

Log File Analysis

Kona Pavan Sai Subhash

April 29, 2025

Contents

| | | |
|----------|--------------------------------|----------|
| 1 | Introduction | 2 |
| 2 | Requirements | 2 |
| 2.1 | Mac User | 2 |
| 2.2 | Linux User | 2 |
| 2.3 | Windows User | 2 |
| 3 | Running Instructions | 2 |
| 4 | Website Layout | 2 |
| 4.1 | Home | 3 |
| 4.2 | Log Display | 3 |
| 4.3 | Plots and Graphs | 3 |
| 5 | Modules | 4 |
| 5.1 | NumPy | 4 |
| 5.2 | Matplotlib | 4 |
| 5.3 | Flask | 4 |
| 6 | Directory Structure | 4 |
| 6.1 | Static | 4 |
| 6.2 | Templates | 4 |
| 6.3 | file.read.py | 4 |
| 6.3.1 | check_file() | 4 |
| 6.3.2 | file_upload() | 4 |
| 6.3.3 | parse_csv() | 4 |
| 6.3.4 | get_graphs() | 5 |
| 6.4 | Some Other Files | 5 |
| 6.4.1 | check.sh | 5 |
| 6.4.2 | make_csv.sh | 5 |
| 6.4.3 | custom_csv.sh | 5 |
| 6.4.4 | custom_csv.py | 5 |
| 6.4.5 | plots.py | 5 |
| 6.4.6 | date_functions.py | 5 |
| 7 | Features of the Project | 6 |
| 8 | Project Journey | 8 |

1 Introduction

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

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

This project uses Flask, HTML/CSS/JavaScript, bash, and Python to do the above tasks.

2 Requirements

2.1 Mac User

- If you have a MacBook, then there is a pre-installed bash in your laptop.
- Install Python 3.12 from this [link](#) and also make sure you have matplotlib, numpy, and flask modules installed before you run the program.
- Install them using 'pip install numpy', 'pip install matplotlib', and 'pip install flask' for numpy, matplotlib, and flask, respectively, in the terminal.

2.2 Linux User

- Same as MacBook, you have bash installed on your laptop
- Install Python 3.12 from this [link](#) and ensure you have matplotlib, numpy, and flask modules installed before you run the program.
- Install them using 'pip install numpy', 'pip install matplotlib', and 'pip install flask' for numpy, matplotlib, and flask, respectively, in the terminal.

2.3 Windows User

Go ahead and jump out of your Windows.

Jokes aside, I don't know how to download the requirements here, but try them yourself or install Linux on your laptop. The requirements are the same as above.

3 Running Instructions

- Open terminal and open the project directory. Run the "flask - -app file_read.py run" command.
- Copy the link on your terminal and paste it into a browser. Make sure you don't terminate the process, it should run in the background{Most probably the link will be like this `https://127.0.0.1:5000`}

4 Website Layout

Upon opening the link, you are directed to the Home page.

4.1 Home

You are required to upload the specific log file of your choice — either an Apache Error Log or an Android log.

You should upload a log file strictly in the expected format. Even a single line in an incorrect format will result in an error.

Once submission is done, you will be redirected to the Log Display Page.

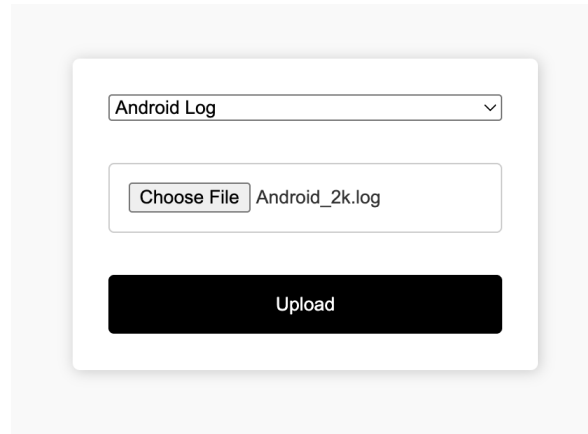


Figure 1: Home Page

4.2 Log Display

- You will see a navigation bar at the top of the page that allows you to move between different parts of the website. It contains Home¹, View Logs² and Plots³
- The uploaded logs are shown in a tabular format.
- You can download the entire log file as a structured CSV.
- You also have the option to filter and sort the log file.

4.3 Plots and Graphs

- The same navigation bar is present here for easy access to all parts.
- Here, you will see visualizations generated from your uploaded log file.
- You need to select the type of plot you want to view from the drop-down menu
- You also have an option to download the plot as a PNG.

¹Redirects to the Home Page

²Redirects to the Log Display Page

³Redirects to Plots and Graphs

5 Modules

Numpy, **Matplotlib**, and **Flask** are the Python modules used in this project.

5.1 NumPy

NumPy [2] is used during the plotting process to handle data more efficiently and concisely. It helps eliminate the need for explicit `for` loops by enabling vectorized operations.

5.2 Matplotlib

Matplotlib [1] is used to generate plots from the uploaded log files in the specified formats.

5.3 Flask

Flask [3] is used as the backend framework to run the entire website. It serves as the backbone, handling routing, file processing, and server-side operations.

6 Directory Structure

6.1 Static

Flask automatically adds a static view that takes a path relative to the `./static` directory and serves it.

It contains the CSS files and the PNG images of the generated plots. The filenames are self-explanatory and easy to understand.

6.2 Templates

This folder contains the HTML files that Flask uses to render content onto the web page.

There is a `home.html` file, along with two separate HTML files each for Apache and Android log displays.

6.3 file_read.py

This is the main Flask file that needs to be run to start the website.

It contains the following functions:

6.3.1 check_file()

This function checks whether the uploaded file is of the log file type.

6.3.2 file_upload()

This function is triggered only when the user is on the Home page. It checks both the file type and its format⁴.

If the file does not match the expected format, an error is returned to the webpage. Otherwise, the user is redirected to the Log Display page.

6.3.3 parse_csv()

This function is triggered only when the user is on the Log Display page for Apache logs. It converts the CSV data generated by `make_csv.sh` and passes it to the HTML template for rendering.

It also processes data submitted through a form, allowing the user to extract specific log entries. The resulting output is then displayed on the webpage.

⁴By running a Bash script: `check.sh` or `check.android.sh`, based on the selected option.

This function specifically handles Apache error logs.
A similar version⁵ of this function is implemented separately for Android logs.

6.3.4 `get_graphs()`

This function is triggered only when the user is on the Plots and Graphs page for Apache logs. It receives the plot type along with a few other inputs related to specific log entries, and passes them to `plots.py` to generate the corresponding plots.

This function specifically handles Apache error logs.

A similar version of this function⁶ is implemented separately for Android logs.

6.4 Some Other Files

6.4.1 `check.sh`

This script checks whether the uploaded Apache error log is in the correct format.

A similar version⁷ of this script is implemented separately for Android logs.

6.4.2 `make_csv.sh`

This script converts the log file into a CSV and adds event IDs based on predefined message templates.

A similar version⁸ of this script is implemented separately for Android logs.

6.4.3 `custom_csv.sh`

This script creates a customized CSV based on user input. Currently, it supports only time-based filtering.

6.4.4 `custom_csv.py`

This script handles filtering based on event type and log level.

A similar version⁹ of this script is implemented separately for Android logs.

6.4.5 `plots.py`

This script generates plots based on user-provided data and preferences.

A similar version¹⁰ of this script is implemented separately for Android logs.

6.4.6 `date_functions.py`

This Python file contains functions related to date format conversions and timestamp comparisons.

The date format conversion functions include `change_date()`, `get_date()`, and `get_date_andr()`, with their respective descriptions provided as comments within the code.

The time comparison functions are `time_list_andr(t1)` and `timestamp_comparison(t1, t2)`, which have self-explanatory names.

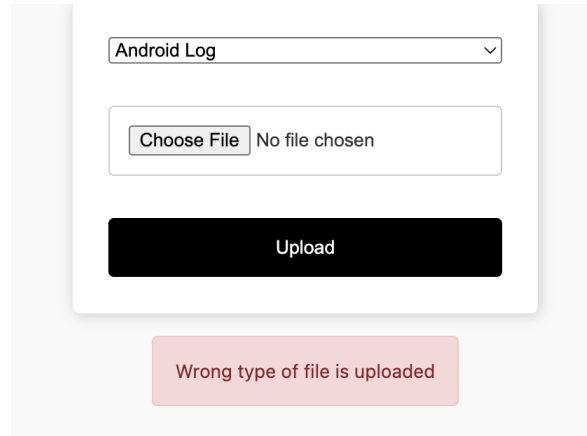
Refer to the code comments for more detailed information.

⁵`parse_csv_android()`
⁶`get_andr_graphs()`
⁷`check_android.sh`
⁸`make_android_csv.sh`
⁹`custom_android_csv.py`
¹⁰`plots_android.py`

7 Features of the Project

On the Home page, if a user uploads a file of the wrong type or with an incorrect log format, an error message is displayed on the screen.

There is also an option to select the type of log file you are uploading—either an Android log or an Apache error log. This can be seen in Figure 1.



Android Log

Choose File No file chosen

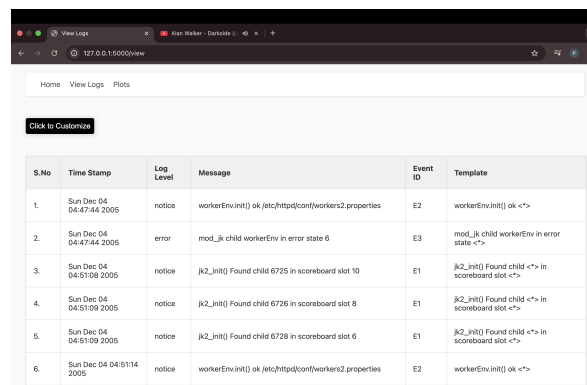
Upload

Wrong type of file is uploaded

Figure 2: Displayed Error Message

Once you upload a correctly formatted file, you will be redirected to the Log Display Page. Here, the log file is shown in a structured tabular format.

By clicking the **Customize** button, you can access options to filter and retrieve specific log entries.



| S.No | Time Stamp | Log Level | Message | Event ID | Template |
|------|--------------------------|-----------|--|----------|--|
| 1. | Sun Dec 04 04:47:44 2005 | notice | workerEnv.init() ok [http://conf/workers2.properties | E2 | workerEnv.init() ok <*> |
| 2. | Sun Dec 04 04:47:44 2005 | error | mod_ik child workerEnv in error state 0 | E3 | mod_ik child workerEnv in error state <*> |
| 3. | Sun Dec 04 04:51:08 2005 | notice | [k2_init] Found child 6725 in scoreboard slot 10 | E1 | [k2_init] Found child <*> in scoreboard slot <*> |
| 4. | Sun Dec 04 04:51:09 2005 | notice | [k2_init] Found child 6726 in scoreboard slot 8 | E1 | [k2_init] Found child <*> in scoreboard slot <*> |
| 5. | Sun Dec 04 04:51:09 2005 | notice | [k2_init] Found child 6728 in scoreboard slot 6 | E1 | [k2_init] Found child <*> in scoreboard slot <*> |
| 6. | Sun Dec 04 04:51:14 2005 | notice | workerEnv.init() ok [http://conf/workers2.properties | E2 | workerEnv.init() ok <*> |

Figure 3: Log Display before clicking the Customize button

After clicking the **Customize** button, a form appears allowing you to enter specific criteria for filtering the log data as per your needs.

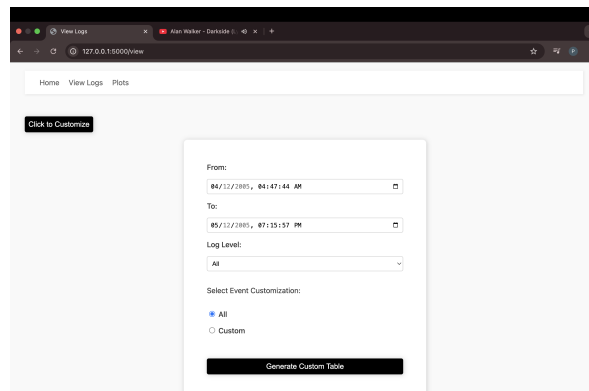


Figure 4: Log Display after clicking the Customize button

You can also select the **Custom** option in the form to enable checkboxes for filtering specific events. You must then choose the types of events you wish to view.

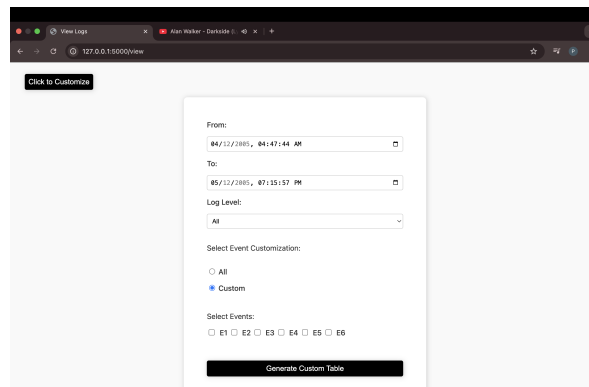


Figure 5: Custom Event Selection Interface

You can navigate to the Plots and Graphs page by clicking on the **Plots** option in the navigation bar. After redirection, you will see a page like the one shown below⁶.

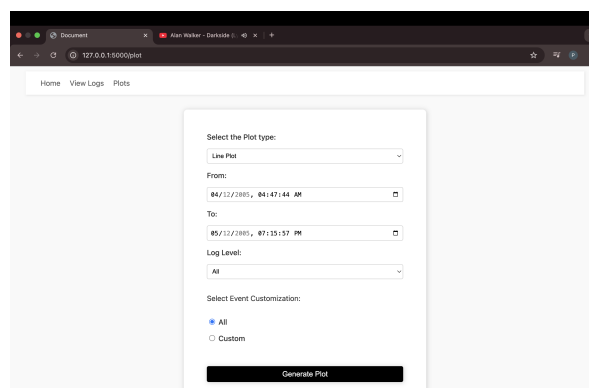


Figure 6: Plots and Graphs Page

After submitting the form, the corresponding plot will be displayed as shown below⁷.

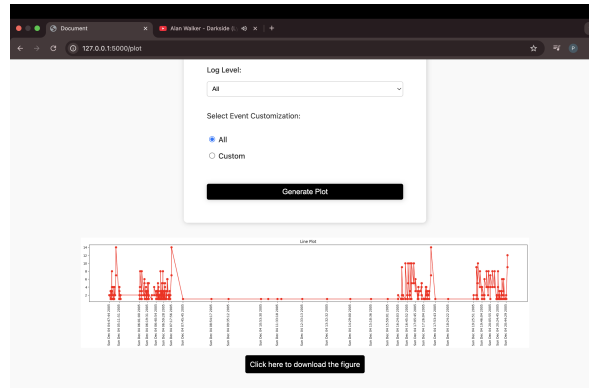


Figure 7: Generated Plot based on User Input

8 Project Journey

- I began learning Flask using a YouTube tutorial [4] along with the official Flask documentation [3].
- I initially struggled with implementing the file upload functionality, which took considerable effort in the first few days.
- Once that part was complete, the rest of the development process became smoother, and I was able to pick up speed.
- Writing the `check.sh` and `make_csv.sh` scripts was relatively easier, and the same goes for generating the table on the log display page.
- I initially implemented filtering in `custom_csv.py`, thinking it could be efficiently handled using Python. However, after discussing with a friend, I realized it was essential to perform the filtering using a Bash script. I then learned that `awk` supports defining functions, which helped me complete the Bash-based filtering functionality.
- I then moved on to developing `plots.py`, which I initially assumed would take only a few hours. However, a friend pointed out an issue with the line plotting logic. Although they mentioned that the existing version was acceptable, I chose to improve it. The issue was that the time intervals between data points were irregular, but I had originally plotted them with equal spacing. To address this, I wrote a function to calculate the time difference between consecutive points and used it to adjust the spacing in the graph. I hope this thoughtful implementation will be recognized and may give me an edge over my peers.
- I then explored additional features such as event-based and log-level filtering. I also integrated JavaScript to enhance the user interface and improve the overall user experience.
- I then shifted my focus to designing the HTML using CSS. Whenever I was unsure how to implement certain design elements, I used help from AI tools. Eventually, I completed the project with a minimal and clean design.
- After my end-semester exams, I attempted to extend the project to support Android logs. Initially, I thought this would be straightforward, but it turned out to be time-consuming, especially the part involving automation of pattern recognition and assigning event IDs. Writing the regex-based automation using a template file was particularly challenging. Eventually, I succeeded, although I had to hard-code two lines in the template.
- This was my first experience completing a full-fledged project. I felt very enthusiastic throughout the process and experienced a great sense of satisfaction after successfully implementing each part. I am truly grateful to this course for giving me the opportunity to work on a hands-on project. I look forward to exploring more interesting projects in the future.

References

- [1] Matplotlib Developers. *Matplotlib Documentation*. URL: <https://matplotlib.org/stable/contents.html>.
- [2] NumPy Developers. *NumPy Documentation*. URL: <https://numpy.org/doc/stable/>.
- [3] Pallets Projects. *Flask Documentation*. URL: <https://flask.palletsprojects.com/>.
- [4] Corey Schafer. *Flask Tutorial Playlist*. URL: <https://youtube.com/playlist?list=PL-osiE80TeTs4UjLw5MM60jgkjFeUxCYH>.