 **Air University Islamabad**

**FACULTY OF COMPUTING & ARTIFICAL INTELLIGENCE**

**Department of Creative Technologies**

**LOG MANAGMENT SYSTEM-REPORT**

Submitted To: Sir Shakeel Ahmed

Submitted By: Malik Usama Arif (233105)

Ehtisham Ul Haq (240011)

Fiza Noor (240459)

Submission Date: 9th June, 2025

Contents

[1. Introduction 3](#_Toc200449946)

[2. Objectives 3](#_Toc200449947)

[3. System Architecture 3](#_Toc200449948)

[3.1 Components Overview 3](#_Toc200449949)

[4. Key Features and Data Structures 4](#_Toc200449950)

[4.1 Real-Time Monitoring 4](#_Toc200449951)

[4.2 Custom Priority Queue 4](#_Toc200449952)

[4.3 Log Relationship Graph 4](#_Toc200449953)

[4.4 Bloom Filter 4](#_Toc200449954)

[5. Interactive Analysis Features 4](#_Toc200449955)

[6. Multithreading and Synchronization 4](#_Toc200449956)

[7. Log File Format 5](#_Toc200449957)

[8.UML: 5](#_Toc200449958)

[8.1 Class Diagram: 5](#_Toc200449959)

[8.2 Sequence Diagram: 6](#_Toc200449960)

[9. Comparison & Optimization 7](#_Toc200449961)

[9.1 Time and Space Complexity Evaluation of Data Structures 7](#_Toc200449962)

[9.2 Time and Space Complexity Evaluation of Sorting and Searching Algorithm 8](#_Toc200449963)

[10. Limitations and Future Work 9](#_Toc200449964)

[11. Conclusion 9](#_Toc200449965)

[12. Member-wise Percentage contribution: 9](#_Toc200449966)

# 1. Introduction

In modern software environments, especially in systems with multiple processes or services, efficient logging and monitoring are essential. This project presents a C++ Log Management System with advanced features including:

* Real-time system call monitoring using strace
* Custom priority queue for log severity management
* Bloom filter for quick log type checks
* Graph-based relationship analysis between logs
* Interactive log analyzer with timeline, statistics, and error tracing

# 2. Objectives

* Implement a real-time process log monitoring system.
* Build custom data structures to manage and analyze logs efficiently.
* Provide a command-line interface for log analysis and interaction.
* Enhance performance using multithreading and thread-safe data structures.

# 3. System Architecture

## 3.1 Components Overview

* LogManager: Core controller for starting, stopping, and analyzing logs.
* PriorityLogQueue: Custom min-heap for priority-based log sorting.
* LogRelationshipGraph: Graph structure to identify relations between log types.
* LogTypeFilter: Bloom filter to optimize log type lookups.
* Monitor Process: Spawns threads to run strace on target PIDs and logs system calls in JSON format.

# 4. Key Features and Data Structures

## 4.1 Real-Time Monitoring

Each target process is monitored using strace, and the log output is captured and written to a .log file in JSON format.

## 4.2 Custom Priority Queue

Used to sort logs based on weight (importance or frequency):

## 4.3 Log Relationship Graph

Stores weighted connections between different types of logs (e.g., program\_name → system\_call):

## 4.4 Bloom Filter

Lightweight mechanism to quickly check if a certain log type has been previously encountered:

# 5. Interactive Analysis Features

The system supports real-time interaction through commands like:

| Command | Functionality |
| --- | --- |
| !!errors | Shows all log entries indicating errors |
| !!stats | Displays count of each log type |
| !!timeline | Orders logs chronologically |
| !!search <query> | Searches logs for specific keywords |
| !!relationships | Prints top N log type relationships |
| exit | Exits analysis mode |

# 6. Multithreading and Synchronization

* Threads are used for non-blocking process monitoring.
* mutex is used to ensure thread safety in all shared data structures.
* Each LogProcess owns a monitoring thread that tracks a PID independently**.**

# 7. Log File Format

All logs are stored in JSON format with the following structure:

{ "pid": 1234,

"program": "bash",

"timestamp": 1718006400,

"entry": "write(1, ...)",

"type": "system\_call" }

# 8.UML:

## 8.1 Class Diagram:

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

## 8.2 Sequence Diagram:

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

# 9. Comparison & Optimization

A graph with blue and pink squares

AI-generated content may be incorrect.

## 9.1 Time and Space Complexity Evaluation of Data Structures

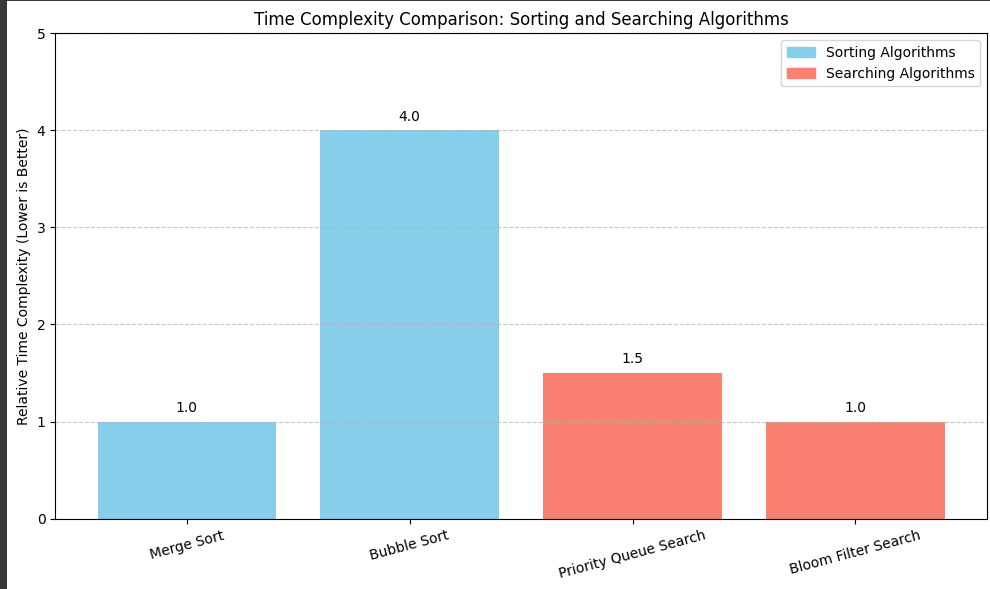
The chart provided presents a comparison of three key data structures used in our Log Management System:

| **Data Structure** | **Time Complexity (Insert/Search)** | **Space Complexity** |
| --- | --- | --- |
| PriorityLogQueue (Min-Heap) | O(log n) / O(1) | O(n) |
| Bloom Filter | O(k) (constant) | O(n \* k) |
| Graph (Adjacency List) | O(1) / O(k) | O(V + E) |

**PriorityLogQueue (Min-Heap):** Enables prioritized log processing with O(log n) insertions and linear space usage.

**Bloom Filter:** Offers fast O(1) membership checks with increased space due to multiple hash functions.

**Graph (Adjacency List):** Models log relationships with linear time lookups and space growing with nodes and edges.



## 9.2 Time and Space Complexity Evaluation of Sorting and Searching Algorithm

| **Algorithm** | **Type** | **Time Complexity** | **Best Use Case** | **Trade-offs** |
| --- | --- | --- | --- | --- |
| Merge Sort | Comparison-based | O(n log n) | Sorting structured log entries | More memory usage (auxiliary arrays) |
| Bubble Sort (baseline) | Comparison-based | O(n²) | Educational comparison | Inefficient for large datasets |
| Custom Priority Search | Heap-based | O(log n) insertion/search | Fetching highest priority logs | Needs custom comparator and heap maintenance |
| Bloom Filter Search | Hash-based | O(1) | Checking if a log has been seen before | Possible false positives, not suitable for deletion |

# 10. Limitations and Future Work

Limitations

* Relies on Unix-based systems due to strace.
* JSON logs can become large without rotation mechanisms.

Future Enhancements

* Add log rotation and archiving support.
* Implement GUI using Qt or ImGui for visualization.
* Integrate severity levels for each log type.
* Store graphs and relationships in persistent databases.

# 11. Conclusion

This Log Management System provides an extensible and efficient tool for real-time log collection and analysis. The custom data structures ensure performance, and the command-line interface allows deep exploration of system events. It bridges system-level tracing with modern C++ programming and concurrent processing techniques.

# 12. Member-wise Percentage contribution:

|  |  |
| --- | --- |
| Name: | Approx Percentage: |
| Malik Usama Arif | 35% |
| Ehtisham ulhaq | 35% |
| Fizza Noor | 30% |