DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
Discover. Learn. Empower.

Experiment No: 2

Student Name: Mohammad Kaif **UID: 23BCS12884**
Branch: B.E./C.S.E. **Section/Group: KRG-2A**
Semester: 6th **Date of Performance: 14/01/2026**
Subject Name: System Design
Subject Code: 23CSH-314

1. **Aim:** To design an online E-commerce platform similar to Amazon/Flipkart for browsing and purchasing products like mobiles, laptops, cameras, and clothes. To implement Kafka, Elasticsearch, and a CDC pipeline for real-time data processing, fast search, and scalability.

2. **Objective:**

- To build a scalable online shopping system where users can view products, search products, and place orders.
- To use **Apache Kafka** to send and receive real-time events between services.
- To use **Elasticsearch** to provide fast and accurate product search.
- To use a **CDC pipeline** to keep databases and services updated in real time.

3. **Tools Required:**

- **Programming Language:** Java / Python / JavaScript
- **Backend Framework:** Spring Boot / Express.js / Flask
- **Database:** MySQL / PostgreSQL / MongoDB
- **API Testing Tool:** Postman
- **Design Tool:** Draw.io (for system diagrams)
- **Web Browser**
- **Elasticsearch** (for fast search)
- **Apache Kafka** (for real-time events)
- **CDC Connector** (for database change tracking)

4. SYSTEM DESIGN / SYSTEM SPECIFICATION

4.1 Functional Requirements

a. User should be able to **search products** using product name or title.

b. User should be able to **view product details** such as description, image, price, available quantity, and reviews.

c. User should be able to **select quantity** and **add products to cart**.

d. User should be able to **checkout and make payment**.

e. User should be able to **check order status** after placing an order.

f. System should **manage limited stock** and prevent over-ordering.

4.2 Non-Functional Requirements

a. **Target Scale:**
   i. 100 million Daily Active Users (DAU)
   ii. Around 10 orders per second

b. **Availability:**
   i. System should be available **24/7**

c. **Consistency & Availability:**
   i. Both are required
   ii. **Strong consistency** for orders and payments
   iii. **High availability** for product search and browsing

d. **Latency:**
   i. Response time should be **around 200 ms**

e. **Scalability:**
   i. Support **horizontal scaling** (preferred)
   ii. Vertical scaling can be used when needed

4.3 Core Entities of the System

a. User / Client

b. Product

c. Cart

d. Order

e. Checkout & Payment

5. API Designing:

1. GET API Call: Prod_Search

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

HTTP Req

GET: <iPhone 16>

HTTP Res

<ProductID:iPhone>

Now, on front-end if multiple data of respective product is coming in that case the FE becomes faulty ->

ultimately increasing the LATENCY.

For that: we can use Pagination 1, 2 ,3 ,4, ----- SO ON

2. GET API Call: View Product Details

Https://Local_Host/products/{product_id}

HTTP Req

GET: <Product id =17>

HTTP Res

Product id:17,

Name: iPhone 17,

Color: Navy Blue,

Price: $1099,

Image Thumbnail: URL Image

3.POST API Call: Item add in cart

Https://Local_Host/cart/add_products

HTTP Req

Product_id = 17,

Product id = 16

HTTP Req Header

User id: 04

HTTP Res

Cart id: 101

4. PUT API Call: To update any order in the cart

Let's Suppose you want to add one more product into the cart.

5.DELETE API Call: To remove any item from the cart

Let's Suppose you want to delete one more product into the cart.

6.POST API Call: for check out & Payement

Https://Local_Host/checkout -> {post body}

HTTP REQ

All products ID's,

Total Quantity,

Total Price

HTTP RES

Order ID

6. HLD (High Level Design):