Experiment - 2

Name: Anmolpreet Singh **UID:** 23BCS11616
Branch: BE-CSE **Section/Group:** KRG-1A
Semester: 6th **Date of Performance:** 12/01/26
Subject Name: System Design **Subject Code:** 23CSH-314

1. Aim:

To design and analyze an E-commerce System by identifying its functional and non-functional requirements and representing the system architecture using a draw.io diagram.

2. Objectives:

1. To gain insight into how an E-commerce system.
2. To determine the functional requirements of the system.
3. To determine the non-functional requirements such as performance and scalability.
4. To design a high-level system flow using draw.io.
5. To design a DB schema for the system using ERD.
6. To enhance understanding of practical system design principles.

3. Procedure-

- Studied real-world E-commerce platforms such as Amazon and Flipkart.
- Identified core components including users, products, carts, orders, inventory, and payments.
- Listed functional requirements required for smooth E-commerce operations.
- Analyzed non-functional requirements like low latency, scalability, and fault tolerance.
- Designed a structured system design diagram using draw.io.
- Designed a DB schema using DB scripts and ERD.
- Reviewed the design to ensure scalability, data consistency, and clarity.

4. Functional Requirements -

- User sign up and login authentication
- Browse and search products by category
- Adding products to the shopping cart
- Update or remove items from cart
- Place orders and process payments

- Manage inventory and product availability.
- Viewing previous order details.
- Provide and submitting product reviews and ratings.

5. Non-functional Requirements

- Fast response time with latency less than 300 ms.
- High system availability with 99.9% uptime.
- Scalability to support millions of concurrent users
- Strong consistency for inventory updates and payment processing
- Secure storage and handling of user and payment data

6. High Level Design (HLD)

The system architecture includes Client, Frontend, Backend Services (Product, Cart, Order, Payment, Inventory), and Database layers. High availability is maintained for search services, while strict consistency is ensured for payment and inventory modules. The design supports both horizontal and vertical scalability.

7. LowLevel Design (LLD)

Product Module

- Stores product information and stock details

- Supports keyword-based search with pagination.

Cart Module

- Maintains a cart specific to each user

- Supports add, update, and delete operations.

Order and Payment Module

- Generates order ID after checkout.

- Confirms payment and updates order status.

Inventory and Race Condition Handling

- Controles limited stock availability

  .

- Uses locking or transactions during flash sales

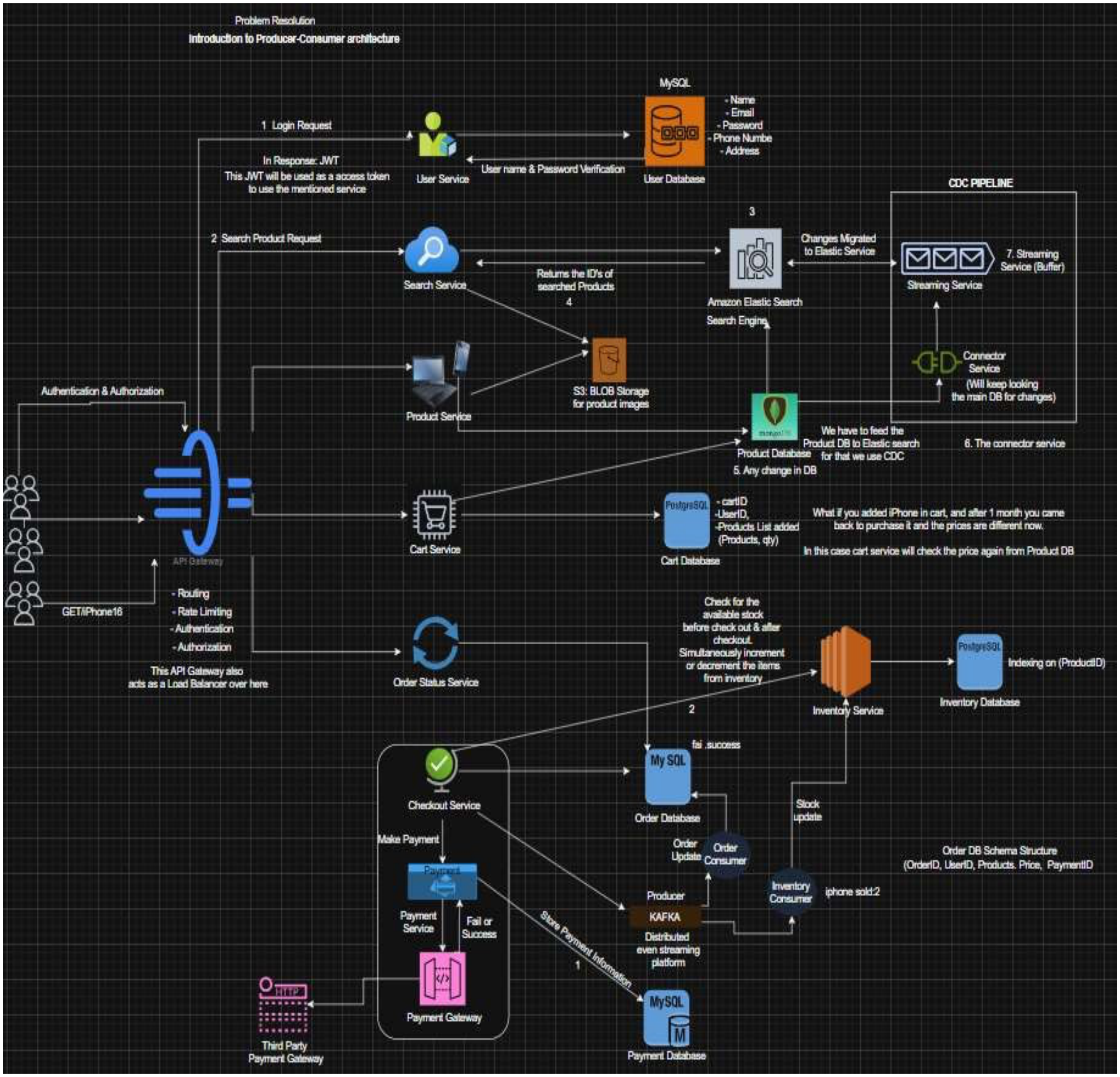
8. Outcome / Result -

- Successfully created an E-commerce system design using draw.io

- Clearly identified both functional and non-functional requirementsUnderstood scalability and consistency challenges in large-scale systems.

- Gained an understanding of scalability and consistency challenges in large-scale systems

REQUIRED SYSTEM DESIGN –


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