Sporty Group

Overview

This assignment evaluates your backend engineering skills using Java (preferably Spring Boot), with a focus on distributed coordination, locking strategies, and concurrency-safe design.

You'll build a simplified distributed ticketing system backend that ensures ticket updates (like assignment and status changes) are consistent even under concurrent access in a multi-instance environment. Designed to be completed within 90 minutes.

Requirements

Build a backend system that manages support tickets with distributed locking for concurrent-safe updates.

- Create support tickets via a REST API
- Support updates to ticket status and assignment
- Implement distributed locking to ensure consistency (e.g., Redis-based lock)
- Prevent race conditions across multiple service instances

Sporty Group

Functional Requirements

1. Submit a new support ticket
2. Update ticket status and assign an agent
3. Ensure mutual exclusion using distributed locking (e.g., Redisson, Redis SETNX, etc.)
4. Store tickets in an in-memory DB (or mock persistent store)

Data Modeling

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

Locking Requirement:
- Ensure only one process can update a ticket at a time.

Example: If two agents try to assign themselves to the same ticket concurrently, only one succeeds.

Sporty Group

API Specification

1. Create Ticket

POST /tickets

Payload:

```
{
  "userId": "user-001",
  "subject": "Login not working",
  "description": "I can’t sign in to my account."
}
```

2. Update Ticket Status

PATCH /tickets/{ticketId}/status

```
{
  "status": "resolved"
}
```

3. Assign Ticket

PATCH /tickets/{ticketId}/assign

```
{
  "assigneeId": "agent-123"
}
```

Delivery

- A GitHub link (public repo)
- A README.md including:
    - Setup and run instructions
    - Description of your locking strategy
    - Sample concurrent update test case
    - AI tool usage and validation steps if used (we encourage using AI)
- Docker compose (optional)