TECHNICAL ANALYSIS OF SELENIUM AND CYPRESS AS FUNCTIONAL AUTOMATION FRAMEWORK FOR MODERN WEB APPLICATION TESTING

Fatini Mobaraya and Shahid Ali

Department of Information Technology, AGI Institute, Auckland, New Zealand

ABSTRACT

Automation testing has become increasingly needed due to the nature of the current software development project which comprises of complex application with shorter development time. Most of the companies in the industry have used Selenium extensively as functional automation tool to verify their web application's functionalities are working as expected. However, the limitation in Selenium with wait Time has significantly affect its test execution and efficiency. Thus, this research project experimenting a new automation tool in the market, Cypress, to overcome the said limitation in Selenium. This research further compares the test execution results in Selenium and Cypress to observe each tool's effectiveness in writing and executing the automation test script. The study results will be helpful towards determining a better tool in automating dynamic modern web application and providing an insight into Cypress as the future of automation testing tool.

KEYWORDS

Automation Testing, Regression Test Suite, Selenium, Cypress, JavaScript Automation Framework

1. INTRODUCTION

In the current world of technology, where every information is just a click away, web application has gained popularity since every information including the retail shops and the government infrastructures are being digitized. Web applications are becoming more complex by having multiple rich features and dynamic rendering to give a rich user experience. Due to this nature, there is a significant challenge in testing modern web application to satisfy end user's expectations.

With a complex web application and shorter product release time, automation testing came into picture. Automation testing is the uses a tool to replicate the behaviours of a real user by executing test autonomously with the goal to reduce execution time and increase test coverage.

Testing contributes 30-60% of software life cycles with a bigger percentage goes to the more complex and critical products [1]. With a shorter development time and complex web pages, modern web testing faced major development time to release with the need for fast test execution. According to [2], 29 out of 40 e-commerce sites are facing failure when being accessed by end-users.

To counter this issue, most organizations in the industry used Selenium to automate their application for functional requirements validation. However, as Selenium was developed back in 2005, the websites existed back then were much simpler than the current 2019 websites. Selenium main limitation is the difficulty in handling the current dynamic web elements, which significantly reduce test execution performance and resulting in a flaky test execution.

Thus, this research proposes Cypress as the automation framework to cater the current dynamic web applications. The main advantage of using Cypress is it simplifies asynchronous testing. Cypress defines an automatic wait in their framework where it waits for the DOM elements to finish loading or any animation to complete rendering, only then it will start looking for the web element. This is the main limitation in Selenium where automation developers need to define implicit and explicit wait to wait for the page finished loading. Testers can add waitTime or thread.sleep commands to counter this, however it will significantly affect the test execution performance.

The only drawback Cypress have is that they mainly support Chrome browser. However, according to Google Trend, as today's population is mainly using Chrome browser 90% of the time, the limitation of browser support would be a negligible drawback at this stage.

This research automates a dynamic web application – AliExpress. AliExpress is chosen as it contains various dynamic elements and rendering, as well as being accessed by over a million users worldwide, which proves it to be a reliable site.

The goals of this research are as follows:

i. To create an automation script using Selenium and Cypress.
ii. To develop a regression automation suites for AliExpress's main business flow which are customer's account and order checkout flow.
iii. To compare the test execution time between Selenium and Cypress.
iv. To compare the test efficiency and test coverage between Selenium and Cypress.

This research paper is organized as follow: Section 2 focuses on the literature review of various studies focusing on Selenium and Cypress. Section 3 is focused on the research methodology for this research. Section 4 of this research is focused on project execution. The comparative analysis of regression suite in Selenium and Cypress results are provided in section 5. Discussion to results and research findings are provided in section 6. Section 7 is dedicated towards the future work recommendations. Finally, in section 8 conclusion to the research is provided.

2. LITERATURE REVIEW

In the past, a lot of researches have been conducted to improve web application test execution by using test automation technique. One of the most common automation tools used in the past researches is Selenium. Selenium is said to be a trusted and robust automation tool due to its long existence in the market dated back to 10 years ago. However, according to [3], [4], [5] and recent studies with automation developers, Selenium possess a significant limitation in handling dynamic elements rendering, page loads and pop-up windows in the current modern web application.

This issue has not been addressed properly in the past and it affects Selenium script performance and reliability as mentioned by [6]. Thus, this research project will propose a better way of executing automation testing by deploying Cypress automation framework with Test Driven Development (TDD) approach and Page Object Model (POM) design pattern.

TDD is one of the automation testing techniques. It focuses on test-first where developers will write automated tests followed by functional code. Based on recent studies by [7] and [8], TDD implementation will produce a simple code, increase test coverage as well as its clarity and maintainability. Furthermore, [8] described that TDD improves test execution by 21% and decrease automation code complexity by 31%.

Meanwhile, POM is a high-level abstraction that separates web pages from the test cases to encourage code reusability. It reduces coupling dependency between test cases and web pages which allows them to be independent of each other and easier for reuse in other parts of the coding. Moreover, test cases are easier to write with POM implementation [3].

The combination of TDD approach and POM design pattern will produce a maintainable test scripts which will reduce cost of maintenance later on. A clean code is necessary in automation scripts as repairing a poor code implementation will cause a huge amount of resources, both in time and energy [9].

On the other hand, test metrics which are used in this research are test execution time, test efficiency based on code effort and requirement-based test coverage. These metrics will be used in this research based on the research works of [10], [11] and [12]. These three test metrics are considered to be the main measurement that will inform us the progress of the test execution. When these three metrics are properly monitored, the high quality of the Application Under Test (AUT) can be maintained.

Besides, agile methodology will be adopted to this research project. Agile is defined as an iterative incremental process where it significantly improves project timeline and increase time to release. Amongst all available agile process, scrum is the most adaptable agile framework [13]. The advantages of scrum implementation for research-oriented project including research works optimization and high-quality research results production [14]. The details of the scrum adoption into this research is described in the following section.

3. RESEARCH METHODOLOGY

This research project will adopt agile scrum project management method. While scrum is widely used in the IT industry, there are significant studies by [15], [16], [17], and [18] showing that scrum implementation in research-based project contributes to better research outcomes. Moreover, a thesis written by [16] shows implementation of agile is possible in managing a construction project.

Agile scrum is fitted into this research project as following:

- Roles: Research owner, research team and supervisor (scrum master)
- Artefacts: Research backlog and sprint backlog
- Ceremonies: weekly scrum meeting, sprint review, sprint retrospective

Justification:

As scrum master is defined to be the person who the scrum team report to, and the one who resolves any hindrance in achieving the project goal, supervisor is appointed to be scrum master. Meanwhile, product owner is defined to be research owner – the one who knows the requirements and specifications of this research. As this research project is evaluated as individual work, research owner and research team will be the same individual.

On the other hand, as daily stand up meeting is not possible in research-based project, sometimes due to different research findings as opposed to software development where they make significant changes in 24 hour period, as well as due to supervisor unavailability handling 10 or more researches at one time, weekly scrum meeting is seen to be more feasible than daily stand up.

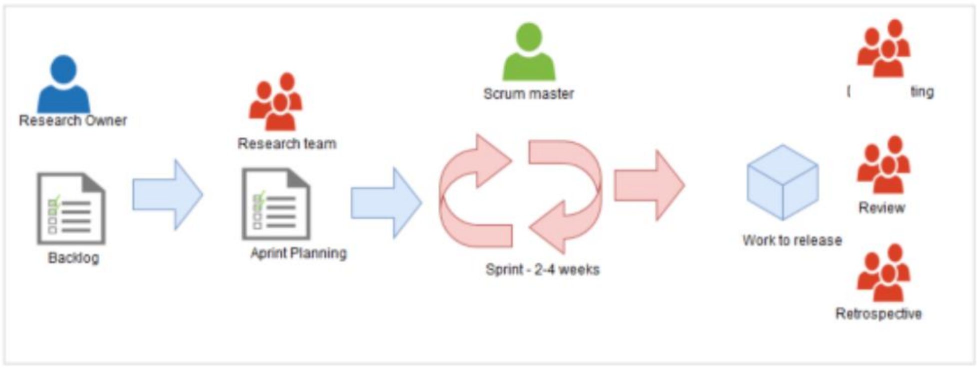


Figure 1. Research Project's Scrum Processes [15]

3.1. Selected Tool

This research uses two automation tools to develop the automation scripts which are Selenium and Cypress. Selenium is selected as it is one of the powerful and stable tools in automating web application while Cypress is selected as it offers a new way in automating modern web application. Cypress is initially a primary work of Brian Mann, a developer who felt testing dynamic websites have been tedious due to inefficient automation test execution. He then conducted a survey on the challenge's automation developers faced in testing current web application [19]. Based on the collected data, automation developers expressed that most of the debugging time was spent on synchronising wait with page loads, though the time should actually be spent on writing more test scripts. Based on these concerns, Cypress is developed and founded in year 2015.

Although there is still less published paper on Cypress due to its rather new entrance in the automation market, the statistics shown by [19] gives a promising view of Cypress's capabilities.

4. Project Execution

4.1. Test Automation Architecture

Both automation scripts in Selenium and Cypress implemented POM design pattern with slightly different style to cater to each programming language syntax used by each tool; automation scripts in Selenium is using Java while Cypress is using JavaScript. The page objects and test cases are separated in both tools to increase code maintainability and modularity. Figure 2 shows the architecture of automation scripts in Selenium and Figure 3 shows the architecture of Cypress. Cypress is made up of all libraries packaged together which made it easier for installation. It also works from within the browser and communicates directly with the Application Under Test (AUT), while Selenium instantiates a

WebDriver which acts as a third party to manipulate the user's actions on the AUT. Both architectures are depicted in the following figures.

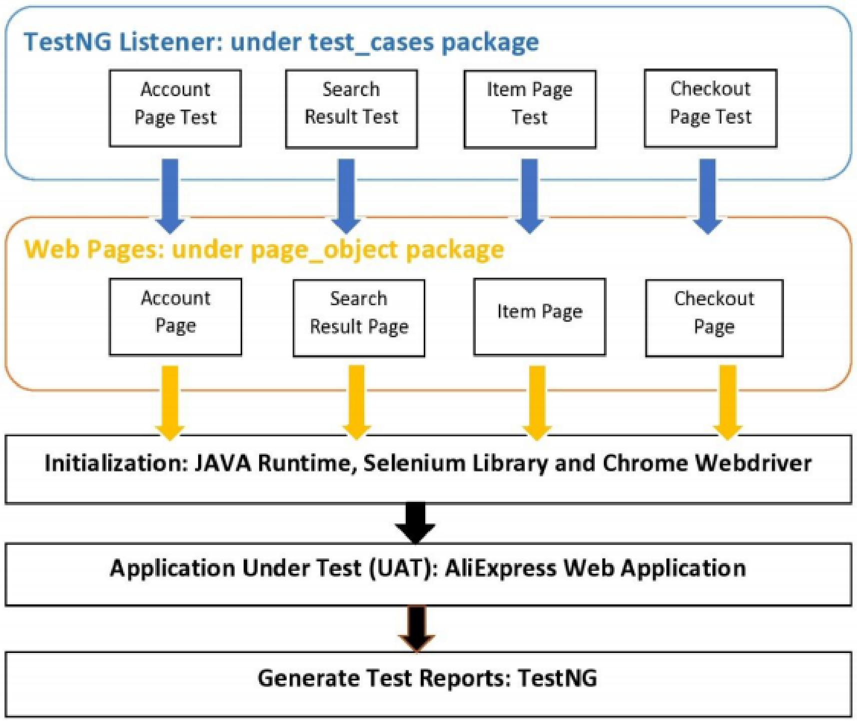

TestNG Listener: under test_cases package
Account Page Test
Search Result Test
Item Page Test
Checkout Page Test
Web Pages: under page_object package
Account Page
Search Result Page
Item Page
Checkout Page
Initialization: JAVA Runtime, Selenium Library and Chrome Webdriver
Application Under Test (UAT): AliExpress Web Application
Generate Test Reports: TestNG

Figure 2. Architecture of Selenium

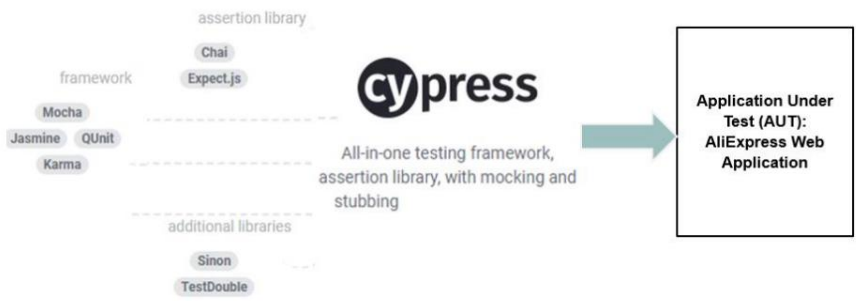
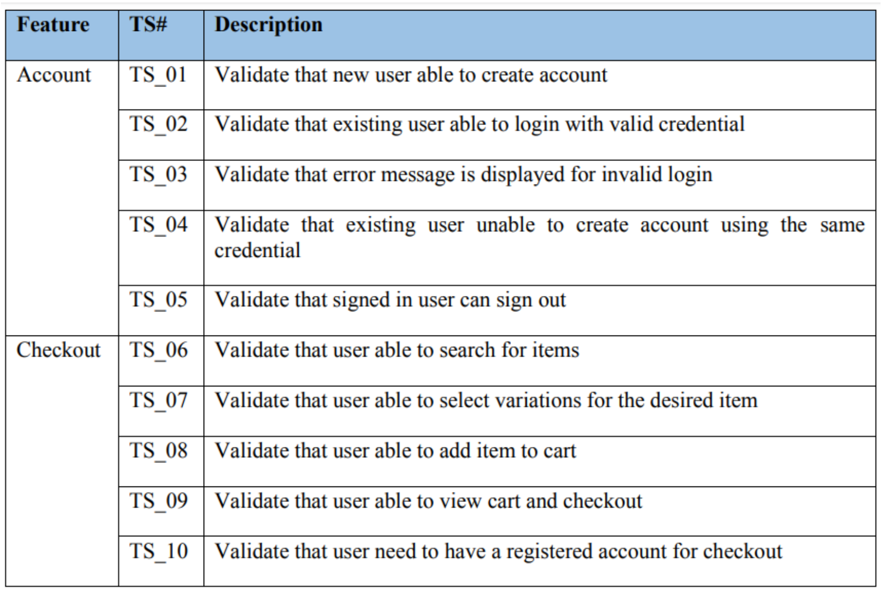

assertion library
Chai
Expect.js
framework
Mocha
Jasmine
QUnit
Karma
cypress
All-in-one testing framework, assertion library, with mocking and stubbing
additional libraries
Sinon
TestDouble
Application Under Test (AUT): AliExpress Web Application

Figure 3. Architecture of Cypress

4.2. Test Scenario

Test scenarios of this research are defined as in Table 1 below while Table 2 shows the detailed test steps for test execution. As mentioned earlier, this research will automate the two key functionalities of AliExpress, the user account and order checkout. Thus, the following test scenarios are derived to cover the functionality of each features.

Table 1. Test Scenarios



Feature	TS#	Description
Account	TS_01	Validate that new user able to create account
	TS_02	Validate that existing user able to login with valid credential
	TS_03	Validate that error message is displayed for invalid login
	TS_04	Validate that existing user unable to create account using the same credential
	TS_05	Validate that signed in user can sign out
Checkout	TS_06	Validate that user able to search for items
	TS_07	Validate that user able to select variations for the desired item
	TS_08	Validate that user able to add item to cart
	TS_09	Validate that user able to view cart and checkout
	TS_10	Validate that user need to have a registered account for checkout

Table 2. Executed Test Steps and Results

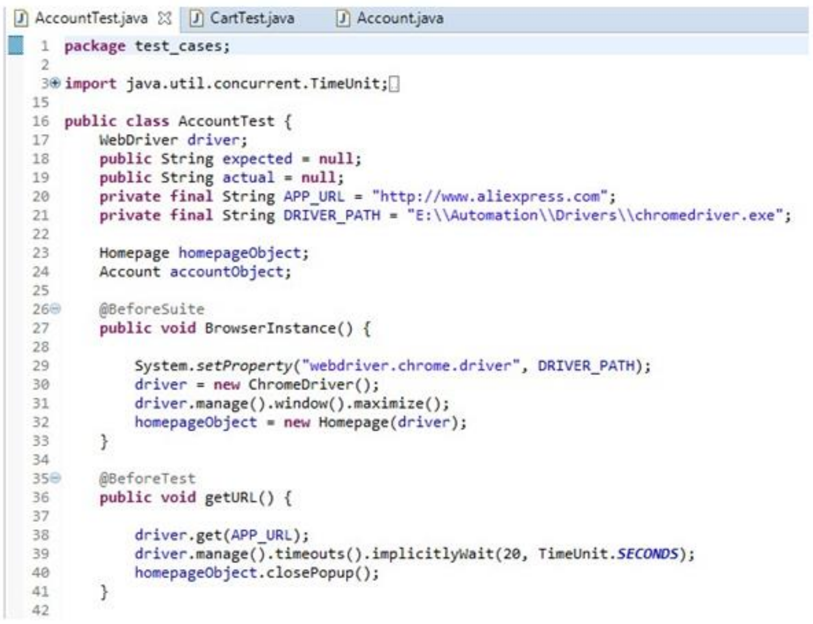
TS#	Test Steps	Test Data	Expected Result	Actual Result	Status
TS_01 - Account	1. Open browser 2. Navigate to AliExpress URL 3. Click on Join In button 4. Enter valid email and password 5. Click Create	URL: http://www.aliexp ress.com Email: era7@yahoo.com Password: Password1234	User successfully created an account and populated to Shop Now page	As expected	PASS
	Account Button				
TS_02 - Account	1. Open browser 2. Navigate to AliExpress URL 3. Click on Sign In button 4. Enter valid email and password	URL: http://www.aliexpress.com Email: era7@yahoo.com Password: Password1234	User successfully logged in and populated back to homepage	As expected	PASS

	5. Click Sign In Button				
TS_03 - Account	1. Open browser 2. Navigate to AliExpress URL 3. Click on Sign In button 4. Enter invalid email and password 5. Click Sign In Button	URL: http://www.aliexpress.com Email: er7@yahoo.com Password: 1234	"Your account name or password is incorrect." error message displayed	As expected	PASS
TS_04 - Account	1. Open browser 2. Navigate to AliExpress URL 3. Click on Join In button 4. Enter an existing email and password 5. Click Create Account Button	URL: http://www.aliexpress.com Email: era7@yahoo.com Password: Password1234	"This email already exists. Sign In >" error message is displayed	As expected	PASS
TS_05 - Account	Pre-condition: TS_02 is passed. 1. Hover to user icon at the top right hand corner of the page		User is successfully logged out and populated back to the	As expected	PASS
	2. Click Sign Out link		homepage		

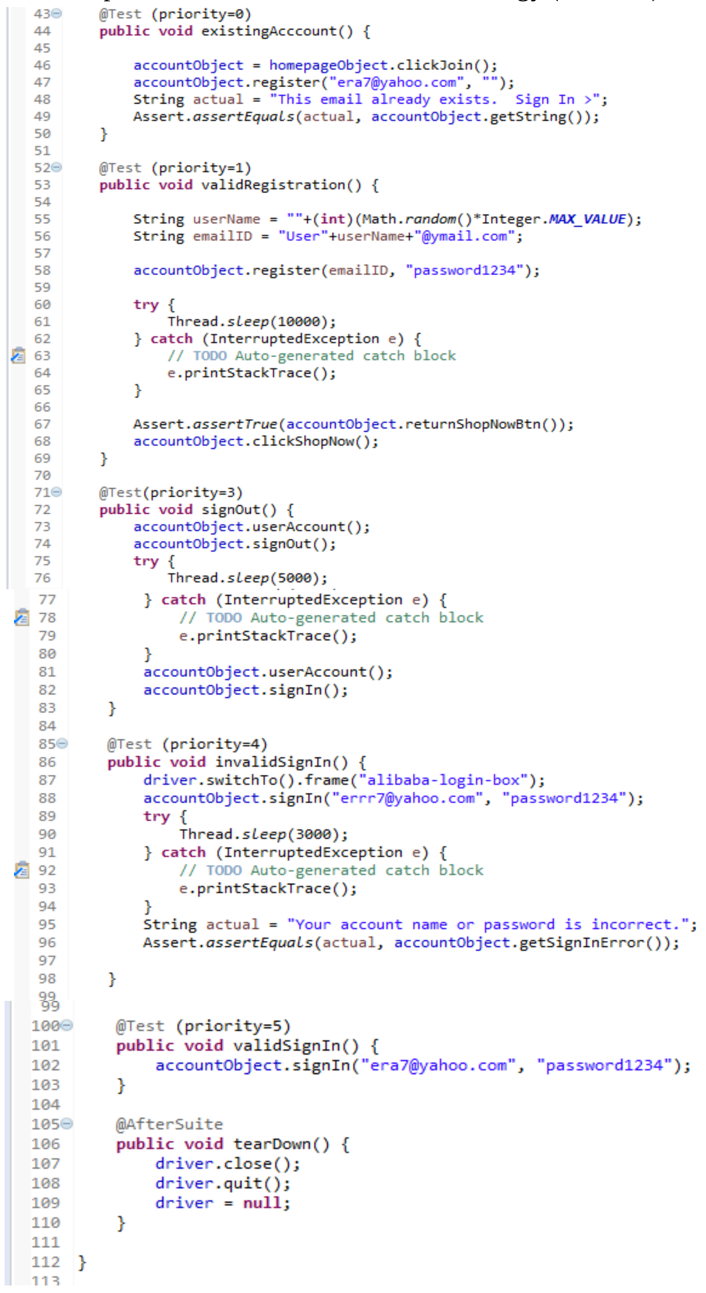
TS_06 - Checkout	Pre-condition: TS_02 is passed. 1. Open browser 2. Navigate to AliExpress URL 3. Type keyword at the search bar 4. Click search icon or press Enter	Keyword: Sweater	User successfully searched item and populated to Search Result page	As expected	PASS
TS_07 - Checkout	Pre-condition: TS_05 is passed. 1. Select the first item on the result page 2. Select any variation.		Upon clicking the item, it opens a new tab and user is populated to the item page	As expected	PASS
TS_08 - Checkout	Pre-condition: TS_06 is passed. 1. Click Add To Cart button		Item is successfully added to the cart and user is populated to recommende d items page	As expected	PASS
TS_09 - Checkout	Pre-condition: TS_06 & TC_07 are passed. 1. Click View Cart button		Item is successfully added to the cart and user is populated to Checkout page	As expected	PASS
TS_10 - Checkout	Pre-condition: TS_05 is passed. 1. Type keyword at the search bar 2. Click search icon or press Enter 3. Select the first item on the result page	Keyword: sweater	Account frame pop Out when user click Add to Cart, which insist user to have an account before proceed checkout	As expected	PASS
	4. Select any variation. 5. Click Add to Cart button				

4.3. Code Snippet

The following section includes a fraction of automation scripts of this research. Figure 4 below shows the automation script for Account regression suite in Selenium. The script starts with browser instantiation and application initialization followed by five Account page test cases: existing account, valid registration, user sign out, invalid sign in, valid sign in and lastly closing the WebDriver instance.



```java
package test_cases;

import java.util.concurrent.TimeUnit;

public class AccountTest {
    WebDriver driver;
    public String expected = null;
    public String actual = null;
    private final String APP_URL = "http://www.aliexpress.com";
    private final String DRIVER_PATH = "E:\\Automation\\Drivers\\chromedriver.exe";

    Homepage homepageObject;
    Account accountObject;

    @BeforeSuite
    public void BrowserInstance() {

        System.setProperty("webdriver.chrome.driver", DRIVER_PATH);
        driver = new ChromeDriver();
        driver.manage().window().maximize();
        homepageObject = new Homepage(driver);
    }

    @BeforeTest
    public void getURL() {

        driver.get(APP_URL);
        driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
        homepageObject.closePopup();
    }

```



```java
    @Test (priority=0)
    public void existingAcccount() {

        accountObject = homepageObject.clickJoin();
        accountObject.register("era7@yahoo.com", "");
        String actual = "This email already exists.  Sign In >";
        Assert.assertEquals(actual, accountObject.getString());
    }

    @Test (priority=1)
    public void validRegistration() {

        String userName = ""+(int)(Math.random()*Integer.MAX_VALUE);
        String emailID = "User"+userName+"@ymail.com";

        accountObject.register(emailID, "password1234");

        try {
            Thread.sleep(10000);
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        Assert.assertTrue(accountObject.returnShopNowBtn());
        accountObject.clickShopNow();
    }

    @Test(priority=3)
    public void signOut() {
        accountObject.userAccount();
        accountObject.signOut();
        try {
            Thread.sleep(5000);
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        accountObject.userAccount();
        accountObject.signIn();
    }

    @Test (priority=4)
    public void invalidSignIn() {
        driver.switchTo().frame("alibaba-login-box");
        accountObject.signIn("errr7@yahoo.com", "password1234");
        try {
            Thread.sleep(3000);
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        String actual = "Your account name or password is incorrect.";
        Assert.assertEquals(actual, accountObject.getSignInError());

    }

    @Test (priority=5)
    public void validSignIn() {
        accountObject.signIn("era7@yahoo.com", "password1234");
    }

    @AfterSuite
    public void tearDown() {
        driver.close();
        driver.quit();
        driver = null;
    }

}
```

Figure 4. Automation Script for Account Regression Suite in Selenium (112 lines)

5. Results

Test metrics serve as an important indicator to measure the progress of the automation test execution. Three metrics have been chosen to serve as an indicator of this research: total test execution time, test execution efficiency and requirement-based test coverage.

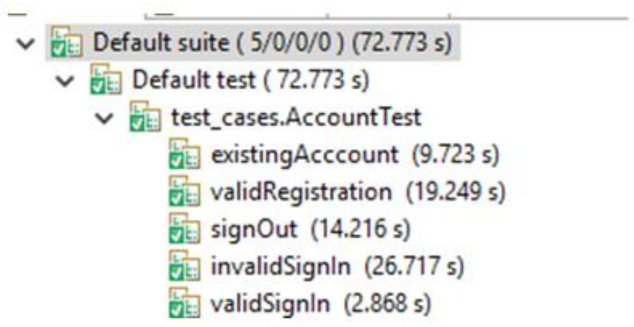
5.1. Test Execution Time

Total test execution time helps to keep track on the overall test execution progress. Figure 5 shows the time execution for each test cases in Selenium's Account regression suite while Figure 6 shows the total test execution time for the whole Account regression suite in

Selenium. Meanwhile, Figure 7 and 8 shows the same metric for Checkout regression suite in Selenium. Additionally, Figure 9 shows the total test execution time for Account regression suite in Cypress and Figure 10 shows the same test metric for Checkout regression suite in Cypress. The summary of the test execution time for both tools is depicted in the Table 3 below. A further discussion is included in the next section of this report.

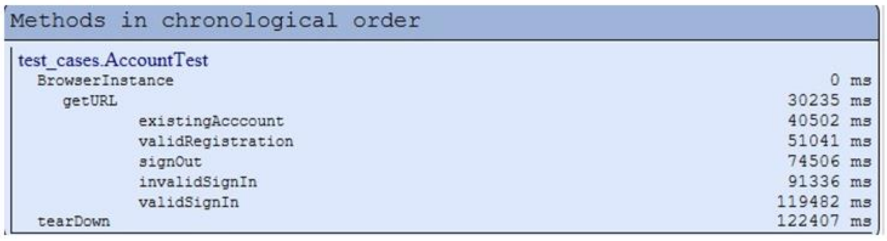
Table 3. Total Test Execution Time for Regression Suite in Selenium and Cypress

Total Execution Time (ms)	Selenium	Cypress
Account Page	122.41 ms	88.91 ms
Checkout Page	100.53 ms	80.55 ms



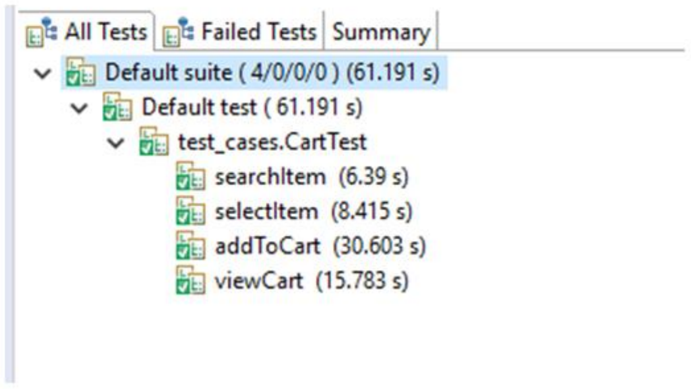
```
Default suite ( 5/0/0/0 ) (72.773 s)
  Default test ( 72.773 s)
    test_cases.AccountTest
      existingAcccount  (9.723 s)
      validRegistration  (19.249 s)
      signOut  (14.216 s)
      invalidSignIn  (26.717 s)
      validSignIn  (2.868 s)
```

Figure 5. Test Execution Time for Account Test Cases in Selenium



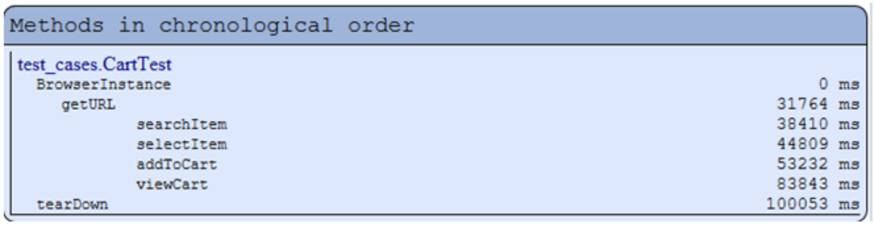
```
Methods in chronological order

test_cases.AccountTest
  BrowserInstance                                  0 ms
    getURL                                     30235 ms
          existingAcccount                     40502 ms
          validRegistration                    51041 ms
          signOut                              74506 ms
          invalidSignIn                        91336 ms
          validSignIn                         119482 ms
  tearDown                                    122407 ms
```

Figure 6. Total Test Execution Time for Account Regression Suite in Selenium



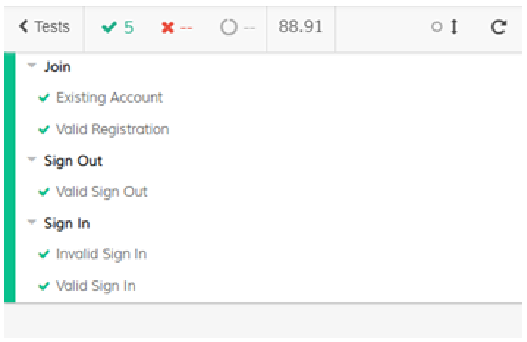
```
Default suite ( 4/0/0/0 ) (61.191 s)
  Default test ( 61.191 s)
    test_cases.CartTest
      searchItem  (6.39 s)
      selectItem  (8.415 s)
      addToCart  (30.603 s)
      viewCart  (15.783 s)
```

Figure 7. Test Execution Time for Checkout Test Cases in Selenium



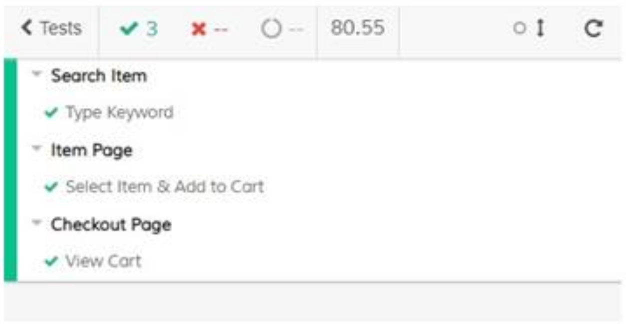
```
Methods in chronological order
test_cases.CartTest
  BrowserInstance                                              0 ms
    getURL                                                 31764 ms
          searchItem                                       38410 ms
          selectItem                                       44809 ms
          addToCart                                        53232 ms
          viewCart                                         83843 ms
  tearDown                                                100053 ms
```

Figure 8. Total Test Execution Time for Checkout Regression Suite in Selenium



```
< Tests   ✓ 5   ✖ --   ○ --   88.91
Join
  ✓ Existing Account
  ✓ Valid Registration
Sign Out
  ✓ Valid Sign Out
Sign In
  ✓ Invalid Sign In
  ✓ Valid Sign In
```

Figure 9. Total Test Execution Time for Account Regression Suite in Cypress



```
< Tests   ✓ 3   ✖ --   ○ --   80.55
Search Item
  ✓ Type Keyword
Item Page
  ✓ Select Item & Add to Cart
Checkout Page
  ✓ View Cart
```

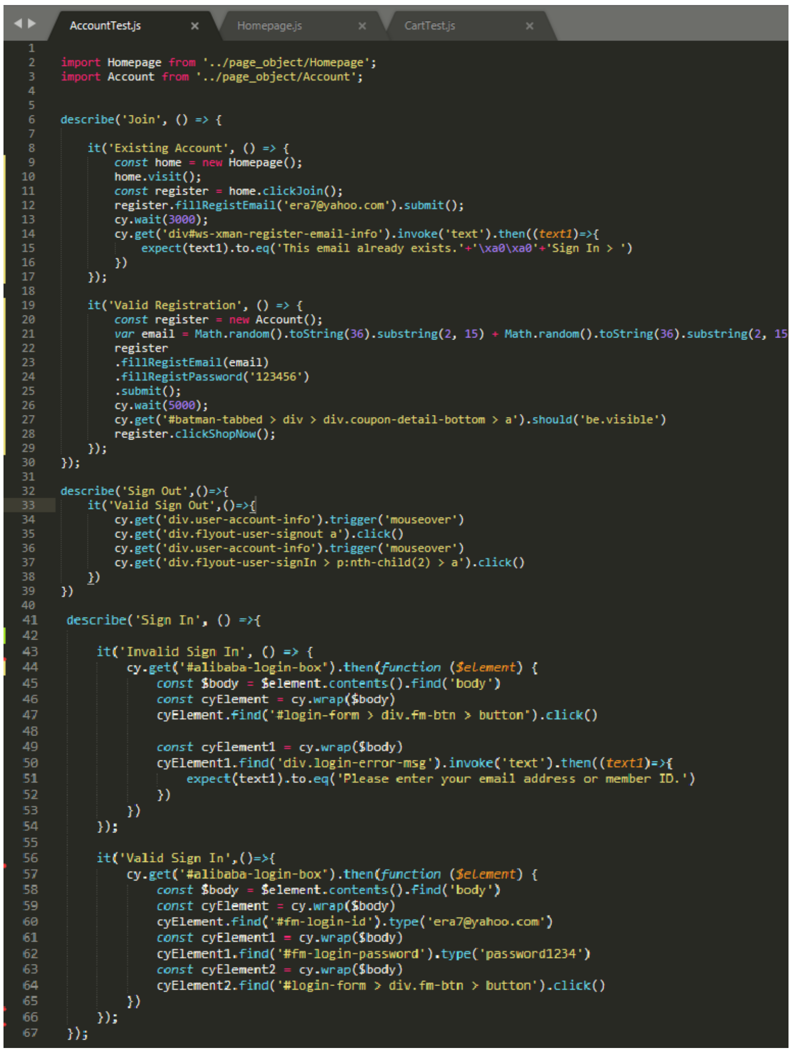
Figure 10. Total Test Execution Time for Checkout Regression Suite in Cypress

5.2. Test Efficiency

Test efficiency measures the cost-effectiveness of testing against the resources of an organization or in this research context, is the total effort in writing the automation script. The optimum test efficiency is the one that is able to reach adequate software quality standard at a lower effort. The following figures show the total line of codes for each tool's automation script. To measure this metric, total number of lines needed to be written to complete an automation scripts in Selenium and in Cypress is being tabulated as in Table 4 below.

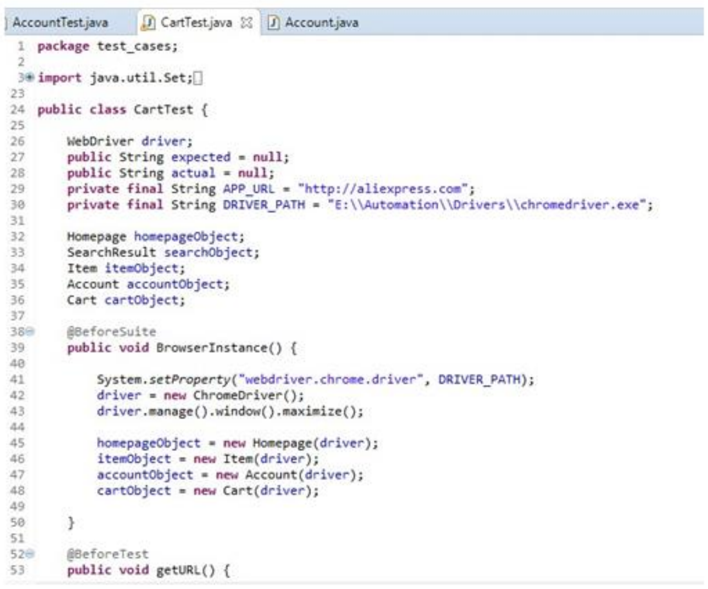
Table 4. Total Lines of Automation Script Code in Selenium and Cypress

Total Lines of Code	**Selenium**	**Cypress**
Account Page	112	67
Checkout Page	118	49

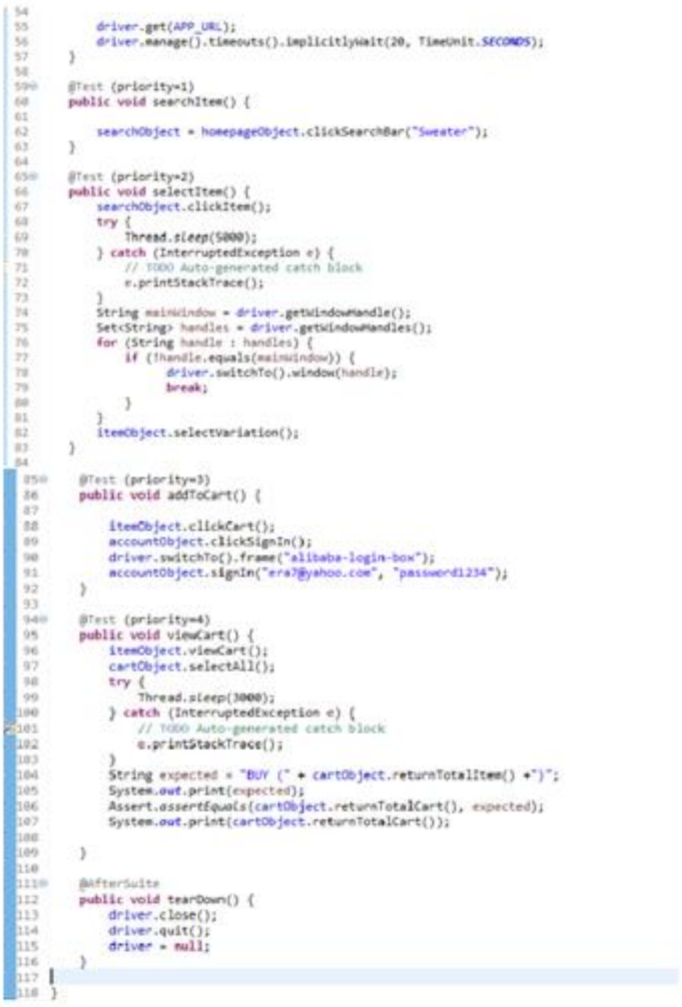


```
AccountTest.js    Homepage.js    CartTest.js

import Homepage from '../page_object/Homepage';
import Account from '../page_object/Account';


describe('Join', () => {

    it('Existing Account', () => {
        const home = new Homepage();
        home.visit();
        const register = home.clickJoin();
        register.fillRegistEmail('era7@yahoo.com').submit();
        cy.wait(3000);
        cy.get('div#ws-xman-register-email-info').invoke('text').then((text1)=>{
            expect(text1).to.eq('This email already exists.'+'\xa0\xa0'+'Sign In > ')
        })
    });

    it('Valid Registration', () => {
        const register = new Account();
        var email = Math.random().toString(36).substring(2, 15) + Math.random().toString(36).substring(2, 15
        register
        .fillRegistEmail(email)
        .fillRegistPassword('123456')
        .submit();
        cy.wait(5000);
        cy.get('#batman-tabbed > div > div.coupon-detail-bottom > a').should('be.visible')
        register.clickShopNow();
    });
});

describe('Sign Out',()=>{
    it('Valid Sign Out',()=>{
        cy.get('div.user-account-info').trigger('mouseover')
        cy.get('div.flyout-user-signout a').click()
        cy.get('div.user-account-info').trigger('mouseover')
        cy.get('div.flyout-user-signIn > p:nth-child(2) > a').click()
    })
})

 describe('Sign In', () =>{

     it('Invalid Sign In', () => {
         cy.get('#alibaba-login-box').then(function ($element) {
             const $body = $element.contents().find('body')
             const cyElement = cy.wrap($body)
             cyElement.find('#login-form > div.fm-btn > button').click()

             const cyElement1 = cy.wrap($body)
             cyElement1.find('div.login-error-msg').invoke('text').then((text1)=>{
                 expect(text1).to.eq('Please enter your email address or member ID.')
             })
         })
     });

     it('Valid Sign In',()=>{
         cy.get('#alibaba-login-box').then(function ($element) {
             const $body = $element.contents().find('body')
             const cyElement = cy.wrap($body)
             cyElement.find('#fm-login-id').type('era7@yahoo.com')
             const cyElement1 = cy.wrap($body)
             cyElement1.find('#fm-login-password').type('password1234')
             const cyElement2 = cy.wrap($body)
             cyElement2.find('#login-form > div.fm-btn > button').click()
         })
     });
 });
```

Figure 11. Account Regression Suite in Cypress (67 lines)

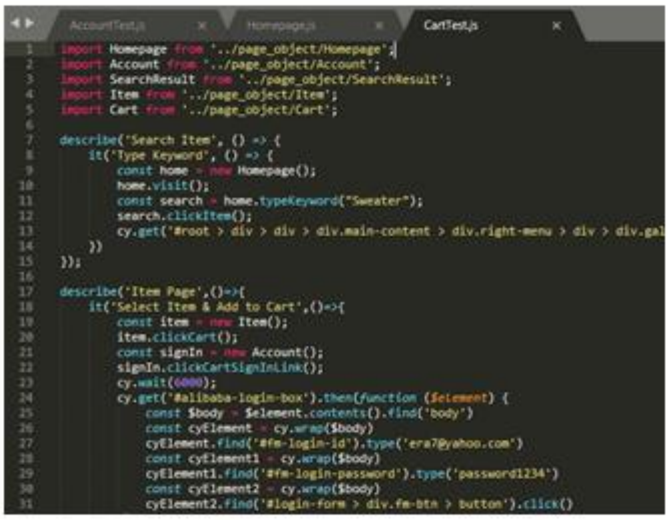


```
AccountTest.java    CartTest.java    Account.java
package test_cases;

import java.util.Set;

public class CartTest {

    WebDriver driver;
    public String expected = null;
    public String actual = null;
    private final String APP_URL = "http://aliexpress.com";
    private final String DRIVER_PATH = "E:\\Automation\\Drivers\\chromedriver.exe";

    Homepage homepageObject;
    SearchResult searchObject;
    Item itemObject;
    Account accountObject;
    Cart cartObject;

    @BeforeSuite
    public void BrowserInstance() {

        System.setProperty("webdriver.chrome.driver", DRIVER_PATH);
        driver = new ChromeDriver();
        driver.manage().window().maximize();

        homepageObject = new Homepage(driver);
        itemObject = new Item(driver);
        accountObject = new Account(driver);
        cartObject = new Cart(driver);

    }

    @BeforeTest
    public void getURL() {
```

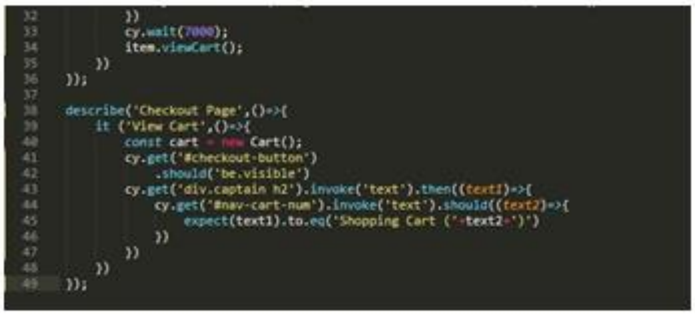


```java

        driver.get(APP_URL);
        driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
    }

    @Test (priority=1)
    public void searchItem() {

        searchObject = homepageObject.clickSearchBar("Sweater");
    }

    @Test (priority=2)
    public void selectItem() {
        searchObject.clickItem();
        try {
            Thread.sleep(5000);
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        String mainWindow = driver.getWindowHandle();
        Set<String> handles = driver.getWindowHandles();
        for (String handle : handles) {
            if (!handle.equals(mainWindow)) {
                driver.switchTo().window(handle);
                break;
            }
        }
        itemObject.selectVariation();
    }

     @Test (priority=3)
     public void addToCart() {

         itemObject.clickCart();
         accountObject.clickSignIn();
         driver.switchTo().frame("alibaba-login-box");
         accountObject.signIn("era7@yahoo.com", "password1234");
     }

     @Test (priority=4)
     public void viewCart() {
         itemObject.viewCart();
         cartObject.selectAll();
         try {
             Thread.sleep(3000);
         } catch (InterruptedException e) {
             // TODO Auto-generated catch block
             e.printStackTrace();
         }
         String expected = "BUY (" + cartObject.returnTotalItem() +")";
         System.out.print(expected);
         Assert.assertEquals(cartObject.returnTotalCart(), expected);
         System.out.print(cartObject.returnTotalCart());

     }

     @AfterSuite
     public void tearDown() {
         driver.close();
         driver.quit();
         driver = null;
     }

 }
```

Figure 12. Checkout Regression Suite in Selenium (118 lines)



```javascript
import Homepage from '../page_object/Homepage';
import Account from '../page_object/Account';
import SearchResult from '../page_object/SearchResult';
import Item from '../page_object/Item';
import Cart from '../page_object/Cart';

describe('Search Item', () => {
    it('Type Keyword', () => {
        const home = new Homepage();
        home.visit();
        const search = home.typeKeyword("Sweater");
        search.clickItem();
        cy.get('#root > div > div > div.main-content > div.right-menu > div > div.gal
    })
});

describe('Item Page',()=>{
    it('Select Item & Add to Cart',()=>{
        const item = new Item();
        item.clickCart();
        const signIn = new Account();
        signIn.clickCartSignInLink();
        cy.wait(6000);
        cy.get('#alibaba-login-box').then(function ($element) {
            const $body = $element.contents().find('body')
            const cyElement = cy.wrap($body)
            cyElement.find('#fm-login-id').type('era7@yahoo.com')
            const cyElement1 = cy.wrap($body)
            cyElement1.find('#fm-login-password').type('password1234')
            const cyElement2 = cy.wrap($body)
            cyElement2.find('#login-form > div.fm-btn > button').click()
```



```
            })
            cy.wait(7000);
            item.viewCart();
        })
    });

    describe('Checkout Page',()=>{
        it ('View Cart',()=>{
            const cart = new Cart();
            cy.get('#checkout-button')
                .should('be.visible')
            cy.get('div.captain h2').invoke('text').then((text1)=>{
                cy.get('#nav-cart-num').invoke('text').should((text2)=>{
                    expect(text1).to.eq('Shopping Cart ('+text2+')')
                })
            })
        })
    });
```

Figure 13. Checkout Regression Suite in Cypress (47 lines)

5.3. Requirement-based Test Coverage

Requirement-based test coverage is measured against the number of requirements that have been covered by the test cases. This metric determines the thoroughness of the testing towards the Application Under Test (AUT). If the test coverage isn't 100%, that means that there are holes in the testing and require more test cases to be added to include all requirements.

Requirement-based test coverage can be easily measured by generating a traceability matrix. Traceability matrix gives an overview of the overall mapped requirements to test cases. Thus, ensuring all requirements have been covered during test execution phase. Furthermore, by having requirement-based test coverage, any unnecessary and redundant test cases are eliminated as test cases are derived based on the requirement. Table 5 shows the requirement traceability matrix for this research project. The *Defects* field is left empty as no defects are found in the system.

Table 5. Traceability Matrix of Requirement to Test Cases

Requirements	**Tests**	**Execution**	**Defects**
User Account User able to create account and sign in by having a valid credential. Error message should be displayed accordingly for invalid attempt of register and sign in. A logged in user also should be able to sign out.	TC_01	PASS	
	TC_02	PASS	
	TC_03	PASS	
	TC_04	PASS	
	TC_05	PASS	
Order Checkout User able to search for item, select and add desired items to cart, view cart and checkout. User shall have a registered account for checkout. The system shall insist for an account registration or account login at a point when user attempts to add items to cart without a logged in account.	TC_06	PASS	
	TC_07	PASS	
	TC_08	PASS	
	TC_09	PASS	
	TC_10	PASS	

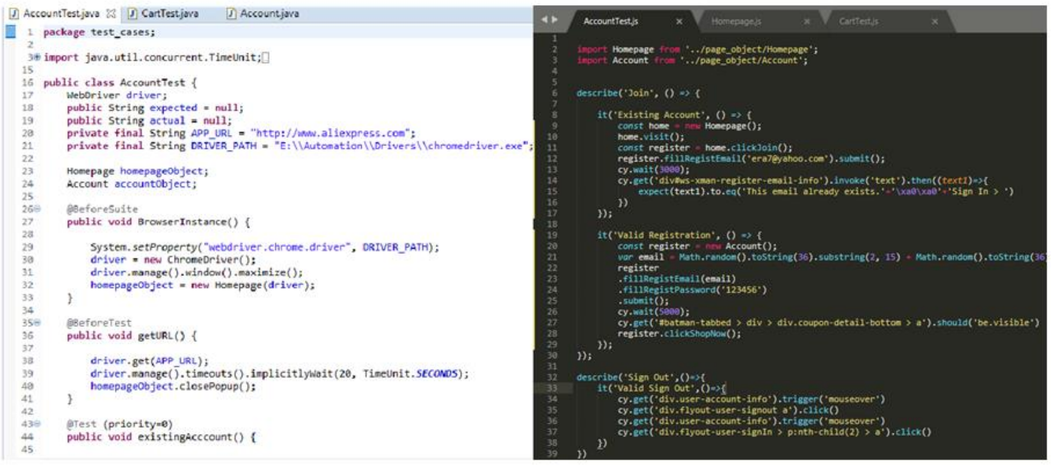
6. DISCUSSION

6.1. Observation on Test Metrics

From the test metrics collected, it is found that the total test execution time does not have much difference between Selenium and Cypress. Based on Table 3, there is only 33.51ms differences in the total test execution time for Account regression suite between Selenium and Cypress. While 19.98ms differences for Checkout regression suite. Most of the execution time consumed in Selenium is used to instantiate the WebDriver and initialize the application. Selenium does have fast execution for each test cases (as depicted in Figure 5 and Figure 7). However, the whole regression suite execution time is slowed down by the browser initialization which contributes to the biggest number of the test execution time.

On the other hand, there is a significant difference for test efficiency between Selenium and Cypress. Based on Table 4, Cypress produces 45 lesser lines of code compared to Selenium for Account regression suite. The same situation is observed for Checkout regression suite where Cypress produces 69 lesser lines of code compared to Selenium's automation script. This indicates less effort is needed to write automation script in Cypress which significantly increase its test efficiency compared to Selenium.

As an example, Figure 14 below compares the differences between the same number of lines of code (42 lines) written in Selenium and Cypress. Most of the codes in Selenium need to be instantiated with libraries and import pages, whereas Cypress directly cater the main part of the automation script. For instance, up until line 42, Selenium is still instantiating the setup of WebDriver and importing libraries, whereas Cypress has already covered three test cases by line 39.



```java
 1  package test_cases;
 2  
 3  import java.util.concurrent.TimeUnit;
15  
16  public class AccountTest {
17      WebDriver driver;
18      public String expected = null;
19      public String actual = null;
20      private final String APP_URL = "http://www.aliexpress.com";
21      private final String DRIVER_PATH = "E:\\Automation\\Drivers\\chromedriver.exe";
22  
23      Homepage homepageObject;
24      Account accountObject;
25  
26      @BeforeSuite
27      public void BrowserInstance() {
28  
29          System.setProperty("webdriver.chrome.driver", DRIVER_PATH);
30          driver = new ChromeDriver();
31          driver.manage().window().maximize();
32          homepageObject = new Homepage(driver);
33      }
34  
35      @BeforeTest
36      public void getURL() {
37  
38          driver.get(APP_URL);
39          driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);
40          homepageObject.closePopup();
41      }
42  
43      @Test (priority=0)
44      public void existingAcccount() {
45  
```

```javascript
 1  
 2  import Homepage from '../page_object/Homepage';
 3  import Account from '../page_object/Account';
 4  
 5  
 6  describe('Join', () => {
 7  
 8      it('Existing Account', () => {
 9          const home = new Homepage();
10          home.visit();
11          const register = home.clickJoin();
12          register.fillRegistEmail('era7@yahoo.com').submit();
13          cy.wait(3000);
14          cy.get('div#ws-xman-register-email-info').invoke('text').then((text1)=>{
15              expect(text1).to.eq('This email already exists.'+'\xa0\xa0'+'Sign In > ')
16          })
17      });
18  
19      it('Valid Registration', () => {
20          const register = new Account();
21          var email = Math.random().toString(36).substring(2, 15) + Math.random().toString(36
22          register
23          .fillRegistEmail(email)
24          .fillRegistPassword('123456')
25          .submit();
26          cy.wait(5000);
27          cy.get('#batman-tabbed > div > div.coupon-detail-bottom > a').should('be.visible')
28          register.clickShopNow();
29      });
30  });
31  
32  describe('Sign Out',()=>{
33      it('Valid Sign Out',()=>{
34          cy.get('div.user-account-info').trigger('mouseover')
35          cy.get('div.flyout-user-signout a').click()
36          cy.get('div.user-account-info').trigger('mouseover')
37          cy.get('div.flyout-user-signIn > p:nth-child(2) > a').click()
38      })
39  })
```

Figure 14. Comparison of Automation Script in Selenium and Cypress

This shows that Cypress provides a better test efficiency compared to Selenium as Cypress takes shorter lines of code needs to write in order to complete a test scenario. Moreover, it will indirectly improve the efficiency of the effort in writing the test automation scripts.

While, for the last test metrics measured which is requirement-based test coverage, both tools able to provide 100% test coverage. Based on the requirement traceability matrix in Table 5, all the requirements are 100% covered by the test cases in Selenium and Cypress. This indicates the automation scripts written in both tools are able to execute the designed test cases accordingly.

6.2. Limitations and Workaround

As AliExpress is a dynamic web application, there are several limitations found in both tools while automating the web application. These limitations are seen to be an advantage as it serves the purpose of this research to test a challenging web application which consists a variety of dynamic elements. The significant limitation found in Selenium and Cypress while automating AliExpress is the difficulty in handling the opening of a new tab. In AliExpress, whenever a user clicked on an item, the browser will trigger a new tab handling and open the content of the item in the new tab.
The workaround for this problem is slightly different for both tools. In Selenium, new tab handling can be easily overcome by switching the tab using the keyboard commands as Selenium support multiple tab handling. However, the workaround for Cypress is trickier as Cypress doesn't support multiple tab handling. New tab handling is normally triggered when the element's attribute is set to blank target. Thus, by removing the blank target attribute, it will open the content in the same tab when user clicks on the link, which solves the issue in Cypress.

The other limitations found in Cypress are:

- Restricted web security access
- Difficulty in interacting with elements hidden in iFrames

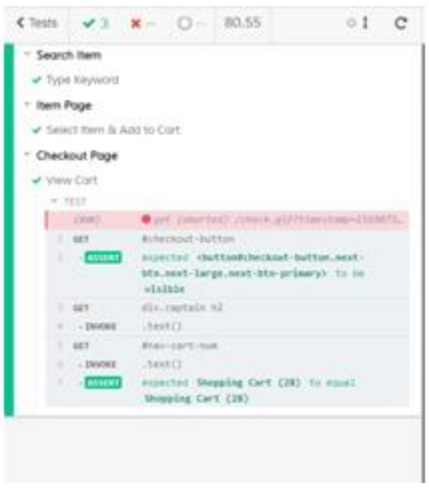
As Cypress acts as the browser itself, most browser will restrict access as browsers adhere to strict same-origin security policy. Thus, this resulting in several links unable to be loaded due to Chrome security commands. By setting the *chromeWebSecurity* to false, it allows Cypress to display insecure content and cross-origin iframes, which works as the workaround for both issues.

6.3. Other Findings

.During this research, it is also found that test cases in Selenium is executed in alphabetical order if test case prioritization is not being assigned. Whereas in Cypress, test cases are executed in sequential order line by line. The test case written at the top will be executed first followed by the rest of the codes.
In term of code readability, automation script in Cypress is more readable than in Selenium for its shorter commands. However, it takes a more technical person to understand the commands used in Cypress whereas Selenium commands is easier to understand.
For an example, to verify the visibility of an element in Selenium, the following commands is needed:

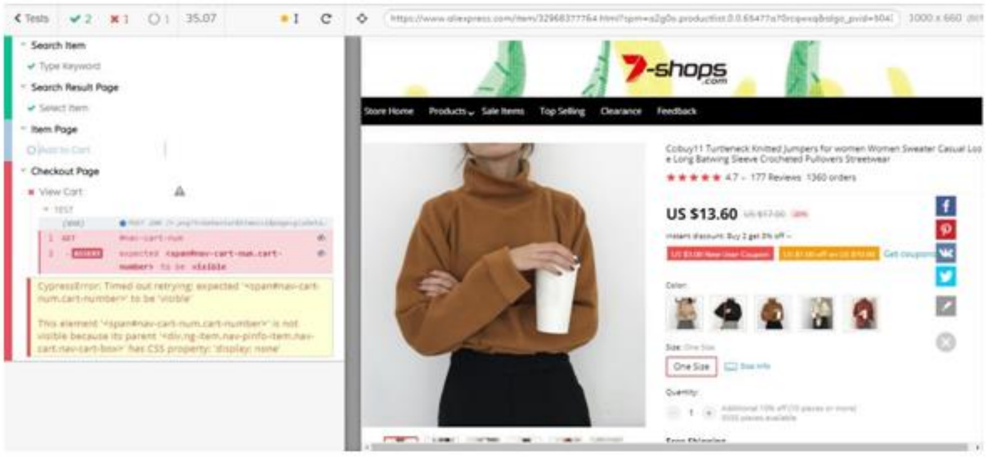
- WebElement element = driver.findElement(By.class("checkout-button"));
- element.isDisplayed();
  Whereas in Cypress, the commands are shorter and simpler, but is more technical to be understood:
- cy.get('#checkout-button').should('be.visible')

Moreover, the intuitive interface of Cypress made it easy to keep track of the test execution. Figure 15 and Figure 16 show the interface of Cypress where the execution progress is displayed on the left hand side of the frame while the AUT is displayed on the right hand side of the frame. This eases the test execution monitoring as tester will be able to see the commands execution as well as the AUT side by side. Figure 15 shows an assertion which includes the expected and actual result without the need to print the output. Finally, Figure 16 shows an example of failed test cases in Cypress where it eases the tester to debug the code as Cypress points exactly on the problem in the code.



```
< Tests   ✓ 3   ✗ --   ○ --   80.55
  Search Item
    ✓ Type Keyword
  Item Page
    ✓ Select Item & Add to Cart
  Checkout Page
    ✓ View Cart
      TEST
        (XHR)      [illegible]
        GET        #checkout-button
        - ASSERT   expected <button#checkout-button.next-btn.next-large.next-btn-primary> to be visible
        GET        div.captain h2
        - INVOKE   .text()
        GET        #nav-cart-num
        - INVOKE   .text()
        - ASSERT   expected Shopping Cart (28) to equal Shopping Cart (28)
```

Figure 15. Example of Passed Assertion for Checkout Regression Suite in Cypress



```
< Tests   ✓ 2   ✗ 1   ○ 1   35.07
  Search Item
    ✓ Type Keyword
  Search Result Page
    ✓ Select Item
  Item Page
    ○ Add to Cart
  Checkout Page
    ✗ View Cart
      TEST
        (XHR)      [illegible]
        1 GET      #nav-cart-num
        2 - ASSERT expected <span#nav-cart-num.cart-number> to be visible
  CypressError: Timed out retrying: expected '<span#nav-cart-num.cart-number>' to be 'visible'

  This element '<span#nav-cart-num.cart-number>' is not visible because its parent '<div.ng-item.nav-pinfo-item.nav-cart.nav-cart-box>' has CSS property: 'display: none'
```

Figure 16. Example of Failed Test Cases in Cypress

7. CONCLUSION

It has been a good experience exploring and expanding automation knowledge in other automation tool other than Selenium. Selenium is undeniably a powerful tool due to its huge community and support as it has been on the market for many years. However, Cypress also gives a promising view of how the future of the automation testing will be. It significantly eases and simplifies the automation configuration processes and produces a better and cleaner code.

With the right amount of resources and support, Cypress can be used to achieve much more. As this research relies heavily on StakeOverFlow, GitHub and Cypress official website to develop the automation scripts in Cypress, it might not be the best industry practice yet as it is conducted on the basis of self-study. It is believed that with the right mentoring, Cypress is a powerful tool in testing the ever changing and complex modern dynamic web application.

8. RECOMMENDATIONS

Several recommendations are suggested to improve and further extend this research project for future works such as:

I. Extend the scripts to cover utility functionalities of AliExpress: Message to seller (where buyer can ask for more details with seller), save item to watch list, history of purchase and refund, and the return policy.
II. ii.Generate a HTML test report in Selenium with Extent Report framework and Mochawesome framework for Cypress, to make it easier for people to understand the test execution status and progress.
III. iii.As Cypress has released a new version where it supported more browsers than Chrome, it is recommended to extend the scripts for other popular browser such as Mozilla and Opera, to cover wider users.
IV. iv.Refractor the code for unnecessary commands to increase code maintainability.

REFERENCES

[1] Polo, M., Reales, P., Piattini, M., & Ebert, C. (2013). Test Automation. IEEE Software', 84-89.

[2] Al-Zain, S., Eleyan, D., & Hassouneh, Y. (2013). Comparing GUI Automation Testing Tools for Dynamic Web Applications. Asian Journal of Computer and Information Systems, 38-48.

[3] Vila, E., Novakova, G., & Todorova, D. (2017). Automation Testing Framework for Web Applications with Selenium WebDriver: Opportunities and Threats. Proceedings of the International Conference on Advances in Image Processing (pp. 144-150). New York: ACM.

[4] Bulla, A., & S, B. (2016). A Study: Automation Technique Using Selenium Web driver. International Journal of Research, 467-470.

[5] Mane, D., Bhadekar, G., & Salukhe, S. (2016). Text and Keyword Driven Automation Testing using Selenium Web Driver. International Research Journal of Engineering and Technology, 515-519.

[6] Colantonio, J. (2014). The #1 Killer of Selenium Script Performance and Reliability. Retrieved from Joe Colantonio: Automation Awesomeness: https://www.joecolantonio.com/selenium-performance-reliability/

[7] Moe, M. M. (2019). Comparative Study of Test-Driven Development TDD, Behavior-Driven Development BDD and Acceptance Test–Driven Development ATDD. International Journal of Trend in Scientific Research and Development, 231-234.

[8] Khanam, Z., & Ahsan, M. (2017). Evaluating the Effectiveness of Test Driven Development: Advantages and Pitfalls. International Journal of Applied Engineering Research, 7705-7716.

[9] Leotta, M., Clerissi, D., Ricca, F., & Spadaro, C. (2013). Comparing the maintainability of selenium WebDriver test suites employing different locators: a case study. Proceedings of the 2013 International Workshop on Joining Academia and Industry Contributions to testing Automation (pp. 53-58). Lugano: ACM.

[10] Damm, L.O. (2002). Evaluating and Improving Test Efficiency. Ronneby: Citeseerx.

[11] Shahid, M., & Ibrahim, S. (2011). An Evaluation of Test Coverage Tools in Software Testing. International Conference on Telecommunication Technology and Applications (pp. 216-222). Kuala Lumpur: IACSIT Press.

[12] Walia, M. (2012). Realizing Efficiency & Effectiveness in Software Testing through a Comprehensive Metrics Model. Bangalore: Infosys.

[13] Hidalgo, E. S. (2019). Adapting the scrum framework for agile project management in science: case study of a distributed research initiative. Heliyon, 1-32.

[14] Hicks, M., & Foster, J. S. (2010). Adopting Scrum to Managing a Research Group.

[15] Rodriguez, D. G. (2016). Using Scrum in your research. Crossroads: The ACM Magazine for Students, pp. 1-5.

[16] Pareliya, M. (2018). Implementing Agile Project Management (Scrum) Approach in the Development of Building Projects. CEPT University, Ahmedabad, India.

[17] Marchesi, M., Mannaro, K., Uras, S., & Locci, M. (2007). Distributed Scrum in Research Project Management. 8th international conference on Agile processes in software engineering and extreme programming (pp. 240-244). Como: Springer.

[18] Hidalgo, E. S. (2018). Management of a Multidisciplinary Research Project: Journal of Research Practice, 1-17.

[19] Bharadwaj, S. (2018). Why Should You Switch to Cypress for Modern Web Testing? Retrieved from DZone DevOps: https://dzone.com/articles/why-should-you-switch-to-cypress-for-modern-web-te?fromrel=true

AUTHORS

Fatini Mobaraya graduated from University of Technology Malaysia in Bachelor of Computer Science (Software Engineering) with Honours. Her first exposure to software testing is during one of the modules back in her bachelor's degree: Software Quality & Assurance. She never looked back ever since and have found passion in software quality. She then pursued a specialized testing course, Graduate Diploma in Software Testing at AGI Institute, New Zealand; to learn in depth about testing and being a tester who can code. She is interested in projects that comprised of automation test engineering, regression testing and performance engineering.

Dr. Shahid Ali is a senior lecturer and IT program leader at AGI Education Limited, Auckland, New Zealand. He has published number of research papers on ensemble learning. His expertise and research interests include ensemble learning, machine learning, data mining and knowledge discovery.