**Documentation**

**Logfile Analyzer tool**

**Khizar Saud**

**Problem statement**:The challenge the PREG team faced is that a lot of valuable resources during a software quality assurance exercise were spent on manually analyzing log files to figure out; which test cases failed, where the error occurred, and the type of error.

**Objective**: Create a log file analyzer tool which analyzes job files and displays tabulated information in a holistic view.

**Required Modules: import re, import zipfile,import flask, import flask\_mail, import pymongo, import tabulate, import numpy**

**Database: MongoDb (User/Pass)**

**cluster = MongoClient("mongodb+srv://root:toor@**[**cluster0.cx5wq.mongodb.net/?retryWrites=true&w=majority**](http://cluster0.cx5wq.mongodb.net/?retryWrites=true&w=majority)**")**

**DB name = Logs**

**Collection = ‘Logs for data’**

**Email Client config:(**[**Mailtrap - Safe Email Testing**](https://mailtrap.io/signin)**)**

**app.config['MAIL\_SERVER']='smtp.mailtrap.io'**

**app.config['MAIL\_PORT'] = 2525**

**app.config['MAIL\_USERNAME'] = '5f91f0b600be8b'**

**app.config['MAIL\_PASSWORD'] = '90bee1986d8bc4'**

**app.config['MAIL\_USE\_TLS'] = True**

**app.config['MAIL\_USE\_SSL'] = False**

**Flask Config:**

**app = Flask(\_\_name\_\_)**

**app.secret\_key = "super secret key"**

**app.config['TEMPLATES\_AUTO\_RELOAD'] = True**

**app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER**

**app.run(host="0.0.0.0", port=8080, threaded=True,debug=True)**

**UPLOAD\_FOLDER = r'Users\kizsa\Desktop\HTMLproject' (Can be changed to any desired)**

**Implementation:**

* Tool runs from the file **APP.PY.** This application uses **Flask** to create a web application to take in user input in form of a file(ZIP)
* User is allowed to type in location of jobfile through raw text

Example: ‘C:\Users\kizsa\Desktop\Logfolder\renn.zip’

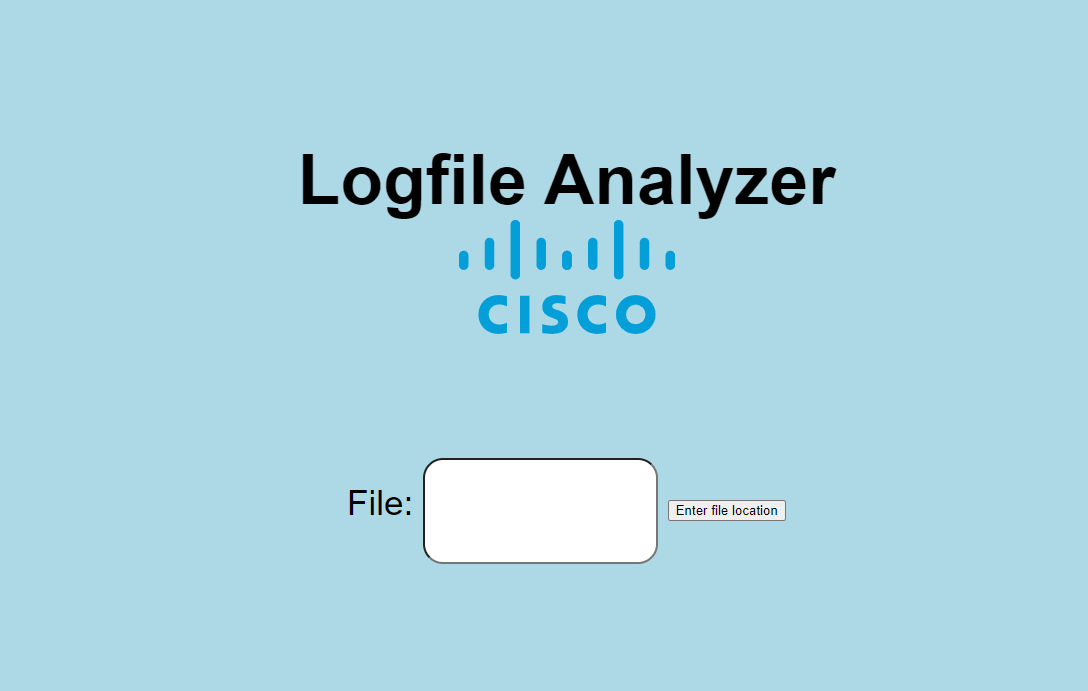
* Once the file location is inputted and the upload button is clicked; a post request is sent to the server which will unzip the file to the UPLOAD FOLDER directory and call the **myfile** function.
* **Myfile**: Is the main function in this application, it grabs the Task log file, shortens it to keep only pertinent information, and grabs the version number of the job.
* Next it calls the **printer.tableoperations()** function which commences analysis of the log file and stores information into respective variables. EX: Test case names,results,failure reasons.
* Next, the **tablebabmaker.tablemaker()** function is called which gathers all the variables and places them into respective categories inside a dictionary and creates a HTML file with the tabulated information. Inside this HTML file there are 3 extra features implemented to enhance user experience. A **detailed reason** button, a **test case log** button, a **delete database** button, and a **filter** function. The **detailed reason** button, is able to show a more detailed reason for failure, this button takes user input in form of a number relating to the test case. EX: if the user wanted to see a more detailed failure reason for the first test case; the user would enter 0 and it would display the detailed reason for failure for the first test case in a HTML alert box, if a detailed reason is already given in the table this button will not work. The **test case** log button, splits the log file by test case and is able to display all information from the log file in one page. EX: if the user wanted to only see what took place in the first test case they would type in zero and the page would display the logfile for the first test case. We also have a delete database button which is connected to a mongodb cluster which will delete the database of the results if necessary. Finally a filter function was implemented to filter the table by keywords. EX: if the log file has 100 test cases and you only want to see the failed test cases you can type FAILED in the filter box and it will show you only the test cases which failed.
* The file **Htmlfilecreatetable.py** is used for the backend operations of the **Test case Log** button

The way this button works is, it splits the log file by test case, and stores it into a list. This file has a method named **bab5()** which returns a list that contains the log file split by testcase. When the **Test case log** button is clicked on the dashboard and user inputs a number the flask application accesses the list from the method and passes it to HTML so the user can view the desired test case.

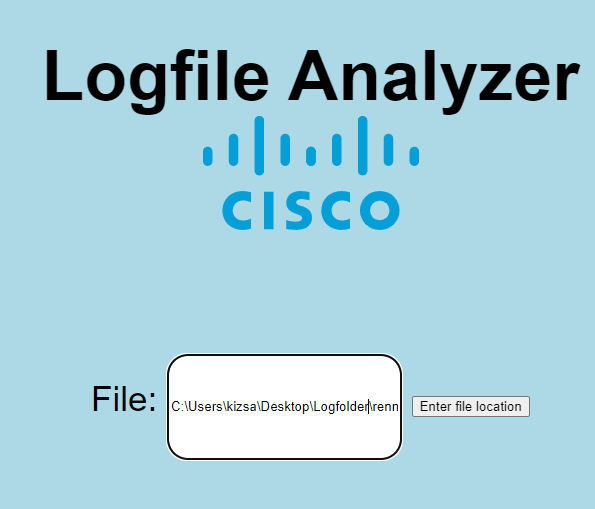
* Next we implemented a SMPT connection to send all tabulated information stored in a HTML file, to an email address as a way to keep track of what has been inputted into the tool.
* Finally the Flask application after completing the above operations returns the information in a holistic view.

**Use**

**Mainpage:**

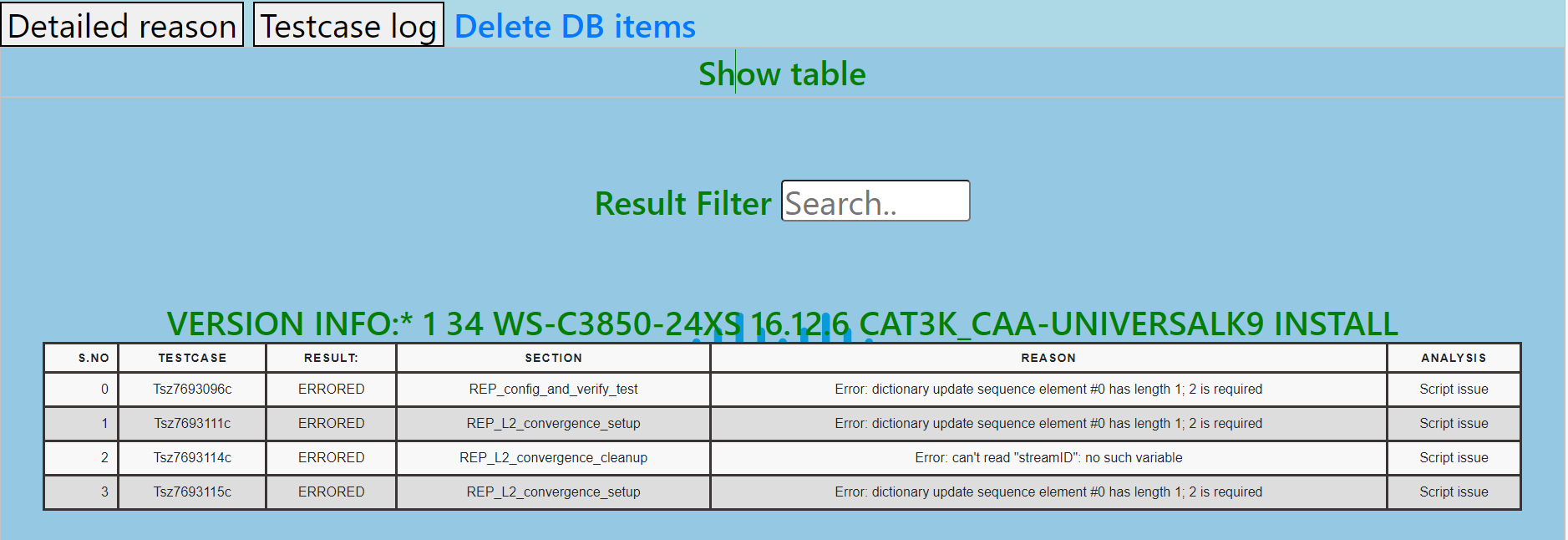
****

Asks for user input as raw text, ex ‘C:\Users\kizsa\Desktop\Logfolder\renn.zip’



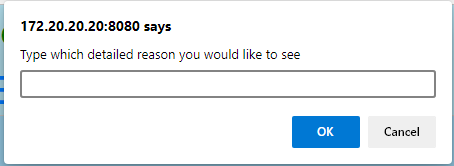
Once the ‘Enter file location’ button is clicked the analysis commences.

**Dashboard:**

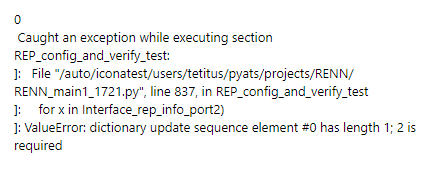


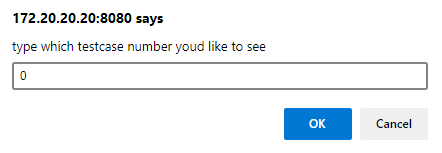
We are shown which test cases ran in the logfile, the result, the reason for failure and an analysis

If we would like to see a **detailed reason** for failure for the first test case we click the detailed reason button and a alert box appears asking for user input, if we would like to see a detailed reason for the first test case we can input 0

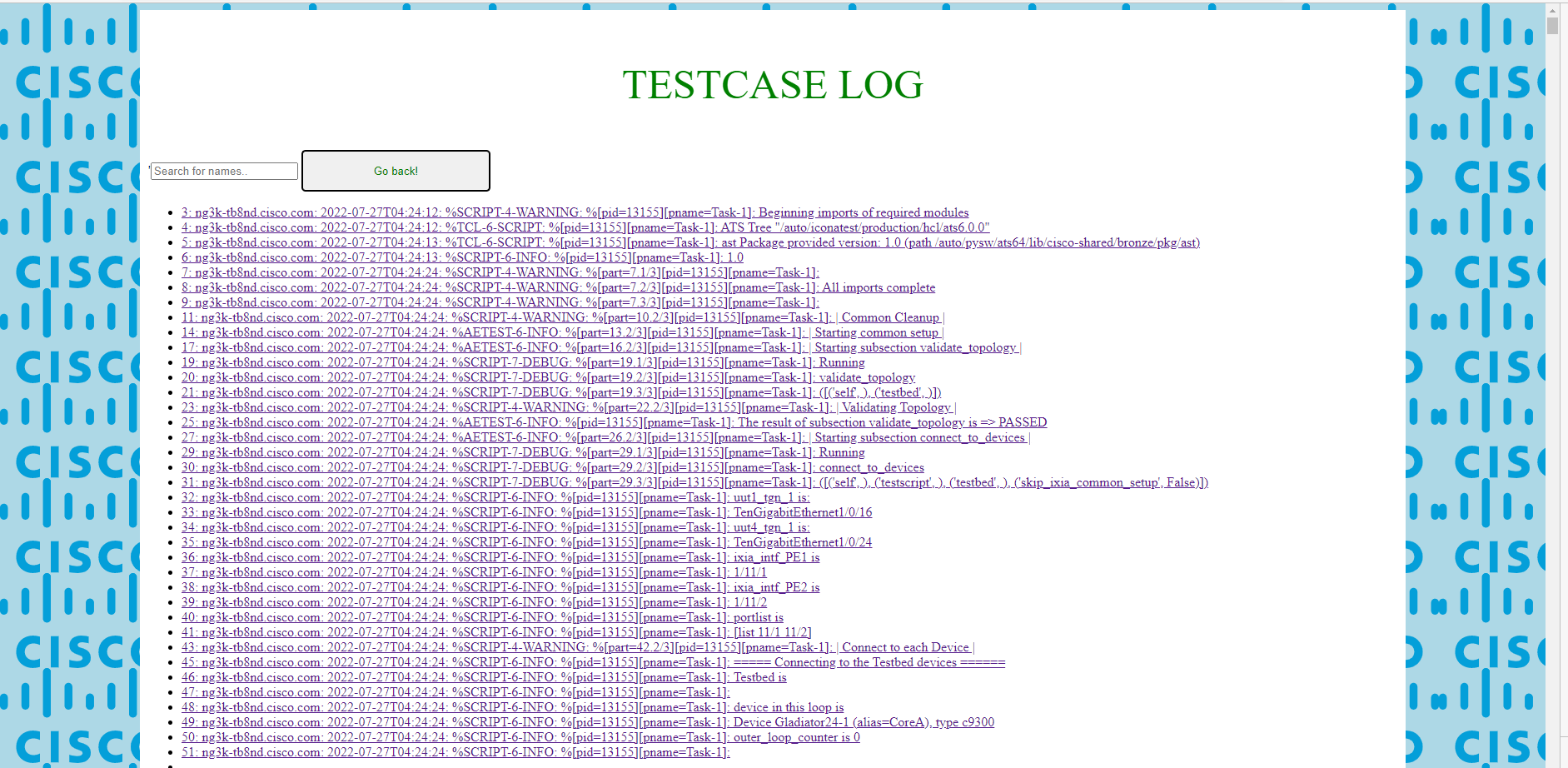
.

Which will display the information in a HTML alert box:

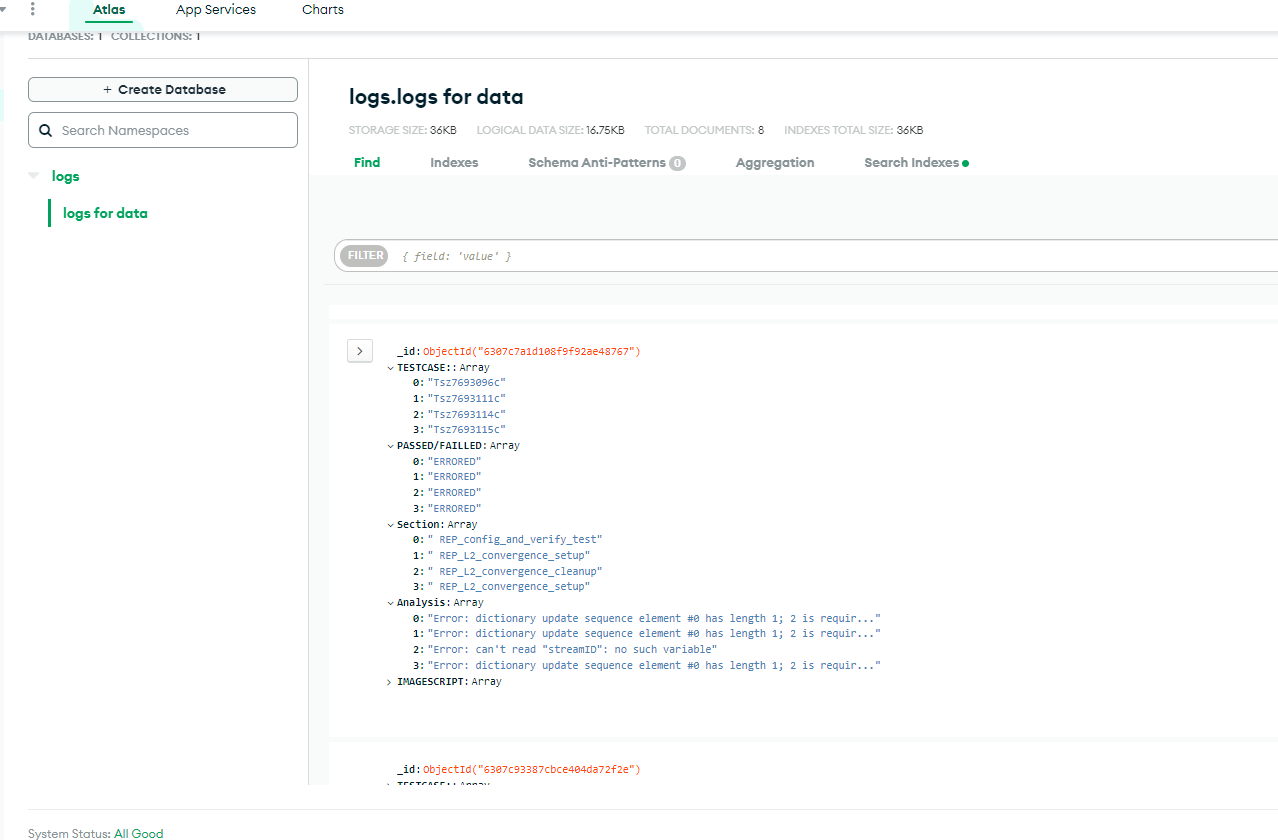


If we would like to see the Test case log for the first test case we would enter 0 

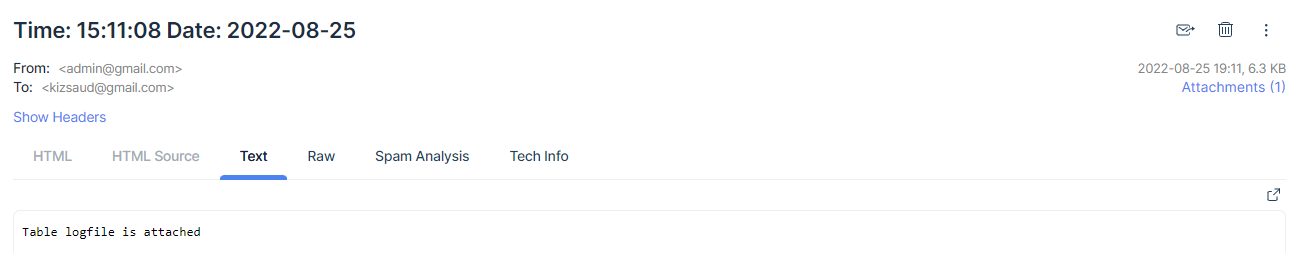
This will take us to a page which will show the test case log and has a button to go back to the table.



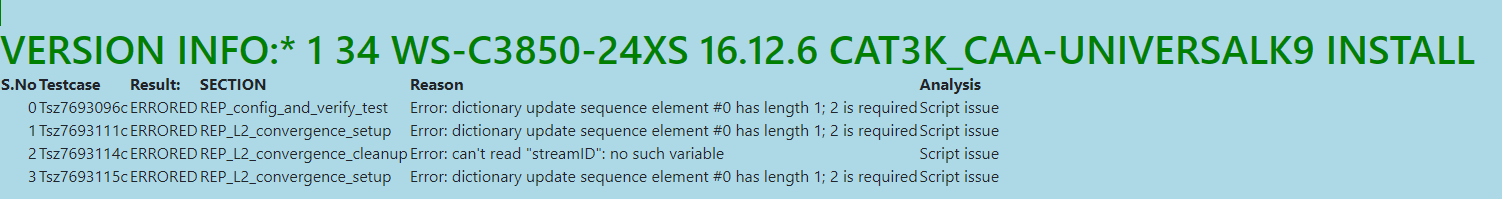
Anytime a file is uploaded and analysis commences, the results are sent to mongodb to keep in a database for later use as a form of a log for the tool:



The table which is displayed on the dashboard is sent to an email address as a form of a log for the tool as well which tells the user what time and date the analysis commenced:



The attachment contains a HTML file with the table information which shows what testcase ran ,result,reason and analysis.



**File Breakdowns:**

