Jurnal Teknik Informatika (JUTIF)
P-ISSN: 2723-3863
E-ISSN: 2723-3871
Vol. 6, No. 5, October 2025, Page. 4022-5236
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.5.5225

From Logs to Insights in the Pulp & Paper Industry: Generating Structured Alarm Reports Using LLMs and RAG

Handri Santoso*[1], Oktavianus Hendry Wijaya[2], Febri Andriani[3], Sonny Prijantono[4]

[1]Graduate Program in Information Technology, Faculty of Science and Technology, Pradita University, Indonesia
[2]Undergraduate Program in Informatics, Faculty of Science and Technology, Pradita University, Indonesia
[3,4]Technical Specialist Department, PT. Yokogawa Indonesia, Indonesia

Email: [1]handri.santoso@pradita.ac.id

Received : Aug 3, 2025; Revised : Sep 24, 2025; Accepted : Oct 22, 2025; Published : Oct 26, 2025

Abstract

Effective alarm management is essential in industrial environments to ensure operational safety and minimize costly downtime. Traditional rule-based reporting systems often struggle to handle heterogeneous alarm log formats and the complexity of natural language queries, limiting their adaptability in real-world applications. To address these limitations, this study proposes a generative alarm reporting system that integrates Large Language Models (LLMs) with a Retrieval-Augmented Generation (RAG) framework. The system converts natural language queries into structured JSON filters, enabling efficient retrieval of contextual information from historical alarm logs. Three open-source LLMs—CodeLlama-7B, LLaMA 3.1-8B, and Mistral-7B—were locally deployed and evaluated using both quantitative and qualitative methods. Experimental results show that CodeLlama-7B achieved the best overall performance, with an Exact Match Accuracy of 0.80, a Field Match score of 93.8%, and a 0% Parse Failure Rate, outperforming the other models in reliability and structural consistency. Compared to conventional rule-based approaches, the proposed LLM-RAG integration demonstrates improved relevance, interpretability, and responsiveness in alarm reporting. This work represents the first systematic benchmarking of locally deployed open-source LLMs for industrial alarm management, providing a replicable framework and highlighting their potential to advance intelligent, real-time, and domain-specific reporting in the pulp and paper industry and beyond

Keywords : Alarm Management, Large Language Model, Retrieval-Augmented Generation, Semantic Search, Text Embedding.

1. INTRODUCTION

Alarm management is a fundamental element in modern industrial control systems, particularly in critical sectors such as pulp & paper, petrochemical, energy, and manufacturing industries. Alarm systems are designed to provide early warnings of abnormal operating conditions to prevent the escalation of events that may endanger safety, damage assets, or cause process downtime. International standards such as ISA-18.2 and IEC 62682 emphasize that an effective alarm system must encompass the entire lifecycle—from planning and rationalization to operation, maintenance, and continuous improvement [1][2]. **Figure 1** showed the lifecycle outlines a structured process for effective alarm system design, deployment, and continuous improvement. It begins with defining an alarm philosophy, followed by identification, rationalization, and detailed design. Implementation leads to active operation, which is sustained through maintenance and monitored using key performance indicators (KPIs). Management of Change (MOC) governs all modifications to ensure traceability, while periodic audits ensure compliance and long-term effectiveness. This iterative model promotes alarm integrity, operator effectiveness, and system safety in complex industrial environments [3].

Vol. 6, No. 5, October 2025, Page. 4022-5236
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.5.5225

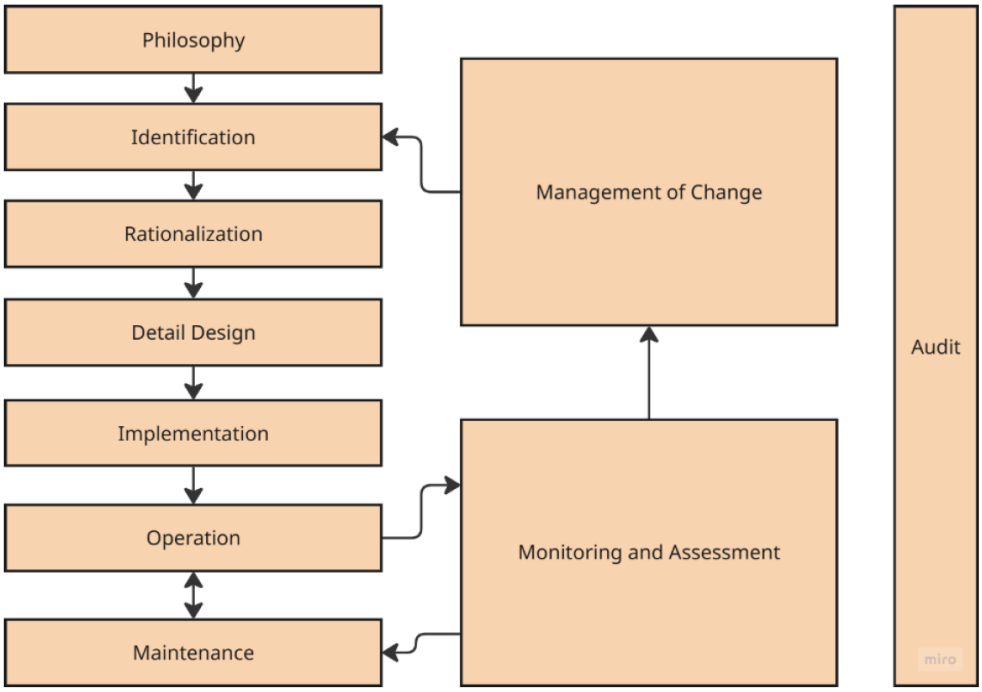

Figure 1 Alarm Management Lifecycle

Despite these guidelines, the implementation of alarm systems in practice often faces major challenges. One of the most critical issues is alarm flooding, a condition in which too many alarms are triggered simultaneously within a short time window. This phenomenon hampers operators' ability to prioritize critical alarms, leading to decision-making errors and potential financial losses. For example, industrial downtime in the United States has been estimated to cause annual losses exceeding USD 20 billion [4]. To address these challenges, researchers have proposed various approaches, including rule-based modeling [5], template-based alarm generation [6], and machine learning–based anomaly detection [7]-[10]. Although these methods have shown significant contributions, their primary limitations lie in their inflexibility when dealing with heterogeneous alarm log formats and their inability to process natural language queries issued by operators [11]-[13].

With the advancement of artificial intelligence, particularly in Natural Language Processing (NLP), approaches based on Large Language Models (LLMs) and Retrieval-Augmented Generation (RAG) have emerged as promising solutions [14]-[18]. LLMs can interpret operator queries expressed in natural language, while RAG enhances this capability by retrieving and filtering factual information from historical log databases [19][20]. This integration allows systems to translate operator instructions into structured representations (e.g., JSON filters), perform semantic searches over historical logs, and generate contextual, interpretable reports in real time. Recent studies, such as Michel et al. [21], Rodrigo et al. [22] and S. Dey [23], have demonstrated the feasibility of AI-driven alarm clustering, causal analysis and anomaly detection, highlighting the importance of data-driven techniques for reducing alarm floods and improving report accuracy.

Unlike prior studies that applied template-based or machine learning–based anomaly detection, this work is the first to integrate locally deployed LLMs with RAG for industrial alarm reporting, specifically targeting the pulp & paper sector**.** This novelty lies in combining the generative capabilities of LLMs with domain-specific retrieval, thereby enabling interpretable, accurate, and real-time alarm reporting in complex industrial environments.

Based on this background, this study proposes the development of a generative alarm reporting system that integrates LLMs with RAG to process natural language queries into structured JSON filters for

Vol. 6, No. 5, October 2025, Page. 4022-5236
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.5.5225

retrieving contextual information from historical alarm logs. The study evaluates three open-source LLMs—CodeLlama-7B, LLaMA 3.1–8B, and Mistral-7B—by benchmarking their performance using both quantitative and qualitative methods. The pulp & paper industry, with its high operational complexity and diverse alarm characteristics, is selected as the case study domain to validate the system's effectiveness and demonstrate its applicability in real-world industrial contexts.

2. METHOD

This study develops and evaluates a generative alarm reporting system that integrates **Large Language Models (LLMs)** with a **Retrieval-Augmented Generation (RAG)** framework. The objective is to convert operator-issued natural language queries into structured JSON filters that enable efficient retrieval of contextual information from historical industrial alarm logs. Importantly, all LLMs are locally deployed, ensuring enhanced data privacy, reduced latency, and seamless integration into on-premises industrial environments.

2.1. System Architecture

The proposed system architecture is depicted in **Figure 2**, which illustrates the workflow for generating contextual alarm reports by integrating Large Language Models (LLMs) with a Retrieval-Augmented Generation (RAG) framework. Alarm and event logs collected from the Distributed Control System (DCS) undergo a pre-processing phase to clean inconsistencies, extract relevant fields, and normalize data formats. These processed log entries are then embedded into vector representations using a pre-trained embedding model and stored in a vector database.

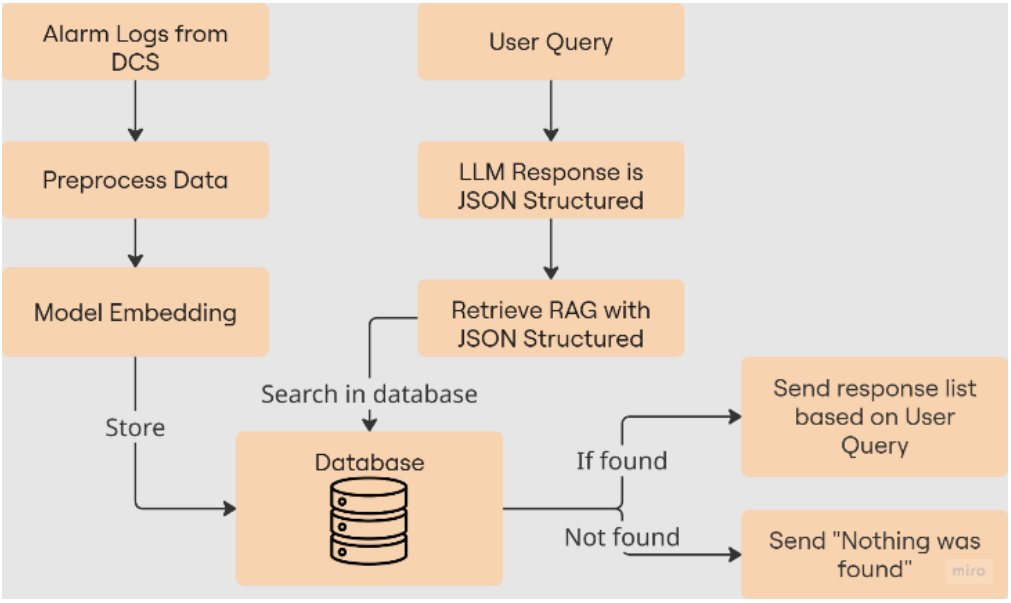

Figure 2 Proposed System architecture for Alarm Report Generation using LLM- RAG Integration

On the user interaction side, operators submit natural language queries, which are interpreted by a locally hosted LLM. The model converts the query into a structured JSON filter using a few-shot prompting strategy. This JSON structure is passed to the RAG module, which conducts a semantic search in the vector database.

If relevant entries are found, the system returns a structured list of alarms that match the user query. If no matches are identified, the system responds with a fallback message: "Nothing was found." This architecture supports real-time, flexible, and context-aware alarm reporting, enabling improved operator decision-making in dynamic and complex industrial environments.

Vol. 6, No. 5, October 2025, Page. 4022-5236
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.5.5225

2.1.1 Alarm Log Collection

Alarm logs were collected directly from the DCS of a pulp & paper manufacturing facility. Each entry contained metadata such as timestamp, domain, area, tag, and descriptive message. In total, 15 log files, each covering a 30-minute window, were collected, yielding a dataset of 69,085 entries (16,843 alarms and 52,242 events across 15 types). This ensured diversity and representativeness for system evaluation.

2.1.2. Data Preprocessing

This stage involves data cleaning (e.g., removal of empty columns), normalization of formats, and extraction of key fields such as *timestamp*, *area_id*, *tag_number*, and *description*. Furthermore, alarm types were classified (as either *alarm* or *event*) using pattern matching (regex), and additional metadata such as domain and area were extracted using rule-based logic.

2.1.3. Embedding and Storage

Log segments were transformed into vector representations using the BAAI/bge-small-en embedding model from HuggingFace. These embeddings were stored in ChromaDB, a vector database that supports efficient similarity-based retrieval.

2.1.4. Natural Language Query Processing

Users interact with the system by inputting natural language queries (e.g., *"Show alarms in domain 2 on July 15 from 12pm to 2pm"*). These queries are then processed by an LLM to generate a structured JSON filter format.

2.1.5. JSON Generation by Local LLMs

Three open-source LLMs—**CodeLlama-7B, LLaMA 3.1–8B, and Mistral-7B**—were locally deployed to generate JSON filters from natural language queries. These models were selected for their structured reasoning capabilities and compatibility with available hardware resources.
A few-shot prompting strategy was applied, providing three JSON examples in the prompt template to guide model generation and ensure consistent field structures across outputs.
The resulting JSON filters included attributes such as `timestamp`, `area_id`, `domain_id`, and `type`, which formed the basis for downstream retrieval.

2.1.6. Retrieval using RAG

The JSON filters are converted into a format compatible with ChromaDB filters, including time transformation to UNIX format and the use of logical operators such as *$gte*, *$lt*, and *$or*. Retrieval is conducted using the RAG approach to combine semantic similarity search with structured filtering, ultimately returning a contextual and relevant alarm report to the user.

2.2. Dataset Description and Data Preparation

The dataset employed in this study was collected from a Distributed Control System (DCS) operating in a pulp and paper manufacturing facility. It contains a total of 69,085 log entries, consisting of both alarm and event records. As shown in **Table 1**, the dataset includes 16,843 alarm entries across 13 alarm types, and 52,242 event entries grouped into 2 event types. The data were extracted from 15 log files, each covering a 30-minute time window, providing sufficient temporal variation and operational diversity for system evaluation.

Vol. 6, No. 5, October 2025, Page. 4022-5236
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.5.5225

Table 1 Summary of Alarm and Event Log Dataset Collected from the DCS System

Aspect	Details
Total Log Entries	69,085
Number of Log Files	15
Log Duration per File	30 minutes
Log Types	Alarm (16,843 entries, 13 types) Event (52,242 entries, 2 types)

To support the semantic modelling and retrieval tasks proposed in this work, the dataset was carefully selected to reflect the operational complexity and high event frequency typically found in real-world industrial environments. The structure of each log entry consists of five key attributes:

- Timestamp: indicating the recorded time of the entry
- Area: representing the source domain or plant zone
- Tag Number: denoting the associated control point or sensor tag
- Description: a textual description of the condition or message
- Alarm Type: used to distinguish between alarm and event entries

To enable downstream processing and semantic querying, the dataset underwent a systematic data preparation pipeline. First, all log entries were cleansed and normalized, including the removal of missing values, correction of formatting inconsistencies, and unification of timestamp formats. Next, key fields such as area, domain, and alarm type were extracted using regular expressions (regex) and standardized across files to ensure consistent field structure. Each cleaned log entry was then converted into a semantic vector representation using a pre-trained transformer embedding model (BAAI/bge-small-en) and stored in a ChromaDB vector database for high-performance similarity search.

These preparation steps ensured that the dataset was not only structurally consistent but also semantically enriched, making it suitable for Retrieval-Augmented Generation (RAG) processing. The dataset's contextual richness and structured format thus provide a robust foundation for evaluating the effectiveness of Large Language Models (LLMs) in generating real-time, context-aware alarm reports that support operator situational awareness and decision-making.

2.3 Model Evaluation

To assess the system's ability to generate accurate and valid alarm reports, three locally hosted open-source LLMs—CodeLlama:7b, LLaMA 3.1:8b, and Mistral:7b—were evaluated using a combination of quantitative and qualitative approaches.

2.3.1 Basic Performance Metrics

The Basic Performance Metrics provide a direct and objective evaluation of each model's output accuracy and efficiency [24][25]. Four key metrics were used:

a. Exact Match Accuracy (EMA)

Measures the proportion of generated outputs that exactly match the expected (ground truth) JSON response.

$$\text{Exact Match Accuracy} = \frac{N_{\text{exact\_match}}}{N_{\text{total}}} \tag{1}$$

where N_{exact_match} is the number of outputs identical to the ground truth, and N_{total} is the total number of test queries.

Vol. 6, No. 5, October 2025, Page. 4022-5236
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.5.5225

b. Average Field Match (AFM)
Calculates the average correctness of individual fields across all generated JSONs.

$$\text{Average Field Match} = \frac{1}{N}\sum_{i=1}^{N} \text{field\_match\_score}_i \tag{2}$$

where N is the number of queries and $\text{field\_match\_score}_i$ is the matching score of each field in the i^{th} query.

c. Average Latency (AL)
Indicates the mean time (in seconds) required by the model to produce an output from the time the query is submitted.

$$\text{Average Latency} = \frac{1}{N}\sum_{i=1}^{N} t_i \tag{3}$$

Where t_i is the latency time for the i^{th} query.

d. Parse Failure Rate (PFR)
Reflects how often a model generates syntactically invalid JSON outputs.

$$\text{Parse Failure Rate} = \frac{N_{\text{parse\_failure}}}{N_{\text{total}}} \tag{4}$$

where $N_{parse\_failure}$ is the number of failed parsing attempts due to syntax errors.

2.3.2. LLM-based Judge Evaluation

To complement the quantitative metrics, a semantic and structural evaluation was conducted using GPT-4o-mini as an automated LLM-based judge [26]. This model was selected due to its ability to provide consistent judgment across generative tasks.
Each output was assessed on a **0–10 scale** across four criteria:

a) **Correctness (C)**
Assesses whether the content of the JSON correctly reflects the user's query intent.
b) **Completeness (Cp)**
Measures whether all relevant fields required to satisfy the query are included in the output.
c) **Format Compliance (F)**
Evaluates whether the output conforms to JSON syntax and expected structural format.
d) **Overall Quality (Q)**
Represents the general coherence, utility, and usability of the generated report.

For each query i, the total judgment score can be represented as:

$$\text{Judge Score}_i = \frac{C_i + Cp_i + F_i + Q_i}{4} \tag{5}$$

Then, the overall LLM-based average evaluation is calculated by:

$$\text{Average LLM-Judge Score} = \frac{1}{N}\sum_{i=1}^{N} \text{Judge Score}_i \tag{6}$$

Vol. 6, No. 5, October 2025, Page. 4022-5236
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.5.5225

2.4 Test Case Design and Ground Truth Construction

To assess the system's ability to accurately convert natural language queries into structured JSON filters, a comprehensive set of 30 test queries was developed. Each query simulates a realistic operator request that an industrial alarm reporting system may encounter. These include variations in spatial filtering (e.g., area, domain), temporal constraints (e.g., specific time ranges), and tag-based identifiers linked to specific sensors or control points.

For each query, a corresponding ground truth JSON filter was manually defined to serve as the reference output. These JSON structures specify the necessary fields—such as area_id, domain_id, timestamp, type, or tag_numbers—in a format compatible with vector-based retrieval and filtering. The test cases were curated to ensure broad coverage of representative use cases, including:

- Single and multi-area/domain queries
- Time-specific alarms or events
- Combined filters with time, area, and tag constraints
- General log retrieval across types

This benchmark suite not only evaluates syntactic accuracy but also reflects semantic fidelity in transforming human language into machine-interpretable formats. The full list of natural language queries and their corresponding JSON representations is presented in **Table 2**.

Table 2 Test Cases: Natural Language Queries and Corresponding Ground Truth JSON Filters for Alarm Retrieval Evaluation

No	***Question***	***Ground Truth***
1	Show all events from area 25	{ "area_id": "25", "type": "events" }
2	List everything in area 30	{ "area_id": "30" }
3	What alarms occurred in area 45	{ "area_id": "45", "type": "alarms" }
4	Show events from area 17	{ "area_id": "17", "type": "events" }
5	Give me all logs from area 22	{ "area_id": "22" }
6	Show all events from domain 1	{ "domain_id": "01", "type": "events" }
7	List alarms from domain 2	{ "domain_id": "02", "type": "alarms" }
8	What happened in domain 3	{ "domain_id": "03" }

Vol. 6, No. 5, October 2025, Page. 4022-5236
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.5.5225

9	Give me events from domain 4	{ "domain_id": "04", "type": "events" }
10	Show alarms from domain 5	{ "domain_id": "05", "type": "alarms" }
11	What happened between 8am to 9am on November 28, 2024?	{ "timestamp": { "$gte": "2024-11-28T08:00:00", "$lt": "2024-11-28T09:00:00" } },
12	Show events on May 8, 2024 between 10am and 11am	{ "timestamp": { "$gte": "2024-05-08T10:00:00", "$lt": "2024-05-08T11:00:00" }, "type": "events" }
13	Give me alarms from noon to 2pm on July 15, 2024	{ "timestamp": { "$gte": "2024-07-15T12:00:00", "$lt": "2024-07-15T14:00:00" }, "type": "alarms" },
14	What events occurred between 3pm and 5pm on May 8, 2024	{ "timestamp": { "$gte": "2024-05-08T15:00:00", "$lt": "2024-05-08T17:00:00" }, "type": "events" }
15	Show alarms from 9am to 12pm on September 10, 2024	{ "timestamp": { "$gte": "2024-09-10T09:00:00", "$lt": "2024-09-10T12:00:00" }, "type": "alarms" }
16	Show alarms from domains 1, 2, and 3	{ "domain_id": ["01", "02", "03"], "type": "alarms" }
17	List events from areas 10 and 20	{ "area_id": ["10", "20"], "type": "events" }
18	Give me alarms from domain 5 or domain 7	{ "domain_id": ["05", "07"], "type": "alarms" }

Vol. 6, No. 5, October 2025, Page. 4022-5236
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.5.5225

19	Show events from areas 25 and 30	{ "area_id": ["25", "30"], "type": "events" }
20	List alarms from domains 8 and 9	{ "domain_id": ["08", "09"], "type": "alarms" }
21	Show events for tag 210LDAH1031	{ "tag_numbers": "210LDAH1031", "type": "events" }
22	List alarms for tag 25GJI8082D1	{ "tag_numbers": "25GJI8082D1", "type": "alarms" }
23	Give me all logs with tag 210LDAH1032	{ "tag_numbers": "210LDAH1032" }
24	Show events for tag 25GJI8082D2	{ "tag_numbers": "25GJI8082D2", "type": "events" }
25	List alarms for tag 210LDAH1033	{ "tag_numbers": "210LDAH1033", "type": "alarms" }
26	Show alarms from area 25 between 2pm and 4pm on June 15, 2024	{ "area_id": "25", "type": "alarms", "timestamp": { "$gte": "2024-06-15T14:00:00", "$lt": "2024-06-15T16:00:00" } }
27	List events from domain 3 with tag 210LDAH1031	{ "domain_id": "03", "tag_numbers": "210LDAH1031", "type": "events" }
28	Show alarms from area 25 for tag 25GJI8082D1 between 9am and 11am on July 1, 2024":	{ "area_id": "25", "tag_numbers": "25GJI8082D1", "type": "alarms", "timestamp": { "$gte": "2024-07-01T09:00:00", "$lt": "2024-07-01T11:00:00" } }
29	Give me events from domain 2 between 1pm and 3pm	{ "domain_id": "02", "type": "events", "timestamp": { "$gte": "2024-07-01T13:00:00",

Vol. 6, No. 5, October 2025, Page. 4022-5236
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.5.5225

		"$lt": "2024-07-01T15:00:00" } }

30	List alarms from area 25 with tag 25GJI8082D2 on August 5, 2024":	{
		"area_id": "25",
		"tag_numbers": "25GJI8082D2",
		"type": "alarms",
		"timestamp": {
		"$gte": "2024-08-05T00:00:00",
		"$lt": "2024-08-05T23:59:59"
		}
		}

3. RESULT

This section presents the evaluation results of three locally hosted Large Language Models (LLMs**):** CodeLlama-7B**,** LLaMA 3.1–8B, and Mistral-7B, integrated within a Retrieval-Augmented Generation (RAG) framework for generative alarm reporting. The evaluation was conducted using three complementary approaches: basic performance metrics**,** LLM-based semantic judgment**,** and human assessment**.**

3.1. Basic Performance Evaluation

Four quantitative metrics were used to measure the effectiveness of each model:

a) **Exact Match Accuracy (EMA)** – Measures how often the generated JSON filter exactly matches the ground truth.
b) **Average Field Match** – Calculates the average proportion of correctly generated fields across all test cases.
c) **Average Latency** – Captures the average response time in seconds.
d) **Parse Failure Rate** – Indicates the percentage of responses that failed JSON syntax parsing.

The results are summarized in **Table 3**:

Table 3. Comparison of Basic Performance Metrics for LLMs-RAG-Based Alarm Filter Generation

Model	Exact Match Accuracy	Field Match (%)	Latency (s)	Parse Failure Rate (%)
CodeLlama-7B	0.8000	93.83	4.04	0.00
LLaMA 3.1–8B	0.7667	87.22	3.73	0.00
Mistral-7B	0.6000	96.67	4.02	0.033

The results in **Table 3** provide a comparative analysis of the three LLM models using four core performance metrics. In terms of Exact Match Accuracy, which evaluates whether the generated JSON output precisely matches the ground truth, CodeLlama-7B achieved the highest score of 0.8000, indicating strong reliability in producing fully correct responses. LLaMA 3.1–8B followed with a score of 0.7667, while Mistral-7B recorded the lowest score at 0.6000, reflecting greater inconsistency in generating structurally and semantically accurate outputs.

However, in Average Field Match, which measures the correctness of individual fields in the JSON structure, Mistral-7B outperformed the other models with a score of 96.67%, demonstrating high precision at the field level despite its lower overall structural accuracy. CodeLlama-7B also performed well in this metric, achieving a score of 93.33%, while LLaMA 3.1–8B trailed slightly with 87/22%.

Vol. 6, No. 5, October 2025, Page. 4022-5236
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.5.5225

Regarding Average Latency, which captures the average time required to generate a response, LLaMA 3.1–8B led with the fastest response time of 3.7350 seconds, followed by Mistral-7B at 4.0228 seconds, and CodeLlama-7B at 4.0407 seconds. Although these differences are relatively minor, latency can impact system responsiveness in real-time applications.

The Parse Failure Rate assesses the ability of each model to generate syntactically valid JSON. Both CodeLlama-7B and LLaMA 3.1–8B achieved a perfect score of 0.0000, indicating that all outputs were valid and parseable. In contrast, Mistral-7B exhibited a small failure rate of 0.0333, due to occasional structural issues, such as the generation of invalid Unicode characters that disrupted JSON validity.

Overall, CodeLlama-7B demonstrated the most balanced performance across all metrics, excelling in accuracy, reliability, and structural robustness. LLaMA 3.1–8B stood out for its latency and strong exact match performance, while Mistral-7B showed the highest field-level precision but suffered from limitations in output structure consistency.

3.2. LLM-Based Evaluation

In addition to quantitative evaluation, a qualitative assessment was conducted using another LLM as an automated judge. This approach aimed to capture aspects that may not be fully reflected in the basic performance metrics, such as the model's ability to accurately interpret instructions and generate semantically complete outputs. The evaluation focused on four key dimensions: Correctness, Completeness, Format Compliance, and Overall Quality. The results of this LLM-based assessment are summarized in **Table 4**.

Table 4 LLM-Based Evaluation Scores Using GPT-4o-mini Across Four Quality Dimensions

Model	Correctness	Completeness	Format Compliance	Overall Quality
CodeLlama-7B	9.80	9.60	10.00	9.73
LLaMA 3.1–8B	9.57	9.67	9.97	9.67
Mistral-7B	8.90	9.33	9.67	9.20

The results of the LLM-Based Judge evaluation, as presented in Table 4, offer qualitative insights into each model's ability to generate semantically accurate and structurally consistent outputs. All three models achieved high scores across the four evaluated criteria: Correctness, Completeness, Format Compliance, and Overall Quality.

CodeLlama-7B emerged as the top-performing model, achieving the highest scores in Correctness (9.8000) and Overall Quality (9.7333), as well as a perfect score of 10.000 in Format Compliance, indicating strong alignment with expected output formats and consistently structured results. LLaMA 3.1–8B followed closely, with a score of 9.5667 in Correctness and 9.6667 in both Completeness and Format Compliance, reflecting reliable performance and strong semantic fidelity. Mistral-7B, while still competitive, recorded slightly lower scores in Correctness (8.9000) and Completeness (9.3333), though it maintained a high Format Compliance score of 9.6667. These results suggest that Mistral-7B may occasionally fall short in capturing complex instructions or producing fully comprehensive outputs.

Overall, the LLM-based evaluation reinforces the findings from the quantitative analysis, further validating CodeLlama-7B as the most semantically accurate and structurally robust model in this evaluation.

3.3. Human Evaluation

In this study, ourselves as human also evaluated to assess the model outputs using the same four dimensions. The average human-rated scores are shown in **Table 5**:

Vol. 6, No. 5, October 2025, Page. 4022-5236
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.5.5225

***Table 5**. Human Evaluation Scores of LLM Outputs Based on Semantic and Structural Quality*

Model	Correctness	Completeness	Format Compliance	Overall Quality
CodeLlama-7B	9.00	8.93	9.80	9.05
LLaMA 3.1–8B	8.00	7.80	8.00	7.90
Mistral-7B	7.33	7.00	8.00	7.45

Human judgments aligned closely with LLM-based evaluation, reinforcing CodeLlama-7B as the most robust model for the task.

3.4. Analysis

The comparative evaluation reveals that CodeLlama-7B consistently outperforms LLaMA 3.1–8B and Mistral-7B across all performance and quality metrics. The zero parse failure rate and high format compliance suggest its superior capability in structured JSON generation—a critical requirement for alarm reporting systems. While LLaMA 3.1 offered stronger performance than Mistral-7B, its relatively higher latency and occasional formatting errors reduced its reliability in real-time scenarios. These findings validate the feasibility of deploying locally hosted LLMs for industrial semantic querying tasks. Moreover, the inclusion of both automated and human evaluations ensures comprehensive model validation across technical and contextual dimensions.

4. DISCUSSIONS

The evaluation results presented in this research highlight several important insights regarding the performance and applicability of open-source Large Language Models (LLMs) in the context of generative alarm reporting using a Retrieval-Augmented Generation (RAG) framework.

First, CodeLlama-7B consistently demonstrated superior performance across all evaluation metrics, including Exact Match Accuracy, Field Match percentage, response latency, and parse validity. This indicates not only its capacity for accurate semantic interpretation of natural language queries but also its robustness in producing syntactically correct JSON structures—an essential feature for downstream alarm filtering and report generation. The absence of parsing errors further reinforces its reliability in high-stakes industrial applications where malformed outputs could cause critical failures or operator confusion.

Second, the alignment between LLM-based evaluation (using GPT-4o-mini) and human judgment underscores the validity of automated scoring methods for assessing semantic correctness, completeness, and structural compliance. Both evaluation approaches consistently rated CodeLlama-7B higher than LLaMA 3.1–8B and Mistral-7B, suggesting a strong correlation between the model's quantitative performance and its perceived contextual adequacy by domain experts.

Although LLaMA 3.1–8B showed moderate performance with a good balance of accuracy and completeness, it was affected by occasional formatting issues and slightly higher latency, making it less optimal for real-time applications. Mistral-7B, while lightweight and faster in inference, exhibited the lowest accuracy and the highest parse failure rate, indicating limitations in both structural generation and context representation.

Another noteworthy observation is the effectiveness of the few-shot prompting strategy, which enabled the LLMs to generalize well across a diverse range of 30 manually curated queries. This shows promise for applying such models in industrial domains without extensive fine-tuning, provided sufficient representative examples are embedded within the prompts.

Vol. 6, No. 5, October 2025, Page. 4022-5236
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.5.5225

From a system perspective, the integration of LLMs with RAG architecture significantly enhanced the system's ability to retrieve and present relevant alarm records in real time. The use of a vector database (ChromaDB) for semantic search proved effective in capturing contextual similarity from historical logs, allowing the model to produce more targeted and factually grounded responses.

However, several challenges remain. The reliance on local deployment introduces computational constraints, especially with larger models like LLaMA 3.1–8B. In production settings, optimization through quantization, caching mechanisms, or hybrid on-device/cloud inference could be explored to improve efficiency without compromising output quality.

In summary, the results validate that open-source LLMs, particularly CodeLlama-7B, can be effectively leveraged within a RAG framework to support intelligent, responsive, and context-aware alarm management systems. These findings contribute to the growing body of research on the practical deployment of LLMs in industrial automation and highlight key considerations for model selection, system architecture, and evaluation methodology.

5. CONCLUSION

This study proposed and evaluated a generative alarm reporting system for industrial environments by integrating open-source Large Language Models (LLMs) with a Retrieval-Augmented Generation (RAG) framework. The system was designed to convert natural language queries into structured JSON filters and retrieve contextually relevant alarm logs from historical data using semantic search.

Three LLMs—CodeLlama-7B, LLaMA 3.1–8B, and Mistral-7B—were compared across multiple evaluation dimensions, including syntactic accuracy, semantic completeness, latency, and human judgment. The results demonstrated that CodeLlama-7B achieved the highest performance across all evaluation metrics, including a 0% parse failure rate, superior exact match accuracy, and top-rated scores in both automated and human evaluations. These findings indicate that local LLMs can be effectively applied to enhance real-time situational awareness and decision-making in industrial alarm management systems.

While the integration of LLM and RAG has shown promising capabilities, several directions for future research remain:

a) **Domain Adaptation and Fine-tuning**: Investigating the impact of further fine-tuning LLMs on alarm-specific corpora to improve accuracy and domain alignment.
b) **Model Optimization**: Exploring quantization and pruning techniques to reduce the computational footprint of local deployments without sacrificing performance.
c) **Multimodal Integration**: Incorporating additional sensor data (e.g., temperature, vibration) to provide richer context for alarm generation and root cause analysis.
d) **Natural Language Feedback Loop**: Enabling continuous improvement of the system through human-in-the-loop learning based on operator corrections and feedback.
e) **Scalability Evaluation**: Assessing system performance in large-scale environments with high-frequency logs and multiple concurrent users.

Overall, this research highlights the practical potential of combining LLMs with RAG to enable intelligent, interpretable, and responsive alarm management solutions—paving the way for next-generation smart control systems in industrial settings.

CONFLICT OF INTEREST

The authors declare that there is no conflict of interest regarding the publication of this paper. There are no financial, professional, or personal relationships with any organizations or entities that

Vol. 6, No. 5, October 2025, Page. 4022-5236
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.5.5225

could have influenced the outcomes of this research. All contributions and results were conducted independently and transparently, including those related to the data provided by industrial collaborators.

ACKNOWLEDGEMENT

The authors would like to express their sincere gratitude to PT Yokogawa Indonesia for providing access to industrial alarm data and supporting the technical aspects of this research. Appreciation is also extended to Pradita University, particularly the Faculty of Science and Technology, for facilitating academic resources and research infrastructure. This work would not have been possible without the collaborative support and contributions from both institutions.

REFERENCES

[1] ISA, *ISA-18.2 Standard: Management of Alarm Systems*, 2016.
[2] IEC, *IEC 62682: Management of Alarm Systems for the Process Industries*, 2014.
[3] B. Hollifield and E. Habibi, *The Alarm Management Handbook*, 2nd ed., ISA, 2015. [Online]. Available: https://www.isa.org/getmedia/b360ad22-4c7b-4859-bfd4-8de6b076cc9e/Alarm-Management-2nd-Ed_Hollifield-Habibi_CH1-Final-3-15.pdf.
[4] S. Akhtar, et al., "Economic impact of alarm flooding," *Journal of Industrial Safety Engineering*, 2025.
[5] Parsa, K., Hassall, M., & Naderpour, M. (2021). Process Alarm Modeling Using Graph Theory: Alarm Design Review and Rationalization. In *IEEE Systems Journal* (Vol. 15, Issue 2, pp. 2257–2268). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/JSYST.2020.3019041
[6] Manca, G., & Fay, A. (2021). Detection of Historical Alarm Subsequences Using Alarm Events and a Coactivation Constraint. *IEEE Access*, *9*, 46851–46873. https://doi.org/10.1109/ACCESS.2021.3067837
[7] Andrade, J. R., Rocha, C., Silva, R., Viana, J. P., Bessa, R. J., Gouveia, C., Almeida, B., Santos, R. J., Louro, M., Santos, P. M., & Ribeiro, A. F. (2022). Data-Driven Anomaly Detection and Event Log Profiling of SCADA Alarms. *IEEE Access*, *10*, 73758–73773. https://doi.org/10.1109/ACCESS.2022.3190398
[8] A. Raveendran, B. Radhakrishnan, and R. Sivakumar, "Best Practices in Alarm Rationalization: A Data-Driven Approach," *ISA Transactions*, vol. 102, pp. 390–402, Jul. 2020, doi: 10.1016/j.isatra.2020.02.006.
[9] Hindy, H., Brosset, D., Bayne, E., Seeam, A., & Bellekens, X. (2019). *Improving SIEM for Critical SCADA Water Infrastructures Using Machine Learning*. https://doi.org/10.1007/978-3-030-12786-2_1.
[10] Schummer, P., del Rio, A., Serrano, J., Jimenez, D., Sánchez, G., & Llorente, Á. (2024). Machine Learning-Based Network Anomaly Detection: Design, Implementation, and Evaluation. *AI*, *5*(4), 2967-2983. https://doi.org/10.3390/ai5040143
[11] Z. Jiang, L. Wang, and M. Zhou, "Evaluation of Industrial NLP for Process Safety: Applications to Alarm Logs," *Computers & Chemical Engineering*, vol. 172, p. 108184, Apr. 2023, doi: 10.1016/j.compchemeng.2023.108184.
[12] Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., & Wang, H. (2023). *Retrieval-Augmented Generation for Large Language Models: A Survey*. http://arxiv.org/abs/2312.10997
[13] W. Zhao, X. Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., Wen, J.-R. (2023). A Survey of Large Language Models. *ArXiv*. http://arxiv.org/abs/2303.18223

Vol. 6, No. 5, October 2025, Page. 4022-5236
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.5.5225

[14] T. Brown, et al., "Language Models are Few-Shot Learners," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 1877–1901, 2020.
[15] OpenAI, "GPT-4 Technical Report," *arXiv preprint* arXiv:2303.08774, 2023.
[16] S. Bubeck, et al., "Sparks of Artificial General Intelligence: Early experiments with GPT-4," *arXiv preprint* arXiv:2303.12712, 2023.
[17] P. Lewis *et al.*, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020
[18] B. Ni, Z. Liu, L. Wang, Y. Lei, Y. Zhao, X. Cheng, Q. Zeng, L. Dong, Y. Xia, K. Kenthapadi, R. Rossi, F. Dernoncourt, M. M. Tanjim, N. Ahmed, X. Liu, W. Fan, E. Blasch, Y. Wang, M. Jiang, and T. Derr**,** "Towards Trustworthy Retrieval-Augmented Generation for Large Language Models: A Survey," *arXiv preprint* arXiv:2404.10968, Apr. 2024.
[19] V. Karpukhin *et al.*, "Dense Passage Retrieval for Open-Domain Question Answering," in *Proc. EMNLP*, 2020.
[20] Q. Zhang *et al.*, "VBase: Unifying Online Vector Similarity Search and Relational Queries via Relaxed Monotonicity," in *Proc. 17th USENIX OSDI*, 2023.
[21] Michel, D. D. E., Clovis, T. N., Christian, T. T., Mohamadou, A., & Sone, M. E. (2023). Machine Learning-Based Alarms Classification and Correlation in an SDH/WDM Optical Network to Improve Network Maintenance. *Journal of Computer and Communications*, *11*(02), 122–141. https://doi.org/10.4236/jcc.2023.112009
[22] Rodrigo, V., Chioua, M., Hagglund, T., & Hollender, M. (2016). *Causal analysis for alarm flood reduction*.
[23] S. Dey, "Effective Semantic Search: Vector Databases in the LLM Era," *Medium*, Dec. 7, 2024. [Online]. Available: https://medium.com/@soumavadey/effective-semantic-search-vector-databases-in-the-llm-era-5720f1bf0bbf
[24] P. Gupta *et al.*, "Evaluating Structured Output in Generative Models: A Case for Field-Level Matching," in *Proc. AAAI Conf. Artificial Intelligence*, vol. 37, no. 13, pp. 14972–14980, 2023.
[25] M. Biesialska, K. Biesialska, and J. M. Pino, "Evaluation of Machine Translation Output for Syntactic Correctness," *Computational Linguistics*, vol. 47, no. 4, pp. 789–822, 2021.
[26] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica, "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena," *arXiv preprint*, arXiv:2306.05685v4, Dec. 2023.