**Synopsis**

**Quick Commerce**

**Introduction:**

Quick commerce (Q-commerce) is revolutionizing the e-commerce industry by enabling ultra-fast deliveries within 10–30 minutes. This project focuses on building a scalable, efficient, and high-performance Q-commerce platform using **microservices architecture** with **Java Spring Boot**. The system ensures seamless order processing, inventory management, real-time tracking, and dynamic delivery management.

**Objectives:**

* Develop a **microservices-based** Q-commerce platform.
* Ensure **low-latency** order fulfilment with **real-time updates**.
* Implement **automated inventory management** for nearby warehouses.
* Enable **dynamic delivery allocation** using location-based assignment.
* Maintain **high scalability** and **fault tolerance** through containerized deployment.

**Technology Stack:**

* **Backend:** Java Spring Boot (Spring Cloud, Spring Data, Spring Security)
* **Database:** PostgreSQL / MongoDB / Redis (for caching)
* **API Gateway:** Spring Cloud Gateway / Netflix Zuul
* **Message Broker:** Apache Kafka / RabbitMQ for event-driven communication
* **Authentication:** OAuth2
* **Containerization & Deployment:** Docker, Kubernetes, CI/CD with Jenkins/GitHub Actions
* **Monitoring & Logging:** Prometheus, Grafana, ELK Stack

**Microservices Architecture:**

The system will be modularized into several microservices, including:

1. **User Service** – Handles user authentication, profiles, and authorization.
2. **Product Catalog Service** – Manages product listings, availability, and pricing.
3. **Inventory Service** – Tracks stock levels and updates based on orders.
4. **Order Service** – Processes orders, applies discounts, and generates invoices.
5. **Payment Service** – Integrates with payment gateways for secure transactions.
6. **Delivery Service** – Assigns and tracks riders for real-time delivery.
7. **Notification Service** – Sends order updates via SMS, email, or push notifications.

**Workflow:**

1. User places an order through the mobile/web app.
2. The **Order Service** validates the cart and interacts with the **Inventory Service**.
3. The **Payment Service** processes the transaction.
4. The **Delivery Service** assigns a nearby rider based on availability.
5. The user gets real-time tracking via the **Notification Service**.
6. The order is fulfilled within 10–30 minutes.

**Expected Outcomes:**

* **A** **scalable, resilient**, and **highly available** Q-commerce platform.
* **Optimized delivery logistics** for faster fulfilment.
* **Improved user experience** through real-time tracking and notifications.
* **Modular and loosely coupled** services for easy maintenance and scalability.