Mini-Project

SaaS Monitoring Platform

Log Analysis and Monitoring Application

with ELK Stack, NoSQL, and Web Interface

Technical Documentation

Author: Mohamed Landolsi

Email: mohamedlandolsi300@gmail.com

Class: 3GLA

Repository: https://github.com/mohamedlandolsi/saas-monitoring-platform

Institution: IT Business School

Scenario: D - SaaS Web Application

BigData Frameworks Project - October 2025

Contents

Chapter 1

Introduction

1.1 Context

In distributed systems, log production has become exponential. These logs from web applications, microservices, databases, and APIs constitute crucial information for incident detection, performance analysis, security compliance, and business intelligence.

1.2 Problem Statement

Modern enterprises face challenges: logs dispersed across servers, no centralization, manual searching, no automatic alerting, and no visualization of trends and patterns.

1.3 Objectives

This platform must:

1. Centralize log collection from different sources
2. Intelligently index data for ultra-fast search
3. Provide relevant visualizations
4. Offer an intuitive web interface
5. Be easily deployable via containerization

1.4 Chosen Scenario: SaaS Web Application

We selected **Scenario D: SaaS Application** - a monitoring platform for a SaaS used by thousands of client companies.

Types of logs: Web server logs, application logs (Flask), database logs, performance logs, API logs.

KPIs: Error rate by service, average response time, slowest endpoints, errors per hour, active users.

Chapter 2

Architecture

2.1 Global Architecture

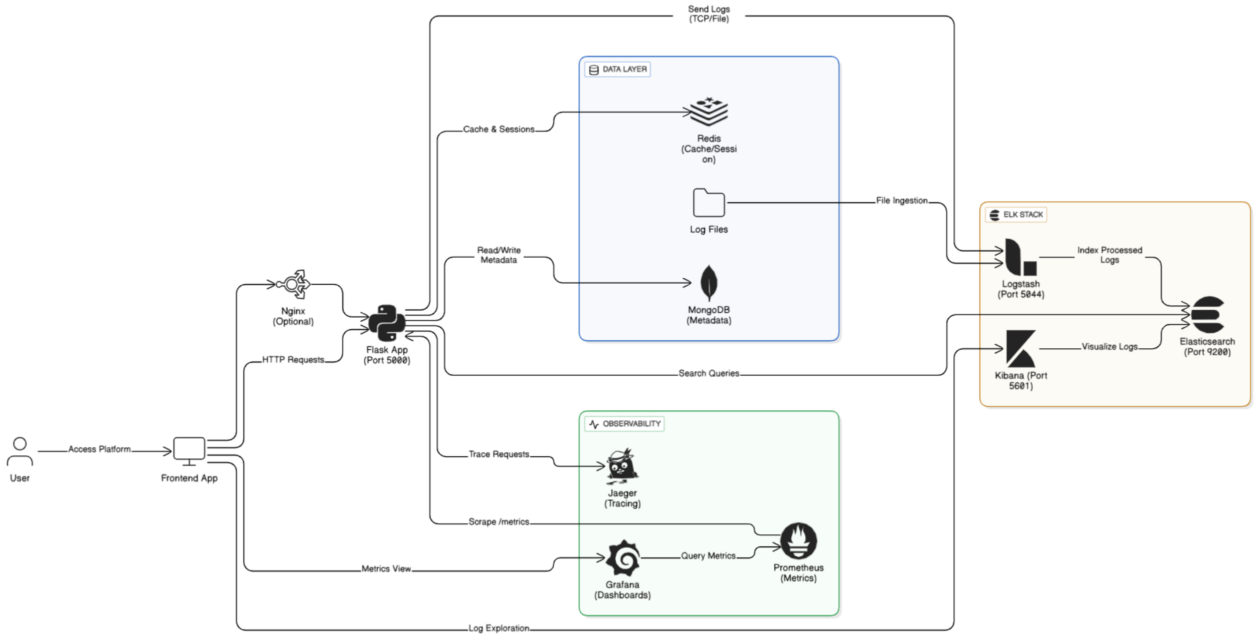


Figure 2.1: System Architecture

2.2 Components

Frontend: Bootstrap 5, Chart.js, Socket.IO

Backend: Flask 3.0, Flask-SocketIO, Gevent

ELK Stack: Elasticsearch 8.11, Logstash 7.17, Kibana 8.11

Databases: MongoDB 7, Redis 7

Observability: Prometheus, Grafana, Jaeger

2.3 Service Ports

Service	**Port**	**Service**	**Port**
webapp	5000	kibana	5601
elasticsearch	9200	prometheus	9090
logstash	5044	grafana	3000
mongodb	27017	jaeger	16686
redis	6379		

Table 2.1: Port Mapping

Chapter 3

Technical Specifications

3.1 Technology Stack

Technology	**Version**	**Purpose**
Python	3.11	Backend runtime
Flask	3.0.0	Web framework
Flask-SocketIO	5.3.6	Real-time WebSocket
Elasticsearch	8.11.0	Search engine
Logstash	7.17.0	Log processing
MongoDB	7	Document storage
Redis	7	Caching
Prometheus	2.48.0	Metrics collection
Grafana	10.2.2	Dashboards
Docker Compose	2.0+	Orchestration

Table 3.1: Technology Versions

3.2 Developed Modules

File Management: Upload, validation, processing of CSV/JSON logs
Web Interface: Dashboard, search, upload, files, live logs pages
MongoDB Integration: Metadata storage with connection pooling
Real-Time Streaming: WebSocket with Redis message queue
Observability: Prometheus metrics, health checks, structured logging

Chapter 4

Code Excerpts

4.1 Prometheus Metrics

```
from prometheus_client import Counter, Histogram, Gauge

http_requests_total = Counter(
    ’http_requests_total’, ’Total␣HTTP␣requests’,
    [’method’, ’endpoint’, ’status_code’]
)

http_request_latency_seconds = Histogram(
    ’http_request_latency_seconds’, ’Request␣latency’,
    [’method’, ’endpoint’],
    buckets=[0.01, 0.05, 0.1, 0.25, 0.5, 1.0, 2.5, 5.0]
)
```

4.2 Health Check Endpoint

```
@app.route(’/api/health’)
def health_check():
    health = {’status’: ’healthy’, ’checks’: {}, ’system’: {}}

    # Elasticsearch check
    es_start = time.time()
    if es_client.ping():
        health[’checks’][’elasticsearch’] = {
            ’status’: ’healthy’,
            ’response_time_ms’: (time.time() - es_start) * 1000
        }

    # System metrics
    health[’system’] = {
        ’cpu_percent’: psutil.cpu_percent(),
        ’memory_percent’: psutil.virtual_memory().percent
    }
    return jsonify(health)
```

Chapter 5

ELK Configuration

5.1 Logstash Pipeline

```
input {
  tcp { port => 5000; codec => json_lines }
  file { path => "/data/uploads/*.csv"; tags => ["csv"] }
}

filter {
  if "csv" in [tags] {
    csv {
      columns => ["timestamp", "level", "endpoint",
                  "status_code", "response_time_ms", "message"]
    }
  }
  date { match => ["timestamp", "ISO8601"] }
}

output {
  elasticsearch {
    hosts => ["http://elasticsearch:9200"]
    index => "saas-logs-%{+YYYY.MM.dd}"
  }
}
```

5.2 Elasticsearch Index Mapping

Field	**Type**	**Purpose**
@timestamp	date	Event time
level	keyword	Log level
endpoint	keyword	API path
status_code	integer	HTTP status
response_time_ms	float	Latency
message	text	Log content

Table 5.1: Field Mapping

Chapter 6

Data Models

6.1 MongoDB Collections

files: {_id, filename, file_type, file_size, status, uploaded_at, records_count}
users: {_id, username, email, password_hash, created_at, role}
search_history: {_id, query, filters, results_count, timestamp, user_id}

6.2 Redis Keys

```
cache:stats:*     - Statistics (TTL: 60s)
cache:search:*    - Search results (TTL: 300s)
session:user:*    - User sessions
queue:logs        - Log streaming queue
```

Chapter 7

Installation Guide

7.1 Prerequisites

Docker 20.10+, Docker Compose 2.0+, 8GB RAM minimum.

7.2 Quick Start

```
git clone https://github.com/mohamedlandolsi/saas-monitoring-platform.git
cd saas-monitoring-platform
docker compose up -d
```

7.3 Verification

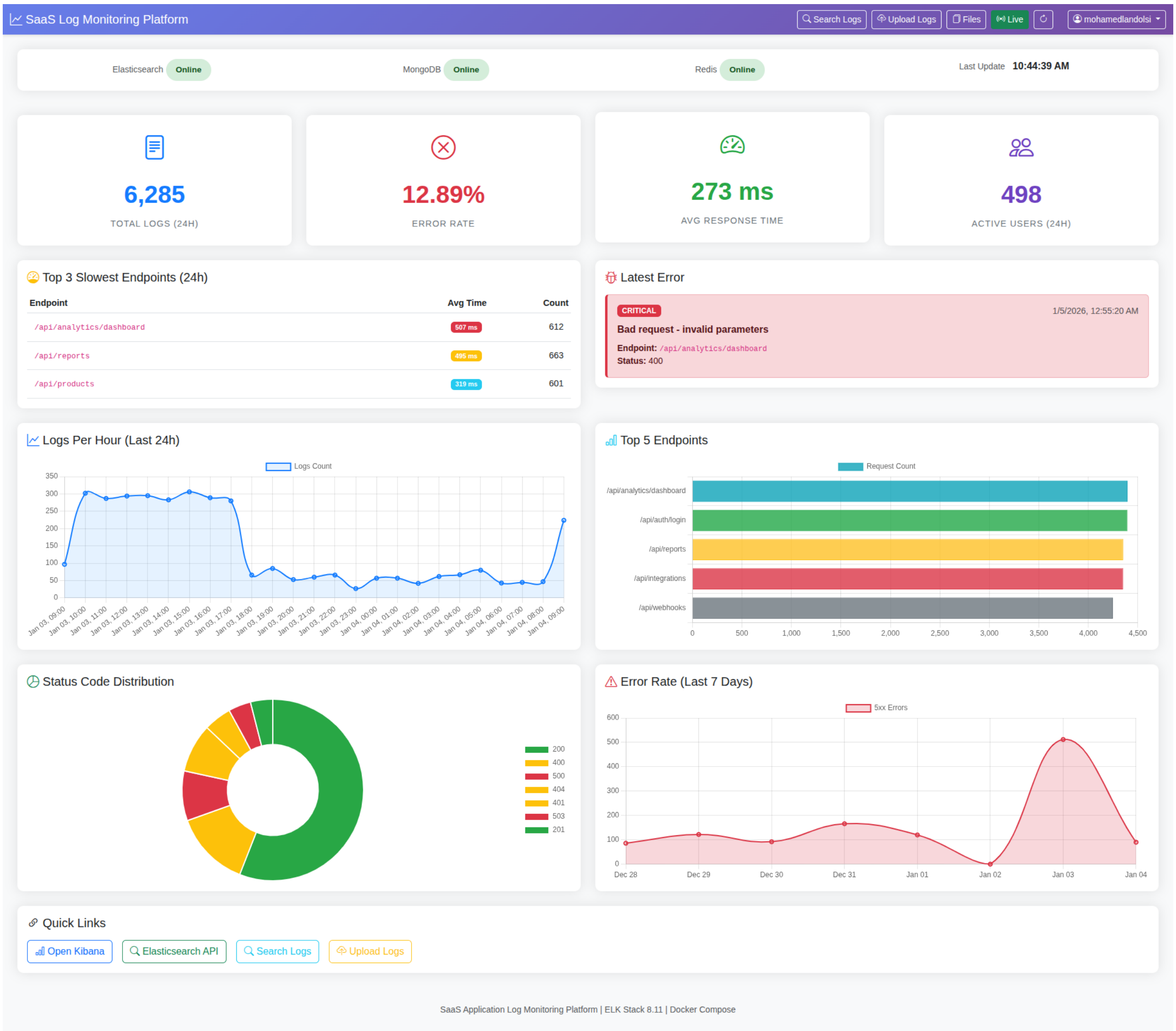
```
docker compose ps                    # Check services
curl http://localhost:5000/api/health  # Test health
curl http://localhost:5000/metrics    # Test metrics
```

Chapter 8

User Guide

8.1 Dashboard

Access `http://localhost:5000` to view KPIs: total logs, error rate, average response time, slowest endpoints.



Elasticsearch Online | MongoDB Online | Redis Online | Last Update 10:44:39 AM

6,285
TOTAL LOGS (24H)

12.89%
ERROR RATE

273 ms
AVG RESPONSE TIME

498
ACTIVE USERS (24H)

Top 3 Slowest Endpoints (24h)

Endpoint	Avg Time	Count
`/api/analytics/dashboard`	507 ms	612
`/api/reports`	495 ms	663
`/api/products`	319 ms	601

Latest Error

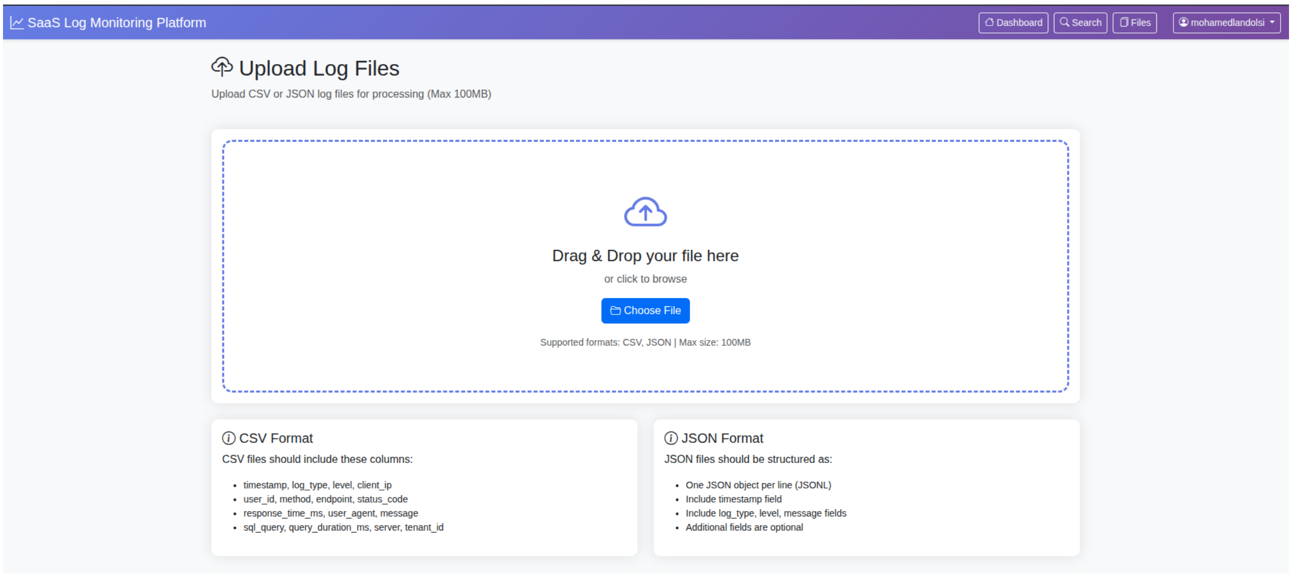
CRITICAL 1/5/2026, 12:55:20 AM
Bad request - invalid parameters
Endpoint: `/api/analytics/dashboard`
Status: 400

SaaS Application Log Monitoring Platform | ELK Stack 8.11 | Docker Compose

Figure 8.1: Main Dashboard

8.2 File Upload

Navigate to Upload page, drag and drop CSV/JSON files, monitor progress, view record count on completion.



Upload Log Files

Upload CSV or JSON log files for processing (Max 100MB)

Drag & Drop your file here

or click to browse

Supported formats: CSV, JSON | Max size: 100MB

CSV Format

CSV files should include these columns:

- timestamp, log_type, level, client_ip
- user_id, method, endpoint, status_code
- response_time_ms, user_agent, message
- sql_query, query_duration_ms, server, tenant_id

JSON Format

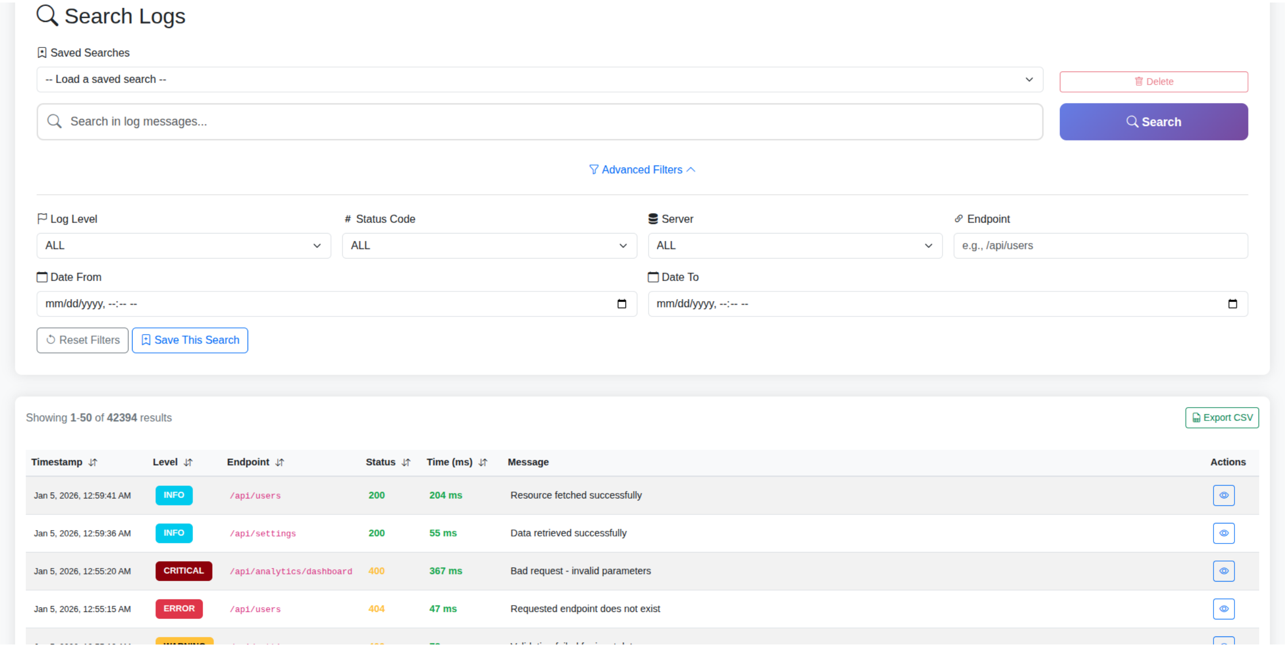
JSON files should be structured as:

- One JSON object per line (JSONL)
- Include timestamp field
- Include log_type, level, message fields
- Additional fields are optional

Figure 8.2: File Upload

8.3 Search

Use search box for keyword search, apply filters (level, time, status code), save frequent searches.



Search Logs

Saved Searches: -- Load a saved search --

Search in log messages...

Log Level: ALL

Status Code: ALL

Server: ALL

Endpoint: e.g., /api/users

Date From: mm/dd/yyyy, --:-- --

Date To: mm/dd/yyyy, --:-- --

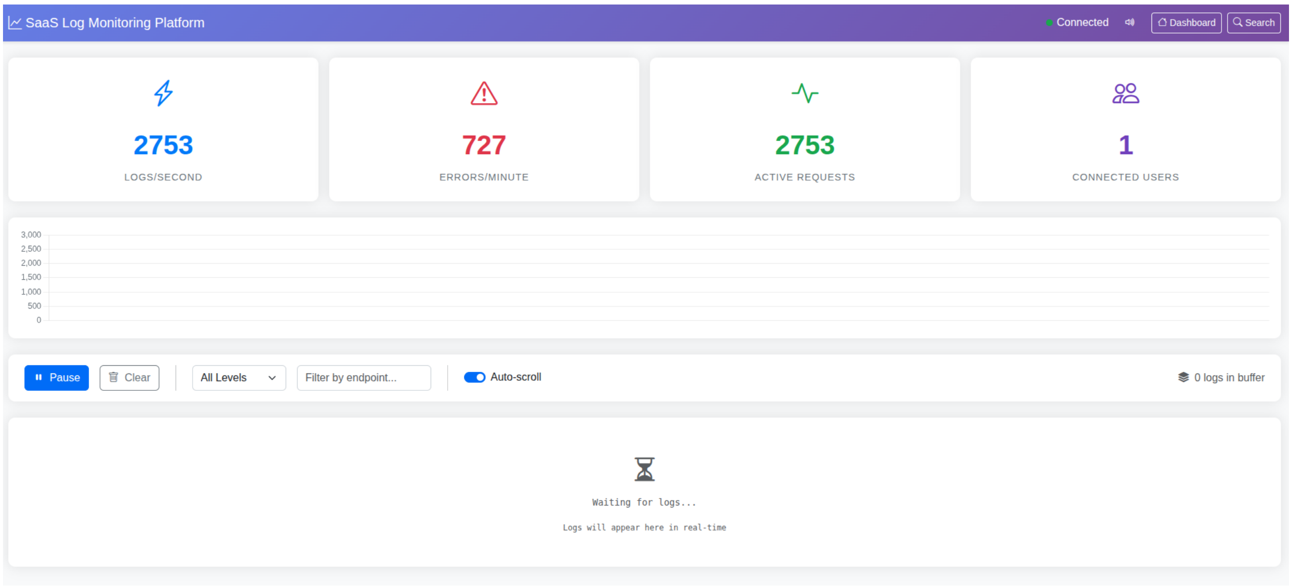
Showing 1-50 of 42394 results

Timestamp	Level	Endpoint	Status	Time (ms)	Message	Actions
Jan 5, 2026, 12:59:41 AM	INFO	/api/users	200	204 ms	Resource fetched successfully	
Jan 5, 2026, 12:59:36 AM	INFO	/api/settings	200	55 ms	Data retrieved successfully	
Jan 5, 2026, 12:55:20 AM	CRITICAL	/api/analytics/dashboard	400	367 ms	Bad request - invalid parameters	
Jan 5, 2026, 12:55:15 AM	ERROR	/api/users	404	47 ms	Requested endpoint does not exist	

Figure 8.3: Search Interface

8.4 Live Logs

Access via green “Live” button. Features: pause/resume, level filter, auto-scroll, desktop notifications for errors.



Waiting for logs...

Logs will appear here in real-time

Figure 8.4: Live Logs

8.5 Grafana Access

URL: `http://localhost:3000`, credentials: admin/admin123. Dashboard: SaaS Monitoring Overview.

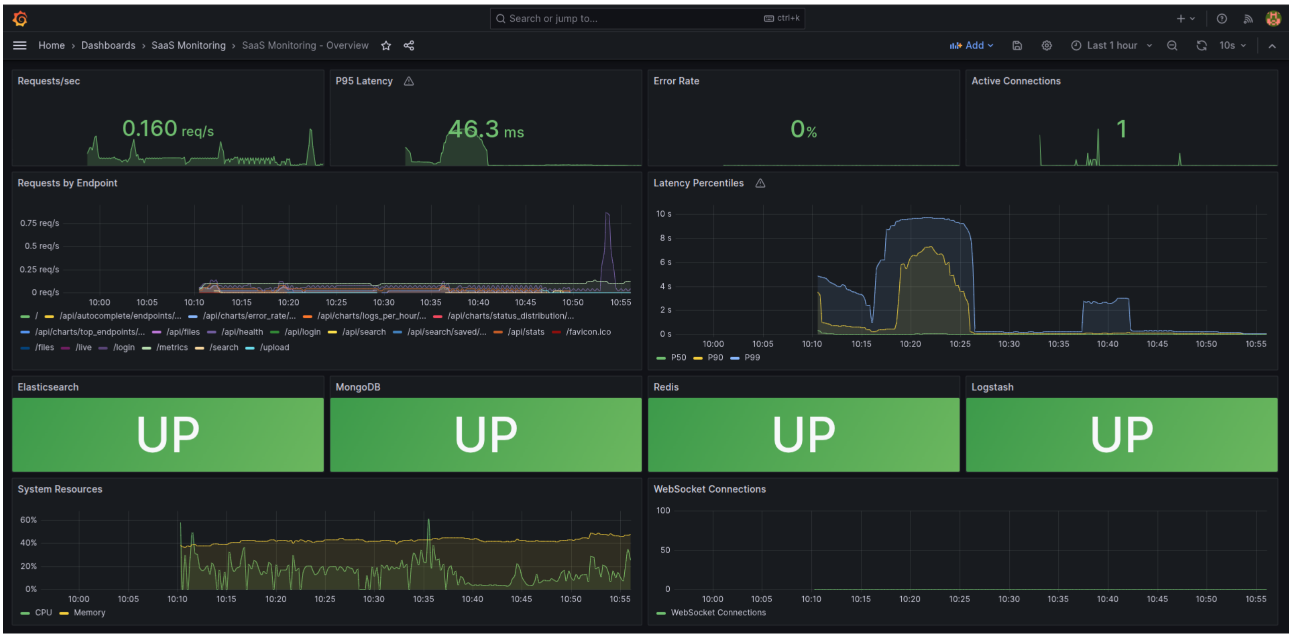


Figure 8.5: Grafana Dashboard

8.6 Kibana Visualization

Access `http://localhost:5601` to explore raw logs and create custom visualizations.

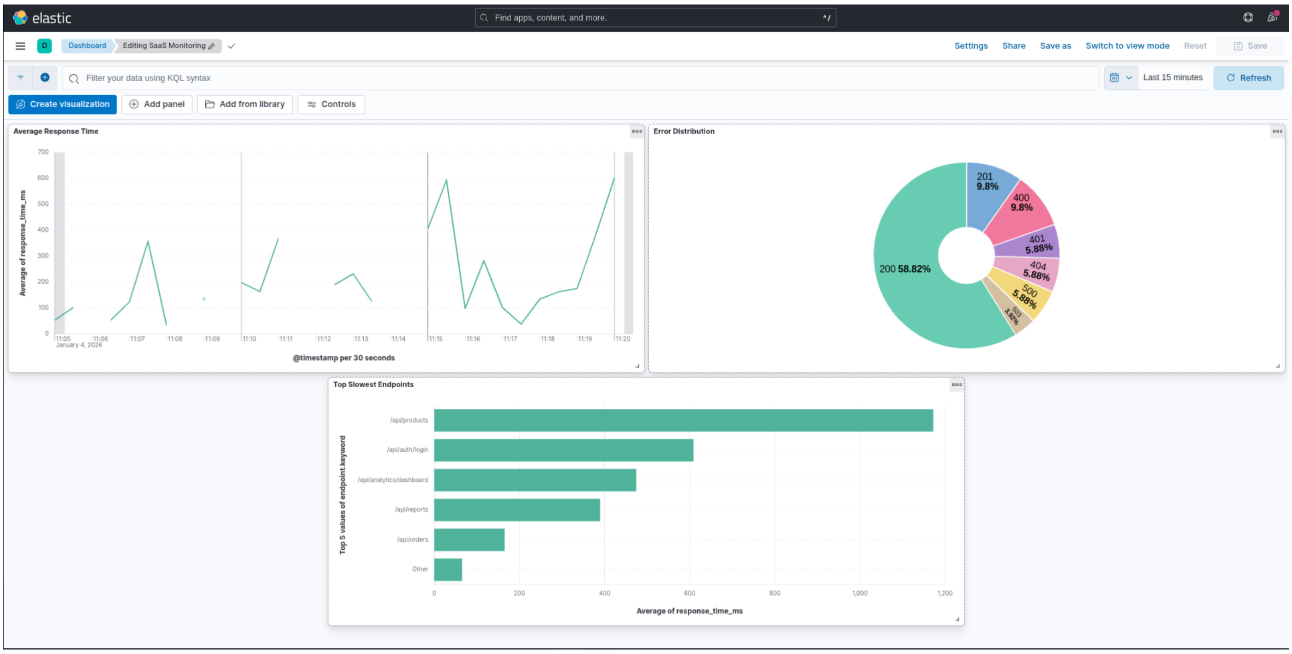


Figure 8.6: Kibana Dashboard

Chapter 9

Testing and Validation

9.1 API Tests

Endpoint	**P50**	**P90**	**P99**
/api/health	25ms	45ms	120ms
/api/stats	80ms	150ms	350ms
/api/search	120ms	280ms	650ms

Table 9.1: Response Time Percentiles

9.2 Validation Checklist

✓ All API endpoints respond correctly
✓ Health checks return accurate status
✓ Real-time streaming functional
✓ File upload and processing works
✓ Search returns relevant results
✓ Prometheus metrics exposed
✓ Grafana dashboards display data

Chapter 10

Challenges and Solutions

Elasticsearch Memory: Reduced heap to 256MB, added resource limits
Logstash Complexity: Used tags for input identification, conditional filters
WebSocket Scalability: Flask-SocketIO with Redis message queue
Kibana Dependencies: Docker health checks with service_healthy condition
Performance: Redis caching, query optimization, pagination

Chapter 11

Future Improvements

Short-term:

- ML anomaly detection
- Slack/email alerting
- Custom dashboards

Long-term:

- Kubernetes deployment
- Multi-tenancy support
- API versioning

Chapter 12

Conclusion

This project successfully delivered a comprehensive SaaS Log Monitoring Platform integrating ELK Stack, MongoDB, Redis, and modern observability tools.

Achievements:

- Centralized log collection from multiple sources
- Sub-second search with Elasticsearch
- Real-time streaming via WebSocket
- Prometheus metrics and Grafana dashboards
- Containerized deployment with Docker Compose

The modular architecture ensures extensibility and easy scaling.

Chapter 13

Appendices

13.1 API Reference

Endpoint	**Method**	**Description**
/api/health	GET	Service health
/api/stats	GET	Log statistics
/metrics	GET	Prometheus metrics
/api/logs	GET	Query logs
/api/search	GET	Search logs
/api/upload	POST	Upload file
/api/files	GET	List files

Table 13.1: API Endpoints

13.2 References

1. Elasticsearch: https://www.elastic.co/guide/
2. Flask: https://flask.palletsprojects.com/
3. Docker: https://docs.docker.com/
4. Prometheus: https://prometheus.io/docs/