ExperimentNo:2

StudentName: Karan
UID:23BCS11811
Branch:BE-CSE
Section/Group:KRG3-A
Semester: 6
Date of Performance:15-01-2026
SubjectName: System Design
SubjectCode:23CSH-314

1- Aim - Design an Online shopping platform similar toAmazon / Flipkart that will allow users to purchase mobiles, laptops, cameras, clothes etc.

2- Requirements:Functional&Non-Functional

A-Functional Requirement ○User should be able to search and find the products based on product title or names.

- User should be able to view the details of the product llike description, image, available quantity, review.
- Usershouldbeabletoselectthequantityandmovetheproduct/itemintothecart.○ Usershouldbeabletomakethepaymentandshouldbeabletoperformthecheckout.
  ○ Usershouldbeabletocheckthestatusoftheorder.
- Systemshouldbeabletomanagepurchaseofitemshavinglimitedstocks.

  Race condition happens during a flash sale when inventory has limited stock, and the system must handle multiple transactions occurring at the same time.

B- Non-Functional Requirement ○The system is designed for 100 million daily active users with around 10 orders handled per second. ○Consistency &Availability: Based on the target scale, both are required, but at different system levels.

i. According to functional requirements, users must search products smoothly, so product search needs high availability.
ii. Strong consistency is required for critical components such as payment processing, order placement, and inventory management.

- Expectedresponsetimeisapproximately200ms.
- Scalingwillbedone eitherHorizontallyorverticallywhereverapplicable.

3- Core-entities of System ○User/Client ○Products ○Cart ○Orders○Checkout followed by Payment

4- API endpointcreationa)GETAPICall:Prod_Search

Https://Local_Host/products/search_item = {Search_keywords}

HTTP Req

{
GET:<iPhone16>

}
HTTPRes
{
List<ProductID:iPhone>
}

Now, on front-endif multiple data of respective product is coming in that case the FE becomes faulty thus ultimately increasing the Latency.
So wewillbeusingPagination(1,2,3,….next)

b) GETAPICall:ViewProductDetails
Https://Local_Host/products/{product_id}

HTTPReq
{
GET:<Product_id=17>
}
HTTPRes
{
Product_id=17,
Name: iPhone17,
Color:NavyBlue,
Price: $1009,
Image_URL:URL_image
}

c) POSTAPICall:Itemaddincart
Https://Local_Host/cart/add_products

HTTPReq
{
Product_id:17,
Product_id:16
}
HTTPReqHeader
{
User_id:04

}
HTTPRes
{
Cart_id:101
}

d) PUTAPICall:Toupdateanyorderinthecart

e) DELETEAPICall:Toremove anyitemfromthecart

f) POSTAPICall:forcheckout&Payement
Https://Local_Host/checkout->{postbody}

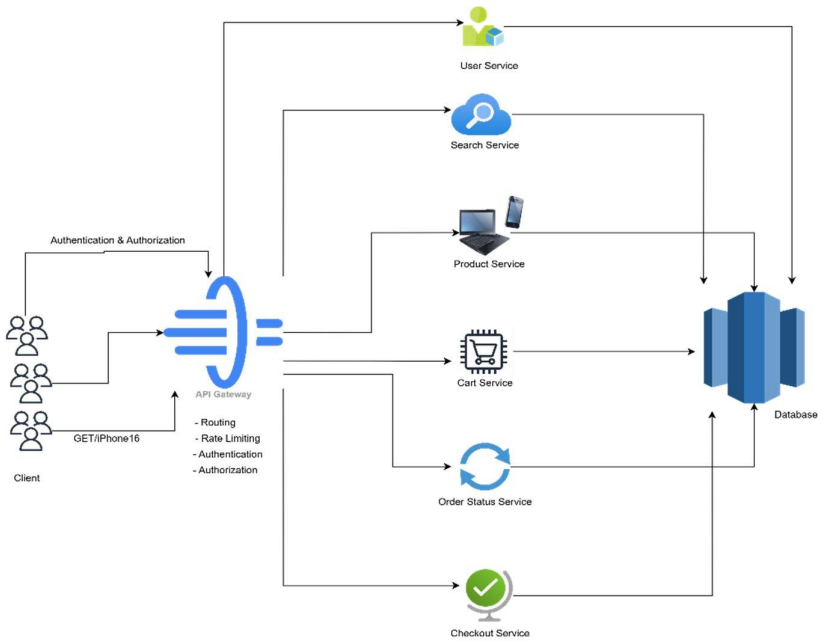
HTTPReq
{
AllProductId's,
Total Quantity,
Total Price
}
HTTPRes
{
Order_id
}

Https://Local_Host/payment -> {post body}
HTTP Req
{
Order_id,
Payment Type,
Payment_Mode
}
HTTPRes
{
Confirmation_Status:Succes/Fail
}

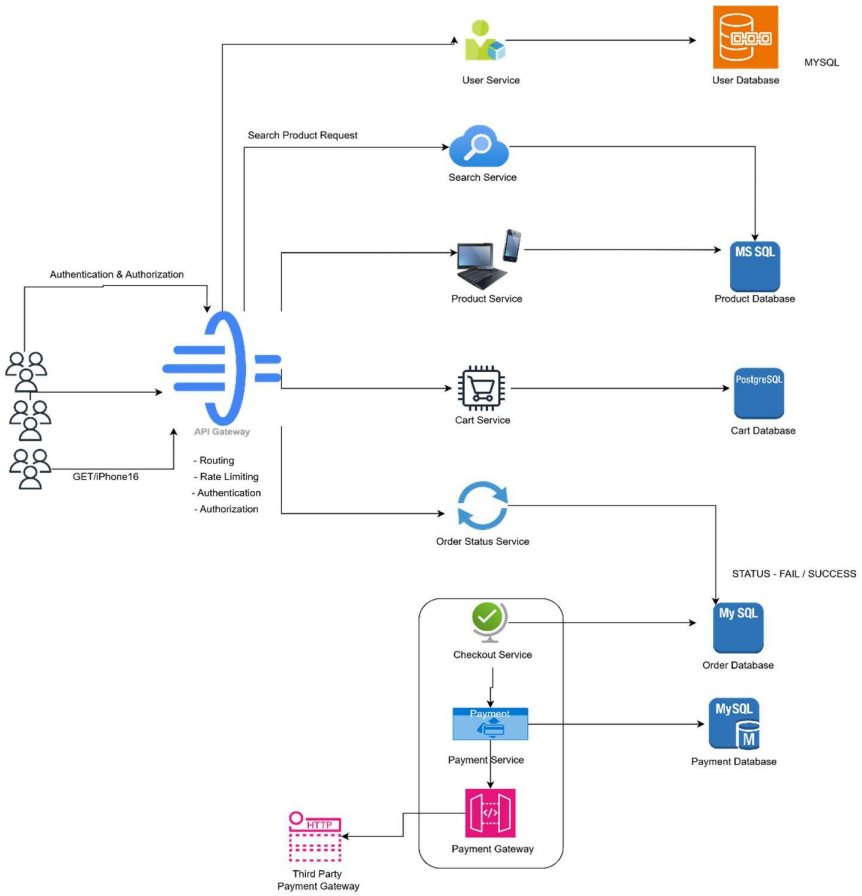
g) GETAPICall:OrderStatus
Https://Local_Host/orde_status={order_id}

5- High-LevelDesign

NowAccording to the functional requirement of the system, wecanidentify that :We have to follow a distributed / micro-services approach not the monolithic one.



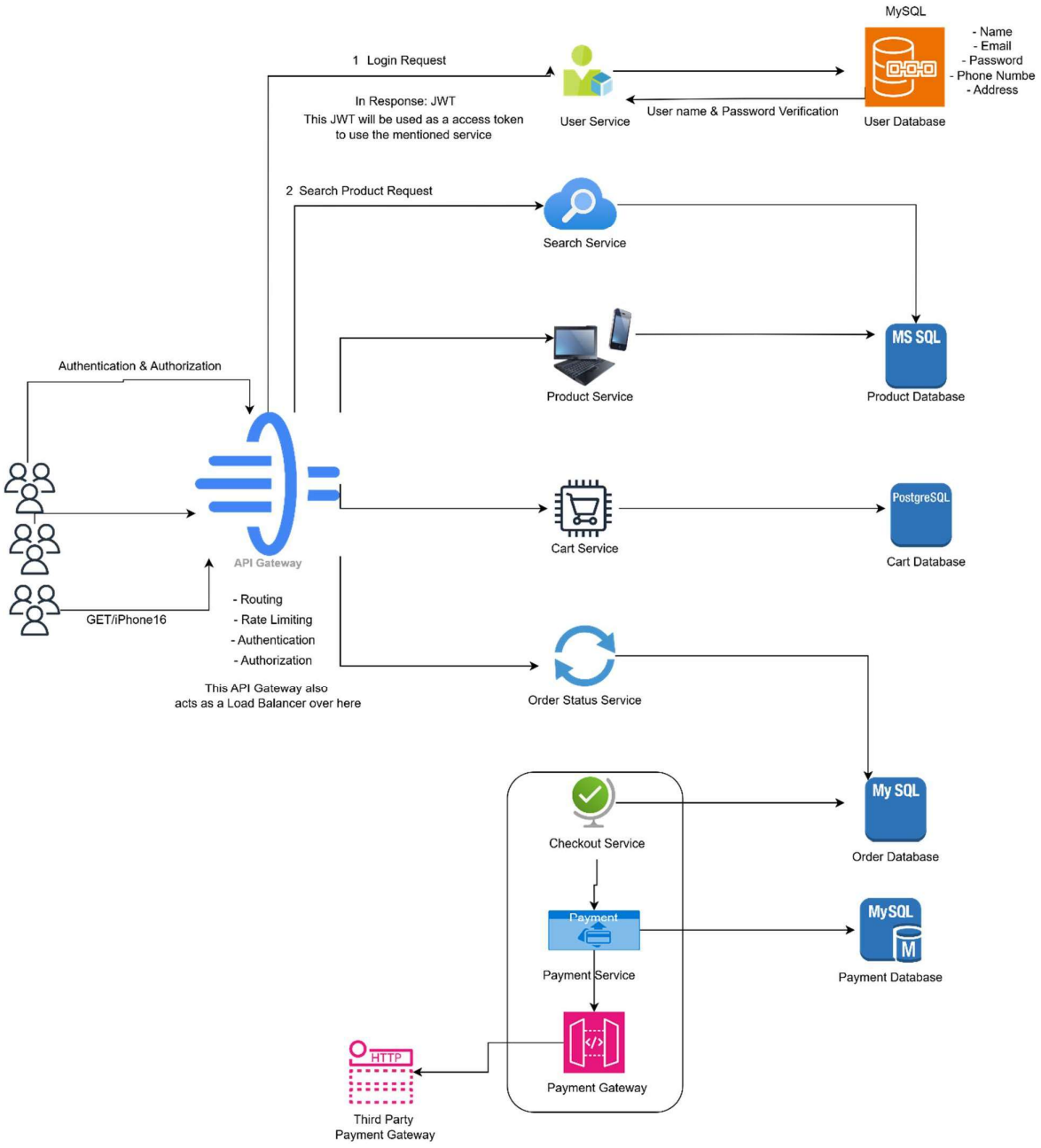
Drawbacks: Multiple-Database calls are being done and single database is being used to handle every service which increases the latency.



This will fullfill all the functional requirements that were listed. Now, we will see the internal implementations of each one of these components in LLD.

6- Low-LevelDesign

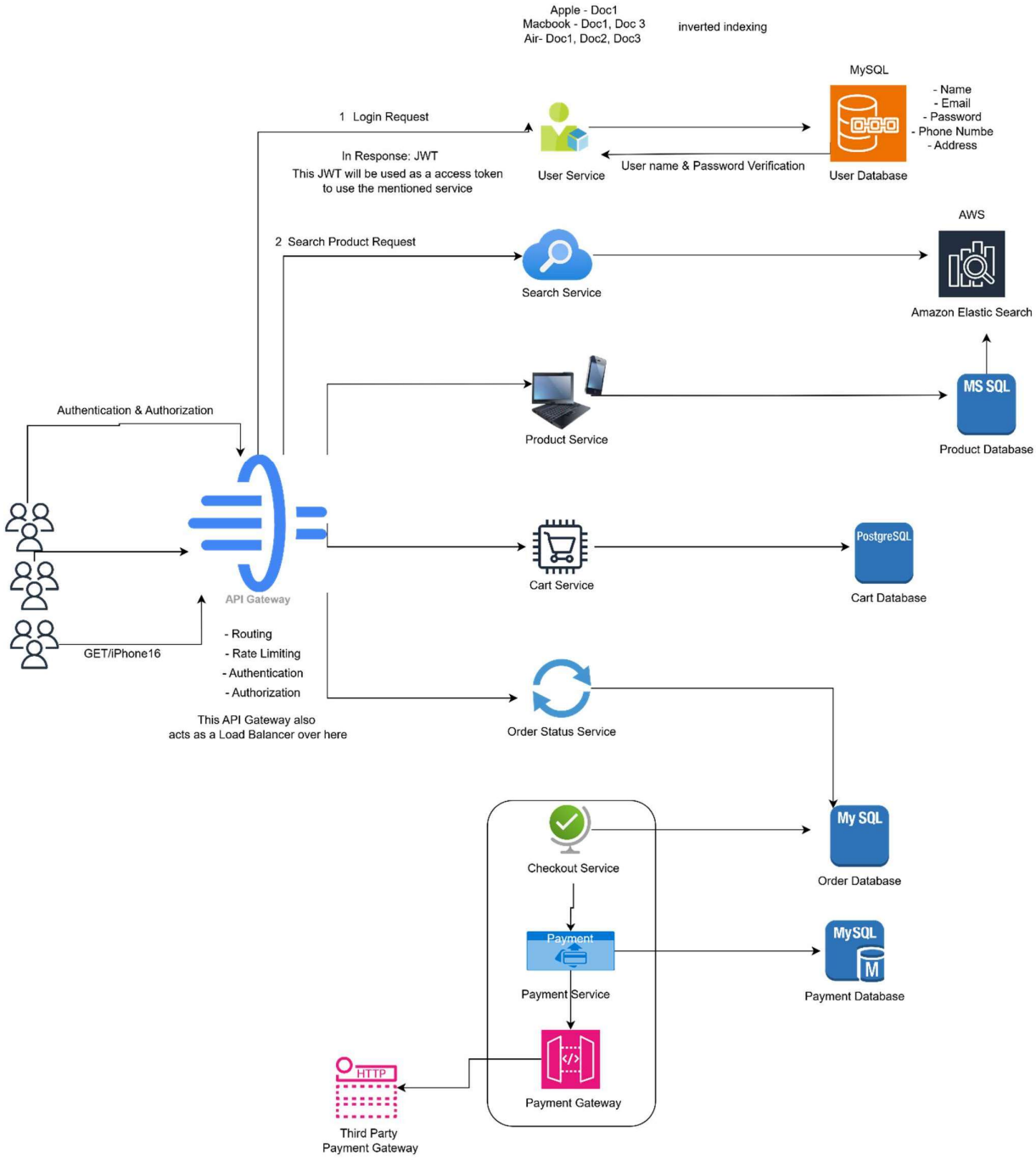
1- UserLoginandSearchFunctionality



Drawbacks-

- As per NFR, 10 million DAU, which means searching the entire DB is very nonoptimized here
- As a solution we can implement INDEXING here, but still database scanning is not prevented.
- O(n)– time

Solutiontosearchfunctionalityproblem:**ELASTICSEARCH.**


Apple - Doc1
Macbook - Doc1, Doc 3
Air- Doc1, Doc2, Doc3
inverted indexing
MySQL
- Name
- Email
- Password
- Phone Numbe
- Address
1 Login Request
In Response: JWT
This JWT will be used as a access token
to use the mentioned service
User Service
User name & Password Verification
User Database
AWS
2 Search Product Request
Search Service
Amazon Elastic Search
Authentication & Authorization
Product Service
MS SQL
Product Database
API Gateway
Cart Service
PostgreSQL
Cart Database
GET/iPhone16
- Routing
- Rate Limiting
- Authentication
- Authorization
This API Gateway also
acts as a Load Balancer over here
Order Status Service
Checkout Service
My SQL
Order Database
Payment
Payment Service
MySQL
Payment Database
HTTP
Payment Gateway
Third Party
Payment Gateway

Elasticsearch is a **Search Engine**, not a traditional database.
It is built on top of a library called **Apache Lucene.**

Document ID	Content (Stored as a String)
Doc 1	"Apple iPhone 15 Pro"
Doc 2	"Samsung Galaxy S23"
Doc 3	"Apple MacBook Air"

Elastic Search

Tokenization

Word (Token)	Document List (Occurrences)	Frequency (Count)
Apple	Doc 1, Doc 3	2
iPhone	Doc 1	1
15	Doc 1	1
Pro	Doc 1	1
Samsung	Doc 2	1
Galaxy	Doc 2	1
S23	Doc 2	1
MacBook	Doc 3	1
Air	Doc 3	1

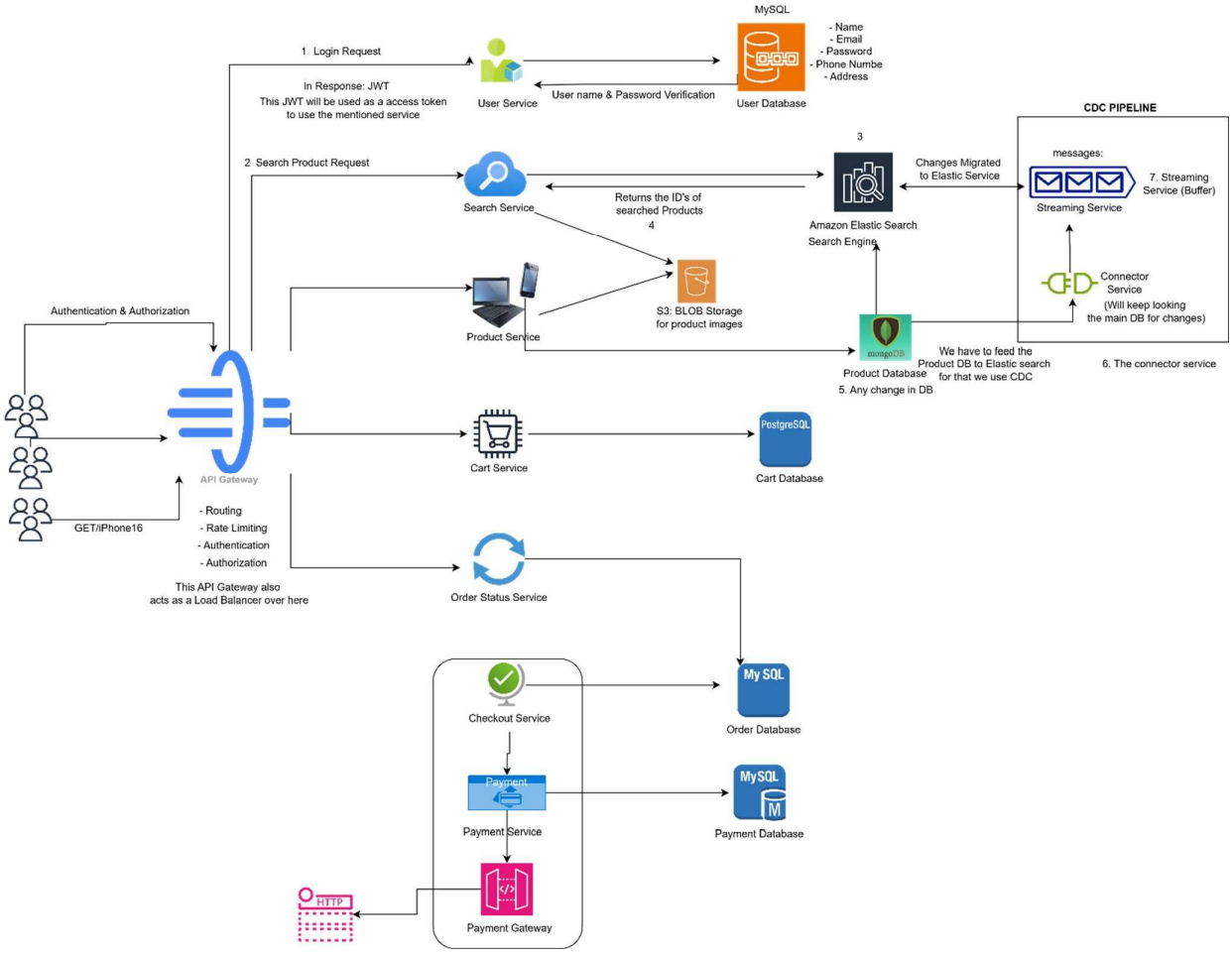
Multi-term Logic

Word	Document IDs
Apple	{Doc 1, Doc 3}
Macbook	{Doc 3}

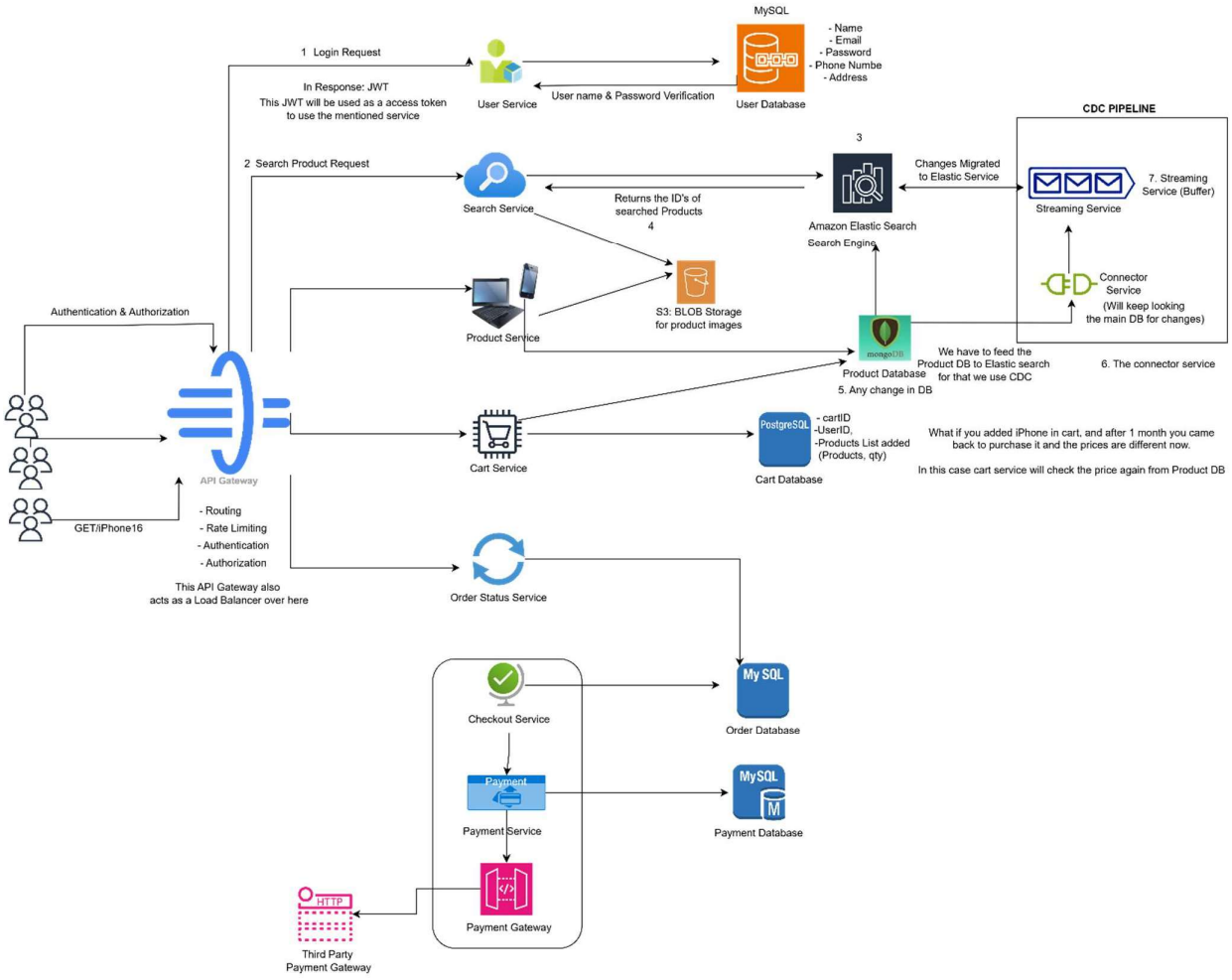
o(1)

```
search_product_call {
    1. RECEIVE: search_term (e.g., "shoe")
    2. ANALYZE: Break term into lowercase tokens.
    3. QUERY: Ask Elasticsearch Inverted Index for "shoe".
    4. RANK: Get IDs of products containing "shoe" sorted by relevance.
    5. FETCH: Get full product details from the main DB using those IDs.
    6. RETURN: Fast, accurate results to the user.
}
```

We will use CDC (Change Data Capture) to send data from original database to ES in real time.



2- CartService



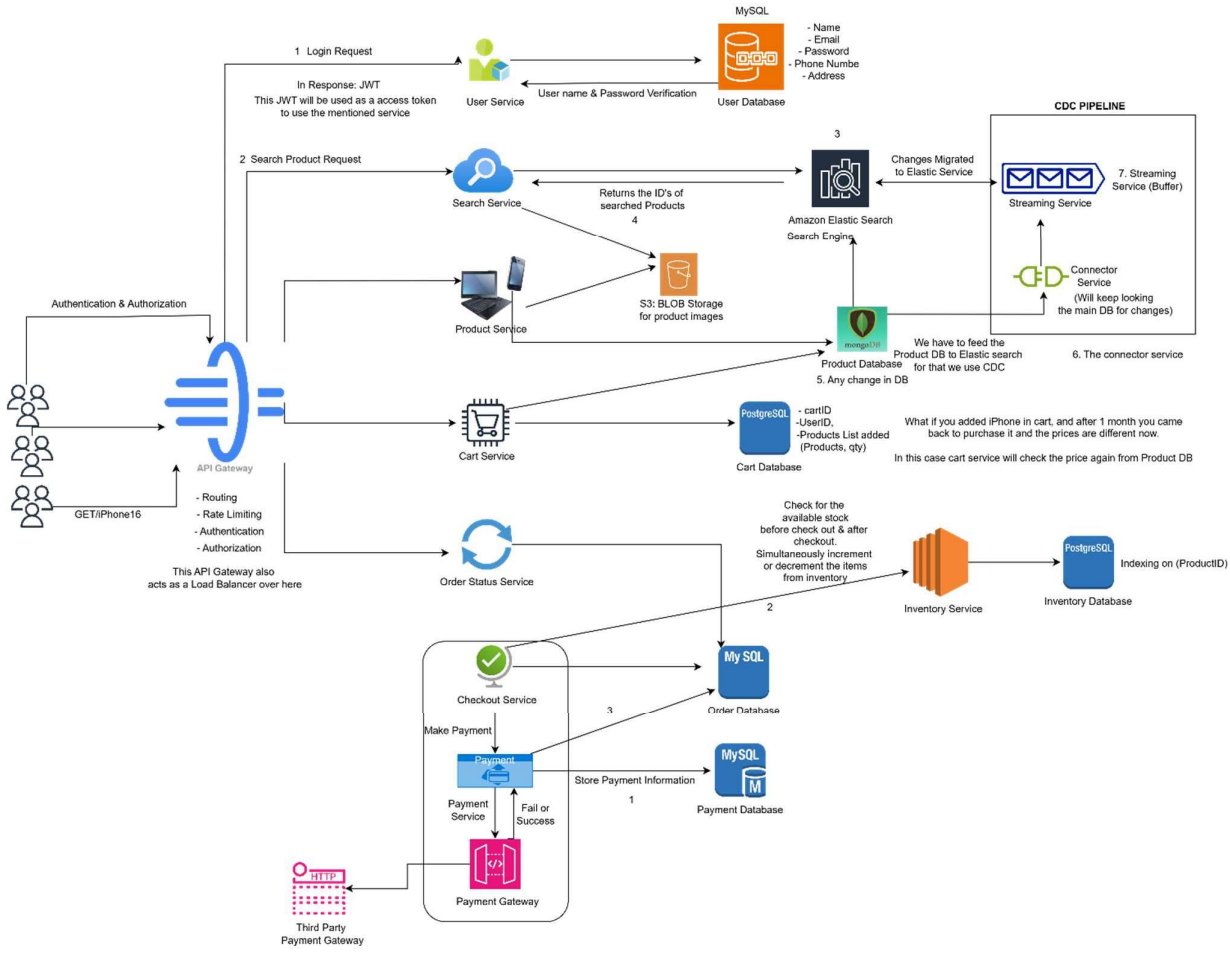
3- CheckoutService

For a user to do the checkout, the quantities in real-time from the DB should be verified, i.e., whether we have the requested amount of qty available in the inventory or not?

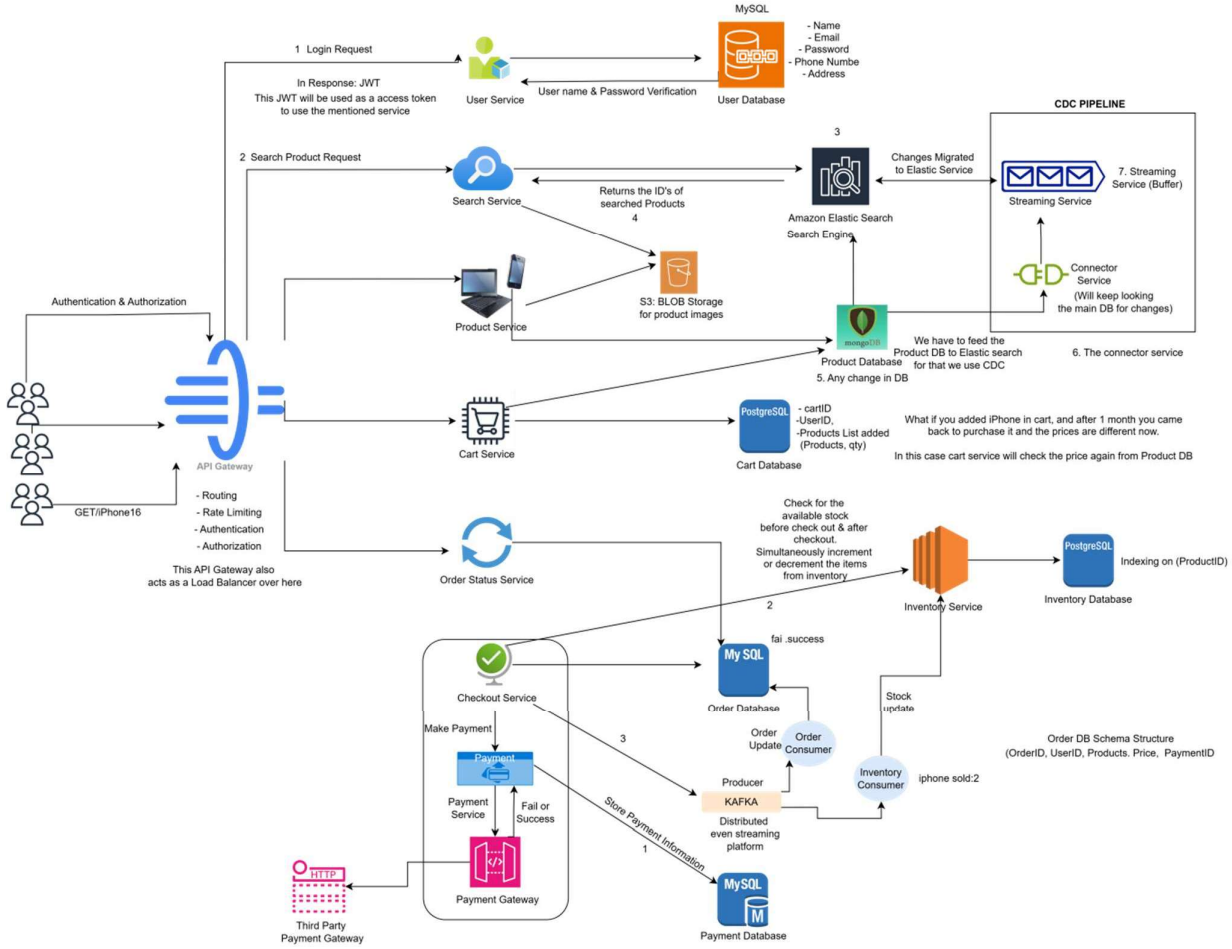
For that we will use a seperate service:Inventory Service:For Concurrecy DRAWBACKS

–

- 3APIcallsaretheretoperformasingletransaction
- Now For this single transaction, it can happen that payment done successfully, inventory was not updated.
  This isaverycriticalissuew.r.tconsistency
- Rateoffailureisveryhighoverhere



Solutions for above drawbacks is to introduce **Producer-Consumerarchitecture** using KAFKA.


MySQL
- Name
- Email
- Password
- Phone Numbe
- Address
1 Login Request
In Response: JWT
This JWT will be used as a access token to use the mentioned service
User Service
User name & Password Verification
User Database
CDC PIPELINE
3
2 Search Product Request
Search Service
Returns the ID's of searched Products
4
Changes Migrated to Elastic Service
Amazon Elastic Search
Search Engine
Streaming Service
7. Streaming Service (Buffer)
Connector Service
(Will keep looking the main DB for changes)
Authentication & Authorization
Product Service
S3: BLOB Storage for product images
mongoDB
Product Database
We have to feed the Product DB to Elastic search for that we use CDC
6. The connector service
5. Any change in DB
API Gateway
- Routing
- Rate Limiting
- Authentication
- Authorization
This API Gateway also acts as a Load Balancer over here
GET/iPhone16
Cart Service
PostgreSQL
- cartID
-UserID,
-Products List added (Products, qty)
Cart Database
What if you added iPhone in cart, and after 1 month you came back to purchase it and the prices are different now.
In this case cart service will check the price again from Product DB
Order Status Service
Check for the available stock before check out & after checkout.
Simultaneously increment or decrement the items from inventory
2
Inventory Service
PostgreSQL
Indexing on (ProductID)
Inventory Database
fai .success
Checkout Service
My SQL
Order Database
Stock update
Make Payment
3
Order Update
Order Consumer
Order DB Schema Structure
(OrderID, UserID, Products. Price, PaymentID
Payment
Payment Service
Fail or Success
Producer
KAFKA
Distributed even streaming platform
Inventory Consumer
iphone sold:2
Store Payment Information
1
HTTP
Payment Gateway
MySQL
Payment Database
Third Party Payment Gateway