

Computer Engineering Department

DOS

Bazar.com

Students:

Sahar Saleh 11821693

Marah Al-Najjar 11819426

Dr. Samer Arandi

Introduction:

In this project we used python language, and flask framework, for the database we use a JSON file.

Prepare two Ubuntu virtual machine with flask project and use visual studio code as editor.

Write the catalog server on one machine, order server on other machine, and write the front end at local device.

Code basics:

1. @app.route("/info/<id>",methods=['GET'])

Use this line to set the URL format as required in each operation. In this example info and id change as required in every case. and the method change as the operation requirement.

1. With open('/home/sahar/Desktop/pyox/part1/

BooksDB.json', 'r') as DBfile:

Use this code to open JSON file and create a nick name for the file, such as DBfile, in order to make it easier to handle.

1. data = DBfile.read()

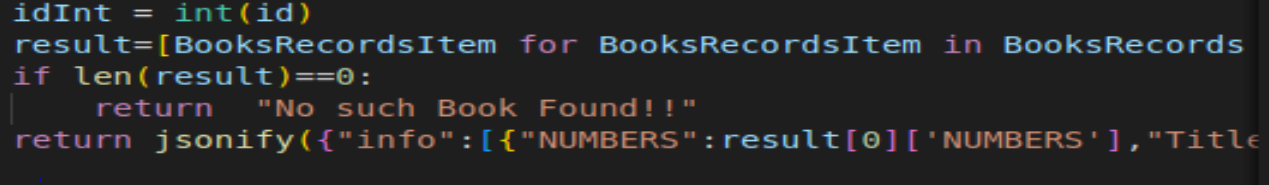
|  |
| --- |
| jsonObject = json.loads(data) |
|  |

Read data from DBfile and convert it to JSON format

1. BooksRecords = jsonObject['BOOK']

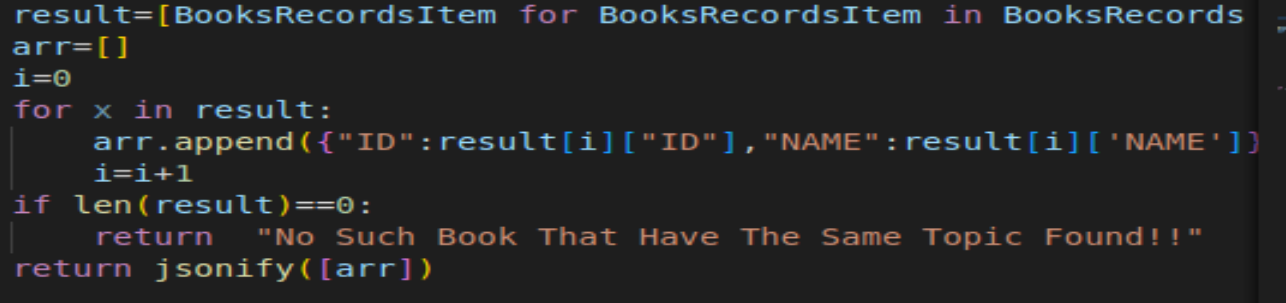
Separate the books database to many record to deal with every record alone, every record contain one book with all its properties.

1. Write the result statement as require in every question:
2. In info question:



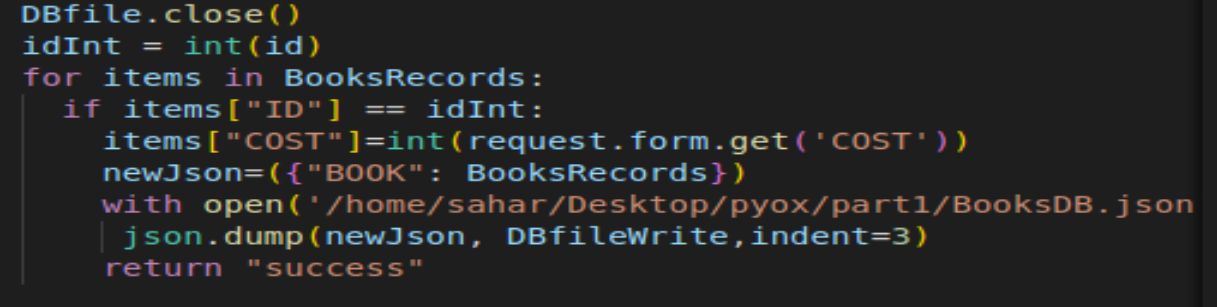
Make a for loop that loops for every book in the database, checks if its ID is the same as the enter ID, then displays the title, cost, and number of copies. The result will be just one book. Because the ID is unique.

1. In search question:



Search according to book topic, find all books that have the same entering topic, append the book name and id into array to allow convert it to JSON format then show all of them. Then check if the array contain elements so we show the information of the book that have the same topic, but if there is no element then it means no book have the same topic.

1. Update question:



When updating, you must close the file after reading data, find the book with the required id, then update the cost, open the file again as a write operation, and dump the updated information.

1. queryNumbers:

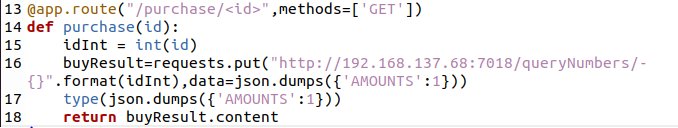
This operation check if the required book (assigned with its ID) is available at data base by check if numbers of book is more than zero, if it’s available decrease the number of books with the enter amount.

1. IncreaseNumbers:

The same as queryNumbers operation but here increase the books number with the enter amounts.

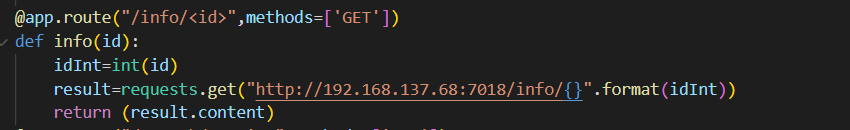
Order server:

Has one operation which is purchase:



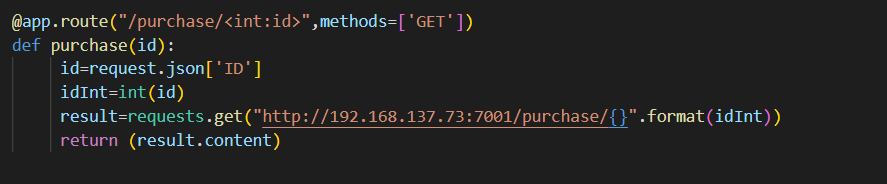
In this function send a request to catalog server to buy the book if it’s available (Numbers of book is greater than one to make sure that you can buy it).

Front end client for catalog server:



For example the info operation, send request to catalog server. Then receive the replay from the server. Same as other operations like search, updateCost, queryNumbers, IncreaseNumbers, all of these operations send a request to catalog server to perform the operation, then receive the replay from the server.

Front end of order server:



In this operation, the front end send a request to order server, then receive the replay from order server to show if the book order or not.

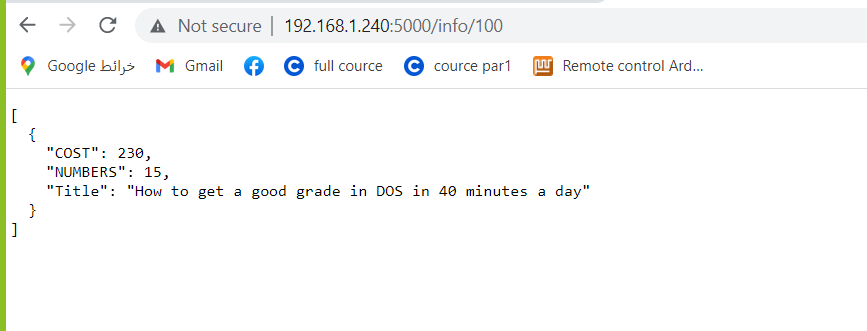
In order to connect two virtual machines, we must detect their IP addresses, then make the connection. Unfortunately, the connection fails even though the code is correct.

Catalog server IP: 192.168.137.68

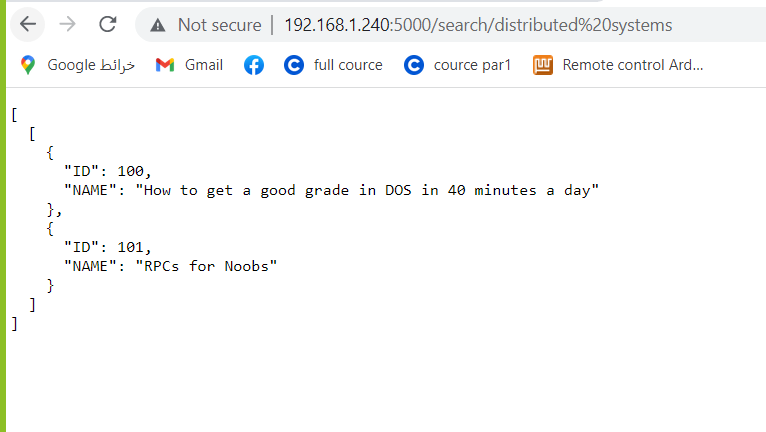
Order server: 192.168.137.73

Here are the results of the three requests :

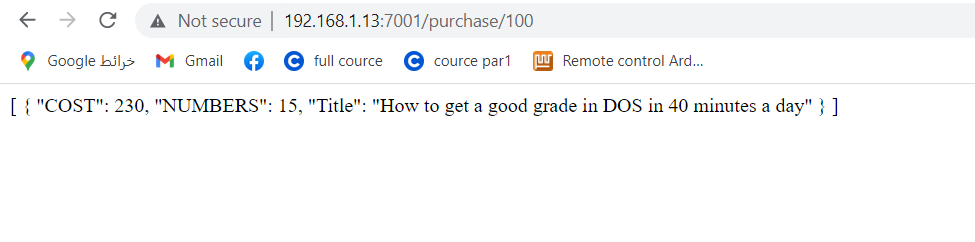
1. info



2-search



3-purchase

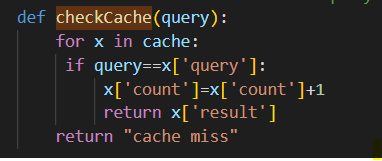


Part 2:

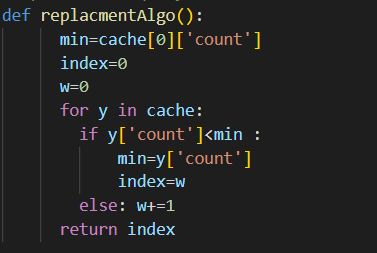
In this part, we added replication and caching to improve request processing latency. While the front-end server in part 1 was a very simple component, in this part, we added two types of functionalities to the front-end node. First, we added an in-memory cache that caches the results of recent requests to the order and catalog servers.

Adjustments and functions we added to our code :

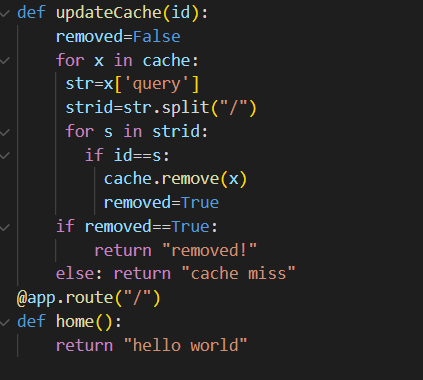
1-this function is to check if the query is inside the cash or not



2-this function is for doing replacement algorithm if the cash is full replace less query used

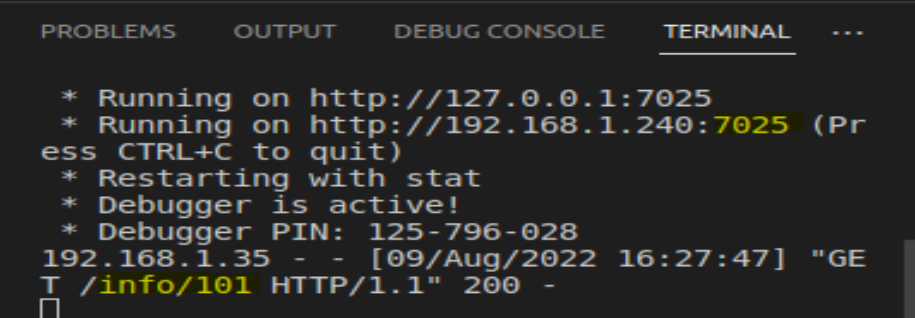
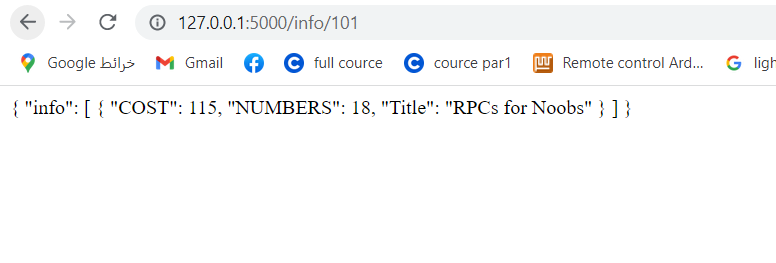
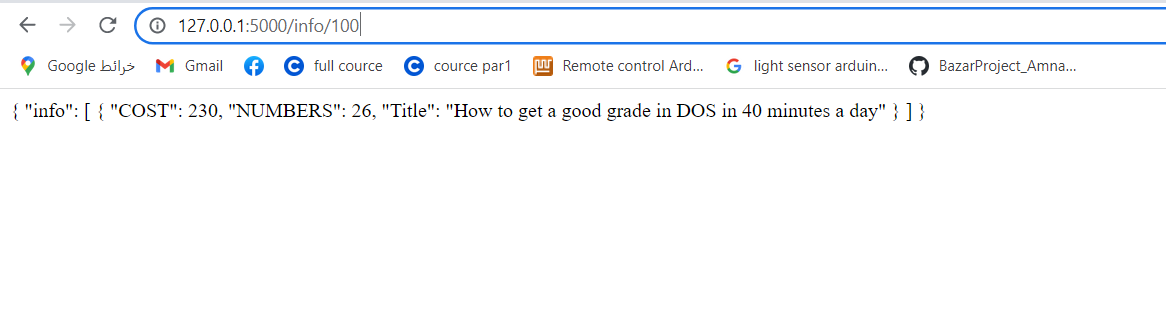


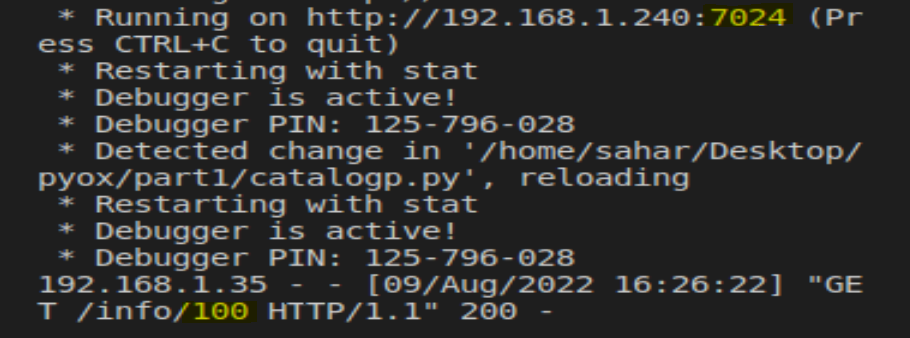
3-this function is to delete certain id from cash when write requests



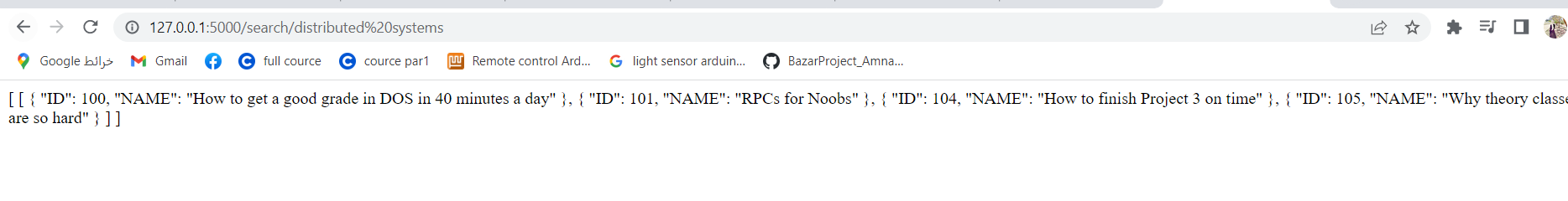
Requests results:

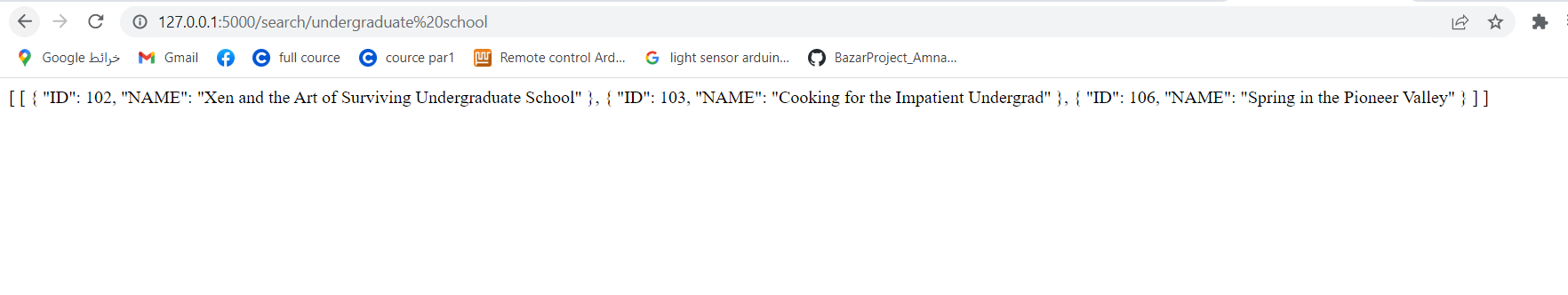
Info /100 and info/101 as shown below every one is made on different catalog serverand this is how we balanced our code .

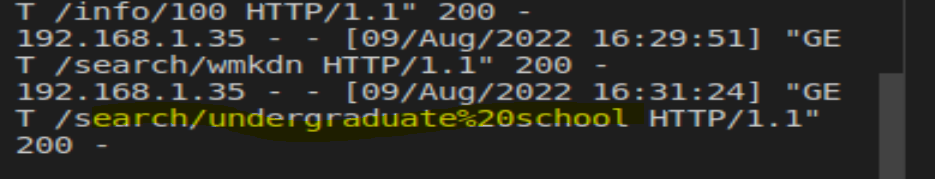


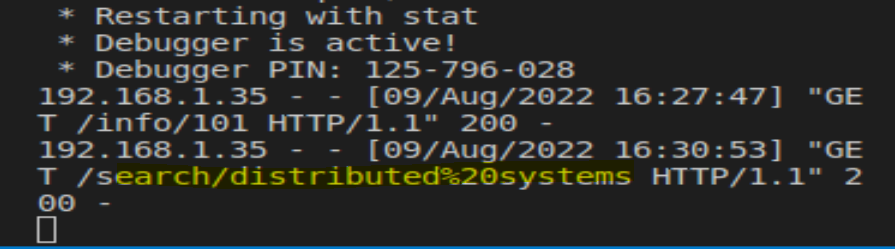


1. search requests is the same as info we made two requests every one on a different server

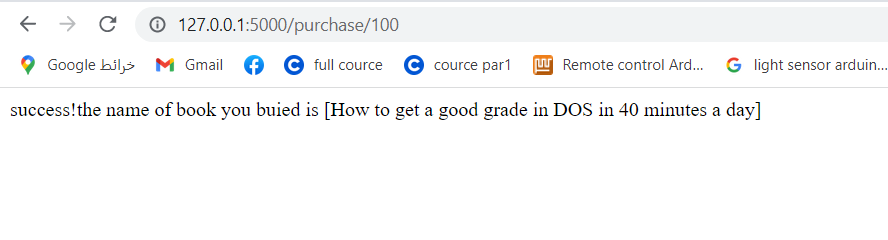


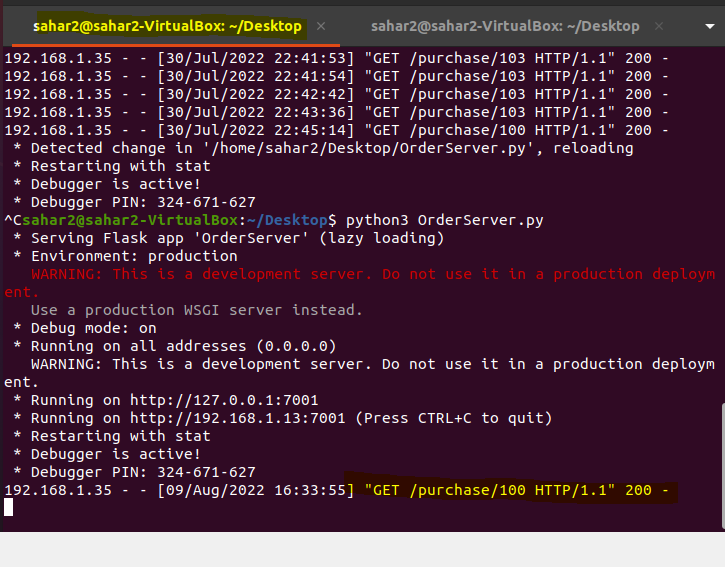
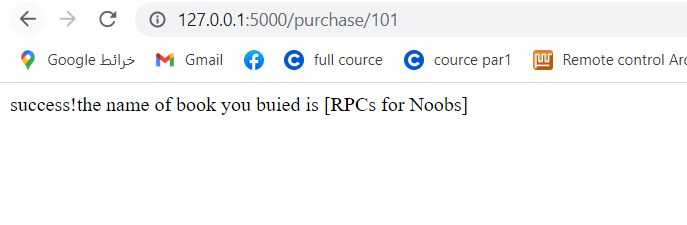


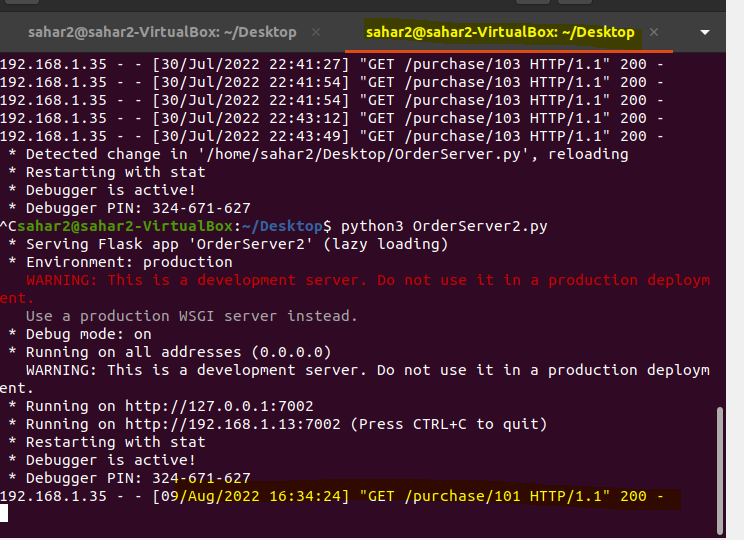


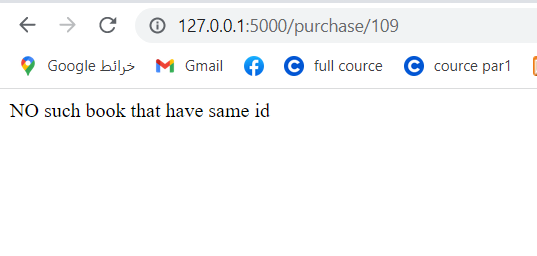


3-purchase: same principle as above









Update cost :

