Log collector test Ferenc Demjanics

Task1 testing stability in 4 hours

First, we need to define which OS was given and ignore other OS functionalities.
I would suggest manual testing and verify if the results are matching expectations.

1 – GUI structural test

- test if the app starts stable (trigger 4 times) in admin mode
- verify that it does not start in safe mode (trigger once)
- check all the checkboxes working
- check select/deselect tool (log collection profile) function
- verify the "logs age" field's default value on start
- check the available options of log age limit
- verify the "log collection mode" default value on start
- check the available options of log collection mode
- verify the default value of "archive path"
- check the manual edit of the archive path
- check the copy(cut) paste option for the archive path
- check the path search button
- verify the new archive path
- visually check the log window and its contents
- no need to verify other OS versions in this run

2 - Functional test

- verify the collect button function (select one artifact only) (scripted in "quicktest1" below)
- get or prepare testing data (example logs)
- prepare a template for the output expectation (using the example logs)
- select all the artifact to collect
- verify the output location
- verify the output (compare with the template)
- visually check the log window

3 - Design test

- visually check the window sizes
- try all the scroll bars and drop downs
- verify that all the data are visible
- make sure that the OS navigation buttons are there (minimize, maximize, close)
- visually check the title bar for correct name and icon

4 - Negative test

Does a warning message appear if you:

- deselect all checkboxes and run collection
- select all checkboxes then run and cancel the collection immediately
- restrict access to some testing data(example logs)

Task2 Design automated test

After a short research I have selected Python and “pyautogui” library as a testing environment.

Test aspects

Same as defined in manual testing above, extended for all functions, notifications and features.

Testing methodology

This solution is using pre-captured screenshots for its function. The script defines the center coordinates on screen of a similar image, then emulates cursor movement, mouse click and keypress to control the GUI behavior.

The package contains

- this documentation
- the recent version of the log collector application
- main script and other components of the test
- images required to navigate through screen during the test
- a capture of an example run in gif

Execution notes and conditions

Make sure to extract all components to one folder
The main script file is “quicktest1.py”
I suggest using IDLE (Python 3.8 32-bit) to execute

This script needs to be run as administrator
It requires Python 3.8 and “pyautogui” lib to be installed
The log messages and the result are printed to the console.

The behavior of “quicktest1”

- it starts the log collector application
- finds and selects Collection profile as “None”
- selects the first artifact to be collected
- starts the log collection by using the Collect button
- handles the overwrite check if needed
- waits for successful log collection
- handles the success notification
- closes the log collector application
- shares the test result (fail in timeout, pass on finished archive collection)

Further needed

Further test scenarios to be scripted using the same methodology and re-using the components/modules from “quicktest1”. The tasks should be similar to the manual testing requirements, but extended for the additional functions, notifications and features.

Log the result to a file or database directly instead of the console.

Verify if the result archive does actually exist, on overwrite prompt and at the end of collection.

Adapt the testing solution for the other supported OS versions.