Problem Statement

Description

A growing e-commerce platform needs basic real-time analytics to track user behaviour. Your task is to build a focused, production-ready analytics service that processes user events and provides basic real-time insights.

Build a streamlined analytics platform that can:

1. Ingest user events.
2. Process events in real-time
3. Provide basic analytics through a simple dashboard
4. Handle reasonable load (100 events/second)

Technical Requirements

Core Features

1. Event Ingestion
    - Accept JSON events.
    - Basic validation and error handling.
    - Provide a mechanism to rate limit/control events being consumed by the system.
    - Events include: timestamp, user_id, event_type, and page_url
      Refer to Notes for more details.
2. Real-Time Processing
    - Calculate and store rolling metrics for:
        - Active users (last 5 minutes)
        - Page views by URL (last 15 minutes)
        - Active sessions against a user (last 5 minutes)
    - Use any DB/cache for real-time aggregations.
3. Dashboard
    - Simple single-page dashboard showing:
        - Active users count
        - Top 5 pages
        - Number of active sessions for the same user
    - Auto-refresh every 30 seconds
4. Mock Data Generator:
    - A simple standalone utility that generates random events in the given contract in regular intervals until stopped.
    - This is just a mock event generator.

Technology Stack

- Backend: In Java or Python programming language
- Database: DB/Cache of your choice.
- Frontend: Simple Frontend in any stack of your choice.
- Docker for containerization

Deliverables

1. Source code in a Git repository
2. Docker compose file for easy startup
3. README with:
    - Setup instructions
    - API Documentation
    - Architecture overview diagram
    - Future improvements section

Evaluation Criteria

1. **Code Quality (30%)**
    - Clean, maintainable code
    - Error handling
    - Testing strategy
2. **System Design (30%)**
    - Architecture choices
    - Data modelling
    - Performance considerations
3. **Functionality (20%)**
    - Core features working
    - Reliability
    - User experience
4. **Documentation (20%)**
    - Clear setup instructions
    - API Documentation
    - Priority should be given to Setup instructions and Architecture Overview Diagram

Time Allocation (4 days)

Notes

1. Focus on getting core features working reliably.
2. Prioritize code quality and error handling.
3. Implement unit test cases wherever possible.
4. Document any assumptions and tradeoffs.
5. Below is the sample JSON Event Structure:

```
{
  "timestamp": "2024-03-15T14:30:00Z",
  "user_id": "usr_789",
  "event_type": "page_view",
  "page_url": "/products/electronics",
  "session_id": "sess_456"
}
```

6. Feel free to use Docker-based installations wherever possible.