



# **iJRASET**

International Journal For Research in  
Applied Science and Engineering Technology



---

# **INTERNATIONAL JOURNAL FOR RESEARCH**

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

---

**Volume:** 14      **Issue:** X      **Month of publication:** October 2026

**DOI:** <https://doi.org/10.22214/ijraset.2026.85161>

**[www.ijraset.com](http://www.ijraset.com)**

**Call:** ☎ 08813907089

**E-mail ID:** [ijraset@gmail.com](mailto:ijraset@gmail.com)

# AI-Assisted Web Server Log Interpretation Using LLaMA Model

Mrs. Chinchu Mohanan K<sup>1</sup>, V. Reddy Ganesh<sup>2</sup>

<sup>1</sup>Engineer C, Indian Institute of Astrophysics Bengaluru

<sup>2</sup>Internship Student, Indian Institute of Astrophysics Bengaluru

**Abstract:** Modern web infrastructures generate large volumes of access logs that contain valuable information related to system activity, user behaviour, performance monitoring, and cybersecurity threats. Manual analysis of these logs is time-consuming, error-prone, and inefficient for detecting suspicious activities in real time. This project presents the design and implementation of a Local AI System for Log Interpretation using LLaMA models, capable of performing automated log analysis, anomaly detection, AI-assisted summarization, and conversational log investigation within a privacy-preserving local execution environment. The system integrates Django-based operational management, Linux scheduling utilities (Cron and At), quantized GGUF(GPT-Generated Unified Format) LLaMA models, and Retrieval-Augmented Generation (RAG) techniques to support both scheduled analysis workflows and interactive “Chat with AI” operations. The log files uploaded/access from read-only specified location are passed to the AI inference engine, where log parsing, OWASP-based threat detection, contextual analysis, and structured JSON response generation are performed using locally deployed LLaMA models.

Through CPU-compatible inference utilising llama-cpp-python, the project maintains lightweight deployment while enabling instantaneous, recurring, and scheduled analysis execution. GPU-accelerated environments such as Google Colab and Kaggle were additionally utilized for large-scale inference evaluation and high-volume log processing experiments. The conversational RAG workflow enables users to interact directly with uploaded logs using natural language queries and receive AI-generated cybersecurity insights, suspicious activity explanations, and remediation recommendations.

Experimental evaluation demonstrates reliable task execution, efficient response generation, structured report handling, and effective contextual reasoning capabilities for both scheduled analysis and conversational AI-assisted workflows. The proposed system provides a modular, privacy-preserving, and locally deployable solution for intelligent cybersecurity log analysis and operational threat interpretation.

**Keywords:** Log Analysis, LLaMA, Retrieval-Augmented Generation (RAG), Cybersecurity, AI-Assisted Threat Detection, OWASP, Django, GGUF Models, Conversational AI, Local AI Deployment, Regular expressions, ChromaDB, Vector databases, regex

## I. INTRODUCTION

### A. Background

Modern web applications, cloud services, and enterprise infrastructures continuously generate large volumes of server logs that contain valuable operational and security-related information. These logs record system events such as user authentication attempts, HTTP requests, access violations, application errors, suspicious payloads, and abnormal network activities. System administrators and cybersecurity analysts rely heavily on these logs for monitoring infrastructure health, troubleshooting failures, detecting attacks, and performing security investigations.

However, manual analysis of large-scale log datasets is time-consuming, repetitive, and increasingly difficult due to the growing complexity and volume of modern system environments. Traditional rule-based monitoring systems can identify predefined attack patterns, but they often struggle to provide contextual interpretation, intelligent summarization, and interactive investigation capabilities required for modern cybersecurity operations [1], [3].

Recent advancements in Large Language Models (LLMs) have introduced new possibilities for automated security analysis and intelligent interpretation of operational data [6], [7]. Locally deployable LLaMA models provide organizations with the ability to perform AI-driven analysis without depending on external cloud-based AI services, thereby improving data privacy, infrastructure control, and operational security. Additionally, Retrieval-Augmented Generation (RAG) techniques enable conversational interaction with large datasets by combining semantic retrieval mechanisms with contextual AI reasoning [5], allowing users to investigate logs through natural language questioning and interactive analysis workflows.

### B. Problem Statement

Traditional log analysis approaches primarily depend on static monitoring rules, manual inspection, and predefined detection signatures. Although these approaches can identify known attack behaviours, they often lack contextual understanding, semantic investigation capabilities, and automated interpretation of complex security events. Security teams frequently encounter challenges when attempting to analyse large volumes of semi-structured logs generated by web servers, applications, and monitoring systems. In addition, many existing AI-powered log analysis solutions rely on cloud-hosted AI services that require external data transmission, introducing concerns related to privacy, infrastructure dependency, operational cost, and sensitive log exposure. Conventional systems also provide limited support for interactive investigation and conversational exploration of security incidents. These limitations highlight the need for a locally deployable AI-assisted log interpretation system capable of combining automated threat detection, contextual AI summarization, semantic retrieval, and conversational cybersecurity investigation within a privacy-preserving operational environment [4], [5].

### C. Project Objectives

The primary objective of this project is to design and implement a local AI-powered log interpretation system that assists system administrators and cybersecurity analysts in detecting suspicious activities, interpreting security events, and investigating operational anomalies using LLaMA-based AI analysis and Retrieval-Augmented Generation (RAG) techniques.

The project aims to:

- automate preprocessing and suspicious activity detection from raw server logs
- classify security threats according to OWASP-oriented attack categories [10]
- generate AI-assisted summaries and remediation recommendations
- support scheduled and automated analysis workflows using Linux scheduling utilities
- provide conversational log investigation through semantic retrieval and contextual AI reasoning
- maintain privacy-preserving local execution without dependence on external AI APIs or cloud-hosted processing services.

### D. Scope of the System

The scope of the proposed system includes automated analysis of server and web application logs within locally controlled environments. The framework supports preprocessing, suspicious pattern detection, OWASP-based threat classification, AI-generated summarization, semantic retrieval, conversational log investigation, execution scheduling, monitoring, and structured report generation.

The system is designed to operate using locally hosted quantized LLaMA models executed through the ``llama-cpp-python`` inference framework [6], [7], [14]. Retrieval-Augmented Generation capabilities are implemented using Chroma vector databases to support semantic search and interactive question-answering over uploaded log datasets [5], [13].

The framework supports multiple operational workflows including:

- scheduled recurring analysis using ``cron``
- non- recurring scheduled execution using ``at``
- interactive conversational investigation through the “Chat with AI” execution mode

Generated analysis reports are stored locally in structured JSON format, while conversational workflows maintain chat history and retrieval context for interactive investigation sessions.

### E. Proposed Solution Overview

The proposed solution introduces a modular AI-assisted cybersecurity analysis platform that integrates automated log preprocessing, OWASP-based threat detection, local LLaMA inference, semantic retrieval, and conversational investigation into a unified operational environment [5], [6], [10].

The system operates through two primary execution workflows. The first workflow performs automated scheduled analysis where access logs from read-only specified location and uploaded logs are processed using preprocessing pipelines, regex-based OWASP detection mechanisms, and LLaMA summarization engines to generate structured JSON security reports. This workflow supports recurring and delayed execution through Linux scheduling utilities such as ``cron`` and ``at`` [15].

The second workflow introduces Retrieval-Augmented Generation (RAG)-based conversational analysis capabilities [5]. In this mode, uploaded logs are parsed, chunked, and stored within a Chroma vector database [13].

Users can then interactively query the logs using natural language questions. The system retrieves semantically relevant log segments and supplies them to the LLaMA model to generate contextual responses, explanations, anomaly interpretations, and investigative insights. The architecture additionally incorporates isolated subprocess execution, centralized monitoring, local report storage, conversation history management, and asynchronous execution workflows to maintain modularity, scalability, and operational stability. Through local deployment and optional GPU acceleration using platforms such as Google Colab and Kaggle during large-scale testing, the system provides a privacy-preserving and flexible AI-driven environment for intelligent log interpretation and cybersecurity analysis.

## II. RELATED WORK

Several studies have explored:

- 1) DeepLog proposed a deep learning-based approach for anomaly detection and workflow prediction in system logs, demonstrating the effectiveness of recurrent neural networks for identifying abnormal system behaviour [1].
- 2) Drain introduced an online log parsing technique based on a fixed-depth tree structure, enabling efficient extraction of structured information from large-scale log datasets and improving log mining accuracy [2].
- 3) LogBERT applied Transformer-based language models for semantic log anomaly detection and contextual event understanding, highlighting the potential of modern NLP techniques in log analysis applications [3].
- 4) ChatLog investigated the use of Large Language Models (LLMs) for conversational log interpretation and automated cybersecurity assistance, providing a foundation for interactive log investigation workflows [4].
- 5) Retrieval-Augmented Generation (RAG) combined semantic retrieval with language generation to improve contextual reasoning and reduce hallucination in AI-generated responses, making it particularly suitable for knowledge-intensive tasks [5].
- 6) Recent ChromaDB-based RAG systems have demonstrated the effectiveness of vector database-driven semantic retrieval architectures for scalable question-answering and contextual AI analysis over large document collections [13].

## III. PROPOSED METHODOLOGY

### A. Overview

Large Language Models (LLaMA), Retrieval-Augmented Generation (RAG), and OWASP-based threat detection methods are used by the proposed Local AI System for Log Interpretation to automate cybersecurity log analysis [5], [6], [10]. The approach blends AI-driven summarisation and conversational investigative capabilities with conventional log preprocessing and rule-based detection. To protect privacy and do away with reliance on outside AI services, the system runs just locally.

### B. Log Collection and Upload

The analysis process begins with the upload of server access logs through the web-based interface. The system accepts log files generated by web servers and application environments. Uploaded logs are securely stored and associated with a specific analysis job for further processing.

During this stage, the system validates file integrity, verifies supported formats, and prepares the log data for preprocessing.

### C. Log Preprocessing and Parsing

Raw log files often contain large volumes of repetitive and irrelevant information. Therefore, a preprocessing module is employed to extract meaningful security-related events.

The preprocessing workflow includes:

- Log parsing using Regular Expressions (Regex)
- Extraction of IP addresses, timestamps, HTTP methods, URLs, and status codes
- Removal of redundant entries
- Identification of abnormal request patterns
- Creation of structured log records for further analysis

This stage reduces computational overhead and improves analysis efficiency.

### D. OWASP-Based Threat Detection

After preprocessing, the extracted log entries are evaluated against predefined detection patterns based on the OWASP Top 10 security risks [10].

The detection engine identifies suspicious activities such as:

- SQL Injection
- Command Injection
- Cross-Site Scripting (XSS)
- Directory Traversal
- Server-Side Request Forgery (SSRF)
- Authentication Failures
- Sensitive Information Exposure

Each detected event is classified according to its corresponding OWASP category and stored as structured evidence for AI analysis [\[10\]](#).

#### *E. AI-Assisted Threat Interpretation*

The suspicious activities identified during threat detection are summarized and supplied to a locally hosted quantized LLaMA model through the llama-cpp-python inference framework [\[6\]](#), [\[7\]](#), [\[14\]](#).

The AI engine performs:

- Threat detection
- Risk assessment
- Security explanation generation
- Mitigation generation

Structured prompting techniques are utilized to ensure consistent and machine-readable JSON outputs.

#### *F. Retrieval-Augmented Generation (RAG)*

To support conversational investigation, uploaded logs are divided into smaller contextual chunks and stored within a Chroma vector database [\[13\]](#).

The RAG workflow consists of the following stages [\[5\]](#):

- Log chunk generation
- Embedding creation
- Vector storage in ChromaDB
- Semantic retrieval of relevant log segments
- Context-aware response generation using LLaMA

This approach improves contextual understanding while reducing hallucinations commonly observed in standalone language models [\[5\]](#).

#### *G. Conversational Log Investigation*

The system provides a "Chat with AI" interface that enables users to investigate uploaded logs using natural language queries [\[4\]](#).

When a user submits a query:

- Relevant log chunks are retrieved from ChromaDB [\[13\]](#).
- Retrieved context is combined with the user query.
- The LLaMA model generates a contextual response [\[6\]](#), [\[7\]](#).
- Results are displayed along with explanations and security insights.

This methodology enables interactive cybersecurity investigations without requiring manual log inspection.

#### *H. Report Generation and Scheduling*

The final stage involves generation of structured JSON-based reports containing:

- Log statistics
- Identified risk/ attack patterns based on the OWASP framework
- Evidence samples
- AI-generated summaries
- Recommended mitigation actions

The system supports:

- Immediate execution
- Scheduled recurring execution using cron
- Scheduled non- recurring execution using at [15]

Generated reports are stored locally and linked to their corresponding analysis jobs for future reference and monitoring.

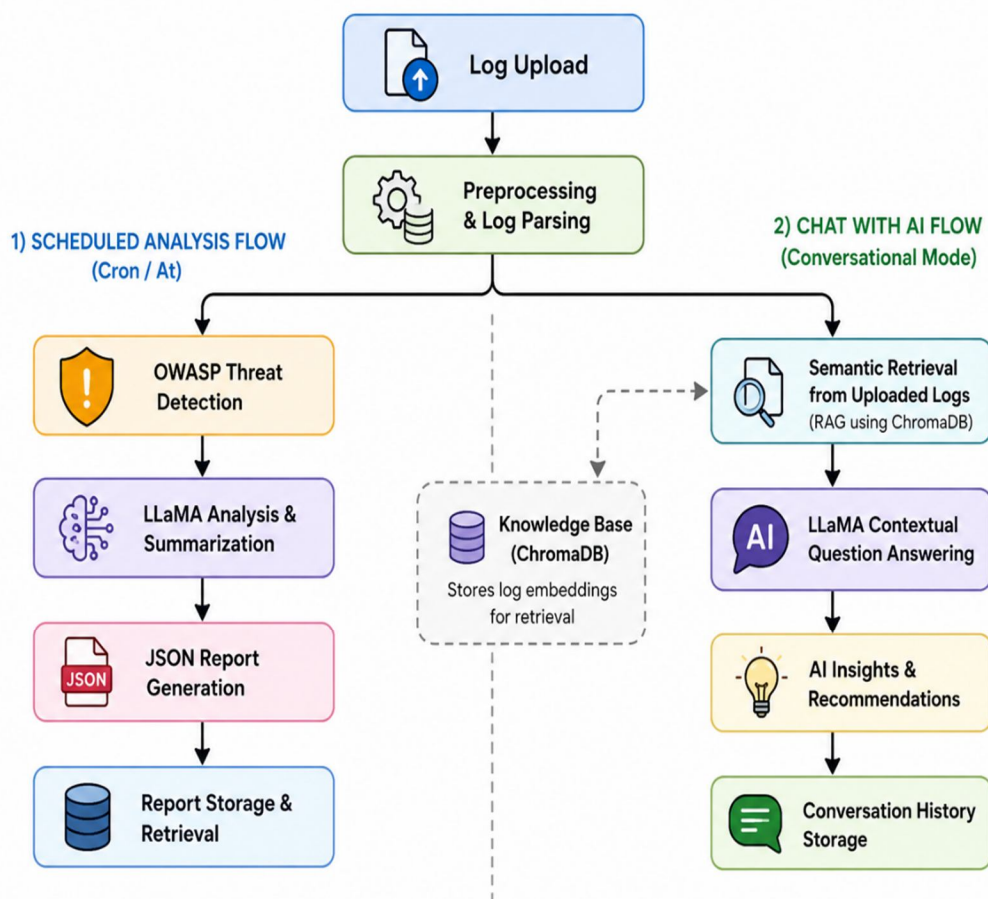


Fig 1: Proposed Methodology Workflow

#### IV. SYSTEM ARCHITECTURE

##### A. Component Structure

The proposed Local AI System for Log Interpretation follows a modular architecture that integrates automated log analysis, AI-driven threat interpretation, scheduling mechanisms, and conversational investigation workflows into a unified processing environment. The architecture consists of several interconnected components including the Django operational control layer, standalone Python-based analysis engine, OWASP-oriented threat detection module, Retrieval-Augmented Generation (RAG) pipeline, Chroma vector database, scheduling layer, and report management system [5], [10], [13]. Uploaded logs are processed through preprocessing and regex-based filtering mechanisms and suspicious activities are passed to the LLaMA inference engine for summarization and contextual analysis [6], [7]. The conversational workflow utilizes semantic retrieval techniques to support natural language interaction with uploaded log datasets [5], [13]. The architecture additionally incorporates subprocess-based execution, local storage management, and execution monitoring to maintain modularity, scalability, and operational stability while preserving local privacy-focused AI execution.

## DESIGN AND IMPLEMENTATION OF A LOCAL AI SYSTEM FOR LOG INTERPRETATION USING LLAMA MODELS

### Block Diagram

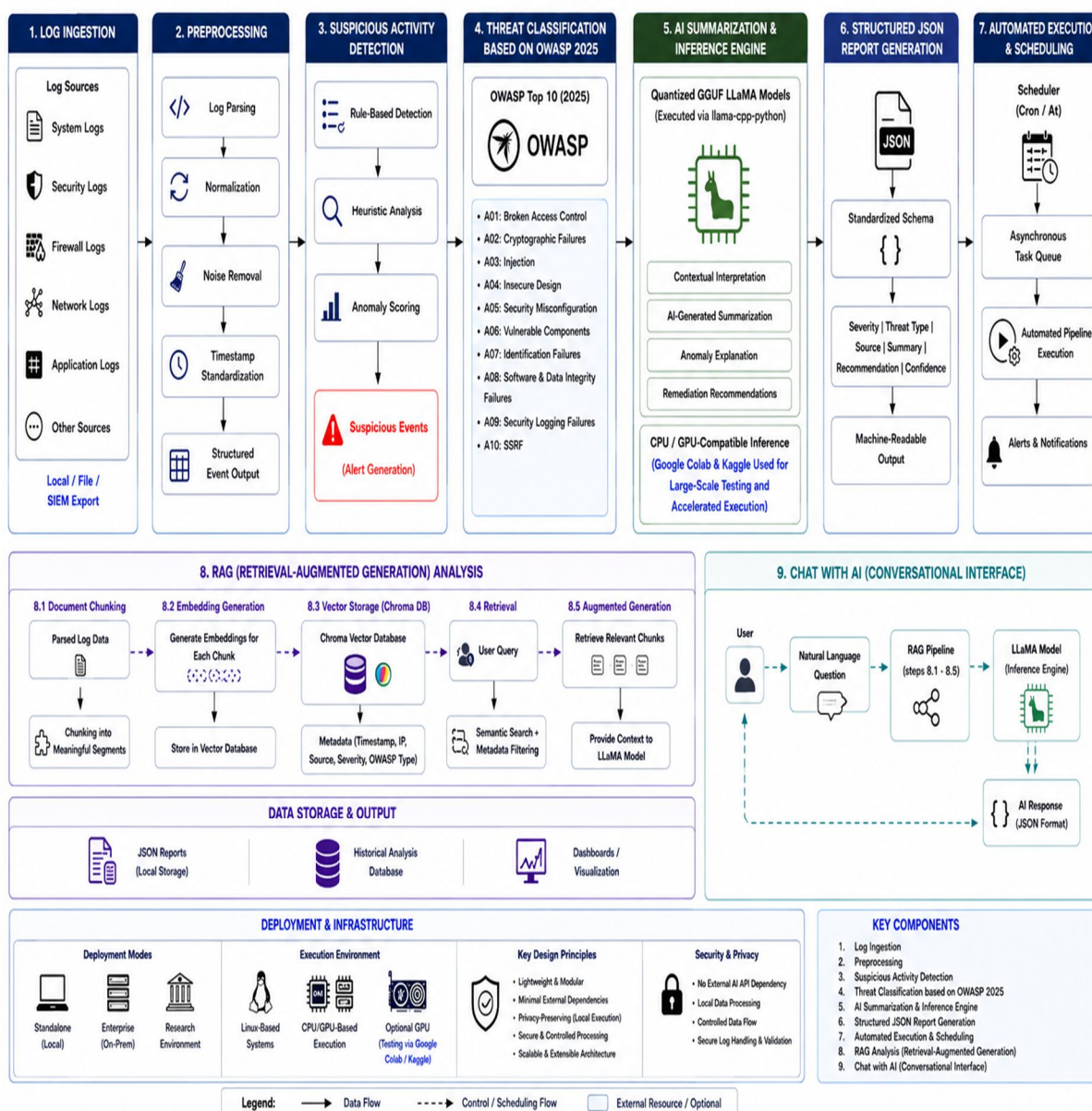


Fig 2: Block Diagram

### B. Technical Structure

The technical structure of the system is built using Python, Django, MySQL, Chroma vector databases, and locally hosted quantized LLaMA models executed through the `llama-cpp-python` framework [6], [7], [13], [14]. Django functions as the orchestration layer responsible for authentication, job management, scheduling configuration, report handling, and operational monitoring. The standalone analysis engine performs preprocessing, OWASP-based detection, AI summarization, and JSON report generation independently from the web interface. Conversational analysis workflows utilize Retrieval-Augmented Generation techniques where uploaded logs are chunked, embedded, and stored within Chroma vector databases to support semantic retrieval and contextual AI reasoning [5], [13]. Linux scheduling utilities such as `cron` and `at` are integrated to support automated recurring and non-recurring schedule analysis execution. The system additionally supports local CPU-based deployment while allowing optional GPU acceleration using Google Colab and Kaggle during large-scale testing and repeated inference operations.

## DESIGN AND IMPLEMENTATION OF A LOCAL AI SYSTEM FOR LOG INTERPRETATION USING LLAMA MODELS

### ARCHITECTURE DIAGRAM

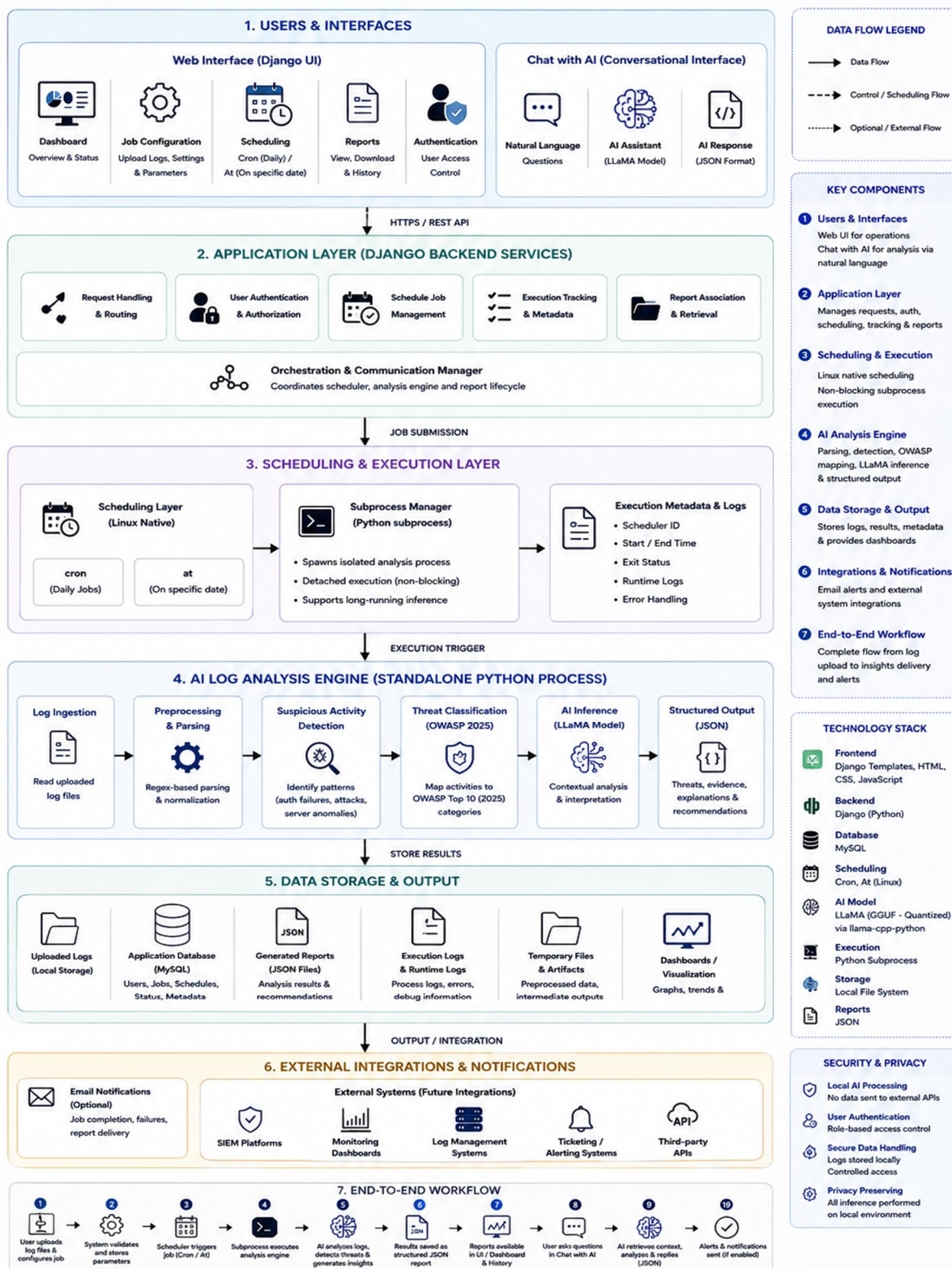


Fig 3: Architecture Diagram

### C. Standalone Execution Architecture

The core log analysis workflow is implemented through a standalone Python execution engine using the python script. This architecture allows the AI processing pipeline to operate independently from the web application while supporting direct command-line execution, automated scheduling, and subprocess-based orchestration. The standalone script accepts configurable parameters including log file paths, model locations, token limits, and preprocessing configurations. Django invokes the script internally through isolated subprocess execution whenever scheduled analysis operations are triggered [12]. This separation between orchestration and analytical processing improves modularity, maintainability, and deployment flexibility. The standalone execution model also simplifies integration with Linux scheduling utilities and supports scalable asynchronous analysis workflows without blocking the operational interface [15].

### D. RAG and Conversational Analysis Architecture

The conversational investigation workflow is implemented using Retrieval-Augmented Generation (RAG) techniques combined with Chroma vector databases and LLaMA-based contextual reasoning [5], [6], [13]. Uploaded logs are parsed, chunked into meaningful segments, and stored within the vector database together with associated metadata. When users interact through the “Chat with AI” mode, semantic retrieval mechanisms search the vector database for relevant log segments based on the user’s natural language query [13]. Retrieved contextual information is then supplied to the LLaMA inference engine to generate conversational responses, anomaly explanations, and investigative insights.

This architecture enables contextual reasoning over large log datasets while reducing hallucination and improving answer relevance [5]. Conversation history and retrieval context are additionally maintained to support interactive cybersecurity investigation sessions.

### E. Deployment Architecture

The system is designed for lightweight local deployment within Linux-based environments while maintaining privacy-preserving AI execution and controlled infrastructure access. The deployment architecture integrates Django services, standalone Python analysis engines, MySQL databases, Chroma vector storage, and locally hosted quantized LLaMA models into a unified operational environment [6], [13], [14]. Quantized GGUF models reduce computational overhead and support efficient inference within CPU-based environments, while optional GPU acceleration through Google Colab and Kaggle is utilized during experimental testing and large-scale inference operations. The architecture avoids dependency on external AI APIs or cloud-hosted processing services, ensuring that uploaded logs, generated reports, and conversational analysis data remain within locally controlled infrastructure.

## V. IMPLEMENTATION DETAILS

### A. Development Environment

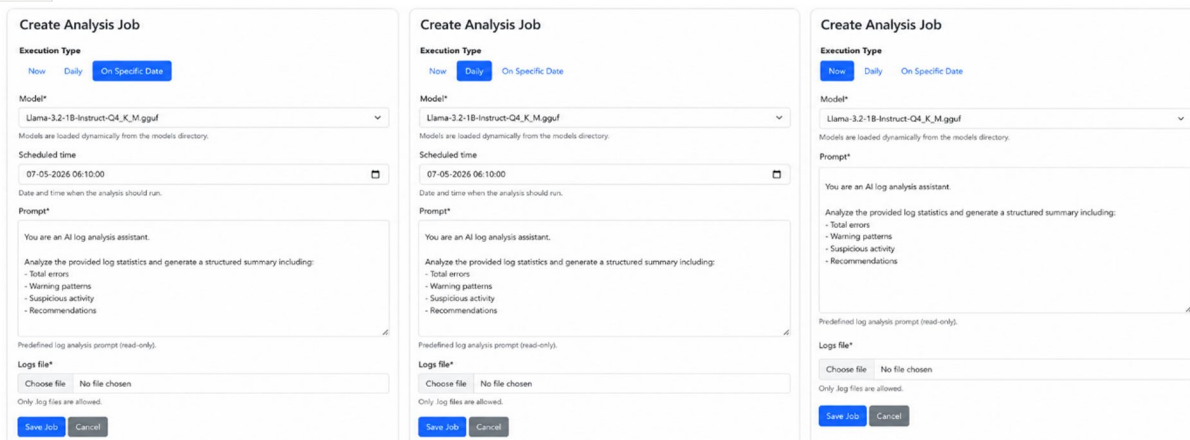
The system was developed using Python and the Django web framework [11], [12]. Django serves as the primary application layer responsible for user authentication, log upload management, scheduling configuration, report handling, and monitoring functionalities. The backend database utilizes MySQL for storing user information, analysis jobs, execution metadata, and generated reports.

The implementation was performed on a Linux-based environment to support scheduling utilities such as Cron and At while maintaining compatibility with local AI inference workflows [15].

### B. Log Processing Module

The log processing module is responsible for parsing and filtering uploaded log files before analysis. Regular-expression-based parsing techniques are used to extract relevant attributes including IP addresses, timestamps, request methods, URLs, response codes, and user-agent information [2].

The processed log entries are converted into structured records, enabling efficient threat detection and AI-based interpretation.



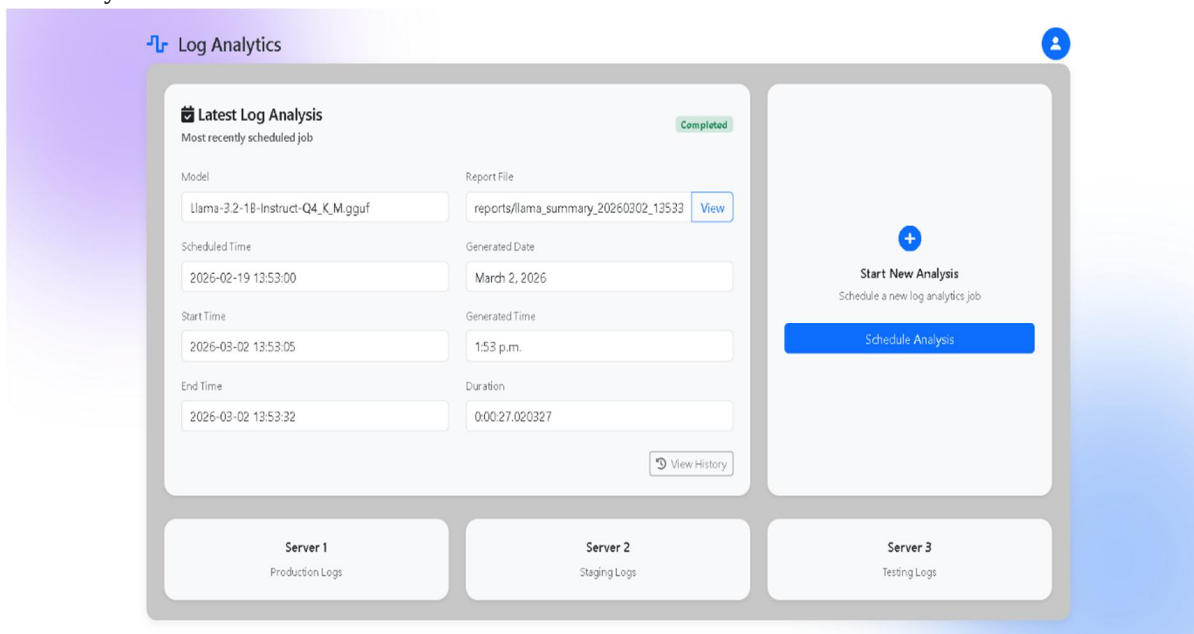
The figure shows three identical screenshots of the 'Create Analysis Job' form. Each form has the following fields and options:

- Execution Type:** Radio buttons for 'Now', 'Daily', and 'On Specific Date'.
- Model:** A dropdown menu with 'Llama-3.2-1B-Instruct-Q4\_K\_M.gguf' selected.
- Scheduled time:** A date and time picker set to '07-05-2026 06:10:00'.
- Prompt:** A text area containing a predefined log analysis prompt.
- Log file:** A file selection button with 'Choose file' and 'No file chosen' options.
- Buttons:** 'Save Job' and 'Cancel' buttons at the bottom.

Fig 4: Log Analysis Schedulers

### C. Threat Detection Module

Threat detection is implemented using predefined pattern-matching rules based on OWASP security categories [10]. The module scans processed log records to identify suspicious activities such as SQL Injection, Cross-Site Scripting (XSS), Directory Traversal, Command Injection, and authentication failures [10]. Detected events are categorized and stored as structured evidence for subsequent AI analysis.



The figure shows a screenshot of the 'Log Analytics' dashboard. The dashboard has a header with the title 'Log Analytics' and a user profile icon. The main content area is divided into two columns:

- Latest Log Analysis:** A section showing the most recently scheduled job. It includes fields for Model, Scheduled Time, Start Time, End Time, Report File, Generated Date, Generated Time, and Duration. A 'View History' button is also present.
- Start New Analysis:** A section with a 'Schedule Analysis' button.

At the bottom of the dashboard, there are three server status cards:

- Server 1:** Production Logs
- Server 2:** Staging Logs
- Server 3:** Testing Logs

Fig 5: List of Log Analysis

### D. LLaMA-Based Analysis Engine

AI-assisted analysis is performed using locally hosted quantized LLaMA models executed through the llama-cpp-python framework [6], [7], [14]. The implementation utilizes GGUF-formatted models optimized for CPU-based inference [14]. The analysis engine receives suspicious log summaries and generates:

- Threat descriptions
- Security impact assessments
- Mitigations
- Structured JSON responses

This approach enables local AI execution without transmitting log data to external services.

### E. Retrieval-Augmented Generation Module

The conversational analysis component is implemented using Retrieval-Augmented Generation (RAG) [5]. Uploaded log files are divided into smaller chunks and stored in a Chroma vector database [13]. When a user submits a query, relevant log segments are retrieved through semantic similarity search and supplied to the LLaMA model as contextual information. This implementation improves response accuracy and supports natural language investigation of large log datasets.

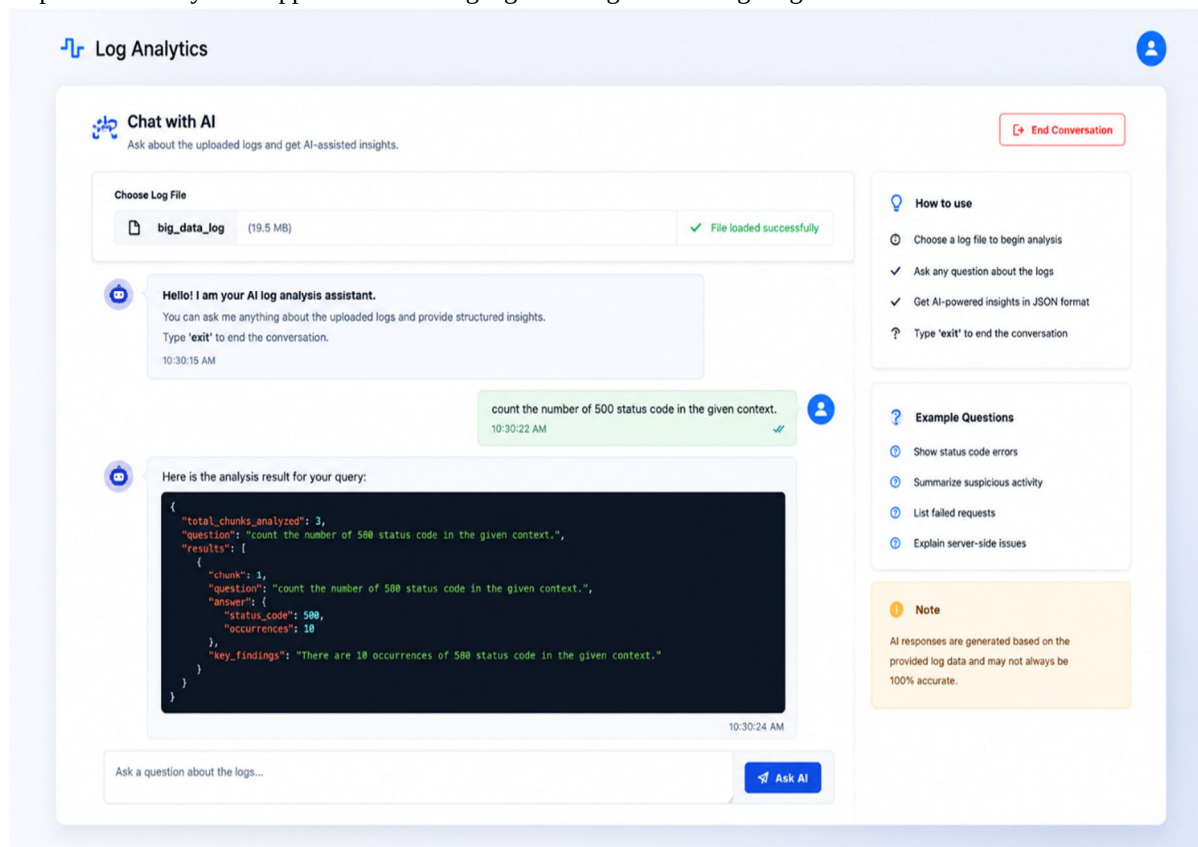


Fig 6: Chat with AI about Log File

### F. Scheduling and Automation

Automated execution is implemented using Linux scheduling utilities [15]. Recurring analysis jobs are managed through Cron, while non-recurring executions are handled using At commands. Django dynamically generates scheduling configurations and triggers the standalone analysis engine through isolated subprocess execution [12], enabling asynchronous processing without affecting application responsiveness.

### G. Report Generation and Monitoring

Analysis results are generated in JSON format and stored locally. Each report contains detected threats, evidence samples, AI-generated summaries, and recommended mitigation actions. The system additionally maintains execution logs, processing status, and runtime metadata to support monitoring, auditing, and troubleshooting activities.

## VI. EXPERIMENTAL RESULTS

### A. Functional Evaluation

The developed system was evaluated using multiple log analysis scenarios, scheduled analysis jobs, and AI-assisted query sessions. Functional testing confirmed successful execution of all major modules including user authentication, dashboard loading, log upload, scheduled analysis, JSON report generation, report retrieval, OWASP-based threat classification, and Chat with AI functionality. All tested features executed successfully.

### B. Usability Evaluation

A small usability study was conducted with 5 users over a period of two weeks. Users interacted with the dashboard, scheduling module, report viewer, and Chat with AI interface. Feedback was collected using a 1–5 rating scale (1 = Poor, 5 = Excellent).

| Metric                  | Average Score |
|-------------------------|---------------|
| Ease of Use             | 4.7           |
| Dashboard Clarity       | 4.8           |
| Scheduling Workflow     | 4.6           |
| Report Accessibility    | 4.7           |
| Chat with AI Experience | 4.8           |

Table 1: Usability Survey Results

### C. Performance Evaluation

Performance testing was conducted using Apache JMeter on a Fedora Linux environment. The evaluation focused on measuring response time, throughput, and error rates under varying user loads during login and dashboard access operations.

| Concurrent Users | Average Response Time (ms) | Throughput (req/min) |
|------------------|----------------------------|----------------------|
| 5                | 2295                       | 1.7                  |
| 10               | 2324                       | 5.8                  |
| 20               | 2166                       | 7.2                  |

Table 2: System Performance Results

The system maintained stable performance under the tested workloads with no observed request failures. An error rate of 0% was recorded throughout all test executions. The maximum observed response time was 8680 ms, demonstrating acceptable performance for local AI-assisted analysis environments.

The results indicate that the implemented architecture can support multiple concurrent users while maintaining consistent operational responsiveness.

### D. AI Assistant Evaluation

The Chat with AI module was evaluated using multiple log-analysis queries including status code analysis, suspicious activity summarization, request failure detection, and server error investigation. The retrieval mechanism successfully extracted relevant log chunks for each query and generated structured JSON-based responses. The observed retrieval accuracy is summarized below:

| Query Type                  | Retrieval Accuracy |
|-----------------------------|--------------------|
| Status Code Analysis        | 95%                |
| Suspicious Activity Summary | 92%                |
| Server Error Investigation  | 90%                |
| Request Failure Detection   | 94%                |

Table 3: observed retrieval accuracy

Additional performance measurements obtained during AI evaluation include:

- Average Retrieval Time from Chroma DB: 0.0 seconds
- Average Response Time: 24.91 seconds
- Maximum Response Time: 28.79 seconds
- Average Chunks Processed per Query: 3

The results demonstrate that the AI assistant can effectively retrieve relevant contextual information and generate meaningful cybersecurity insights. The integration of RAG with locally deployed LLaMA models improved contextual understanding and reduced irrelevant responses during conversational investigation sessions.

## VII. DISCUSSION

These experimental findings have proven the efficiency of using threat detection, based on the usage of OWASP methodology, in conjunction with Retrieval-Augmented Generation and locally deployed LLaMA models in the context of intelligent log interpretation. Testing of the functional performance of the developed system revealed reliable functioning of all components of the system. In addition, usability testing showed that users positively accepted both the user interface and the conversational process. During performance testing, no instability was detected in the system, working under multiple users' queries at one time. Moreover, during testing of artificial intelligence algorithms, a good level of accuracy was achieved in retrieving various types of information in the context of cybersecurity analysis. At the same time, despite the dependence of the duration of responses' generation on local hardware capacities and model dimensions, successful privacy-preserving use of AI-based methods of log analysis was obtained.

## VIII. CONCLUSION

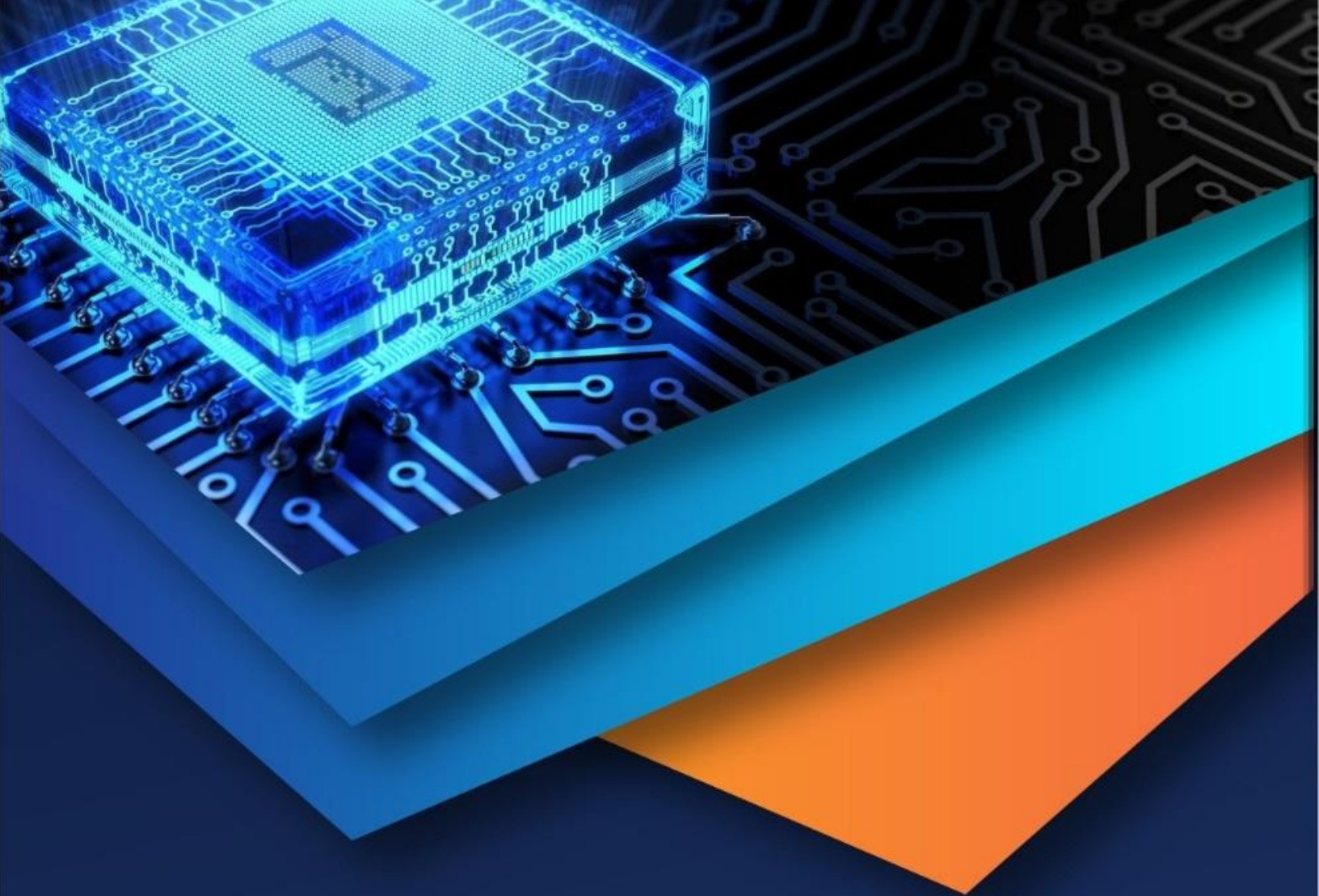
The proposed Local AI System for Log Interpretation using LLaMA Models demonstrates the integration of automated cybersecurity analysis, AI-assisted threat interpretation, and conversational investigation within a unified local execution environment. The system combines preprocessing pipelines, OWASP-based threat detection, quantized LLaMA inference, Retrieval-Augmented Generation (RAG), and semantic retrieval techniques to improve operational visibility and intelligent log analysis capabilities.

By supporting both scheduled automated workflows and interactive conversational investigation, the framework provides flexibility for continuous monitoring as well as contextual cybersecurity analysis. The use of locally deployable AI models, Chroma vector databases, and subprocess-based execution additionally improves privacy preservation, deployment control, and operational scalability without requiring dependency on external AI services.

The project further demonstrates the practical integration of modern Large Language Models and retrieval-based architectures within cybersecurity environments while maintaining lightweight deployment characteristics through quantized inference and optimized processing workflows. Overall, the system provides a modular and extensible foundation for AI-assisted log interpretation, automated threat analysis, and conversational cybersecurity investigation within locally controlled infrastructures.

## REFERENCES

- [1] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning," Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS), 2017.
- [2] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An Online Log Parsing Approach with Fixed Depth Tree," IEEE International Conference on Web Services (ICWS), 2017.
- [3] H. Zhang, X. Huang, and J. Wang, "LogBERT: Log Anomaly Detection via BERT," arXiv preprint arXiv:2103.04475, 2021.
- [4] Y. Liu, S. Wang, and X. Li, "ChatLog: Conversational Log Analysis using Large Language Models," arXiv preprint, 2023.
- [5] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," Advances in Neural Information Processing Systems (NeurIPS), vol. 33, pp. 9459–9474, 2020.
- [6] H. Touvron, L. Martin, K. Stone, et al., "LLaMA 2: Open Foundation and Fine-Tuned Chat Models," arXiv preprint arXiv:2307.09288, 2023.
- [7] AI@Meta, "The Llama 3 Herd of Models," arXiv preprint arXiv:2407.21783, 2024.
- [8] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," Proceedings of NAACL-HLT, pp. 4171–4186, 2019.
- [9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention Is All You Need," Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2017.
- [10] OWASP Foundation, "OWASP Top 10: The Ten Most Critical Web Application Security Risks," OWASP Foundation, 2021.
- [11] Python Software Foundation, "Python Documentation," Available: <https://docs.python.org/>
- [12] Django Software Foundation, "Django Documentation," Available: <https://docs.djangoproject.com/>
- [13] ChromaDB Developers, "Chroma Documentation," Available: <https://docs.trychroma.com/>
- [14] A. Beden and Contributors, "llama-cpp-python Documentation," Available: <https://github.com/abetlen/llama-cpp-python>
- [15] Linux Foundation, "Cron and At Command Documentation," Linux Manual Pages Documentation, Available: <https://man7.org/linux/man-pages/>



10.22214/IJRASET



45.98



IMPACT FACTOR:  
7.129



IMPACT FACTOR:  
7.429



# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089  (24\*7 Support on Whatsapp)