AUTOMATION EXPERINCE

Portfolio by Zamir Yusof

INTRODUCTION

Hello, I'm Zamir Yusof.

This document records my experience related to automated software testing or automation testing where I believe it deserves a special highlight than my resume.

In the next few pages, I will explain how I have played my role as automation test engineer from the test cases planning to reporting. From my experience, it's not easy for hiring manager or interviewer to have an idea on how automation testing works and the role of QA tester in the activity especially when there are only few of automation test engineer compared to manual test engineer.

With that regard, this is not a detailed guide on automated software testing but rather my own personal experience. I hope it can helps hiring manager to evaluate me as applicant and estimate how I can contribute to the organization.

TEST CASES SELECTION

Although automation testing has always being touted as the future of software testing where most of the test cases can be executed through automation, it is not as easy as it sounds.

Before we can proceed to automate test execution, there are a few key criteria:

- The System Under Test (SUT) must be relatively stable
- The SUT must be automation-friendly
- There should be no or minimal dependency on manual or complex tasks
- Test data can be easily prepared
- The results must be machine-interpretable

Below are the more detailed explanations on the above's key points.

1. The System Under Test (SUT) must be relatively stable

Trying to implement automation testing while the SUT are still bug-ridden (for example during SIT) will be a waste of effort since the test will be more likely to fail due to flow-stopping bugs and changes in UI or the test flow.

We can expect SUT to be stable after it passes UAT, which is during or after regression testing. So automated testing will be greatly helpful to execute hundreds of regression test cases.

2. The SUT must be automation-friendly

One of the challenges in implementing automated testing is identifying locators for the UI elements. Both web and mobile app has similar approach to locators, but different level of challenges.

In web application, most of UI elements have text so automation tester can rely on short but still unique Xpath.

For example, a locator for a webform title on a webpage can be written as simple as

//b [text()='Total Free Unit']

On mobile application, there is more dependency on colored button or images, where the app developer may add the element without providing unique identifier. In such cases, automation tester can ask the devs team to add the identifier but there's also an option to use indexing although this may lead to lengthy and less user-friendly locator.

Example below is a unique locator for a number pad on mobile app where the same text '1' also appear in other elements.

(//android.view.ViewGroup[@index='2'])[3]//android.widget.TextView[@text='1']

3. There should be no or minimal dependency on complex tasks

One such example is login into user's account. Although to a human it sounds simple, it can be challenging task to interpret in automation.

For example where user need to login with OTP required, almost all the time the OTP code must be sourced from separate application like Google Authenticator or email. This will add more complexity as automation tester not only need to automate the testing for SUT, they also need to consider automating the action between the SUT and the OTP application.

Most of the time, this feature required to be disabled in testing environment. But there was a time where in a project that I worked, OTP can't be disabled. In this case, I have provided solution as follows;

i. Create a program to generate OTP within the test automation system (TAS).

```java
package com.born;

import java.util.Properties;


public class Authenticator {

    /* Perform steps below for 1st time setup
     * 1. Navigate to https://xxxx.microsoft.com/security-info
     * 2. Add sign-in method
     * 3. Choose method Authenticator app
     * 4. Choose "I want to use different app"
     * 5. Proceed until asked to scan QR code, click "Can't scan image"
     * 6. Copy both account name and secret key to related variables below then save this file.
     * 7. Back to MFA setup page, click Next to enter 6 digit code
     * 8. Run this program to generate the 6 digit code and complete the MFA setup using the same.
     */

    public static String otpcode;
    public static String MFAaccount = "xxxxx:V15705X@xxxx.com.my"; //user's email
    public static String secretkey = "2pgrlcryxxxxx"; //user's secret key


    public static void main(String[] args) {
        otpcode = getTOTPCode();
        System.out.println(otpcode);
        try {
            Thread.sleep(2000);
        } catch(InterruptedException e) {
            System.out.println("got interrupted!");
        }

    }
    public static String getTOTPCode() {

        Totp totp = new Totp(secretkey);
        String twoFactorCode = totp.now();
        return twoFactorCode;
    }
}
```

ii. Create a method to trigger the OTP code generation, fetch the code and then proceed with login with the code

```java
public String getOTPcode (String object, String data) {
    APP_LOGS.debug("Fetching and entering OTP code " + object);
    try {

            driver.findElement(By.xpath(OR.getProperty(object))).click();
            System.out.println("Generating OTP for account " + Authenticator.MFAaccount);
            String gencode = Authenticator.getTOTPCode();
            Thread.sleep(2000);
            System.out.println("OTP code is " + gencode);
            driver.findElement(By.xpath(OR.getProperty(object))).sendKeys(gencode);
        }
    catch (Exception e) {
        return Constants.KEYWORD_FAIL + "Unable to get code";
    }
    return Constants.KEYWORD_PASS;
}
```

4. Test data can be easily prepared

This may be often overlooked by automation tester especially those that have not involved during manual testing. Depending on complexity of the test cases, the test data must be accurate and also can be easily prepared.

For example during new postpaid line registration, user need to provide unique NRIC that has not been registered while also matching the availability of MSISDN based on the NRIC number pattern (eg. If the last 4 NRIC digit is 1234, the system will suggest MSISDN that has pattern of 1234 to the customer).

I have written a method and test script to perform such action during automating new line registration as follows;

i. Create a method to generate random numbers and also allow tester to set how many digits to use

```
public void randomnumber() throws Exception
{
    String num = keyRefPar.getCurrentInputData_Pays();
    By byobjectname= uimap.getStoreFinder(keyRefPar.getCurrentObjectName());
    try {
        Random random = new Random();
        String randNumber = String.format("%0"+num+"d", random.nextInt(10000));
        driver.findElement(byobjectname).sendKeys(randNumber);
        report.updateTestLog("Generate" +num+ "digits Random Number"+keyRefPar.getCurrentObjectName(),
              "Random Number" +randNumber+ "set successfully ",Status.PASS);
    }catch (Exception e) {
         report.updateTestLog("Generate" +num+ "digits Random Number"+keyRefPar.getCurrentObjectName(),
                 "Random Number Failed ",Status.FAIL);
    }
}
```

ii. Set where and when to use predefined NRIC digits and randomized digits

Action	LabelName	ObjectName	TestDataReference_Pays	InputData_Pays
comment	***Step 2 Navigate to URL defined in the test case***			
goToURL				URL_ATLNLPO_TC01
comment	***Step 3 Fill Personal Details and proceed***			
click	Select NRIC	txtNL_NricFld1		
settext	Enter NRIC	txtNL_NricFld1		<NRIC1>
randomnumber	Enter NRIC	txtNL_NricFld2		2
settext		txtNL_NricFld3		0
randomnumber	Enter NRIC	txtNL_NricFld3		3
click	Click Check Terms and condtions	chkTnC_EMV	JSScript	
click	Click Proceed button	btnProceed_NRICScrn		

5. The results must be machine-interpretable

Automated testing can only be used when there is element that can be easily interpreted by the machine.

For example, automation testing is not really feasible for identifying quality of images displayed on the web page. But, we can rely on automation to identify elements such as image file with specific file name, or specific success message text to indicate the automated test is success. In the screenshot below, the final steps are the verify the text and also the UI element on line registration success page.

Action	LabelName	ObjectName	TestDataReference_Pays	InputData_Pays	ExpectedData_Pays	Opt_Param_Pays
waitForsec						
waittillobjexist		elmOrderConfMsg				
comment	***Step 12 Verify order summary page***					
verifyelementtextcontains	Verify Order Message	elmOrderConfMsg		Your order is successful!		h1
getObjectTextRunTime		lblOrderConf_OrdrID	order			CJ:strOrderID
scrollPage				down		
verifyobjectexist		elmAmt_NLTC01_PromoSucc				
comment	***End of the test case***					

WRITING TEST CASES

Being an automation test engineer does not mean there will be no need to get involve and understand the manual test execution. One of the importance is recognizing the passing criteria of the test cases. An automation test engineer will simply focus on achieving the outcome eg. Placing an order. But for me as someone with years of experience in manual testing, there should be multiple area/pages where additional validation must be performed to achieve accurate result.

In the same example I mentioned before, there was a hidden requirement where the MSISDN to be auto-populated based on NRIC digit during new postpaid line registration. Since MSISDN pool in test environment I worked with was limited, the NRIC to be used must be customized according to the availability of the MSISDN pool eg. number starting with digit '0'.

Generally automation test cases are based on the manual test cases, but with more variations in term of test data and product step up since quicker execution time allow us to add more test cases (usually more than 100).

Action	LabelName	ObjectName	TestDataReference_Pays	InputData_Pays
comment	***Step 2 Navigate to URL defined in the test case***			
goToURL				URL_ATLNLPO_TC01
comment	***Step 3 Fill Personal Details and proceed***			
click	Select NRIC	txtNL_NricFld1		
settext	Enter NRIC	txtNL_NricFld1		<NRIC1>
randomnumber	Enter NRIC	txtNL_NricFld2		2
settext		txtNL_NricFld3		0
randomnumber	Enter NRIC	txtNL_NricFld3		3
click	Click Check Terms and condtions	chkTnC_EMV	JSScript	
click	Click Proceed button	btnProceed_NRICScrn		

SELECTING AUTOMATION FRAMEWORK

In the organization that I am working currently, there are 2 Selenium-based custom automation frameworks with different capabilities as such (not with their actual name);

- Framework A – allows parallel execution but only supports web application, not user-friendly to execute same test cases with multiple test data set
- Framework B – support both web and mobile application in 1 test script, allows repetition of test case with multiple test data set but does not allow parallel test execution.

As common with any Selenium-based automation framework, setting up the framework requires high level expertise in automation and programming since it need to be configured with different modules for reading external test data files, test recording and reporting etc.

With my limited programming experience, my task only involves in adding methods or functionalities according to project requirement, updating existing methods and maintaining the locators library. Further development of the frameworks is being handled by RnD team in the organization, but I have experience creating basic Selenium framework as part of my study.

I do have some experience with other frameworks such as Cypress or Katalon due to them being more user-friendly and thus considered as an option but have not committed in favor of our in house Selenium-based automation frameworks which provides more flexibilities.

TEST EXECUTION

Both frameworks that I used are hybrid keyword-data automation framework. Which means, the functionalities such as clicking element, capturing text etc has been pre-programmed and tied to specific keyword while test data can be stored outside of the script in another excel sheet.

As shown in few screenshots before, the test case looks similar to manual test case except for more emphasize on the steps and also additional automation-specific instructions such as 'wait'.

Action	LabelName	ObjectName	TestDataReference_Pays	InputData_Pays	ExpectedData_Pays	Opt_Param_Pays
waitForsec						
waittillobjexist		elmOrderConfMsg				
comment	***Step 12 Verify order summary page***					
verifyelementtextcontains	Verify Order Message	elmOrderConfMsg		Your order is successful!		h1
getObjectTextRunTime		lblOrderConf_OrdrID	order			CJ:strOrderID
scrollPage				down		
verifyobjectexist		elmAmt_NLTC01_PromoSucc				
comment	***End of the test case***					

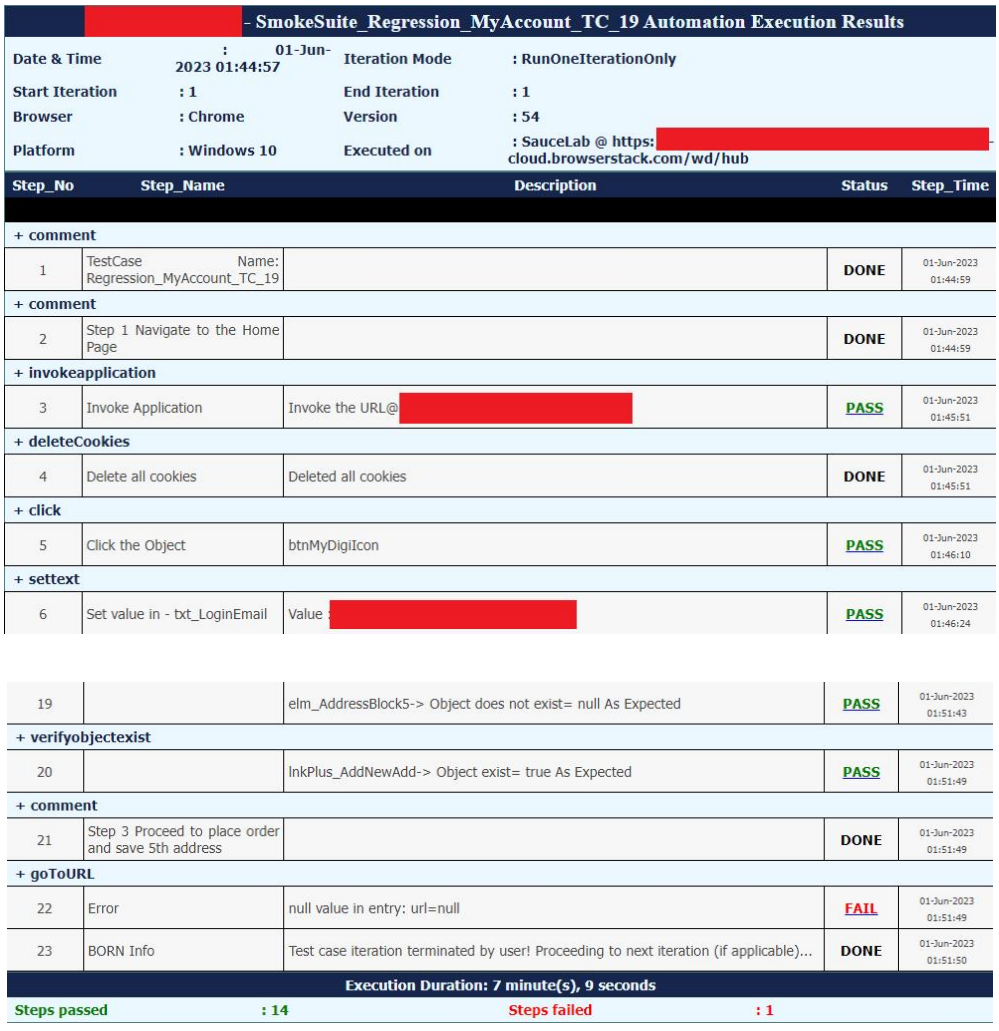
TEST REPORTING AND DEBUGGING

Finally, similar to manual testing, at the end of automation test execution there will be reporting and debugging.

But unlike manual testing where the bugs to be fixed by developer team, failure in automation testing usually relates to automation only and thus it need to be fixed by the automation test engineer only. For example, a step for clicking a home icon could be failed to be executed due to changes in the code or Xpath which will not be an issue when tested manually by human.

However in automation, it's not encouraged to add an optional conditions (eg. clicking home icon based on their image file name if their element ID are different) by implementing 'if-else' statement for example as this will lead to more time taken for the script to perform the action. This is the reason, where the stability of the SUT is the main criteria before automating the test cases.

Below is the sample of report generated automatically after the end of automation test session.



[redacted] - SmokeSuite_Regression_MyAccount_TC_19 Automation Execution Results

Date & Time	: 01-Jun-2023 01:44:57	Iteration Mode	: RunOneIterationOnly
Start Iteration	: 1	End Iteration	: 1
Browser	: Chrome	Version	: 54
Platform	: Windows 10	Executed on	: SauceLab @ https:[redacted]-cloud.browserstack.com/wd/hub

Step_No	Step_Name	Description	Status	Step_Time
+ comment				
1	TestCase Name: Regression_MyAccount_TC_19		DONE	01-Jun-2023 01:44:59
+ comment				
2	Step 1 Navigate to the Home Page		DONE	01-Jun-2023 01:44:59
+ invokeapplication				
3	Invoke Application	Invoke the URL@[redacted]	PASS	01-Jun-2023 01:45:51
+ deleteCookies				
4	Delete all cookies	Deleted all cookies	DONE	01-Jun-2023 01:45:51
+ click				
5	Click the Object	btnMyDigiIcon	PASS	01-Jun-2023 01:46:10
+ settext				
6	Set value in - txt_LoginEmail	Value :[redacted]	PASS	01-Jun-2023 01:46:24
19		elm_AddressBlock5-> Object does not exist= null As Expected	PASS	01-Jun-2023 01:51:43
+ verifyobjectexist				
20		lnkPlus_AddNewAdd-> Object exist= true As Expected	PASS	01-Jun-2023 01:51:49
+ comment				
21	Step 3 Proceed to place order and save 5th address		DONE	01-Jun-2023 01:51:49
+ goToURL				
22	Error	null value in entry: url=null	FAIL	01-Jun-2023 01:51:49
23	BORN Info	Test case iteration terminated by user! Proceeding to next iteration (if applicable)...	DONE	01-Jun-2023 01:51:50

Execution Duration: 7 minute(s), 9 seconds

Steps passed : 14 Steps failed : 1