How it works

How the System Works (End-to-End)

Read Operation (/info/{id})

1. Client sends request to front-end
2. Front-end checks in-memory cache
3. If cache hit → return response immediately
4. If cache miss → forward request to catalog replica
5. Store response in cache and return to client

Write Operation (/purchase/{id})

1. Client sends purchase request
2. Front-end forwards to order service replica
3. Order service updates catalog inventory
4. Cache invalidation request is sent to front-end
5. Cache entry is removed
6. Confirmation returned to client

Design Tradeoffs

Advantages

- Low read latency due to caching
- Horizontal scalability through replication
- Clear separation of responsibilities
- Simple and transparent consistency model

Tradeoffs

- Write operations are slower due to cache invalidation
- Single front-end is a potential bottleneck
- In-memory cache is volatile (lost on restart)

Possible Improvements and Extensions

1. **Distributed Cache**
    - Replace local cache with Redis or Memcached
2. **Adaptive Load Balancing**
    - Route requests based on replica load or health
3. **Fault Tolerance**
    - Detect failed replicas dynamically
4. **Persistent Storage**
    - Replace CSV with a database
5. **Asynchronous Invalidation**
    - Reduce write latency using async cache updates

How to Run the Program

Prerequisites

- Java JDK 11+
- Maven
- Git

Steps

1. Clone repository:

git clone https://github.com/Ameedjabr/Bazarcom.git

2. Start Catalog replicas:

cd catalog-service

mvn exec:java -Dexec.args=5000

mvn exec:java -Dexec.args=5001

3. Start Order replicas:

cd order-service

mvn exec:java -Dexec.args=7000

mvn exec:java -Dexec.args=7001

4. Start Front-End:

cd frontend-service

mvn exec:java

5. Access via browser or curl:

http://localhost:9000/info/5

http://localhost:9000/purchase/5

Conclusion

This project demonstrates a complete distributed system that balances performance, consistency, and simplicity. Through replication, caching, and careful cache invalidation, the system achieves improved read performance while maintaining correctness for write operations.

Important screenshots

1. Front-end service running (terminal)

```
PS C:\Users\ME\Desktop\Bazarcom\frontend-service> mvn exec:java
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< bazar:frontend-service >----------------------
[INFO] Building frontend-service 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- exec:3.1.0:java (default-cli) @ frontend-service ---
FrontEndService running at http://localhost:9000
```

2. Catalog service running (terminal)

```
[INFO] ------------------------------------------------------------------------
PS C:\Users\ME\Desktop\Bazarcom\catalog-service> mvn exec:java
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< bazar:catalog-service >-----------------------
[INFO] Building catalog-service 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- exec:3.1.0:java (default-cli) @ catalog-service ---
Loaded 7 items from CSV
CatalogService running on http://localhost:5000
```

3. Order service running (terminal)

```
Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine
Run 'Import-Module PSReadLine'.

PS C:\Users\ME\Desktop\Bazarcom\order-service> mvn exec:java
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< bazar:order-service >-------------------------
[INFO] Building order-service 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- exec:3.1.0:java (default-cli) @ order-service ---
OrderService running at http://localhost:7000/order?id=1
```

Replication & Load Balancing

4. Catalog replica running on port 5000

```
[INFO] ------------------------------------------------------------------------
PS C:\Users\ME\Desktop\Bazarcom\catalog-service> mvn exec:java
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< bazar:catalog-service >-----------------------
[INFO] Building catalog-service 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- exec:3.1.0:java (default-cli) @ catalog-service ---
Loaded 7 items from CSV
CatalogService running on http://localhost:5000
```

5. Catalog replica running on port 5001

```
PS C:\Users\ME\Desktop\Bazarcom\catalog-service> mvn exec:java "-Dexec.args=5001"
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< bazar:catalog-service >-----------------------
[INFO] Building catalog-service 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- exec:3.1.0:java (default-cli) @ catalog-service ---
Loaded 7 items from CSV
CatalogService running on http://localhost:5001
```

6. Successful /search request through front-end

localhost:5001/search/undergraduate%20school

Pretty-print


```
{
  "items": [
    {
      "id": "3",
      "title": "Xen and the Art of Surviving Undergraduate School"
    },
    {
      "id": "4",
      "title": "Cooking for the Impatient Undergrad"
    },
    {
      "id": "5",
      "title": "How to finish Project 3 on time"
    },
    {
      "id": "6",
      "title": "Why theory classes are so hard"
    },
    {
      "id": "7",
      "title": "Spring in the Pioneer Valley"
    }
  ]
}
```

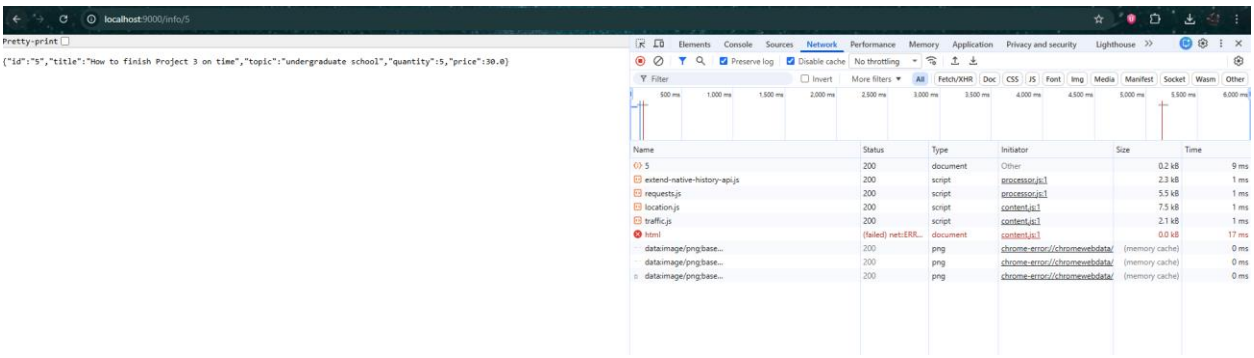

localhost:5000/search/undergraduate%20school

Pretty-print

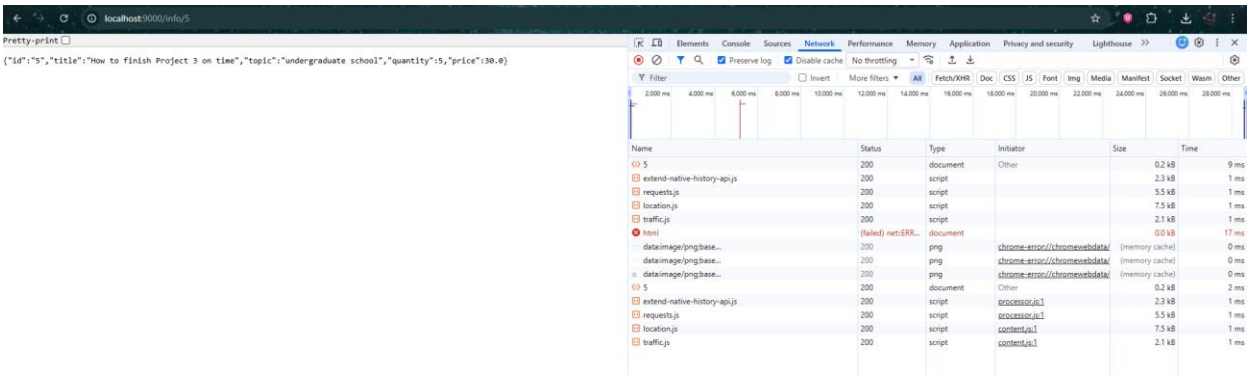

```
{
  "items": [
    {
      "id": "3",
      "title": "Xen and the Art of Surviving Undergraduate School"
    },
    {
      "id": "4",
      "title": "Cooking for the Impatient Undergrad"
    },
    {
      "id": "5",
      "title": "How to finish Project 3 on time"
    },
    {
      "id": "6",
      "title": "Why theory classes are so hard"
    },
    {
      "id": "7",
      "title": "Spring in the Pioneer Valley"
    }
  ]
}
```


Caching

7. /info/5 request (first time – cache miss)

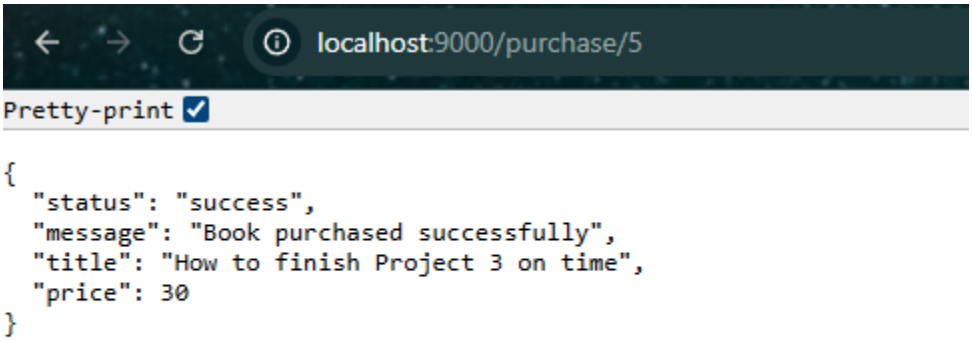


8. /info/5 request (second time – cache hit)



Cache Invalidation

9. /purchase/5 request success



10. /info/5 request after purchase (cache miss)

Performance Measurement

11. PowerShell output – query with cache

```
Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes. If you want to re-enab
it, run 'Import-Module PSReadLine'.

PS C:\Users\ME\Desktop\Bazarcom> $times=@()
PS C:\Users\ME\Desktop\Bazarcom> 1..30 | % {
>>   $ms = (Measure-Command { Invoke-WebRequest -UseBasicParsing "http://localhost:9000/info/5" }).TotalMilliseconds
>>   $times += $ms
>> }
>> $times | Measure-Object -Average -Minimum -Maximum
>>


Count    : 30
Average  : 14.6565533333333
Sum      :
Maximum  : 128.2336
Minimum  : 8.5454
Property :


PS C:\Users\ME\Desktop\Bazarcom>
```

12. PowerShell output – query without cache

```
Count    : 30
Average  : 12.9259066666667
Sum      :
Maximum  : 95.1761
Minimum  : 9.051
Property :
```

13. PowerShell output – buy with cache

```
Count    : 20
Average  : 23.78468
Sum      :
Maximum  : 153.9493
Minimum  : 8.6469
Property :
```

14. PowerShell output – buy without cache

```
Count    : 20
Average  : 11.860335
Sum      :
Maximum  : 94.4014
Minimum  : 5.9603
Property :
```