Log File Analysis

Author & TA: Kritin Gupta

1. Introduction

This project encompasses bash, html/css/javascript and python.

Log files are essential for tracking system performance, debugging errors, and ensuring security. However, analyzing raw log files manually can be time-consuming and inefficient.

This project aims to develop a Flask-based web application that allows users to:

- Upload log files and convert them into structured CSV format.
- Filter and sort logs to extract meaningful insights.
- Generate visualizations to interpret log data effectively.
- Download processed logs and visualizations for further analysis.

To ensure efficiency, log file parsing, conversion to CSV, and filtering must be handled using Bash, while all other functionalities may be implemented in Python or any other suitable technology.

2. Mandatory Features

1. Web Interface with Three Pages

The application must have at least three distinct pages:

1. Log Upload Page

- Users can upload log files (.log). See link under resources.
- The system automatically detects log format, parses logs and converts them into a structured CSV format using Bash. See link under resources.
- If the log format is incorrect, display an error message and do not proceed.

2. Log Display Page

- Displays the structured logs in a tabular format on the webpage. See link under resources.
- Provides an option to download the entire log file as a CSV file.

3. Graphs & Plots Page

- Generate some predefined basic plots based on the uploaded log data and display them.
- Allows users to select a specific range of data based on filters before plotting.

- Users can download plots as images (PNG, JPEG, etc.).

2. File Upload & Automatic Processing

Users can upload log files. A bash script will validate the log file for format and give an error message if it is incorrect and proceed only if it is correct. Bash script will also be used for extracting data in a structured format, and converting to CSV to display.

Bash scripts handle:

- Parsing and extracting structured data from logs.
- Converting structured data into CSV format.

3. Predefined Visualizations

You must support Apache Logs (see link under resources, specifically structured csv) . The supported log file type must have at least three predefined visualizations. We have provided the mandatory plots for Apache log files below.

Log File	Mandatory Plots
Apache Logs	Events logged with time (Line Plot) Level State Distribution (Pie Chart) Event Code Distribution (Bar Plot)

By default the visualization should show the three plots for the entire log. But, users should be given an option to filter the logs based on a selected from and to dates (e.g. from Nov 23 22:29:28 2005 to Jan 18 02:06:40 2006). In this case, the plots have to show the data for this duration only.

- In Events with time plot, determine the number of events that are in a second over time and plot this number vs time.
- In Level state distribution, plot a pie chart that shows the percentage of each level (notice and error)
- In Event Code distribution, plot a bar chart that captures the number of occurrences of events E1-E6.

4. Downloading & Exporting Features

By default download should be over the entire log. But, users should be given an option to filter the logs based on a selected from and to dates (e.g. from Nov 23 22:29:28 2005 to Jan 18 02:06:40 2006). In this case, download files should only contain relevant duration.

- Users can download log data (after processing to the structured format) as a CSV file.
- Users can download generated plots in PNG or JPEG format.

3. Constraints

- Bash scripts must handle log file parsing, CSV conversion, and filtering. Bash files may use sed, awk etc.
- All other functionalities (visualizations, web UI, additional processing) can be implemented in Python.
- You must generate CSV using bash, getting data for plotting from CSV can be gathered using python.
- You can use matplotlib, numpy, flask, os, subprocess modules for basic task completion. Using any other module must be approved by the TA.

4. Optional Features (Customizations)

Students are encouraged to implement additional features to enhance their project. Below are some suggestions, but you are free to come up and follow your ideas.

1. Log Display page

- You can provide sort options for the table. E.g. clicking on eventid or level will sort based on that.
- You can also provide a filter, where if you chose “notice” under level or E1 under eventid, it will only show those.

2. Embedded Python Code Editor for Custom Analysis

- Users can write and execute custom Python scripts to generate their own plots.
- Provide a textbox/embedded code editor for users to write code; and display the generated plots

- You can load the log file's csv as a pandas dataframe and provide that as a variable in the code
- User can write some script to generate the plot they want and you must run that in the background and display the plot that was generated
- You can provide a note / have some pre-written lines of the code that tells the user what modules/variables they can use

3. Extend the analysis to other log formats present in the website like below.

Log File Type	Plots
System Logs (Syslog)	Event Count Over Time (Line Plot) Severity Breakdown (Pie Chart) Top Log Sources (Bar Chart)
Android Log	Event Component Distribution (Bar Chart) Traffic trends (Line Plot) Level Breakdown (Pie Chart)

4. Interactive Dashboard

- A drag-and-drop interface for uploading logfiles.
- A dropdown for selecting logfiles/plots.
- An interactive and intuitive user interface for selecting filters/ranges

5. Resources

- https://github.com/logpai/loghub: You can find a collection of logfiles here.
- Apache logs specifically are at https://github.com/logpai/loghub/tree/master/Apache. This shows the logs, structured csv as well as template for mapping events. If an event cannot be mapped to E1-E6, leave it blank.
- https://flask.palletsprojects.com/en/stable/ : A resource for getting started with flask
- Flask Tutorial (Real Python)(https://realpython.com/tutorials/flask/)

- Flask Mega-Tutorial by Miguel Grinberg (https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world)

6. Other Instructions:

The below will be updated in 1-2 weeks based on feedback and student progress

- The project needs a report written in latex. Details of what this should contains will be provided later in this document itself
- The main focus of the project is to perform the analysis and plotting of logfiles, and not building a very good-looking website. So, even some basic CSS would do for this project. However, absence of any CSS will affect the evaluation negatively.
- The mark distribution for basic tasks (15 marks) will also be updated here.
- Customization will be extra credit and can compensate for poor performance in other exams. But this is capped at 3 Marks (20% of 15).
- Any corrections to the problem statement based on feedback will also be updated in this document and will be highlighted.

7. Modules

- You can use matplotlib, numpy, flask, os, subprocess, re modules, time function from time module for basic task completion
- For customizations only (not for basic tasks), you can use pandas, csv.

8. Report

You must write a project report in **LaTeX.** It must include:

- **Introduction:** brief introduction of the project.
- **Running Instructions:** How to run the application.
- **Website Layout:** Overview of the website and all the pages.
- **Modules:** List of all external libraries used and their purpose.
- **Directory Structure:** Description of all files and the functions they contain (briefly and not a function wise description).
- **Basic & Advanced Features:** Implemented features with explanations and images.
- **Project Journey:** Key learnings, challenges, and how they were addressed.
- **Bibliography**: References and links to external sources.