Experiment 2

Name: Aaditya Anand
UID: 23BCS10446
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:**

- Identify functional and non-functional requirements of an online shopping platform.
- Design a scalable microservices-based architecture.
- Analyse CAP theorem trade-offs across system components.
- Create HLD designs for product search, cart, order management, payment, and inventory modules.
- Understand the use of Kafka, Elasticsearch, and CDC pipelines in large-scale distributed systems.
- Handle race conditions in flash sales and limited inventory scenarios.

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 fast product search and indexing.
- **Apache Kafka** – Used for event streaming and message processing.
- **CDC Pipeline** – Used to sync product data with Elasticsearch.

4. **System Requirements:**

A. **Functional Requirements**

Users can search products by name/title.

- Users can view full product details (description, images, price, stock, reviews).
- Users can select quantity and add items to cart.
- Users can check out and make successful payments.
- Users can track order status after placing the order.
- System must handle limited inventory efficiently.
- System must support flash sale concurrency without overselling.

B. Non – Functional Requirements

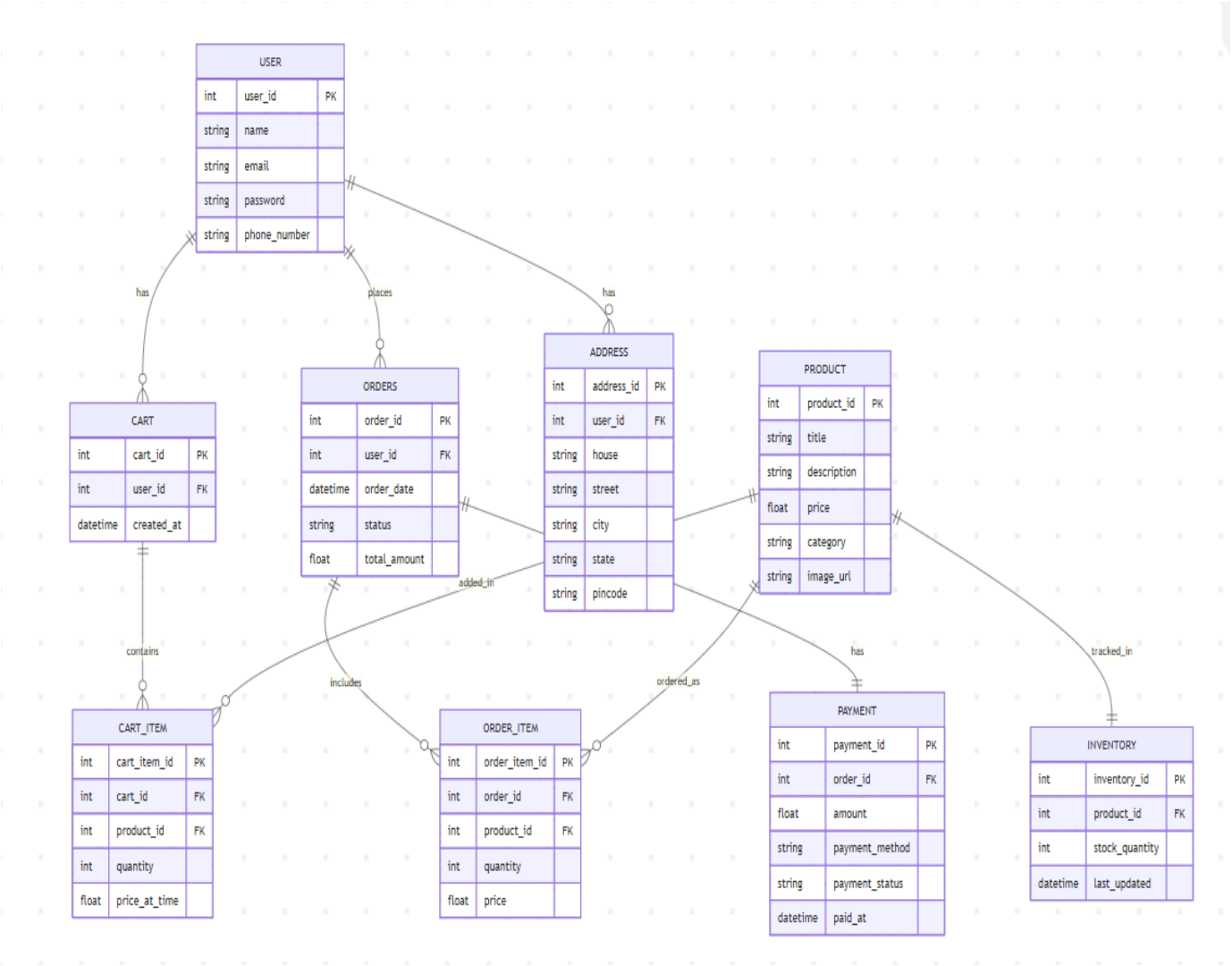
- **Scalability**
  - Target user base: up to 100 million daily active users (DAU)
  - Order processing throughput: approximately 10 orders per second
- **Availability & Consistency**
  - High availability is required for:
    - Product search
    - Product listings
  - Strong consistency is required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

6. Database Schema:

- The e-commerce database is designed using an Entity–Relationship (ER) model, showing key entities, attributes, and relationships to support users, products, cart, orders, inventory, and payments.
- The ER diagram is created in PostgreSQL (pgAdmin ERD Tool) using tables with primary and foreign keys, ensuring strong referential integrity.

1. Entity Relationships:

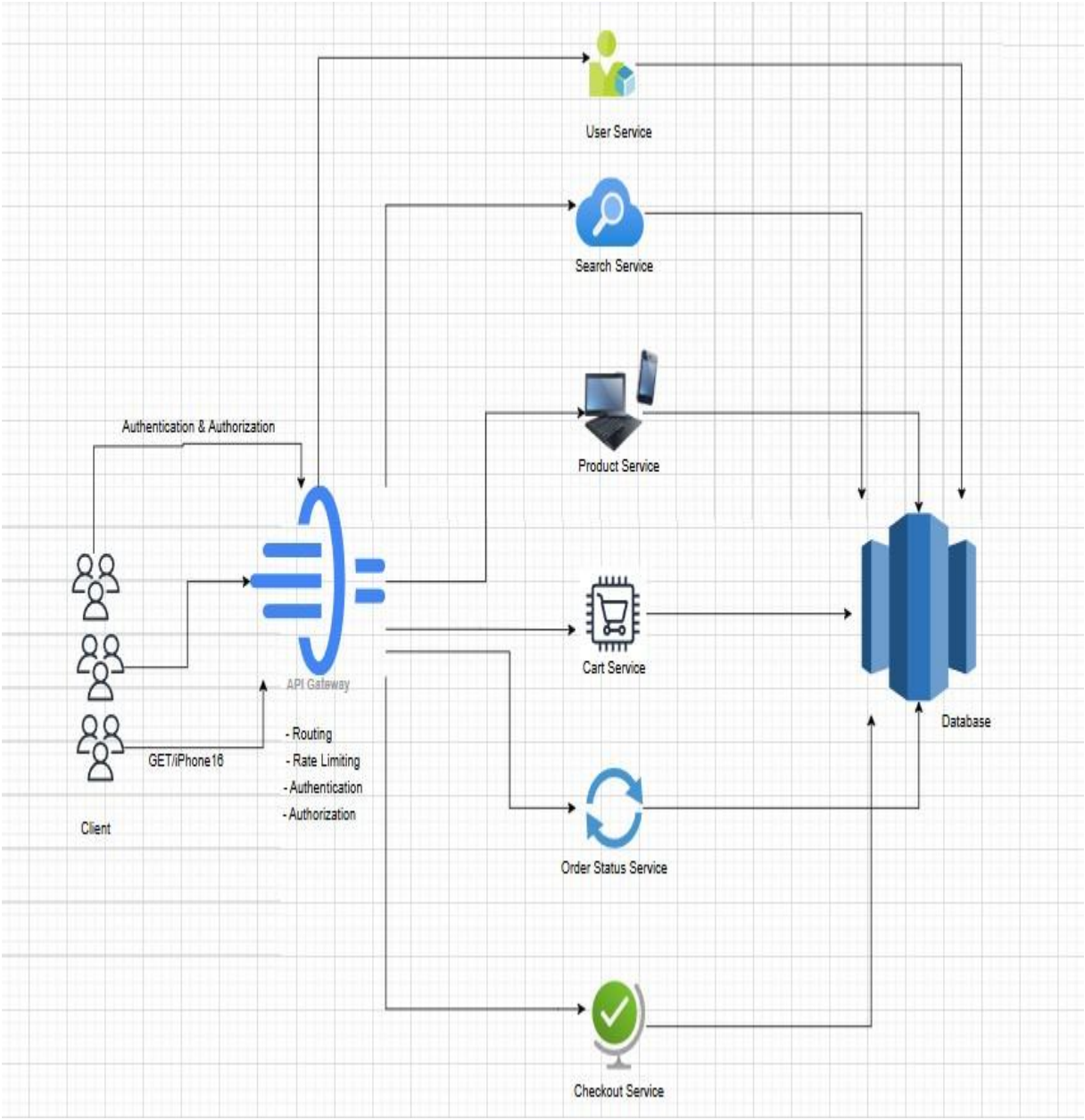
• A user can maintain multiple addresses (one-to-many).
• Each user is linked to exactly one cart (one-to-one).
• A cart can contain multiple products through cart items (many-to-many).
• A user can place multiple orders over time (one-to-many).
• An order can include multiple products through order items (many-to-many).
• Each product is assigned to a single category (many-to-one).
• Each product has a dedicated inventory record (one-to-one).
• Each order has one corresponding payment record (one-to-one).
• A product can store multiple images (one-to-many).



7. High-Level Design (HLD):

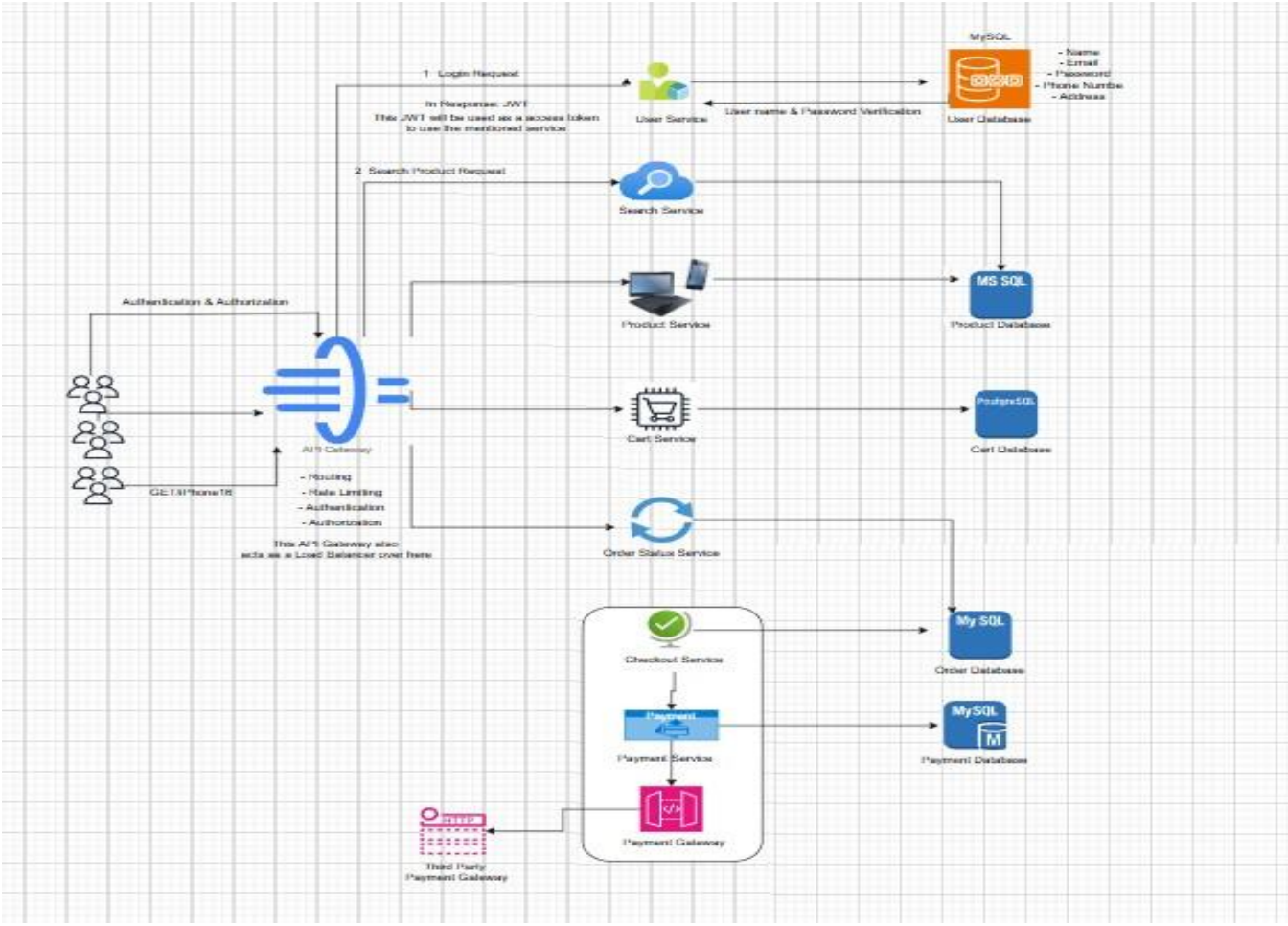
The system is built using a microservices-based architecture:

- **API Gateway** – Handles request routing, authentication, authorization, and rate limiting.
- **User Service** – Manages user authentication and profile-related data.
- **Product Service** – Maintains the product catalogue and its related metadata.
- **Search Service** – Uses Elasticsearch to deliver fast and efficient product search.
- **Cart Service** – Manages user cart operations and validates pricing details.
- **Order Service** – Handles order creation and tracks order status.
- **Inventory Service** – Maintains stock levels and ensures overselling is prevented.
- **Payment Service** – Processes payments using third-party payment gateways.
- **Kafka + CDC Pipeline** – Keeps product data synchronized with Elasticsearch in real-time.



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.



9. Scalability Solution:

• Microservices are horizontally scaled to handle high traffic and increased load.
• The API Gateway also acts as a load balancer to distribute incoming requests.
• Elasticsearch is used to efficiently manage read-heavy search operations.
• Kafka enables asynchronous service-to-service communication.
• The CDC pipeline ensures near real-time consistency between the Product database and the Elasticsearch search index.

10. Learning Outcomes:

• Learned to design a real-world, scalable e-commerce system.
• Gained clear understanding of functional and non-functional requirements.
• Developed understanding of CAP theorem trade-offs in distributed systems.
• Learned the production use of Elasticsearch and Kafka.
• Understood how to manage concurrency and prevent race conditions.
• Recognized the importance of low latency and high availability in distributed systems.