FILE STATISTICS & LOG ANALYZER SYSTEM

Author: Yahiwenisi Filmon (LQ7886)
Institution: HiLCoE School of Computer Science & Technology
Course: Programming II (c++)
Date: Sunday - February 1, 2026

BASELINE PROJECT PLAN

Introduction

A. Project Overview

As system environments become more complex, the need for efficient disk space management and log analysis has become critical. The "File Statistics & Log Analyzer" is a local utility designed to recursively scan directory structures and parse system logs to provide insights into storage distribution and system health. The project aims to replace slow, manual checks with an automated, high-performance C++ tool.

B. Background

System administrators often face challenges when identifying "disk space hogs" or detecting error patterns in large log files (Apache/Nginx). Manual searching is time-consuming and prone to human error. Deeper issues include:

- **Storage Invisibility:** Difficulty in identifying which file types or directories consume the most space.
- **Log Complexity:** Parsing thousands of lines of logs to find status codes or specific IP patterns is inefficient.
- **Performance Gaps:** Existing general-purpose tools can be bloated or slow for quick local diagnostics.

1. System Description

A. Alternatives

1. Use standard Unix tools (`du`, `grep`, `awk`) which require expert command-line knowledge.
2. Deploy complex enterprise solutions (Splunk, ELK) which are overkill for local diagnostics.
3. **Implement a dedicated C++ analyzer:** A lightweight, high-performance tool tailored for local FS and Log analysis.

B. System Description

The File Statistics & Log Analyzer will be a command-line application focusing on deep directory inspection and structured log parsing. Key features include:

- **File System Analysis:** Recursive scanning with metadata collection (size, extension, modification dates).
- **Log Parsing:** Automated detection of Apache, Nginx, and JSON log formats.
- **Aggregation Logic:** Histogram generation for size distribution and error rate calculations for logs.
- **Dual Reporting:** High-quality Text summaries for users and JSON output for automation.

2. Feasibility Assessment

A. Economic Analysis

I. Project Benefit

1. **Tangible Benefits:**
    - **Time Efficiency:** Saves ~80% of time spent on manual log audits.
    - **Resource Optimization:** Identifies redundant large files, freeing up storage and reducing hardware costs.
2. **Intangible Benefits:** Improved system reliability through early error detection, and better data-driven decisions regarding storage upgrades.

II. Project Cost

1. **One-time Costs:** Development labor, testing environments, and standard library integration. Estimated at $15,000 in labor value.
2. **Recurring Costs:** Minimal, primarily consisting of codebase maintenance and updates for new log formats.

B. Technical Analysis

Utilizes C++20 for performance and `std::filesystem` for reliable I/O. The system is designed to handle directories with millions of entries using efficient heap-based sorting for Top-N lists.

C. Operational Analysis

Integrates directly into the developer/admin workflow. Replaces manual `grep` commands with a single execution that produces structured reports.

D. Legal & Contractual Analysis

Adheres to MIT/GPL licensing for open-source components. Ensures no unauthorized data is transmitted from the local machine (privacy by design).

3. Schedules, Timeline & Resource Analysis

A. Schedule & Timeline (January 28, 2026 - February 2, 2026)

Phase	Start Date	End Date	Key Tasks
1. Project Initiation	Jan 28, 2026	Jan 28, 2026	Establish goals, define scope, identify core analyzer needs.
2. Project Planning	Jan 28, 2026	Jan 28, 2026	Feasibility study (Performance vs Ease of Use).
3. System Analysis	Jan 29, 2026	Jan 29, 2026	Gather requirements for Apache/JSON log formats and FS metadata.
4. System Design	Jan 29, 2026	Jan 29, 2026	Design class architecture, ReportGenerator interface, and data models.
5. Development	Jan 30, 2026	Jan 30, 2026	Code FileSystem and Log analyzers, integrate `nlohmann/json`.
6. Testing	Jan 31, 2026	Jan 31, 2026	Unit testing, system validation with large data sets, regex benchmarking.
7. Deployment	Feb 1, 2026	Feb 1, 2026	Compile executable, post-deployment assessment on target environment.
8. Documentation	Feb 1, 2026	Feb 1, 2026	Compile README and Amharic documentation.
9. Project Closure	Feb 2, 2026	Feb 2, 2026	Release resources, archive code, compile developer feedback.

B. Resource Analysis

Category	Resources Required	Description
Hardware	Developer Workstation	High-performance machine for compilation and testing.
Software	Clang/GCC Compiler	Support for C++20/23 standards.
Libraries	nlohmann/json	Single-header library for JSON reporting.
People	Yahiwenisi Filmon	Sole developer and system analyst.
Environment	Testing Data	100GB+ sample directories and multi-format log files.

4. Management Issues

A. Team Configuration & Management

As the sole project lead, **Yahiwenisi Filmon** will fulfill the following roles:

- **Lead Architect:** Technical direction and performance optimization.
- **Core Developer:** Writing the C++ implementation.
- **QA Lead:** Performance benchmarking and data validation.

5. Other Project-Specific Tasks

- **Regex Optimization:** Ensuring log parsing doesn't bottleneck on large files.
- **JSON Schema Validation:** Ensuring the output is strictly valid for downstream tools.