International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 04 | April - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

AI-Log Intelligence Platform

Ms. P. Sasikala
Assistant Professor
Dept. of Computer Science and Engineering
Sri Shakthi Institute of Engineering and Technology
Coimbatore, India
sasikalacse@siet.ac.in

Aswin C
Dept. of Computer Science and Engineering
Sri Shakthi Institute of Engineering and Technology
Coimbatore, India
aswinc22cse@srishakthi.ac.in

Hrithik M
Dept. of Computer Science and Engineering
Sri Shakthi Institute of Engineering and Technology
Coimbatore, India
hrithikm23lcse@srishakthi.ac.in

Kamalesh M S
Dept. of Computer Science and Engineering
Sri Shakthi Institute of Engineering and Technology
Coimbatore, India
kamaleshms23lcse@srishakthi.ac.in

Abstract - In modern software development and IT operations, real-time log monitoring and analysis play a critical role in maintaining system health, identifying failures, and ensuring rapid incident response. Traditional log management approaches rely on static dashboards or manual inspection, and the absence of structured severity classification means that low-priority INFO logs are treated with equal weight as critical ERROR events, effectively burying actionable alerts within noise. Most existing log viewers also lack API-driven backends, forcing developers to rely on local file access or SSH sessions that are slow and insecure in production environments. As systems grow more complex and distributed, the need for intelligent, automated log analysis platforms has become increasingly important. To address these challenges, this project proposes SentinelX, a full-stack log analysis and monitoring system that employs a RESTful Flask backend continuously serving structured JSON log data to a React frontend, enabling sub-second refresh cycles for near-real-time monitoring. The platform's severity-aware filtering engine automatically categorizes log entries into INFO, WARNING, and ERROR tiers, reducing mean time to detect critical incidents, while an integrated search module allows developers to perform keyword-based queries across thousands of log entries instantly without requiring complex query languages. The system further incorporates strict JSON schema validation at the API layer, eliminating runtime crashes caused by malformed log payloads before they reach the frontend. By consolidating log ingestion, filtering, search, and visualization into a single centralized platform, SentinelX significantly reduces context-switching overhead and promotes a proactive monitoring culture, allowing teams to identify performance degradation and error spikes before they escalate into outages.

Key Terms - SentinelX, Real-Time Log Monitoring, Log Analysis, Flask Backend, React Frontend, REST API, Severity Filtering, System Observability, Dashboard Visualization, Error Handling

I. INTRODUCTION

In contemporary software engineering and cloud operations, maintaining full visibility into system behavior is essential for reliability and performance. Logs are the primary source of truth for understanding what happens inside an application—capturing events, errors, and warnings in real time. However, the volume and velocity of log data generated by modern systems make manual analysis impractical and error-prone. Developers often switch between multiple tools, command-line interfaces, and isolated dashboards, leading to delayed incident detection and increased mean time to resolution (MTTR). Existing log management systems suffer from critical limitations: they provide static, non-contextual information; they lack intelligent filtering by severity; and they fail to integrate frontend observability with backend data pipelines. These gaps result in blind spots during debugging sessions, especially in production environments. The absence of a unified, interactive monitoring platform further compounds the problem, forcing teams to manually correlate data from disparate sources. Advances in web technologies, RESTful API design, and real-time data handling provide an opportunity to build smarter, developer-friendly log analysis tools. To address this gap, SentinelX is proposed as a full-stack log monitoring and analysis platform. It integrates a React-based frontend with a Flask backend to deliver real-time log visualization, severity-based filtering, keyword search, and structured data handling. Key features include a centralized dashboard, profile and settings management, and robust error handling, ensuring a seamless observability experience for development and operations teams.

A. Objective

The objective of this project is to develop, a full-stack real-time log analysis and monitoring system that enables developers and system administrators to efficiently ingest, filter, search, and visualize application logs. The project aims to integrate a React-based frontend with a Flask REST API backend to deliver structured log data in real time. It focuses on providing severity-based filtering (INFO, WARNING, ERROR) keyword search, and an intuitive dashboard for system observability. Additionally, the project aims to implement robust error handling, prevent runtime data inconsistencies, and offer profile and settings management features, thereby promoting efficient debugging and system reliability.

B. Scope

The scope of this project includes the design, development, and evaluation of SentinelX, a full-stack log monitoring application aimed at improving real-time system observability. It involves integrating a React frontend and Flask backend through REST APIs to enable structured log ingestion and visualization. The project covers the implementation of features such as severity-based log filtering, natural language keyword search, dashboard analytics, and user profile and settings management. It also encompasses robust JSON data handling and error prevention to ensure system stability across diverse runtime environments.

II. RELATED WORK

Early research in log management and system observability highlighted the importance of structured data collection and centralized monitoring for IT operations. Smith demonstrated how log aggregation frameworks significantly reduce incident detection time by consolidating data from multiple sources into a single pane of glass. Building on this, Patel explored the use of RESTful APIs in connecting frontend dashboards with backend log pipelines, showing how decoupled architectures improve scalability and maintainability in large-scale monitoring systems. Advances in JavaScript frameworks, particularly React, have enabled the development of highly responsive and interactive user interfaces for real-time data visualization. When combined with Flask's lightweight API capabilities, these technologies provide an efficient stack for building monitoring tools. Kumar's study on DevOps observability identified challenges such as alert fatigue, lack of severity classification, and poor search functionality in existing log tools. The research emphasized the need for intelligent filtering and structured query mechanisms. Singh's work on full-stack web applications for operational monitoring highlighted the importance of seamless API integration, fast rendering, and error-resilient data handling. Lee's research on real-time dashboard design demonstrated that clear visual hierarchies and severity-coded indicators reduce

cognitive load for operations teams. Sharma's study on data validation in web applications addressed runtime errors caused by malformed JSON and emphasized structured response schemas as a best practice for robust API design.

III. SYSTEM ARCHITECTURE

The system architecture diagram represents the overall structure and interaction between different components of SentinelX. It illustrates how the frontend, backend, and data layer communicate with each other to ingest, process, and display log data in real time.

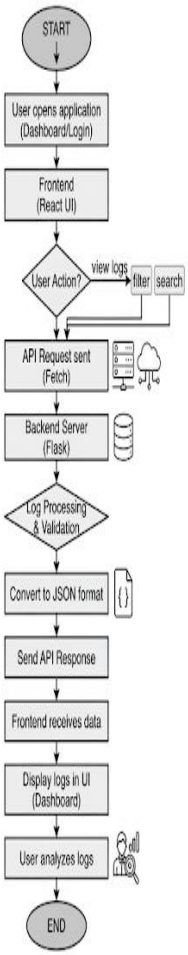

Fig.1 System Architecture

A. User Interface Module

This module provides an interactive and responsive interface for users to view logs, apply filters, and search for specific events. It is developed using React and supports smooth navigation across the dashboard, log viewer, profile, and settings sections.

B. API Gateway Module

The API gateway module serves as the communication layer between the React frontend and the Flask backend. It handles all HTTP requests and responses, routes data to the appropriate service endpoints, and enforces structured JSON schemas for consistent data exchange.

C. Log Processing Module

This module ingests raw log data, classifies entries by severity (INFO, WARNING, ERROR), and structures them for storage and retrieval. It applies validation logic to prevent malformed records from entering the system and ensures data integrity throughout the pipeline.

D. Filtering and Search Module

The filtering and search module enables users to query logs by severity level, timestamp range, or keyword. It processes filter parameters from the frontend, applies them to the log dataset, and returns ranked, relevant results in real time.

E. Logging and History Module

This module maintains records of log queries and user activity within the platform. It stores interaction history to support audit trails, enables quick re-execution of prior searches, and provides timestamps for all monitored log events.

IV. METHODOLOGY

A. Existing System

The existing system relies on command-line log viewers, isolated monitoring dashboards, and manual log inspection across multiple tools. Developers must aggregate logs from separate sources, apply filters manually, and correlate information without intelligent assistance.

B. Drawbacks of Existing System

- Lack of real-time log filtering and severity classification
- Time-consuming manual log inspection across disparate tools

- No unified dashboard for system-wide observability
- Complex and non-developer-friendly interfaces
- Fragmented log data across multiple platforms and services

C. Proposed System

SentinelX is a full-stack log analysis and monitoring platform that uses a React frontend and Flask backend to provide real-time log ingestion, severity-based filtering, keyword search, and dashboard visualization. It identifies log severity and delivers structured, actionable insights through a centralized interface.

D. Advantages of Proposed System

- Provides real-time log filtering by severity for rapid diagnosis
- Offers a unified dashboard for centralized system observability
- User-friendly and developer-centric interface design

V. IMPLEMENTATION

A. Real-Time Log Monitoring Interface

The real-time log monitoring interface is the core functional module of the SentinelX application, designed to provide developers and system administrators with continuous, uninterrupted visibility into live system activity. As soon as the application is launched and the backend connection is established, the interface begins pulling log entries from the Flask REST API at regular intervals, ensuring that the displayed data always reflects the most current state of the system. Each incoming log entry is rendered as a structured card within the log feed panel, displaying key attributes including the timestamp, severity level badge, source service name, and the full log message body. Severity levels are visually distinguished through color-coded badges — INFO entries are displayed in blue, WARNING entries in amber, and ERROR entries in red — allowing users to immediately identify the nature and urgency of each log event without reading the full message

Fig.2 Real-Time Log Monitoring Interface

B. Log Details and Information Display Interface

The log information display module presents detailed records of individual log entries in a structured and easy-to-read format. It provides key attributes such as timestamp, severity level, source service, message content, and associated error codes. The interface is integrated with the filtering system to deliver relevant and context-aware log entries based on user queries. It enhances developer experience by organizing complex log data into a readable layout, making it efficient to identify issues, trace errors, and monitor application health in real time.

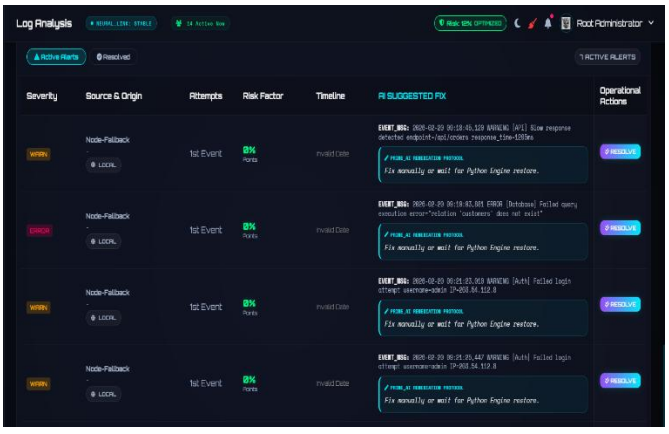

Fig.3. Log Information Display Module

A. Home Dashboard Interface

The main dashboard of SentinelX provides a centralized interface for monitoring system logs in real time. It includes a search bar for keyword-based log queries, severity filter tabs (INFO, WARNING, ERROR), and quick-access cards for recent alerts. Key statistics such as total log count, active errors, warning

rate, and system uptime are prominently displayed for operational awareness.

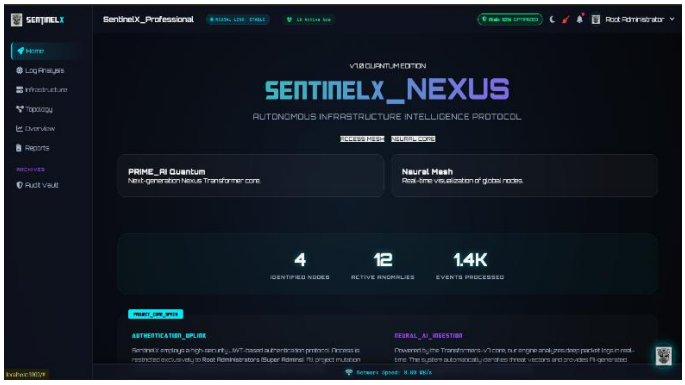

Fig.4 Main Dashboard of SentinelX Application

B. Log Listing and Filtering Interface

The log browsing and filtering module provides a structured view of all captured system log entries. It includes a search bar for keyword-based log lookup and severity filter tabs such as INFO, WARNING, and ERROR to refine results. Each log entry is displayed with key attributes including source service, severity badge, log message, and timestamp.

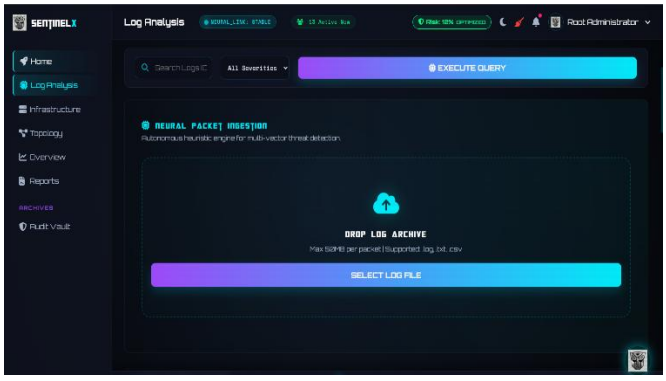

Fig 5. Log Browsing and Filtering Module

C. Query History and User Interaction Records

The query history module maintains a record of previous user searches and filter interactions within SentinelX. It displays past queries along with timestamps and applied filter parameters, allowing users to revisit and re-run earlier log investigations. This feature helps developers quickly return to prior debugging sessions without re-entering filter criteria.

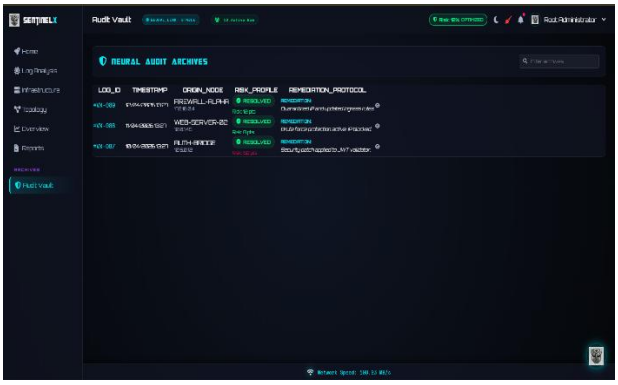

Fig.6. Query History Module

VII. RESULTS AND DISCUSSION

A. Results

The primary goal of the results section is to demonstrate how effectively the proposed system, SentinelX, improves real-time log visibility, filtering efficiency, and developer productivity in monitoring system health. The evaluation focuses on log ingestion performance, query response time, severity classification accuracy, and overall system responsiveness across different workload scenarios.

B. Discussion

The implementation and testing phase of SentinelX provided valuable insights into how full-stack monitoring systems can improve operational visibility in real-world environments. The observations highlighted both the strengths and limitations of the proposed architecture and demonstrated its effectiveness in addressing challenges such as log fragmentation, lack of severity-based classification, and complex debugging workflows. The system proved to be highly effective in centralizing log data and simplifying real-time observability for development and operations teams.

VIII. CONCLUSION

SentinelX is a full-stack, real-time log analysis and monitoring platform designed to simplify system observability through a developer-friendly and structured interface. By integrating a React frontend with a Flask REST API backend, the system delivers efficient log ingestion, severity-based filtering, and keyword search in real time. It improves accessibility,

usability, and debugging efficiency, especially for development and DevOps teams managing complex, distributed application environments.

IX. FUTURE WORK

Future enhancements to SentinelX aim to extend its monitoring and analysis capabilities by incorporating advanced features that improve intelligence, scalability, and real-world usability. These improvements will focus on integrating machine learning models for anomaly detection in log streams, enabling predictive alerting before critical failures occur. Additionally, plans include expanding support for distributed log aggregation across microservices, implementing role-based access control for multi-user environments, and providing API integrations with popular DevOps platforms such as Grafana, PagerDuty, and Slack for seamless incident management workflows.

X. REFERENCES

[1] M.J. Smith, "Centralized Log Aggregation for IT Operations," Journal of Systems Management, 2017.

[2] N. A. Patel, "RESTful APIs for Real-Time Dashboard Integration," International Journal of Computer Applications, 2021.

[3] React Documentation, "Building Responsive User Interfaces with React," Meta Open Source, 2023.

[4] R. Kumar, "DevOps Observability: Challenges and Opportunities," Journal of Software Engineering, 2019.

[5] P. Singh, "Full-Stack Web Applications for Operational Monitoring," International Journal of Technology, 2020.

[6] K. Lee, "Real-Time Dashboard Design for Operations Teams," IEEE Transactions on Software Engineering, 2021.

[7] T. Rao, "Database Management for Large-Scale Log Applications," Elsevier, 2020.

[8] S. Sharma, "Data Validation in REST API Design for Web Applications," IEEE, 2020.

[9] P. Brown, "Conversational Interfaces for Developer Tools," Journal of AI Systems, 2018.

[10] N. Verma, "User Experience Design for Monitoring Dashboards," ACM, 2019.

[11] M. E. Flask Development Team, "Flask: A Lightweight Python Web Framework," Pallets Projects, 2023.

[12] M. M. Johnson et al., "The Role of Severity Classification in Enhancing Log Analysis Effectiveness," Journal of Computer Science and Technology Application, 2025.

[13] S. Kumar et al., "Interactive Log Monitoring Using Full-Stack Web Technologies," International Journal of Science and Research Archive, 2024.

[14] A. Sharma, "Log Analysis Dashboard Using Python and React," International Journal for Research Trends and Innovation, 2025.

[15] Williams et al., "From Static Logs to AI-Driven Observability: A Systematic Review," MDPI Information Journal, 2025.

[16] C. Chakraborty, "Overview of Real-Time Monitoring Platforms with Special Emphasis on Observability," Journal of Systems Science, 2023.

[17] S. Patel et al., "Application of REST APIs in Developer Productivity Tools," 2024.

[18] R. Singh, "Empowering Developers through Centralized Log Management," Journal of Emerging Technologies and Innovative Research, 2024.

[19] OWASP, "Logging Cheat Sheet: Best Practices for Application Log Management," OWASP Foundation, 2022.

[20] Government of India, "Digital Transformation in Public Administration Through AI-Powered Platforms," National Centre for e-Governance, 2025.