An-Najah National University

Faculty of Engineering and IT

جامعة النجاح الوطنية

كلية الهندسة وتكنولوجيا المعلومات

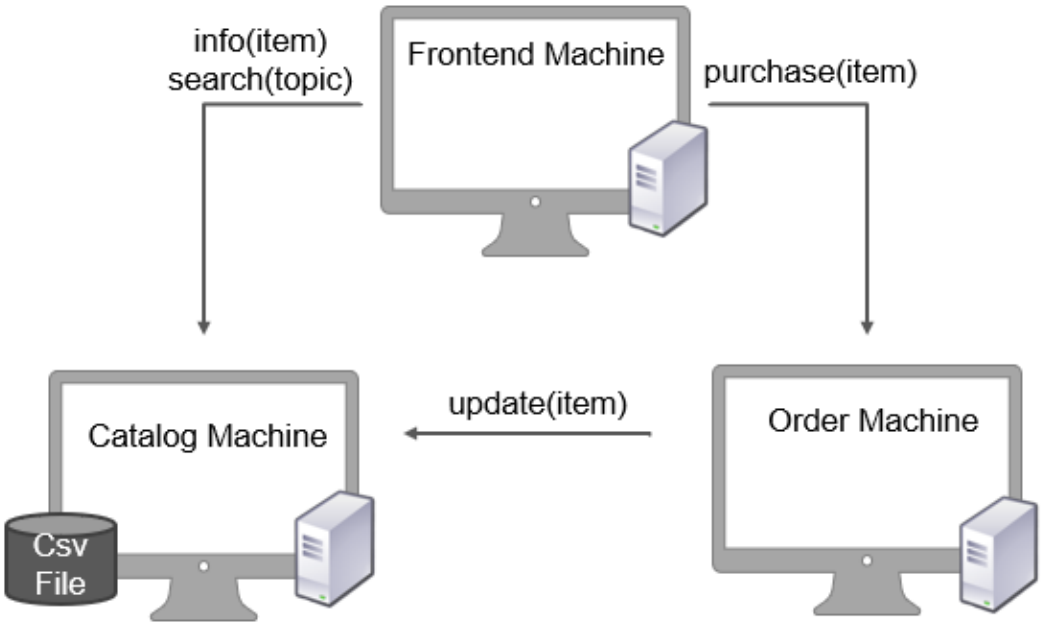
Computer Engineering Department	
Course Name: Distributed Operating Systems	Number: 10636456/1394
Report	
Academic Year: 2022/2023	Semester: 1st
Homework 2	
Topic: Microservices (Bazar.com - A Multi-tier Online Book Store)	
Instructor Name:	
Dr. Samer Arandi	
Student:	
Name: Gharam Jamil Sarsour	**ID: 11819668**

Testing the API using different methods

- **How does it work?**
  1- It started with a simple python API for the catalog on the host operating system, because it is the server that both the order and the frontend send requests to, then a csv file was created to store the data and used the catalog server to manipulate the data inside it, after that the server was run on the Ip of the device and tested to make sure there was no errors.
  2- Next the order server was created on a virtual machine the same way, and it was tested on the device Ip, then the redirect for the catalog server was added and tested by putting the Ip and port of the catalog server (host os).
  3- Finally, it was time to create the frontend server, which was done on a second virtual machine, and after adding the redirecting for the catalog with its Ip and port, and adding the redirecting for the order with its Ip and port, the front end was tested using postman API platform, all virtual machines was using bridge network adapter which allowed for the http requests to be directed from one server to another, and ending the chain of request with a response to the frontend server.
  In Summary: the frontend API receives http requests from the user then redirects the requests to the catalog and order servers, if the request is search or info, it sends a query request to the catalog server with the parameters it has, and it takes the response from the catalog and sends it as a response to the user, if the request is a purchase, the frontend requests a purchase from the order with the item number as a parameter, the order request takes the item number and requests an update request to the catalog to decrease the stock, then it takes the response, sends it to the frontend, and the frontend sends it to the user.

- **Design Trade off!**
  The application is relatively small witch means there is more complexity than needed, at the same time if we want to scall some part of the application and update, there is no need to shut all the services to update it, only the needed one.
- **Possible Improvement?**
  Adding an operation to edit the prices and stock of the catalog would be useful to make the book store more efficient.

- **How to run the program?**
  1- First, we need three machines or rather 3 operating systems, we need to ensure that they are all running on a bridge network adapter, we also need to have flask working successfully on all the machines.
  2- Then, we put each of the codes in a machine, then we get the Ip of each machine and use them in the right codes, we put the catalog Ip in both frontend and order servers, we put the order Ip in the frontend server.
  3- After we save the codes with the new Ips, we run each server with the command (flask run -h server_ip -p server_port).
  4- Finally we have all the servers working and we can start sending requests to the frontend server and get responses in return.
- **The Design**



- **The output**

the csv file with data:

app.py data.csv

catalog > data.csv

```
item_number,title,topic,stock,cost
1,How to get a good grade in DOS in 40 minutes a day,distributed systems,20,35
2,RPCs for Noobs,distributed systems,30,30
3,Xen and the Art of Surviving Undergraduate School,undergraduate school,44,50
4,Cooking for the Impatient Undergrad,undergraduate school, 25,30
```

the catalog server running on 192.168.1.101 and port 5000:

app.py data.csv

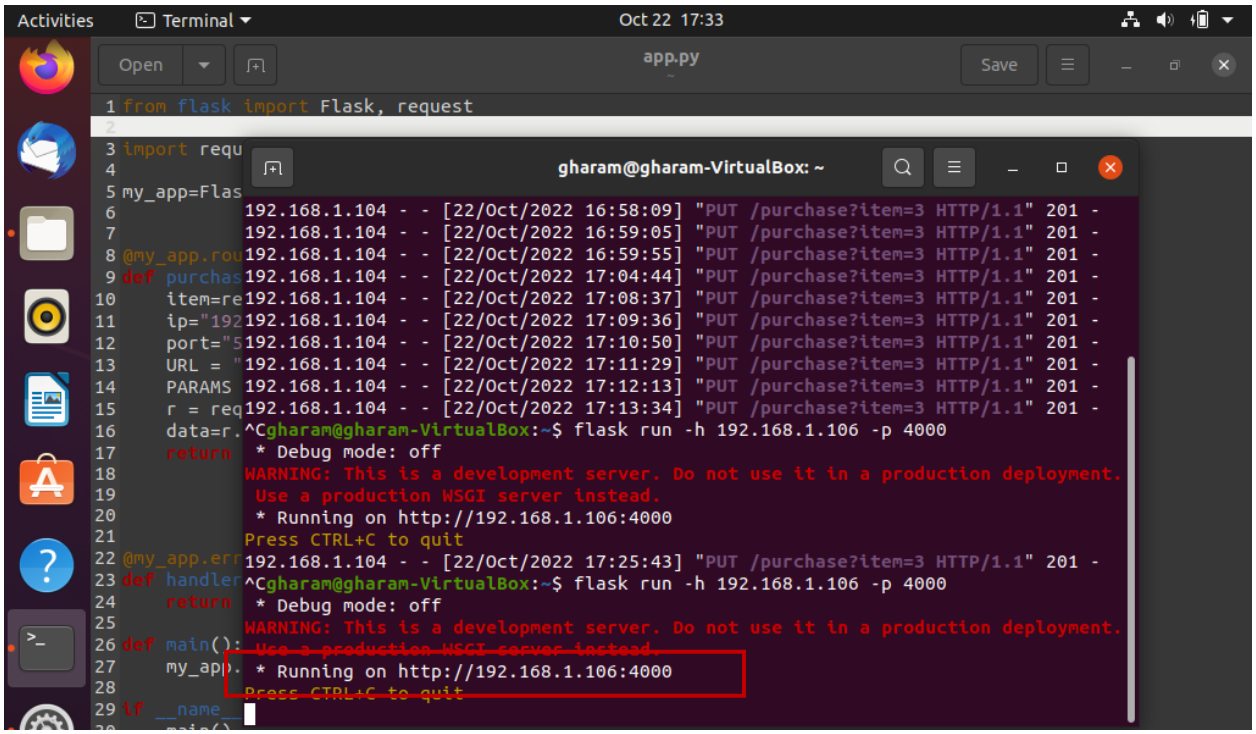
catalog > app.py > update

```
        header = next(csvreader)
        for row in csvreader:
            rows.append(row)
    for row in rows:
        print(row)
    for row in rows:
        if row[0] == item:
            title=row[1]
            stock=row[3]
```

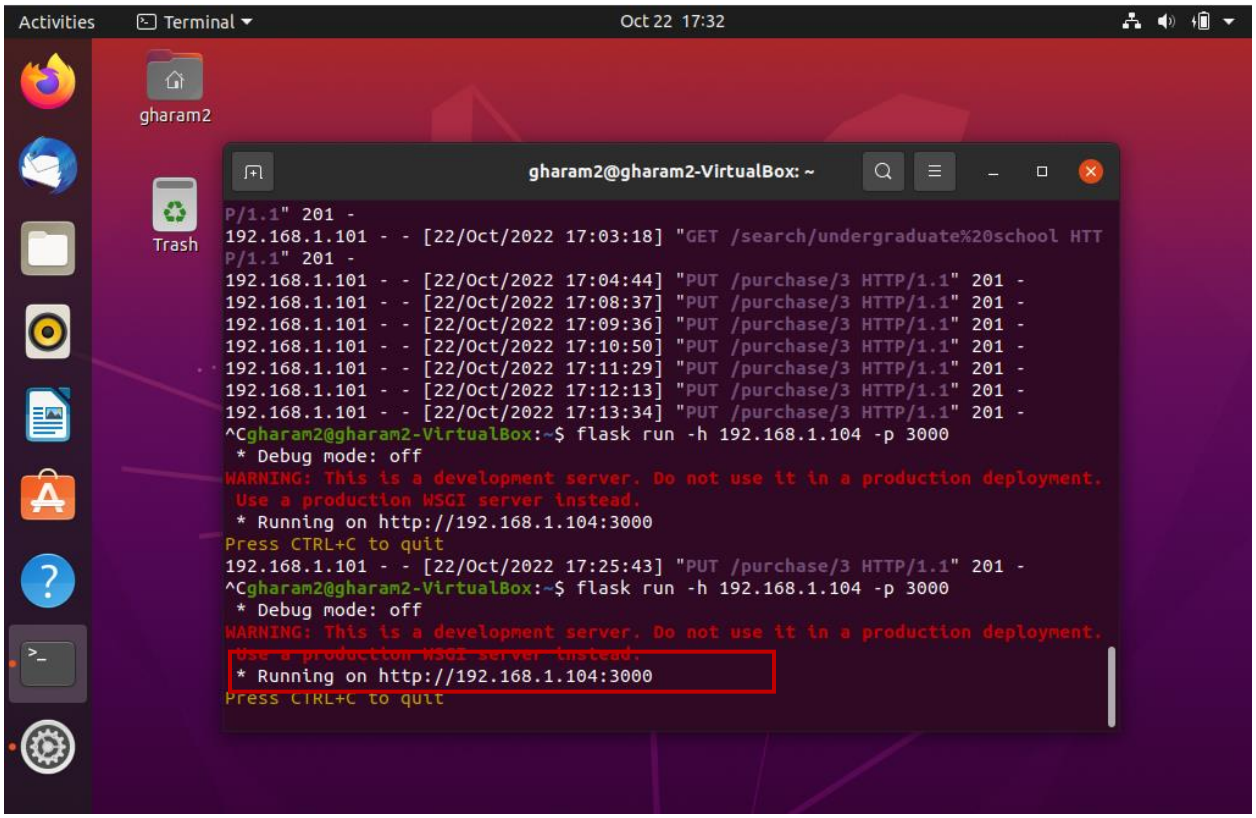
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
['4', 'Cooking for the Impatient Undergrad', 'undergraduate school', ' 25', '30']
45
['item_number', 'title', 'topic', 'stock', 'cost']
[['1', 'How to get a good grade in DOS in 40 minutes a day', 'distributed systems', '20', '35'], ['2', 'RPCs for Noobs', 'dist
ributed systems', '30', '30'], ['3', 'Xen and the Art of Surviving Undergraduate School', 'undergraduate school', 45, '50'],
'4', 'Cooking for the Impatient Undergrad', 'undergraduate school', ' 25', '30']]
192.168.1.106 - - [22/Oct/2022 17:25:43] "PUT /update?item=3 HTTP/1.1" 201 -
PS C:\Users\Asus\Desktop\catalog> flask run -h 192.168.1.101 -p 5000
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://192.168.1.101:5000/ (Press CTRL+C to quit)
```

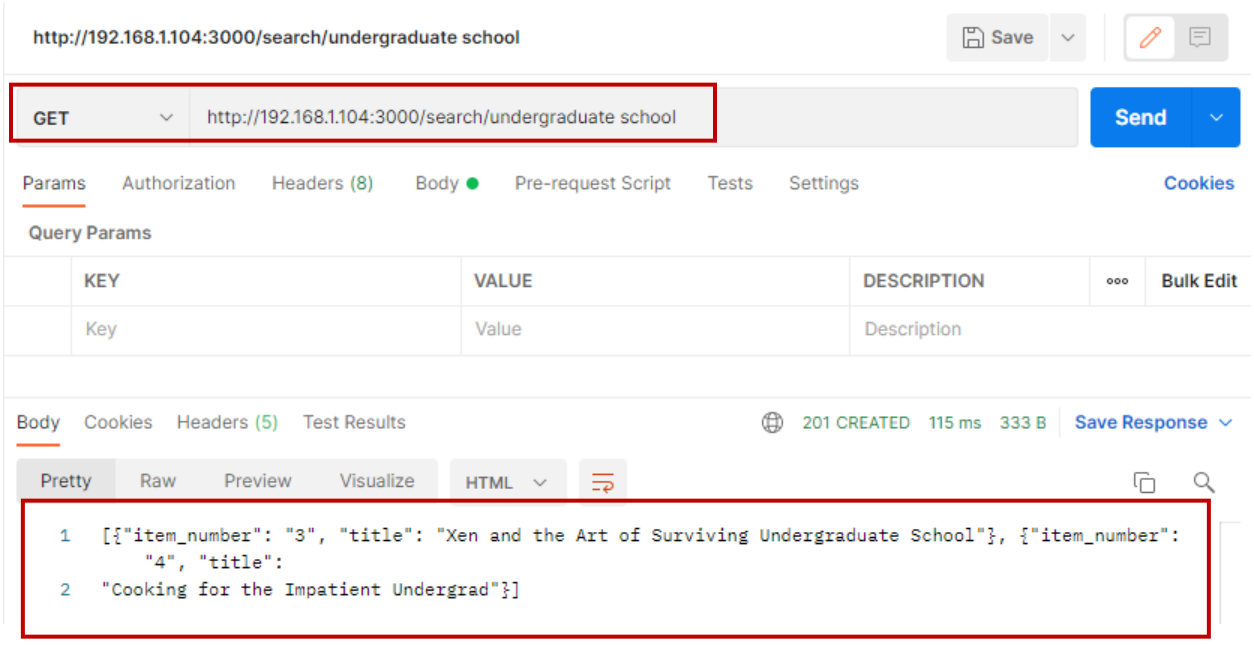
the order server running on a different machine with ip 192.168.1.106

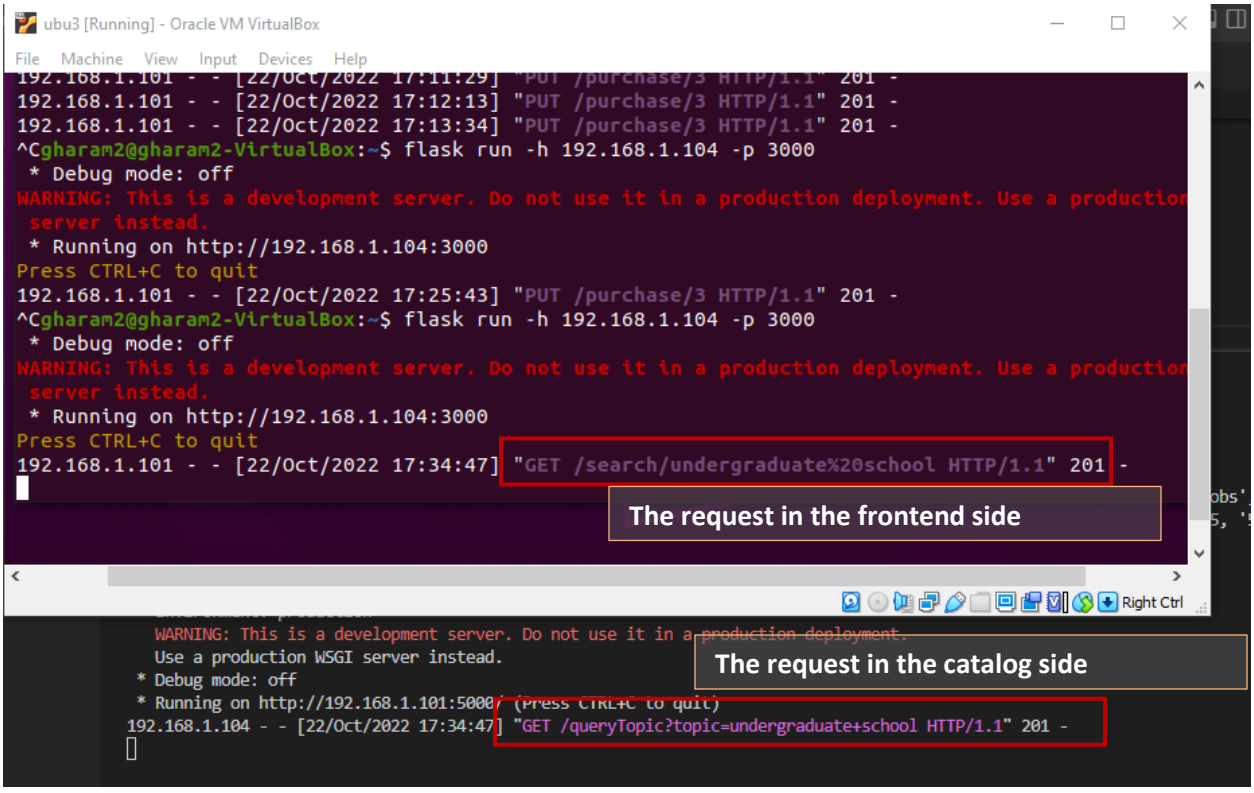


the front end server running on a different machine with ip 192.168.1.104

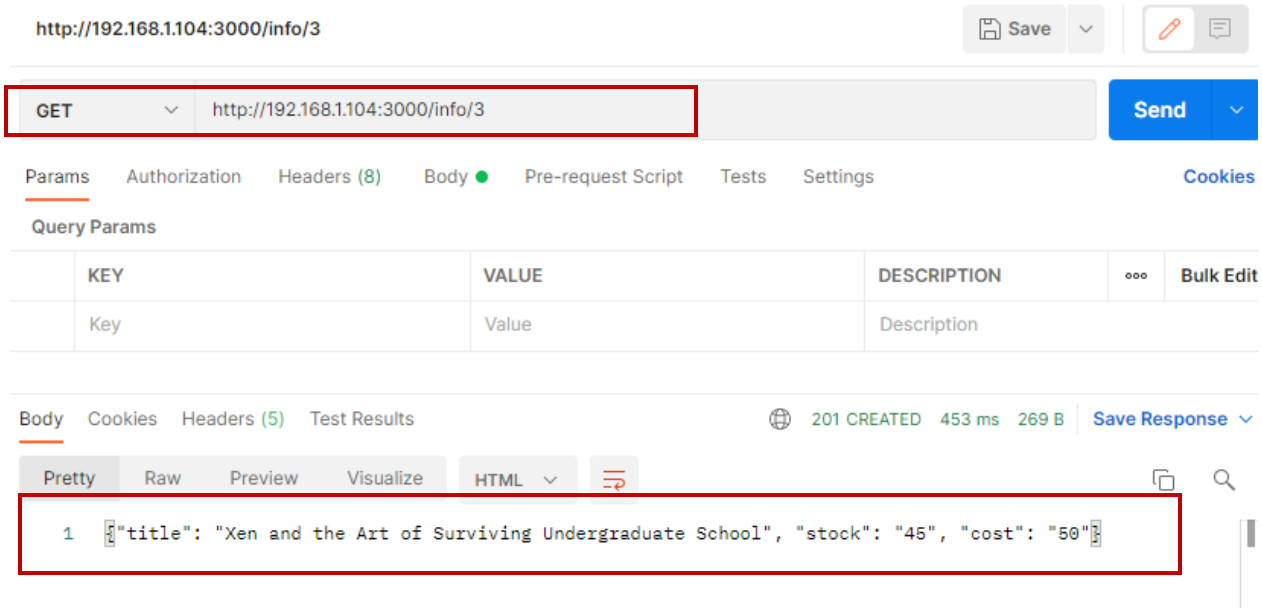


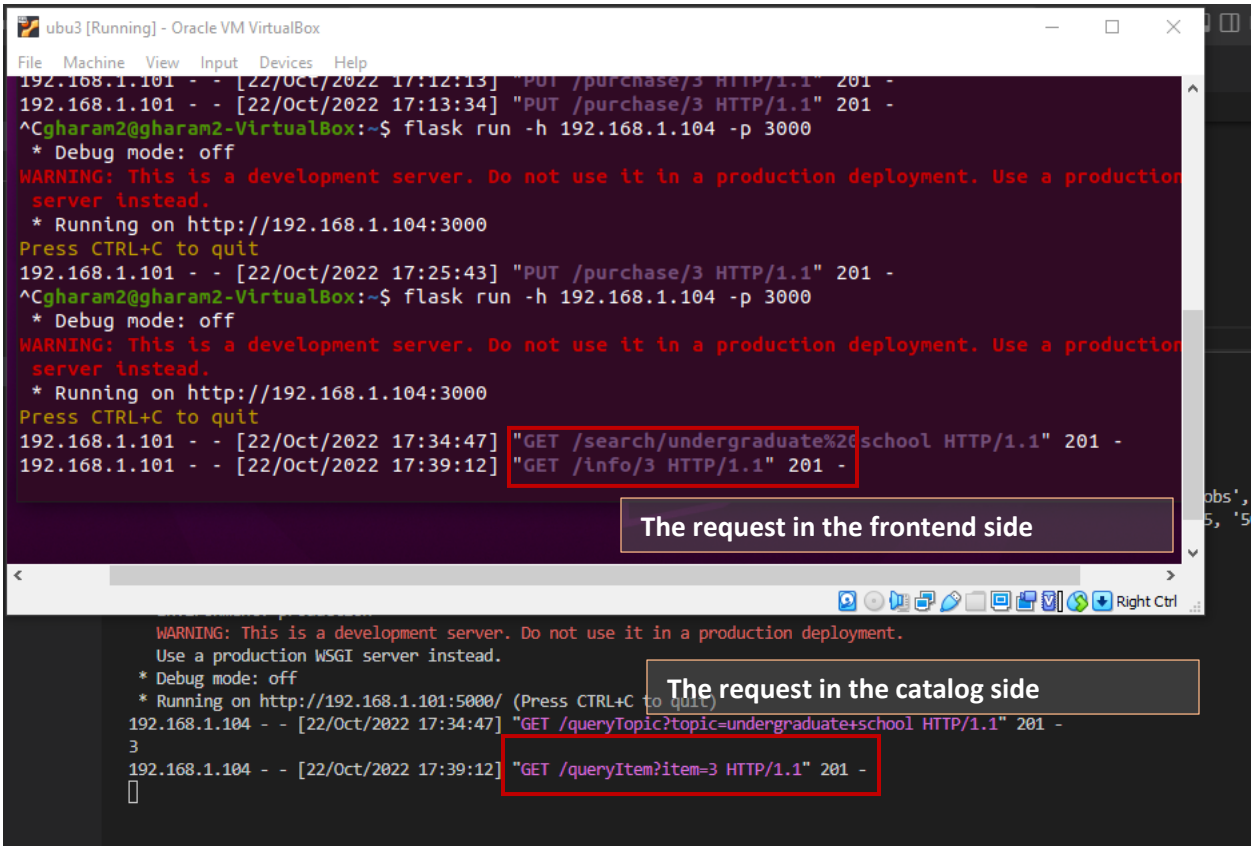
sending a search request to the frontend server using postman



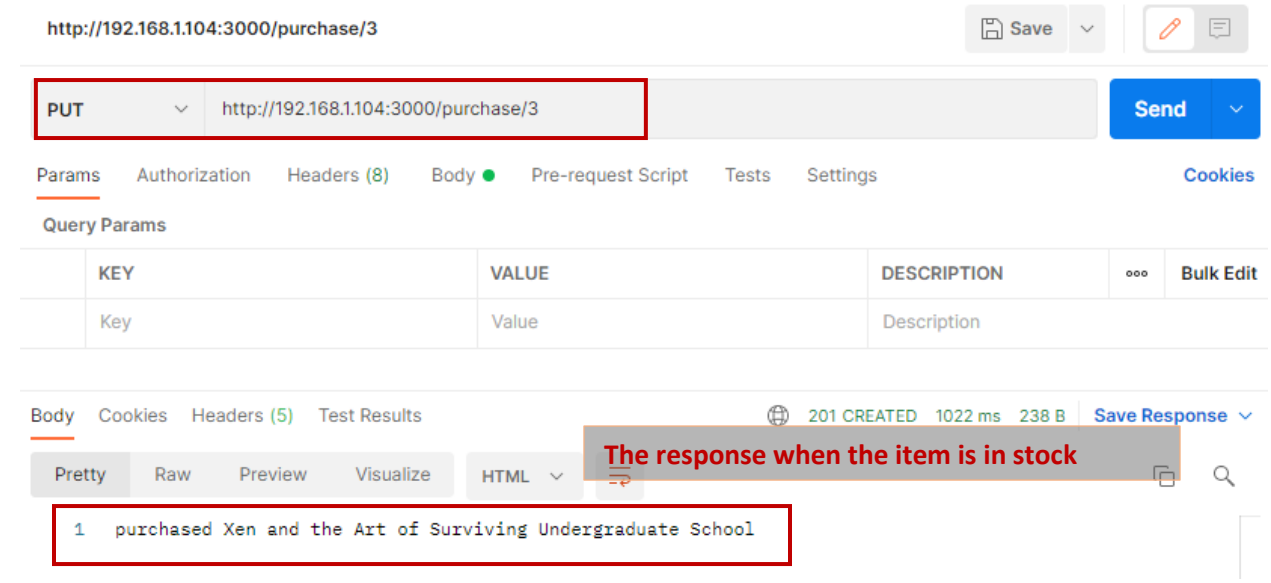


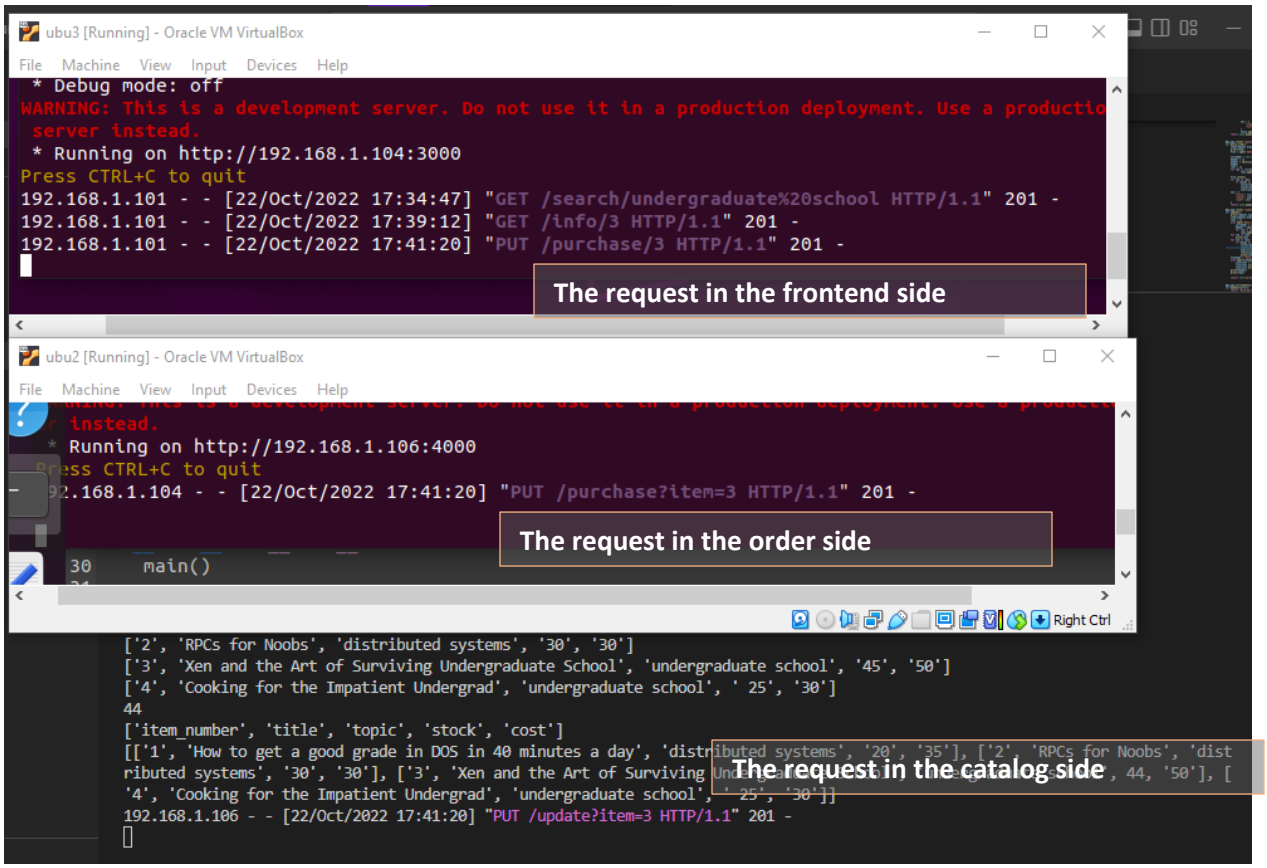
sending an info to the frontend server using postman

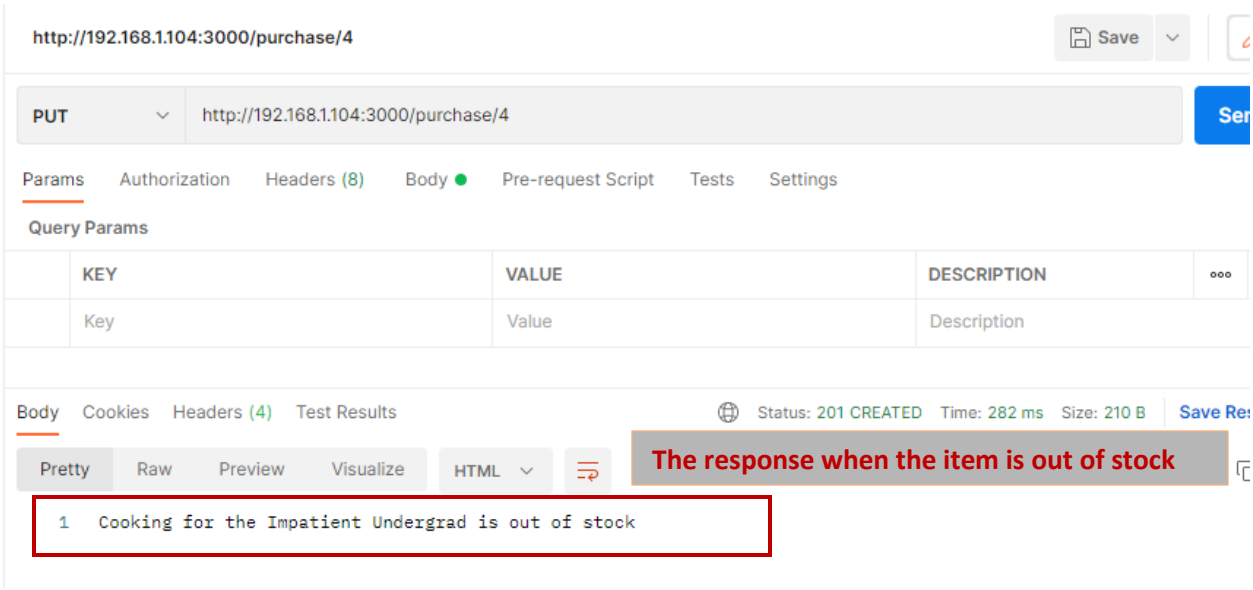




sending a purchase request to the frontend server using postman







The response when the item is out of stock