HPE UFT offers **testing automation** for functional and regression **testing** for the software applications. Visual Basic Scripting Edition scripting language is used by this **tool** to register the test processes and operates the various objects and controls in **testing** the applications

I used Selenium to test Automate scenarios

Latest Scenario U Automated : In our client portal there are lot of requirements for that we need to check whether all the urls are working properly or not for that we used broken links concept and first I used anchor tag by this I get all the response and by using a for loop I tried to get the response.

This way I tried to automate the scenario.

**How do you automate testing?**

1. Decide what Test Cases to Automate.
2. Test Early and Test Often.
3. Select the Right Automated Testing Tool.
4. Divide your Automated Testing Efforts.
5. Create Good, Quality Test Data.
6. Create Automated Tests that are Resistant to Changes in the UI.

**Why do we need automation testing?**

**Automated** software **testing** can increase the depth and scope of **tests** to help improve software quality. Lengthy **tests** that are often avoided during manual **testing** can be run unattended. They can even be run on multiple computers with different configurations.

To increase the test coverage.

Reduces the need for manual testing and discovers defects manual testing cannot expose and also manual testing is error prone and a time consuming process.

Running the tests again and again gives us the confidence that the new work we added to the system did not break the code that used to work and also to make sure that the changes we introduced are working.

Executing the tests (particularly acceptance tests) can also help us understand what portion of the desired functionality has been implemented.

The set of the automated test suite can form a regression test suite.

The purpose of the regression suite is to make sure that the software behavior is unchanged unless due to data change or latest software.

Automating also reduces the time taken for regression testing. Automated unit test suite helps find the problems at an earlier stage and solve them.

**What is a Testing Framework ?**

A testing framework or more specifically a testing automation framework is an execution environment for automated tests. It is the overall system in which the tests will be automated. It is defined as the set of assumptions, concepts, and practices that constitute a work platform or support for automated testing. The Testing framework is responsible for:

§ Defining the format in which to express expectations.

§ Creating a mechanism to hook into or drive the application under test

§ Executing the tests

§ Reporting results Properties of a testing framework:

§ It is application independent.

§ It is easy to expand, maintain and perpetuate.

**Why we need a Testing Framework?**

If we have a group of testers and suppose if each project implements a unique strategy then the time needed for the tester become productive in the new environment will take long. To handle this we cannot make changes to the automation environment for each new application that comes along. For this purpose we use a testing framework that is application independent and has the capability to expand with the requirements of each application. Also an organized test framework helps in avoiding duplication of test cases automated across the application. In short Test frameworks helps teams organize their test suites and in turn help improve the efficiency of testing.

**Types Of Testing Frameworks ?**

Testing Frameworks :

Modular

DataDriven   
KeywordDriven

Hybrid

**Modular Testing Framework**

• The Modularity testing framework is built on the concept of abstraction.

• This involves the creation of independent scripts that represent the modules of the application under test. These modules in turn are used in a hierarchical fashion to build large test cases.

• Thus it builds an abstraction layer for a component to hide that component from the rest of the application. Thus the changes made to the other part of the application do not effect that component.

Module1 Module2 … Module N Module N+1 Module N+2 … Module N+10 Test Script Test Script

Example of Modular Testing Framework

To demonstrate the modular framework we use the calculator program.

Consider the basic functions of the calculator such as addition, subtraction, multiplication, division which are part of the Standard view. We create scripts for these functions as follows: Add: Sub Main Window Set Context, "Caption=Calculator", "“ PushButton Click, "ObjectIndex=10“ ‘Press 5 PushButton Click, "ObjectIndex=20“ ‘Press + PushButton Click, "ObjectIndex=14“ ‘Press 6 PushButton Click, "ObjectIndex=21“ ‘Press = Result = LabelUP (CompareProperties, "Text=11.", "UP=Object Properties") ‘Compare Expected to Actual Results End Sub In a similar way we create scripts for subtraction, multiplication and division.

At the next level of hierarchy, we create two scripts for standard view and scientific view of which the standard view contains calls to the scripts we created as before. Add Subtract Multiply Division Standard View Scientific View Driver Script Log Sin The Driver script is the top most level of hierarchy which contains the scripts of standard and scientific view. Driver Script: Sub Main 'Test the Standard View CallScript "Test Script Mod Framework - Standard" 'Test the Scientific View CallScript "Test Script Mod Framework - Scientific“ End Sub Thus this framework introduces a high level of modularization. So when there is a change in the functionality we can change the bottom level script without effecting all the other test cases that test that control.

**Advantages:**

Modular division of scripts leads to easier maintenance and also the scalability of the automated test suites. The functionality is available in easy to use test libraries so creating new driver scripts for different tests is easy and fast.

**Disadvantages:**

• The main problem with modular frameworks is that the test script have test data embedded in them. So when the test data needs to be updated we need to change the code of the script. This becomes a big problem when the test script is large. For this purpose, data- driven testing frameworks have been introduced.

**Data-Driven Testing Framework**

Data driven testing is where the test input and the expected output results are stored in a separate data file (normally in a tabular format) so that a single driver script can execute all the test cases with multiple sets of data.

The driver script contains navigation through the program, reading of the data files and logging of the test status information.

Example of Data Driven Testing Framework

• To demonstrate the data driven testing framework we use the login page of the flight reservation • The first step involves creating the test data file. (testdata.csv)

• This data file contains the different types of input data which will be given to the driver script. Test Case Number1 Operator Number2 Expected Result Add 2 + 3 5 Subtract 3 - 2 1 Multiply 2 \* 3 6 Divide 2 / -2 -1

• In the next step we create a driver script and make references to the test data file.

data = open ( ’ testdata.csv’ ) . read ( )

l i n e s = data . s p l i t l i n e s ( )

#excluding the header row for line in lines: Read Number1 Read Number2 Read Operator Calculate the result using the Operator on Number 1 and Number2 Compare the result to the expected result

• This driver script reads the data from the data file computes the value and compares it with the expected result from the data file.

**Advantages:** • This framework reduces the number of overall test scripts needed to implement all the test cases.

• Less amount of code is required to generate all the test cases.

• Offers greater flexibility when it comes to maintenance and fixing of bugs.

• The test data can be created before test implementation is ready or even before the system to be tested is ready.

**Disadvantages:**

• The test cases created are similar and creating new kind of tests requires creating new driver scripts that understand different data. Thus the test data and driver scripts are strongly related that changing either requires changing the other. For this purpose keyword driven testing frameworks have been introduced

**Keyword- Driven Testing Framework**

Keyword driven testing is an application independent framework utilizing data tables and self explanatory keywords to explain the actions to be performed on the application under test. Not only is the test data kept in the file but even the directives telling what to do which is in the test scripts is put in external input data file. These directives are called keywords.

The keyword based testing is an extension to the data driven testing.

Keywords/ Actions 🡪Test Data🡪 (input)Driver Script 🡪(output)Test Results

**Advantages:** It has all the advantages that data driven testing has. Automation expertise is not required to maintain or create a new set of test cases. Keywords are reused across multiple test cases. **Disadvantages:** The main problem is that this requires a more complicated framework than the data driven framework. With the keyword driven approach the test cases get longer and complex and this is due to the greater flexibility that this approach offers. So in order to combine the strengths of all the frameworks and mitigate their weaknesses we use the hybrid testing framework.