**Design Document: Turning the Bazar into an Amazon**

**داليا حامد سليم دويكات 11923878**

**Introduction:**

**In this project, we elevate Bazar.com's online store by implementing Replication for resilience and load balancing. Caching enhances response times through an in-memory cache, while Consistency ensures synchronized data updates across servers. These features collectively create an advanced and efficient online shopping experience for users.**

**Describe Servers- System Architecture:**

* **Front-end server:**
* **this file works on localhost with port 5000**
* **has 3 endpoints inside it (search, info, purchase)**
* **has a cache inside it, this will store all the responses that the user sends to this server if the user resends the same request or any request the server will check first the cache, and inside it, we achieve the (Caching).**
* **I used a round-robin algorithm, to help the front-end server choose between 4 servers (catalog and order and their replicas), so in each operation that happened, here I achieved the (Replication)**
* **Catalog Server:**
* **in this server, only the change that happens is when any update on the data file, will send a request to the frontend server to tell the cache that the data is changed.**
* **also when the update operation happens, it will send a request for the catalog server -replica, to tell it that the data is changed, and the replica will do its part to update its database replica.**
* **Catalog Server - Replica:**

**this server is the same as the catalog server, but only the changes are in the port and also in the database file so consistency here is very important between the servers, and this we achieved in our code, by making each server tell the other server when it is updated to update there databases file.**

* **Order Server:**
* **this server has not changed that in part one, only simple additions, so when the order server takes with the catalog server, the order server uses the round robin algorithm here also to choose between these servers, and that makes the overhead and the performance better. (Replication)**
* **also when the order files are changed as a result of the catalog database, the order server will tell the order server ---replica to change his data. (Consistency)**
* **Order Server- replica:**

**this server is the same as the order server, but the difference is the file that is written on it, so when any change happens, also the replica will tell the order server to update the data file for it, it uses the round-robin to choose between catalog and replica.**

**Functional Requirements:**

* **Replication:**

**we achieved that when using the replica servers, we divided the implementation into several servers, which speeds up and improves the implementation process and relieves pressure on servers.**

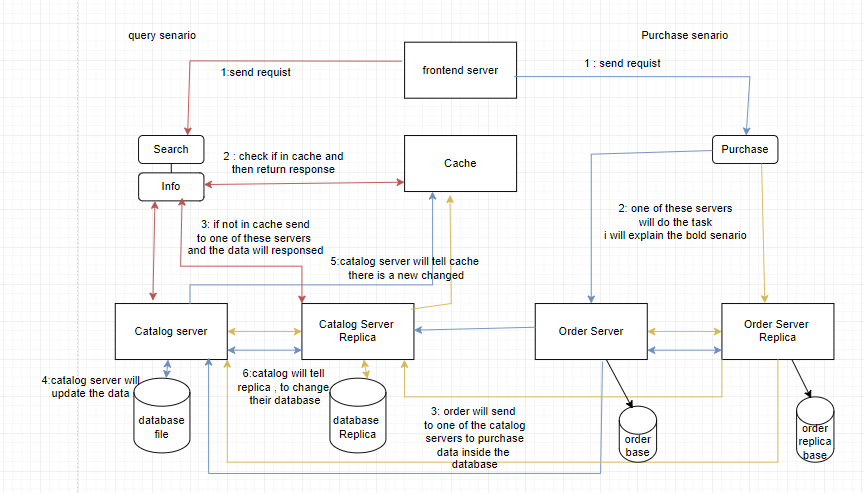
* **Caching:**

**we achieved this when we added cache in our front-end server (cache = LRUCache(maxsize=1000)), so any response will store inside the cache, the cache is useful in the speed of execution of any request that comes to it if its result is stored inside it, which improves performance for the code.**

* **Consistency:**

**This is achieved when any modification occurs on any data, the rest of the servers are told to update their database file, as well as the cache is informed of any modification, this way the implementation of the program is improved smoothly and helped us in the replication.**

**Code Scenario:**



**Extensions and enchantments:**

**Expiration Policies:(blue color box in next image )**

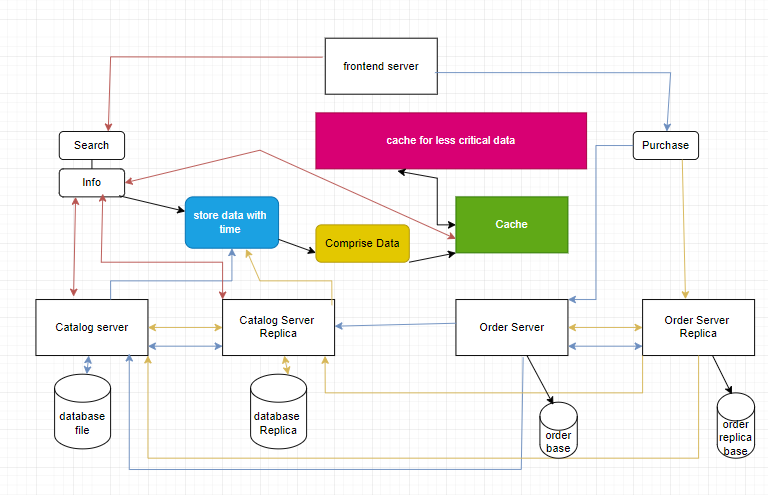
* + **Add expiration policies to the caching system, allowing cached items to be automatically removed after a specified time. This ensures that the cache always contains fresh and relevant data.**

**Tiered Caching:(Pink and green boxes in next image)**

* **Implement a tiered caching approach where frequently accessed or critical data is stored in a high-speed, low-capacity cache, while less critical data is stored in a larger, slower cache. This tiered structure optimizes memory usage for different types of data.**

**Compression Techniques:(Yellow color box in next image)**

* **Introduce memory compression techniques to reduce the storage footprint of cached data. Compressing data before storing it in the cache can significantly minimize memory usage while maintaining fast access times.**

****

**How can run the code?**

**I run each server on PowerShell on different ports,**

**on postman:**

**this foe id = 1, and can change the id and use any element**

**localhost:5000/search/1 \_GET: this is for search operation by id**

**localhost:5000/info/1 \_ GET: this Info operation by id**

**localhost:5000/purchase/1 \_POST: This purchase operation**

* **Note : the performance tabel in the output document.**