**Usecase 1:** Distributed Event-Driven Microservices Order Management System

**Technologies:** Spring Boot, Kafka, AWS ECS, DynamoDB, Redis

**Design Guidelines**:

* Design a scalable microservices architecture for an e-commerce order processing system
* Implement event-driven communication using Apache Kafka for order tracking
* Create services for order creation, payment processing, inventory management, and shipping
* Use AWS ECS for containerization and deployment
* Implement caching with Redis for performance optimization
* Add distributed tracing and monitoring

**Key Requirements:**

* Real-time order status updates
* Event sourcing pattern
* Fault-tolerant design
* Performance monitoring and logging

**Usecase 2:** Real-Time Analytics Dashboard with Stream Processing

**Technologies:** Spring Boot, Apache Kafka Streams, Elasticsearch, Kibana, AWS Lambda

**Design Guidelines:**

* Build a real-time analytics platform for processing large-scale event streams
* Create a microservice that consumes and processes streaming data using Kafka Streams
* Implement complex event processing and aggregations
* Store processed data in Elasticsearch
* Develop a dashboard using Kibana for visualization
* Use AWS Lambda for serverless event processing

**Key Requirements:**

* Real-time data aggregation
* Complex event processing
* Scalable stream processing
* Interactive dashboard with real-time updates

**Usecase 3:** Secure Multi-Tenant Cloud-Native Authentication Service

**Technologies**: Spring Boot, Spring Security, OAuth2, MongoDB, AWS Cognito, Kubernetes

**Design Guidelines:**

* Develop a robust, multi-tenant authentication and authorization microservice
* Implement JWT-based authentication with refresh token mechanism
* Create role-based access control (RBAC)
* Use MongoDB for flexible user and permission storage
* Deploy on Kubernetes with horizontal scaling
* Integrate with AWS Cognito for additional security features

**Key Requirements:**

* Secure token management
* Multi-tenant support
* Scalable authentication service
* Comprehensive security features

**Usecase 4:** IoT Device Management and Telemetry Platform

**Technologies:** Spring Boot, Apache Kafka, InfluxDB, Grafana, AWS IoT Core, gRPC

**Design Guidelines**:

* Build an IoT device management platform with real-time telemetry processing
* Implement device registration and management microservice
* Use Kafka for device event streaming
* Store time-series data in InfluxDB
* Create gRPC-based communication between services
* Develop a Grafana dashboard for device monitoring
* Implement device simulation for testing

**Key Requirements:**

* Device registration and management
* Real-time telemetry processing
* Scalable device communication
* Comprehensive monitoring dashboard

**YOUR TITLE HERE**