DOS PROJECT 2

Hassan Saed Hassan Abu Thiab **11714642**

Libraries and packages used:

- RESTFUL flask
- SQLALCHEMY
- Requests
- SQLlite
- Flask

Design of the Code:

Replication: My design was to copy the contents of the old code and paste into a new program. Copying contents of old databases into new ones and connecting the replicas to their databases.

- **Catalog Replication**: If Catalog1 changes then it sends a request to Catalog2 to perform a change with the ID that got purchased, this way both Replicas keep the same database. So if we -1 then we -1 there too.
- **Orderlog Replication**: If Orderlog1 changes then we send the log of the Order towards the other Orderlog2 so they add into the Log. We send the entire JSON as text then write it into txt that behaves as an orderlog
- **Load balancing**: We perform very simple load balancing where the request will be sent to One replica then the next request to the other. And repeat. So the load will be split in half equally.

Better design choices:

- We can perform backup in case the consistency operation fails.
- We can use a more complex algorithm if we have more than two replicas but for two this works just fine.

Cache: It's done as an inline memory within the ClientServer. I used dictionary for that since it behaves the closest to a Hashmap.

- **Search**: Stores misses in the form of Topic as key and the entire list as data
- **Info**: Stores misses in form of ID as key and one item as data
- **Consistency**: We have two operations, both remove keys from the dictionary, both TOPIC and ID of the Item that got purchased will be removed. This will be called by the Catalog replicas meaning keeping the consistency will be very costly in term of performance

Better Design choices:

- Backup in case the operation fails for consistency
- We can have multiple keys be saved for the topic using ID rather than using topic

How to run the Code:

- Put each file in a machine
- Make sure to read the IP address of the machines
- Fill the IP address based on what's required
- Interact with the servers using terminal
- SAME AS LAB1 OF PROJECT, ONLY CHANGE IS MORE MACHINES

Response time difference:

Operation/Cache	With Cache	Without Cache
INFO	0.0046	0.0101
TOPIC	0.0042	0.044

We can notice that the Cache sped up TOPIC request by 10 times since we don't perform query in cache+it's inline

We also notice cache spend up INFO by 2.5. Since the operation was originally shorter in time.

Consistency:

Purchase with Consistency = 0.1837

Purchase without = 0.065

We can see consistency affects the performance heavily so it has it's downsides