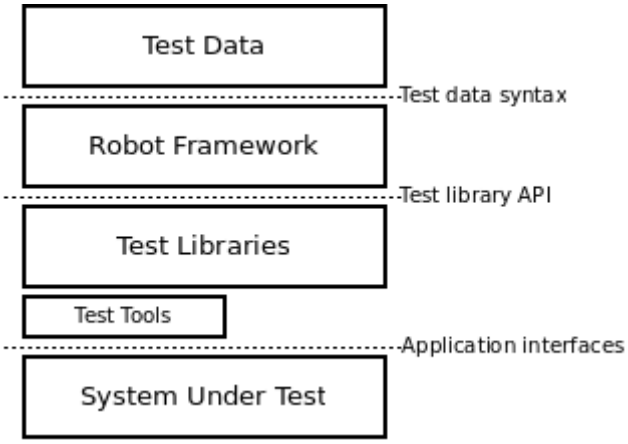
Web automation using Robot Framework with Python and Selenium Library

High-level architecture:

Robot Framework is a generic, application and technology independent framework. It has a highly modular architecture illustrated in the diagram below.



Robot Framework architecture

Here are step-by-step instructions for setting up web automation using Robot Framework with Python and Selenium Library:

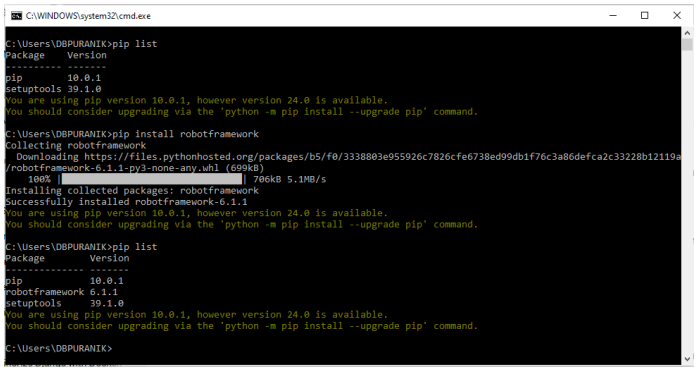
Install Python:

If you haven't already installed Python, download and install it from the official website: https://www.python.org/downloads/.

Install Robot Framework:

Once Python is installed, open a command prompt and execute the following command:

pip install robotframework



Install Selenium Library:

SeleniumLibrary is a web testing library for Robot Framework. Install it using pip:

pip install robotframework-seleniumlibrary

Set Up an Editor:

Choose an editor for writing Robot Framework scripts. Options include PyCharm, Eclipse, Sublime Text, Notepad++, Visual Studio, etc. For this tutorial, we'll use Notepad++.

Download ChromeDriver:

Download the ChromeDriver executable from the official ChromeDriver website: https://sites.google.com/a/chromium.org/chromedriver/.

Make sure to download the appropriate version compatible with your Chrome browser.

Create a New Directory:

Create a new directory to store your Robot Framework test files and place a ChromeDriver executable in the Scripts folder of your Python installed path.

Write Your Test Script:

Open Notepad++ (or your chosen editor) and create a new file with a **.robot** extension.

Write your test script using Robot Framework syntax.

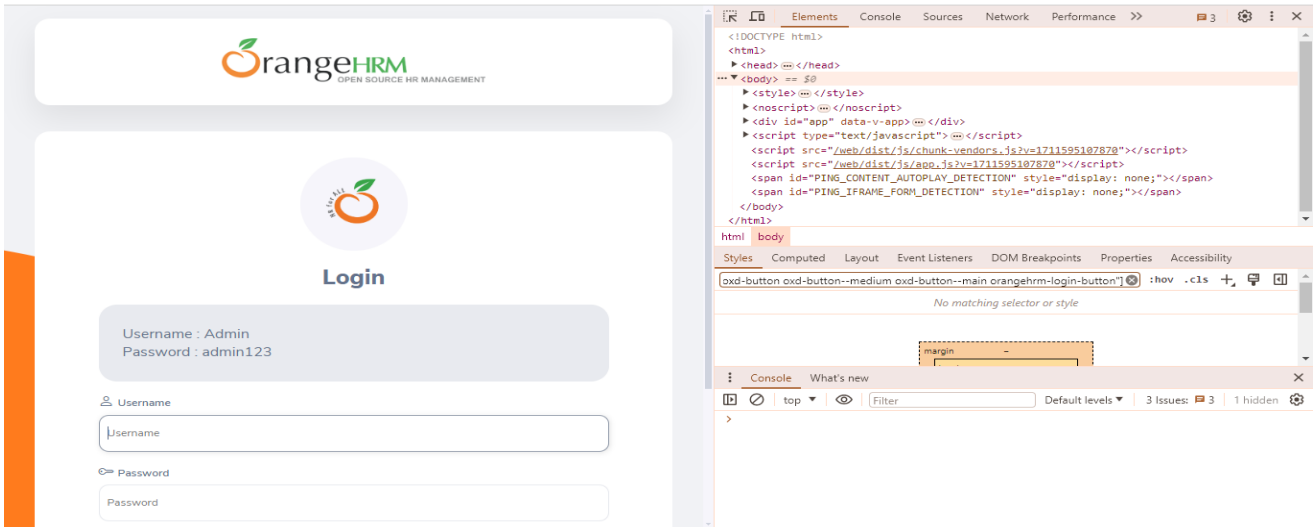
Keyword Document Reference: https://robotframework.org/SeleniumLibrary/SeleniumLibrary.html

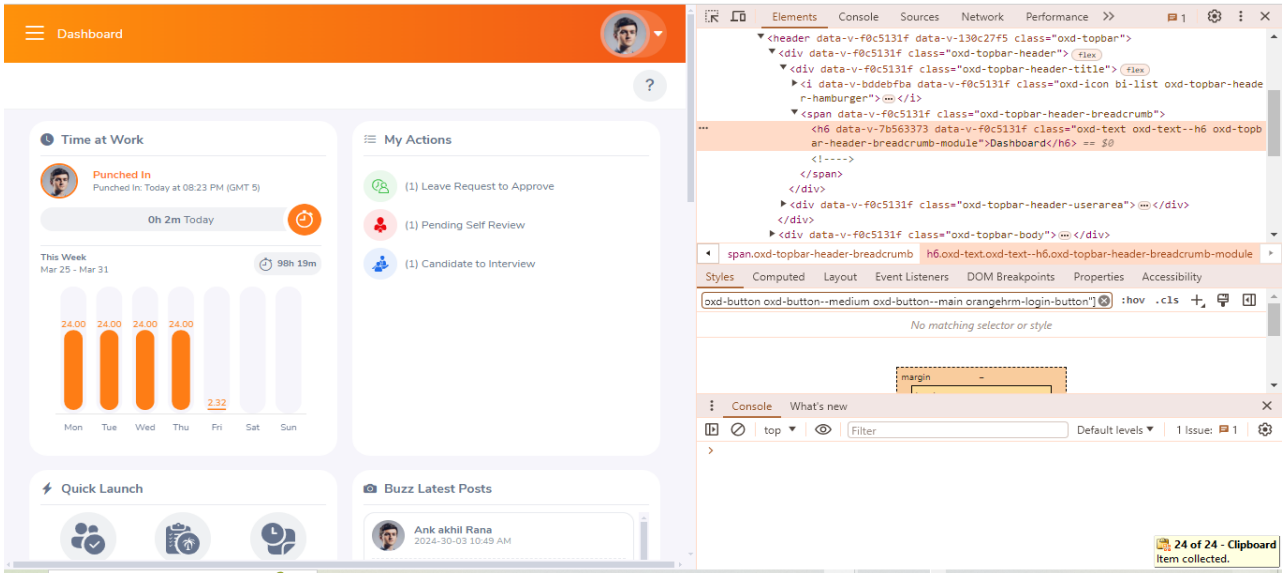
For example:

Login.robot | Logout.robot

```
${USERNAME}    Admin
${PASSWORD}    admin123

*** Keywords ***
Enter USERNAME
        Input Text    css=input[name="username"]    ${USERNAME}
Enter PASSWORD
        Input Text    css=input[name="password"]    ${PASSWORD}
Click on Login Button
        Click Element    css=.orangehrm-login-button
Verify Dashboard for user
        Wait Until Page Contains Element    css=h6[data-v-7b563373][data-v-f0c5131f]    timeout=10s
        ${dashboard_header_text} =    Get Text    css=h6[data-v-7b563373][data-v-f0c5131f]
        Should Be Equal As Strings    ${dashboard_header_text}    Dashboard
Logout From the Application
        Click Element    css=img[data-v-bdd6d943][alt="profile picture"]
        Click Element    css=a[href="/web/index.php/auth/logout"]
        Sleep           3s
*** Test Cases ***
Login and Logout Test With Valid credentials
    [Tags]    Regression
    [Setup]    Open Browser    ${URL}    ${BROWSER}
    Maximize Browser Window
    Wait Until Page Contains Element    css=input[name="username"]    timeout=10s
    Enter USERNAME
    Enter PASSWORD
    Click on Login Button
    Verify Dashboard for user
    Logout From the Application
    [Teardown]  Close Browser
```





In this script:

- **Sections**:
    - **Settings**: Import libraries and define settings.
    - **Variables**: Define variables such as browser, URL, username, and password.
    - **Test Cases**: Define test cases along with setup, teardown, and actual test steps.
    - **Keywords**: Define custom keywords for reusable actions.
- **Documentation**: Use the **[Documentation]** setting to provide descriptive comments about test cases or sections.
- **Tags**: Use **[Tags]** to categorize test cases for easy filtering and organization.
- **Setup and Teardown**:
    - **[Setup]** is used to perform actions before executing test cases (e.g., opening the browser).
    - **[Teardown]** is used to perform clean up actions after executing test cases (e.g., closing the browser). Even if there is any failure it make sure to execute the command in the [Teardown]
- **Browser Maximization**: Use **Maximize Browser Window** to maximize the browser window for better visibility during test execution.
- **Assertions**: Use **Should Be Equal As Strings** to verify that the expected text matches the actual text on the page.

For using Headless Chrome, you can specify it in the **${BROWSER}** variable as **"headlessChrome"**.This runs the activity in the background and the execution speed will be high. Selenium screenshots are typically available after test execution if the selenium built in functions fail to do an action and it will capture and provide the detailed reports. Screenshots are not available if the failure is because of any other assertions.

Running in Command Line Interface(CLI):

Open the Command prompt in the path where your Robot scripts are placed.

To run a specific script -> robot Filename.robot

To run all the scripts (Executes sequentially) -> robot *.robot

To run the scripts which are starting from a specific word-> robot My*.robot

Running tests in parallel:

Go to https://pabot.org/ and click on Github link for more information.

Install pabot: pip install -U robotframework-pabot

Command-line options

pabot [--verbose|--testlevelsplit|--command .. --end-command|

--processes num|--pabotlib|--pabotlibhost host|--pabotlibport port|

--processtimeout num|

--shard i/n|

--artifacts extensions|--artifactsinsubfolders|

--resourcefile file|--argumentfile[num] file|--suitesfrom file]

[robot options] [path ...]

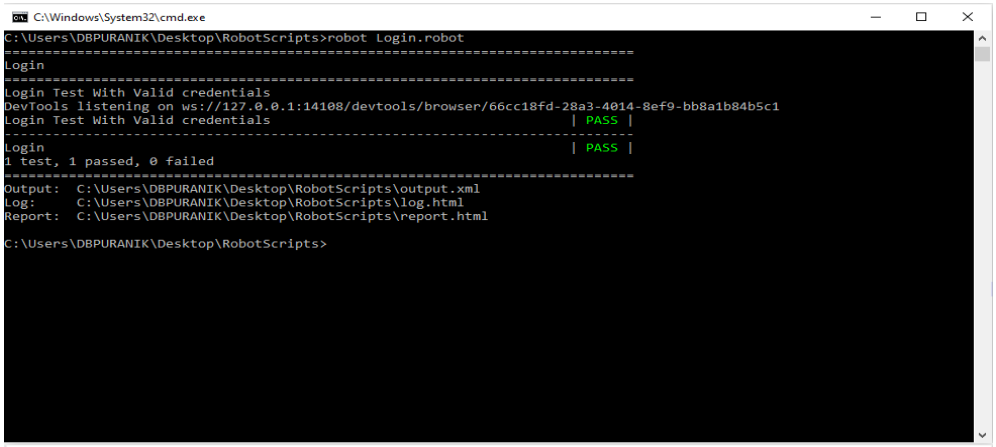
It supports all Robot Framework Command line operations: https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#command-line-options-for-post-processing-outputs

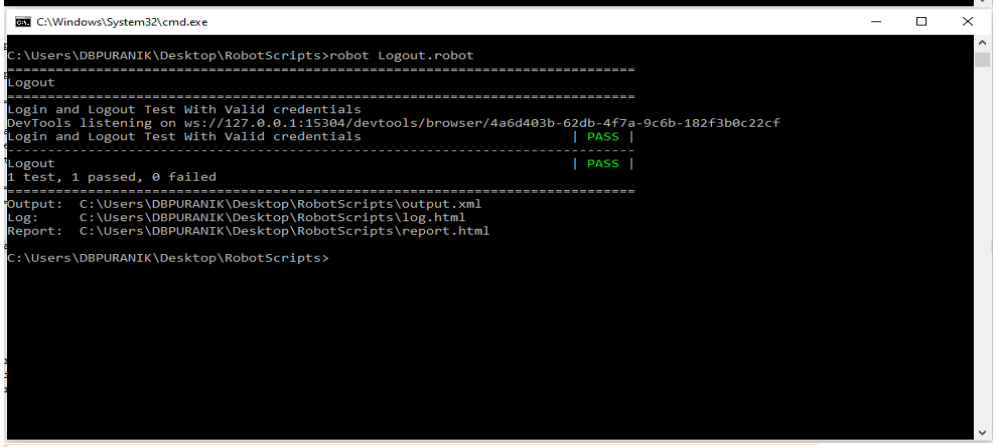
Example:

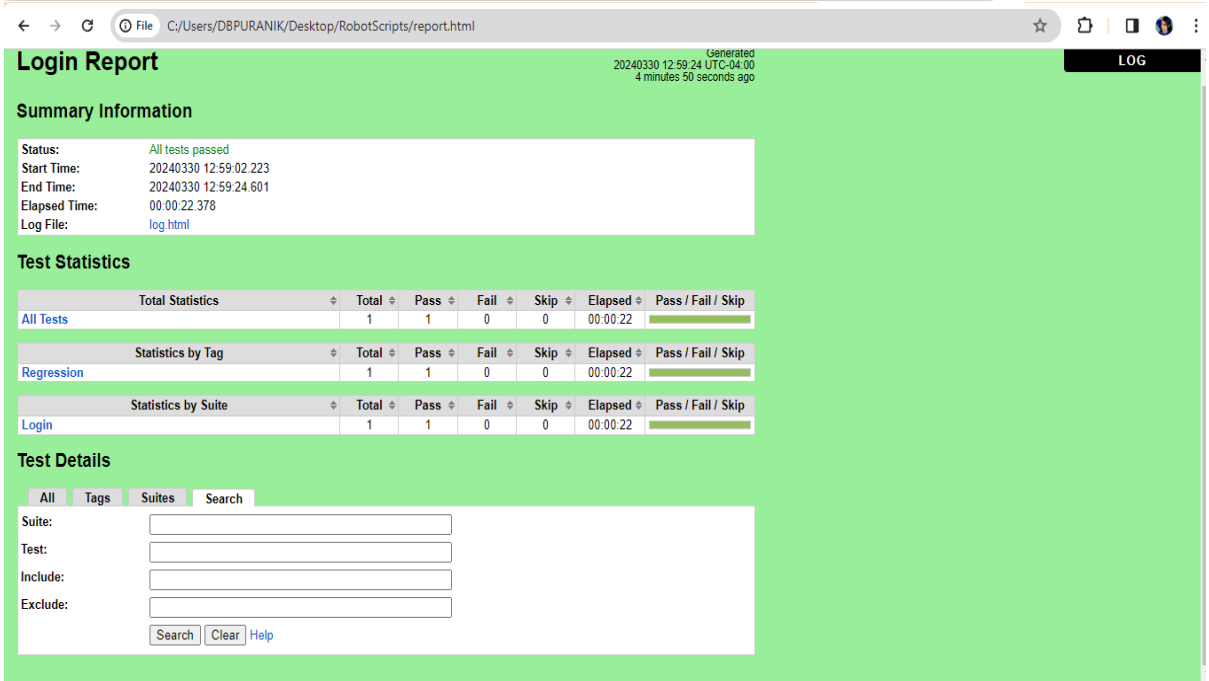
In command line type: pabot --processes 2 –outputdir Report *.robot

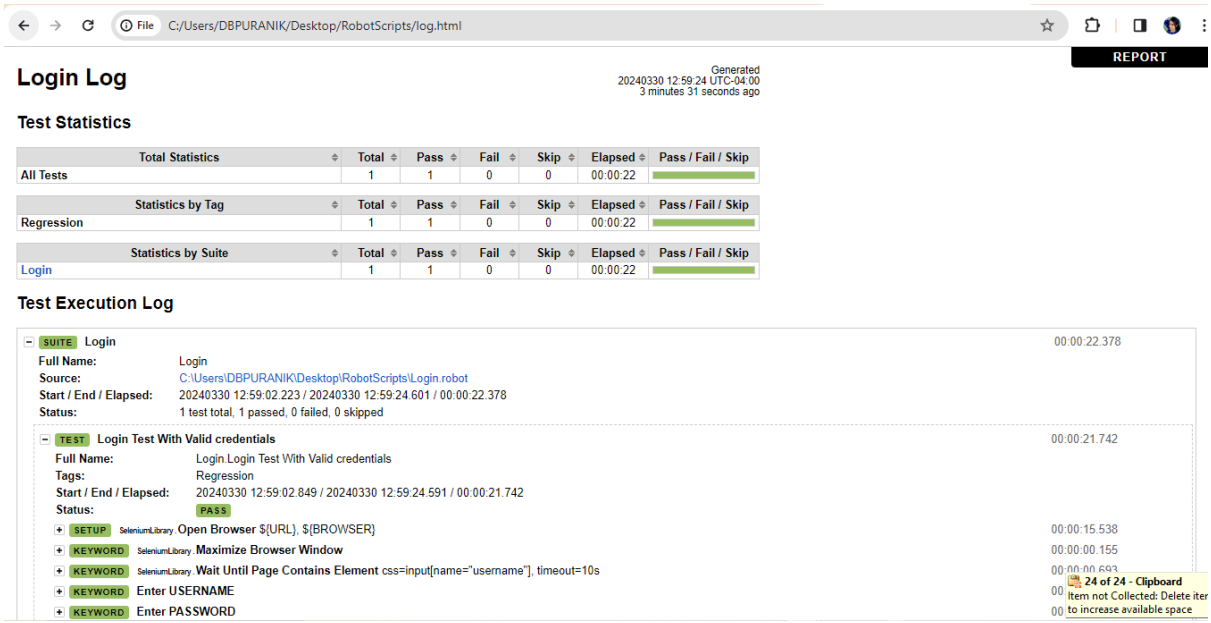
This command executes all the scripts in parallel using 2 processes and keep the consolidated reports in the Report folder.

Logs and Reports:

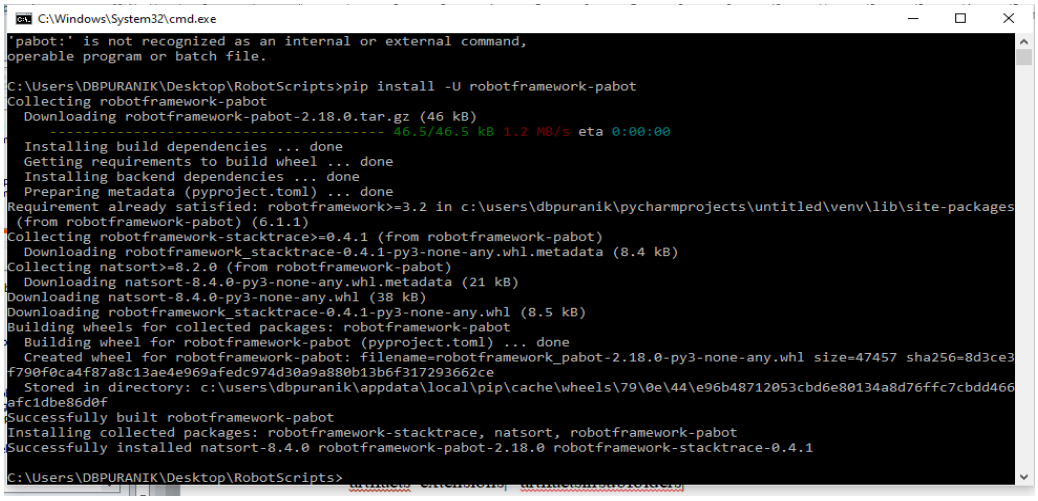


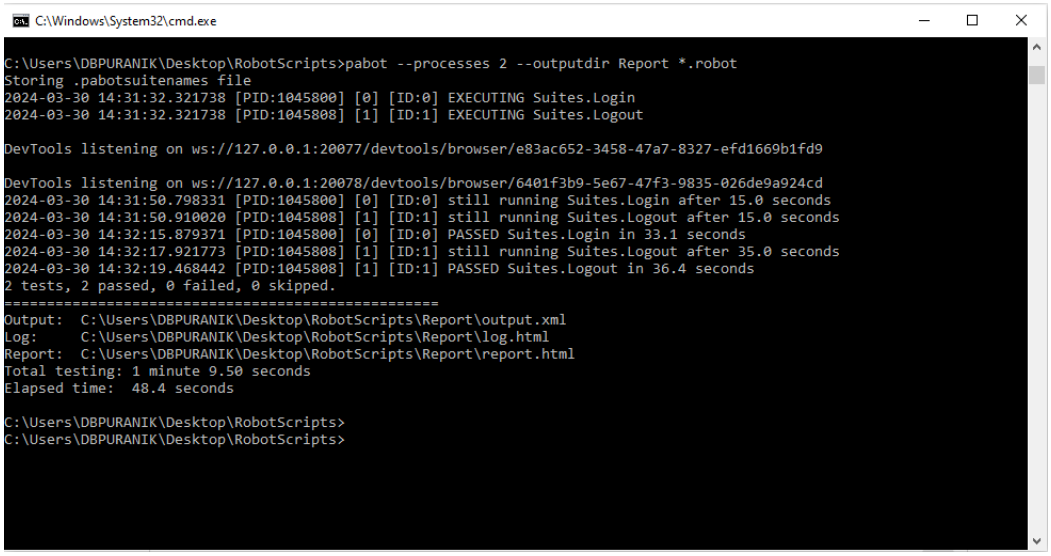


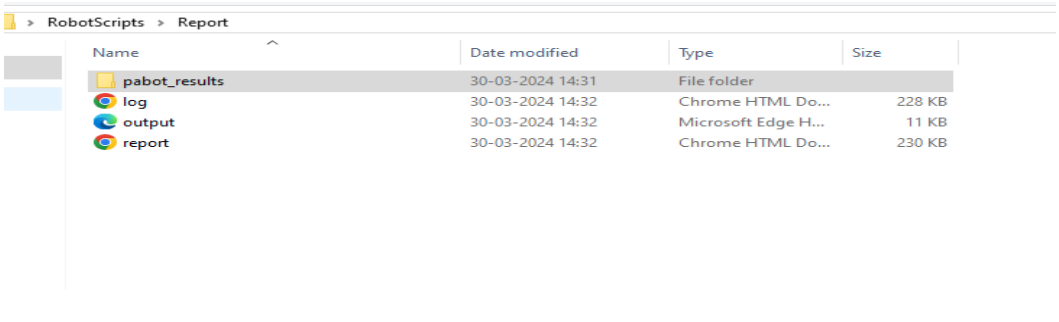




Parallel Execution Pabot Reports:







File C:/Users/DBPURANIK/Desktop/RobotScripts/Report/log.html

REPORT

Suites Log

Generated
20240330 14:32:20 UTC-04:00
1 minute 55 seconds ago

Test Statistics

Total Statistics	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
All Tests	2	2	0	0	00:01:13	

Statistics by Tag	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
Regression	2	2	0	0	00:01:13	

Statistics by Suite	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
Suites	2	2	0	0	00:00:48	
Suites . Login	1	1	0	0	00:00:36	
Suites . Logout	1	1	0	0	00:00:40	

Test Execution Log

SUITE Suites 00:00:47.909

Full Name: Suites
Documentation: Pabot result from 2 executions.
Start / End / Elapsed: 20240330 14:31:32.136 / 20240330 14:32:20.045 / 00:00:47.909
Status: 2 tests total, 2 passed, 0 failed, 0 skipped

SUITE Login 00:00:35.789

SUITE Logout 00:00:39.740

File C:/Users/DBPURANIK/Desktop/RobotScripts/Report/report.html

LOG

Suites Report

Generated
20240330 14:32:20 UTC-04:00
2 minutes 36 seconds ago

Summary Information

Status:	All tests passed
Documentation:	Pabot result from 2 executions.
Start Time:	20240330 14:31:32.136
End Time:	20240330 14:32:20.045
Elapsed Time:	00:00:47.909
Log File:	log.html

Test Statistics

Total Statistics	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
All Tests	2	2	0	0	00:01:13	

Statistics by Tag	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
Regression	2	2	0	0	00:01:13	

Statistics by Suite	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
Suites	2	2	0	0	00:00:48	
Suites . Login	1	1	0	0	00:00:36	
Suites . Logout	1	1	0	0	00:00:40	

Test Details

All | Tags | Suites | Search

Suite:
Test:
Include: