Read Me - PYTHON SCRIPT FOR LOG ANALYSIS

1. Import the necessary modules:

- tkinter, which provides GUI elements
- filedialog, which provides dialogs for selecting files and directories
- os, which provides functions for interacting with the operating system

2. Define a function named "extract_errors_warnings" that takes a path to a log file as input and returns two lists:

- "warnings" containing lines with the string "WARNING"
- "errors" containing lines with the string "ERROR"

This function reads the log file, extracts the relevant lines and appends them to their respective lists.

3. Define a function named "generate_report" that takes three arguments:

- "warnings", a list of warning messages
- "errors", a list of error messages
- "output_file_path", the path to the output file

This function creates a text report of the warnings and errors and writes it to a file specified by "output_file_path".

4. Define a function named "select_input_file" that opens a file dialog and sets the value of the "input_file_entry" widget to the selected file's path.

5. Define a function named "select_output_file" that opens a file dialog and sets the value of the "output_file_entry" widget to the selected file's path.

6. Define a function named "generate_report_action" that gets the values of "input_file_entry" and "output_file_entry" widgets, checks if the input file exists, extracts the warnings and errors from the log file using the "extract_errors_warnings" function, and generates a report using the "generate_report" function.

7. Create the main window using the "tk.Tk()" method and set its title and geometry.

8. Create a label widget with the text "Select input and output files" and pack it to the main window with some padding.

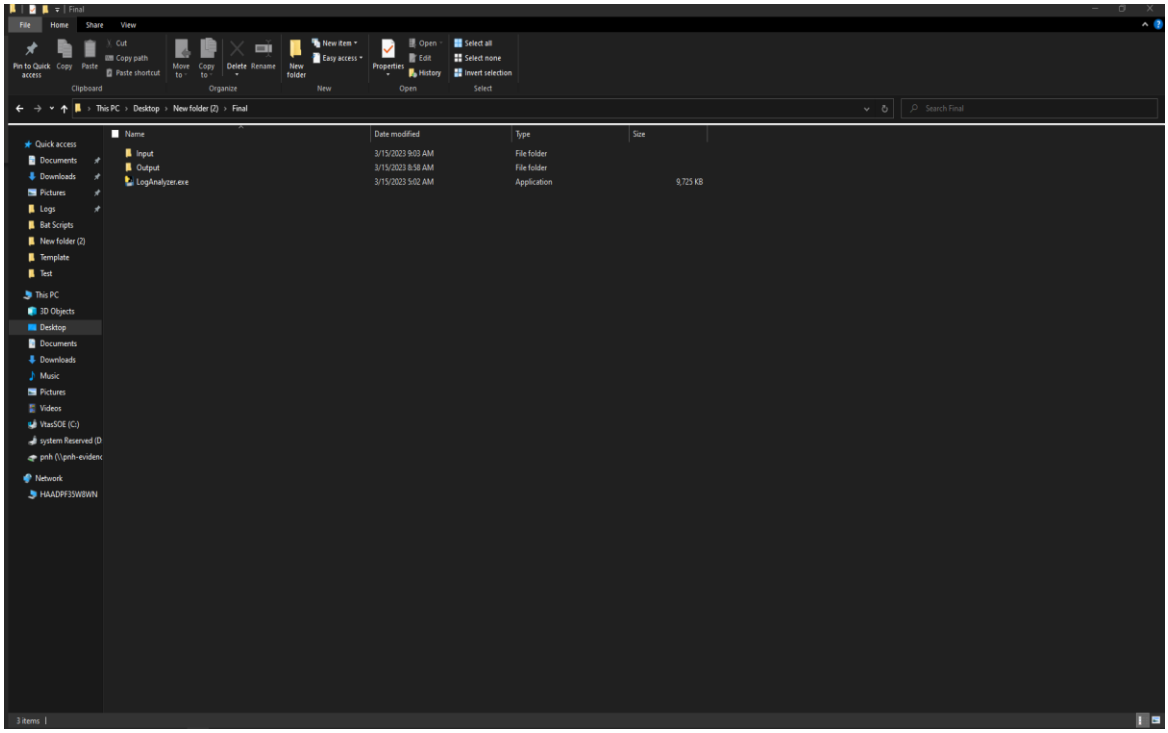
9. Create two button widgets: one for selecting the input file, and another for selecting the output file. Assign the respective functions "select_input_file" and "select_output_file" to them, and pack them to the main window with some padding.

10. Create two entry widgets with width 50 for displaying the paths of the input and output files, and pack them to the main window with some padding.

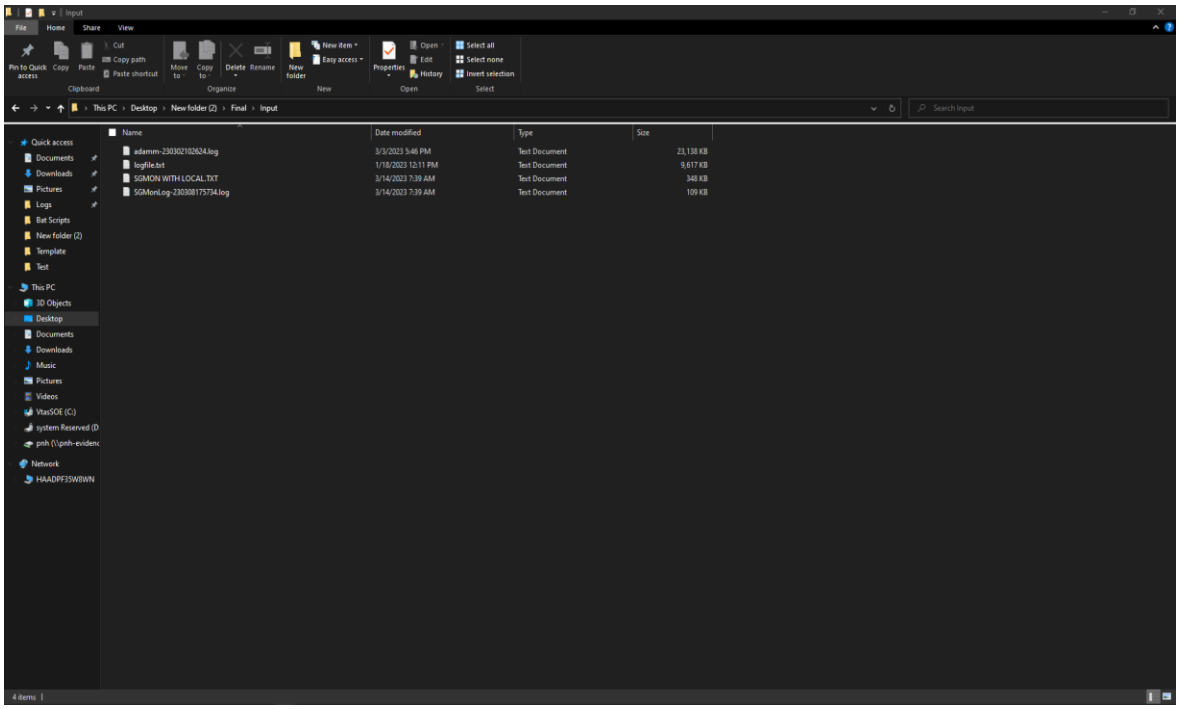
11. Create a button widget with the text "Generate Report" and assign the "generate_report_action" function to it. Pack it to the main window with some padding.

12. Start the main event loop using the "root.mainloop()" method, which waits for user input and responds to events such as button clicks.

How the executable works:

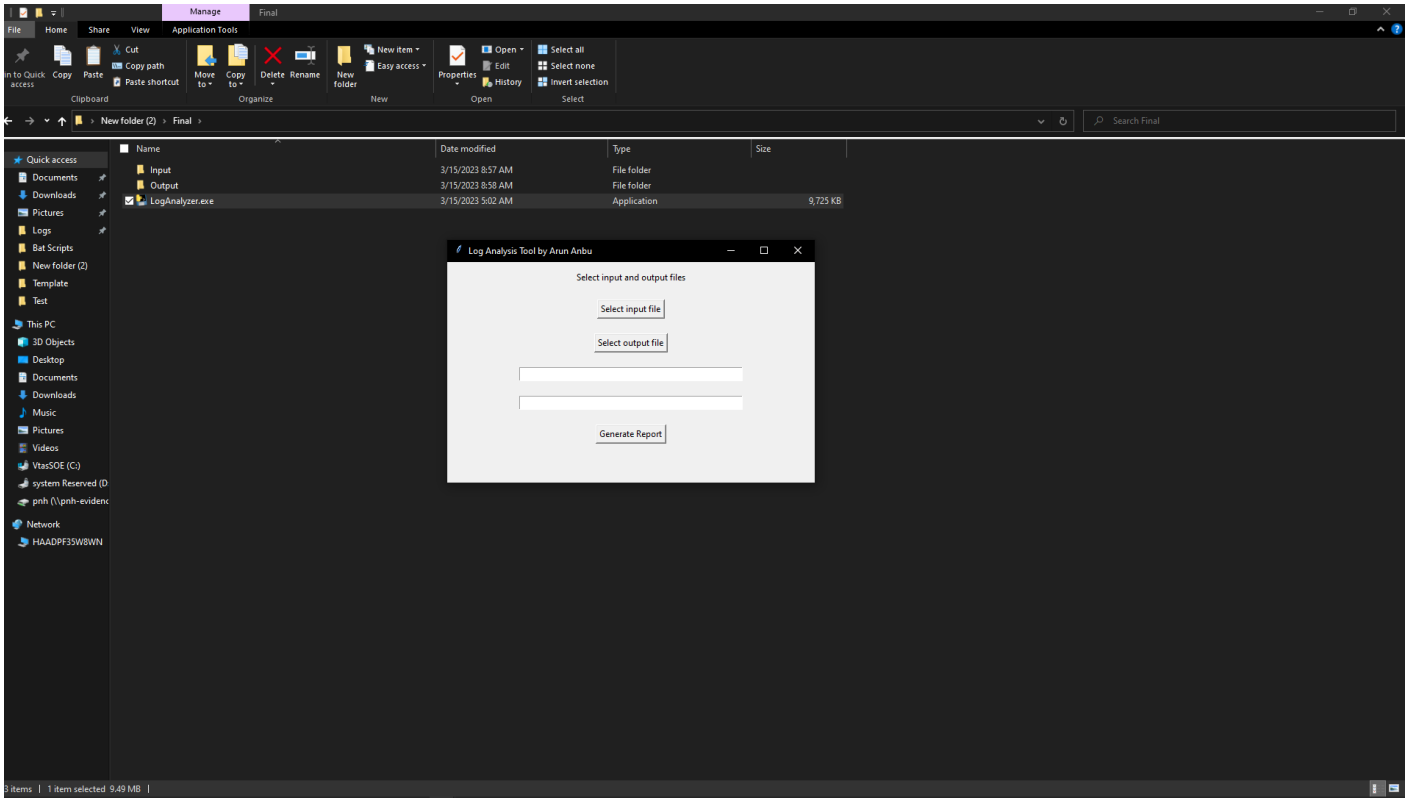


The above image shows the Exe file and the input and output folders created for examples.

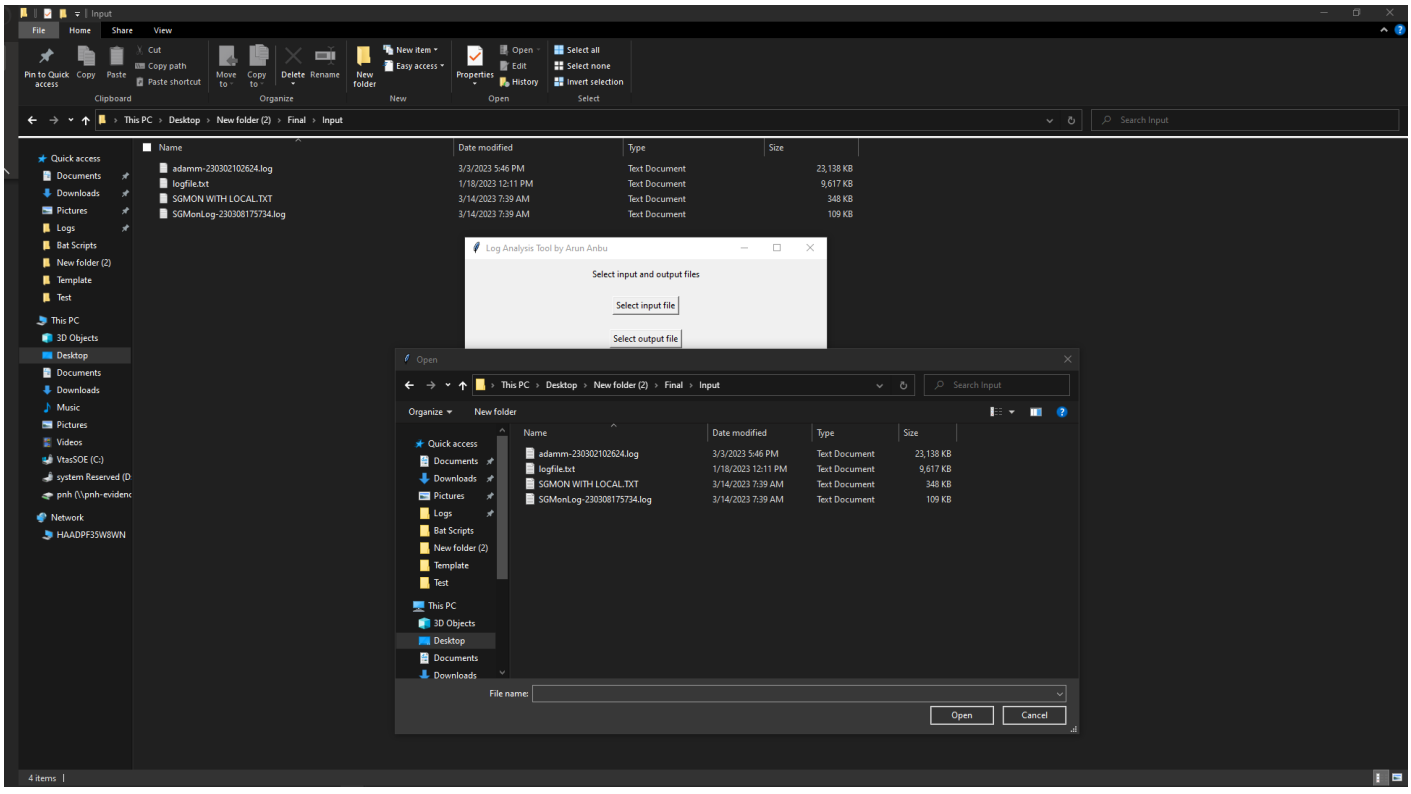


Copied the logs into the input file for accessing the logs required to be used as inputs. This can be anywhere in the PC just for clarity copied into a specific folder.

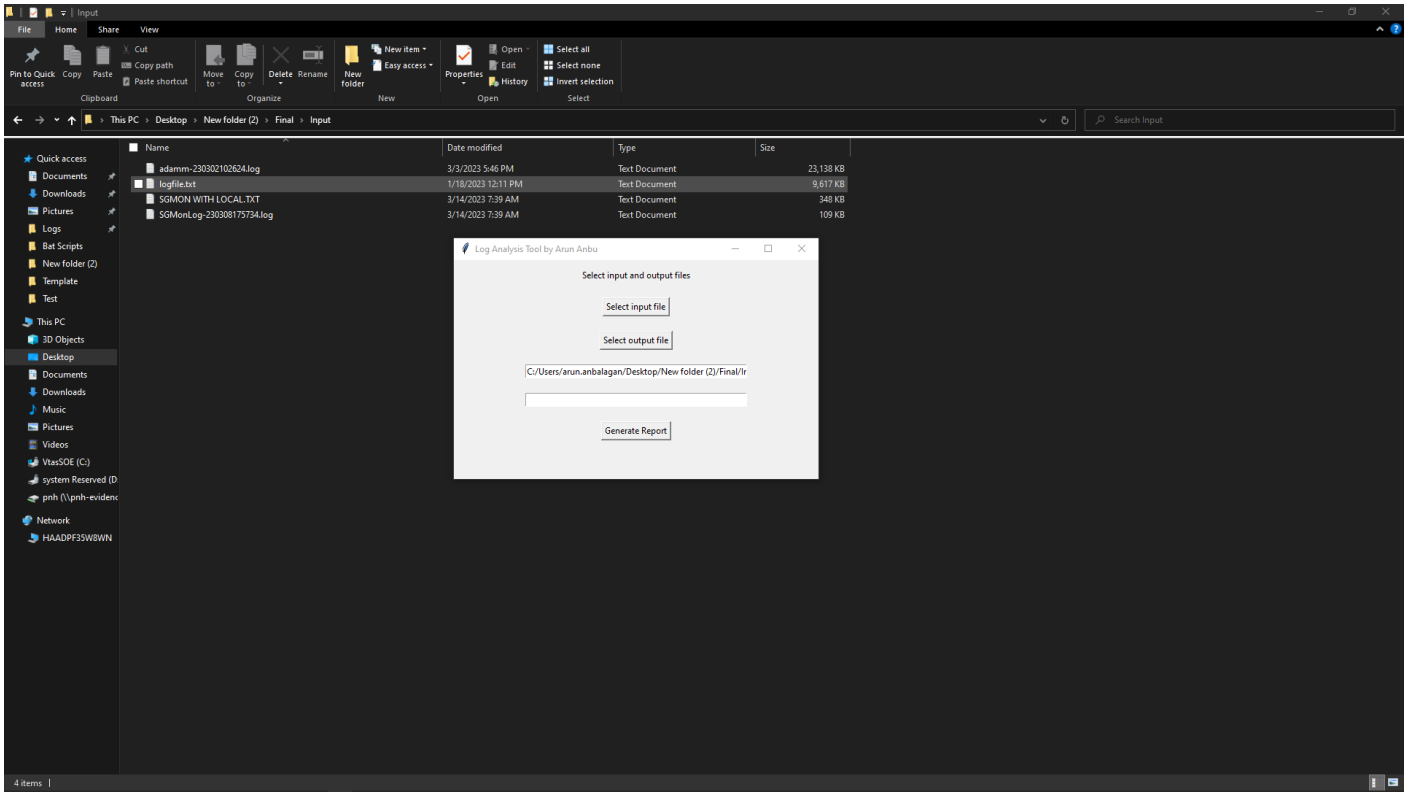
Copied logs for SGMON & Adamm logs for testing purposes.



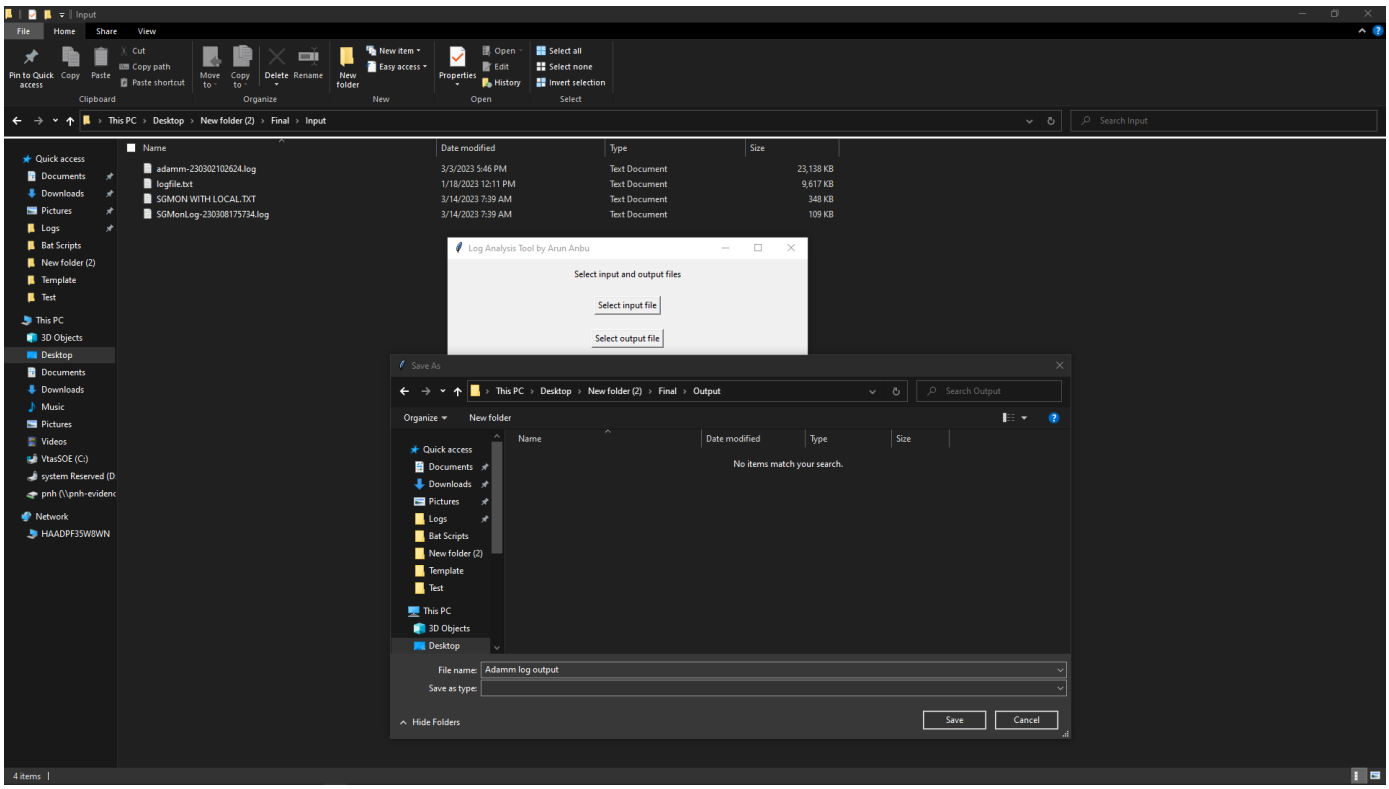
The above image shows the log analyzer exe file while running, it has three options simple interface, one to input the log file, second to save the output generated, and third to generate the report as selected.



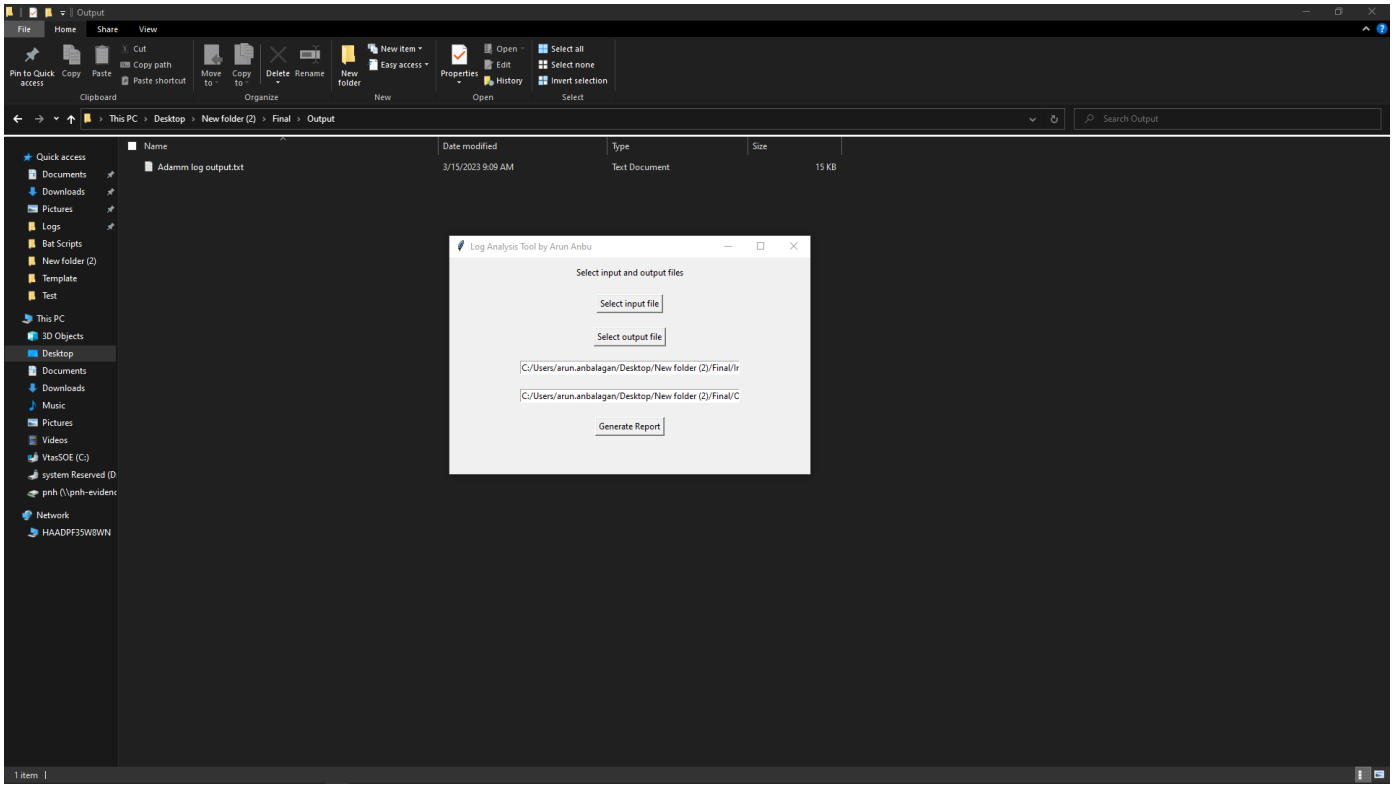
The above image refers to the input folder and selects the log file that needs to be analyzed. Here selected the adamm log for an example run.



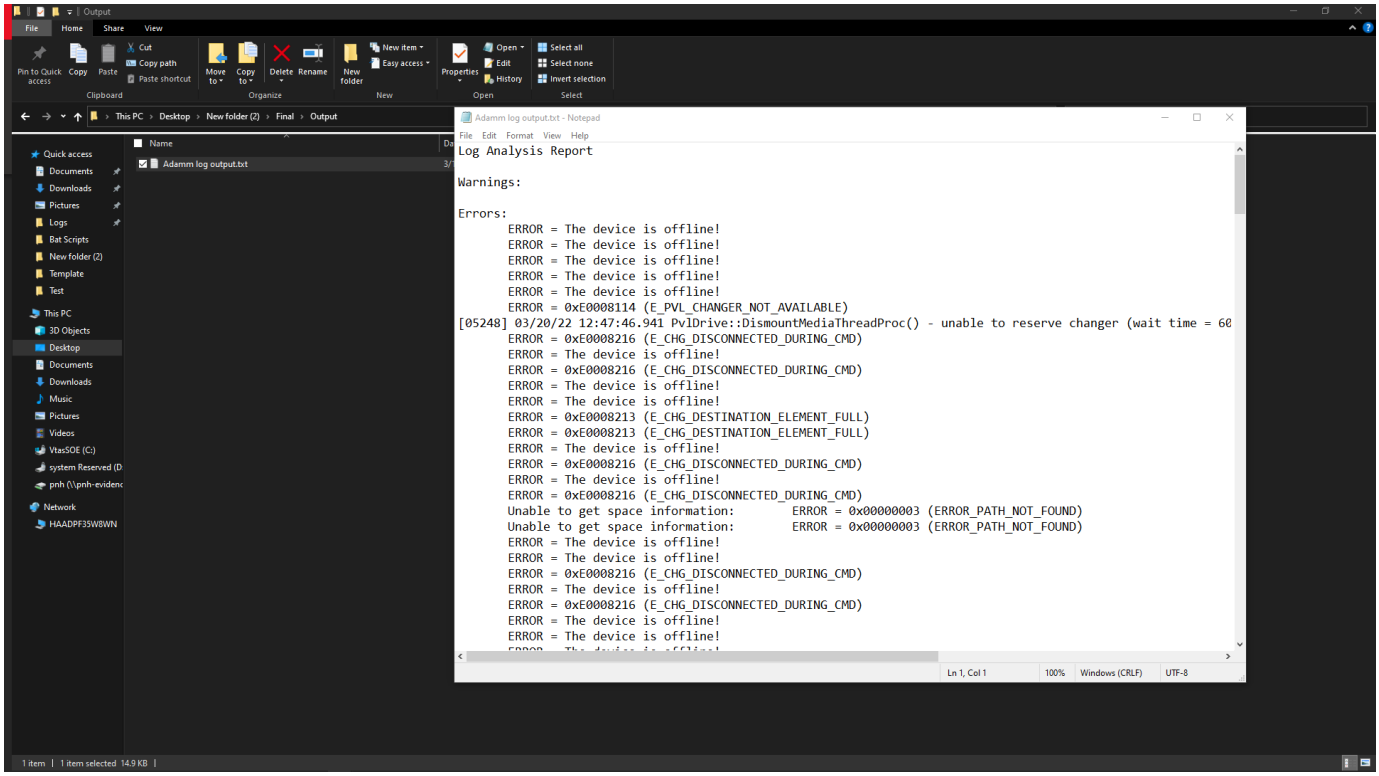
Once after selecting the input log file need to select the output file name and location.



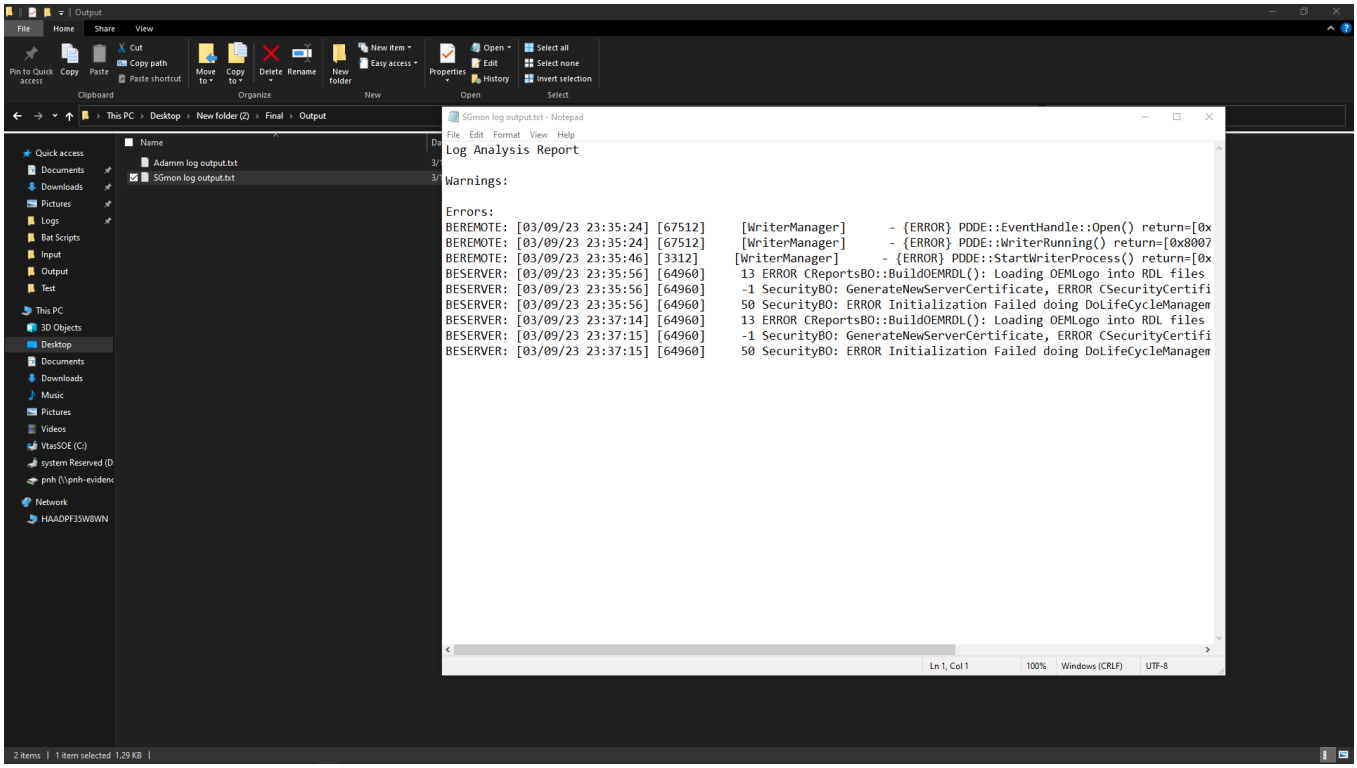
Once the output is named and the location is selected, we can click on save.



Now the generated report can be clicked, and the output file will be created with the logs analyzed only for the error and warning messages.



The output of the file will be similar to the above image which would filter the errors alone and the same can be cross-verified with the time stamp on the collected job logs to identify the issues. In this image, the reference is for the adamm log errors.



The above image shows the output file for the SGMon log.

_____ Thank you _____