Framework Elements

1. **Input Data (XLS):** The testing process begins with the Input Data stored in an Excel spreadsheet (XLS format). This data comprises the test cases, input parameters, function keywords and any necessary information required to execute the tests.

2. **Pester (Test Runner):** Pester, the PowerShell testing framework, serves as the test runner. It orchestrates the test execution by reading the input data and invoking the corresponding test scripts. Pester is responsible for initiating the tests and managing the test flow.

3. **Test Execution:** During the Test Execution phase, Pester calls the Modular Function Library to interact with the APIs and perform the required actions as per the test cases defined in the Input Data.

4. **Modular Function Library:** The Modular Function Library is a collection of PowerShell functions tailored to carry out specific tasks such as API interactions (GET, POST, PUT, DELETE) and processing the results. This library provides modularity and reusability, allowing for efficient test script management.

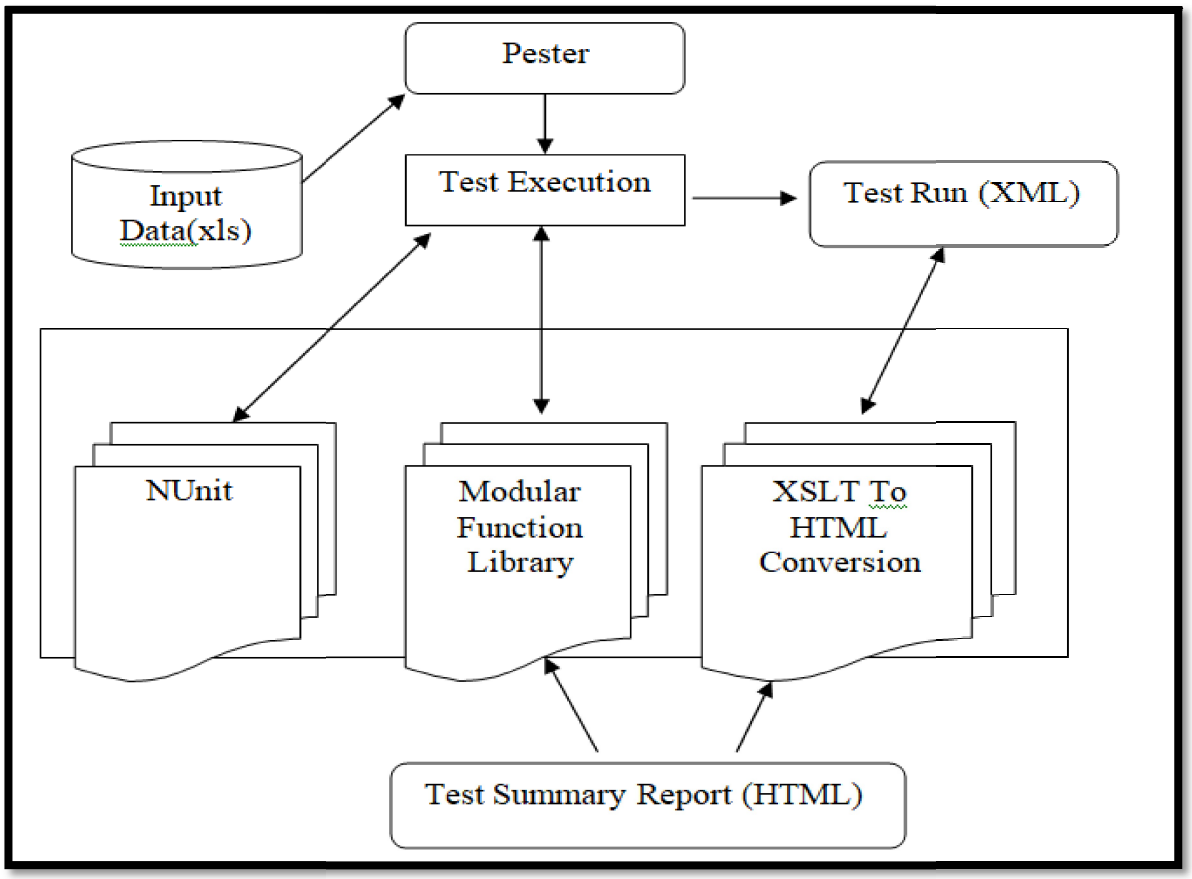
5. **NUnit:** NUnit is utilized for its result management capabilities. After the tests are executed, Pester generates an XML file with the test results in the NUnit format.

6. **Test Run (XML):** The XML file contains the outcomes of the test cases executed by Pester. It captures details such as which tests passed, which failed, and any exceptions or errors encountered.

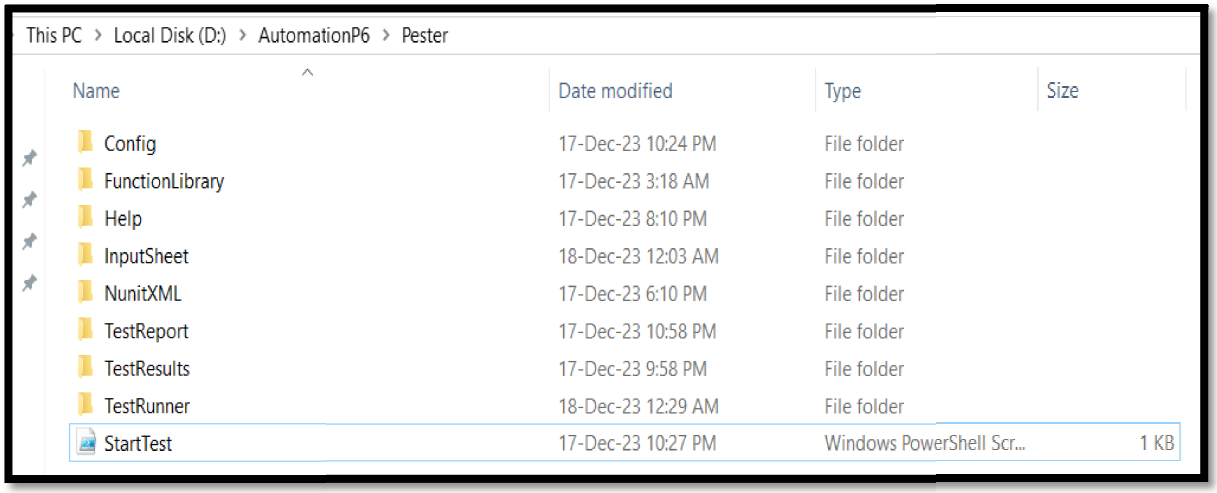
7. **XSLT to HTML Conversion:** The XSLT to HTML Conversion component takes the NUnit XML output and applies an XSLT stylesheet transformation. This process converts the XML data into a user-friendly HTML format.

8. Test Summary Report (HTML): The final output is the Test Summary Report in HTML format. This report provides a visual and detailed summary of the test results, including pass/fail status, test metadata, and failure details when applicable.

Framework Overview



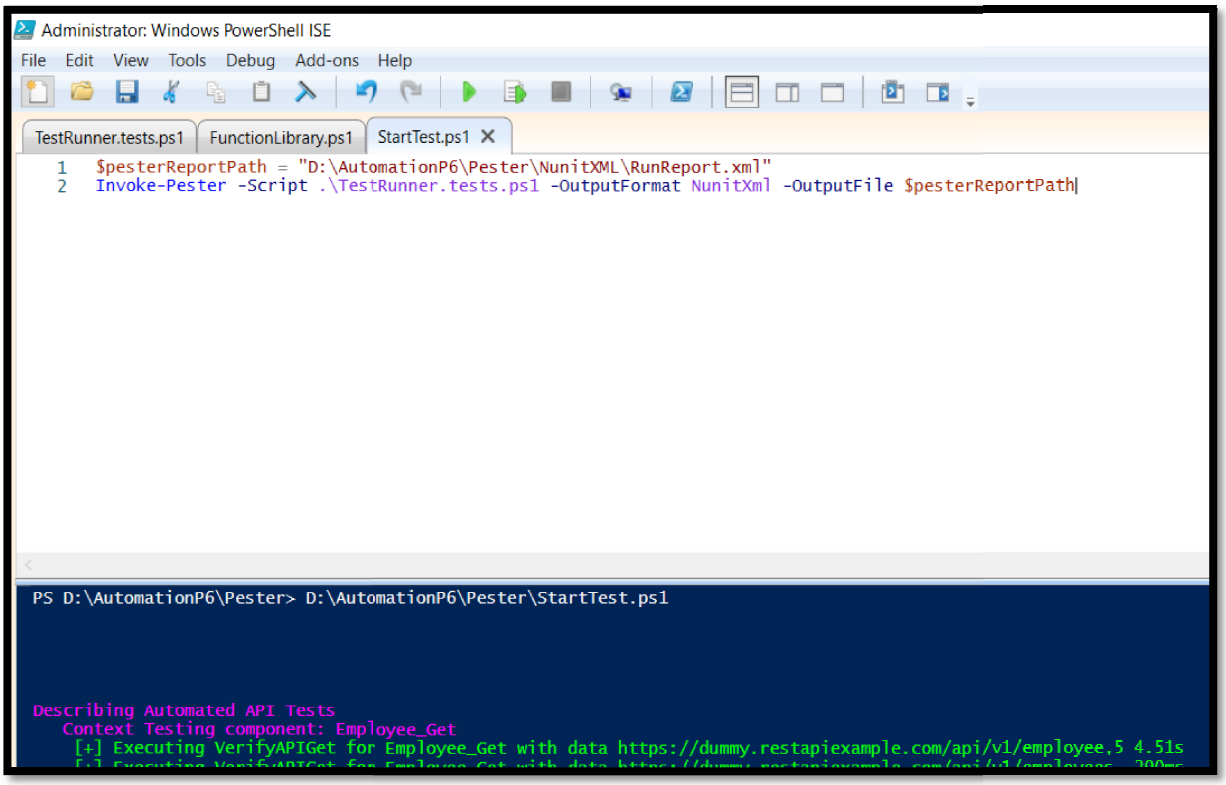
Folder Structure



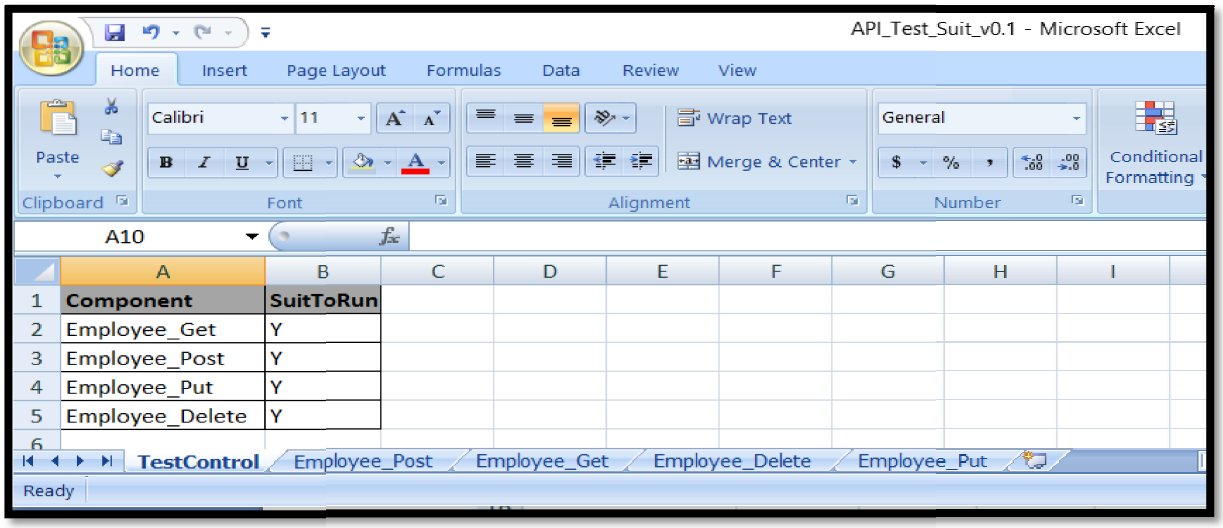
Test Execution

** I used (https://dummy.restapiexample.com/) in this example.

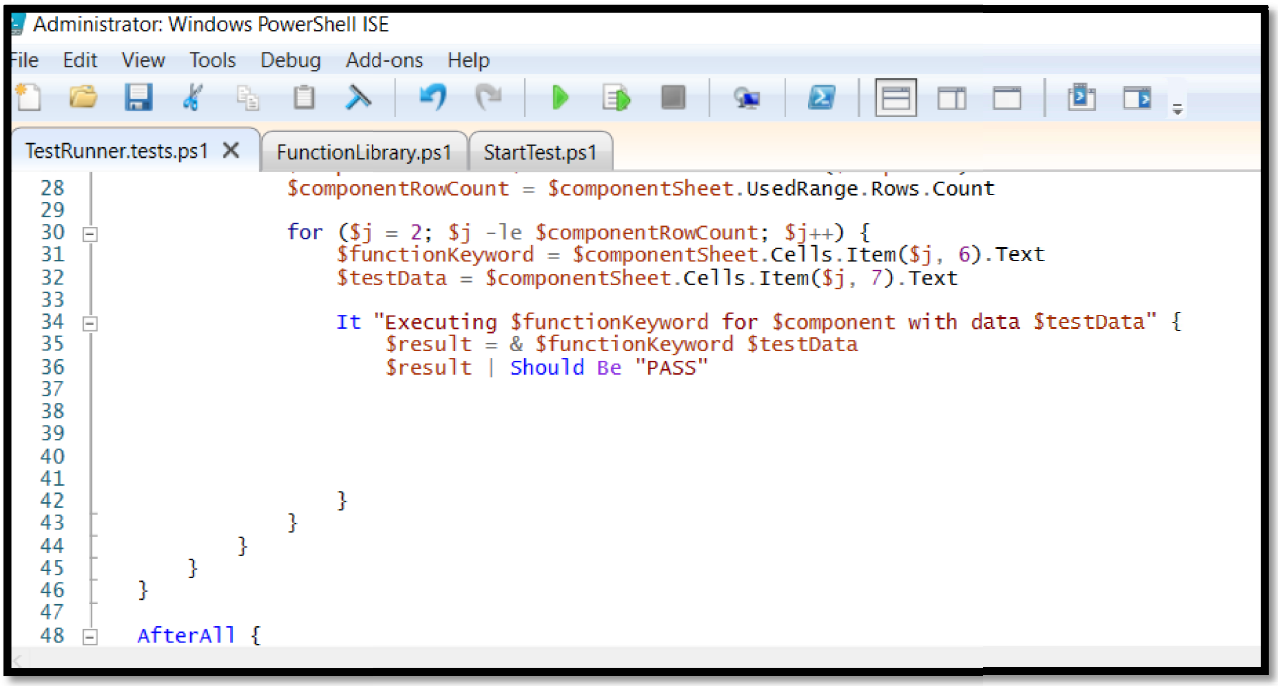
1. The test execution initiates with the PowerShell ISE invoking the Pester script.



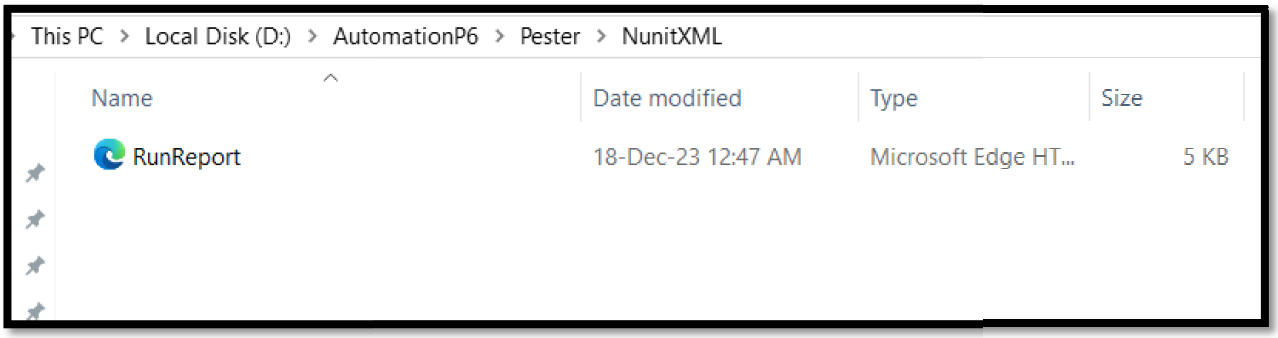
2. This process begins by reading from the Excel file, starting with the control sheet which lists available test suites and their corresponding execution flags (Y/N). The script selectively executes suites marked with 'Y', proceeding to the specific sheet of each suite to run the test cases.



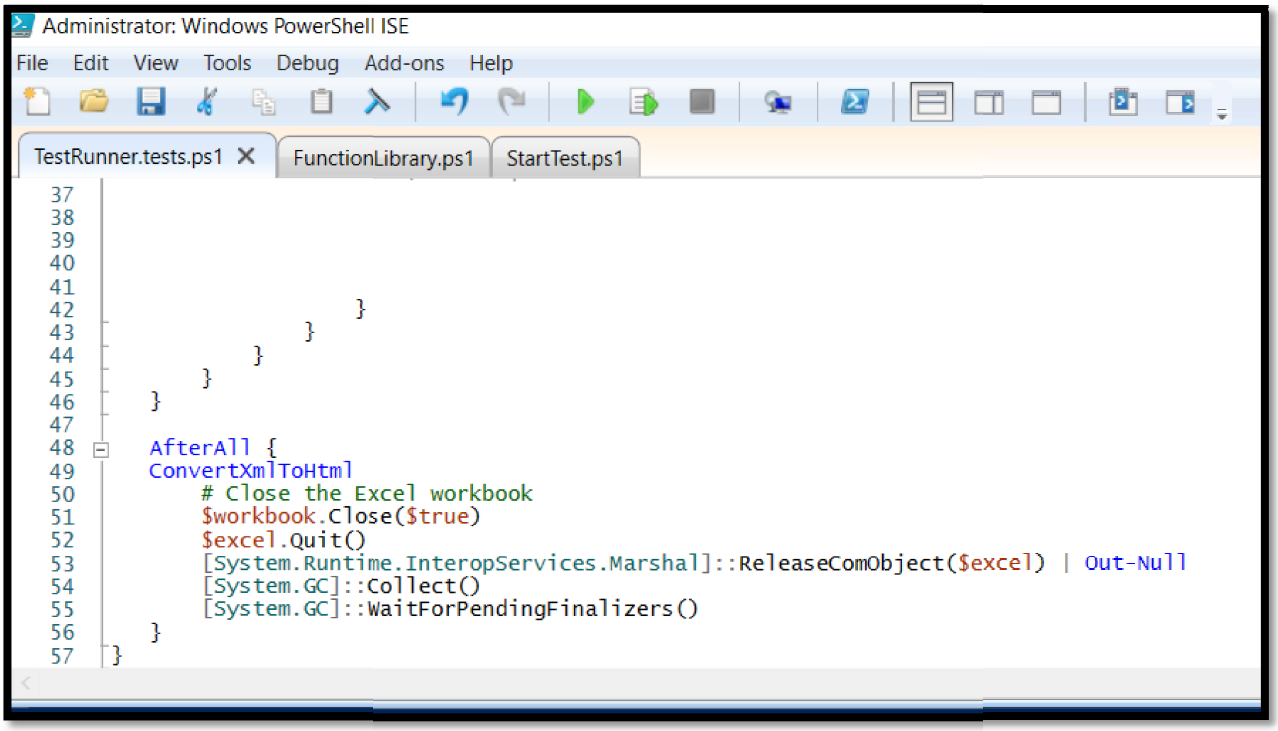
3. During suite execution, the '**It**' block within the Pester script is iteratively processed to validate specific conditions or behaviours in the test case against predefined expectations. This block contains assertions that must return true for the test case to pass. The results, whether pass or fail, contribute to the comprehensive NUnit XML report generated post-execution.

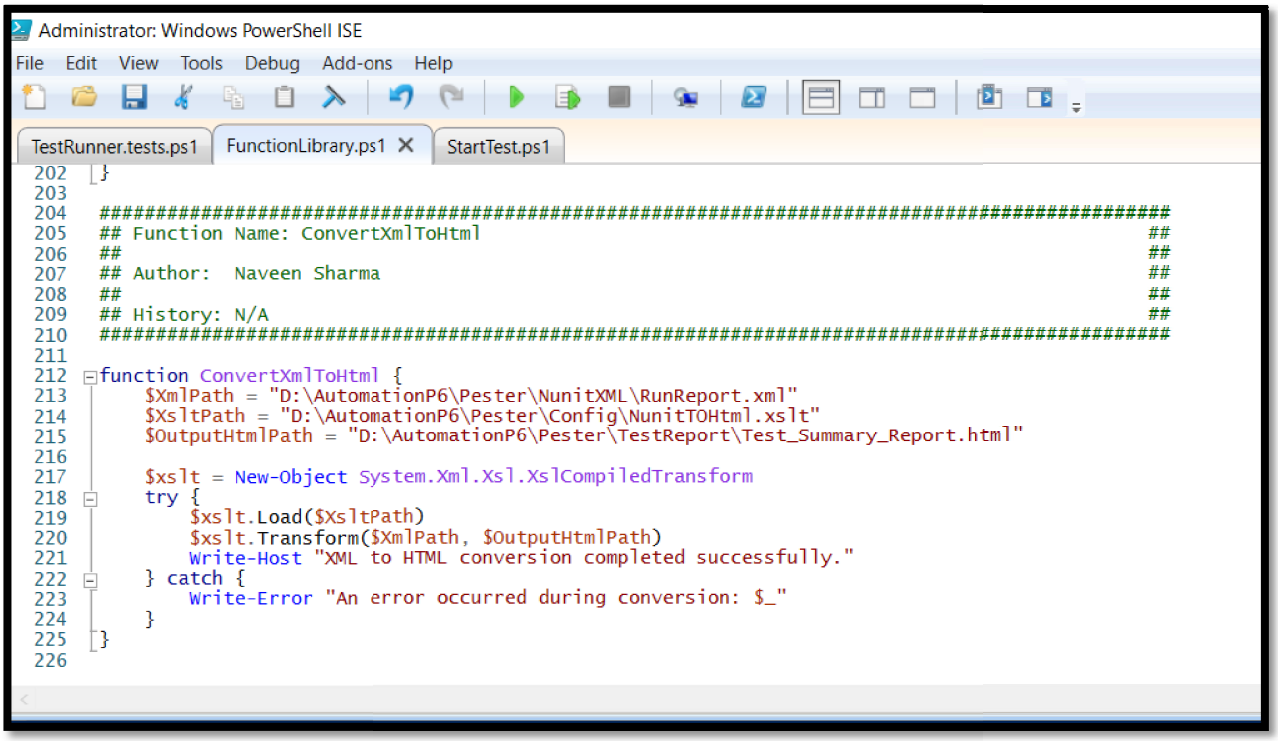


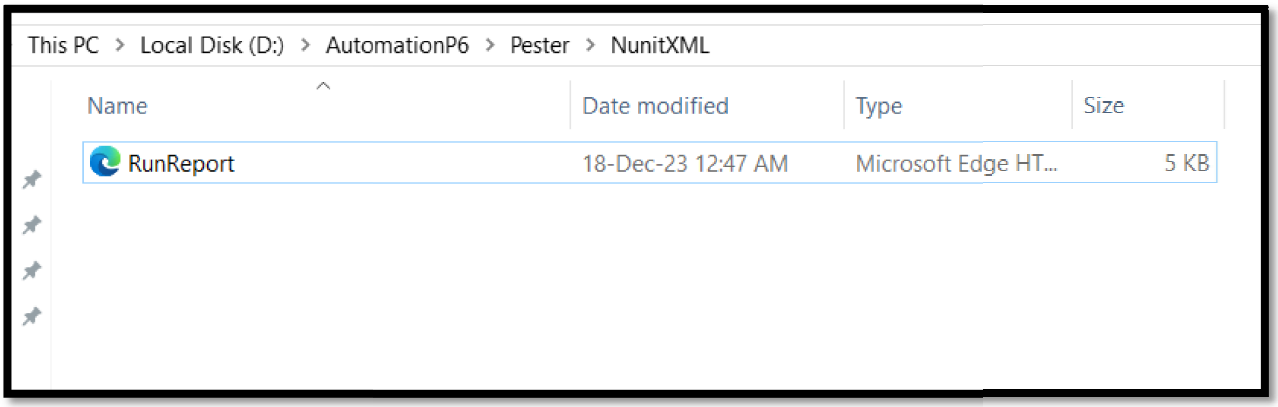
4. Upon completion of all suites, an NUnit XML report is generated in a designated folder.

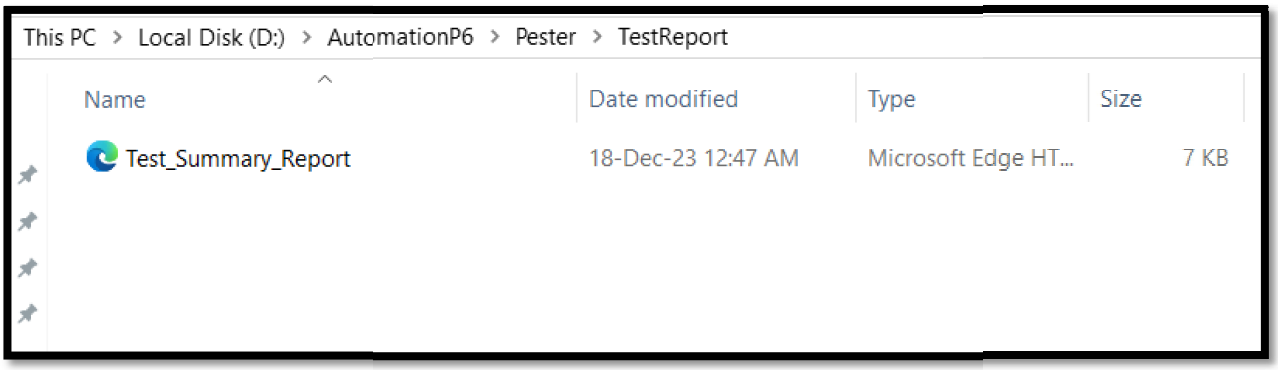


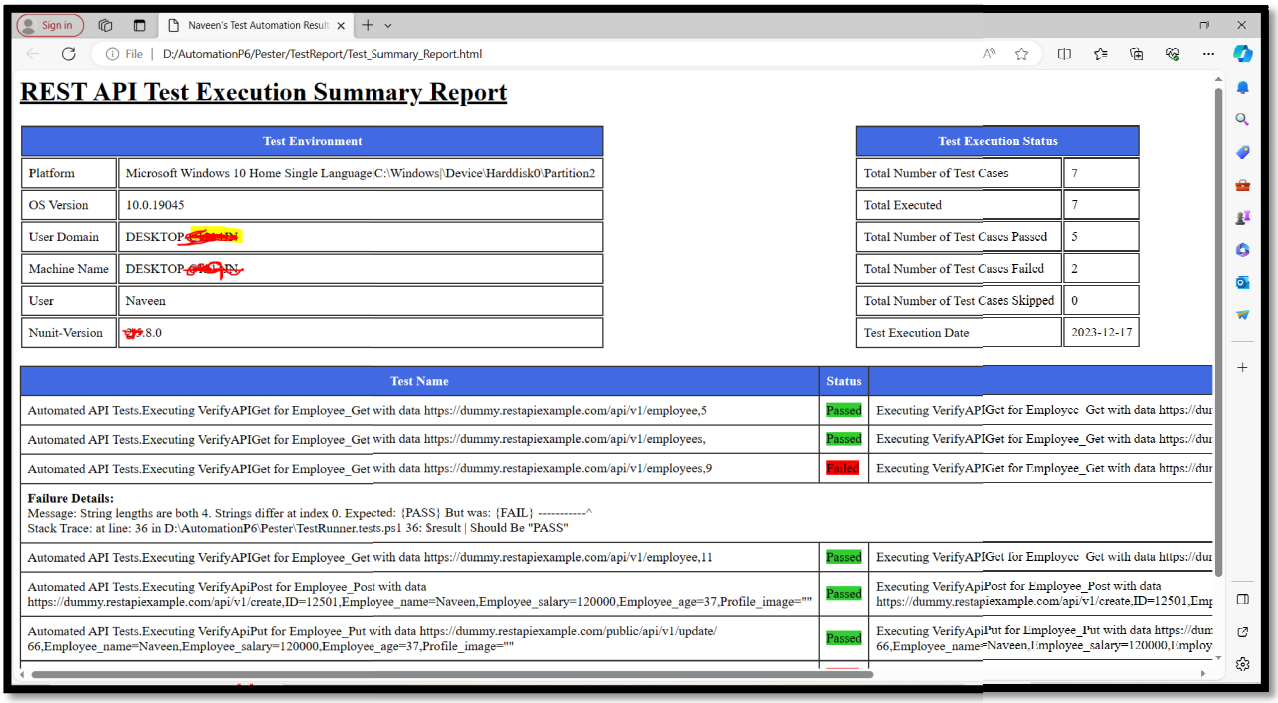
5. The **'AfterAll'** block within the script automatically converts this XML into a comprehensive HTML report stored in the test report folder.



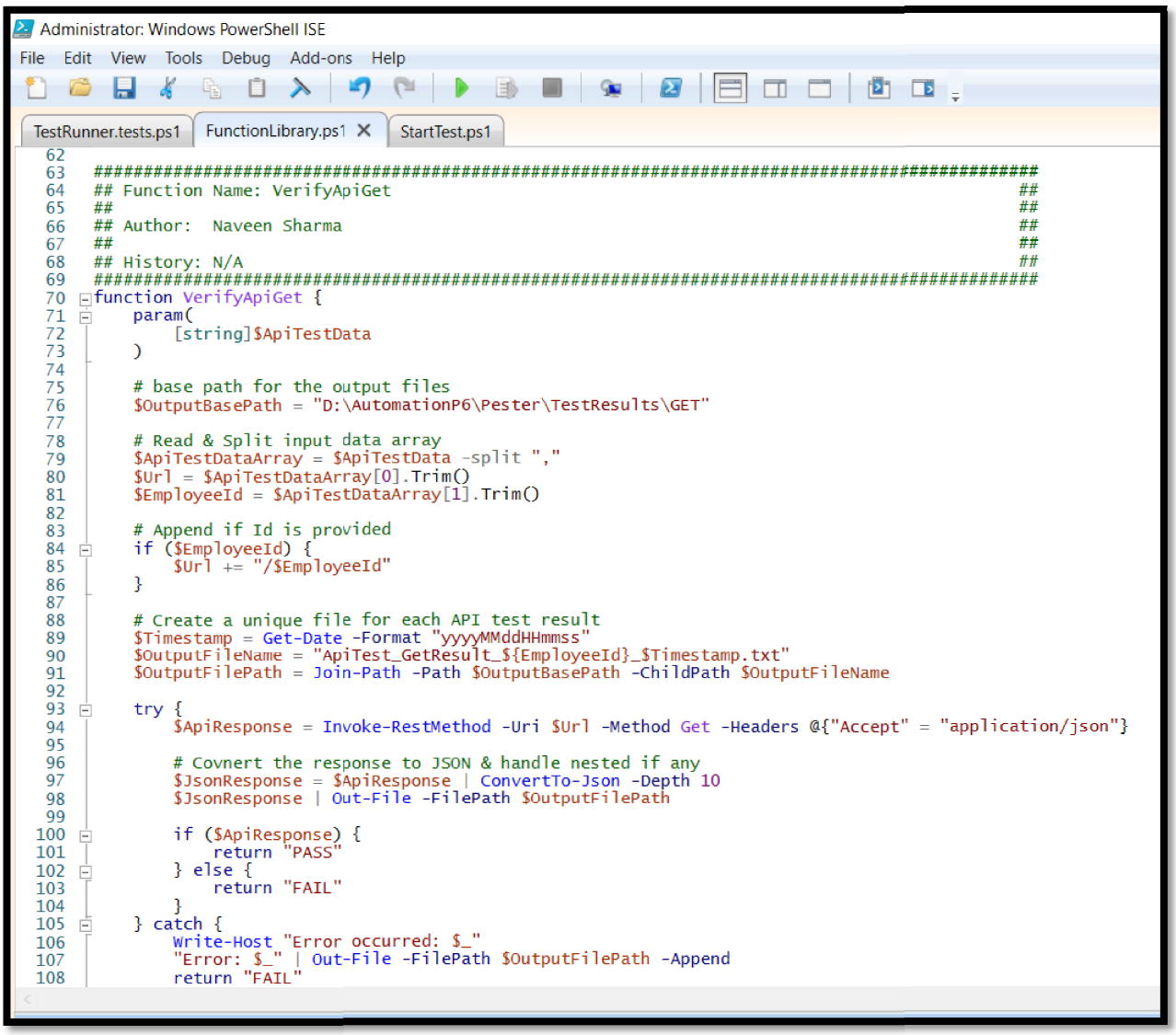


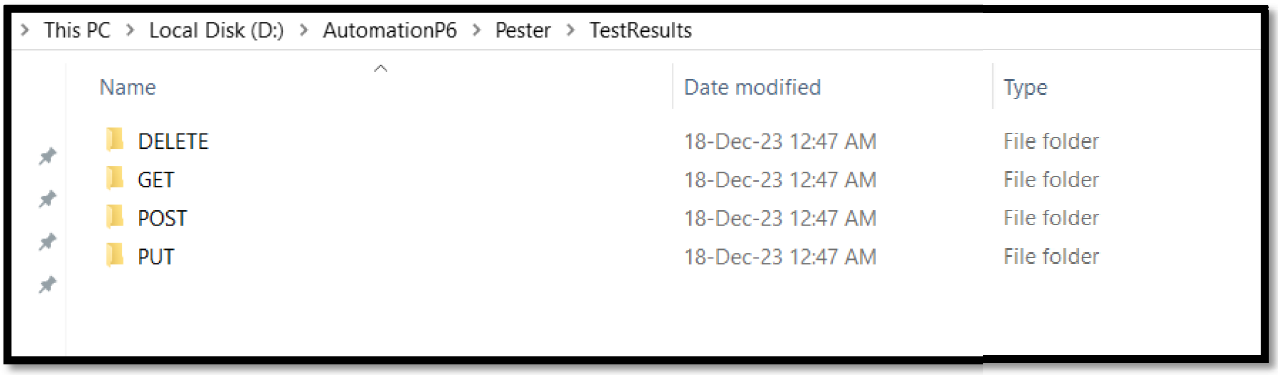

This PC > Local Disk (D:) > AutomationP6 > Pester > NunitXML
Name
Date modified
Type
Size
RunReport
18-Dec-23 12:47 AM
Microsoft Edge HT...
5 KB

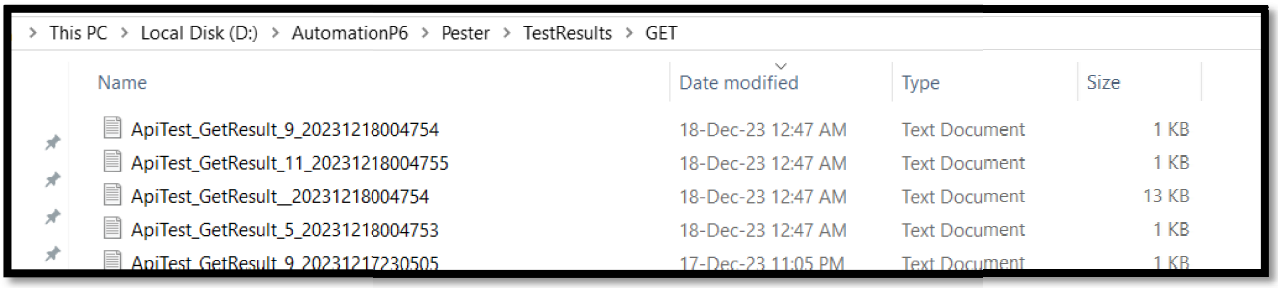

This PC > Local Disk (D:) > AutomationP6 > Pester > TestReport
Name
Date modified
Type
Size
Test_Summary_Report
18-Dec-23 12:47 AM
Microsoft Edge HT...
7 KB


Naveen's Test Automation Result
File | D:/AutomationP6/Pester/TestReport/Test_Summary_Report.html
REST API Test Execution Summary Report
Test Environment
Platform
Microsoft Windows 10 Home Single Language C:\Windows|\Device\Harddisk0\Partition2
OS Version
10.0.19045
User Domain
User
Naveen
Nunit-Version
Test Execution Status
Total Number of Test Cases
7
Total Executed
7
Total Number of Test Cases Passed
5
Total Number of Test Cases Failed
2
Total Number of Test Cases Skipped
0
Test Execution Date
2023-12-17
Test Name
Status
Automated API Tests.Executing VerifyAPIGet for Employee_Get with data https://dummy.restapiexample.com/api/v1/employee,5
Passed
Automated API Tests.Executing VerifyAPIGet for Employee_Get with data https://dummy.restapiexample.com/api/v1/employees,
Passed
Automated API Tests.Executing VerifyAPIGet for Employee_Get with data https://dummy.restapiexample.com/api/v1/employees,9
Failed
Failure Details:
Message: String lengths are both 4. Strings differ at index 0. Expected: {PASS} But was: {FAIL} ----------^
Stack Trace: at line: 36 in D:\AutomationP6\Pester\TestRunner.tests.ps1 36: $result | Should Be "PASS"
Automated API Tests.Executing VerifyAPIGet for Employee_Get with data https://dummy.restapiexample.com/api/v1/employee,11
Passed
Automated API Tests.Executing VerifyApiPost for Employee_Post with data https://dummy.restapiexample.com/api/v1/create,ID=12501,Employee_name=Naveen,Employee_salary=120000,Employee_age=37,Profile_image=""
Passed
Automated API Tests.Executing VerifyApiPut for Employee_Put with data https://dummy.restapiexample.com/public/api/v1/update/66,Employee_name=Naveen,Employee_salary=120000,Employee_age=37,Profile_image=""
Passed

6. Additionally, outputs for individual test cases, including GET, POST, PUT, and DELETE operations, are systematically captured and saved in the test results folder, providing detailed test case evidence. The process concludes once all outputs are documented.







** This was fun for me to share. It's totally extendable & you can enhance it as per your requirements.

Cheers!
Naveen