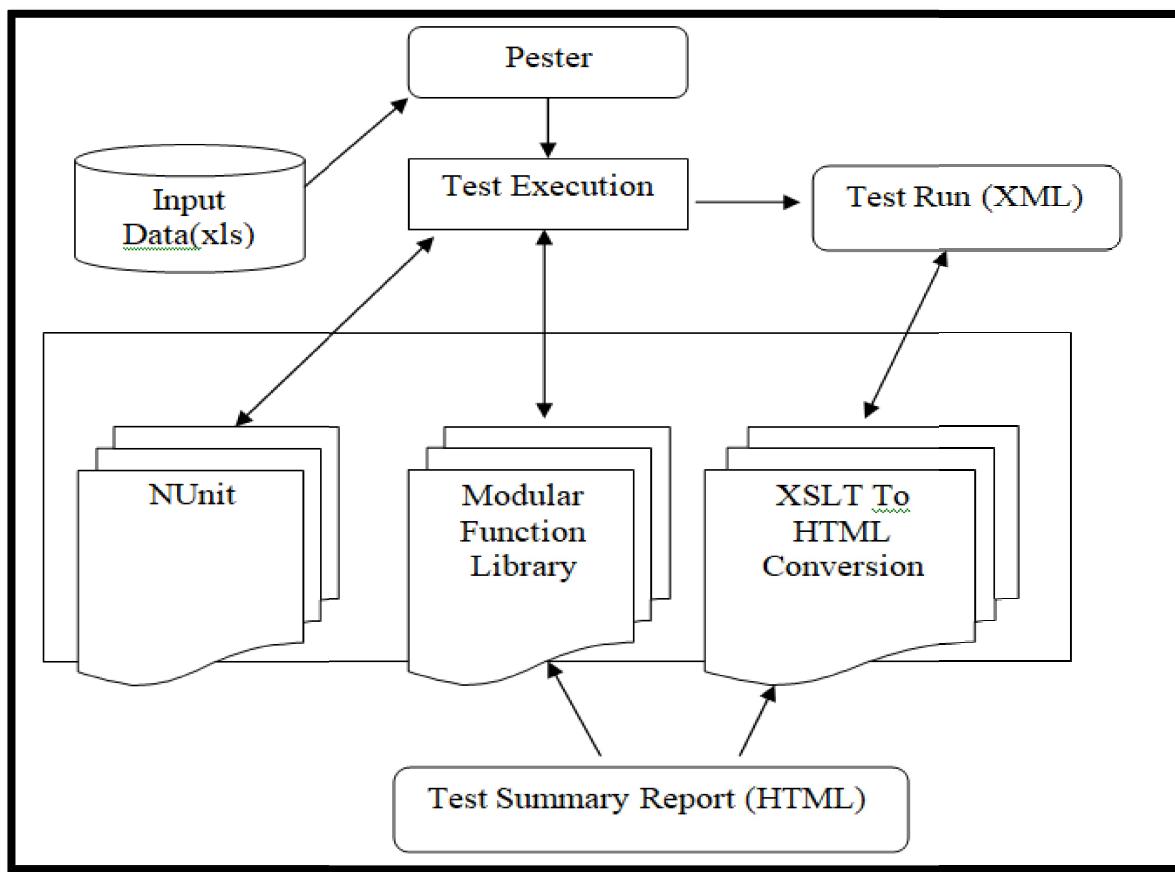


Framework Elements

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

This PC > Local Disk (D:) > AutomationP6 > Pester			
Name	Date modified	Type	Size
Config	17-Dec-23 10:24 PM	File folder	
FunctionLibrary	17-Dec-23 3:18 AM	File folder	
Help	17-Dec-23 8:10 PM	File folder	
InputSheet	18-Dec-23 12:03 AM	File folder	
NunitXML	17-Dec-23 6:10 PM	File folder	
TestReport	17-Dec-23 10:58 PM	File folder	
TestResults	17-Dec-23 9:58 PM	File folder	
TestRunner	18-Dec-23 12:29 AM	File folder	
StartTest	17-Dec-23 10:27 PM	Windows PowerShell Scr...	1 KB

Test Execution

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

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

The screenshot shows the Windows PowerShell ISE interface. The title bar says "Administrator: Windows PowerShell ISE". The menu bar includes File, Edit, View, Tools, Debug, Add-ons, and Help. The toolbar has various icons for file operations. There are three tabs at the top: "TestRunner.tests.ps1", "FunctionLibrary.ps1", and "StartTest.ps1" (which is currently selected). The code pane contains the following PowerShell script:

```
1 $pesterReportPath = "D:\AutomationP6\Pester\NunitXML\RunReport.xml"
2 Invoke-Pester -Script .\TestRunner.tests.ps1 -OutputFormat NunitXml -OutputFile $pesterReportPath
```

The output pane below shows the command being run:

```
PS D:\AutomationP6\Pester> D:\AutomationP6\Pester\StartTest.ps1
```

Describing Automated API Tests
Context Testing component: Employee_Get
[+] Executing VerifyAPIGet for Employee_Get with data https://dummy.restapiexample.com/api/v1/employee,5 4.51s
[+] Executing VerifyAPIGet for Employee_Get with data https://dummy.restapiexample.com/api/v1/employees 200ms

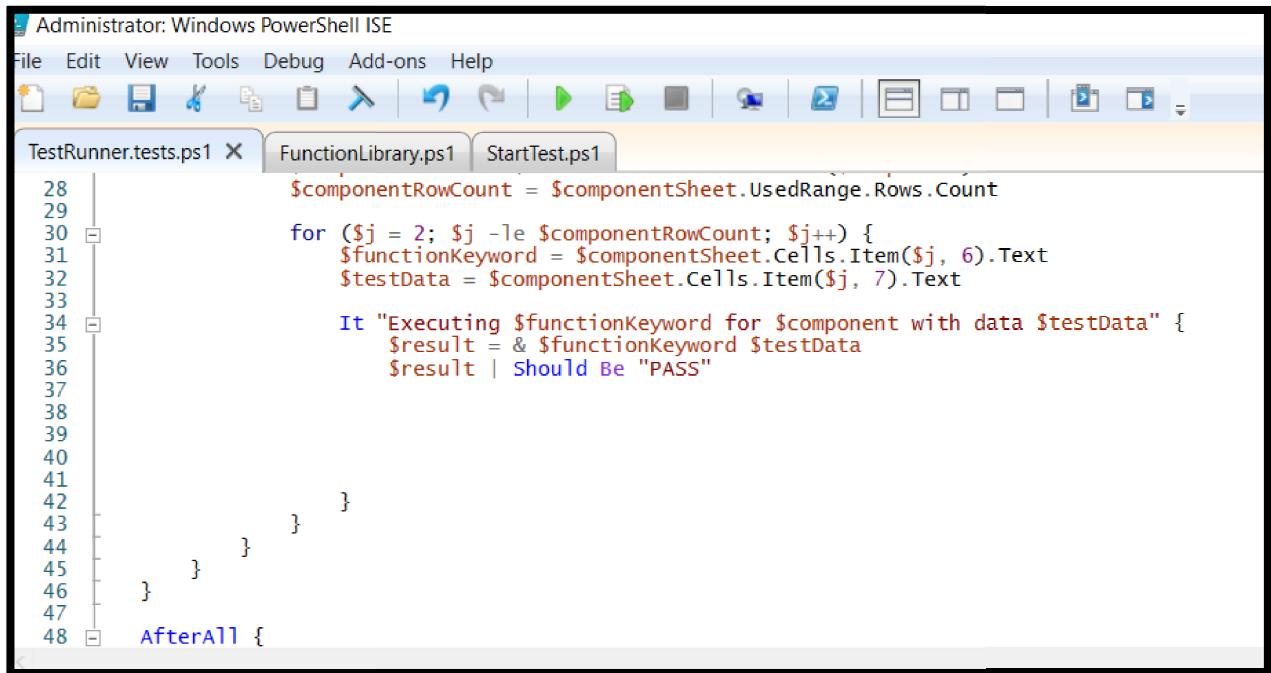
2. This process begins by reading from the Excel file, starting with the control sheet which lists available test suites and their corresponding execution flags (Y/N). The script selectively executes suites marked with 'Y', proceeding to the specific sheet of each suite to run the test cases.

The screenshot shows a Microsoft Excel spreadsheet titled "API_Test_Suit_v0.1 - Microsoft Excel". The ribbon menu is visible with tabs for Home, Insert, Page Layout, Formulas, Data, Review, and View. The Home tab is selected, showing the ribbon bar with various tools like Paste, Font, Alignment, and Number. The main area displays a table with the following data:

	A	B	C	D	E	F	G	H	I
1	Component	SuitToRun							
2	Employee_Get	Y							
3	Employee_Post	Y							
4	Employee_Put	Y							
5	Employee_Delete	Y							

The bottom of the screen shows the Excel ribbon tabs: TestControl, Employee_Post, Employee_Get, Employee_Delete, Employee_Put, and a few others that are partially visible. The status bar at the bottom left says "Ready".

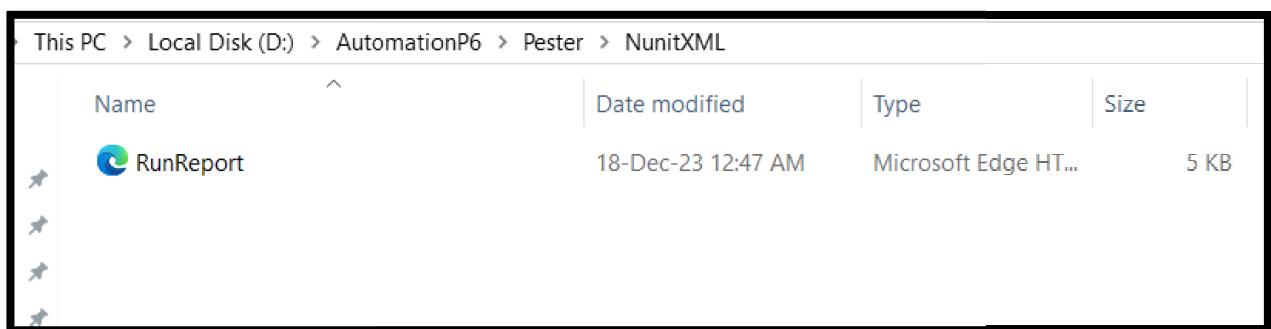
- During suite execution, the '**It**' block within the Pester script is iteratively processed to validate specific conditions or behaviours in the test case against predefined expectations. This block contains assertions that must return true for the test case to pass. The results, whether pass or fail,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



The screenshot shows the Windows PowerShell ISE interface with three tabs: 'TestRunner.tests.ps1' (closed), 'FunctionLibrary.ps1' (selected), and 'StartTest.ps1'. The code in 'FunctionLibrary.ps1' is a Pester test script. It defines a function 'RunReport' that iterates through rows in a component sheet, executing a command based on the function keyword and comparing its result against expected data. An 'AfterAll' block is present at the end of the function.

```
28 $componentRowCount = $componentSheet.UsedRange.Rows.Count
29
30 for ($j = 2; $j -le $componentRowCount; $j++) {
31     $functionKeyword = $componentSheet.Cells.Item($j, 6).Text
32     $testData = $componentSheet.Cells.Item($j, 7).Text
33
34     It "Executing $functionKeyword for $component with data $ testData" {
35         $result = & $functionKeyword $ testData
36         $result | Should Be "PASS"
37
38
39
40
41     }
42 }
43 }
44 }
45 }
46 }
47
48 AfterAll {
```

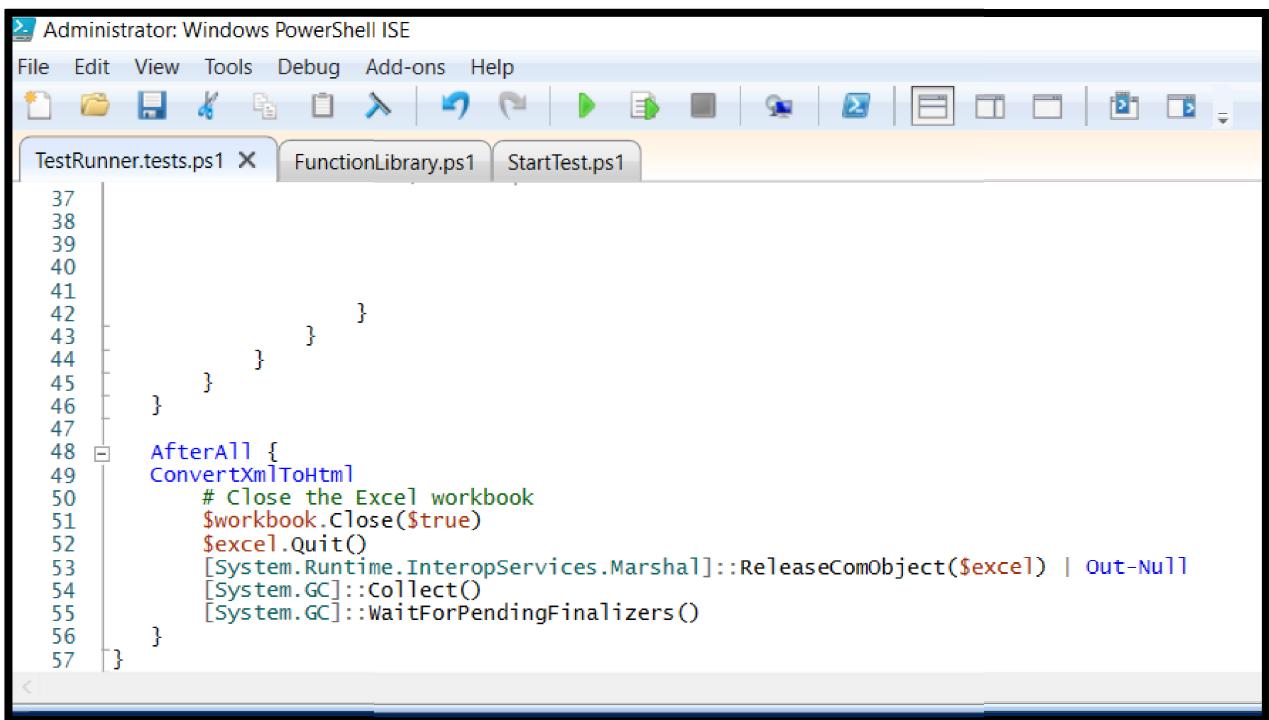
- Upon completion of all suites, an NUnit XML report is generated in a designated folder.



The screenshot shows a file explorer window displaying a file named 'RunReport' located in the 'NunitXML' folder under 'AutomationP6\Pester'. The file was modified on 18-Dec-23 at 12:47 AM and is a Microsoft Edge HTML file (5 KB).

Name	Date modified	Type	Size
RunReport	18-Dec-23 12:47 AM	Microsoft Edge HT...	5 KB

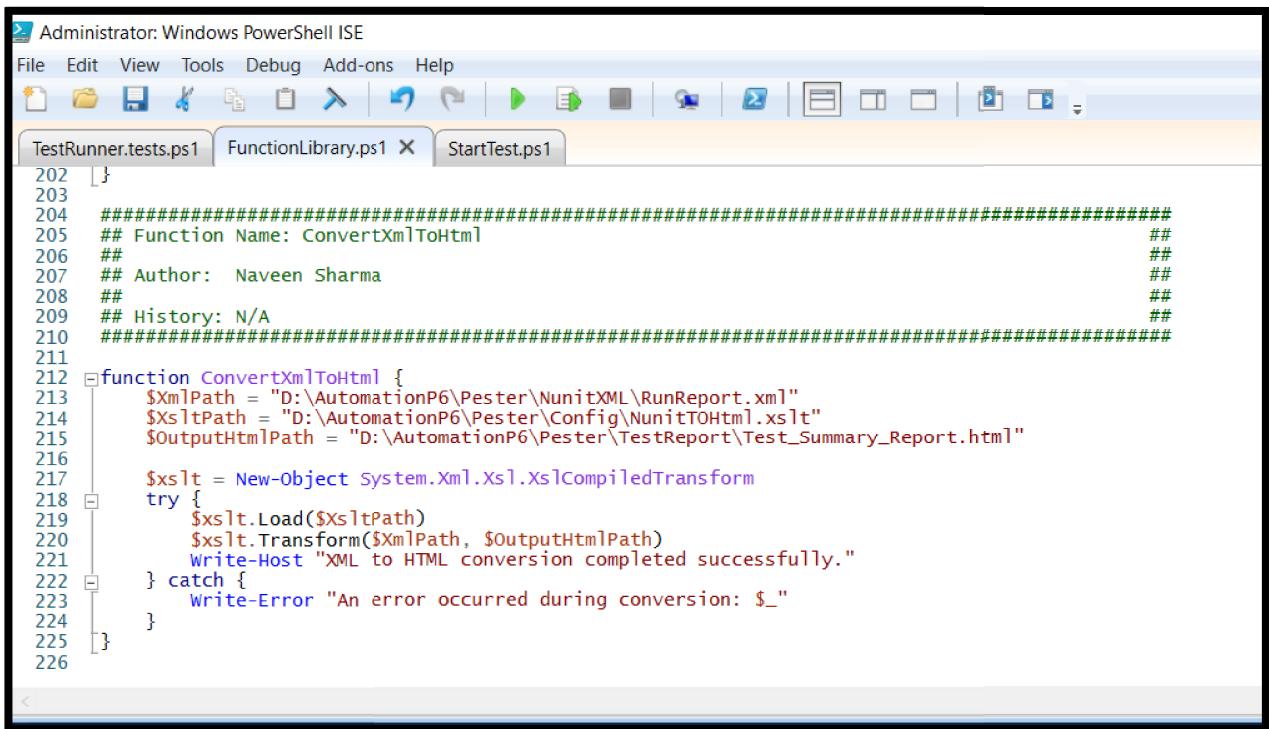
5. The '**AfterAll**' block within the script automatically converts this XML into a comprehensive HTML report stored in the test report folder.



```

Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
TestRunner.tests.ps1 X FunctionLibrary.ps1 StartTest.ps1
37
38
39
40
41
42     }
43 }
44 }
45 }
46 }
47
48 AfterAll {
49     ConvertXmlToHtml
50     # Close the Excel workbook
51     $workbook.Close($true)
52     $excel.Quit()
53     [System.Runtime.InteropServices.Marshal]::ReleaseComObject($excel) | Out-Null
54     [System.GC]::Collect()
55     [System.GC]::WaitForPendingFinalizers()
56 }
57

```



```

Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
TestRunner.tests.ps1 X FunctionLibrary.ps1 StartTest.ps1
202 }
203 #####
204 ## Function Name: ConvertXmlToHtml
205 ## Author: Naveen Sharma
206 ##
207 ## History: N/A
208 ##
209 #####
210 #####
211
212 function ConvertXmlToHtml {
213     $xmlPath = "D:\AutomationP6\Pester\NunitXML\RunReport.xml"
214     $xsltPath = "D:\AutomationP6\Pester\Config\NunitTOHtml.xslt"
215     $outputHtmlPath = "D:\AutomationP6\Pester\TestReport\Test_Summary_Report.html"
216
217     $xslt = New-Object System.Xml.Xsl.XslCompiledTransform
218     try {
219         $xslt.Load($xsltPath)
220         $xslt.Transform($xmlPath, $outputHtmlPath)
221         Write-Host "XML to HTML conversion completed successfully."
222     } catch {
223         Write-Error "An error occurred during conversion: $_"
224     }
225 }
226

```

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		Test Execution Status	
Platform	Microsoft Windows 10 Home Single Language C:\Windows\Device\Harddisk0\Partition2	Total Number of Test Cases	7
OS Version	10.0.19045	Total Executed	7
User Domain	DESKTOP- REDACTED	Total Number of Test Cases Passed	5
Machine Name	DESKTOP- REDACTED	Total Number of Test Cases Failed	2
User	Naveen	Total Number of Test Cases Skipped	0
Nunit-Version	8.0	Test Execution Date	2023-12-17

Test Name	Status	Details
Automated API Tests.Executing VerifyAPIGet for Employee_Get with data https://dummy.restapiexample.com/api/v1/employee,5	Passed	Executing VerifyAPIGet for Employee_Get with data https://dummy.restapiexample.com/api/v1/employee,5
Automated API Tests.Executing VerifyAPIGet for Employee_Get with data https://dummy.restapiexample.com/api/v1/employees,	Passed	Executing VerifyAPIGet for Employee_Get with data https://dummy.restapiexample.com/api/v1/employees,
Automated API Tests.Executing VerifyAPIGet for Employee_Get with data https://dummy.restapiexample.com/api/v1/employees,9	Failed	Executing VerifyAPIGet for Employee_Get with data https://dummy.restapiexample.com/api/v1/employees,9

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	Executing VerifyAPIGet for Employee_Get with data https://dummy.restapiexample.com/api/v1/employee,11
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	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=""
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	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=""

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.

```

Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
TestRunner.tests.ps1 FunctionLibrary.ps1 StartTest.ps1
62 #####
63 ## Function Name: VerifyApiGet #####
64 ## Author: Naveen Sharma #####
65 ## History: N/A #####
66 #####
67 #####
68 #####
69 #####
70 function VerifyApiGet {
71     param(
72         [string]$ApiTestData
73     )
74
75     # base path for the output files
76     $OutputBasePath = "D:\AutomationP6\Pester\TestResults\GET"
77
78     # Read & Split input data array
79     $ApiTestDataArray = $ApiTestData -split ","
80     $Url = $ApiTestDataArray[0].Trim()
81     $EmployeeId = $ApiTestDataArray[1].Trim()
82
83     # Append if Id is provided
84     if ($EmployeeId) {
85         $Url += "/$EmployeeId"
86     }
87
88     # Create a unique file for each API test result
89     $Timestamp = Get-Date -Format "yyyyMMddHHmmss"
90     $OutputFileName = "ApiTest_GetResult_${EmployeeId}_${Timestamp}.txt"
91     $OutputFilePath = Join-Path -Path $OutputBasePath -ChildPath $OutputFileName
92
93     try {
94         $ApiResponse = Invoke-RestMethod -Uri $Url -Method Get -Headers @{"Accept" = "application/json"}
95
96         # Convert the response to JSON & handle nested if any
97         $JsonResponse = $ApiResponse | ConvertTo-Json -Depth 10
98         $JsonResponse | Out-File -FilePath $OutputFilePath
99
100        if ($ApiResponse) {
101            return "PASS"
102        } else {
103            return "FAIL"
104        }
105    } catch {
106        Write-Host "Error occurred: $_"
107        "Error: $_" | Out-File -FilePath $OutputFilePath -Append
108        return "FAIL"
109    }
110 }

```

Name	Date modified	Type
DELETE	18-Dec-23 12:47 AM	File folder
GET	18-Dec-23 12:47 AM	File folder
POST	18-Dec-23 12:47 AM	File folder
PUT	18-Dec-23 12:47 AM	File folder

> This PC > Local Disk (D:) > AutomationP6 > Pester > TestResults > GET

Name	Date modified	Type	Size
ApiTest_GetResult_9_20231218004754	18-Dec-23 12:47 AM	Text Document	1 KB
ApiTest_GetResult_11_20231218004755	18-Dec-23 12:47 AM	Text Document	1 KB
ApiTest_GetResult__20231218004754	18-Dec-23 12:47 AM	Text Document	13 KB
ApiTest_GetResult_5_20231218004753	18-Dec-23 12:47 AM	Text Document	1 KB
ApiTest_GetResult_9_20231217230505	17-Dec-23 11:05 PM	Text Document	1 KB

ApiTest_GetResult_20231218004754 - Notepad

```

File Edit Format View Help
{
  "status": "success",
  "data": [
    {
      "id": 1,
      "employee_name": "Tiger Nixon",
      "employee_salary": 320800,
      "employee_age": 61,
      "profile_image": ""
    },
    {
      "id": 2,
      "employee_name": "Garrett Winters",
      "employee_salary": 170750,
      "employee_age": 63,
      "profile_image": ""
    },
    {
      "id": 3,
      "employee_name": "Ashton Cox",
      "employee_salary": 86000,
      "employee_age": 66,
      "profile_image": ""
    }
  ]
}
  
```

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-16 LE

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

Cheers!

Naveen