University Institute of Engineering
Department of Computer Science & Engineering

EXPERIMENT: 2

NAME: AAYUSH AMAN **UID: 23BCS14046**
BRANCH: BE-CSE **SECTION/GROUP: KRG_1B**
SEMESTER: 6TH **SUBJECT CODE: 23CSH-314**
SUBJECT NAME: SYSTEM DESIGN

1. Aim:

To design a robust, scalable, and highly available online shopping platform (similar to Amazon or Flipkart) that enables users to browse, search, and purchase various consumer goods like electronics and apparel.

2. Objective:

To design a robust, scalable, and highly available online shopping platform (similar to Amazon or Flipkart) that enables users to browse, search, and purchase various consumer goods like electronics and apparel.

3. Tools Used: Postman, Draw.io, Lucidchart, Excalidraw

4. Requirements Specification:

4.1. Functional Requirements:

- Search: Users must search for products by title or name.
- Product Details: View description, images, quantity, and reviews.
- Cart Management: Select quantities and move items to a persistent cart.
- Checkout & Payment: Securely process payments and finalize orders.
- Order Tracking: Check real-time status of orders.
- Inventory Control: Manage purchase of items with limited stock.

4.2. Non-Functional Requirements:

- Scale: 100 million Daily Active Users (DAU) with 10 orders/sec.
- Availability: High availability for the Product Search module.
- Consistency: High consistency for Payment, Order Placement, and Inventory.
- Latency: Target response time of ~200 ms.
- Reliability: Use of JWT for secure Authentication & Authorization.

5. API Design:

5.1. GET /products/search_item={keywords}
Returns a paginated list of Product IDs.

5.2. GET /products/{product_id}
Returns details (Name, Price, Thumbnail URL).

5.3. POST /cart/add_products
Adds items to the user's cart (requires User_id).

5.4. POST /checkout
Initiates checkout with Product IDs and Total Price.

5.5. POST /payment
Processes payment and returns Success/Fail status.

5.6. GET /order_status={order_id}
Fetches current status of a specific order.

6. Database Schema:

The system is divided into domain-specific schemas to support microservices:

- Identity (MySQL): users table storing credentials and contact info.
- Catalog (MS SQL): products, categories, and sellers tables.
- Media (S3): product_images table linked to image URLs in S3.
- Cart & Inventory (PostgreSQL): cart, cart_items, and inventory (with reserved_quantity).
- Orders & Payment (MySQL): orders, order_items, and payments tables.
- Social: reviews table for user feedback and ratings.

7. High Level Design (HLD):

The HLD follows a Microservices pattern:

- API Gateway: Acts as a Load Balancer and handles Routing, Rate Limiting, and Authentication (JWT).
- Service Layer: Independent services (User, Search, Product, Cart, Checkout) communicate via the Gateway.
- Persistence Layer: Each service owns its database to prevent a single point of failure.

8. Low Level Design (LLD):

The LLD addresses specific NFR challenges using an Event-Driven Architecture:

- Search Optimization: A CDC (Change Data Capture) Pipeline monitors the Product DB. A Connector service sends changes to a Streaming Buffer (Kafka), which updates ElasticSearch.
- Inventory Logic: Uses an Inventory Consumer to handle Stock update messages from Kafka. It manages reserved_quantity to prevent overselling.
- Payment Flow: Integration with a Third Party Payment Gateway via HTTP. Status updates (Fail/Success) are propagated back to the Order Status Service.

9. Learning Outcomes:

9.1. Understanding Distributed System Latency
9.2. Collision Resolution Strategies
9.3. Architectural Scalability and Reliability
9.4. Distributed State Management
9.5. Single Points of Failure (SPOF)