Log File Analysis

Vedant Saini
CS-104 Project

Spring 2024–25

Abstract

Documentation of the web-application made for the including its features, customizations and a brief overview of how it works.

Contents

1 Introduction

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

2 Modules

The external dependencies used are:

- `flask` - Handles the backend of the web application. To install, run `pip install flask` in the project directory
- `subprocess` - To call the bash script from python
- `matplotlib` - For showing graphs and plots of the processed logs. To install, run `pip install matplotlib` in the project directory
- `numpy` - For plotting datetime values in matplotlib, they are first converted to a numpy array of *dtype='datetime64'*. To install, run `pip install numpy` in the project directory
- `csv` - To read CSV files (for customizations)
- `GAWK` - Parsing log files, converting them to CSV, filtering according to datetime. More powerful than standard AWK, provides various built-in functions. To install, run `sudo apt install gawk` in the project directory

3 Descriptions of Files and Functions

- **app.py** - Controlling the backend of the application and integrating everything together
- **bash** - Contains all the bash, SED and AWK code required to convert the raw logfile to a CSV, determining the event for each log entry according to the event templates and parsing it for datetime ranges.

  bash/make_csv.sh - Determines the format of the raw logfile, does pre-processing and sends it for further processing to the respective .awk file

  bash/*.events - Contain the event template and the corresponding regular expression for each event ID, which the .awk files refer to and determine the event ID of each log (for Android and System log formats)

  bash/*.awk - Assign events (direct matching of events for Apache log format, no .events file), check for datetime range validity and converts the raw log to a CSV file
- **graphs** - Handling plotting for respective log formats
- **processed** - Contains *upload.csv*, processed version of uploaded log file and *graphs.csv*, processed version according to the datetime range specified for plots
- **static** - Stored the CSS files and images of the plots which are downloaded
- **templates** - HTML code for the webpages
- **uploads** - Stores the raw logfile chosen by the user as 'input_file'
- **all_uploads** - Stores all the files uploaded by the user

4 Directory Structure

The project directory is as follows:

```
project
├── app.py
├── bash
│   ├── make_csv.sh
│   ├── android.events
│   ├── system.events
│   ├── apache.awk
│   ├── android.awk
│   └── system.awk
├── graphs
│   ├── graph_android.py
│   ├── graph_apache.py
│   └── graph_system.py
├── processed
│   ├── graphs.csv
│   └── upload.csv
├── static
│   ├── bar.png
│   ├── line.png
│   ├── pie.png
│   ├── graphs.css
│   ├── log_display.css
│   └── log_upload.css
├── templates
│   ├── graphs.html
│   ├── log_display.html
│   └── log_upload.html
├── uploads
│   └── input_file
└── all_uploads
    └── (all user uploads)
```

5 Running the Site and Layout

First install the required modules and dependencies as listed above. To start the application, run `python app.py` in the `project` directory.
The application consists of three pages: log upload page, log display page and plot graph page. Back buttons have been introduced for ease of navigation between various pages. Their descriptions follow:

5.1 Log Upload

The application starts with this page[1] where the user can upload multiple files from which they can choose which to process and specify the datetime range of log entries to be processed. The application auto-detects the format and flashes a message if the log has invalid format (not one of Apache, Android or Linux System), there are no logs between the given dates or the chosen file is a byte file.
If the log doesn't contain some datetime component (year or milliseconds), any value can be input in the corresponding form field.

Once files are uploaded, they are listed on the left pane. The user can choose a file from the drop-down menu. Datetime filter can also be applied before processing. The left pane shows all uploaded files and the chosen file.

After processing, the user can:

- Download the CSV file
- View the CSV as a table and apply more filters
- Plot and visualize data and download the plots

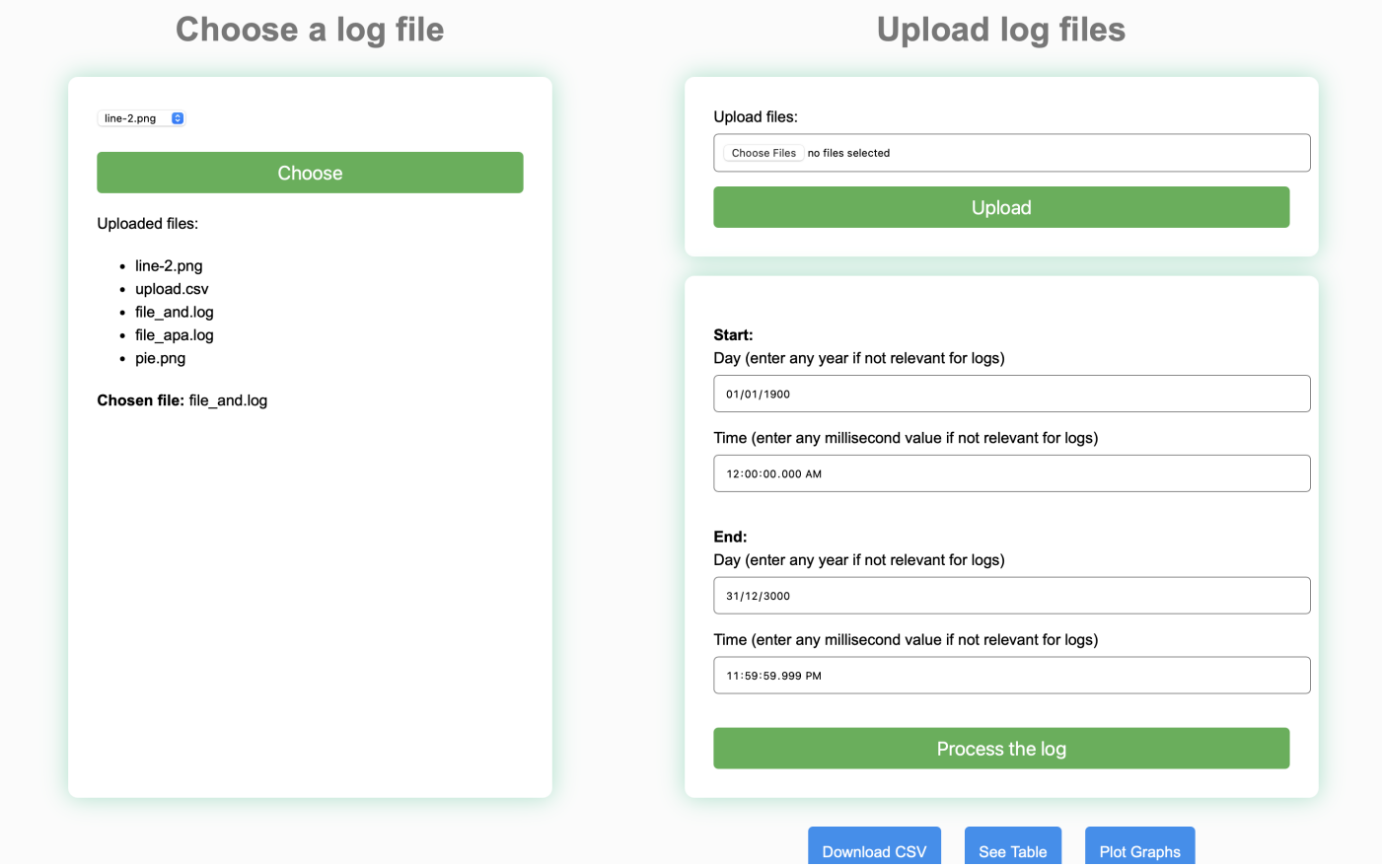

Figure 1: Log Upload Page, showing uploaded files and chosen file - 'file_and.log'

5.2 Log Display

This page[2] renders the CSV file as a table for the user to view, filter and sort records.

Clicking on any header cell sorts the corresponding field either numerically or alphanumerically depending on the cell content. Clicking again toggles the order of the sorting (ascending to descending and vice-versa). To remove the sort, a 'Remove Sort' button is also placed.

Clicking on any cell in the table body shows only those rows having the same entry in their respective field. Clicking again toggles off the filter and everything is visible again. Clicking on another field while a filter is set removes the previous filter and applies the new one.

5.3 Plotting Graphs

The user is again given an option to choose the datetime range they want to plot data from. Checks for file upload have been put and a message in red is flashed in case of any error. The user can then see three plots[3]: line graph, pie chart and a bar graph which they can choose to download.

Figure 2: Log Display Page, showing rows filtered for 'error' in column 3 and sorted in descending order by EventId

(a) Graph form

(b) Plotted line graph

Figure 3: Plot Graph Page

6 Customizations and extra features

Following is the list of customizations done in the webapp:

- Added support for all features for three log formats - Apache, Android, System (Linux)
- Descriptive error handling via message flashing in Flask
- Allowing upload of multiple files
- Drop-down menu for choosing logfile
- Independent datetime filter for plotting graphs
- Filtering rows of the table based on a field
- Sorting of rows of the table based on a field
- User friendly interface with ready confirmation about file uploads, file choice and row filter, sorting in table (via coloured cells)

7 Project Journey

I chose this project because it looked the most intimidating and offered the most opportunity for learning. And indeed, I finally got to know what backend is! I learnt how things work in a server and a bit about the transfer protocols -

'GET' and 'POST'. Also, this project was not just limited to the curriculum but involved learning stuff on the fly (Jinja templating, CSS features like flexbox, and of course - Flask)

The development of the project was an enjoyable experience where I learnt lots of new things. The initial phase was the most uncomfortable because whenever I thought I was done with modularisation and segregation, I fell short in the face of every new feature I was implementing. Only slowly did I stabilize upon a directory structure which would accommodate the needs of the project and at the same time, be compact.

Applying customizations were the parts which required a bit of searching and figuring stuff out.
For example, for a long time, I was trying to find a way to sort and filter the rows of the table using Python! It took me a while to see JavaScript was also a possibility.

My first draft had the option of uploading just a single file. But then accommodating multiple files required me to restructure my code and directory structure and make them more organised.

Overall, this was a nice amalgamation of many of the topics covered in the course and more. All that is left is to 'wc -l' the files I have written :P

8 Bibliography

References

[1] StackOverflow Forums https://stackoverflow.com/.

[2] Flask Official Documentation https://flask.palletsprojects.com/en/stable/.

[3] W3 Schools Tutorials https://www.w3schools.com/.

[4] MDN Web Documentation https://developer.mozilla.org

[5] GNU Awk User's Guide https://www.gnu.org/software/gawk/manual/gawk.html