# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

## Experiment-2

**Student Name:Astik Joshi** **UID:** 23BCS10627

**Branch:** CSE **Section/Group:** KRG-3B

**Semester:** 6th **Date of Performance:** 14/01/26

**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 develop a scalable online shopping system for product listing, search, and order management.
- To use Apache Kafka for real-time event streaming and inter-service communication.
- To implement Elasticsearch for fast and efficient product search.
- To integrate a CDC pipeline for real-time synchronization between databases and services.

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 HLD diagrams)
- Web Browser
- ElasticSearch
- Kafka
- CDC connector

## 4. SYSTEM DESIGN / SYSTEM SPECIFICATION:

### 4.1. Functional Requirements:

• The platform should allow users to search products using keywords such as product name, brand, or title.

• Users should be able to open a product page to see complete information including description, images, price, available stock, and customer reviews.

• The system should allow users to choose the required quantity of a product and add it to their shopping cart.

• Users should be able to proceed to checkout and complete payments using supported payment methods.

• After placing an order, users should be able to track the current status of their order (placed, shipped, delivered, etc.).

• The system must handle inventory efficiently and prevent users from ordering products that are out of stock or have limited availability.

### 4.2. Non-functional Requirements:

• **Scalability Target:**
The system should support up to **100 million daily active users (DAU)** and handle approximately **10 orders per second** without performance degradation.

• **Availability:**
The application must be operational **24/7** with minimal downtime.

• **Consistency vs Availability:**
Different components may have different requirements: Order and payment services require **strong**

• **Latency:**
The system should respond to user requests within **~200 milliseconds** for a smooth user experience.

• **Scalability Approach:**
The architecture should primarily support **horizontal scaling**, with vertical scaling used only where necessary.

### 4.3. Core-Entites of the System:

- **User** – Represents customers interacting with the platform
- **Product Catalog** – Stores product-related information
- **Shopping Cart** – Temporarily holds items selected by the user
- **Order** – Represents confirmed purchases
- **Checkout & Payment** – Handles billing and transaction processing

### 4.4. API Endpoints Creation:

1. **GET API Call – Product Search**

   Purpose:
   This API allows users to search products using keywords such as product name or category.
   Endpoint:
   https://localhost/products/search
   HTTP Request:
   Search keyword = “Wireless Headphones”
   HTTP Response:
   The system returns a list of matching product IDs such as 45, 52, and 60.
   Note:
   If multiple products match the search term, sending all data at once may increase frontend load and latency. To avoid this, pagination (Page 1, Page 2, Page 3, etc.) is applied so that limited data is sent per request.

2. **GET API Call – View Product Details**

   **Purpose:**
   **Fetches complete details of a selected product.**
   **Endpoint:**
   https://localhost/products/{product_id}

   HTTP Request:
   Product ID = 45

   HTTP Response:
   Product ID: 45
   Product Name: Noise Cancelling Headphones
   Brand: SoundMax
   Color: Matte Black
   Price: 7,999
   Available Quantity: 25
   Product Image: Image URL

3. **POST API Call – Add Product to Cart**

   **Purpose:**

   **Allows users to add selected products to their shopping cart.**

   Endpoint:

   https://localhost/cart/add

   HTTP Request Body:

   Product IDs: 45, 60

   HTTP Request Header:

   User ID: 12

   HTTP Response:

   Cart ID: 208

4. **PUT API Call – Update Cart**

   **Purpose:**

   **Used to update the quantity of products already present in the cart.**

   **Example:**

   **If a user wants to increase the quantity of headphones from one unit to two units, this API is used.**

5. **DELETE API Call – Remove Item from Cart**

   **Purpose:**

   **Removes a selected product from the shopping cart.**

   **Example:**

   **If a user decides not to purchase a power bank, this API removes it from the cart.**

6. **POST API Call – Checkout and Payment**

   Checkout API

   Purpose:

   Confirms the order before payment.

   Endpoint:

   https://localhost/checkout

   HTTP Request:

   Selected Product IDs: 45, 60

   Total Quantity: 3

   Total Price: 14,997

   HTTP Response:

   Order ID: 9021

Payment API
Purpose:
Processes payment for the placed order.
**Endpoint:**
https://localhost/payment
HTTP Request:

Order ID: 9021
Payment Type: UPI
Payment Mode: Online
HTTP Response:
Payment Status: Success or Failed

7. **GET API Call – Order Status**

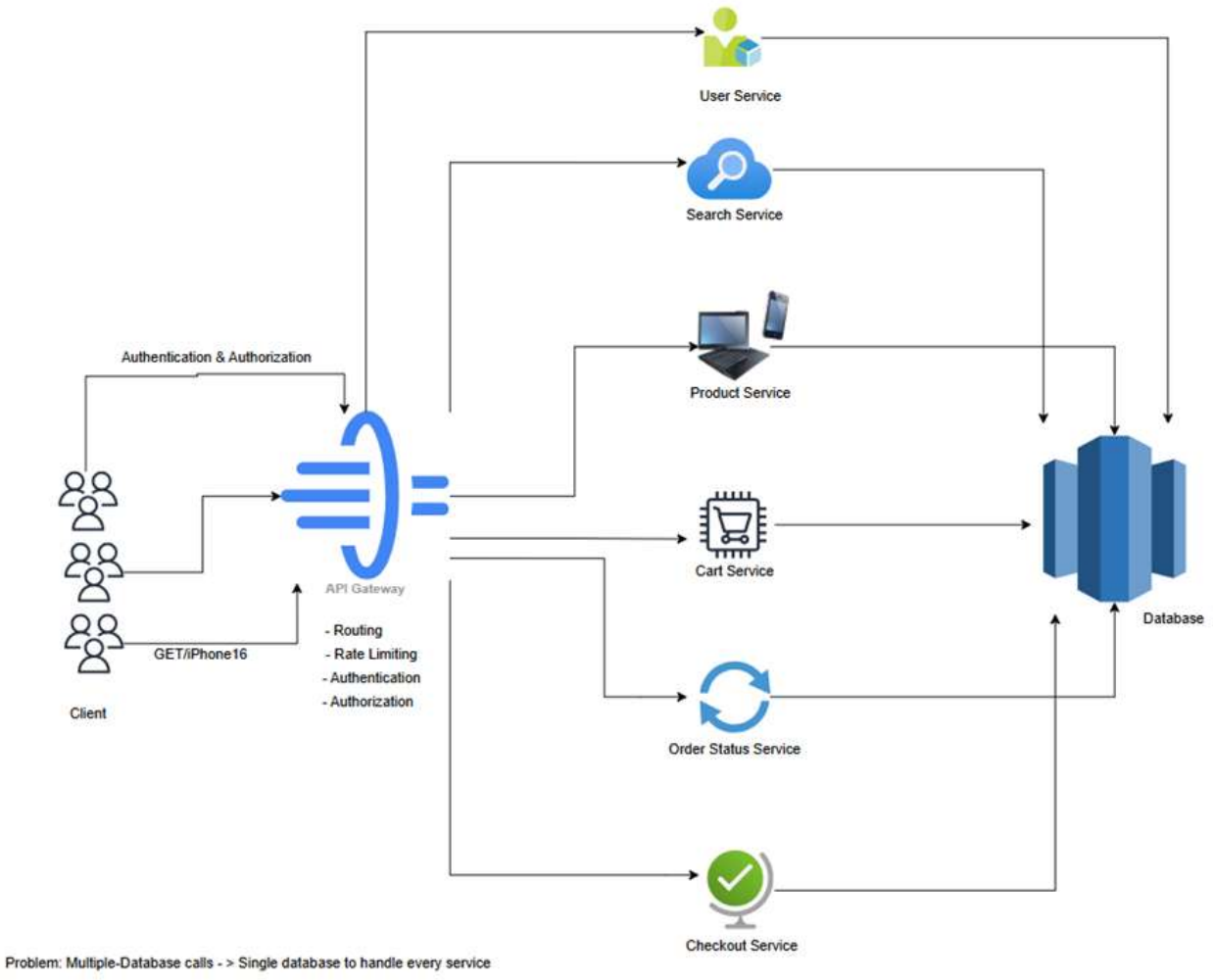
**Purpose:**
**Allows users to track the current status of their order.**

**Endpoint:**
**https://localhost/order/status**

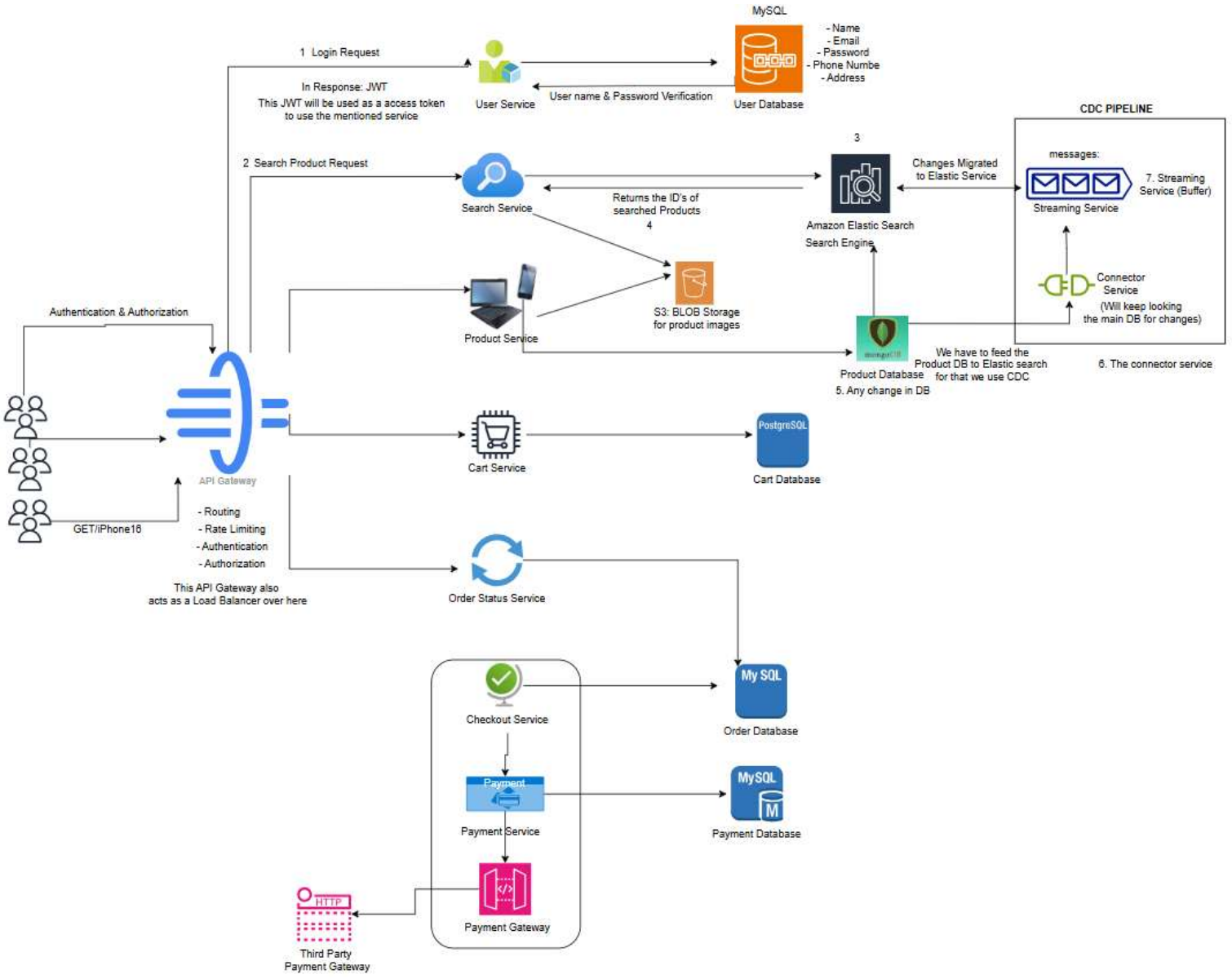
HTTP Request:
Order ID = 9021
HTTP Response:
Order Status: Out for Delivery

**5. HLD(High Level Design):**

We have to follow a distributed / micro-services approach not the monolithic one.




Problem: Multiple-Database calls - > Single database to handle every service


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
messages:
7. Streaming
Service (Buffer)
Streaming Service
Amazon Elastic Search
Search Engine
Connector
Service
(Will keep looking
the main DB for changes)
Authentication & Authorization
Product Service
S3: BLOB Storage
for product images
We have to feed the
Product DB to Elastic search
for that we use CDC
Product Database
5. Any change in DB
6. The connector service
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

**7. Learning Outcomes**

• Gain a clear understanding of how large-scale E-commerce systems are designed and structured.

• Learn how real-time event streaming works by using Apache Kafka for communication between services.

• Understand how search engines like Elasticsearch improve product discovery with low response time.

• Acquire knowledge of Change Data Capture (CDC) mechanisms to keep databases and services synchronized in real time.

• Build practical knowledge of integrating multiple distributed components to achieve high availability, fault tolerance, and system scalability.