PROJECT ABSTRACT

Log File Analyzer with Visual Dashboard

A web-based log analysis tool: upload Apache/Nginx/application logs, get parsed structured records, error-rate trends, status-code splits, slow-endpoint tables and suspicious-IP detection on visual dashboards. Delivered built-to-order with the full web application, parser engine, report, PPT and viva kit.

01. Title

Log File Analyzer with Visual Dashboard.

02. Introduction / Background

Every web server and application writes logs, yet most small teams read them only during outages — grepping through millions of lines under pressure. Log analysis turns raw text into operational insight: which endpoints are slow, when errors spiked, which IPs are attacking, and what the single most frequent exception is. Enterprise tools for this are expensive and complex. This project builds a focused, visual log analyzer: paste or upload a log file and get structured parsing plus dashboards in seconds.

03. Problem Statement

Developers debugging production issues face unstructured text: timestamps in one format, status codes buried in the line, and no aggregation — finding “when did checkout start failing” means manual scanning. Existing open tools need infrastructure setup; SaaS tools cost per GB. The project delivers a self-contained analyzer: format-aware parsing (Nginx/Apache combined, common app formats), structured storage of each record, and a dashboard layer computing error trends, latency percentiles, endpoint tables and anomaly flags — no infrastructure to run.

04. Objectives

- Implement format-aware log parsing with regex-based grammars for common formats.
- Store parsed records with indexed fields: timestamp, IP, method, path, status, latency.
- Compute dashboards: requests over time, error rate, status-code split, p50/p95 latency.
- Detect slow endpoints and burst/scan patterns in client IPs.
- Group recurring application exceptions with first/last-seen tracking.

05. Proposed Methodology

The analyzer is built as a parse-then-query pipeline. (1) Ingestion: file upload or paste; format auto-detection tries known grammars and falls back to a custom-regex builder. (2) Parsing: streaming line parser extracts fields into structured records; malformed lines are counted and quarantined, not dropped silently. (3) Aggregation engine: time-bucketed counts, status-code histograms, per-endpoint latency percentiles, and per-IP rates computed in a single pass. (4) Presentation: dashboard views with charts and drill-down from an aggregate (e.g. the 09:00 error spike) to the underlying raw lines.

06. System Architecture / Working

Uploaded logs stream through the parser, which tags each line with its grammar and extracts fields into a record store. The aggregation engine then builds the dashboard dataset: per-hour request/error series, status-code distribution, endpoint latency tables (avg, p95, error %), and IP behavior profiles (request rate, 404 ratio). Heuristics flag anomalies — error-rate jumps vs a rolling baseline, IPs with scan-like 404 patterns, and endpoints whose p95 exceeds thresholds. Application stack traces are fingerprinted (normalized exception + top frame) so 412 occurrences of one root cause collapse into a single actionable card with suggested remediation.

07. Software Requirements

- Backend: Python (Flask/FastAPI) — parser and aggregation engine
- Frontend: HTML5, CSS3, JavaScript with canvas-based charts
- Storage: SQLite/PostgreSQL for parsed records; streaming parse for large files
- Runs on any laptop: no external services required
- Sample log generators included for demonstration and testing

08. Expected Outcomes

A working analyzer that parses a real Nginx log into dashboards: error trends, status splits, slow endpoints and flagged IPs, plus grouped exception cards. The single-pass aggregation design and the anomaly heuristics are the technical core for the viva.

09. Applications

- Small teams without a centralized logging platform
- Freelancers debugging client production issues
- Teaching aid for web-operations and DevOps courses

10. Future Scope

- Live tail mode with streaming dashboards
- Alert rules (error-rate threshold → webhook/Slack)
- Correlation of access logs with application logs by request ID