Backend Engineer Technical Assessment 1

Overview

At EatClub Brands, we are building backend systems that power a full-stack food delivery chain, handling thousands of orders daily with high performance and reliability. This assignment is designed to evaluate your ability to design, implement, and deploy scalable backend systems.

Mission

Develop a scalable order management and processing system that can handle real-time order placement, status updates, and inventory management. Your solution should focus on performance, scalability, and robustness while considering real-world challenges in food delivery systems.

Time Expectation

- Expected time: 8-10 hours
- Deadline: 5 days from receipt

Challenge: Real-Time Order Management System

Context

Managing real-time orders in a food delivery chain requires robust systems that can handle high throughput, low latency, and fault tolerance. Your task is to build a backend system to manage the lifecycle of an order from placement to delivery.

Requirements

Part 1: Order Management API

1. Implement RESTful APIs for:
    - Placing an order
    - Updating order status (e.g., "Preparing," "Out for Delivery," "Delivered")
    - Fetching order details
2. Ensure input validation and error handling.

Part 2: Inventory Management

1. Implement an inventory tracking system:

- Deduct inventory when an order is placed.
- Increment inventory when order is cancelled.
- Alert when inventory falls below a threshold.

2. Use a database (PostgreSQL preferred) to store inventory data.

Part 3: Event-Driven Architecture

1. Integrate an event-driven system:
    - Emit events for order status changes.
    - Consume events to update analytics, notify external systems or send tracking notifications to the customer

Performance Requirements

- API response time: < 200ms
- Concurrent users supported: 500+
- Error rate: < 0.5%

Technical Requirements

Core Features

1. Scalable architecture
2. Database design optimized for high read/write operations.
3. Proper use of version control (Git) with meaningful commits.
4. Unit tests covering at least 80% of the codebase.

Infrastructure Requirements

1. Use Docker for containerization.(optional, but preferred)
2. Deploy the system using AWS or any cloud provider (optional).
3. Provide Infrastructure as Code (IaC) scripts if applicable. (optional)

Submission Requirements

1. GitHub repository or a zipped file with:
    - Source code
    - README file explaining the architecture, setup and run instructions
2. Documentation including:
    - System architecture overview
    - API contracts (e.g., request/response formats)
    - Scaling considerations

- Flow/Sequence Diagrams

Evaluation Criteria

Criteria	Weightage
System Design	40%
Code Quality	30%
Testing & Documentation	20%
Innovation & Optimizations	10%

Success Metrics

Metric	Target
Latency	<200ms
Throughput	>500 req/s
Error Rate	<0.5%

Test Coverage	>80%