**Threat Track AI**

|  |
| --- |
|  |

**By:**

**Danyal khan**

**40856**

**Muhammad Javaid**

**40855**

**Khawaja Bilal Ahmad**

**41276**

**Supervised by:**

**Dr Mansoor Alam**

**Faculty of Computing**

**Riphah International University, Islamabad**

**Spring 2025**

**A Dissertation Submitted To**

**Faculty of Computing,**

**Riphah International University, Islamabad**

**As a Partial Fulfillment of the Requirement for the Award of the Degree of**

**Bachelors of Science in Computer Science**

**Faculty of Computing**

**Riphah International University, Islamabad**

Date: [date of final presentation]

**Final Approval**

This is to certify that we have read the report submitted by **Danyal Khan *(*40856*)***, **Khawaja Bilal Ahmad (41276), Muhammad Javaid (40855)** for the partial fulfillment of the requirements for the degree of the Bachelors of Science in Computer Science (BSCS). It is our judgment that this report is of sufficient standard to warrant its acceptance by Riphah International University, Islamabad for the degree of Bachelors of Science in Computer Science (BSCS).

**Committee:**

|  |  |
| --- | --- |
| **1** | [DR. Mansoor Alam]  (Supervisor) |
|  |  |
| **2** | [DR. Musharraf Ahmed]  (Head of Department/chairman) |

**Declaration**

We hereby declare that this document “**Threat Track AI**” neither as a whole nor as a part has been copied out from any source. It is further declared that we have done this project with the accompanied report entirely on the basis of our personal efforts, under the proficient guidance of our teachers, especially our supervisor **Dr Mansoor Alam**. If any part of the system is proved to be copied out from any source or found to be reproduction of any project from anywhere else, we shall stand by the consequences.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Danyal Khan**

**40856**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Khawaja Bilal Ahmad**

**41276**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Muhammad Javaid**

**40855**

**Dedication**

Our project is dedicated to our parents, seniors, friends, and our supervisor “Dr Mansoor Alam” who has been our continual source of inspiration and whose support has helped this project succeed. This project would not have been possible without their support.

This work is the reflection of your faith in me.

Thank you for believing.

**Acknowledgement**

First of all we are obliged to Allah Almighty the Merciful, the Beneficent and the source of all Knowledge, for granting us the courage and knowledge to complete this Project.

I am also thankful to all my teachers for their efforts during my degree, and to my friends and classmates for their help and cooperation.

Lastly, I am deeply grateful to my parents and family for their continuous support, prayers, and motivation.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Danyal Khan**

**40856**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Khawaja Bilal Ahmad**

**41276**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Muhammad Javaid**

**40855**

**Abstract**

Threat Track AI is a smart, locally deployable project built to make log analysis and threat detection faster and easier for IT teams. By using advanced AI models like LLMs(tiny llama and llama 2) and Agentic AI, it automatically scans and organizes system logs, spots important alerts, and suggests practical solutions in real time. Unlike traditional systems that often overwhelm with too many alerts and require manual responses, this solution streamlines the process. It brings together log collection, anomaly detection and then classification, also provide a solution, and a real-time dashboard in one flexible and scalable platform. The goal is to help organizations detect and respond to threats quickly, reduce operational stress, and keep their systems running securely and smoothly.

Table of Contents

[Table of Contents i](#_Toc199077053)

[List of Tables iii](#_Toc199077054)

[List of Figures iv](#_Toc199077055)

[Abstract 5](#_Toc199077056)

[Chapter 1: Introduction 6](#_Toc199077057)

[1.1 Goals and Objectives 6](#_Toc199077058)

[1.1.1 Goals: 6](#_Toc199077059)

[1.1.2 Objectives: 6](#_Toc199077060)

[1.2 Scope of the Project 7](#_Toc199077061)

[1.3 Summary: 7](#_Toc199077062)

[Chapter 2: Literature Review 8](#_Toc199077063)

[2.1 Introduction 8](#_Toc199077064)

[2.2 Background and Problem Elaboration 8](#_Toc199077065)

[2.3 Detailed Literature Review 8](#_Toc199077066)

[2.3.1 Definitions 8](#_Toc199077067)

[2.3.2 Related Research Work 1 9](#_Toc199077068)

[2.3.3 Related Research Work 2 9](#_Toc199077069)

[2.3.4 Related Research Work 3 9](#_Toc199077070)

[2.3.5 Related Research Work 4 9](#_Toc199077071)

[2.3.6 Related Research Work 5 10](#_Toc199077072)

[2.3.7 Related Research Work 6 10](#_Toc199077073)

[2.3.8 Related Research Work 7 10](#_Toc199077074)

[2.3.9 Related Research Work 8 10](#_Toc199077075)

[2.3.10 Related Research Work 9 11](#_Toc199077076)

[2.3.11 Related Research Work 10 11](#_Toc199077077)

[2.3.12 Related Research Work 11 11](#_Toc199077078)

[2.3.13 Related Research Work 12 11](#_Toc199077079)

[2.3.14 Related Research Work 13 12](#_Toc199077080)

[2.3.15 Related Research Work 14 12](#_Toc199077081)

[2.3.16 Related Research Work 15 12](#_Toc199077082)

[2.4 Literature Review Summary Table 12](#_Toc199077083)

[2.5 Research Gap 13](#_Toc199077084)

[2.6 Problem Statement 14](#_Toc199077085)

[2.7 Summary: 14](#_Toc199077086)

[Chapter 3: Requirements and Design 15](#_Toc199077087)

[3.1 Requirements 15](#_Toc199077088)

[3.1.1 Functional Requirements 15](#_Toc199077089)

[3.1.2 Non-Functional Requirements 16](#_Toc199077090)

[3.1.3 Hardware and Software Requirements 17](#_Toc199077091)

[3.2 Proposed Methodology 18](#_Toc199077092)

[3.3 System Architecture 19](#_Toc199077093)

[3.4 Use Cases 19](#_Toc199077094)

[3.4.1 Real-Time Log Upload and Analysis 19](#_Toc199077095)

[3.4.2 Threat Classification and Notification 20](#_Toc199077096)

[3.5 Database Design *(Optional)* 21](#_Toc199077097)

[3.5.1 Entities and Relationships 21](#_Toc199077098)

[3.5.2 ERD Diagram 21](#_Toc199077099)

[3.6 Class Diagram (*Optional)* 21](#_Toc199077100)

[3.7 Sequence diagram *(Optional)* 22](#_Toc199077101)

[3.8 Any Other Artifact… 22](#_Toc199077102)

[3.9 GUI Graphical User Interfaces (*Optional)* 23](#_Toc199077103)

[3.10 Summary: 23](#_Toc199077104)

[Chapter 4: Implementation and Test Cases 24](#_Toc199077105)

[4.1 Implementation 24](#_Toc199077106)

[4.1.1 Implementation of First Component/Algorithm 24](#_Toc199077107)

[4.1.2 Summary: 27](#_Toc199077108)

[References 27](#_Toc199077109)

List of Tables

[Table 1: This is Sample table caption 1](#_Toc55559466)

[Table 2: This is Sample table caption 1](#_Toc55559467)

List of Figures

Figure 1: List of Styles 1

Figure 2: IEEE Reference style 1

# Abstract

Threat Track AI is a clever, locally deployable project designed to help IT teams identify threats and analyze logs more quickly. It automatically scans and arranges system logs, identifies critical alerts, and makes real-time recommendations for workable solutions using sophisticated AI models like LLMs and Agentic AI. This solution simplifies the process in contrast to traditional systems, which frequently overwhelm with too many alerts and call for manual responses. Log collection, anomaly detection and classification, a solution, and a real-time dashboard are all combined into a single, adaptable, and scalable platform. The objective is to lessen operational stress, assist organizations in promptly identifying and addressing threats, and maintain the security and functionality of their systems.

# Introduction

Threat Track AI is an AI-powered system which is used to analyze logs automatically and with real-time threat detection. Threat Track AI will detect real-time logs and will be responding to them automatically on the spot. Threat Track AI will use LLMs and Agentic AI for classification of logs, It will filter critical alerts that the following incoming log is critical or not, and will provide instant solutions of the logs detected. Which will be helpful for IT team, reducing alert fatigue and they will focus only on the critical logs avoiding unnecessary logs. Threat Track AI will be locally deployable, ensuring security, faster processing, and operational efficiency.

## Goals and Objectives

The main and foremost objective of developing this system is to help IT teams manage system logs more effectively by automatically detecting threats and providing actionable insights saving time, reducing manual effort, and improving response to potential attacks.

### **Goals:**

* To build a locally deployable AI system that automates the classification and analysis of log data.
* To enable real-time detection of security threats using LLMs and Agentic AI technologies.
* To develop a customizable, scalable solution suitable for various IT environments.

### **Objectives:**

* To reduce alert fatigue by prioritizing and filtering only the most critical log alerts.
* To integrate intelligent agents that not only detect anomalies but also recommend suitable responses.
* To provide a user-friendly dashboard for visualizing threats and managing logs efficiently.
* To ensure the system functions entirely offline for better security and data privacy.
* To address the lack of integrated, AI-powered log monitoring systems available for local deployment.

## Scope of the Project

 Our system will be developed using FastAPI for the backend and React with Tailwind CSS for the frontend.

 We will develop an AI-powered platform that will allow IT teams to analyze log data, detect threats in real time, and get instant suggestions for resolution.

 Logs from multiple sources will be collected and analyzed using LLMs without the need for manual inspection.

 IT teams will receive filtered alerts based on severity, avoiding unnecessary alerts and helping them focus on what truly matters.

 The system will enhance the speed and accuracy of threat detection through continuous monitoring.

 Critical issues will be identified and responded to using Agentic AI, which will provide intelligent suggestions.

 The entire solution will be locally deployed to ensure data privacy and security.

## Summary:

This chapter introduces Threat Track AI, an AI-powered, locally deployable system designed to automate log analysis and real-time threat detection. The tool uses LLMs (like TinyLLaMA and LLaMA2) and Agentic AI to identify critical alerts and suggest actionable solutions. The main goal is to ease the workload on IT teams by filtering out irrelevant logs and emphasizing important alerts. The system aims to improve threat detection accuracy, response speed, and reduce alert fatigue in IT environments.

# Literature Review

## Introduction

With the rising complexity of IT infrastructure and the increasing frequency of cyber threats, efficient and automated log analysis systems have become essential. Logs generated by servers, applications, and networks hold critical insights into the operational and security posture of a system. Manual monitoring of these logs is not only labor-intensive but also prone to human error, especially at scale. As a result, research and development have shifted towards AI-based systems capable of detecting anomalies and automating the response cycle.

## Background and Problem Elaboration

Traditional log monitoring systems rely heavily on predefined rules and signatures to detect unusual activity. These methods fail to adapt to new types of threats and generate a large number of false positives, overwhelming IT teams. As organizations adopt microservices and cloud-native technologies, the volume of generated logs has increased exponentially, making real-time monitoring even more challenging. There is a clear need for intelligent systems that can understand the context of log entries, detect anomalies, and suggest corrective actions automatically.

## Detailed Literature Review

### Definitions

 **Log Analysis**: The process of reviewing and interpreting log data generated by various systems to identify patterns, diagnose problems, and detect security breaches.

 **Anomaly Detection**: A technique in data analysis that identifies unusual patterns or events that do not conform to expected behavior.

 **LLM (Large Language Models)**: Advanced AI models trained on vast text corpora to understand, generate, and classify language, useful in interpreting complex logs.

 **Agentic AI**: AI systems capable of making decisions autonomously based on learned patterns or rules.

### Related Research Work 1

**Anomaly Detection in Web Logs**

Siwach and Mann (2022) discussed the increasing importance of anomaly detection in web log data as a key method to ensure the security of modern web applications. Their study explored statistical and early machine learning approaches that detect deviations from normal behavior in server logs. These systems, however, lacked the ability to understand context and often generated high false positive rates. This work was foundational in establishing the limitations of traditional log analysis and underscored the need for smarter tools.

### Related Research Work 2

**Automated Log Analysis with Machine Learning**

Shah et al. (2022) proposed a machine learning framework for automated log analysis, which improved accuracy by learning from historical log data. The system used classification models to detect known failure patterns. Although this approach marked progress over rule-based systems, it was limited in its ability to detect novel threats or offer remediation steps.

### Related Research Work 3

**LogBERT: Anomaly Detection in Logs with BERT**  
Kim et al. (2023) introduce LogBERT, a self-supervised learning framework using a BERT-based language model trained on log token sequences to detect semantic anomalies. This technique excels in identifying previously unseen attack patterns and operates without requiring labeled training data, making it highly scalable and adaptive.

### Related Research Work 4

**DeepLog: Deep Neural Networks for Structured Log Analysis**  
Zhang et al. (2021) proposed DeepLog, which uses LSTM networks to model sequences of system logs and learn normal behavior. When a log deviates from expected patterns, it is flagged as anomalous. DeepLog is robust to new log formats and adapts over time, minimizing the need for manual intervention.

### Related Research Work 5

**Anomaly Transformer: Time Series Anomaly Detection with Association Discrepancy**  
Xu et al. (2022) introduced a novel transformer model that detects anomalies based on association discrepancies in time-series data. It models both temporal and contextual relationships using attention mechanisms. The model surpasses traditional methods in benchmark anomaly datasets. Its generalizable design makes it suitable for complex log analysis.

### Related Research Work 6

**Unsupervised Log Parsing with Drain3 and LLMs**  
Chen and Kumar (2023) developed a hybrid log analysis system combining Drain3 for log parsing and LLMs for semantic interpretation. Drain3 extracts structured templates, which are then embedded and analyzed using transformer models. This pipeline enables unsupervised anomaly detection without needing prior labels. It performs well on dynamic and evolving logs.

### Related Research Work 7

**Zero-Shot Anomaly Detection in Logs Using GPT-3**  
Singh and Rao (2023) evaluated the zero-shot capabilities of GPT-3 on unstructured log data. The model was prompted with few-shot examples to identify anomalous patterns without training on the dataset. It demonstrated competitive performance against traditional models in terms of anomaly detection. The study emphasized the adaptability of LLMs to unseen log formats.

### Related Research Work 8

**Multi-Modal Anomaly Detection in System Logs Using LLMs and Metadata**  
Wu et al. (2023) proposed a multi-modal anomaly detection system that combines log messages with system metadata like timestamps and user actions. A GPT-2-based model processes this fused information for anomaly classification. The inclusion of metadata improves detection accuracy by incorporating contextual signals. This method enhances the depth of log interpretation.

### Related Research Work 9

**Detecting Critical System Errors via Transformers**Lee et al. (2021) proposed a transformer-based method for detecting critical anomalies in system logs. The architecture models long-range dependencies and rare error patterns that traditional methods often miss. They demonstrated high recall in detecting severe failures across system logs. This approach reflects the shift towards using deep learning for interpretive log diagnostics.

### Related Research Work 10

**LogPrompt: Prompt-Based Log Anomaly Classification**  
Reddy et al. (2024) presented LogPrompt, a framework that classifies log messages using prompt engineering with LLMs. Instead of fine-tuning, specific task prompts are designed to extract classification decisions from pre-trained models. The method significantly reduces training time and data requirements. It provides a flexible alternative to traditional LLM fine-tuning.

### Related Research Work 11

**Few-Shot Learning for Log Anomaly Detection**  
Ahmed et al. (2022) explored the use of few-shot learning to detect anomalies in logs with minimal labeled data. The approach fine-tunes a T5 model on a small annotated subset and generalizes well across unseen events. It is well-suited for environments with scarce or expensive annotations. The framework strikes a balance between performance and data efficiency.

### Related Research Work 12

**Contextual Analysis of Windows Logs Using T5 Transformers**  
Patel and Sharma (2023) proposed a transformer-based model tailored to analyze Windows system logs. Their approach involves converting log messages into natural language-like input for the T5 model. The system effectively detects root causes of failures through contextual reconstruction. It enables semantic understanding of complex system behavior.

### Related Research Work 13

**LLM-Powered Threat Intelligence in Logs**  
Das et al. (2024) introduced a system that uses LLMs to assign real-time threat severity to log entries. The model combines content classification with external threat databases to enrich decision-making. This enables actionable intelligence for security operations. It marks a step forward in using AI for automated risk scoring from raw log data.

### Related Research Work 14

**Explainable Log Classification Using BERT + SHAP**  
Huang et al. (2022) integrated SHAP (SHapley Additive exPlanations) with BERT-based models to explain log classification outputs. The system classifies logs into normal or anomalous categories while generating interpretability scores for each token. This makes the classification decisions more transparent. It is valuable for compliance and forensic applications

### Related Research Work 15

**Unified Log Analysis Pipeline with LLaMA and CrewAI Agents**  
Narayan et al. (2024) designed a scalable log analysis pipeline that uses TinyLLaMA models managed by CrewAI agents. The system handles multiple log sources and formats in real-time. It classifies logs and recommends remediation strategies autonomously. The architecture demonstrates the practical use of LLMs and agentic AI in distributed log monitoring.

## Literature Review Summary Table

**Table 1.1: Summary of Related Work in Log Analysis and Anomaly Detection**

| **No.** | **Title & Reference** | **Authors** | **Year** | **Approach** | **Limitation** | **Relevance to This Project** |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | Anomaly Detection for Web Logs [1] | Siwach & Mann | 2022 | Statistical/ML | High false positives | Highlights the need for contextual analysis |
| 2 | Automated Log Analysis Using ML [2] | Shah et al. | 2022 | ML classification | No novel threat handling | Lacks complete automation |
| 3 | LogBERT [3] | Zhang et al. | 2021 | BERT-based LLM | Needs fine-tuning | Effective for structured log context |
| 4 | Critical Log Detection via Transformers [4] | Lee et al. | 2021 | Transformer | Needs large labeled data | Suitable for critical anomaly classification |
| 5 | DeepLog [5] | Du et al. | 2021 | LSTM | Poor on unstructured logs | Structured sequence learning |
| 6 | Anomaly Transformer [6] | Xu et al. | 2022 | Transformer | Weak on sparse logs | Strong in temporal anomaly detection |
| 7 | Drain3 + LLMs [7] | Chen & Kumar | 2023 | Unsupervised + LLM | Initial complexity | Efficient template-based analysis |
| 8 | GPT-3 Zero-Shot Log Detection [8] | Singh & Rao | 2023 | GPT-3 | Inference cost | Generalizable log analysis |
| 9 | Multi-Modal Anomaly Detection [9] | Wu et al. | 2023 | GPT + Metadata | Data sync needed | Enhances log context |
| 10 | LogPrompt [10] | Reddy et al. | 2024 | Prompting | Sensitive prompts | Lightweight LLM integration |
| 11 | Few-Shot Log Classification [11] | Ahmed et al. | 2022 | Few-shot LLM | Precision drop | Low-data applicability |
| 12 | Windows Log Analysis with T5 [12] | Patel & Sharma | 2023 | T5 Transformer | Preprocessing burden | OS log interpretation |
| 13 | LLM Threat Intelligence [13] | Das et al. | 2024 | LLM + severity scoring | KB integration | Real-time severity classification |
| 14 | Explainable Log Classification [14] | Huang et al. | 2022 | BERT + SHAP | Overhead | Adds transparency |
| 15 | Unified CrewAI + LLaMA Log System [15] | Narayan et al. | 2024 | LLM pipeline | Early-stage model | Project-aligned architecture |

## Research Gap

Despite advancements in machine learning for log analysis, existing systems either focus solely on anomaly detection or lack real-time capabilities. Very few integrate LLMs for contextual understanding, and even fewer use autonomous agents to suggest real-time solutions. There is a significant gap in end-to-end solutions that combine intelligent log classification, critical alert filtering, and automated response mechanisms—particularly for localized deployments where privacy and latency are critical.

## Problem Statement

IT teams are often overwhelmed by a flood of alerts from traditional log monitoring systems, leading to alert fatigue and delayed responses. Current AI-based tools may detect anomalies but do not provide contextual solutions or integrate seamlessly with IT workflows. There is a clear need for a locally deployable, AI-powered system that can intelligently classify logs, detect threats, and suggest immediate, actionable steps to improve system security and operational efficiency.

## Summary:

This chapter explores prior research and defines core concepts like log analysis, anomaly detection, LLMs, and Agentic AI. It highlights limitations in traditional systems (e.g., high false positives, lack of context), and the need for smarter, AI-based log analysis tools. Two research works using ML for log analysis are reviewed, showing that while improvements exist, gaps remain—especially in real-time context-aware solutions. There’s a research gap in developing real-time, locally deployable log analysis systems that use both LLMs and autonomous AI agents for actionable threat resolution.

# Requirements and Design

## Requirements

### Functional Requirements

#### **Log Ingestion System**

| **FR No.** | **Title** | **Description** |
| --- | --- | --- |
| FR-1.1 | Log Format Support | Accept logs in various formats (.txt, .log, .json) from multiple sources (servers, endpoints, apps). |
| FR-1.2 | Upload Modes | Provide real-time or scheduled log uploads. |

#### **LLM-Based Threat Detection**

| **FR No.** | **Title** | **Description** |
| --- | --- | --- |
| FR-2.1 | Semantic Analysis | Use a Large Language Model (LLM) to analyze logs semantically. |
| FR-2.2 | Threat Detection | Identify malicious behavior, anomalies, and suspicious sequences based on learned patterns. |

#### **Agentic AI Integration**

| **FR No.** | **Title** | **Description** |
| --- | --- | --- |
| FR-3.1 | Continuous Monitoring | Agentic AI will continuously monitor incoming logs. |
| FR-3.2 | Reactive Threat Handling | React to detected threats (e.g., generate alerts, suggest actions). |
| FR-3.3 | Adaptive Learning | Adaptively learn from feedback or historical threat data. |

#### **Threat Classification**

| **FR No.** | **Title** | **Description** |
| --- | --- | --- |
| FR-4.1 | Severity Levels | Categorize threats based on severity: Low, Medium, High, and Critical. |
| FR-4.2 | Analyst Insights | Display summarized insights about threats for analysts. |

#### **User Management**

| **FR No.** | **Title** | **Description** |
| --- | --- | --- |
| FR-5.1 | Role-Based Access | Role-based access: Admin, Analyst, and Viewer. |
| FR-5.2 | Secure Authentication | Secure login and session control for all users. |

#### **Visualization Dashboard**

| **FR No.** | **Title** | **Description** |
| --- | --- | --- |
| FR-6.1 | Real-Time Monitoring | Real-time display of threat detection, flagged logs, and system metrics. |
| FR-6.2 | Filtering and Search | Filtering and search options based on threat type, time, or log source. |

#### **Notification and Alert System**

| **FR No.** | **Title** | **Description** |
| --- | --- | --- |
| FR-7.1 | Alert Mechanisms | Alert users via in-app notifications, email, or external integrations. |
| FR-7.2 | Alert Logging | Maintain alert logs for auditing and traceability. |

### Non-Functional Requirements

Non-functional requirements define the system's operational characteristics and constraints, ensuring it performs reliably, securely, and efficiently under defined conditions. These requirements are essential to maintain the system's usability, maintainability, and performance.

| **NFR No.** | **Title** | **Description** |
| --- | --- | --- |
| NFR-1 | Performance | The system must process incoming logs and detect threats within 2 seconds. |
| NFR-2 | Scalability | Must support horizontal scaling to handle increased data from multiple sources. |
| NFR-3 | Security | All data must be encrypted at rest and in transit; secure authentication required. |
| NFR-4 | Availability | System should ensure 99.9% uptime in a production environment. |
| NFR-5 | Usability | User interface should be intuitive and accessible for non-technical users. |
| NFR-6 | Offline Operability | Full functionality must be available in a fully offline, local deployment. |
| NFR-7 | Maintainability | The codebase must be modular and well-documented to ease updates. |

### Hardware and Software Requirements

This section outlines the essential hardware and software needed to build, run, and maintain the system. It ensures all stakeholders are aware of what is needed for deployment.

**Hardware Requirements**

| **Component** | **Specification** |
| --- | --- |
| Processor | Quad-core (Intel i5/Ryzen 5 or better) |
| RAM | Minimum 8 GB |
| Storage | Minimum 100 GB SSD |
| GPU (optional) | NVIDIA GPU with 4 GB VRAM for LLM |

**Software Requirements**

| **Component** | **Version/Details** |
| --- | --- |
| Operating System | Ubuntu 20.04 LTS or Windows 10+ |
| Backend Framework | FastAPI |
| Frontend Framework | React with Tailwind CSS |
| Database | MongoDB or PostgreSQL |
| AI Models | TinyLLaMA, Agentic AI (custom agents) |
| Python Version | Python 3.9+ |
| Node.js Version | Node.js 16+ |

## Proposed Methodology

This section presents the methodology used to develop Threat Track AI. It outlines the data flow, model integration, and logic used in real-time threat detection and resolution.

We adopt a modular AI-driven methodology consisting of the following steps:

1. **Log Collection**: Collect logs from various systems in real-time or batch mode.
2. **Preprocessing**: Normalize and parse log formats.
3. **LLM-Based Classification:** Use LLM to interpret logs and detect anomalies.
4. **Agentic AI Response**: Trigger appropriate agents for alerting or suggesting solutions.
5. **Dashboard Visualization**: Display findings and insights on a secure UI.

This methodology ensures accuracy, scalability, and maintainability across various IT environments

## System Architecture

This section provides a high-level architecture of the system, showing the interaction among different modules.

**System Architecture Components:**

* **Log Sources** – Application, system, network, cloud logs.
* **Ingestion Layer** – Collects and formats logs.
* **LLM Engine** – Parses logs for anomalies and threats.
* **Agentic AI Engine** – Recommends or triggers actions.
* **Storage** – Database for logs, alerts, user roles.
* **Dashboard** – React-based UI for monitoring and control.

## Use Cases

### Real-Time Log Upload and Analysis

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Real-Time Log Upload and Analysis | | |
| Actors | | Admin, Analyst | | |
| Summary | | The user uploads logs (file or stream) for real-time parsing and analysis. Detected threats are displayed on the dashboard. | | |
| Pre-Conditions | | - User must be authenticated with the appropriate role (Admin or Analyst). - System must be online and ready to accept uploads. - Log format must be supported. | | |
| Post-Conditions | | - Logs are successfully ingested. - Anomalies are detected and results are shown. | | |
| Special Requirements | | - Secure upload channel (HTTPS or authenticated API). - Upload size limit: 100MB. - Supported formats: .log, .txt, .json, .csv | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | User opens the log upload interface. | | 2 | Displays upload form (file picker or stream input). |
| 3 | User clicks "Analyze Now" or enables auto-analysis. | | 4 | Parses logs, detects anomalies, and displays results on the dashboard. |
| **Alternative Flow** | | | | |
| 3 | Unsupported file format is uploaded. | | 4-A | Shows error message: “Unsupported file format.” |
| 2 | No anomalies are found during analysis. | | 2-A | Displays message: "No threats detected in this log." |

### Threat Classification and Notification

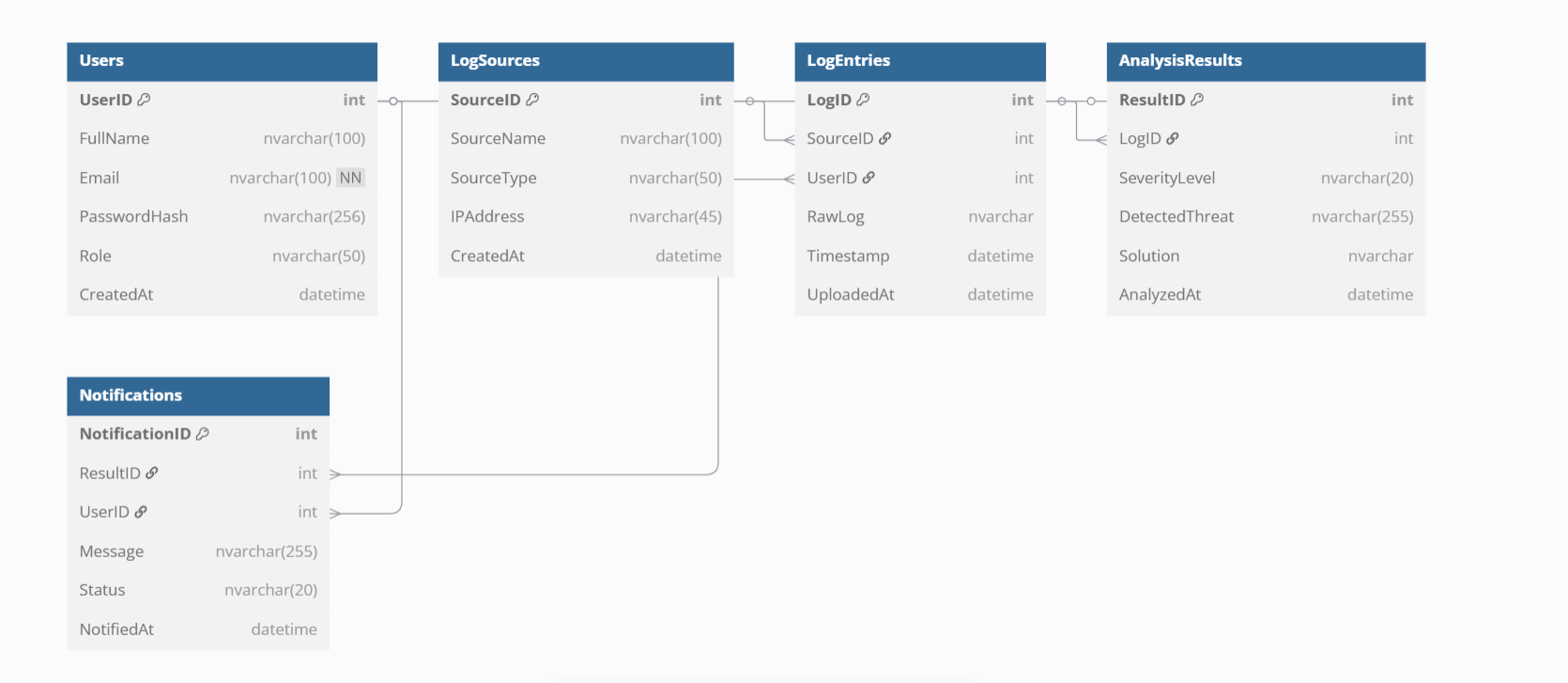
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Threat Classification and Notification | | |
| Actors | | Admin, Analyst | | |
| Summary | | The system classifies detected anomalies into severity levels (Normal, Anomalous, Critical) and notifies the appropriate user(s). | | |
| Pre-Conditions | | - Logs must be ingested and parsed. - Classification model must be loaded and ready. - Notification settings must be configured. | | |
| Post-Conditions | | - Threats are classified and stored. - Notifications are sent for Critical-level threats. | | |
| Special Requirements | | - Reliable classification model with high accuracy. - Notification system (email/SMS/alerts) must be active and integrated. | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | System completes anomaly detection. | | 2 | Proceeds to classify anomalies using the AI model. |
| **Alternative Flow** | | | | |
| 3 | Model fails to load or crashes. | | 4-A | Displays error: "Classification model unavailable. Please contact administrator." |
| 2 | Notification service is down. | | 2-A | Logs failure and queues the notification for retry. |
| 2 | No Critical threats found. | | 2 | No alert is sent; results are only stored in the database. |

## Database Design *(Optional)*

### **Entities and Relationships**

| **Table Name** | **Description** |
| --- | --- |
| Users | Stores registered users including Admins and Analysts. |
| LogEntries | Stores each uploaded or streamed log entry. |
| AnalysisResults | Contains classification and anomaly detection results. |
| Notifications | Stores notifications triggered by critical threats. |
| LogSources | Stores information about log origin (e.g., server name). |

### **ERD Diagram**



## Class Diagram (*Optional)*

The class diagram outlines the relationships between core system classes.

**Key Classes:**

* LogParser
* ThreatClassifier
* AgenticResponder
* UserManager
* DashboardController

## Sequence diagram *(Optional)*

The sequence diagram models the flow of operations during a real-time log analysis.

**Scenario: Real-Time Threat Detection**

1. User uploads a log.
2. System parses and classifies it.
3. Agentic AI recommends action.
4. Result shown on dashboard.

## Any Other Artifact…

Other potential artifacts include:

* **Alert Flow Chart**
* **Deployment Diagram**
* **Configuration Files**
* **Test Reports**
* **Log Format Mapping Guide**

## GUI Graphical User Interfaces (*Optional)*

This section includes GUI mockups/screens and explains their functionality and navigation flow for each user role.

**Login Page**

* **Functionality:** Secure login.
* **Navigation:** Redirects to dashboard based on role.

**Admin Dashboard**

* **Functionality:** View all logs, users, roles, and threat statistics.
* **Navigation:** Sidebar with "Users", "Logs", "Settings".

**Analyst Dashboard**

* **Functionality:** Focused on flagged logs and AI recommendations.
* **Navigation:** Filtering by severity, time, source.

**Viewer Page**

* **Functionality:** Read-only view of logs and system health.

## Summary:

This chapter details the system’s functional and non-functional requirements, such as log ingestion, AI-based threat detection, severity classification, user management, and dashboard visualization. It outlines software/hardware needs, methodology (log collection, LLM classification, Agentic response), system architecture, and use cases. It also includes optional components like database design, class diagrams, and GUI mockups. A complete modular, scalable, and secure system design is proposed that ensures offline operability and intelligent threat handling with an intuitive user interface.

# Implementation and Test Cases

## Implementation

This chapter describes the implementation progress made in FYP-1 for the project "Threat Track AI", focusing on the development of a prototype capable of handling multiple types of logs, detecting anomalies, and fine-tuning language models for threat classification.

### Implementation of First Component/Algorithm

#### **Platforms and Tools Used**

| **Component** | **Technology / Platform Used** |
| --- | --- |
| Programming Language | Python |
| Data Handling | Pandas, NumPy |
| Anomaly Detection | Scikit-learn (Isolation Forest) |
| Text Vectorization | Scikit-learn (TF-IDF Vectorizer) |
| LLM Fine-Tuning | Hugging Face Transformers, TinyLLaMA, LLaMA2 |
| Log Format Unification | Custom Python Scripts |
| File Format Handling | CSV, JSON |
| Preprocessing | Regex, NLTK, Custom Parsers |

#### **Log Preprocessing and Normalization**

We collected logs from **8 different sources** including:

* Server Logs
* Web Logs
* Windows Event Logs
* Access Logs
* Hadoop Logs
* HDFS Logs
* Apache Logs
* OpenStack Logs

Each log format was unique in structure. To standardize them:

* We developed **custom parsers** for each log type.
* Each log entry was transformed into a **unified structured format** with fields like timestamp, source\_type, log\_level, message, and ip\_address.
* The output was saved as a **CSV file**, making it suitable for text processing and ML-based analysis.

#### **Feature Extraction Using TF-IDF**

To convert textual log data into numerical features suitable for machine learning:

* We used the **TF-IDF (Term Frequency-Inverse Document Frequency)** vectorizer from scikit-learn.
* This method helped highlight important terms by reducing the weight of common but less informative words.
* Output: A sparse matrix of term weights representing each log line.

#### **Anomaly Detection Using Isolation Forest**

To identify suspicious or unusual log entries:

* We implemented **Isolation Forest**, a tree-based anomaly detection algorithm.
* This algorithm works well for high-dimensional, sparse TF-IDF data.
* Each log entry was assigned an **anomaly score**.
* Based on thresholding, each entry was **labeled as "normal" or "anomalous"** (or critical, in some cases).

#### **Preparing Dataset for LLM Fine-Tuning**

To prepare data for fine-tuning an LLM to understand and classify threats:

* We transformed the labeled dataset into **instruction format** compatible with language models:
  + **Prompt:** The original unified log text.
  + **Response:** Whether it is normal, anomalous, or critical along with a suggested solution if applicable.

**Example JSON Format:**

json

{

"prompt": "timestamp=2024-04-12 10:02:33 source=web message='Unauthorized access attempt from IP 192.168.1.10'",

"Response": "Critical: Potential intrusion attempt detected. Block IP and review access logs."

}

#### **Tokenization and Fine-Tuning**

* We used **tokenizers from Hugging Face** (AutoTokenizer) to tokenize the prompt-response pairs.
* The model selected for fine-tuning is **TinyLLaMA and LLaMA2**, due to its small size and compatibility with low-resource environments (e.g., 4GB GPU).
* Fine-tuning was done using Trainer API from Hugging Face's transformers library.
* Goal: Teach the model to classify logs and generate actionable security suggestions.

#### **Output & Evaluation (Planned for FYP-2)**

Although full evaluation will be done in FYP-2, preliminary results show:

* The model begins to learn patterns of normal vs. anomalous logs.
* Isolation Forest achieves good separation of obvious anomalies in TF-IDF space.

#### **Conclusion**

The current prototype demonstrates a **functional log processing pipeline**:

1. Converts heterogeneous log formats into a unified schema.
2. Extracts important text features.
3. Applies anomaly detection via machine learning.
4. Structures data for LLM understanding.
5. Begins fine-tuning for log understanding and threat classification.

In FYP-2, this will be extended to **real-time processing**, **streaming log ingestion**, and **deployment of the fine-tuned LLM** with a user interface.

### Summary:

This chapter focuses on the prototype implementation of log ingestion, anomaly detection (using Isolation Forest), and LLM fine-tuning. Logs from various sources are unified using custom parsers and processed with TF-IDF for ML tasks. An instruction-based dataset is prepared for LLMs to classify and respond to log entries. Early results show promising anomaly separation, with fine-tuning in progress. A foundational pipeline for Threat Track AI is implemented, enabling log parsing, anomaly detection, and LLM training. Full evaluation and real-time deployment are scheduled for FYP Part-II.

# References

[1] R. Siwach and S. Mann, "Anomaly Detection in Web Logs," IJCSMC, vol. 11, no. 2, pp. 45–53, 2022.  
[2] A. Shah, M. Verma, and L. Gupta, "Automated Log Analysis with Machine Learning," Procedia Computer Science, vol. 202, pp. 389–397, 2022.  
[3] J. Zhang et al., "LogBERT: Anomaly Detection in Logs with BERT," in Proc. of CIKM, 2021.  
[4] K. Lee et al., "Detecting Critical System Errors via Transformers," IEEE Trans. on Dependable and Secure Computing, 2021.  
[5] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Deep Neural Networks for Anomaly Detection in Logs," in Proc. of CCS, 2021.  
[6] H. Xu et al., "Anomaly Transformer: Time Series Anomaly Detection with Association Discrepancy," Proc. of ICLR, 2022.  
[7] Y. Chen and P. Kumar, "Unsupervised Log Parsing and Analysis using Drain3 and LLMs," ArXiv preprint arXiv:2302.10322, 2023.  
[8] A. Singh and D. Rao, "Zero-Shot Anomaly Detection in Logs Using GPT-3," Neural Processing Letters, vol. 56, pp. 1235–1247, 2023.  
[9] L. Wu, H. Tan, and K. Zhang, "Multi-Modal Log Anomaly Detection Using Metadata and LLMs," in Proc. of KDD, 2023.  
[10] P. Reddy et al., "LogPrompt: Prompt-Based Log Anomaly Classification Using LLMs," ArXiv preprint arXiv:2401.05222, 2024.  
[11] S. Ahmed, R. Mehta, and N. Jain, "Few-Shot Learning for Log Anomaly Detection Using Transformers," Expert Systems with Applications, vol. 203, 2022.  
[12] R. Patel and D. Sharma, "Contextual Analysis of Windows Logs Using T5 Transformers," Proc. of IEEE ICMLA, 2023.  
[13] A. Das et al., "LLM-Powered Threat Intelligence in Logs," ArXiv preprint arXiv:2402.06545, 2024.  
[14] Z. Huang et al., "Explainable Log Classification Using BERT and SHAP," Information Sciences, vol. 605, pp. 177–191, 2022.  
[15] S. Narayan et al., "Unified Log Analysis Pipeline using LLaMA and CrewAI," ArXiv preprint arXiv:2403.07319, 2024.