E-commerce System Design

(Amazon, Flipkart, Ajio, Myntra, H&m, Zara)

1) Functional Requirements:

a) searching for a product

b) Adding to the cart

c) Adding to the wish-list

d) checkout

e) view order

2) Non-Functional Requirements:

a) Low Latency

b) High Availability

c) High Consistency.

We cannot come up with a system that is high available & highly consistent at a time. In our case, For a set of service, We need high consistency (Ex: payment system) & For few other service, high availability is required.

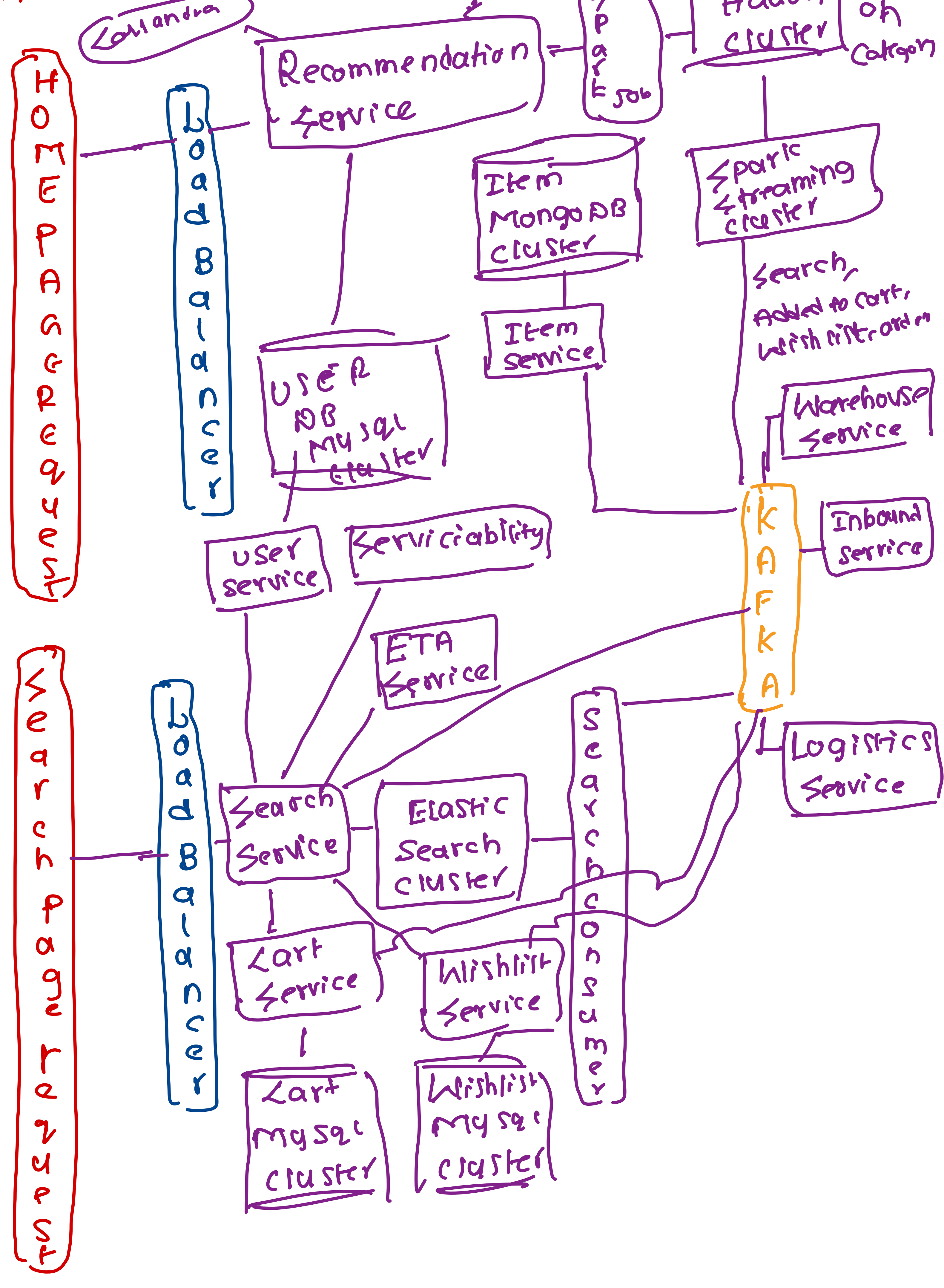
Therefore, We need a system that is highly consistent for a set of services and highly available for other set of services.

Architecture:

Client-Server Architecture

Communication channel between the client and the Server: HTTPS | HTTP. (Most rest APIs)

Architecture Diagram:



(1) In Bound Service:

This service is responsible for adding new items to the system.

Role: It acts as an interface for the suppliers. Suppliers will add the items into the E-commerce.

Role 2: The newly added items are pushed into KAFKA TOPIC so that dependent services like search consumer, Item service can consume.

(2) ITEM SERVICE:

This service adds the newly added item details into MongoDB.

Why MongoDB ?? Different items have different descriptions. Therefore, MongoDB (Document DB) is the best choice.

Example 1:

```
{ name: TV
      l : l cms
      h : h cms
      w : w kgs
      B : Sony
}
```

Example 2:

```
{
  name: shirt
  Type: cotton
  Gender: male
}
```

3) Search Consumer:

This service consumes newly added items data to form Elastic search cluster.

Why Elastic Search cluster??

ESC helps in Fuzzy search. Along with that even if we provide item description we can land up with the searching

4) Searching Service:

This service is an interface b/w the user search page & Elastic Search cluster

Role: Listens to all user Rest API calls like GET /search="Xvideos" etc...

Role 2: Before populating search results, search service communicates with USER SERVICE to get default address or current location details

Role 3: It also communicates with Servicability Service to get the information of all warehouses

* Segregate if warehouse delivers to the user address or not

Role 4: It gets information related to ESTIMATED TIME OF DELIVERY FROM ETD Service

Role 5: It filters out all items that cannot be delivered and display just items that can be delivered to the user location.

Role 6: It informs KAFKA about the search information of the user so that the information can be used in populating items in Recommendation service

Role 7: Once List of Search items are populated, The user can add to wishlist or add to cart

Therefore, searching service acts as intermediatary for them.

5) CART SERVICE:

This service helps in adding items to cart & send information to kafka about being carted so that it can be used for creating recommendations

Mysql DB cluster is used to store cart items details since data is structured

6) Wish list Service: Similar to cart service

7) Ware House Service:
It provides all information related to items in every warehouse to Serviciability service

8) Logistics Service:
It provides available logistics (road, air, train) to Estimation of delivery Time service

9) Spark Streaming cluster:
It distributedly data process on information from searching service, cart service and wishlist service to form Hadoop cluster

Why Hadoop cluster: Huge data MoFfos

10) Spark jobs:
ON Hadoop cluster, perform spark jobs to come up with best recommendations and share with recommendation service

11) Recommendation service:

It gets recommendation list from spark jobs and store it to cassandra cluster.

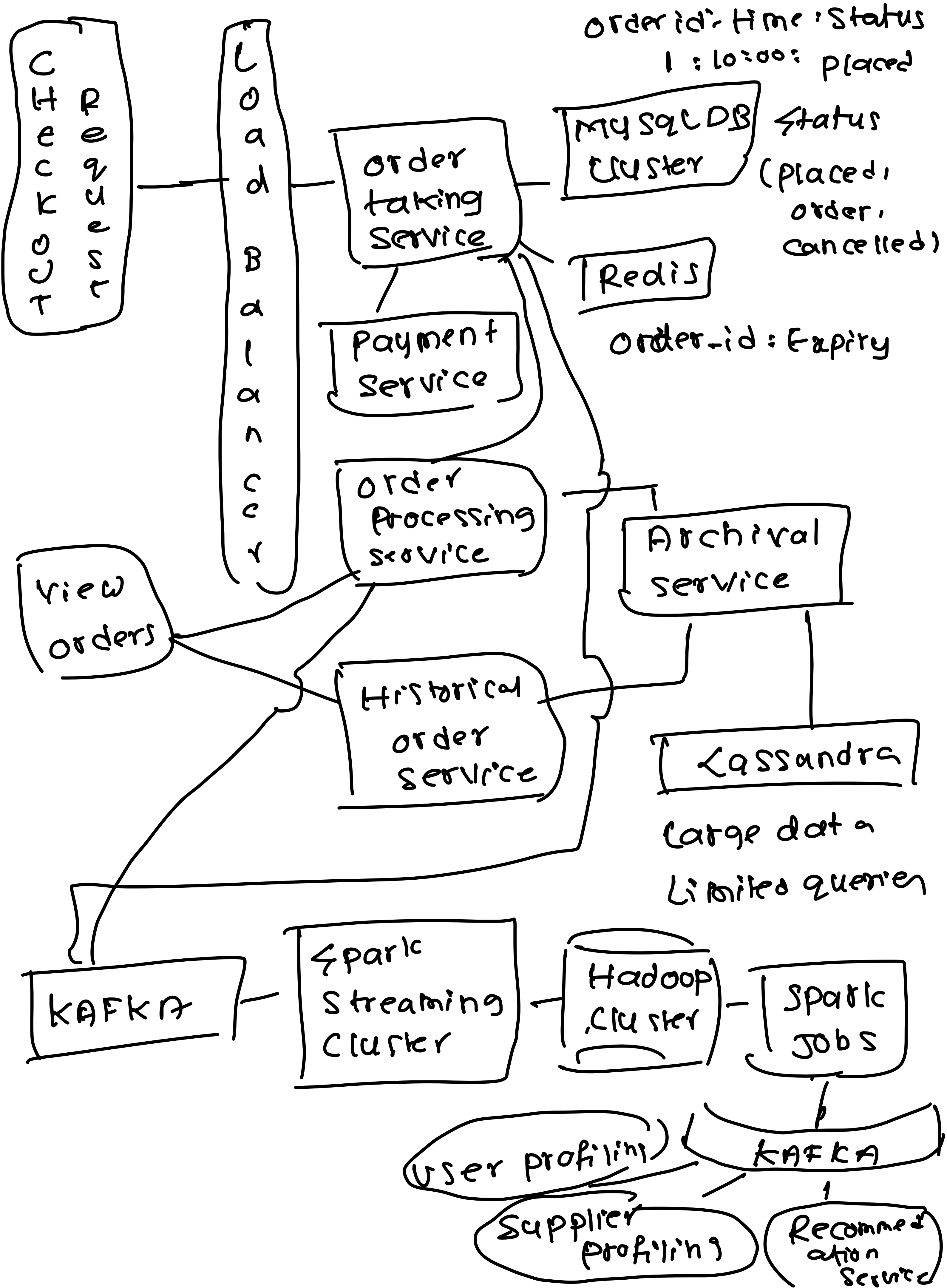
Why Cassandra?? Limited queries on huge data. The best option is cassandra.

When user opens the app

If old user - based on searching history, cart history and wishlist history display items

If new user - based on popular items bought in the last few hours for each category like electronics, sports etc..
display them.

ARCHITECTURAL DIAGRAM : checkout



(2) ORDER TAKING SERVICE:

While user placing the order, this service hits first.

Role: update Redis.

update Redis with orderid, expirytime

Role 2: update mysql cluster with status, and other details

Status: Placed, ordered, Cancelled

order-id, expiry-time, Status

Why Mysql Cluster:

The changes involve multiple tables

1) User Table
2) Inventory Table
3) Supplier Table
4) order Table

All these changes must follow ACID. Therefore Mysql cluster so that we can do all of them in a Transaction If not we will revert Transaction

Case 1: Payment successful

Delete the record from Redis and update status in Mysql cluster ; before this decrement itemcount in inventory

1, 1000, placed (Mysql)

Case 2: Payment failed before/after Expiry

update Mysql cluster

1, 1000, cancelled

Revert or increment itemcount in the inventory

Inform about this to kafka

Case 3: payment successful followed by expiry

To eliminate any issues. once payment is completed deleted record from redis and update mysql

Case 4: Expiry followed by payment sucess

1) Refund amount, update inventory, inform kafka

2) Place New order, inform kafka

13) ORDER PROCESSING SERVICE:

If we don't move completed orders out of Mysql, It will become bottleneck in future cuz of huge no-of records.
Therefor, with the help of this service we move completed orders out of it to Cassandra with the help of Archival servic

14) Archival service:

Copy Completed order records to Historical Records service and ask order processing service to delete them

15) Historical Records service:

push completed order to Cassandra

16) View orders:

Get on-going orders from order processing service and historical orders from Historical Records service

Once order is placed. push the information to Kafka so that it can be used for analytical services.

1) Identifying premium customer
2) Identifying .and rating suppliers
3) Improve recommendation list
   for the user