Experiment 2

Student Name: Deepanshu Mandhyan **UID: 23BCS12327**
Branch: CSE **Section/Group: KRG 3-A**
Semester: 6th **Date of Performance:15/01/2026**
Subject Name: System Design **Subject Code: 23CSH-314**

1. Aim:

To design and analyse a scalable e-commerce platform like Amazon or Flipkart that enables product search, cart management, order placement, payment processing, and order tracking while ensuring high availability, consistency, scalability, and low latency.

2. Objective:

- To identify the functional and non-functional requirements of an online shopping platform.
- To design a scalable, microservices-based architecture.
- To evaluate CAP theorem trade-offs across various system components.
- To design high-level architectures (HLD) for product search, cart, order management, payment, and inventory systems.
- To understand the role of Kafka, Elasticsearch, and CDC pipelines in large-scale distributed systems.
- To manage race conditions during flash sales and scenarios involving limited inventory.

3. Tools Used:

- **Draw.io –** Used for designing the system's high-level architecture (HLD).
- **PostgreSQL –** Used for relational database modelling and storage.
- **Elasticsearch –** Used for efficient product search and indexing.
- **Apache Kafka –** Used for event streaming and message processing**.**
- **CDC Pipeline –** Used to synchronize product data with Elasticsearch**.**

4. System Requirements:

A. Functional Requirements

- Users should be able to search for products using the product name or title.
- Users should be able to view detailed product information, including description, images, price, available stock, and reviews.
- Users should be able to choose the desired quantity and add products to their cart.
- Users should be able to complete checkout and successfully process payments.
- Users should be able to track the status of their orders after placement.
- The system should efficiently manage products with limited inventory.
- The system should support concurrent transactions during flash sales without causing overselling.

B. Non – Functional Requirements

- **Scalability**
  a. Target user base: up to 100 million daily active users (DAU)
  b. Order processing throughput: approximately 10 orders per second
- **Availability & Consistency**
  a. High availability is required for:
    i. Product search
    ii. Product listings
  b. Strong consistency is required for:
    i. Payment processing
    ii. Order placement
    iii. Inventory management
- **Latency**
  a. System response time should be within 200 ms
- **Reliability**
  a. The system should be fault-tolerant and capable of automatic recovery.
- **Scalability Approach**
  a. Support both horizontal and vertical scaling.

6. Database Schema:

- The database for the e-commerce platform is designed using an Entity–Relationship (ER) model. The ER diagram illustrates the key entities, their attributes, and the relationships necessary to support user management, product catalog management, cart operations, order processing, inventory control, and payment handling.

- The ER diagram is created using PostgreSQL (pgAdmin ERD Tool) based on tables defined with primary and foreign keys, ensuring strong referential integrity.

1. **Entity Relationships:**
   - A user can have multiple addresses (one-to-many).
   - A user can have a single cart (one-to-one).
   - A cart can include multiple products through cart items (many-to-many).
   - A user can place multiple orders (one-to-many).
   - An order can contain multiple products through order items (many-to-many).
   - Each product belongs to a single category (many-to-one).
   - Each product has one associated inventory record (one-to-one).
   - Each order has one corresponding payment record (one-to-one).
   - A product can have multiple images (one-to-many).

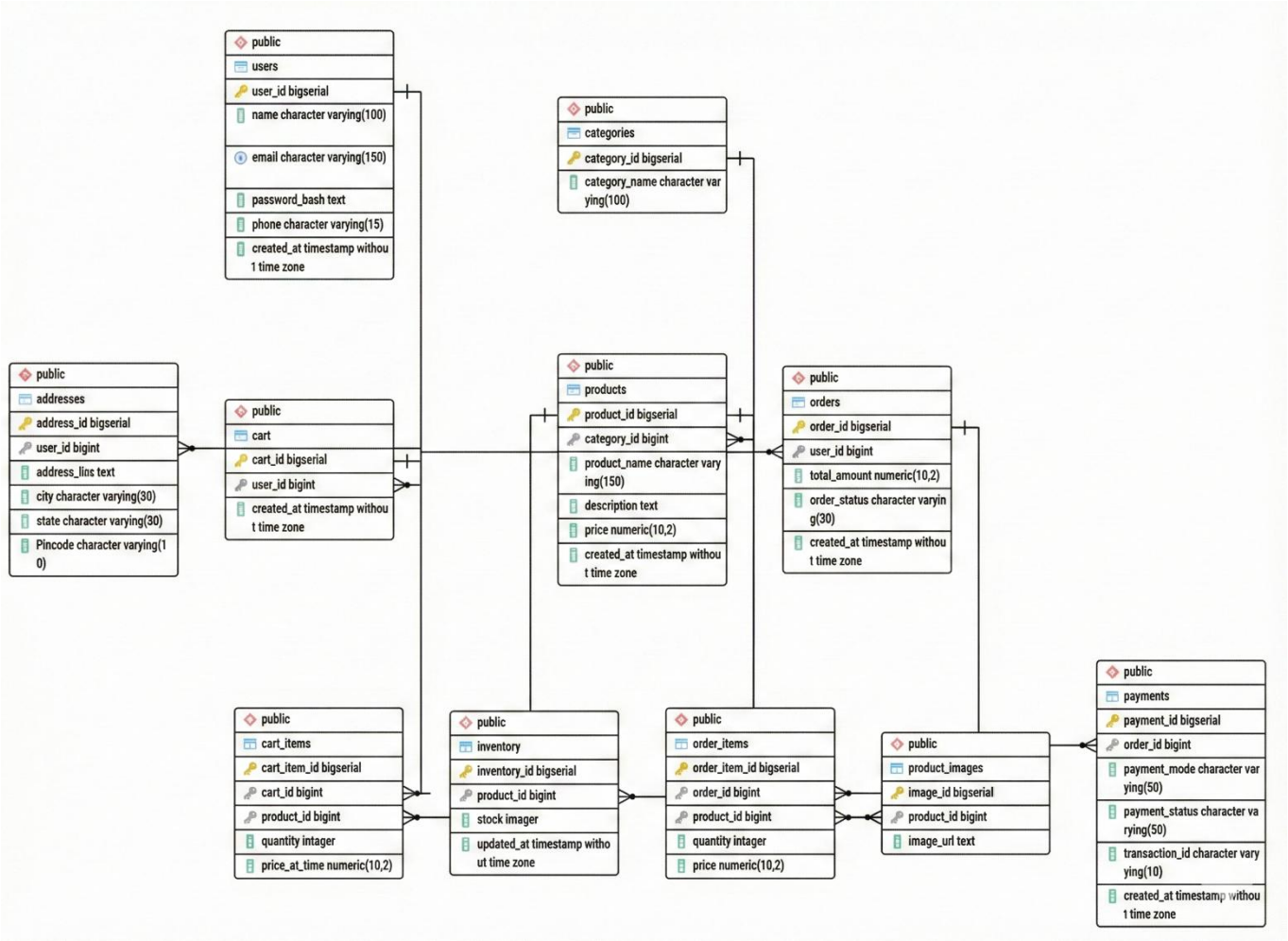

Fig. (a): Database Schema

7. High-Level Design (HLD):

The system is built using a microservices-based architecture:

1. API Gateway
- o Responsible for request routing, authentication, authorization, and rate limiting.

2. User Service
- o Handles user authentication and manages profile information.

3. Product Service
- o Maintains the product catalog and related metadata.

4. Search Service
- o Leverages Elasticsearch to provide fast and efficient product search.

5. Cart Service
- o Manages user carts and validates pricing details.

6. Order Service
- o Manages order creation and tracks order status.

7. Inventory Service
- o Controls stock levels and ensures overselling is prevented.

8. Payment Service
- o Processes payments through third-party payment gateways.

10. Kafka + CDC Pipeline
- o Synchronizes product data updates with Elasticsearch.

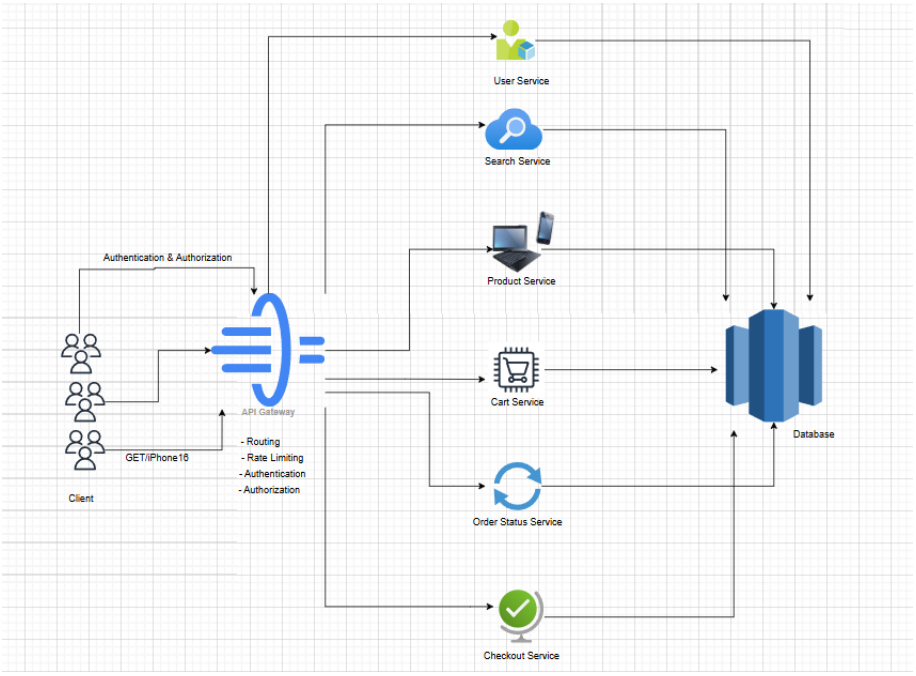

Fig. (b): HLD Diagram

8. Low- Level Design (LLD):

- **Inventory Management (Flash Sale Handling)**
- Inventory updates are performed using atomic operations.
- Stock is decremented only after successful order confirmation.
- Database transactions and row-level locking are used to prevent race conditions.
- Kafka events are used to ensure eventual synchronization across services.

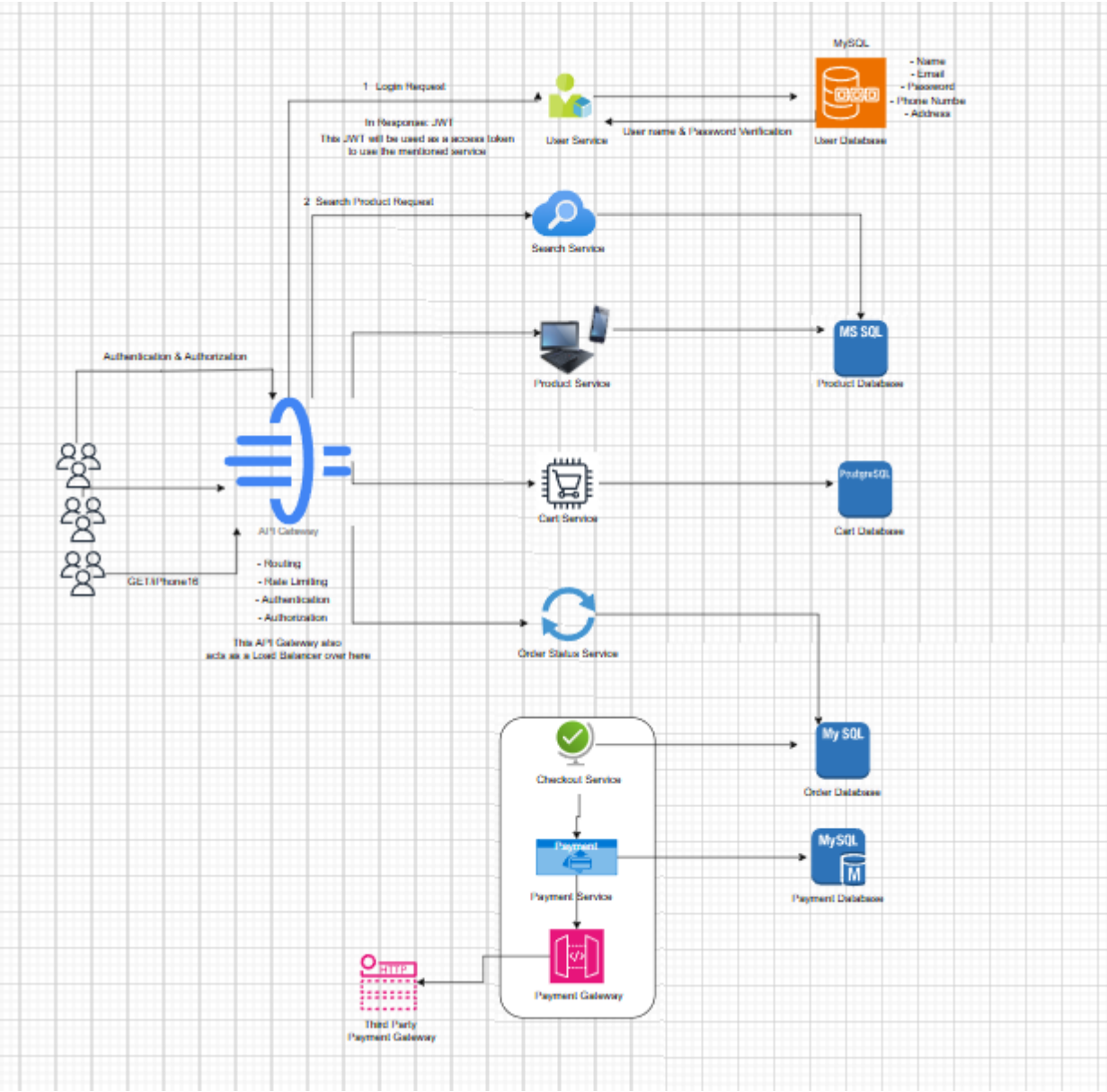

Fig. (c): LLD Diagram

9. Scalability Solution:

• Microservices are horizontally scaled to handle increased load.
• The API Gateway functions as a load balancer.
• Elasticsearch is used to manage read-heavy search workloads.
• Kafka enables asynchronous communication between services.
• The CDC pipeline ensures near real-time data consistency between the Product database and the search index.

10. Learning Outcomes:

• Learned how to design a real-world, scalable e-commerce system.
• Gained clarity on functional and non-functional requirements.
• Developed an understanding of CAP theorem trade-offs.
• Learned how Elasticsearch and Kafka are used in production environments.
• Understood how to handle concurrency and race conditions effectively.
• Recognized the importance of low latency and high availability in distributed systems.