International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2015): 78.96 | Impact Factor (2015): 6.391

Design and Implementation of UI Automation Framework for e-Learning Application

Rohini P[1], Dr. Nagaraj G Cholli[2]

[1]Department of ISE, R.V College of Engineering, Bengaluru, India

[2]Associate Professor, Dept. of ISE, R.V College of Engineering, Bengaluru, India

Abstract: ***Any product is accepted or approved only after it is tested. Testing is the important phase that any product has to go through at least once during the development life cycle. In early 40's it was the developers who performed testing and it was a part of debugging as well. In today's world where software is the essential element for any product to work, testing of the software has to be performed before we release the product. Software testing is necessary to increase the reliability of the product, deliver a quality product and to deliver in time. Manual testing is accompanied by automation testing because this uses automation tools to reduce the human involvement. In today's world e-Learning is one of the biggest achievements in the education domain. Helping the faculty and students to connect easily anywhere is a milestone achieved. In this paper we represent an automation framework for testing the Intellus application. The framework is implemented with the help of Selenium webdriver tool and output is a report in HTML/XML format that is generated after the test execution is complete. To reduce the manual effort, network lags, request timeouts, application load times. Hence the framework improves the performance of the application.***

Keywords: software testing, e-Learning, automation testing, Selenium webdriver, HTML report

1. Introduction

Software testing is one of the major software development methods that emerged during 1970s. The main purpose is to ensure that a computer program will satisfy the needs and requirements of the customers, perform its intended functions. Software testing is widely defined as an eight step process for a test strategy which is based on customer requirements, project plan on how to conduct the testing process, design methodology for tests, to verify a range of test cases with the help of independent test cases, test mechanism equivalent to specified set of conditions, logs for recording the test actions, test summary to report the observations of the testing process and a test item transmittal for the communication purpose.

Software testing can be classified into two types :

- Manual Testing : This is the technique where the software defects are found manually i.e., test cases are manually prepared and executed by the test engineer to identify the bugs in the software. Manual testing is said to be one of the old method of software testing. The tester has to play the role of end user and should ensure that all the features of the application are correct. Manual testing requires the tester to possess a set of qualities like patient, observant, creative and innovative. It may become difficult to perform this type of testing if the application has very large data set coverage.
- Automation Testing : This is the technique where the testing of the application is performed without human intervention. It mainly involves writing scripts with the help of any programming language like Java, Python, .Net and so on. Due to the scripts there are less human errors and coverage of large data set will become easy. It is also less time consuming and very efficient.The initial setup cost is said to be high but once the setup is done, the cost will be reduced because it doesn't need any human interaction compared to the manual testing. Automation testing has more advantages compared to manual testing, hence it will overcome the disadvantages of previous methods of testing.

Figure 1: Various activities in the testing cycle.

From the Figure 1, execution of test cases is one of the main activity. It may be by the manual test engineer or test scripts by an automation test engineer. Even the manual tester can execute the automation scripts if he has the knowledge tool and the framework designed by the automation tester and also regarding the programming language to debug the error in the scripts [1].

1.1 e-Learning Overview

e-Learning is the new learning technique through Internet. e-Learning is digital learning which provides individual multi-interaction that eliminates the disadvantages of traditional

classroom teaching [2], [3]. In future e-Learning will be the main sources of learning because of its advantages. Some of them are listed below [4], [5]:

- The learning source is shared hence no space limitation. There is no time limitation because the teaching platform server and Internet functionality is proper, that means we can learn anytime online.
- Learning resources are available online and the reusability of the same is possible.
- Faculty can express the meaning of the content with the help of multimedia.
- This is one-one methodology and learning, the faculty can determine each and every learning situation so as to help the students.
- It interests more people to follow this technique as this automation is more active than the traditional classroom approach.

Intellus is one such platform that enables easy and effective creation of course and helps in publishing the content resources which are appropriate and aligned. Faculty are the clients and they create courses with the course outcomes to make it available for students. With the help of Learning management system, students can view the courses that are made available by the faculty.

2. Automation: A Priority

Our application was highly dependent on manual testing. Several problems were devised due to which we came to conclusion that automation should be made as a priority. The problems faced by following the traditional approach were as follows:

- Not one kind of LMS : As there were many clients, it was not only one LMS used by the students so testing those manually was a tedious task.
- RESET / SET data tables : Every time we test the application it was required to reset the data in the database, doing this manually was time consuming.
- Test Coverage : By following the traditional method it was not possible to have the complete test coverage.
- Man hours : Performing manual testing always demanded more man hours which was not cost effective.
- Correctness of data : Even though manual test was performed quite few number of times, it was not possible to ensure the correctness of the data.
- Test Sanity : With the new features being added frequently, performing sanity test became very difficult.

To overcome the above problems we followed the automation testing . It is a process of endorsing the application functionality by designing scripts using a programming language and implementing it anywhere and anytime whenever required [1]. They are tools available for automation which are the softwares to perform the testing activities like test management, functional testing, performance testing etc [6].

2.1 Requirement Strategy

The automation framework for any application should be according to the requirement strategy. We also took some of the following things into consideration before designing the framework.

- Application : The user interface of the application is developed using AngularJs [7] and Python.
- List of test cases : We have to first automate the acceptance test cases, the one's that always pass.
- Selection of tool : With all the advantages of Selenium, we used Selenium webdriver for automating the application. Selenium does not overtake other testing tools like QTP [8]. Some of main reason for using Selenium webdriver to automate our application are as follows :

a) Our application is web based and is growing very fast, we found selenium webdriver was very helpful to build a test suite that can be used for regression because it enables easy browser based regression suite.
b) The primary intention was to automate user actions and also to check for the easy compatibility of browser along with selenium. Selenium is the tool which is open source and has large community support, which supports multiple scripting or programming languages.
c) We wanted a tool that can be extended to honour our user actions that are exposed or public APIs that can be used or modified.

- Scripting language : We decide to code our test scripts in Python because of its versatile nature, easy to write and understand , has more support and form of modules when compared to Java it has less construction rules, english like syntax and type casting is also easier [9].
- Results : To validate that the tests have passed, we generate a report in HTML/XML format. For this purpose, we made use of HTMLTestRunner and if any tests failed it should have the location screenshot. For this we have made use of the third party tool that would take the screenshot of the failed test case [10].

3. Implementation of Framework

First of all, the test scripts are written for the acceptance test cases. Since we follow BDD (Business Driven Development) the test cases are initially known to us.

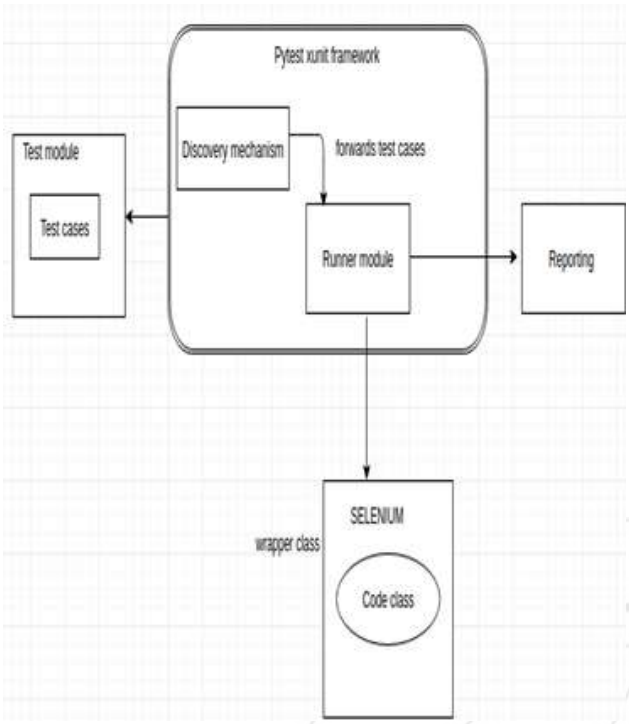

Figure 2: Block diagram of the framework.

Functionality of the different modules:

- **Test Module** : This module contains the tests cases are written inside this module. The test cases are to be run/executed. It also should say what it needs, what kind it is whether UI or API. Always the acceptance tests are automated, a list of tests to be automated was first developed then they were converted into scripts.
- The **discovery mechanism** which is built in the Pytest framework will find the test module, identifies the type of test case and then forward it to the Runner module i.e., Test Runner. The mechanism is to identify if the test script is UI or API based.
- **Test Runner/ Runner module** : This is a library which will provides a simple, repeatable pattern for writing and running Selenium tests within Node that are synchronous in nature. This will setup, execute and tear down.
- The discovery mechanism and Runner module are in the Pytest framework.
- **Pytest xunit framework** makes it easy to write small tests, also scales to support complex functional testing for applications and libraries. Python unit test framework inherits its root from PyUnit module.
- After the test cases are executed a report is generated and this result is done by **Reporting** tool, either XML or HTML tool. This tool will collect the run information from the Runner module, collects the logs and generates report from the logs.
- Runner module with the help of **Selenium code class**, which is a wrapper class and will have two classes one for UI and one for API and the Selenium APIs will take code from Selenium code class for executing test cases. If the test case is identified as a API test case then Pytest APIs are used to execute them. If the test case is for UI then this will make use of the code class of the selenium and its APIs to execute them.
- Mostly we write logic to run the acceptance test cases and some more test cases are added with the help of test design techniques like Boundary Value Analysis, Equivalence Class Partitioning.
- For any scenario to be automated we should develop a test case class. By inheriting the Test class of unit test module we can create more tests. We should also provide corresponding test method /test handler to the derived class if we want to add the test case.
- We can use assert to finalise the test case and get reporting status. Some of the most common assert functions used:
  a) assert_checkbox() - To verify element is the checkbox.
  b) assert_textfield() - To verify element is a textfield.
  c) assert_link() - To verify element is link.
  d) assert_url() - To verify element is url.
- In addition to test handlers, we also have routines like setUp() and tearDown().
  a) setUp() - all basic requirements to run a test case.
  Ex: launch browser, here we have start() called at start of each script.
  b) tearDown() - clean up after test case run is complete.
  Ex: close browser, here we have stop() called at end of each test script.

```
#Function to start browser
def startBrowser()
    if browser_type is None:
        Browser_type = 'Firefox'
    "Firefox browser is started for every session.
    This function is called at start of the test
    script"
    |
    else if browser_type = 'Chrome':
        #Chrome webdriver is called and the browser
        is launched.

def stopBrowser()
    "This function is executed at end of the session.
    This function is called at end of the test
    script."
```

Figure 3 : Function module to start the browser.

The above Figure 3, will describe the function written to call the browser for executing the automation scripts. By, default the browser called is Firefox. Our application calls the Firefox browser to execute the test scripts. The stratBrowser() function is executed at start of the test script and stopBrowser() function is called at the end of the test script.

```
"""Function to run HTMLTestRunner report"""

import pyunittest
import HTMLTestRunner
  #tests are defined#
if name == ' main ' :
  """Main class of HTMLTestRunner is called."""
"HTMLTestRunner object is instantiated"

##Output of the file##
report.html file is created.
Runner object is give the stream name of the file,
title and description.

##run the test##
runner.run(my_test_suite)
```

Figure 4: Function to generate HMTL test report.

As shown in the above Figure 4, after the test scripts are executed a report is generated to let us know the status of the test cases. The above figure shows the module for the same. We had an option of generating the test report in .xml format but to beautify the report format we have used HTMLTestRunner which is the a third party tool, this help us to generate the reports in .html format.

A test script is executed in command line, as we have implemented it using python, we use python command to execute the test script. This is the main command that we run to execute our test script.

python intellus_run -r html -s -d tests/intellus test_case1.py
The above is the syntax used to execute the test suites in the command line.

a) intellus_run tells us where the test cases are present.
b) r html is defined and it means a HTML report has to be generated.
c) s is also defined, it means screenshot of the browser window is taken wherever it is defined in the script.
d) d is defined in the command.py , it means that it is directory.
e) tests/intellus is the path where the tests are present.
f) test_case1.py is the test suite where actually the code is present.

The following is the skeleton of test suite:

From intellus.util.action_wrapper import run_action
run_action('params','kwargs')**

- params – app_name , method name and the type of test case.
- **kwargs – Keyword arguments with Key : Value pair.

4. Framework Test Report

The results are generated with the help of HTMLTestRunner. The reports can be either in the form of HTML/XML but we prefer it to be in HTML as it is very easy to understand and debug the errors. Whenever there are errors found the report will give us the detailed description of the errors. As HTMLTestRunner is already available we did not take the risk to write the function to generate the reports.

Test Group/Test case	Count	Pass	Fail	Error	View
	9	9	0	0	Detail
test_Demo_arpEcoCheck			pass		
test_Demo_arpBioCheck			pass		
test_Demo_moodleBioCheck			pass		
test_Demo_canvasBioCheck			pass		
test_Demo_arpCheck			pass		
test_Demo_canvasEconCheck			pass		
test_Demo_brightspaceEconCheck			pass		
test_Demo_bbEconCheck			pass		
test_Demo_moodleEconCheck			pass		
Total	9	9	0	0	

Intellus-UI Test Report

Start Time: 2017-06-14 15:54:02
Duration: 0:25:26.033197
Status: Pass 10

Figure 6: HTML report generated when the test scripts are passed.

The Figure 6, shows the report generated after the execution of the tests scripts. The scripts contain the report title, start time of the automation test, duration of the test cases, status of how many reports are passed or failed. The report also give us a clear picture of the assertions in the test case.

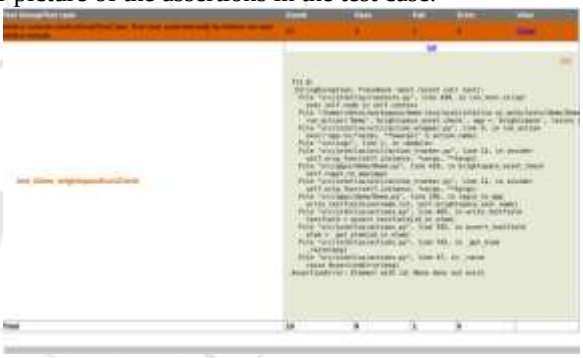

Figure 7: HTML report generated when test scripts fail to execute.

Sometimes, we also see the test cases failing when executed in a group and it is difficult to identify which case has failed and at which point and the reason for failure. So report generation function also has an option to display the error report. The Figure 7 shows the HTML report when test script fail to execute, the reason for failure and also the line number where the script has failed. This kind of report helps us to identify errors in the script and correct them.

5. Conclusion

In this paper, we have shown how the automation framework is designed for the Intellus application. It also describes why we choose automation testing and the advantages of the automation testing. The framework developed using Selenium webdriver is a open source web based testing tool, which supports multiple browsers and multiple scripting languages that enhances the performance of the application. With the

help of the framework test reports we can also conclude that our automation framework is very time efficient as well as cost effective which can be used for repeatable execution of the task.

References

[1] Rishab Jain C and Rajesh Kaluri "Design of Automation Scripts Execution Application for Selenium Webdriver and TestNG Framework" ARPN Journal of Engineering and Applied Sciences, April 2015.
[2] W. L. Shi, "Digital teaching strategies based on learning theory," Living Technology Education, vol. 40, no. 2, pp. 32-41, 2008.
[3] C. L. Hsu, "The Study of Computer Assisted Instruction and E-learning in Instructional Design Perspective in the Future," Educational Resources and Research, vol. 78, pp. 21-40, 2007.
[4] R. H. Huang, "The functions and standards of e-learning management system," Information and education, vol. 89, pp. 21-22, 2002.
[5] S. Russ, "10 Reasons E-Learning Is a Successful Training Tool," Multi-Housing News, vol. 38, pp. 24, 2003.
[6] Monika Sharma, Rigzin Angmo,"Web based Automation Testing and Tools," (IJCSIT) International Journal of Computer Science and Information Technologies. 5(1): 908-912, 2014.
[7] Sneha Ambulkar, "AngularJS," International Journal of Scientific & Engineering Research,Volume 7, Issue 2, February 2016.
[8] Jagannatha S et al, "Comparative Study on Automation Testing using Selenium Testing Framework and QTP," International Journal of Computer Science and Mobile Computing, Vol.3 Issue.10, October 2014.
[9] Fabian Trautsch, Jens Grabowski, "Are There Any Unit Tests? An Empirical Study on Unit Testing in Open Source Python Projects," Software Testing, Verification and Validation IEEE International Conference, March 2017.
[10] Sherry Singla, Harpreet Kaur, "Selenium Keyword Driven Automation Testing Framework," International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 6, June 2014.