An-Najah National University

Faculty of Engineering and Information Technology

Distributed Operation Systems – 10636456

Lab #2

Turning the Bazar into an Amazon

Replication, Caching and Consistency

Students names:

1- **Abdallah Omar Adas 11924993**

2- Obaida Aws 11923787

In this part of the previous project, we add two main enhancements:

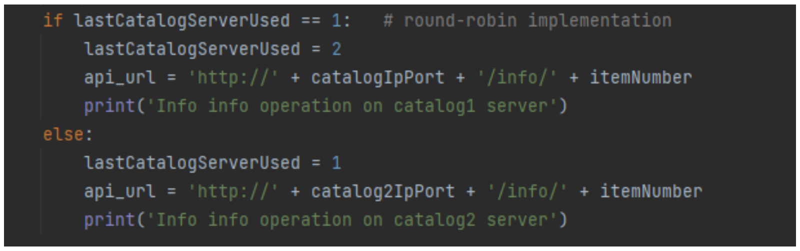
1- **Implement two caches:**
   a. **Search for book by item number:**

      here we implement cache that stores a list for the resent searched books, and we implement a cache consistency by delete the book from the cache when add a book to the stock, and when purchase a book, then the book will return to the cache in the next search on the book.

   b. **Search by topic name:**

      Here we implement cache that store the books under the specific topic name.

2- **Replicate the order server and the catalog server.**
   a- **Here we copy the order and catalog server, and we copy the database, so we have 2 orders servers and 2 catalog servers and 2 databases.**
   b- **we have maintained the databases consistency by apply the purchasing and book adding process on the two catalog servers.**
   c- **We used the round-robin balancing algorithm to distributing the request load to all server copies, and the load balancing is on the per-request basis, so in the first time the request will directed to the server1 and the next time will be directed to the second server (catalog and order).**



```
if lastCatalogServerUsed == 1:   # round-robin implementation
    lastCatalogServerUsed = 2
    api_url = 'http://' + catalogIpPort + '/info/' + itemNumber
    print('Info info operation on catalog1 server')
else:
    lastCatalogServerUsed = 1
    api_url = 'http://' + catalog2IpPort + '/info/' + itemNumber
    print('Info info operation on catalog2 server')
```

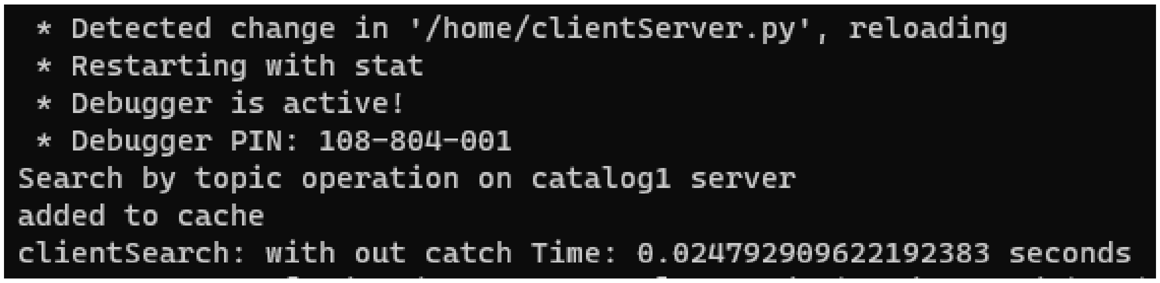
Above is an example of use round-robin load balancing algorithm, in this code each request to the catalog servers will be directed to one of them, if in the first request the catalog1 serve this request, in the second time the request will directed to catalog2 and so on.

Screenshots for the output:

1- **Cache**

a- **Topic search:**

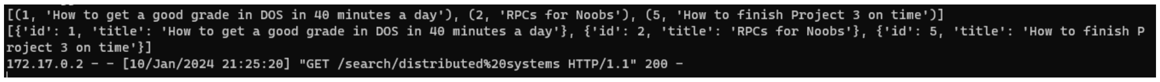
clientServer.py



```
 * Detected change in '/home/clientServer.py', reloading
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 108-804-001
Search by topic operation on catalog1 server
added to cache
clientSearch: with out catch Time: 0.024792909622192383 seconds
```

Here the data (distributed systems) is not in the cache so the client server direct the request to the catalog1 server.

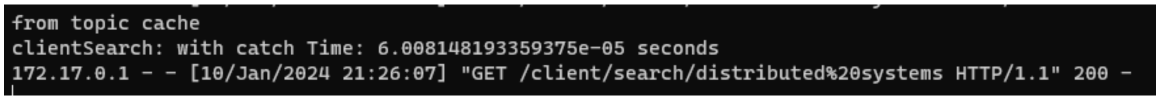
Catalog1Server.py



```
[(1, 'How to get a good grade in DOS in 40 minutes a day'), (2, 'RPCs for Noobs'), (5, 'How to finish Project 3 on time')]
[{'id': 1, 'title': 'How to get a good grade in DOS in 40 minutes a day'}, {'id': 2, 'title': 'RPCs for Noobs'}, {'id': 5, 'title': 'How to finish P
roject 3 on time'}]
172.17.0.2 - - [10/Jan/2024 21:25:20] "GET /search/distributed%20systems HTTP/1.1" 200 -
```

Above is the Catalog1 server output

clientServer.py

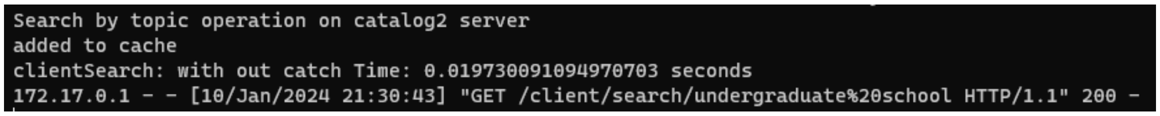


```
from topic cache
clientSearch: with catch Time: 6.008148193359375e-05 seconds
172.17.0.1 - - [10/Jan/2024 21:26:07] "GET /client/search/distributed%20systems HTTP/1.1" 200 -
```

Here the data in the cache so the client server return it from cache.

---

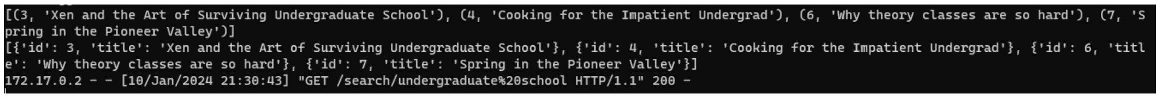
clientServer.py



```
Search by topic operation on catalog2 server
added to cache
clientSearch: with out catch Time: 0.019730091094970703 seconds
172.17.0.1 - - [10/Jan/2024 21:30:43] "GET /client/search/undergraduate%20school HTTP/1.1" 200 -
```

Here the data (undergraduate school) is not in the cache so the client server direct the request to the catalog2 server.

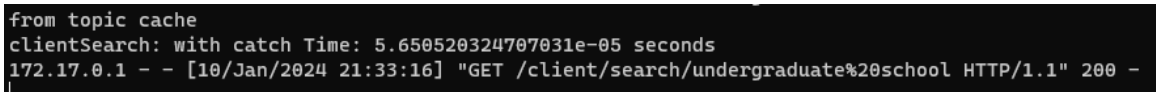
Catalog2Server.py



```
[(3, 'Xen and the Art of Surviving Undergraduate School'), (4, 'Cooking for the Impatient Undergrad'), (6, 'Why theory classes are so hard'), (7, 'S
pring in the Pioneer Valley')]
[{'id': 3, 'title': 'Xen and the Art of Surviving Undergraduate School'}, {'id': 4, 'title': 'Cooking for the Impatient Undergrad'}, {'id': 6, 'titl
e': 'Why theory classes are so hard'}, {'id': 7, 'title': 'Spring in the Pioneer Valley'}]
172.17.0.2 - - [10/Jan/2024 21:30:43] "GET /search/undergraduate%20school HTTP/1.1" 200 -
```

Above is the Catalog2 server output

clientServer.py

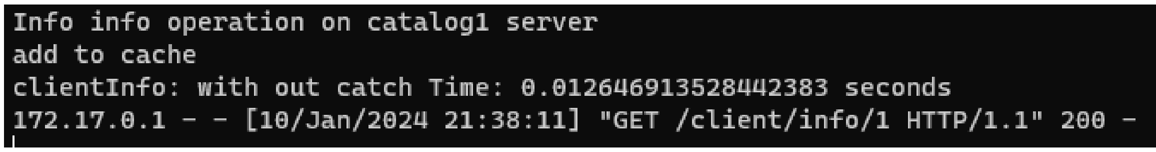


```
from topic cache
clientSearch: with catch Time: 5.650520324707031e-05 seconds
172.17.0.1 - - [10/Jan/2024 21:33:16] "GET /client/search/undergraduate%20school HTTP/1.1" 200 -
```

Here the data in the cache

b- Search by book id

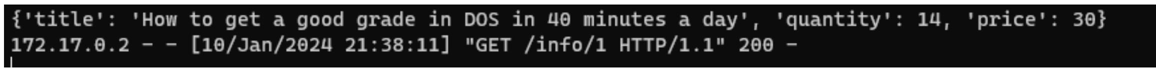
clientServer.py



```
Info info operation on catalog1 server
add to cache
clientInfo: with out catch Time: 0.012646913528442383 seconds
172.17.0.1 - - [10/Jan/2024 21:38:11] "GET /client/info/1 HTTP/1.1" 200 -
```

Here the book (id = 1) info is not in the cache so the client server directs the request to the catalog1 server.

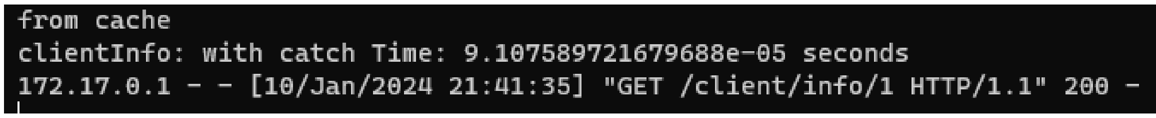
Catalog1Server.py



```
{'title': 'How to get a good grade in DOS in 40 minutes a day', 'quantity': 14, 'price': 30}
172.17.0.2 - - [10/Jan/2024 21:38:11] "GET /info/1 HTTP/1.1" 200 -
```

Above is the Catalog1 server output

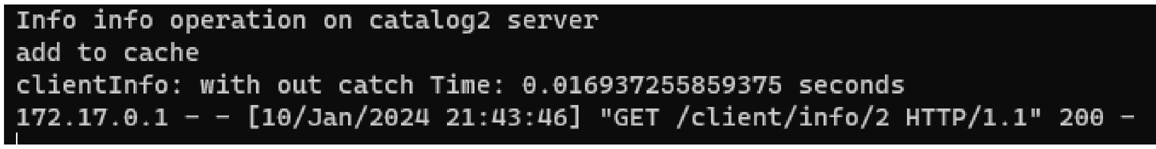
clientServer.py



```
from cache
clientInfo: with catch Time: 9.107589721679688e-05 seconds
172.17.0.1 - - [10/Jan/2024 21:41:35] "GET /client/info/1 HTTP/1.1" 200 -
```

Here the data (book id 1 info) in the cache so the client server return the value from the cache

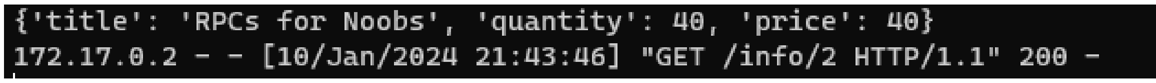
clientServer.py



```
Info info operation on catalog2 server
add to cache
clientInfo: with out catch Time: 0.016937255859375 seconds
172.17.0.1 - - [10/Jan/2024 21:43:46] "GET /client/info/2 HTTP/1.1" 200 -
```

Here the book (id = 2) info is not in the cache so the client server directs the request to the catalog2 server.

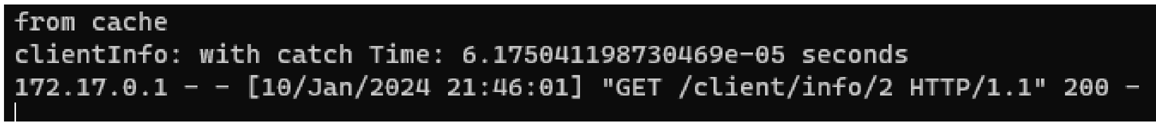
Catalog2Server.py



```
{'title': 'RPCs for Noobs', 'quantity': 40, 'price': 40}
172.17.0.2 - - [10/Jan/2024 21:43:46] "GET /info/2 HTTP/1.1" 200 -
```

Above is the Catalog2 server output

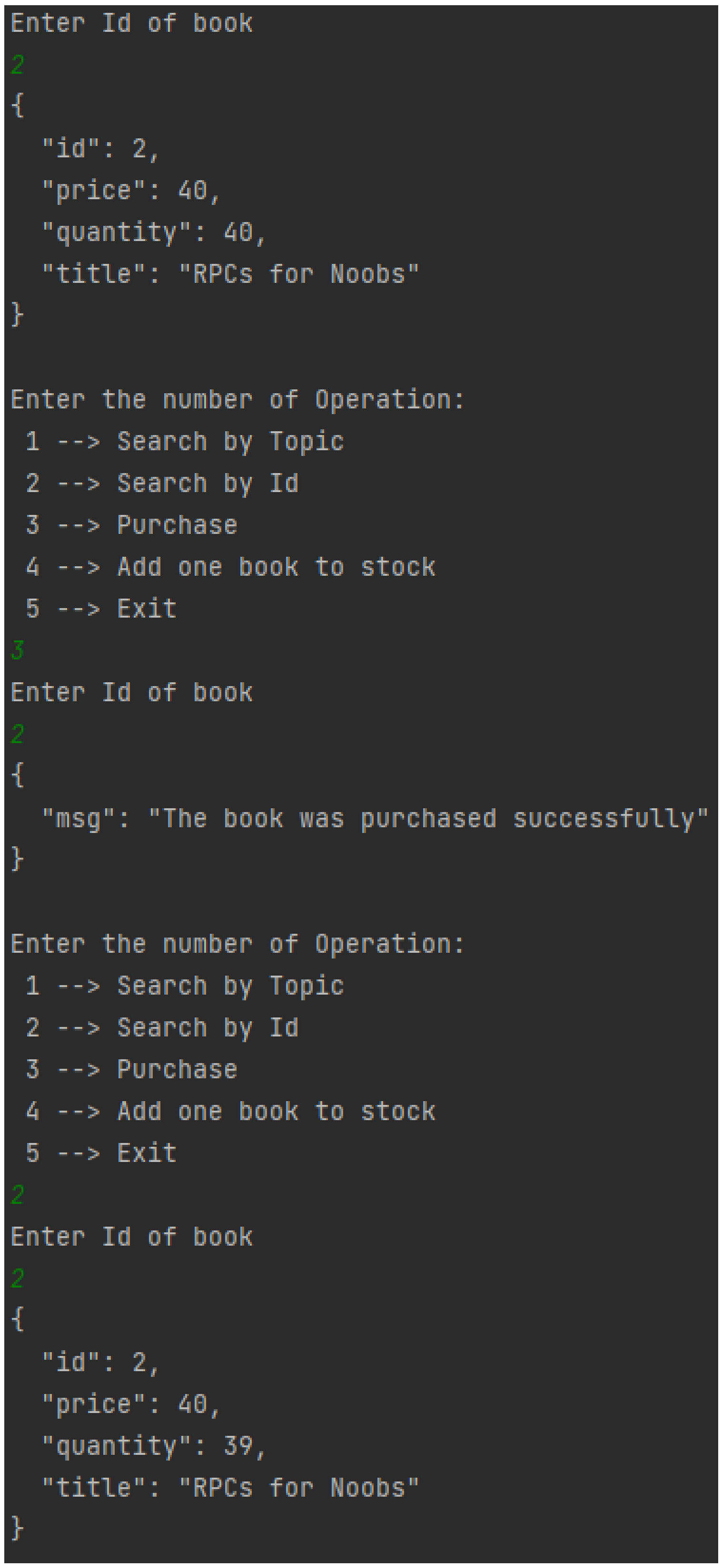
clientServer.py



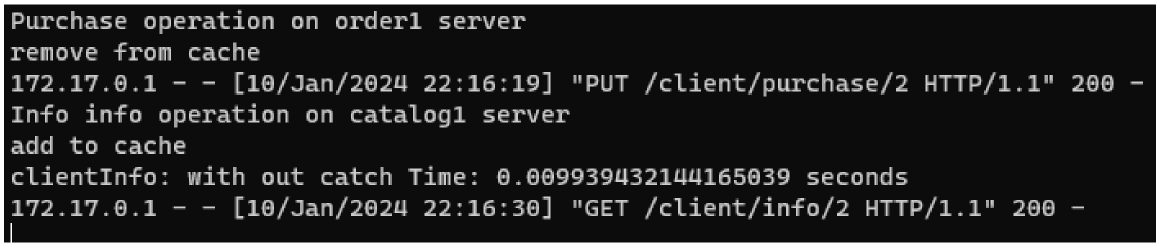
```
from cache
clientInfo: with catch Time: 6.175041198730469e-05 seconds
172.17.0.1 - - [10/Jan/2024 21:46:01] "GET /client/info/2 HTTP/1.1" 200 -
```

In the second request for book id 2 info, the data is in the cache so the client return it directly

c- Purchase books


```
Enter Id of book
2
{
  "id": 2,
  "price": 40,
  "quantity": 40,
  "title": "RPCs for Noobs"
}

Enter the number of Operation:
 1 --> Search by Topic
 2 --> Search by Id
 3 --> Purchase
 4 --> Add one book to stock
 5 --> Exit
3
Enter Id of book
2
{
  "msg": "The book was purchased successfully"
}

Enter the number of Operation:
 1 --> Search by Topic
 2 --> Search by Id
 3 --> Purchase
 4 --> Add one book to stock
 5 --> Exit
2
Enter Id of book
2
{
  "id": 2,
  "price": 40,
  "quantity": 39,
  "title": "RPCs for Noobs"
}
```

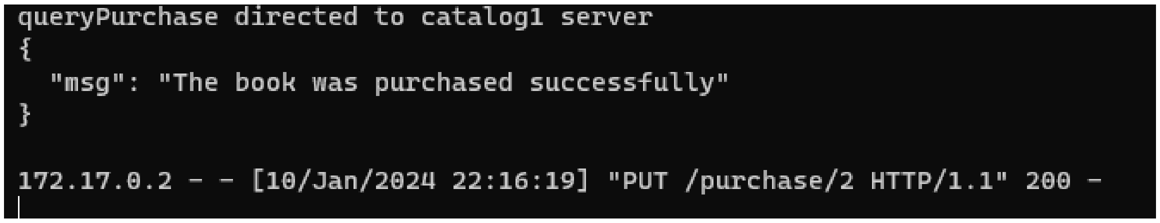
clientServer.py



```
Purchase operation on order1 server
remove from cache
172.17.0.1 - - [10/Jan/2024 22:16:19] "PUT /client/purchase/2 HTTP/1.1" 200 -
Info info operation on catalog1 server
add to cache
clientInfo: with out catch Time: 0.009939432144165039 seconds
172.17.0.1 - - [10/Jan/2024 22:16:30] "GET /client/info/2 HTTP/1.1" 200 -
```

Purchase book(id=2) the book was in the cache, so the client server removes the book from the cache then send the request to the order1 server to perform the purchase operation, then we search for the book Id=2 and add it to the cache in the second search request to the book.

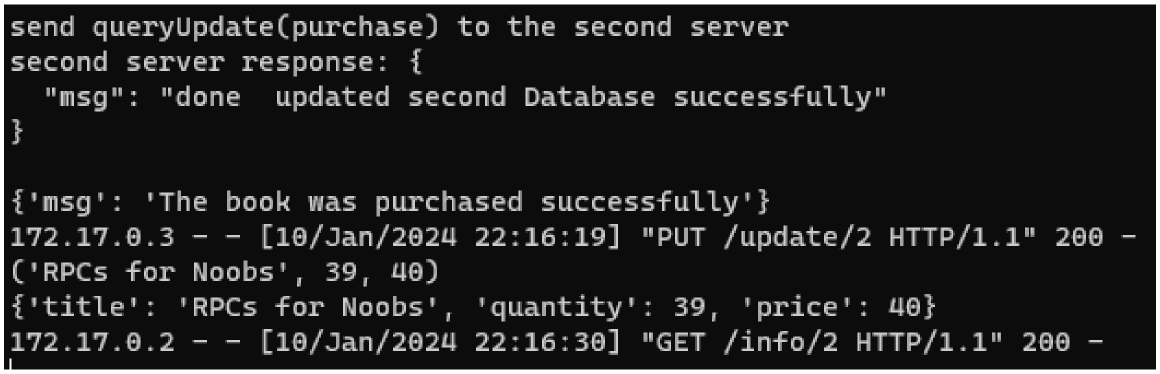
Order1Server.py



```
queryPurchase directed to catalog1 server
{
  "msg": "The book was purchased successfully"
}

172.17.0.2 - - [10/Jan/2024 22:16:19] "PUT /purchase/2 HTTP/1.1" 200 -
```

Above is the Order1 output, the order 1 direct the update query to the catalog1(in the next time the update query to the catalog2, and so on).

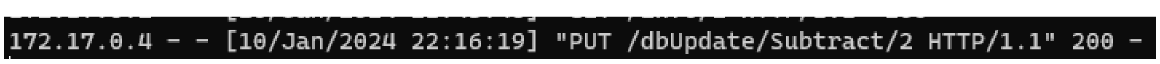
Catalog1Server.py



```
send queryUpdate(purchase) to the second server
second server response: {
  "msg": "done  updated second Database successfully"
}

{'msg': 'The book was purchased successfully'}
172.17.0.3 - - [10/Jan/2024 22:16:19] "PUT /update/2 HTTP/1.1" 200 -
('RPCs for Noobs', 39, 40)
{'title': 'RPCs for Noobs', 'quantity': 39, 'price': 40}
172.17.0.2 - - [10/Jan/2024 22:16:30] "GET /info/2 HTTP/1.1" 200 -
```

Above is the Catalog 1 output

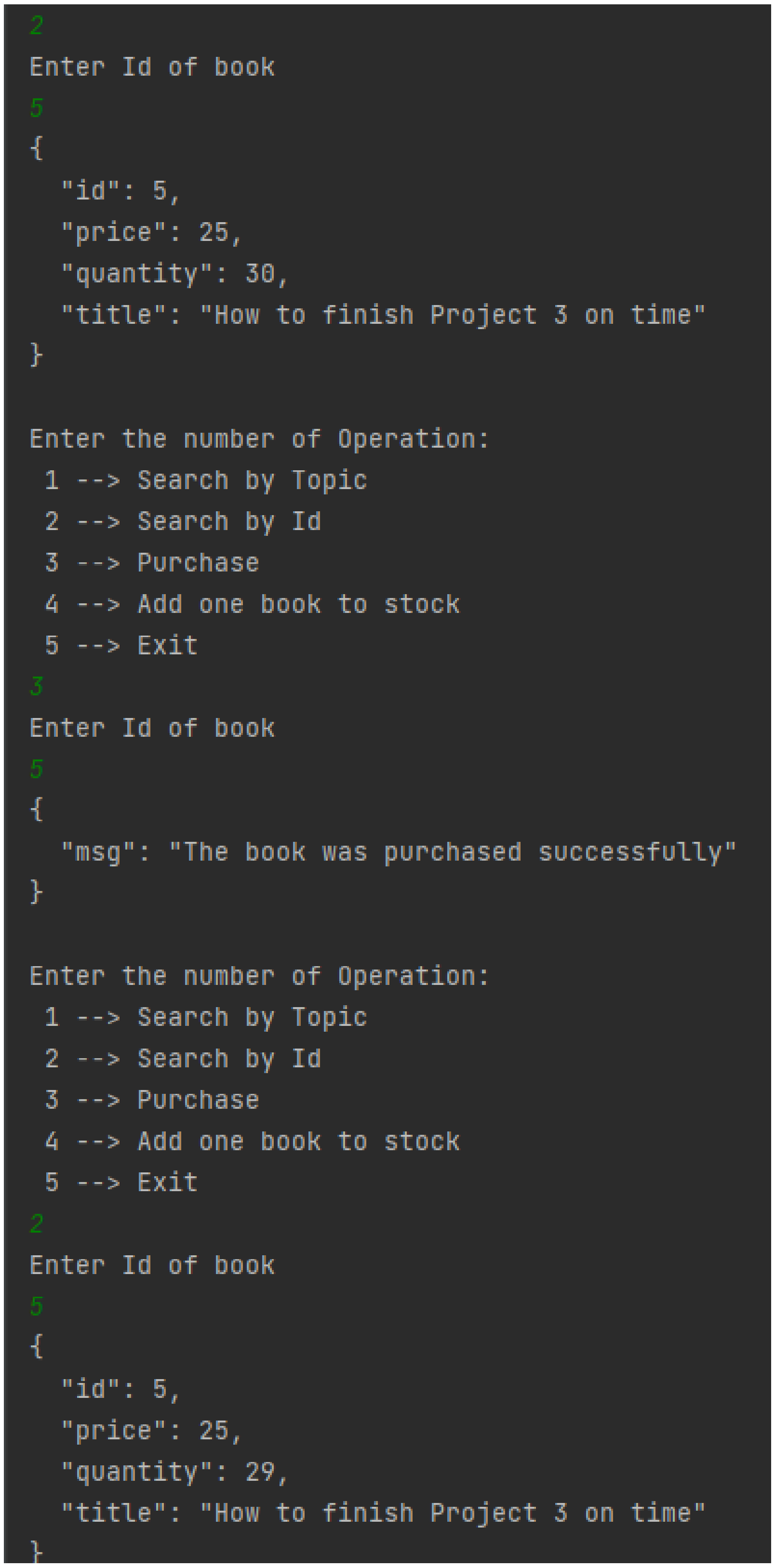
Catalog2Server.py



```
172.17.0.4 - - [10/Jan/2024 22:16:19] "PUT /dbUpdate/Subtract/2 HTTP/1.1" 200 -
```

Above is the Catalog 2 output, this is the database update, to decrement the number of books stock.

The second request to purchase a book

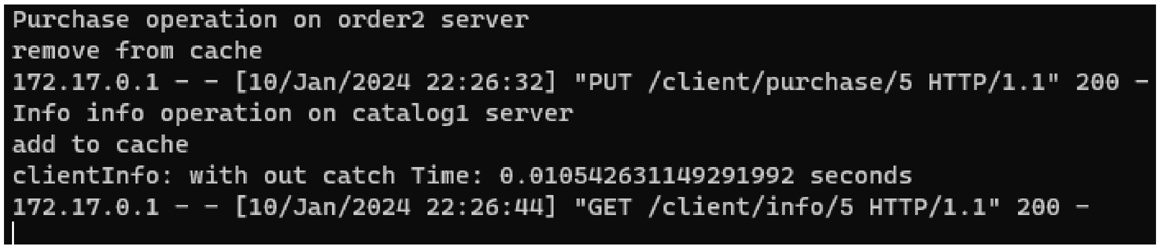


```
2
Enter Id of book
5
{
  "id": 5,
  "price": 25,
  "quantity": 30,
  "title": "How to finish Project 3 on time"
}

Enter the number of Operation:
 1 --> Search by Topic
 2 --> Search by Id
 3 --> Purchase
 4 --> Add one book to stock
 5 --> Exit
3
Enter Id of book
5
{
  "msg": "The book was purchased successfully"
}

Enter the number of Operation:
 1 --> Search by Topic
 2 --> Search by Id
 3 --> Purchase
 4 --> Add one book to stock
 5 --> Exit
2
Enter Id of book
5
{
  "id": 5,
  "price": 25,
  "quantity": 29,
  "title": "How to finish Project 3 on time"
}
```

clientBrowser.py

This is the output of client server:

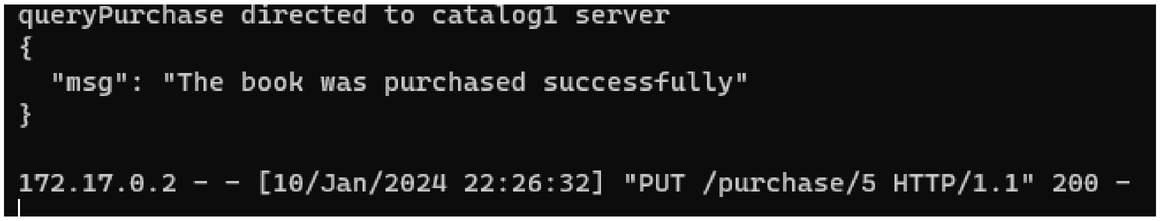
clientServer.py



```
Purchase operation on order2 server
remove from cache
172.17.0.1 - - [10/Jan/2024 22:26:32] "PUT /client/purchase/5 HTTP/1.1" 200 -
Info info operation on catalog1 server
add to cache
clientInfo: with out catch Time: 0.010542631149291992 seconds
172.17.0.1 - - [10/Jan/2024 22:26:44] "GET /client/info/5 HTTP/1.1" 200 -
```

This is the output of order2 server:

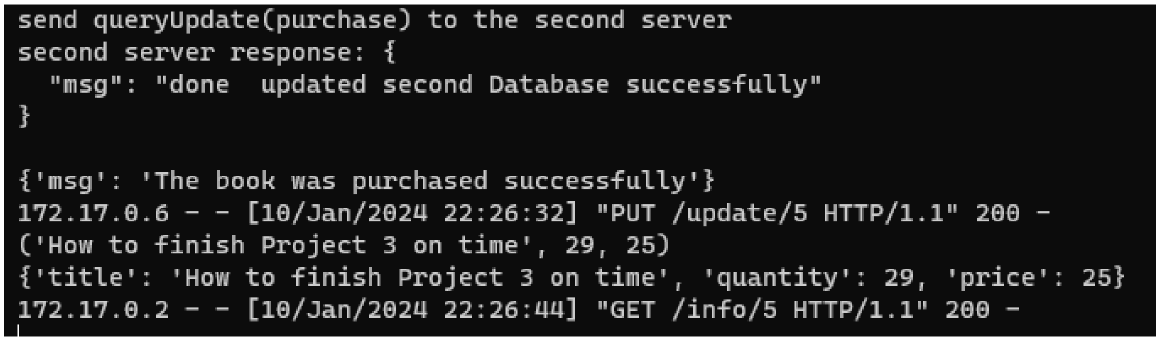
Order2Server.py



```
queryPurchase directed to catalog1 server
{
  "msg": "The book was purchased successfully"
}

172.17.0.2 - - [10/Jan/2024 22:26:32] "PUT /purchase/5 HTTP/1.1" 200 -
```

This is the output of catalog1 server, at the next purchase operation on the order2 server the update query will be directed to the catalog2 server:

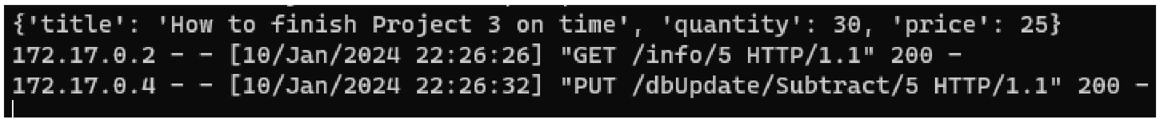
Catalog1Server.py



```
send queryUpdate(purchase) to the second server
second server response: {
  "msg": "done  updated second Database successfully"
}

{'msg': 'The book was purchased successfully'}
172.17.0.6 - - [10/Jan/2024 22:26:32] "PUT /update/5 HTTP/1.1" 200 -
('How to finish Project 3 on time', 29, 25)
{'title': 'How to finish Project 3 on time', 'quantity': 29, 'price': 25}
172.17.0.2 - - [10/Jan/2024 22:26:44] "GET /info/5 HTTP/1.1" 200 -
```

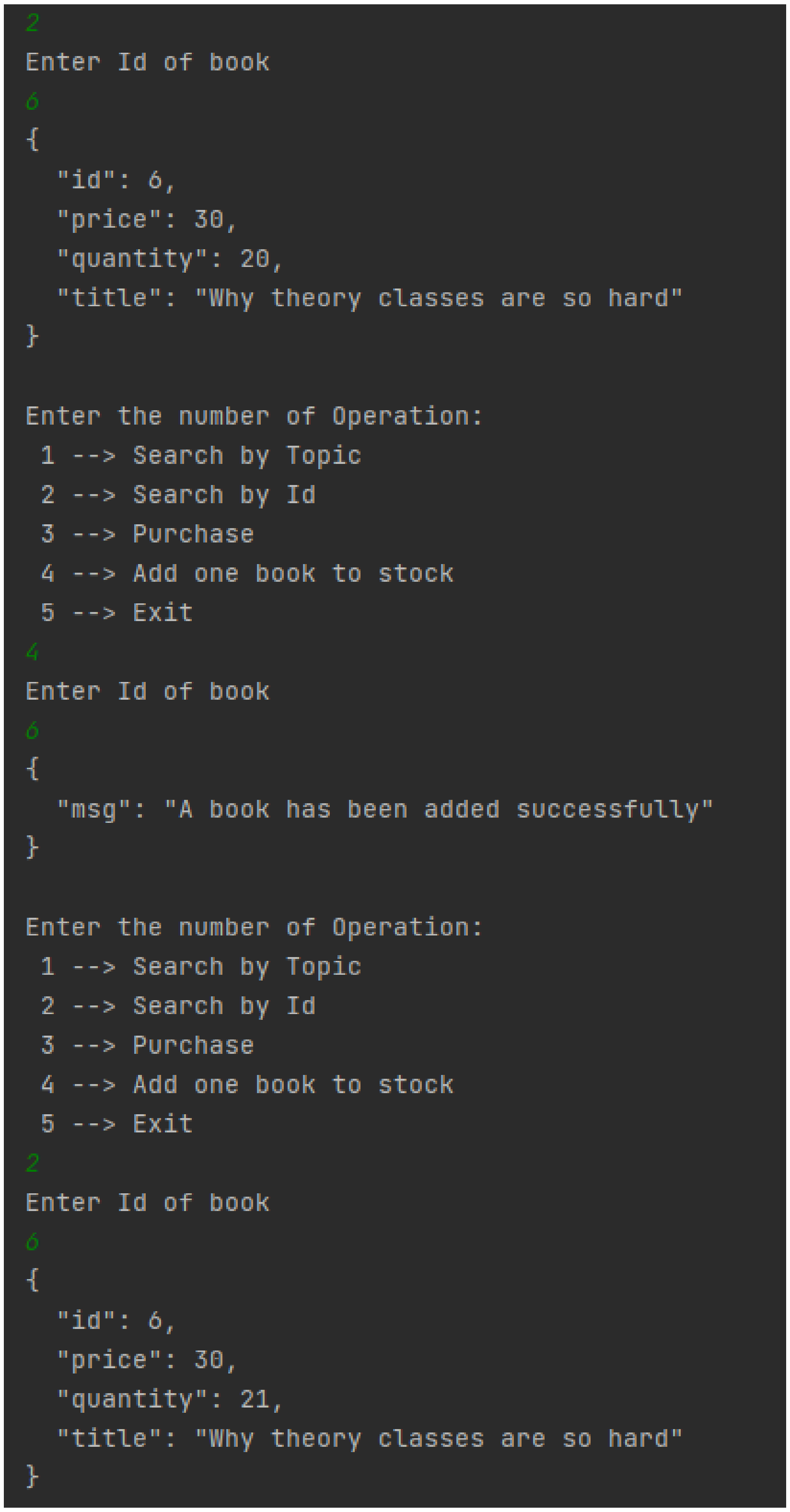
Catalog2 output:

Catalog2Server.py



```
{'title': 'How to finish Project 3 on time', 'quantity': 30, 'price': 25}
172.17.0.2 - - [10/Jan/2024 22:26:26] "GET /info/5 HTTP/1.1" 200 -
172.17.0.4 - - [10/Jan/2024 22:26:32] "PUT /dbUpdate/Subtract/5 HTTP/1.1" 200 -
```

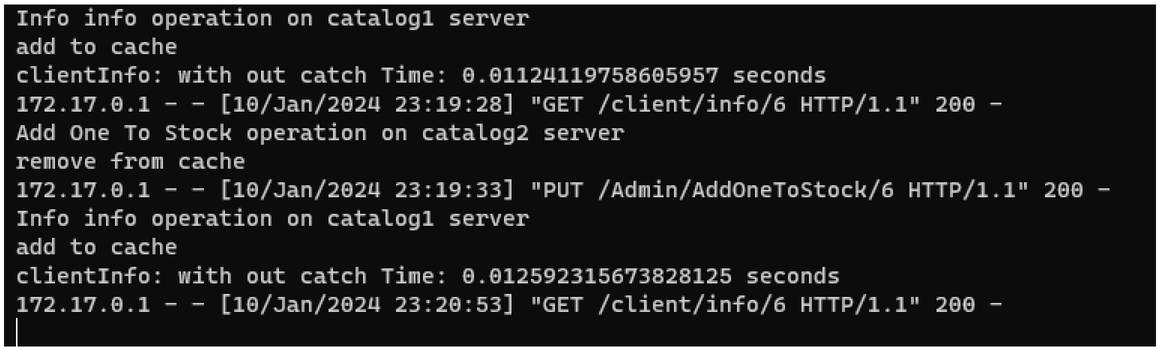
d- Add one book to the stock



```
2
Enter Id of book
6
{
  "id": 6,
  "price": 30,
  "quantity": 20,
  "title": "Why theory classes are so hard"
}

Enter the number of Operation:
 1 --> Search by Topic
 2 --> Search by Id
 3 --> Purchase
 4 --> Add one book to stock
 5 --> Exit
4
Enter Id of book
6
{
  "msg": "A book has been added successfully"
}

Enter the number of Operation:
 1 --> Search by Topic
 2 --> Search by Id
 3 --> Purchase
 4 --> Add one book to stock
 5 --> Exit
2
Enter Id of book
6
{
  "id": 6,
  "price": 30,
  "quantity": 21,
  "title": "Why theory classes are so hard"
}
```

The client server output:

clientServer.py

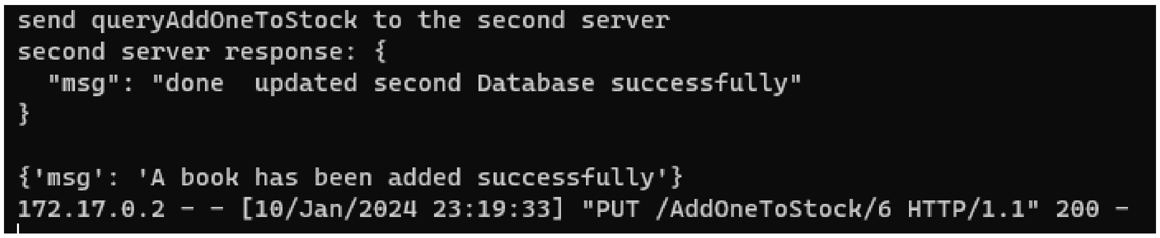


```
Info info operation on catalog1 server
add to cache
clientInfo: with out catch Time: 0.01124119758605957 seconds
172.17.0.1 - - [10/Jan/2024 23:19:28] "GET /client/info/6 HTTP/1.1" 200 -
Add One To Stock operation on catalog2 server
remove from cache
172.17.0.1 - - [10/Jan/2024 23:19:33] "PUT /Admin/AddOneToStock/6 HTTP/1.1" 200 -
Info info operation on catalog1 server
add to cache
clientInfo: with out catch Time: 0.012592315673828125 seconds
172.17.0.1 - - [10/Jan/2024 23:20:53] "GET /client/info/6 HTTP/1.1" 200 -
```

Here we add one book (id=6) to the stock quantity, and remove the book from the cache, because the data is invalid, then in the next search request the book added to the cache.

Here is the catalog2 output:

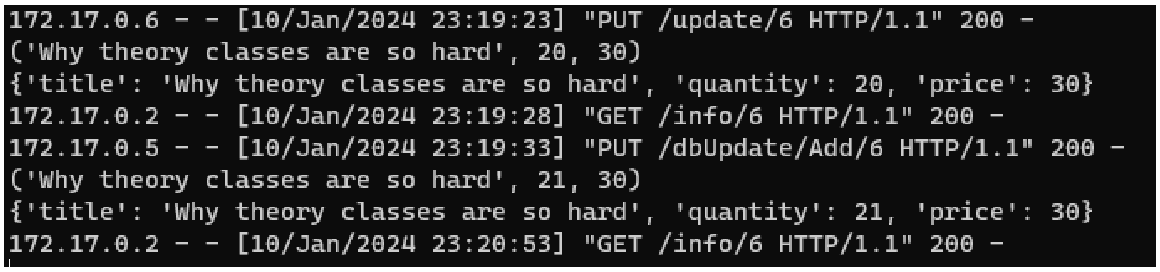
Catalog2Server.py



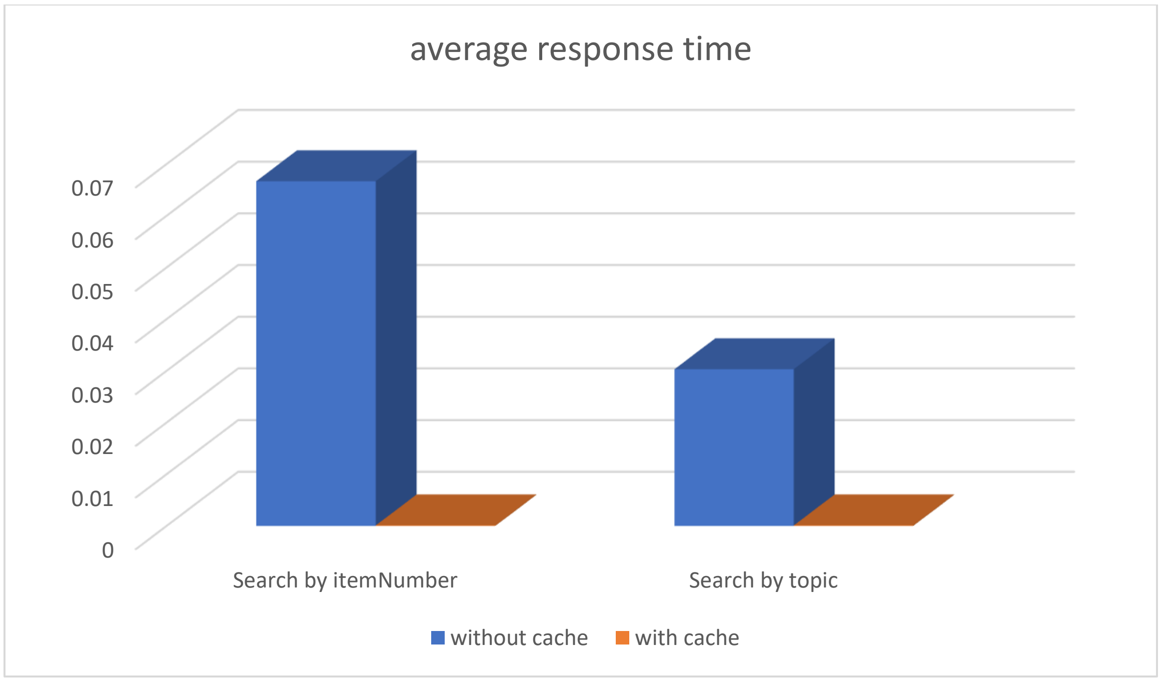
```
send queryAddOneToStock to the second server
second server response: {
  "msg": "done  updated second Database successfully"
}

{'msg': 'A book has been added successfully'}
172.17.0.2 - - [10/Jan/2024 23:19:33] "PUT /AddOneToStock/6 HTTP/1.1" 200 -
```

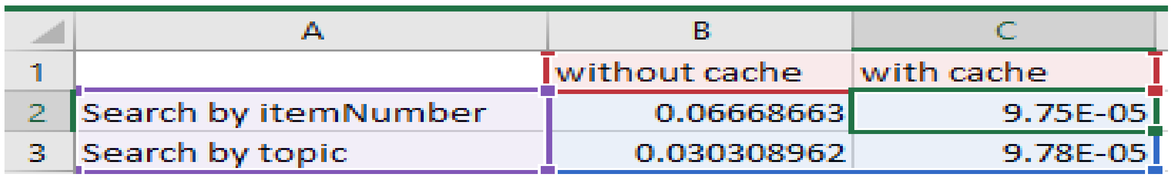
Here is the catalog1 output:

Catalog1Server.py



```
172.17.0.6 - - [10/Jan/2024 23:19:23] "PUT /update/6 HTTP/1.1" 200 -
('Why theory classes are so hard', 20, 30)
{'title': 'Why theory classes are so hard', 'quantity': 20, 'price': 30}
172.17.0.2 - - [10/Jan/2024 23:19:28] "GET /info/6 HTTP/1.1" 200 -
172.17.0.5 - - [10/Jan/2024 23:19:33] "PUT /dbUpdate/Add/6 HTTP/1.1" 200 -
('Why theory classes are so hard', 21, 30)
{'title': 'Why theory classes are so hard', 'quantity': 21, 'price': 30}
172.17.0.2 - - [10/Jan/2024 23:20:53] "GET /info/6 HTTP/1.1" 200 -
```





	A	B	C
1		without cache	with cache
2	Search by itemNumber	0.06668663	9.75E-05
3	Search by topic	0.030308962	9.78E-05

From this data we concluded that using cache is very fast than without cache.

in the previous screenshots we explain when the cache miss happened, the cache miss happened when we want to add or purchase books, so we search for the book in the cache then we delete the item from cache, in the next search query there will be a cache miss then the clientServer will forword the request to the catalog server to get the updated data and store it in the cache to use it faster than make API call.