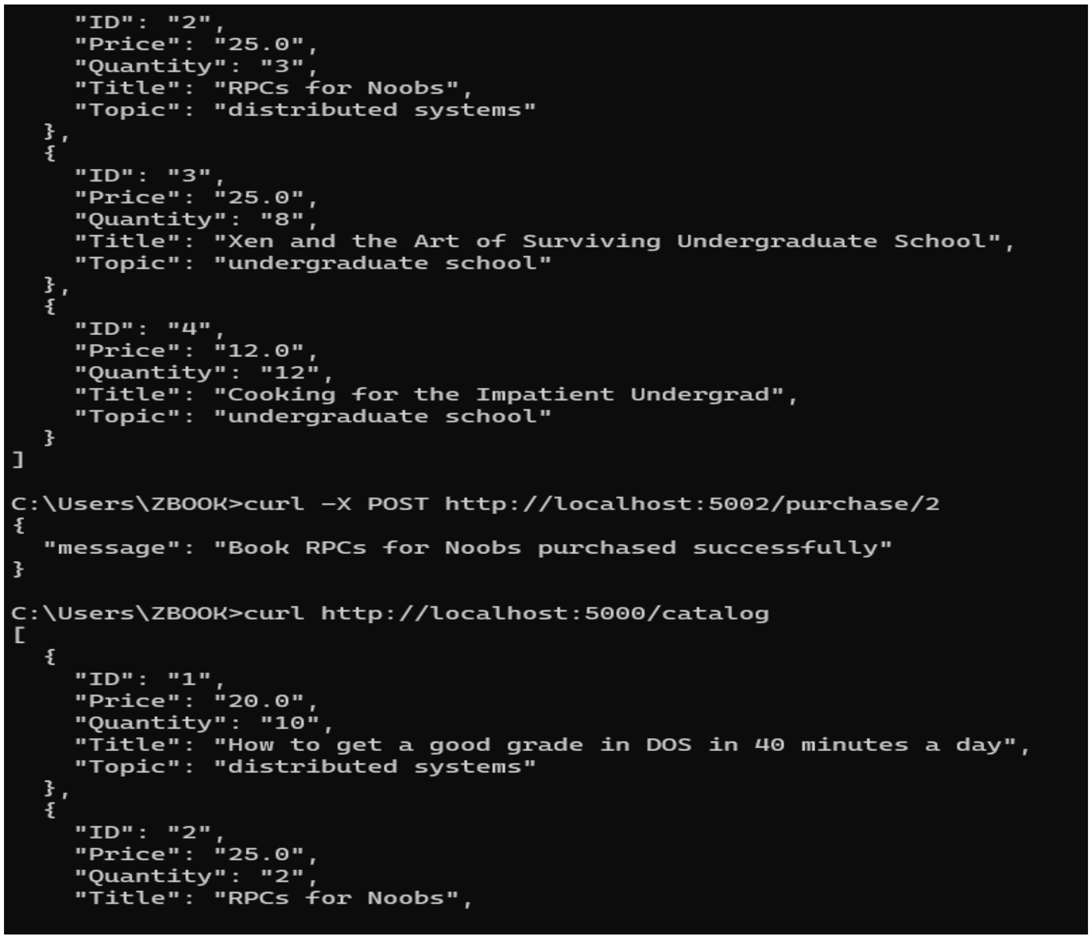


```
C:\Users\ZBOOK\microservices>docker-compose up -d
[+] Building 0.0s (0/0)
[+] Running 3/3
 ✓ Container microservices-catalog-service-1   Running
 ✓ Container microservices-order-service-1     Running
 ✓ Container microservices-frontend-service-1  Started

C:\Users\ZBOOK\microservices>curl http://localhost:5002/info/2
{
  "price": 30.0,
  "quantity": 10,
  "title": "RPCs for Noobs"
}

C:\Users\ZBOOK\microservices>curl -X POST http://localhost:5002/purchase/2
{
  "message": "Book RPCs for Noobs purchased successfully"
}
```



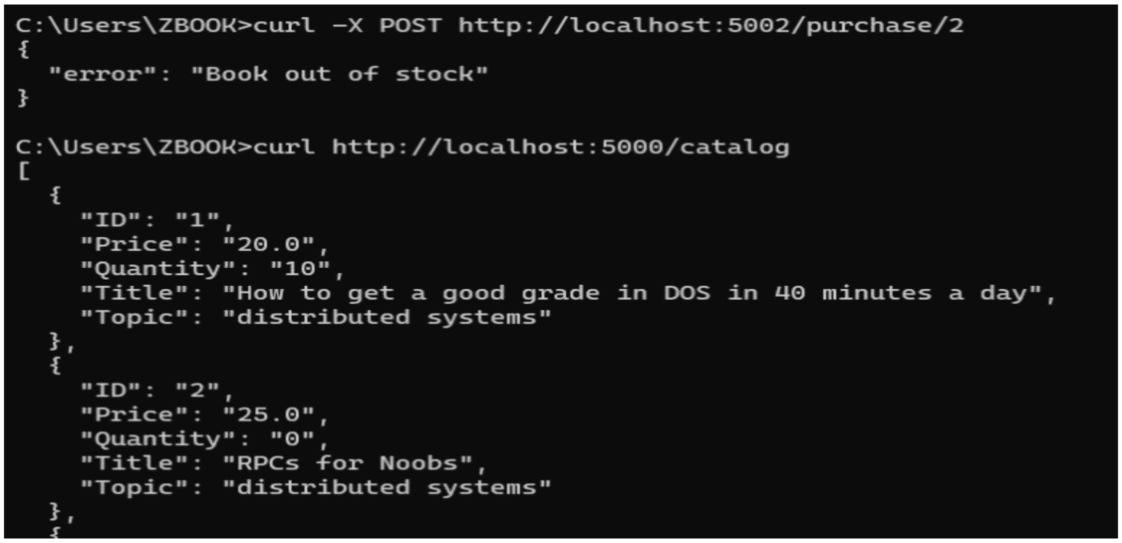
```
C:\Users\ZBOOK\microservices>docker-compose up -d
[+] Building 0.0s (0/0)
[+] Running 3/3
 ✓ Container microservices-catalog-service-1   Running
 ✓ Container microservices-order-service-1     Running
 ✓ Container microservices-frontend-service-1  Started

C:\Users\ZBOOK\microservices>curl http://localhost:5002/info/2
{
  "price": 30.0,
  "quantity": 10,
  "title": "RPCs for Noobs"
}

C:\Users\ZBOOK\microservices>curl -X POST http://localhost:5002/purchase/2
{
  "message": "Book RPCs for Noobs purchased successfully"
}

C:\Users\ZBOOK\microservices>curl -X POST http://localhost:5001/purchase/1
{
  "message": "Book How to get a good grade in DOS in 40 minutes a day purchased successfully"
}

C:\Users\ZBOOK\microservices>curl -X GET http://localhost:5000/search/distributed%20systems
[
  {
    "id": "1",
    "title": "How to get a good grade in DOS in 40 minutes a day"
  },
  {
    "id": "2",
    "title": "RPCs for Noobs"
  }
]
```

Now let's try buying the purchase through frontend service, let's check the quantity before and after the purchase. As you see the quantity of the item with id 2 was "3" and became "2" after purchase.



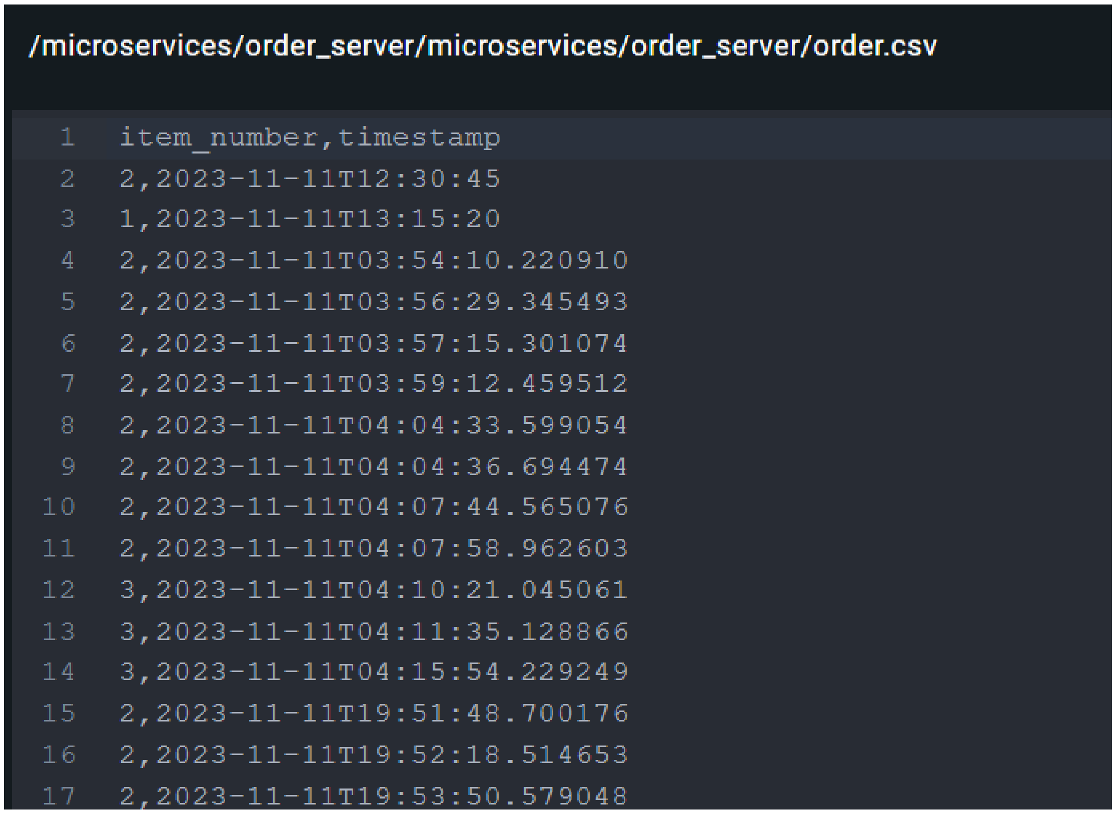
```
    "ID": "2",
    "Price": "25.0",
    "Quantity": "3",
    "Title": "RPCs for Noobs",
    "Topic": "distributed systems"
  },
  {
    "ID": "3",
    "Price": "25.0",
    "Quantity": "8",
    "Title": "Xen and the Art of Surviving Undergraduate School",
    "Topic": "undergraduate school"
  },
  {
    "ID": "4",
    "Price": "12.0",
    "Quantity": "12",
    "Title": "Cooking for the Impatient Undergrad",
    "Topic": "undergraduate school"
  }
]

C:\Users\ZBOOK>curl -X POST http://localhost:5002/purchase/2
{
  "message": "Book RPCs for Noobs purchased successfully"
}

C:\Users\ZBOOK>curl http://localhost:5000/catalog
[
  {
    "ID": "1",
    "Price": "20.0",
    "Quantity": "10",
    "Title": "How to get a good grade in DOS in 40 minutes a day",
    "Topic": "distributed systems"
  },
  {
    "ID": "2",
    "Price": "25.0",
    "Quantity": "2",
    "Title": "RPCs for Noobs",
```

Also, when you try to buy an item that out of stock, this is what you see:



```
C:\Users\ZBOOK>curl -X POST http://localhost:5002/purchase/2
{
  "error": "Book out of stock"
}

C:\Users\ZBOOK>curl http://localhost:5000/catalog
[
  {
    "ID": "1",
    "Price": "20.0",
    "Quantity": "10",
    "Title": "How to get a good grade in DOS in 40 minutes a day",
    "Topic": "distributed systems"
  },
  {
    "ID": "2",
    "Price": "25.0",
    "Quantity": "0",
    "Title": "RPCs for Noobs",
    "Topic": "distributed systems"
  },
```

Let’s see the content of order.csv file:



/microservices/order_server/microservices/order_server/order.csv

```
item_number,timestamp
2,2023-11-11T12:30:45
1,2023-11-11T13:15:20
2,2023-11-11T03:54:10.220910
2,2023-11-11T03:56:29.345493
2,2023-11-11T03:57:15.301074
2,2023-11-11T03:59:12.459512
2,2023-11-11T04:04:33.599054
2,2023-11-11T04:04:36.694474
2,2023-11-11T04:07:44.565076
2,2023-11-11T04:07:58.962603
3,2023-11-11T04:10:21.045061
3,2023-11-11T04:11:35.128866
3,2023-11-11T04:15:54.229249
2,2023-11-11T19:51:48.700176
2,2023-11-11T19:52:18.514653
2,2023-11-11T19:53:50.579048
```

This file is updated at every purchase, as you see the item number and the timestamp the purchase is store.