Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

AI LAD: Lightweight Log Anomaly Detection System with Hybrid Detection and LLM-Assisted Analysis

R. Sivasubramanian
Assistant Professor, Dept. of Artificial Intelligence & Machine Learning
Malla Reddy University, Hyderabad,India

N. Srikar Reddy
Dept. of Artificial Intelligence & Machine Learning
Malla Reddy University
Hyderabad, India

S.Nirupam Srivarma
Dept. of Artificial Intelligence & Machine Learning
Malla Reddy University
Hyderabad, India

P.Dhanush Pavan
Dept. of Artificial Intelligence & Machine Learning
Malla Reddy University
Hyderabad, India

S.Siva Sathvik
Dept. of Artificial Intelligence & Machine Learning
Malla Reddy University
Hyderabad, India

Abstract - ***The increasing adoption of distributed architectures and cloud-based services has resulted in a rapid growth of system-generated log data produced across multiple heterogeneous platforms. Conventional log monitoring approaches mainly depend on static rule-based techniques, which are often ineffective at detecting emerging or subtle anomaly patterns. To address this limitation, this study introduces AI LAD, a lightweight log anomaly detection framework that combines heuristic methods, machine learning techniques, and LLM-assisted forensic summarization to enable efficient real-time log analysis.The proposed system applies TF–IDF feature extraction along with an Isolation Forest model to detect anomalous log entries originating from diverse log sources. A structured preprocessing pipeline is designed to manage noisy, inconsistent, and semi-structured log data. Several detection strategies were evaluated during experimentation, after which a hybrid detection framework was selected based on its superior F1-score and balanced accuracy performance.The solution is implemented as a real-time desktop application capable of generating structured outputs that include anomaly classifications, severity indicators, and concise forensic summaries produced through a lightweight large language model integration. Experimental evaluation demonstrates that the system achieves strong cross-source generalization, maintains efficient runtime performance, and offers practical applicability for automated log monitoring and anomaly detection in modern computing environments.***

Index Terms - Log Anomaly Detection, Hybrid Detection, Isolation Forest, TF-IDF, LLM-Assisted Forensics, Cross-Source Evaluation, Real-Time Monitoring, Machine Learning.

I. INTRODUCTION

Modern distributed platforms, cloud infrastructures, and enterprise software systems continuously produce vast amounts of system and application log data. These logs capture critical information about system activities, including security events, authentication attempts, performance indicators, operational states, and failure occurrences. Proper analysis of this log data plays a key role in maintaining system reliability, identifying potential security threats, and ensuring stable system operations. However, the increasing volume and heterogeneity of log data make manual monitoring both inefficient and impractical. Conventional log monitoring solutions generally depend on predefined rules and static threshold mechanisms to identify abnormal behavior. Although these techniques can effectively detect previously known patterns, they often struggle to recognize new or evolving anomalies.

Deep learning–based approaches have been introduced to improve contextual understanding of log patterns, but such methods typically require substantial computational resources, complex deployment environments, and are not always suitable for real-time applications. Another challenge arises from the structural differences among logs generated by heterogeneous systems, including HPC clusters, Windows servers, Apache web servers, and Linux-based environments. This variability complicates the development of models that can generalize effectively across multiple log sources. To overcome these challenges, this study proposes AI LAD, a lightweight hybrid log anomaly detection framework designed for efficient real-time monitoring across diverse environments. The proposed system combines heuristic severity scoring with machine learning–based anomaly detection using TF–IDF feature extraction and the Isolation Forest algorithm.

Additionally, the framework incorporates selective

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

integration of a lightweight large language model to generate structured forensic summaries for detected anomalies, improving interpretability while preserving runtime efficiency. Through lightweight modeling techniques and a modular system architecture, the proposed solution offers a scalable and practical approach for automated log analysis in real-world operational settings.

II. LITERATURE REVIEW

Log anomaly detection has progressed considerably over the past decade, evolving from traditional rule-based monitoring techniques toward more advanced machine learning and deep learning approaches. Earlier systems mainly depended on statistical analysis and predefined rules to identify abnormal patterns within system logs. Although these approaches were useful for detecting known anomalies, the rapid expansion of distributed systems, cloud platforms, and large-scale enterprise applications has created complex and heterogeneous log environments that require more flexible and scalable anomaly detection methods.

Liu et al. [1] introduced the Isolation Forest algorithm, an unsupervised anomaly detection technique based on the concept of isolating abnormal data points through recursive partitioning. Because of its computational efficiency and its ability to handle high-dimensional data, Isolation Forest has become widely used in anomaly detection tasks across multiple domains.

Chandola et al. [2] presented a comprehensive survey of anomaly detection techniques, providing an overview of various detection methods including statistical, proximity-based, and machine learning approaches.

Aggarwal [3] provided an extensive discussion of outlier detection algorithms and their applications in large-scale data analysis. In the context of log analysis, preprocessing and log parsing play an important role in enabling effective anomaly detection.

He et al. [4] proposed Drain, an online log parsing method that converts raw log messages into structured templates using a fixed-depth tree structure. This transformation allows log data to be processed more efficiently by automated analysis systems.

Du et al. [5] later introduced DeepLog, which applies recurrent neural networks to learn sequential patterns in system logs and detect anomalies when deviations from normal sequences occur. More recent research has focused on improving robustness and adaptability by combining multiple detection techniques.

Zhang et al. [6] conducted a survey of modern log anomaly detection approaches and highlighted key challenges such as heterogeneous log formats, cross-source variability, data imbalance, and real-time processing constraints.

Chen et al. [7] proposed hybrid frameworks that integrate rule-based heuristics with machine learning algorithms to improve detection accuracy and reliability in operational environments.

Kumar et al. [8] further explored cross-source log analysis and emphasized the difficulty of building anomaly detection models that generalize effectively across logs generated from different platforms.

Sharma et al. [9] demonstrated that combining TF-IDF feature extraction with Isolation Forest can effectively identify anomalous patterns within log datasets while maintaining efficient computational performance. In addition, recent advancements in large language models have introduced new possibilities for automated log interpretation.

The Gemini model family, introduced by Google Research [10], demonstrates strong capabilities in contextual language understanding and summarization, which can support automated explanation and forensic analysis of detected anomalies.

Despite these advancements, many existing approaches either rely on computationally intensive deep learning architectures or depend on static rule-based monitoring systems that lack adaptability. Furthermore, relatively little research has focused on lightweight and deployable log anomaly detection platforms capable of performing real-time monitoring across heterogeneous environments.

To address these challenges, this research proposes AI LAD, a lightweight hybrid log anomaly detection framework that integrates heuristic analysis, TF-IDF feature representation, Isolation Forest-based anomaly scoring, and LLM-assisted forensic summarization within a practical desktop-based deployment architecture.

III. PROPOSED METHODOLOGY

The system follows a structured hybrid anomaly detection methodology designed to analyze system logs efficiently across heterogeneous environments. The methodology integrates log preprocessing, feature extraction, anomaly detection modeling, rule-based evaluation, and LLM-assisted forensic summarization. The overall workflow supports real-time monitoring while maintaining low computational overhead.

The processing pipeline follows the sequence:

Log Input → Log Preprocessing → Feature Representation → Hybrid Anomaly Detection → Rule-Based Evaluation → LLM Forensic Summarization → Alert Output and Storage

This pipeline ensures systematic processing of log events while enabling scalable monitoring and automated anomaly detection.

1. Log Preprocessing

System logs collected from different platforms often contain semi-structured data with varying formats. Therefore, preprocessing is performed to normalize log messages and extract meaningful attributes required for further analysis.

The preprocessing stage includes the following steps:

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

- Regex-based parsing of raw log messages
- Extraction of severity indicators and relevant keywords
- Identification of IP addresses, timestamps, and event patterns
- Removal of redundant metadata fields

These operations transform raw log entries into structured representations while preserving anomaly-related patterns present in the data.

2. Feature Representation

Before applying machine learning algorithms, log messages must be converted into numerical representations. In this system, TF-IDF (Term Frequency–Inverse Document Frequency) encoding is used to convert textual log messages into sparse numerical feature vectors. TF-IDF measures the relative importance of words within the dataset and provides an efficient representation for textual anomaly detection tasks involving high-dimensional log data.

3. Anomaly Detection Modeling

The system employs a hybrid detection strategy that combines heuristic severity scoring with machine learning–based anomaly detection.The machine learning component uses the Isolation Forest algorithm, an unsupervised anomaly detection method that isolates anomalous data points through recursive partitioning. Anomaly scores are generated based on how easily log events can be separated from normal observations.

Multiple detection modes are supported:

- Heuristic-based detection
- Machine learning–based detection
- Hybrid detection combining both approaches

4. Severity Classification

Instead of using a binary anomaly classification, the system applies a multi-level severity classification mechanism. Detected anomalies are categorized into four severity levels, enabling better prioritization of system alerts.

The classification component outputs probability scores for each severity category. The final severity label corresponds to the class with the highest probability value. A refinement stage ensures that predicted severity levels remain consistent with contextual indicators present in the log message.

5. Deployment Integration

After model training and evaluation, the detection pipeline is integrated into a desktop-based monitoring platform. Incoming log events are processed using the same preprocessing and detection pipeline applied during system development.

For each analyzed log entry, the system generates structured outputs including:

- Anomaly label
- Severity level
- Anomaly score

These outputs enable automated alert generation and operational monitoring.

IV. SYSTEM ARCHITECTURE

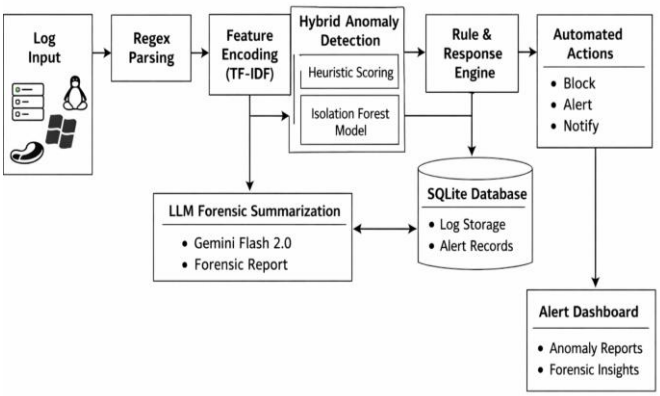

Fig. 1. AI-LAD System Architecture.

The lightweight log anomaly detection framework follows a modular and layered architecture designed for scalability, efficiency, and real-time deployment. The architecture integrates preprocessing, hybrid anomaly detection, rule-based automation, LLM-assisted forensic summarization, and persistent storage into a unified monitoring system.

4.1 Overall Architecture

The architecture consists of five primary components:

4.1.1 User Interface Layer

The user interface accepts log inputs through live monitoring streams or dataset ingestion within a desktop-based interface developed using CustomTkinter. The interface enables users to monitor logs, view detected anomalies, and review system alerts.

4.1.2 Preprocessing Module

The preprocessing module parses and normalizes raw log messages using regex-based extraction techniques. It identifies attributes such as severity indicators, timestamps, keywords, and IP patterns to structure the log data for analysis.

4.1.3 Hybrid Detection Engine

The hybrid detection engine processes structured log entries using TF-IDF feature encoding and Isolation Forest–based anomaly scoring. The anomaly score generated by the model is combined with heuristic severity indicators to produce the final anomaly classification.

4.1.4 Rule and Response Layer

The rule layer applies monitoring rules such as priority-based triggers, repeated-event windows, and temporary blocklist

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

logic. These rules enable automated alert generation and operational response handling.

4.1.5 LLM and Database Layer

The final layer integrates forensic explanation and persistent storage. The system utilizes Gemini Flash 2.0 via OpenRouter to generate structured forensic summaries describing detected anomalies. Processed logs, alerts, and responses are stored in a thread-safe SQLite database.

4.2 Detection Engine Layer

The core detection pipeline consists of the following components:

- Regex-based log parser
- TF-IDF vectorizer
- Isolation Forest model
- Heuristic severity scoring module
- Hybrid decision logic

The TF-IDF vectorizer converts processed log messages into numerical feature vectors, while the Isolation Forest algorithm computes anomaly scores based on data isolation principles. These scores are combined with heuristic severity indicators to determine the final anomaly classification.

4.3 LLM Service Integration

The system integrates Gemini Flash 2.0 through OpenRouter to generate forensic explanations for detected anomalies. The LLM service performs the following operations: Receives anomalous log entries Generates structured forensic summaries Validates outputs using Pydantic schema enforcement Handles truncated or incomplete JSON responses This integration improves interpretability while maintaining efficient detection performance.

4.4 Modularity and Scalability

The architecture follows a modular design that allows independent modification of system components such as preprocessing, detection models, rule evaluation, and LLM services. This modular structure supports future enhancements including cloud-based deployment, distributed log ingestion pipelines, additional anomaly detection models, and advanced monitoring dashboards.

V. TRAINING CONFIGURATION AND OPTIMIZATION

A. Training Configuration

The anomaly detection framework is trained using an unsupervised learning strategy that models the normal structural patterns present in system logs. The dataset is divided into training and testing subsets to evaluate the model's ability to generalize across different log sources. During training, log messages are first converted into numerical representations using TF-IDF feature encoding. These feature vectors are then used to train the Isolation Forest model, which learns the distribution of normal log patterns and identifies outliers that deviate from this distribution.

- Feature Representation: TF-IDF vectorization
- N-gram Range: (1,2)
- Maximum Features: Determined based on dataset characteristics
- Anomaly Detection Model: Isolation Forest
- Training Iterations: Approximately 3–4 optimization cycles

Validation monitoring is applied during training to ensure stable performance and to reduce the risk of overfitting. The training process focuses on learning representative patterns from normal log behavior while maintaining computational efficiency.

B. Model Optimization

The Isolation Forest model constructs an ensemble of random decision trees designed to isolate anomalous data points efficiently. The training procedure follows three main steps:

1.Random subsets of the training data are selected to construct isolation trees.

2. Each tree recursively partitions the feature space by selecting random features and split values.

3. The path length required to isolate each instance is calculated and averaged across all trees to determine the final anomaly score.

Model performance is evaluated on validation datasets to ensure consistent anomaly scoring behavior. Training continues until stable performance is achieved across evaluation metrics, ensuring reliable anomaly detection across diverse log sources.

VI. MATHEMATICAL FORMULATION

The anomaly detection task is formulated as an unsupervised outlier detection problem over system log messages. Each log entry is first transformed into a numerical feature representation and then evaluated using the Isolation Forest model to determine its anomaly score.

1. Log Representation

Let the set of system log messages be represented as:

$$L = \{x_1, x_2, x_3, \ldots, x_n\}$$

where xi represents an individual log message and n denotes the total number of log entries in the dataset.

Since log messages are textual in nature, they must be converted into numerical vectors before applying machine learning algorithms. This transformation is performed using TF–IDF encoding.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

For a given log message x x, the TF–IDF representation is defined as:

$$\phi(x) = (w1, w2, w3, \ldots, wm)$$

where m m represents the number of features (terms) extracted from the log corpus and w j denotes the TF–IDF weight of the j th term.

The TF–IDF weight for a term t t in a log message x x is calculated as:

$$TFIDF(t,x) = TF(t,x) \times IDF(t)$$

Where:

$$TF(t,x) = \sum kf(k,x)f(t,x)$$

And:

$$IDF(t) = \log(df(t)N)$$

Here:

- f(t,x) represents the frequency of term t in log message x
  - N denotes the total number of log
- d f(t) represents the number of documents containing term t

2. Isolation Forest Anomaly Scoring

After feature representation, anomaly detection is performed using the Isolation Forest algorithm, which isolates anomalies by randomly partitioning the data space.

Let h(x) denote the path length required to isolate instance x in a randomly generated isolation tree. The expected path length across all trees is represented as:

$$E(h(x))$$

The anomaly score s(x,n) for a data point x is calculated as:

$$s(x,n) = 2 - c(n)E(h(x))$$

where:

- E(h(x)) is the average path length of instance x across all isolation tress
- n is thenumber of samples used to consturct the tress
- c(n) is the average path length of unsuccesfull searches in a binary search tree

The value of c(n) is approximated as:

$$c(n) = 2H(n-1) - (n2(n-1)/n)$$

Where H(i) represents the harmonic number:

$$H(i) = \ln(i) + \gamma$$

and γ is the Euler – Mascheroni constant (approximately 0.577).

An anomaly score close to 1 indicates a highly anomalous instance, while values closer to 0 represent normal observations.

A log entry is classified as anomalous if:

$$s(x,n) > \tau$$

where τ is a threshold determined by the contamination parameter of the Isolation Forest model.

3. Hybrid Decision Function

To improve detection robustness, the system combines the anomaly score generated by the Isolation Forest model with heuristic severity indicators extracted from log attributes.

Let:

- M(x) represent the anomaly score from the isolation forest model
- H(x) represent the heuristic severity score derived from log features such as keywords, failed authentication,or suspicious IP patterns

The final anomaly decision score is defined as:

$$D(x) = \alpha H(x) + \beta M(x)$$

Where:

- α and β β are weighting parameters controlling the contribution of heuristic and model-based components
  - $\alpha+\beta=1$

The final classification decision is obtained using:

$$y(x) = \{1,0\} if\ D(x) > \theta\ otherwise$$

Where:

- y(x)=1 indicates an anomalous log event
- y(x)=0 indicates an anomalous log event
- θ represents the decision threshold.

4. Severity Classification

Detected anomalies are further categorized into multiple

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

severity levels based on contextual indicators present in the log message. Let the severity classification function be:

$$S(x) = \arg\max P(y_k \mid x)$$

Where:

- θ represents the decision threshold
- P(y k ∣ x) represents the probability of log message x belonging to severity class k
- k∈{1,2,3,4} corresponds to severity levels such as Low, Medium, High, and Critical

VII. RESULTS

After training, the final anomaly detection model is integrated into the monitoring platform to enable real-time analysis of incoming log events. During system initialization, the trained model and TF-IDF feature extractor are loaded so that the application can perform efficient inference on new log data. Incoming log entries are processed using the same preprocessing pipeline used during training, including normalization, regex-based parsing, and feature transformation. The transformed feature vectors are then evaluated by the Isolation Forest model to compute anomaly scores and determine whether a log entry represents normal activity or suspicious behavior.

For each processed log entry, the system generates structured outputs containing the anomaly label, severity level, and anomaly score. These outputs are used by the monitoring interface to update alert notifications and visualize abnormal system behavior. The platform continuously processes incoming events and maintains system statistics such as total logs processed, detected anomalies, and overall system status indicators.

Fig. 2. AI LAD monitoring dashboard displaying anomaly statistics, processed logs, system health indicators, and recent alert notifications.

The monitoring dashboard provides a centralized overview of system activity and anomaly detection results. It presents aggregated metrics including the number of detected anomalies, processed logs, and current system health indicators. Graphical visualizations are used to illustrate anomaly trends over time and system efficacy, enabling administrators to quickly assess the operational status of the monitoring environment.

Recent alerts are also displayed to highlight newly detected threats requiring attention.

To support continuous monitoring, the system includes a live log streaming interface that displays incoming log events in real time. This interface allows administrators to observe system activity as it occurs and detect abnormal patterns immediately.

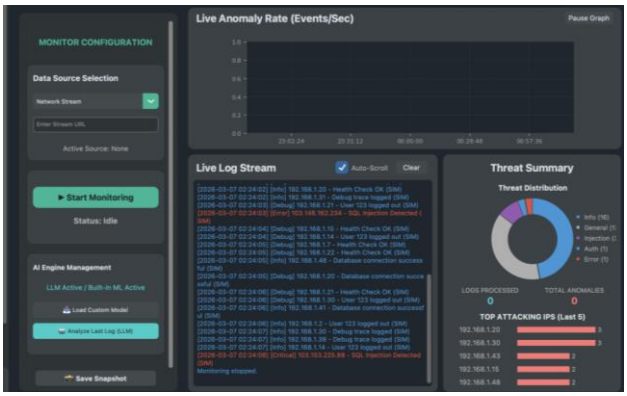

Fig. 3. Live monitoring module showing real-time log stream, anomaly indicators, and threat distribution statistics.

The live monitoring module continuously updates as new events are received. Log messages are displayed sequentially while the anomaly detection engine evaluates each entry in real time. Detected anomalies are highlighted using severity indicators to assist operators in identifying suspicious activity quickly. In addition, the interface provides statistical summaries such as threat distribution and frequently occurring attack sources, allowing users to understand system behavior at a glance.

Detected anomalies can be further examined using the forensic analysis module, which provides contextual interpretation of suspicious events.

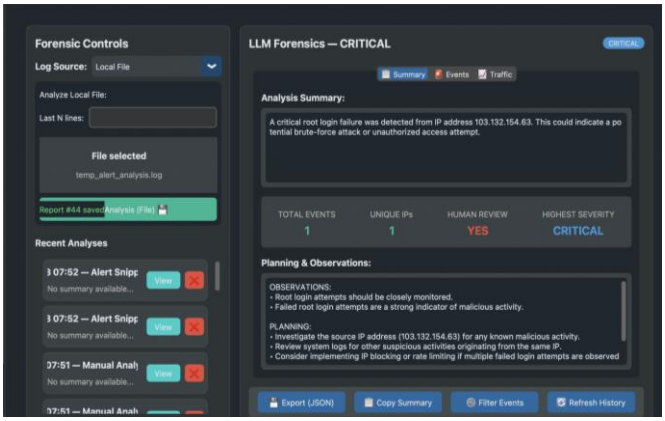

Fig. 4 Forensic analysis module presenting automated summaries, detected threat information, and recommended investigation actions.

The forensic interface presents structured summaries that describe the potential cause and impact of detected anomalies. It highlights important attributes such as detected attack type,

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

source IP addresses, and event severity levels. The system also provides investigation suggestions and monitoring recommendations, assisting analysts in understanding the context of anomalous behavior without manually inspecting large volumes of log data.

All processed events, alerts, and forensic summaries are stored for later review and reporting. This deployment structure allows the system to operate as a real-time log monitoring and anomaly detection platform, combining automated analysis, interactive visualization, and structured forensic interpretation within a unified monitoring environment.

VIII. CONCLUSION

The developed system demonstrates stable and efficient performance during experimental evaluation and cross-source testing. The obtained results indicate reliable anomaly detection capability, achieving strong F1-scores and balanced accuracy across heterogeneous log datasets. In addition, the system maintains low inference latency and high processing throughput, confirming its suitability for real-time monitoring scenarios. The integration of LLM-assisted forensic summarization improves interpretability by generating structured explanations for detected anomalies. This capability helps analysts understand abnormal system behavior more effectively without requiring manual inspection of large volumes of raw log data.

The modular desktop-based architecture further highlights the system's practical applicability for operational environments. Its lightweight design allows efficient deployment while maintaining scalability and adaptability across different log sources and monitoring conditions.

Overall, the framework provides a practical and efficient solution for automated log monitoring and anomaly detection. By combining hybrid detection techniques with structured forensic analysis, the system offers a lightweight yet powerful approach for real-time identification and interpretation of anomalous system events.

REFERENCES

[1] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation Forest," Proc. IEEE International Conference on Data Mining (ICDM), 2008.
[2] G. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," ACM Computing Surveys, vol. 41, no. 3, 2009.
[3] C. C. Aggarwal, Outlier Analysis, 2nd ed., Springer, 2017.
[4] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An Online Log Parsing Approach with Fixed Depth Tree," Proc. IEEE International Conference on Web Services (ICWS), 2017.
[5] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly Detection and Diagnosis from System Logs," Proc. ACM Conference on Computer and Communications Security (CCS), 2017.
[6] X. Zhang et al., "A Survey on Log Anomaly Detection Techniques," IEEE Access, 2021.
[7] S. Chen et al., "Hybrid Log Anomaly Detection Frameworks for Real-World Systems," IEEE Transactions on Services Computing, 2023.
[8] A. Kumar et al., "Cross-Source Log Analysis and Generalization in Anomaly Detection," ACM Transactions on Internet Technology, 2024.
[9] S. Sharma et al., "Efficient Log Anomaly Detection Using TF-IDF and Isolation Forest," Journal of Systems and Software, 2023.
[10] Google Research, "Gemini: A Family of Highly Capable Multimodal Models," 2023