HW08 — Web Log Evaluation (Final Report)

Introduction

We analyzed web-server access logs to understand content demand, traffic patterns, and error hotspots. Haitham parsed the raw logs and produced structured CSV/JSON datasets; Yassin generated the diagrams; this report Haitham summarizes key insights and documents the workflow and reproducibility steps. Overall, traffic concentrates around a small set of pages, peaks occur during specific hours, and a minority of requests trigger 4xx/5xx errors that should be investigated.

Data & Context

Log source & format. The inputs are HTTP access logs in (or compatible with) the **Apache Combined Log Format**, which records client IP, identity, timestamp, request line (method + path + protocol), status, bytes, referrer, and user-agent. This format is widely documented and used as a default in many setups.

Pre-processing assumptions.

- Timestamps are parsed to a standard `DATETIME` (server's local TZ unless specified).
- Paths can be normalized by removing query strings when computing "page popularity."
- Obvious malformed lines are skipped and counted.

Method (Team Workflow)

3.1 Haitham — Parsing Script (Inputs → Structured Data)

Implemented the parsing script \{parse_logs.php} in the directory {/public_html/hw08/}.
The script reads {access.log} and {error.log} and:
- Parses each line into fields such as timestamp, HTTP method, path, status code, IP address, and user-agent.
- Normalizes the requested path by stripping any query string when computing page popularity.
- Converts Apache timestamps into a standard {YYYY-MM-DD HH:MM:SS} format.
- Detects a simple browser family (Chrome, Firefox, Safari, etc.) from the user-agent string.

The script then writes the following CSV files in {/public_html/hw08/}:
- {access_clean.csv} – one row per request.
- {page_counts.csv} – total hits per page.
- {browser_counts.csv} – total hits per browser family.
- {error_clean.csv} – structured entries from the error log.

3.2 Yassin — Diagrams (From CSV/SQL → Figures)

Where: `/public_html/hw08`
Expected figures in the diagrams file (`diagrams_report_clean.pdf`):

- `access_frequency`- Top pages by hits
- `timeline_access` - Requests per hour/day
- `timeline_errors` - Error (4xx/5xx) counts over time
- `browser_stats` - User-agent/browser (optionally “bot” vs “non-bot”)

3.3 Fatima — This Report (Synthesis & Clarity)

- Insert Yassine’s figures with captions and insights.
- Briefly describe Haitham’s script and the data flow.
- Ensure professional formatting and reproducibility.

Results

- *Figure 1 – Top pages by hits.*
  The diagram of page hits shows that a small number of core pages (such as the landing page, search pages, and maintenance-related pages) receive most of the requests. Less frequently used pages form a long tail, which is typical for web applications where users focus on a few main workflows.
- *Figure 2 – Traffic timeline.*
  The traffic timeline reveals that requests are clustered in a few time windows, which most likely correspond to development and testing sessions by the project team. Outside of these periods, the server activity is low or zero, reflecting that the site is not publicly advertised.
- *Figure 3 – Error timeline.*
  The error timeline indicates occasional spikes in errors, most of which are expected during development (e.g., missing resources, testing broken links, or form submissions with incomplete data). There are no signs of continuous high error rates, which suggests that the system behaves reasonably under the observed load.
- *Figure 4 – Browser / user-agent distribution.*
  The browser distribution diagram shows that the service was mainly accessed from a small set of modern browsers. This matches the expectation that the traffic originates from a few student machines rather than a wide variety of devices.

Discussion

Key takeaways.

- A small set of core pages drives most of the observed traffic. These pages are the primary candidates for further optimization and testing.
- The traffic pattern is clearly linked to development and testing sessions rather than continuous public usage, which is consistent with a student project.
- Errors appear in short bursts and seem related to debugging and experimenting with forms and URLs rather than persistent production problems.

Limitations.

- The logs mainly contain traffic generated by the project team, not real end-users, so the results cannot be interpreted as representative usage statistics.
- Standard access logs do not include response times or detailed performance metrics; deeper performance analysis would require additional instrumentation.
- User-agent based classification of browsers and bots is heuristic and may misclassify some requests.

Files and Reproducibility

The log parsing script is stored in the project repository under {hw08/parse_logs.php}. The cleaned CSV files and diagrams can be regenerated by copying the current Apache logs into \{public_html/hw08/} and re-running the script, then recreating the charts from the CSV outputs.

Conclusion

The team converted raw HTTP logs into structured data and visual summaries that highlight popular content, traffic rhythms, and error hotspots. The main next steps would be to fix recurring errors (especially 4xx codes), keep an eye on peak testing times when making changes, and, if the project were deployed more broadly, separate automated traffic from real users for cleaner analysis.