Sporty Group

Overview

This assignment evaluates your backend engineering skills using Java (preferably Spring Boot), with a focus on asynchronous architecture, event-driven design, and clean code practices.

You'll build a simplified ticketing system backend that simulates a support platform, with decoupled services via messaging (Kafka/RabbitMQ). The assignment is designed to be completed in 90 minutes.

Requirements

Implement a backend ticketing system with the following capabilities using an asynchronous messaging approach:

- Users submit support tickets via a REST API
- Tickets are published to a Kafka (or RabbitMQ) topic
- A consumer service processes tickets from the topic and stores them
- Agents can asynchronously assign and update tickets

Event Flow

1. User submits a support ticket (HTTP POST /tickets)
2. Ticket is sent to a Kafka or RabbitMQ topic (e.g., support-tickets)
3. Consumer service processes the event and stores the ticket in an in-memory DB
4. Agent assignment and ticket updates are performed by publishing to corresponding topics (e.g., ticket-assignments, ticket-updates)

Data Modeling

Ticket:
- ticketId (UUID)
- subject
- description
- status (open, in_progress, resolved, closed)
- userId
- assigneeId (nullable)
- createdAt / updatedAt

Event types:
- TicketCreated
- TicketAssigned
- TicketStatusUpdated

Sporty Group

API Specification

1. Submit Ticket

POST /tickets

Payload:

```
{
  "userId": "user-001",
  "subject": "Login problem",
  "description": "Cannot reset my password"
}
```

2. Assign Ticket (Event)

Event (topic: ticket-assignments):

```
{
  "ticketId": "abc123",
  "assigneeId": "agent-007"
}
```

3. Update Ticket Status (Event)

Event (topic: ticket-updates):

```
{
  "ticketId": "abc123",
  "status": "in_progress"
}
```

Sporty Group

Delivery

- A GitHub link (public repo)
- A README.md including:
    - Setup and run instructions
    - Message formats used
    - Tests included
    - Design decisions
    - AI tool usage and validation steps if used (we encourage using AI)
- Docker compose (optional)