Project Report

Rohan Jadhav

April 2025

1 Introduction

This project is a web-based application developed using Python and the Flask framework. Its primary function is to parse, filter, and visualize Apache log files in an interactive and user-friendly manner. Users can upload log files in the Apache format, apply various filters based on time periods, severity levels, and event identifiers, and download the filtered logs in CSV format. The application also offers dynamic data visualizations to help identify trends and patterns in the log data. These visualizations can be tailored for specific time frames using the filtering options available on the plots page.

The application includes several core features:

- **Apache File Support**: Users can upload Apache log files, which are automatically converted into CSV format to enable easier analysis.
- **Filtering Capabilities**: Users can filter the log data based on date, time, log severity level, and specific event IDs, allowing for focused analysis.
- **Data Visualization**: The system generates insightful plots including time series graphs, distribution charts, and frequency plots.
- **Export Options**: Users can download both the filtered log data and generated visualizations in user-friendly formats such as CSV and PNG.

2 Requirements

All the dependencies and libraries used in the project are documented in the `requirements.txt` file, which is located in the root directory of the project alongside the `app.py` file. Below is a comprehensive list of the Python modules used:

```
blinker==1.9.0
click==8.1.8
contourpy==1.3.1
cycler==0.12.1
Flask==3.1.0
```

```
fonttools==4.57.0
itsdangerous==2.2.0
Jinja2==3.1.6
kiwisolver==1.4.8
MarkupSafe==3.0.2
matplotlib==3.10.1
numpy==2.2.4
packaging==24.2
pillow==11.2.1
pyparsing==3.2.3
python-dateutil==2.9.0.post0
six==1.17.0
Werkzeug==3.1.3
```

To set up and run the project, Python and pip must be installed on your system. The steps to install the dependencies are as follows:

1. Verify that Python and pip are installed on your system.
2. Download the `requirements.txt` file.
3. Execute the following command in your terminal to install all required modules:

```
pip install -r requirements.txt
```

3 Running Instructions

Once all dependencies are installed, navigate to the directory containing `app.py` and run the following command to start the application:

```
flask --app app run
```

This will launch the Flask development server, allowing you to access the website in your browser.

4 Website Layout

Below is a detailed description of each page included in the website.

4.1 View Logs Page

This page provides a comprehensive overview of all uploaded log files. At the top of the page, there is a navigation button that leads to the Upload Logs page. For each uploaded log file, there are quick action buttons that allow users to view associated graphs, display the log content in CSV format, and download the log as a CSV file.

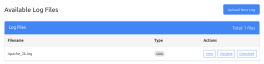

Figure 1: View Logs Page

4.2 Upload Log File Page

This page enables users to upload a new log file for processing. Only files with a `.log` extension are accepted. If the uploaded file does not conform to the Apache log format, the system returns an error message.

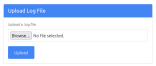

Figure 2: Upload Logs Page

4.3 Display Log Page

Here, the contents of a log file are displayed in CSV format. Users can apply filters based on specific months, days, hours, severity levels, and event IDs. After applying the filters, users can download the refined data in CSV format or navigate to the plots page for visual analysis.

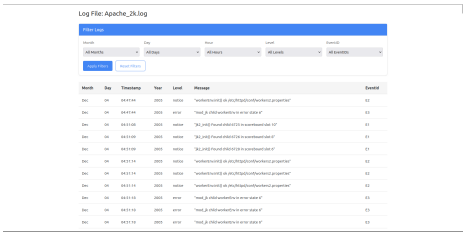

Figure 3: Display Logs Page

4.4 Plots Page

This section presents visual representations of the log data. Users can view different types of graphs, such as time series plots, pie charts for log level distribution, and bar charts for event frequency. A filter form at the top of the page allows users to generate plots for specific time periods. Visualizations can be downloaded in PNG format.

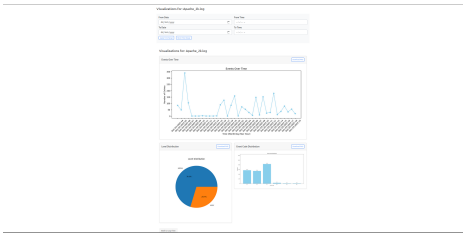

Figure 4: Plots Page

5 Modules Used

5.1 Flask

Flask is the core web framework that powers the application. It handles routing between pages, manages HTTP requests and responses, and connects frontend templates with backend Python functions. The `subprocess` module is used within Flask to invoke Bash scripts for file validation and processing.

5.2 matplotlib

Used extensively for data visualization, matplotlib generates time series line plots, pie charts for log levels, and bar charts for event frequency. These plots are rendered dynamically and can be downloaded.

5.3 Numpy

Although used minimally, numpy provides crucial functions like `isnan()` for data validation and `max()` for calculating dynamic chart scales.

5.4 OS

The os module facilitates file handling and directory management, such as reading uploaded files and saving filtered versions.

5.5 Subprocess

Subprocess is essential for running Bash scripts that handle parsing, filtering, and validating log files. These scripts process the data efficiently using commands like awk and sed.

5.6 base64

The base64 module encodes binary image data into strings that can be embedded directly into web pages or downloaded as PNG files.

5.7 IO

The io module is used to stream plot images directly to users' devices without saving them to the server, thereby optimizing storage and improving speed.

6 Directory Structure

/scripts

- `filter_logs.sh`: Filters the log file based on user-defined criteria.
- `parse_logs.sh`: Parses the log file into CSV format and assigns event IDs.
- `validate_logs.sh`: Checks whether the uploaded file conforms to Apache log format.

/static/uploads: Stores uploaded and filtered log files.
/templates:

- `base.html`: The base HTML structure used across all pages.
- `display.html`: Renders filtered log data and associated controls.
- `plots.html`: Displays plots and filtering options.
- `upload.html`: Upload interface for log files.
- `view_logs.html`: Shows all uploaded files and navigation options.

/app.py: Contains core logic, route definitions, and script integrations.
/requirements.txt: Lists all project dependencies.

Basic Features

- **Log Processing**
  - Accepts Apache log files via a secure upload interface.

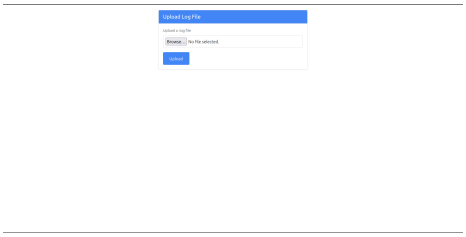

Figure 5: The Upload Page

  - Automatically validates format and converts the data to CSV using Bash scripts.

- **Data Filtering**
  - Users can apply filters such as date, time, level, and event ID.
  - Filtering is executed through a combination of Python and efficient Bash scripts.

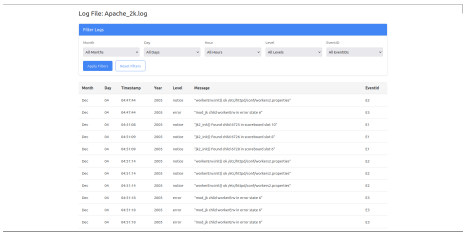

Figure 6: Data Filtering Page

- **Visualization**
  - Plots include line graphs, pie charts, and bar charts.
  - Interactive interface with filter options for customized visualizations.
  - Users can export both data and plots.

Advanced Features

- **Dynamic Analysis**
  - Real-time plot generation using current data.
  - Filter plots by time ranges to observe changes over specific periods.
- **Security**

Figure 7: Visualization Page

  - Ensures safe file handling by validating file names.
  - Prevents unsupported file uploads through format checks.

- **Upload History**
  - Maintains a list of previously uploaded files.
  - Quick-access buttons let users revisit previous analyses easily.

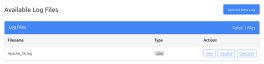

Figure 8: Upload History and File Navigation

7 Project Journey

Working on this project provided me with a deep understanding of how modern web applications are built and deployed. It taught me not only how data can be processed and visualized on the backend but also how different technologies—Python, Bash, and HTML/CSS—can be integrated seamlessly.

I faced several technical challenges, such as mastering Flask routing and Jinja templating, fixing errors related to carriage returns that affected CSV file generation, and understanding modules like `os`, `subprocess`, `base64`, and `io`. These challenges turned into valuable learning experiences as I resolved them through reading documentation, discussing with peers, and seeking support from generative AI tools when necessary.

8 Bibliography

- Class slides provided during the course

- Official documentation for all used Python modules
- YouTube tutorials for Flask and Jinja2
- Generative AI tools for code explanation and bug fixing