# Bazar.com

**Online Book Store lab 2**

**Supervised by Dr. Samer AL-Arandi**

**Done by:**

| | |
|---|---|
| **Ayham Baarah** | **12218367** |
| **Ayham Thiab** | **12218365** |

## ❖ Introduction:

Due to the increasing popularity of Bazar.com, the system began to experience high load and increased latency when handling user requests, especially during book queries and purchase operations.
The objective of **Lab 2** is to redesign the system developed in Lab 1 to handle higher workloads by introducing **replication** and **caching**, while maintaining **strong consistency**.
This lab focuses on improving performance, scalability, and fault tolerance using distributed systems concepts.

## ❖ System Architecture Overview

The system consists of the following components:

- **Frontend Server**
- **Catalog Service (2 replicas)**
- **Order Service (2 replicas)**

All components communicate using **REST APIs**.
The frontend server acts as the single entry point for all client requests and is responsible for caching and load balancing.

## ❖ Frontend Server Design

The frontend server performs the following tasks:

- Receives all client requests.
- Implements an **in-memory LRU cache** for catalog read requests.
- Forwards read requests to catalog replicas on cache miss.
- Forwards write requests (purchase operations) to order replicas.
- Performs **cache invalidation** before any write operation.
- Uses **load balancing and failover** when replicas are unavailable.

## ❖ Caching Mechanism

- The cache is implemented inside the frontend server.
- It stores recently accessed book information.
- Cache is used **only for read requests** (/info/<book_id>).
- Write requests bypass the cache.
- Cache entries are invalidated before any stock update to maintain consistency.
-

## ❖ Replication Strategy

### ➢ Catalog Replication

- Two catalog replicas maintain identical copies of the books database.
- Read requests are distributed across replicas.
- If one replica fails, the system continues serving requests using the remaining replica or the cache.

### ➢ Order Replication

- Two order replicas process purchase requests.
- Each purchase is written locally and replicated to the other replica.
- The frontend retries another replica if one order server is unavailable.

## ❖ Experimental Evaluation

All experiments were conducted using **Command Prompt (CMD)** and curl commands.

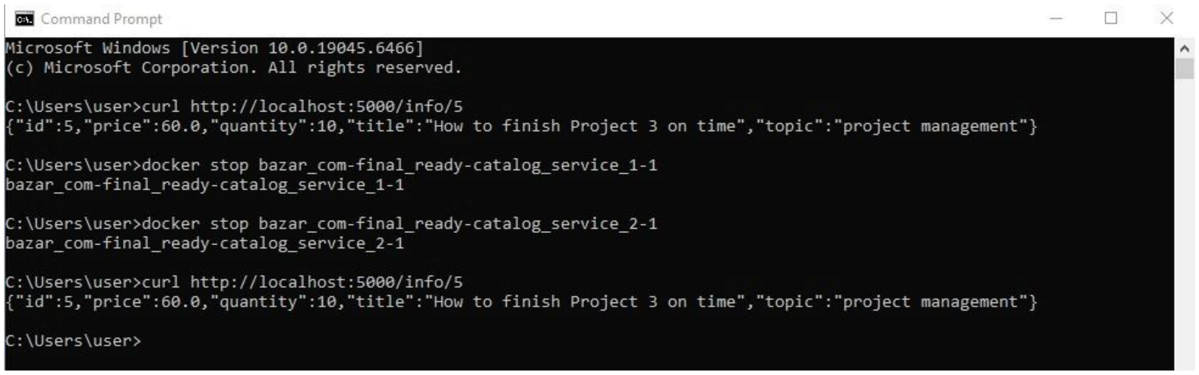
### Test 1: Cache Usage and Availability

The following steps were performed:

1. A book information request was sent to the frontend.
2. Both catalog replicas were stopped.
3. The same request was sent again.

Despite both catalog services being down, the frontend successfully returned the book information from its cache.

**This confirms correct cache usage and availability.**



```
Microsoft Windows [Version 10.0.19045.6466]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>curl http://localhost:5000/info/5
{"id":5,"price":60.0,"quantity":10,"title":"How to finish Project 3 on time","topic":"project management"}

C:\Users\user>docker stop bazar_com-final_ready-catalog_service_1-1
bazar_com-final_ready-catalog_service_1-1

C:\Users\user>docker stop bazar_com-final_ready-catalog_service_2-1
bazar_com-final_ready-catalog_service_2-1

C:\Users\user>curl http://localhost:5000/info/5
{"id":5,"price":60.0,"quantity":10,"title":"How to finish Project 3 on time","topic":"project management"}

C:\Users\user>
```

*(Cache Hit with both catalog replicas stopped)*

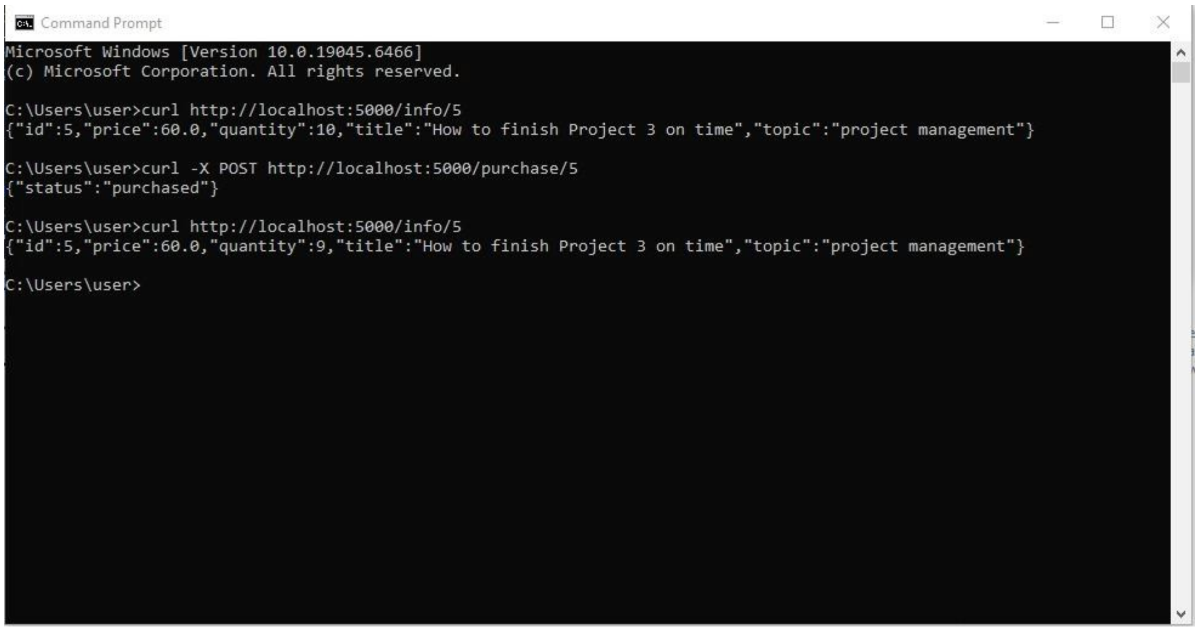
### Test 2: Cache Invalidation after Purchase

The following steps were performed:

1. Book information was retrieved.
2. A purchase request was executed.
3. Book information was retrieved again.

The quantity of the book decreased correctly after the purchase, indicating that the cache entry was invalidated and refreshed.

**This confirms correct cache invalidation and strong consistency.**



```
Microsoft Windows [Version 10.0.19045.6466]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>curl http://localhost:5000/info/5
{"id":5,"price":60.0,"quantity":10,"title":"How to finish Project 3 on time","topic":"project management"}

C:\Users\user>curl -X POST http://localhost:5000/purchase/5
{"status":"purchased"}

C:\Users\user>curl http://localhost:5000/info/5
{"id":5,"price":60.0,"quantity":9,"title":"How to finish Project 3 on time","topic":"project management"}

C:\Users\user>
```

*(Cache invalidation after purchase operation)*

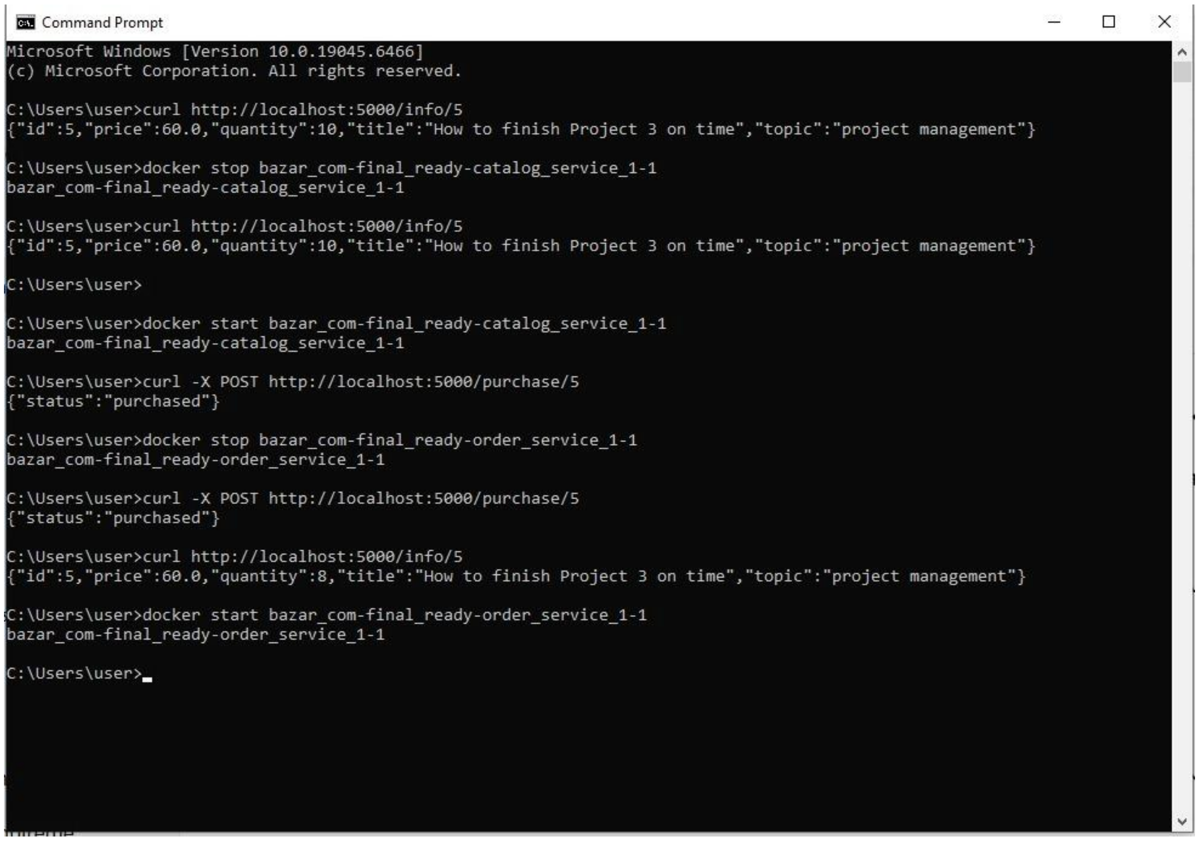
## Test 3: Replication and Failover

The following steps were performed:

1. One catalog replica was stopped and read requests were still served successfully.
2. One order replica was stopped and purchase requests were still processed successfully.
3. The final book quantity reflected all successful purchases.

The system continued functioning correctly even when individual replicas were unavailable.

**This confirms successful replication and failover for both catalog and order services.**



```
Microsoft Windows [Version 10.0.19045.6466]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>curl http://localhost:5000/info/5
{"id":5,"price":60.0,"quantity":10,"title":"How to finish Project 3 on time","topic":"project management"}

C:\Users\user>docker stop bazar_com-final_ready-catalog_service_1-1
bazar_com-final_ready-catalog_service_1-1

C:\Users\user>curl http://localhost:5000/info/5
{"id":5,"price":60.0,"quantity":10,"title":"How to finish Project 3 on time","topic":"project management"}

C:\Users\user>

C:\Users\user>docker start bazar_com-final_ready-catalog_service_1-1
bazar_com-final_ready-catalog_service_1-1

C:\Users\user>curl -X POST http://localhost:5000/purchase/5
{"status":"purchased"}

C:\Users\user>docker stop bazar_com-final_ready-order_service_1-1
bazar_com-final_ready-order_service_1-1

C:\Users\user>curl -X POST http://localhost:5000/purchase/5
{"status":"purchased"}

C:\Users\user>curl http://localhost:5000/info/5
{"id":5,"price":60.0,"quantity":8,"title":"How to finish Project 3 on time","topic":"project management"}

C:\Users\user>docker start bazar_com-final_ready-order_service_1-1
bazar_com-final_ready-order_service_1-1

C:\Users\user>_
```

*(Catalog and Order failover with successful reads and writes)*

## ❖ Conclusion

In this lab, Bazar.com was successfully enhanced with caching and replication mechanisms.
The redesigned system handles higher workloads efficiently while maintaining strong consistency and fault tolerance.

All requirements of **Lab 2** were successfully implemented and validated.