**📊 Project Report: Application Monitoring Dashboards**

**1️ - Project Overview**

This project implements a **Log Analytics Platform** that collects, processes, and visualizes log data in real-time. It is built using a **microservices architecture** and includes the following core components:

* 🌐 **REST API Server** (Node.js)
* 🔄 **Apache Kafka** for message brokering
* 🛢️ **PostgreSQL** for data storage
* 📈 **Grafana** for visualization and monitoring

**2️ - System Architecture**

The system follows a **distributed architecture**, composed of the following layers:

* 🧩 **API Layer** – Handles all HTTP requests and responses
* 📬 **Message Queue** – Uses Kafka for log ingestion and asynchronous processing
* 🗃️ **Database** – PostgreSQL is used for persistent and structured storage
* 👁️ **Visualization** – Grafana dashboards provide real-time insights

**3 - API Endpoints Implementation**

**System Overview Our monitoring system provides 5 core endpoints focused on application performance testing and health monitoring:**

**1️. Basic Testing**

**Test Endpoint GET /api/test**

**• Purpose: Basic API health verification**

**• Response: { "message": "Test endpoint successful" }**

**• Status: 200 OK**

**2️. Error Simulation**

**Error Endpoint**

**GET /api/error**

**• Purpose: Simulates error scenarios**

**• Response: { "error": "Simulated error" }**

**• Status: 500 Internal Server Error**

**3. Performance Testing**

**Delay Endpoint**

**GET /api/delay**

**• Purpose: Tests latency handling**

**• Response: { "message": "Delayed response (X ms)" }**

**• Delay Range: 100-1100ms**

**• Status: 200 OK**

**4. Reliability Testing Unreliable Endpoint**

**GET /api/unreliable**

**• Purpose: Tests service reliability**

**• Success (70%): { "message": "Service is working" }**

**• Failure (30%):**

**{ "error": "Service temporarily unavailable" }**

**• Error Status: 503 Service Unavailable**

**5. Health Monitoring**

**Health Check**

**GET /health**

**• Purpose: System health verification**

**• Response: { "status": "healthy" }**

**• Status: 200 OK**

**6. Error Handling**

**All undefined routes return:**

**{ "error": "Not found" }**

**Status: 404 Not Found**

**4 – Procedure**

* **API requests come in through different endpoints (/api/test, /api/error, etc.)**
* **Each request is processed and logged via Kafka producer**
* **Logs are sent to specific Kafka topics based on endpoint**
* **Kafka consumer processes messages and stores them in PostgreSQL tables**
* **Grafana dashboards query PostgreSQL to display metrics**
* **Run python load\_test.py from the scripts directory**
* **Generates simulated traffic to various endpoints**
* **Creates a mix of successful and error responses**
* **Uses multiple concurrent workers to simulate real-world load**

**4 - Technical Features**

**📡 Monitoring Capabilities**

* Real-time log collection
* Performance metric tracking
* Error detection and visualization

**🧪 Load Testing Support**

* Pre-built load testing scripts in Python
* Simulates **concurrent requests**
* Captures benchmarking results

**5 - Deployment and Operations**

The system is **Docker-based** and can be easily deployed using docker-compose.

* 🔌 Default Ports:
  + API: 8080
  + Grafana: 3000
  + Kafka: 9092
* 🔐 Default Credentials for Grafana:
  + Username: admin
  + Password: admin

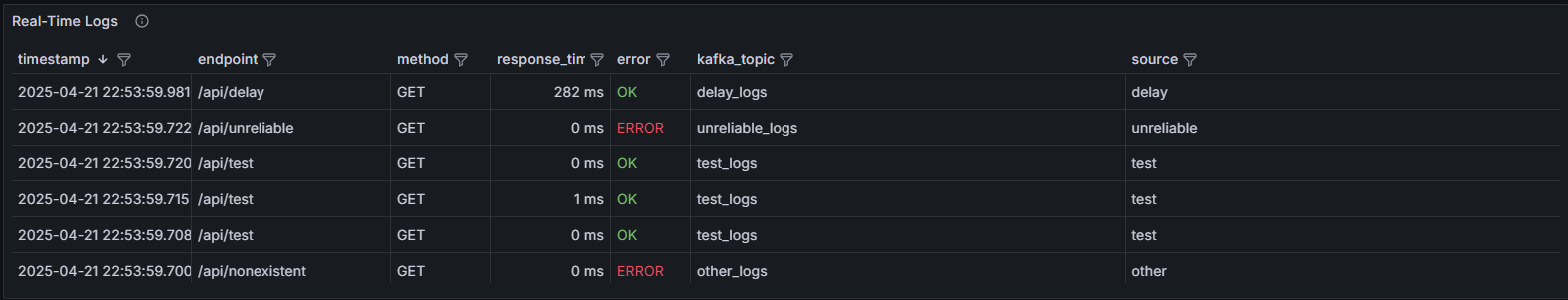
**6 - Monitoring Dashboard Features**

The **Grafana dashboards** offer powerful real-time visualizations including:

* 📈 Request count per endpoint
* ⏱️ Response time trends
* ❌ Error counts per endpoint



* 🪵 Live log stream and log-based alerts



**7 - System Requirements**

To deploy and run this system, the following tools and resources are required:

* ✅ Docker & Docker Compose
* ✅ Node.js
* ✅ Python (for test automation)
* ✅ Adequate system memory & CPU for container orchestration

**✅ Conclusion**

* This project delivers a scalable, fault-tolerant **log monitoring solution** for managing and visualizing the operations of a 3D printing platform. With structured **API endpoints**, a robust backend, and **Grafana-powered dashboards**, it enables comprehensive **real-time monitoring** of distributed systems.

**CODE - https://github.com/Mdr-Pranav/ApplicationMonitoringDashboards**

**Team Information**

PES1UG22CS312 - M K Vishwaas

PES1UG22CS329 - Manas Shetty

PES1UG22CS360 - Mohul Y P

PES1UG22CS362 - Mudar Pranav