DS-WS-Team No. 03

Deepak Seervi  
Sanjit Kumar  
Yashumithaa M

LOG ANALYSER

Table of Contents

[Requirement 2](#_Toc49019750)

[Solution Approach 3](#_Toc49019751)

[Tools & Libraries Used 4](#_Toc49019752)

[Steps to Compile & Execute 5](#_Toc49019753)

[Prerequisite 5](#_Toc49019754)

[Software Required 5](#_Toc49019755)

[How to execute 5](#_Toc49019756)

[Verification 6](#_Toc49019757)

[Approaches Tried 7](#_Toc49019758)

[Solution Improvements 8](#_Toc49019759)

# Requirement

Following is an extract from a log file that records User’s activities on a web application:

2018-09-18 04:49:38,215 ERROR (default task-95)

IP-Address=157.49.141.133#,!User-Agent=Mozilla/5.0 (Windows NT 10.0; WOW64;

Trident/7.0; rv:11.0) like

Gecko#,!X-Request-From=UIX#,!Request-Type=POST#,!API=/v1/admin/developers#,!U

ser-Login=test@demo.com#,!User-Name=testUser#,!EnterpriseId=2#,!EnterpriseNam

e=Enterprise-2#,!Auth-Status=#,!Status-Code=200#,!Response-Time=346#,!Request

-Body=

2018-09-18 04:49:39,842 ERROR (default task-21)

IP-Address=157.49.141.133#,!User-Agent=Mozilla/5.0 (Windows NT 10.0; WOW64;

Trident/7.0; rv:11.0) like

Gecko#,!X-Request-From=UIX#,!Request-Type=GET#,!API=/v2/developers#,!User-Log

in=test@demo.com#,!User-Name=testUser#,!EnterpriseId=2#,!EnterpriseName=Enter

prise-2#,!Auth-Status=#,!Status-Code=200#,!Response-Time=1240#,!Request-Body=

2018-09-18 04:49:41,946 ERROR (default task-127)

IP-Address=157.49.141.133#,!User-Agent=Mozilla/5.0 (Windows NT 10.0; WOW64;

Trident/7.0; rv:11.0) like

Gecko#,!X-Request-From=UIX#,!Request-Type=GET#,!API=/v2/developers#,!User-Log

in=test@demo.com#,!User-Name=testUser#,!EnterpriseId=2#,!EnterpriseName=Enter

prise-2#,!Auth-Status=#,!Status-Code=200#,!Response-Time=1117#,!Request-Body=

To analyse these logs manually takes significant effort and time. It requires a developer to extract the logs,

parse and review them, making it significantly time consuming.

**WORK SAMPLE OBJECTIVE**

Design a Log analyser which takes a file as Input, extracts, parses and stores data in a format that can be used

to easily filter out activities based on different parameters and the final result should contains at least following

information

1. IP Address

2. User Agent

3. Status Code (200/401/500 etc).

4. Request Type (GET/POST/PUT etc)

5. API

6. User

7. Enterprise Id

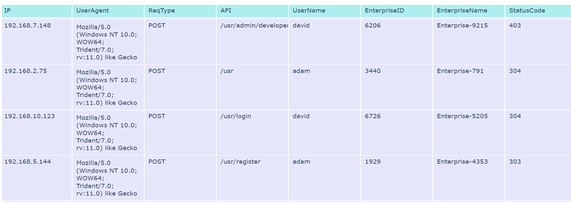
8. Enterprise Name

# Solution Approach

Top to bottom approach is taken to provide a solution for the requirement.

* The inputs are log files which gives an insight on the usage of API from different enterprises with significant details.
* Python scripting language is used because it is known for its automation and data analysis libraries. With a lot of libraries available in python, it is best suited for our work sample.
* Flask library (Web Application Framework) is used to provide REST API through which the user interface can take inputs from the user namely the log file which the user wants to analyse and the parameter they want to filter.
* User Interface is provided using HTML, CSS and JavaScript.
* Parsing of the input is done using Regex because log files follow a pattern which is key=value and separators are used to separate them. Regex are used when you have a common pattern which is present in all the input data.
* Processed input is chunked at the rate of 100,000 entries and updated into a csv file. Since, time is critical, a flat database will reduce the time taken for insertion.
* Pandas library dataframe is used for querying the data because it is easy to load the csv into dataframes for performing the queries.
* Plotly library is used to provide tabular output for the query.

Query output as a Table



# Tools & Libraries Used

* Flask
* Python Scripting Language
* re
* time
* sys
* threading
* json
* plotly

# 

# Steps to Compile & Execute

## Prerequisite

1. Input- Log File
2. Assumptions are captured in the project (Assumptions.md)

## Software Required

1. Operating System- Linux/Windows
2. Required Packages to be installed as per [Tools & Libraries Used](#_1fob9te)

## How to execute

1. Unzip the LogAnalyser.zip file (Eg. E://LogAnalyser)
2. To start the flask server from the terminal, go to the file path, give the following command: python main.py
3. Open local host “127.0.0.1:5000” in a browser
4. Upload the log file
5. Provide inputs for filtering required parameters.

# Verification

Log file has the following parameters-

1. IP Address

2. User Agent

3. Status Code (200/401/500 etc).

4. Request Type (GET/POST/PUT etc)

5. API

6. User

7. Enterprise Id

8. Enterprise Name

The above parameters are taken for filtering.

|  |  |  |
| --- | --- | --- |
| TEST STEP | EXPECTED OUTPUT | RESULT |
| Validate input file with varying file size.   1. Input a file of size 4KB 2. Input a file of size 100 KB 3. Input a file of size 2GB | File uploaded successfully  Validated Input file of different sizes could be uploaded. | PASS |
| Parsing of log file | Log files of all sizes (4KB to 2GB) were successfully parsed with varying time for the given pattern. | PASS |
| Query is taken as an input from the user. | Added debug prints to check if the query parameters entered by the user is received through REST-API properly. | PASS |
| Validate Filtered output is proper with respect to the query.  For example, in the query page when the user enters User-Name=”alex”, check the query output. | All the logs where User-Name=”alex” should be filtered and stored as filtered\_output.csv file. | PASS |
| Depending on query output size, output is represented as a table. | Table is plotted using plotly. | PASS |

# Approaches Tried

* Tried using C++ and GO for conversion of log to csv but was taking longer time for processing than the present solution.
* Tried using SQL(Mysql) but took longer time for processing than the present solution.
* Tried to plot a bar graph with bar mode as a group.
* Tried to plot a pie chart with API and Status Code.
* Tried to implement multithreading for parsing the file.
* Once main.py is run, after the Flask server connection is created, tried to write a script to open the Flask server client in a browser.

# Solution Improvements

* To provide a modernised dashboard-
  + Tried to represent different parameters using different graph styles in the same page.
  + To provide graphical insights for the query output was difficult using Plotly due to its limitation for large dataset. This problem can be addressed if we use Dash library as Dash supports large datasets.
* Multithreading while writing CSV onto the secondary storage.
* SQL Engine (easier queries, user has more control over the data)
* Dynamic key-value pairing
* Using compile level language (like Rust) for conversion of log file to csv.
* Support for IPV6 parsing.
* Support for log file in .txt format.