

Evaluating Test Automated Tools

by

Bindu Latha Kondreddi
kondreddib15@students.ecu.edu
Masters in Computer Science
East Carolina University
Spring-2016

Advisor

Dr. Mark Hills

Committee Members

Dr. M. H. Nassehzadeh Tabrizi
Dr. Sergiy Vilkomir
Dr. Mark Hills

Abstract

Testing plays a significant role in the software development cycle because it helps to improve the reliability, dependability and quality of software by detecting unexpected bugs or errors in it. In an era of highly interactive and responsive software processes where many organizations are using some form of agile methodology, automated testing is frequently becoming a requirement for web based projects. Automated testing means using a software tool to run repeatable tests against the application to be tested. Through automated testing we can save both money and time with less manual intervention. Main objective of the project is to understand & evaluate features of three different web browser Automated testing tools available in the market. A shopping cart web application has been created which supports registration, login, add to cart and logout functionalities. The evaluation has been done by testing those functionalities using three different automated testing tools Selenium, CasperJs, and Sahi and comparing them with each other.

Contents

Abstract	ii
Chapter 1: Introduction	1
1.1: Why Automated testing?	1
1.2: Advantages of Automated testing.....	1
Chapter 2: Testing Approach.....	2
2.1: Web Application.....	2
2.2: FirePath	6
2.3: Automated testing Tools Used:.....	6
2.3.1: CasperJs	6
2.3.2: Sahi.....	6
2.3.3: Selenium.....	8
2.3.3.1: Selenium IDE	8
2.3.3.2: Selenium WebDriver	11
2.4: Evaluation:	14
2.5: Implementation	37
2.5.1: Success Test Cases.....	37
2.5.1.1: CasperJs.....	37
2.5.1.2: Sahi	41
2.5.1.3: Selenium	45
2.5.2: Failure Test Cases.....	49
2.5.2.1: Test case 1	49
2.5.2.2: Test case 2	50
2.5.2.3: Test case 3	51
2.5.2.4: Test case 4	52
2.5.2.5: Test case 5	52
Chapter 3: Conclusion.....	53
References	54
Appendix.....	54

List of Figures

Figure 1: Soccer homepage.....	3
Figure 2: Register/Login	4
Figure 3: Add to cart	5
Figure 4: Logout	5
Figure 5: Sahi.....	7
Figure 6: Selenium.....	8
Figure 7:Selenium IDE	9
Figure 8: Selenium 2.0.....	11
Figure 9: Selenium Framework Execution Process	12
Figure 10:CasperJs register	37
Figure 11:CasperJs screenshot 1.....	38
Figure 12:CasperJs execution	38
Figure 13: CasperJs screenshot 2.....	39
Figure 14: CasperJs screenshot 3.....	40
Figure 15: CasperJs screenshot 4.....	41
Figure 16: CasperJs screenshot 5.....	41
Figure 17:Sahi Register	42
Figure 18:Sahi Login	43
Figure 19:Logs	44
Figure 20:Selenium IDE register.....	45
Figure 21:Selenium IDE login	46
Figure 22:Selenium 2.0 register	47
Figure 23:Selenium 2.0 login.....	48
Figure 24:Selenium 2.0 add to cart.....	49
Figure 25:Test case 1.....	50
Figure 26: Test case 2.....	51
Figure 27: Test case 3.....	51
Figure 28: Test case 4.....	52
Figure 29: Test case 5.....	52

Chapter 1: Introduction

Main objective of the project is to understand and evaluate features of different web browser automated testing tools available in the market. Selenium, CasperJs and Sahi Automated testing tools have been chosen for evaluation. A shopping cart web application has been created for above three automated testing tools evaluation. Shopping cart application supports registration, login and add cart functionalities. Test scripts have been created in Selenium, CasperJs and Sahi tools to support automated testing of shopping cart application and evaluation.

1.1: Why automated testing?

In today's world, it is a challenge for any company to continuously maintain and improve the quality and efficiency of software systems development. In an era of highly interactive and responsive software processes where many organizations are using some form of agile methodology, automated testing is frequently becoming a requirement for web based projects. The main objective of Automated testing is to simplify as much of the testing effort as possible with a minimum set of scripts. Automated testing scripts can run fast and frequent, which is cost-effective for software products with a long maintenance life. When testing in an agile environment, the ability to quickly react to ever-changing software systems and requirements is necessary. New test cases are generated continuously and can be added to existing automated in parallel to the development of the software itself.

1.2: Advantages of automated testing

- Reliable: eliminating human error.
- Reusable and repeatable.
- Better quality software.
- Faster execution.
- Cost reduction.
- Increased coverage.

Chapter 2: Testing Approach

2.1: Web Application

A shopping cart web application(soccer) has been created using Apache tomcat, CSS Bootstrap, CSS framework, HTML, JavaScript, PHP, Simple cart plugin, SQL. Soccer application is created to test the basic ecommerce functionalities using three automated testing tools to evaluate those tools. This application has multiple functionalities.

- In register functionality the user can enter his/her details and the user password has met the criteria, the user will be registered successfully and the user details will be stored in the database.
- In login functionality, the user will login using username and password, if the username and password matches with the database stored credentials, the user will be successfully login. The user can view the users username on the navbar which the welcome before it.
- After successful login, the user will be redirected to the home page, where the user can view all the clothing.
- In the homepage, the user can view the desired products by clicking on the category items Sportswear, Mens and Womens and clicking on brands under those category items and the products will be displayed according to the selected condition.
- The user can even view items through brands by clicking on the brands under brands section and the products will be displayed according to the brands.
- The user can even view the products according to their price range, by moving the slider to their desire price range and the products will be displayed in that price range.
- The user can click on the desired product add to cart functionality which automatically add that item to the users cart showing the alert i.e., for example Arsenal Home Men's Jersey (L) was added to the cart!.
- The user can even click on the desired product name which directs to the product details page where the user can change the quantity of the product the user need and add product to the cart where the alert will popup, product was added to the cart!.
- The user will be redirected to the homepage, if the user clicks on the home logo “ALL THINGS SOCCER”
- The user will be redirected cart, if the user clicks on the cart option which displayed in navbar.
- In the cart page, the user can view all the products he/she has added to the cart. The user can increment/decrement the quantity of the desired product by clicking on the ‘+’/’-’ sign. By clicking on remove the user can even remove the product.
- By clicking on the logout option on the navbar, the user can logout from the website.

This website helps to know how to locate elements, how to handle pop-up dialog windows which cannot be inspected or included in the page content, how to fill text box, how to click buttons, how to navigate pages etc..

Soccer homepage

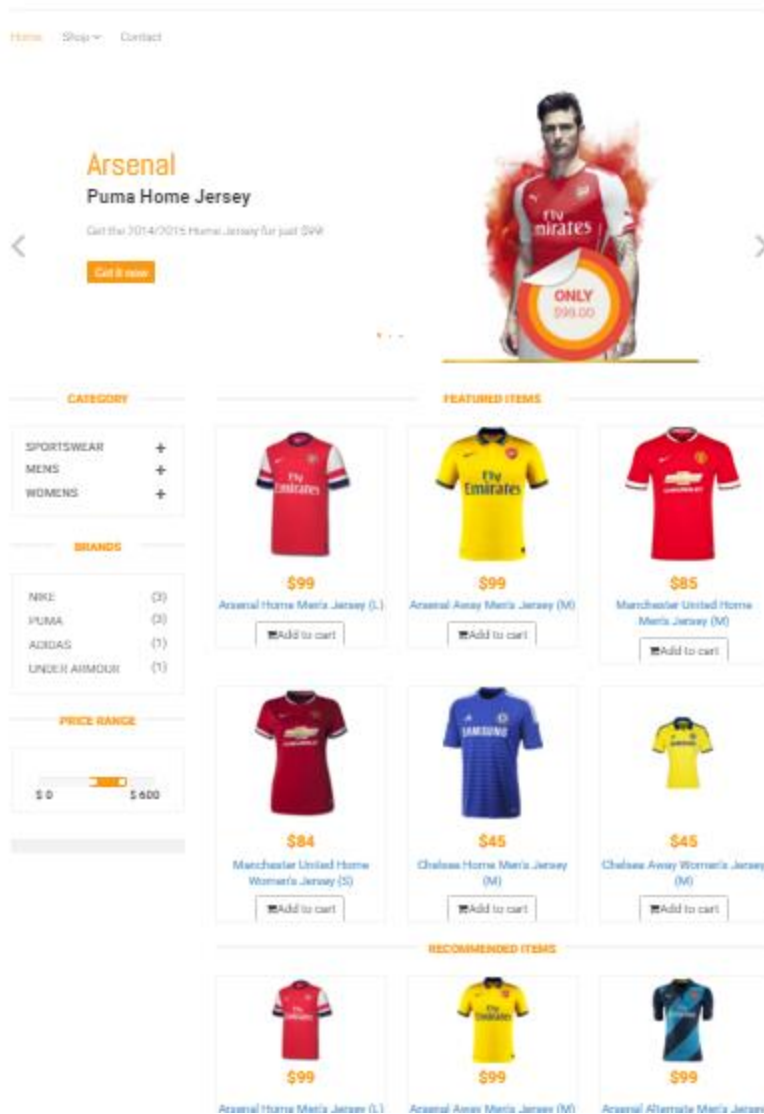


Figure 1: Soccer homepage

This is the home page of the soccer website. In this page, the user can view the products by selecting their desired category like sportswear, mens and womens. The user can view the products by even selecting various brands like Nike, Puma, Adidas and Under Armor. The user

can view the products by even selecting the price range which display the products in that price range. From this page the user can navigate to the login\register page, cart page.

Register/Login functionality:

The image shows a web interface for user authentication. It is divided into two main sections: 'Login to your account' on the left and 'New User Signup!' on the right. An orange circle with the text 'OR' is placed between the two sections. The login section contains two text input fields (one for username, one for password) and an orange 'Login' button. The signup section contains several text input fields: 'Username', 'Email Address', 'Password(Minimum of 6 Characters)', 'Re-enter Password', 'Address Line 1', 'Address Line 2 (Optional)', 'City', and 'Zipcode'. It also features a dropdown menu for the state, currently set to 'Alabama', and an orange 'Signup' button. A vertical scrollbar is visible on the right side of the page, and an orange button with an upward arrow is located at the bottom right corner.

Figure 2: Register/Login

This is the login page where the user can register/login by entering their details. If the user wants to register, the user can enter his/her details, if the user matches the password criteria the user details will be stored in the database and the user is successfully registered. If the user wants to login, the user can enter the username and the password. If the username and password matches the database stored username and password, the user will successfully login.

Add to cart functionality:

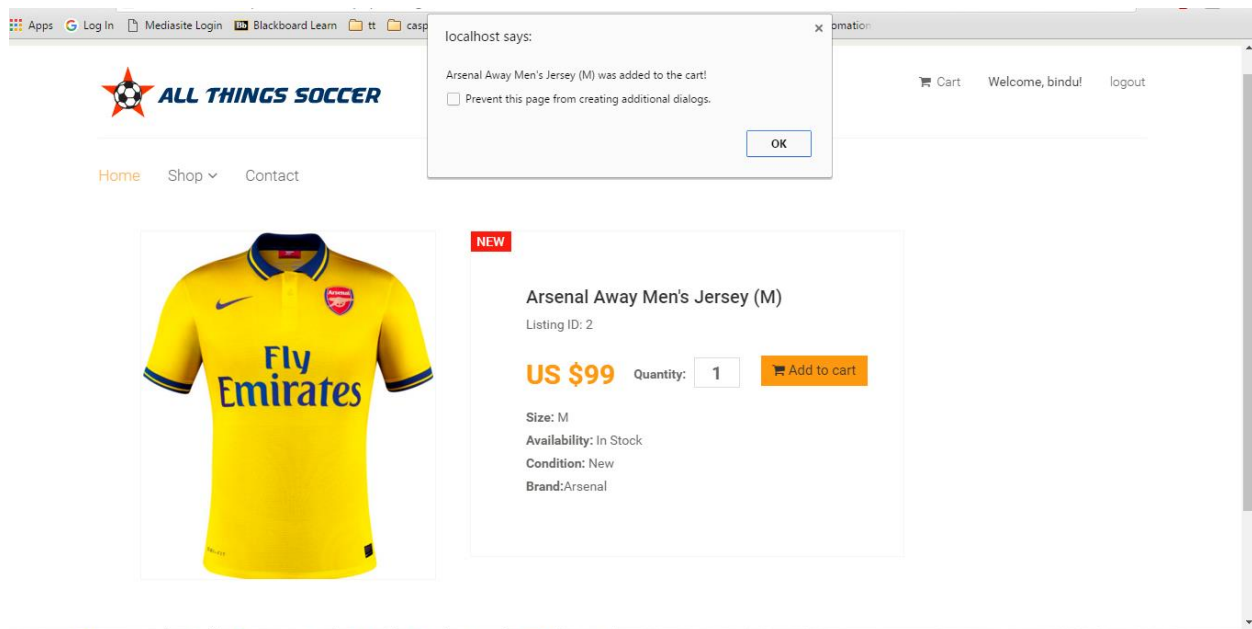


Figure 3: Add to cart

When the user logs in, the user name will be displayed on the navbar accompanied by welcome and then the username. The user can click on the desired product name, he/she will be directed to the product details page where the user can change the quantity and add the product to the cart. The alert box will appear saying the product was added to cart. The user can directly add the desired product to the cart by directly clicking add to cart button below the desired product without being redirected to the product details page and the default quantity be one.

Cart\Logout functionality:

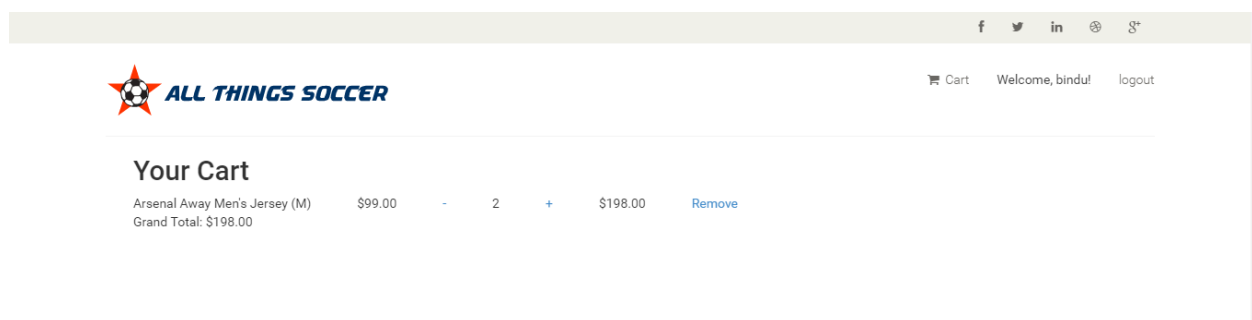


Figure 4: Logout

The user can increment/decrement the quantity of the product by clicking on the plus/minus symbols. The user can even remove the product at once by clicking on remove button. The user can logout as soon as he/she finishes with shopping.

2.2: FirePath:

Note: [https://en.wikipedia.org/wiki/Firebug_\(software\)](https://en.wikipedia.org/wiki/Firebug_(software))

FirePath has been used to inspect the elements using CSS selectors in CasperJs. “Firebug is a free and open-source web browser extension for Mozilla Firefox that facilitates the live debugging, editing, and monitoring of any website CSS, HTML, DOM, XHR, and JavaScript”. FirePath is a Firebug extension that adds a development tool to edit, inspect and generate XPath, CSS and Sizzle.

CasperJs uses headless browser execution. A headless browser is a web browser without a graphical user interface. Headless browsers provide automated control of a web page in an environment similar to popular web browsers, but are executed via command line interface. Step by step execution has been clearly shown in command prompt. Screenshots can be captured during execution.

2.3: Test Automated Tools Used:

Three test automated tools have been used to test the shopping cart web application in this project.

2.3.1: CasperJs

Note: <http://casperjs.org/>

CasperJs is a web UI testing framework build on top of PhantomJs.”CasperJs is an open source navigation scripting & testing utility written in JavaScript”. It can be installed on Linux, Windows, and Mac OS X platforms. In my project CasperJs has been installed on Windows platform. For installation we need to install both CasperJs and PhantomJs. The JavaScript code has been written in notepad++ and saved file using .js extension.

CasperJs finds an element with the help of CSS or XPath selectors.

2.3.2: Sahi

Sahi is a light-weight web testing tool. Sahi is an automated testing tool for web applications, with the facility to record and playback scripts. Sahi is a commercial tool. Sahi open source tool supports limited functionalities. Sahi supports Java and JavaScript.

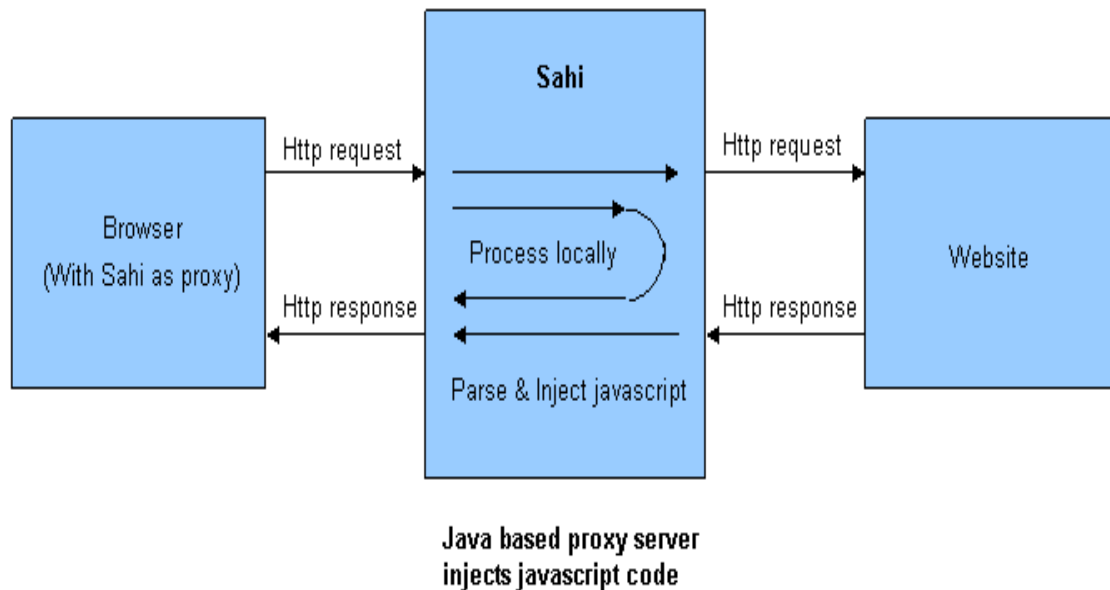


Figure 5: Sahi

Note: The above figure is from <https://sahipro.com/alldocs/v5.1.1/introduction/architecture.html>

Sahi runs as a proxy server and the browser's proxy settings are configured to point to Sahi's proxy. The injected javascript adds event handlers for recording, adds Sahi's APIs for event simulation and mocks some window dialogs like alert, prompt, confirm, print so that playback is not affected

Sahi works on multiple platforms like Windows, Linux, Android etc. Sahi runs on multiple browsers like Firefox, Google Chrome, Internet Explorer, Safari and Opera. Sahi directly recognizes browsers represent in our system.

Sahi open in a browser we choose. To open sahi controller in a browser we need to use ALT+DoubleClick command. Sahi controller has two tabs record and playback.

In record controller, we need to enter a script name and click record button. Then click stop button after when we are done with recording steps. Sahi automatically save the script in script directory. We can even choose other Actions like wait, comments, exists etc. which we can append to the script. Sahi also has inbuilt script editor, where we can directly edit scripts and save them without the use of third party applications.

Sahi playback controller runs the recorded execution steps when we enter the file name and startUrl and click the play button. The step by step execution has been clearly shown in the statements block. We even have the options to view/edit the scripts.

Sahi has in-built logs which shows the Pass/Fail reports clearly.

Sahi doesn't need any tools like eclipse, notepad++ to write sahi scripts. Sahi has inbuilt script editor where we can edit scripts and save them.

2.3.3: Selenium

Selenium is an open source Automated testing suite for web applications across different browsers and platforms. Selenium is a set of different software tools each with different approach to supporting automated testing like Selenium IDE, Selenium Remote Control RC, Selenium WebDriver and Selenium Grid. The standalone Selenium Server acts as a proxy between the script and the browser-specific drivers like Firefox driver, chrome driver etc.



Figure 6: Selenium

Note: The image is copied from the book, Selenium Testing Tools Starter, authored by Unmesh Gundecha, PACKT publishing.

2.3.3.1: Selenium IDE

The information in the Selenium IDE section is based on information from the Selenium documentation page, available at the following location: http://www.seleniumhq.org/docs/02_selenium_ide.jsp)

Selenium IDE is an integrated development environment for Selenium scripts. It is a Firefox plugin and provides an easy-to-use interface for developing automated tests. Selenium IDE has a recording feature, which records user actions as they are performed and then exports them as a reusable script in one of many programming languages that can be later executed.

Selenium IDE includes the entire Selenium Core, allowing to easily and quickly record and play back tests in the actual environment.

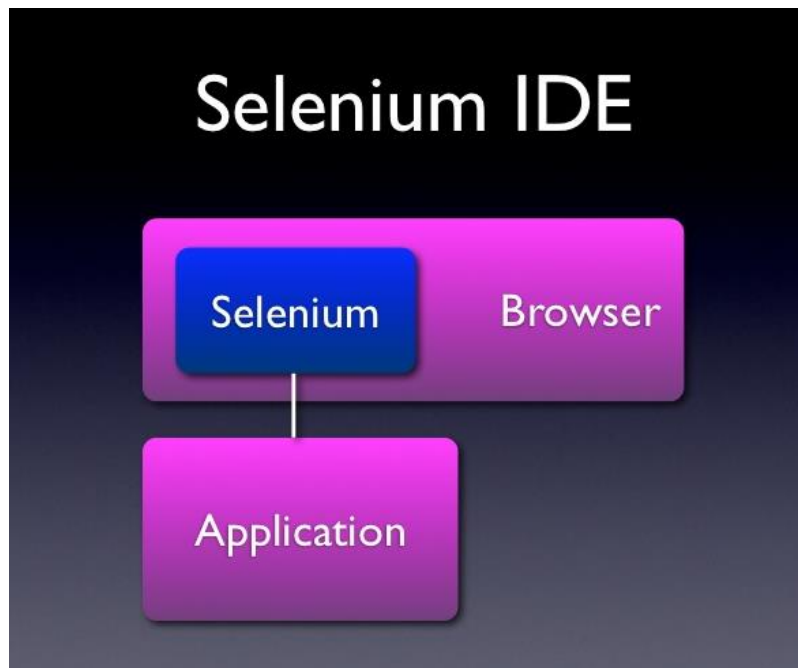


Figure 7: Selenium IDE

Note: <http://www.slideshare.net/kjbuckley/web-application-testing-with-selenium>, slide 23

To run the Selenium-IDE, we select it from the Firefox Tools menu. It opens with an empty script-editing window and a menu for loading, or creating new test cases.

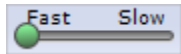
“The File menu has options for Test Case and Test Suite (suite of Test Cases). Using these you can add a new Test Case, open a Test Case, save a Test Case and export Test Case in a language of your choice. You can also open the recent Test Case.”

“The Edit menu allows copy, paste, delete, undo, and select all operations for editing the commands in your test case. The Options menu allows the changing of settings. You can set the timeout value for certain commands, add user-defined user extensions to the base set of Selenium commands, and specify the format (language) used when saving your test cases. The Help menu is the standard Firefox Help menu; only one item on this menu—UI-Element Documentation—pertains to Selenium-IDE.”

“Toolbar

The toolbar contains buttons for controlling the execution of your test cases, including a step feature for debugging your test cases. The right-most button, the one with the red-dot, is the record button.





Speed Control: controls how fast your test case runs.



Run All: Runs the entire test suite when a test suite with multiple test cases is loaded.



Run: Runs the currently selected test. When only a single test is loaded this button and the Run All button have the same effect.



Pause/Resume: Allows stopping and re-starting of a running test case.



Step: Allows you to “step” through a test case by running it one command at a time. Use for debugging test cases.



Apply Rollup Rules: This advanced feature allows repetitive sequences of Selenium commands to be grouped into a single action. Detailed documentation on rollup rules can be found in the UI-Element Documentation on the Help menu.



Record: Records the user’s browser actions.”

“Test Case Pane

“Script is displayed in the test case pane. It has two tabs, one for displaying the command and their parameters in a readable “table” format.

The other tab – Source, displays the test case in the native format in which the file will be stored. By default, this is HTML although it can be changed to a programming language such as Java or C#, or a scripting language like Python.” In the Options menu, we can change the programming languages in which we want the script to display. “The Source view also allows one to edit the test case in its raw form, including copy, cut and paste operations.”

Log

“When you run our test case, error messages and information messages showing the progress are displayed in this pane automatically. These messages are often useful for test case

debugging. The Clear button for clearing the Log. The info button is a drop-down allowing selection of different levels of information to log”.

2.3.3.2: Selenium WebDriver

Selenium 2.0 is the combination of both WebDriver and Selenium 1.0.

WebDriver is a tool for automating web application testing. It aims to provide a friendly API that’s easy to explore and understand, easier to use.

Selenium WebDriver is an open source Automated testing suite for web applications across different browsers and platforms.

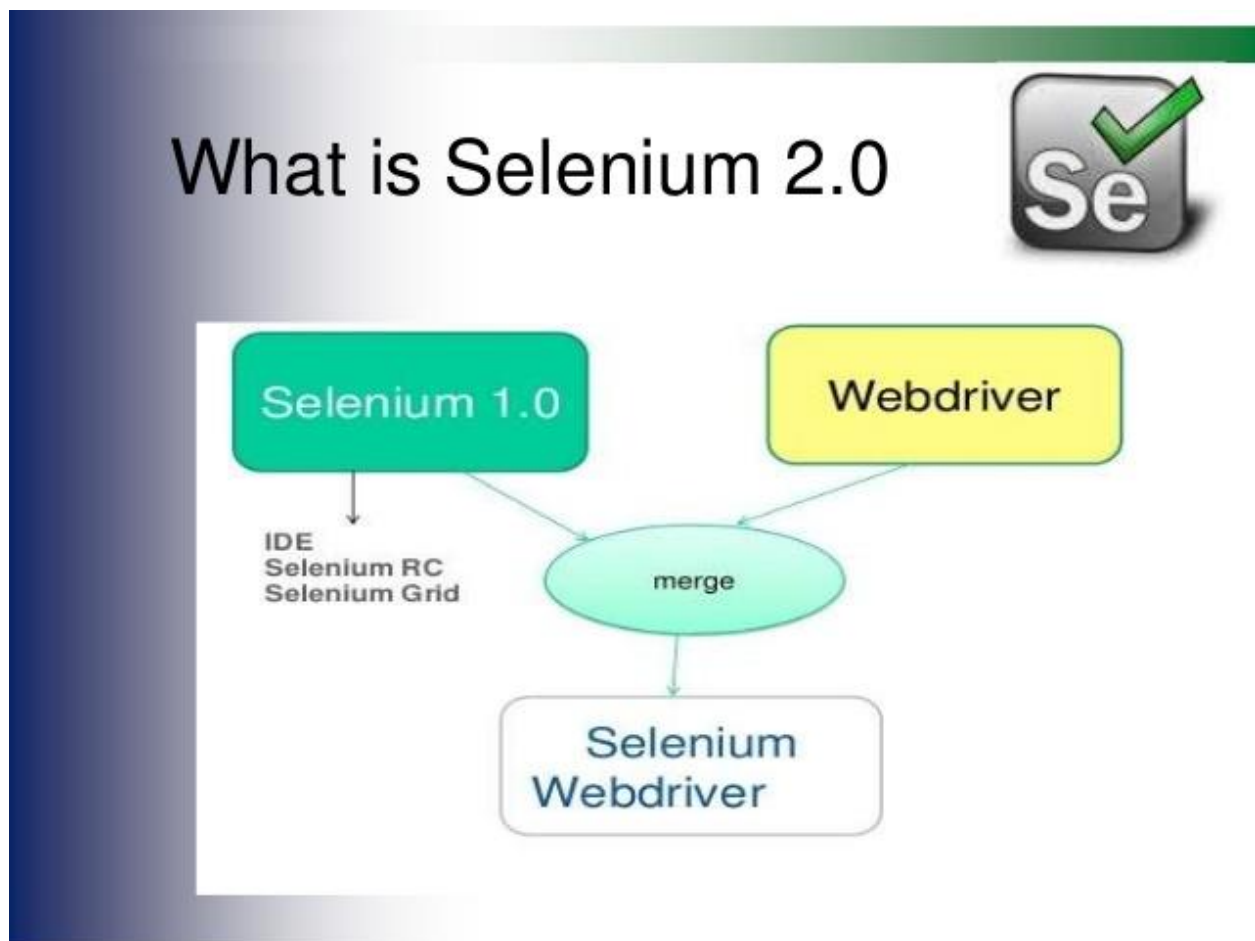


Figure 8: Selenium 2.0

The above figure8: <http://www.slideshare.net/humayunrana786/test-a-27965660> Slide 5

Selenium WebDriver scripts have been written in JAVA language in ECLIPSE IDE tool. A java project have been created in ECLIPSE IDE and selenium jar files have been added to external libraries.

Selenium Framework Execution Process:

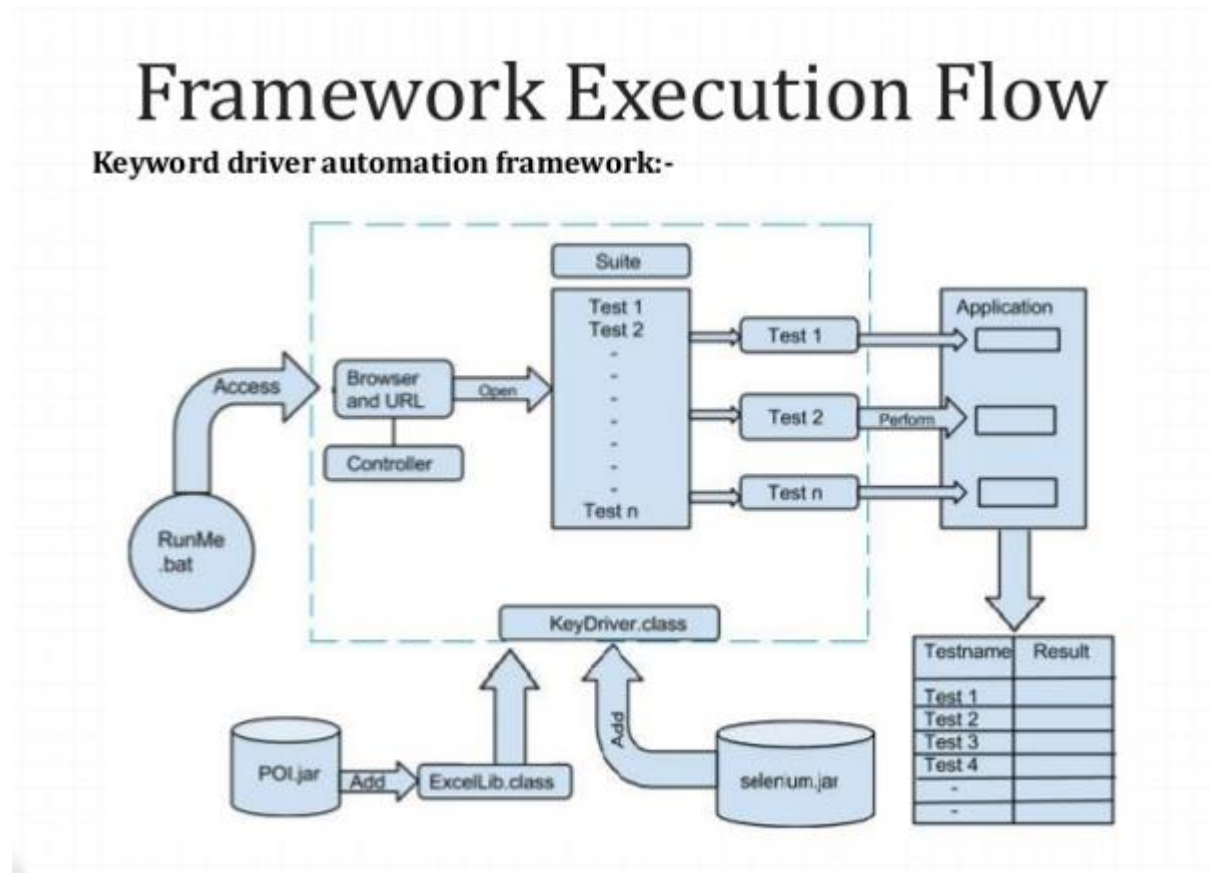


Figure 9: Selenium Framework Execution Process

Note: http://www.slideshare.net/cuelogic/automation-testing-by-selenium-web-driver?next_slideshow=4 slide:51

POI.jar file is an Apache POI whose mission is to create and maintain Java APIs for manipulating various file formats. The POI.jar file has been downloaded and added to external libraries class in eclipse Java Project (Soccer). Selenium.jar file has been downloaded and added to the project Soccer to write scripts in selenium. Test Suite is the combination of multiple test cases. The controller access the Browser and URL by running test suite, which contains multiple test cases and perform multiple tests in the applications and the results will be displayed.

Selenium WebDriver API commands:

- Connecting to a web-browser: `WebDriver driver= new FirefoxDriver ();`
For connection to Google chrome or internet explorer, we need to download `chromedriver.exe`, `IEDriverServer.exe` file and have to set property for those .exe files.
- Fetching a page: `driver.get("www.google.com");`
- Locating UI elements: UI elements can be located by using `id`, `css`, `xpath`, `className`, `tagName`, `name`, `linkText`.
In this project we can use `xpath` to locate UI elements. `FirePath` is the tool used for locating elements `xpath`.
- Popup dialogs: We can accept the alert using
`driver.switchTo().alert().accept();`

Publications:

1. A. Holmes and M. Kellogg. Automating functional tests using Selenium. In Proceedings of AGILE 2006, pages 270-275. IEEE, 2006.

Summary:

In this paper, the team wants to automate functional tests to run often in agile methodology before development. They used few open source tools like Canoo WebTest and HttpUnit but they failed to handle most instances. Then they used QuickTest Professional with record and playback features which interacts with browser but failed to adopt with changes and not suitable for scripts before development. Then they came up with selenium.

Selenium is a web testing tool with cross browser interaction. Selenium uses commands such as `navigate` a page, `locate` and `click` an element using identifiers like `name`, `xpath`, `css` etc., or enter information in textbox, `assertion` commands to allow the user to compare actual and expected values and can also run test suite i.e., combination of multiple test cases and run sequentially.

Selenium tests are integrated with other automated functional tests using FitNesse. FitNesse is an automated testing tool, whose goal is to enable customers, testers, and programmers to collaboratively learn what their software should do, and to automatically compare that to what it actually does and also keeps past versions of pages. Later selenium tests are added to the continuous integration build where CruiseControl is used. CruiseControl is a framework for a continuous build process which incorporates Ant and source control tools and allows

immediate feedback on the build and test status. Cobertura is a code coverage tool which calculates the percentage of code accessed by tests.

Out of all tools team as found that Selenium is the best which allows users to write their own extensions, easy to create custom actions. But selenium doesn't support downloading or uploading files, database tests or other back-end tests.

2. Andreas Bruns, Andreas Kornstädt, Dennis Wichmann, "Web Application Tests with Selenium", IEEE Software, vol.26, no. 5, pp. 88-91, September/October 2009, doi:10.1109/MS.2009.144

Summary:

Selenium help developers to easily run acceptance tests in their Web browsers which is an open source software available for Windows, Linux, and Macintosh.

Selenium Core run a JavaScript application in a host browser and controls Web application under test according to browser's capabilities. For Selenium Core the commands need to be written in Selenese, Selenium Remote Control (RC) scripts can be written in many programming languages like C#, Java, Perl, PHP, Python, and Ruby.

Selenium IDE record and playback tests within Firefox. Selenium Grid's purpose is to simulate an array of multiple users for load testing, use it to break a long test into smaller parts. These parts then execute in parallel on several machines instead of in sequence, which speeds things up as it runs in the browser. Locating elements can be done by using xpath or css selectors. Actions done by selenium are fill text fields, text areas, password fields, select an option from a drop-down menu, click a button, check or uncheck a checkbox, select a radio button, or follow a link, wait until the action a page navigate or refresh, verify whether an element matches the given pattern or whether the element is present, assert which is same as verify, but terminate if verification fails which waits until an element is present. This paper discuss about selenium features.

2.4: Evaluation:

2.4.1: Basic comparison:

Web Testing Tools	Browser Interaction	Current Version	License	Platforms	Recorder	Scripting Language
CasperJs	Headless browser execution	1.1.0	Free, Open source	Windows, Linux, Mac OS X	No	JavaScript/ CoffeeScript
Sahi Pro	Any	6.2.0	Commercial	Windows, Linux, Android, Mac etc...	Any browser	Sahi script.
Selenium	Any	2.0	Free, Open source	Windows, Linux, Android, Mac etc...	Only on Firefox	Java, Ruby, Php, Perl, Python, C#, Groovy.

2.4.2: Web browser connection:

- CasperJs supports headless browser execution.
- Sahi detects all the web browser in our system and display those browsers where we can choose any browser to run our scripts.
- Selenium IDE supports only Firefox browser
- In Selenium WebDriver: `WebDriver driver= new FirefoxDriver ();`
For connection to Google chrome or internet explorer, we need to download chromedriver.exe, IEDriverServer.exe file and have to set property for those .exe files.

2.4.3: Languages:

- CasperJs supports only JavaScript/CoffeeScript. If the user are not familiar with either of the scripts, they may be facing little difficulty in programming.

- Sahi Pro uses sahi script which is an extension of JavaScript language. Sahi also has driver in Java and Ruby, through which we can control the browser, but do not have features like automatic reports.
- Selenium supports multiple languages like Java, Ruby, Php, Perl, Python, C#, Groovy.

2.4.4: Recorder:

- Sahi records actions which are done in any browser and can playback later.
- Selenium IDE can record actions which are done in Firefox and can playback. Doesn't support any other browsers for recording option.

2.4.5: Navigation:

- In CasperJs, navigation to URL page is done by using start function
`casper.start('http://localhost/soccer/');`
- In Sahi, navigation to URL page is done by
`_navigateTo("http://localhost/soccer/ ");`
- In Selenium, the page fetching is done using command
`driver.get('http://localhost/soccer/');`

2.4.6: Wait For Element:

In some situations it takes time to load a webpage, so the element cannot be find immediately and due to late loading, the execution may get aborted in the middle as it cannot find the element. So, "wait" commands wait for some condition to become true. They will succeed immediately if the condition is already true. However, they will fail and halt the test if the condition does not become true within the current timeout setting or default timeout setting.

- The wait code is written by three automated tools for login button as follows.
`<button class="btn btn-default" value="Log in" name="login" type="submit">Login</button>`

- CasperJs: waitForSelector is used to wait for selected element using css selector.
`casper.then(function() {
 this.waitForSelector('.btn.btn-default', function() {
 this.click('.btn.btn-default');
 }); });`
- Sahi: wait is used to wait for selected element.

```
_wait(_submit("btn btn-default"))
```

- Selenium : WebDriverWait is used to wait for selected element.

```
WebDriver d=new FirefoxDriver();  
d.get("http://localhost/soccer/login.php");  
WebDriverWait w= new WebDriverWait(d,10);  
d.findElement(By.xpath("./*[@id='form']/div/div/div[1]/div/form/button"));
```

2.4.7: Locating elements:

The tools support the following strategies for locating elements:

By id:

Select the element with the specified @id attribute.

- CasperJs

```
casper.then(function() {  
    casper.waitForSelector('div.login-form', function() {  
        this.fillSelectors('div.login-form', {  
            'input[id="login_input_username"]': 'sindu',  
            'input[id="login_input_password"]': 'sindu1',  
        }, true);  
    })  
})
```

- Sahi

```
_byId("login_input_username ")
```

- Selenium

```
driver.findElement(By.id("login_input_username"))
```

By name:

- CasperJs

```
casper.then(function() {
```

```

casper.waitForSelector('div.login-form', function() {
  this.fillSelectors('div.login-form', {
    'select[name*="user_name"]': 'sindu',
    'select[name*="user_password"]': 'sindu1',
  }, true);
})

```

- In Sahi, the elements are located and clicked by Name

```
_click(_submit("login"));
```

The above is SahiScript, where `_click` , clicks on the link and `_setValue`, sets value to the selected textbox.

- Selenium

```
driver.findElement(By.name("login"))
```

Xpath:

XPath is the standard navigation tool for XML. XPath is used everywhere where there is XML. XPath is a very powerful language to express which element to identify, It can produce very reliable and low maintenance locators and can even create very brittle test cases.

Pros: Allows very precise locators

Cons: Slower than CSS, Relies on browser's XPath implementation which is not always complete (especially on IE).

Note: <https://testingchronicle.wordpress.com/2014/09/24/web-driver-locators-xpath-vs-css-selectors/>

Locate an element using an XPath expression.

- Using Xpath selector, CasperJs clicks on the login button.

```

casper.start('http://localhost/soccer/login.php', function() {
  this.test.assertExists({
    type: 'xpath',

```

```

        path: '//*[@id='form']/div/div/div[1]/div/form/button '
    }, 'the element exists');
});

```

- Using Xpath selector, Sahi Pro clicks on the Login button

```

_byXPath("//*[@id='form']/div/div/div[1]/div/form/button")

```

- In Selenium, Xpath selectors have been used to locate and click elements in the website. Below is the code for clicking on the login button using Xpath.

```

driver.findElement(By.xpath("//*[@id='form']/div/div/div[1]/div/form/button")).click();

```

driver.findElement, find the element within the current page by using given mechanism like xpath, css etc...

CSS Selector Syntax:

The CSS locator strategy uses CSS selectors to find the elements in the page. Selenium supports CSS 1 through 3 selector's syntax. Browsers implement CSS parsing engines for formatting or styling the pages using CSS syntax.

Pros:

- Much faster than Xpath
- Widely used
- Provides a good balance between structure and attributes
- Allows for selection of elements by their surrounding context

Cons: They tend to be more complex and require a steeper learning curve.

Note: <https://testingchronicle.wordpress.com/2014/09/24/web-driver-locators-xpath-vs-css-selectors/>

- In CasperJs, CSS selectors have been used to locate and click elements in the website.

```

casper.then(function() {
    this.fill('div.login-form', {

```

```
        user_name : 'sindu',  
        user_password : 'sindu1'  
    }, true); });
```

then() function basically adds a new navigation step in a stack. A step is a javascript function which can do two different things: one is waiting for the previous step - if any - being executed and other is waiting for a requested url and related page to load

- Sahi doesn't use CSS selectors.
Note: <https://sahipro.com/docs/introduction/index.html>
- In Selenium, CSS selectors have also been used to locate and click elements in the website.
Below is the code for clicking on the login button using CSS.

```
driver.findElement(By.cssSelector(".btn.btn-default")).click();
```

2.4.8: Pop-up dialogs:

- The Pop-up dialog window is an overlay positioned within the viewport and is protected from page content. They cannot be inspected. Those dialog windows were handled by different automated testing tools differently.
- CasperJs uses waitForAlert function to handle alerts.

```
casper.waitForAlert(function(response) {  
    this.echo("Alert received: " + response.data);  
    return true; });
```
- In Sahi, alerts are handled using

```
_lastalert()
```
- In Selenium,

```
driver.switchTo().alert();
```

```
return true;
```

2.4.9: Screenshots:

- CasperJs captures screenshots during execution using

```
this.capture('item.png');
```


- Sahi captures it using
`_takePageScreenShot();`
- Selenium captures screenshots using
`File srcFile=((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);`
`FileUtils.copyFile(srcFile, new File("C:item.png"));`

2.4.10: Assertions:

- Assertions are like Accessors, but they verify that the state of the application conforms to what is expected.
- There are many types of assertions. They are:

assertEquals() :

AssertTrue() has been done for user_name in login using three automated testing tools to retrieve the assertion as true.

```
<input id="login_input_username" class="login_input" type="text" required="" name="user_name"
>
```

- CasperJs:

```
test.assertEqual((x(' ./*[@id='login_input_username']'), 'sindu' );
```

- Sahi:

```
_assertEqual("sindu", _getValue(_textbox("user_name")));
```

_assertEqual will do nothing if the expected and actual values match

_assertEqual will log failures to the playback logs if the expected and actual values are not equal

- Selenium WebDriver:

```
Assert.assertEquals ("sindu",  
driver.findElement(By.id("login_input_username")).getAttribute("value"));  
System.out.println("assert equal and script continues");
```

Selenium IDE doesnot support assertEquals.

assertTrue() :

AssertTrue() has been done for user_name in login using three automated testing tools to retrieve the assertion as true.

```
<input id="login_input_username" class="login_input" type="text" required="" name="user_name">
```

- CasperJs:

```
casper.test.assertTruthy((x(' .//*[@id='login_input_username']'), 'sindu' );
```

- Sahi:

```
_assertTrue(true,_textbox("user_name"));
```

- Selenium :

```
Assert.assertTrue(driver.findElement(By.id("login_input_username")).isDisplayed());
```

assertFalse():

AssertFalse() has been done for user_name in login using three automated testing tools to retrieve the assertion as failed as it is true.

```
<input id="login_input_username" class="login_input" type="text" required="" name="user_name">
```

- CasperJs:

```
casper.test. assertFalsy ((x(' .//*[@id='login_input_username']'), 'b' );
```

- Sahi:

```
_assertFalse(false,_textbox("user_name"));
```

_assertFalse will do nothing if the condition evaluates to false

_assertFalse will log failures to the playback logs if the condition evaluates to true.

- Selenium:

```
Assert.assertFalse(null, false);
```

Here the script doesnot fail, if the value inserted in the textbox username is not null.

Selenium IDE doesnot have assertFalse

assertExists:

assertExists() has been done for user_name in login using three automated testing tools to retrieve the assertion as true.

```
<input id="login_input_username" class="login_input" type="text" required="" name="user_name">
```

- CapserJs:

```
casper.test.assertExists(x('//*[id='login_input_username']'), 'the element exists');
```

- Sahi:

```
_assertExists(_textbox("user_name"));
```

- Selenium:

Selenium doesnot support assertExists.

assertNotEqual() :

AssertNotEqual() has been used for user_name in login using three automated testing tools, if the to retrieve the assertion as true.

```
<input id="login_input_username" class="login_input" type="text" required="" name="user_name">
```

- CapserJs:

```
casper.test. assertNotEquals ((x('//*[id='login_input_username']'), 'b' );
```

- Sahi:

```
_assertNotEqual("b", _getValue(_textbox("user_name")));
```

_assertNotEqual will do nothing if the expected and actual values do not match
_assertNotEqual will log failures to the playback logs if the expected and actual values are equal

- Selenium:

```
Assert.assertNotEquals ("b",  
driver.findElement(By.id("login_input_username")).getAttribute("value"));  
System.out.println("assert not equal and script continues");
```

Selenium IDE doesnot support assertEquals.

assertVisible:

AssertVisible() has been used for user_name in login using three automated testing tools, if the textbox is visible in that page it retrieves the assertion as true.

```
<input id="login_input_username" class="login_input" type="text" required="" name="user_name">
```

- CasperJs:

```
casper.test.assertVisible(x('.*[@id='login_input_username']'));
```
- Sahi:

```
_assertVisible(_textbox("user_name"));
```
- Selenium: All Selenium Assertions can be used in two modes: "assert" and "verify". For example, you can "assertVisible" and "verifyVisible" . When an "assert" fails, the test is aborted. When a "verify" fails, the test will continue execution, logging the failure. This allows a single "assert" to ensure that the application is on the correct page, followed by a bunch of "verify" assertions to test form field values, labels, etc.

Code using assertVisible:

```
assertVisible(driver.findElement(By.id("login_input_username")));
```

Code using verifyVisible: The verification errors will be logging the failure but the test will continue execution.

```
try {  
    assertTrue(driver.findElement(By.id("login_input_username")).isDisplayed());  
} catch (Error e) {  
    verificationErrors.append(e.toString());  
}
```

assert:

Assert() has been used for user_name in login using three automated testing tools, if the to retrieve the assertion as true.

```
<input id="login_input_username" class="login_input" type="text" required="" name="user_name">
```

- CasperJs:

```
casper.test.assert(x('//*[id='login_input_username']'), 'true');
```

- Sahi:

```
_assert(true,_textbox("user_name"))
```

_assert will do nothing if the condition passed to it is true

_assert will log failures to the playback logs if the condition evaluates to false.

- Selenium: Doesnot have assert() condition.

Selenium IDE doesn't contain assertEquals, assertTrue, assertFalse, assertExists and assertNotEqual. Selenium doesnot have assert() function.

2.4.11: Mode of execution:

- In CasperJs, the mode of execution is done using command line prompt using command
casperjs filename.js
- In Sahi, the mode of execution is done using playback option, by entering the filename and start URL.
- In Selenium IDE, the mode of execution is done by selecting the test case and selecting the option whether to run the entire test suite or the test case.
- In Selenium WebDriver, the mode of execution is done by using another tools like Eclipse.

2.4.12: Execution:

- CasperJs supports headless browser execution, which means the detailed step by step execution process can be viewed in command prompt. Disadvantage of headless browser execution is it gets unmanageable if we have lot of test files.
- Sahi, runs directly in the browser which it saves lot of debugging effort as we can view the activities directly in the browser and the inbuilt PASS/FAIL logs are saved directly for later viewing purpose.
- Selenium, runs directly in the browser so, it saves lot of debugging effort as we can view the activities directly in the browser.
- But, the headless browser execution is much faster than browser execution when it executes multiple scripts at a time.

2.4.13: Selenium vs CasperJs

Selenium	CasperJs
1. Cross-browser execution.	1. Command line based headless browser execution.
2. Supports many languages.	2. Have to be familiar with JavaScript/ CoffeeScript.
3. Does not produce detailed results.	3. Shows detailed results in command prompt.
4. Saves you a lot of debugging effort, as we are able to view the activities of the browser in real time.	4. The output of test passes/ fails gets unmanageable once you have lot of test files.
5. Running multiple test cases with Selenium turns to be a time-consuming.	5. Is capable of executing multiple test cases within a record time period.

2.4.14: Selenium IDE vs Sahi

Selenium IDE	Sahi
1. Works only on Firefox.	1. Works on any browser.
2. Open source	2. Open source offers limited functionality. Sahi Pro commercial.
3. Supports many programming languages.	3. Supports sahi script
4. No In-built logs	4. In-built test success/failure report logs for later reference.
5. Additional tools are required to run scripts. (e.g.: eclipse)	5. No additional tools required. Scripts directly run in sahi controller.

2.4.15: Evaluation Overview:

The evaluation has been done by testing same test case in web application using three automated testing tools CasperJs, Sahi and Selenium. Total nine different test cases have been created using three automated tool each contains nine test cases. Test cases have been created for register, login, add to cart, view products by sportswear brands, view products by mens brands, view products by womens brands, view products by brands, view product in desired price range, cart and logout functionalities. The comparison have been done on how easy to write the script, how many assertions have been used and whether the test case results are pass/fail by all three automated testing tools.

2.4.16: Test case Evaluation:

Test case 1: Register functionality

Scenario: When the user opens the webpage, the home page will be displayed. The test case should click on the login/register button on the homepage which navigates to the register page and enter the details in the register form textboxes and click on the state which is a dynamic dropdown in the register form and clicks on the register button.

Features of the site: In this test case registration functionality, login/register element and register button are being tested.

Features of the testing tools: In this test case navigating to webpage, locating and clicking on elements, sending keys to the textbox, assertions for checking whether the element exists and clicking on the button features are being tested.

Results:

- By testing this test case in CasperJs we have used one assertion `assertVisible()` to check whether the username textbox is visible or not and the assertion has been passed.
- By testing this test case in Sahi we have used two assertions `assertVisible()` to check whether the username textbox is visible or not and `assertEqual()` to check whether the entered password is equal to the expected password and two assertions have been passed.
- By testing this test case in selenium we have used two assertions `assertVisible()` to check whether the username textbox is visible or not and `assertEqual()` to check whether the entered password is equal to the expected password and two assertions have been passed.
- By testing this test case using three automated tools I have found that for filling forms CasperJs is an efficient tool as it clicks on the register form and fills all the text boxes by entering text box name and the text that has to be filled. In Sahi and Selenium each and every text box has been located and clicked to send the value.

Test case 2: Login functionality

Scenario: When the user opens the webpage, the home page will be displayed. The test case should click on the login/register button on the homepage which navigates to the login page and enter their login credentials in the login form textboxes and clicks on the login button.

Features of the site: In this test case Login functionality, login/register element and login button are being tested.

Features of the testing tools: In this test case navigating to webpage, locating and clicking on elements, sending keys to the textbox, assertions for checking whether the actual text is equal to the expected text and clicking on the button features are being tested.

Results:

- By testing this test case in CasperJs we have used one assertion `assertEqual()` to check whether the actual entered user name is equal to the expected username and same for password and the assertions has been passed.
- By testing this test case in Sahi we have used one assertion `assertEqual()` to check whether the actual entered user name is equal to the expected username and same for password and the assertions has been passed.
- By testing this test case in selenium we have used one assertion `assertEqual()` to check whether the actual entered user name is equal to the expected username and same for password and the assertions has been passed.
- By testing this test case using three automated tools I have found three tools are suitable to test the login functionality.

Test case 3: add to cart functionality

Scenario: When the user opens the webpage, the home page will be displayed. in the home page, the test case should click on the add to cart button for the mentioned product where alert has to be handled and it should also clicks on the mentioned product which directs to the product detail page where it should change the quantity of the product and then click on the add to cart button where alert has to be handled.

Features of the site: In this test case add to cart button, clicking on the product directly directs to the product detail page or not and handling alerts are being tested.

Features of the testing tools: In this test case locating and clicking on buttons, sending keys to the textbox and alert handling are being tested.

one assertion `assertEqual()` has been used many times to check whether the actual displayed product price is equal to the expected product price range and the assertion has been failed whereas the test case.

Test case 4: view products by sportswear brands functionality

Scenario: When the user opens the webpage, the home page will be displayed. In the test case by clicking on the sportswear category the dynamic dropdown appears with various brands. By clicking on the brands we have to check whether the page is displaying only the selected brands or not.

Features of the site: In this test case sportswear category dynamic dropdown, selecting brands and displaying only selected brand products are being tested.

Features of the testing tools: In this test case locating and clicking on elements, assertions features are being tested.

Results:

- By testing this test case in CasperJs we have used one assertion `assertEqual()` has been used many times to check whether the actual displayed product is equal to the expected product for all brands and the assertion has been failed whereas the test case.
- By testing this test case in Sahi we have used one assertion `assertEqual()` has been used many times to check whether the actual displayed product is equal to the expected product for all brands and the assertion has been failed whereas the test case.
- By testing this test case in selenium we have used one assertion `assertEqual()` has been used many times to check whether the first actual displayed product is equal to the expected product for all brands and the assertion has been failed whereas the test case.
- Total four `assertEqual()` assertions has been used in CasperJs, Sahi and Selenium where four assertions have been failed as the page is displaying all the products on only selected brand products.

Test case 5: view products by mens brands functionality

Scenario: When the user opens the webpage, the home page will be displayed. In the test case by clicking on the mens category the dynamic dropdown appears with various brands. By clicking on the brands we have to check whether the page is displaying only the selected brands or not.

Features of the site: In this test case mens category dynamic dropdown, selecting brands and displaying only selected brand products are being tested.

Features of the testing tools: In this test case locating and clicking on elements, assertions features are being tested.

Results:

- By testing this test case in CasperJs we have used one assertion `assertEqual()` has been used many times to check whether the actual displayed product is equal to the expected product for all brands and the assertion has been failed whereas the test case.
- By testing this test case in Sahi we have used one assertion `assertEqual()` has been used many times to check whether the actual displayed product is equal to the expected product for all brands and the assertion has been failed whereas the test case.
- By testing this test case in selenium we have used one assertion `assertEqual()` has been used many times to check whether the first actual displayed product is equal to the expected product for all brands and the assertion has been failed whereas the test case.
- Total four `assertEqual()` assertions has been used in CasperJs, Sahi and Selenium where four assertions have been failed as the page is displaying all the products on only selected brand products.

Test case 6: view products by womens brands functionality

Scenario: When the user opens the webpage, the home page will be displayed. In the test case by clicking on the womens category the dynamic dropdown appears with various brands. By clicking on the brands we have to check whether the page is displaying only the selected brands or not.

Features of the site: In this test case womens category dynamic dropdown, selecting brands and displaying only selected brand products are being tested.

Features of the testing tools: In this test case locating and clicking on elements, assertions features are being tested.

Results:

- By testing this test case in CasperJs we have used one assertion `assertEqual()` has been used many times to check whether the actual displayed product is equal to the expected product for all brands and the assertion has been failed whereas the test case.
- By testing this test case in Sahi we have used one assertion `assertEqual()` has been used many times to check whether the actual displayed product is equal to the expected product for all brands and the assertion has been failed whereas the test case.

- By testing this test case in selenium we have used one assertion `assertEqual()` has been used many times to check whether the first actual displayed product is equal to the expected product for all brands and the assertion has been failed whereas the test case.
- Total four `assertEqual()` assertions has been used in CasperJs, Sahi and Selenium where four assertions have been failed as the page is displaying all the products on only selected brand products.

Test case 7: view products by brands functionality

Scenario: When the user opens the webpage, the home page will be displayed. In the test case by clicking on the brands a dynamic dropdown appears with various brands. By clicking on the brands we have to check whether the page is displaying only the selected brands or not.

Features of the site: In this test case brands category dynamic dropdown, selecting brands and displaying only selected brand products are being tested.

Features of the testing tools: In this test case locating and clicking on elements, assertions features are being tested.

Results:

- By testing this test case in CasperJs we have used one assertion `assertEqual()` has been used many times to check whether the actual displayed product is equal to the expected product for all brands and the assertion has been failed whereas the test case.
- By testing this test case in Sahi we have used one assertion `assertEqual()` has been used many times to check whether the actual displayed product is equal to the expected product for all brands and the assertion has been failed whereas the test case.
- By testing this test case in selenium we have used one assertion `assertEqual()` has been used many times to check whether the first actual displayed product is equal to the expected product for all brands and the assertion has been failed whereas the test case.
- Total four `assertEqual()` assertions has been used in CasperJs, Sahi and Selenium where four assertions have been failed as the page is displaying all the products on only selected brand products.

Test case 8: view product in desired price range functionality

Scenario: When the user opens the webpage, the home page will be displayed. In the test case by slider bar in the price range the page should display the products which are within the price range.

Features of the site: In this test case brands both sliders in price range, displaying products which are within the price range are being tested.

Features of the testing tools: In this test case locating elements, mouse bar actions and assertions features are being tested.

Results:

- By testing this test case in CasperJs we have used one assertion `assertEqual()` has been used many times to check whether the actual displayed product price is equal to the expected product price range and the assertion has been failed whereas the test case.
- By testing this test case in Sahi we have used one assertion `assertEqual()` has been used many times to check whether the actual displayed product price is equal to the expected product price range and the assertion has been failed whereas the test case.
- By testing this test case in selenium we have used one assertion `assertEqual()` has been used many times to check whether the actual displayed product price is equal to the expected product price range and the assertion has been failed whereas the test case.
- The `assertEqual()` assertion has been used in CasperJs, Sahi and Selenium where the assertion has been failed as the page is displaying all the products on only displaying according to the price range.

Test case 9: cart and logout functionality

Scenario: When the user opens the webpage, the home page will be displayed. The test case should click on the cart element on the homepage which navigates to the view cart page, can increment/ decrement the quantity of the products, directly remove the products by clicking on remove button and clicks on the logout button.

Features of the site: In this test case increment and decrement the quantity, remove element functionality and logout functionality are being tested.

Features of the testing tools: In this test case navigating to webpage, locating and clicking on elements and assertions feature are being tested.

Results:

- By testing this test case in CasperJs we have used one assertion `assertEqual()` has been to check the cost of the product matches with the expected product price when the quantity is incremented/decremented and the assertion has been passed whereas the test case.
- By testing this test case in Sahi we have used one assertion `assertEqual()` has been to check the cost of the product matches with the expected product price when the quantity is incremented/decremented and the assertion has been passed whereas the test case.
- By testing this test case in selenium we have used one assertion `assertEqual()` has been to check the cost of the product matches with the expected product price when the quantity is incremented/decremented and the assertion has been passed whereas the test case.

2.4.17: Conclusion for evaluation:

- In this evaluation section 2.4.1, we have learned the basic comparison between three automated web testing tools.
- From 2.4.2 we have learned how the web connection will be given in three tools for different browsers, as CasperJs supports headless browser execution there is no need to separate browser connection, sahi automatically detects all the browser in the system and display those browser and connections has to be given for selenium.
- From 2.4.3 we have learned which tool supports which scripting languages.
- From 2.4.4 we have learned about which tools supports record feature and on how many web browsers.
- From 2.4.5 we have learned how to navigate from one page to another web page in three tools.
- From 2.4.6 we have learned when and how to wait For Element in three tools as it may take time to load the webpage.
- From 2.4.7 we have learned, what are the various strategies three tools support for locating elements, which tool doesn't support certain strategy which are being supported by other tools and how to locate the same elements using various strategies by three tools.
- From 2.4.8 we have learned how to handle pop-up dialogs by three testing tools.
- From 2.4.9 we have learned how to capture screenshots using three testing tools.

- From 2.4.10 we have learned various types of assertions. Which tool supports which types of assertions which are not supported by another tool. If three tool support same assertions how those assertions can be written in three tools.
- From 2.4.11 we have learned about the mode of execution of various tools.
- From 2.4.12 we have learned about how the scripts are being executed in three tools.
- In 2.4.13 we discuss about the basic differences between Selenium and CasperJs.
- In 2.4.14 we discuss about the basic differences between Selenium IDE and Sahi.
- In 2.4.15 we discuss about how the evaluating feature of three tools helped in executing test cases in those three tools.
- In 2.4.16 we discuss about the various test cases and the scenario of those test cases and what features of the site and the testing tools it tests.
- Tools are compared by comparing the features of three automated testing tools by detailing about which tools supports what various strategies in the feature and which tool doesn't support those various strategies in the same feature and also how the feature has to be mentioned in the code is shown clearly by taking the same element for feature.
- CasperJs tool works better for filling login/register forms and it need one function to locate the login form and enter the details in the text box whereas in Selenium and sahi the user need to locate each text box and the keys need to be sent.
- When the individual is not familiar with JavaScript then the selenium tool works better for the user as it supports many scripting languages.
- Selenium and Sahi supports record and playback features which reduces the manual effort whereas in CasperJs the individual need to write the scripts manually.
- In assertions, I found that Sahi works easier as it shows the format of the specified assertion and instead of running the whole script it only runs the assertion statement and if it meets the criteria then we can append the statement to the script.
- In execution each tool has its own advantages and disadvantage. Selenium and Sahi execution by interacting with browser which saves the debugging effort but takes time to load the webpage and perform the entire test case whereas in CasperJs it supports headless browser execution which executes fast by viewing pass/fail test output gets unmanageable when we have lot of test files.

2.5: Implementation

The test scripts have been written in three automated testing tools i.e., CasperJs, Sahi, Selenium IDE and Selenium WebDriver.

2.5.1: SUCCESS TEST CASES:

2.5.1.1: CasperJs

Register functionality:

Execution: casperjs register.js

```
C:\Users\kiran\Desktop\soccer casperjs>casperjs register.js
[info] [phantom] Starting...
[info] [phantom] Running suite: 6 steps
[debug] [phantom] opening url: http://localhost/soccer/, HTTP GET
[debug] [phantom] Navigation requested: url=http://localhost/soccer/, type=Other
, willNavigate=true, isMainFrame=true
[debug] [phantom] url changed to "http://localhost/soccer/"
[debug] [phantom] Successfully injected Casper client-side utilities
[debug] [phantom] start page is loaded
[info] [phantom] Step anonymous 3/6 http://localhost/soccer/ <HTTP 200>
[debug] [phantom] Mouse event 'mousedown' on selector: a[href='login.php']
[debug] [phantom] Mouse event 'mouseup' on selector: a[href='login.php']
[debug] [phantom] Mouse event 'click' on selector: a[href='login.php']
[debug] [phantom] Navigation requested: url=http://localhost/soccer/login.php, t
ype=LinkClicked, willNavigate=true, isMainFrame=true
[info] [phantom] Step anonymous 3/6: done in 4362ms.
[debug] [phantom] url changed to "http://localhost/soccer/login.php"
[debug] [phantom] Successfully injected Casper client-side utilities
[info] [phantom] Step anonymous 4/6 http://localhost/soccer/login.php <HTTP 200>
[info] [remote] attempting to fetch form element from selector: 'div.signup-form'
```

Figure 10: CasperJs register

To test the register functionality in the soccer web application, a JavaScript code has been written in CasperJs in Notepad++ which automatically navigates to the register page and enter the mentioned username, email, password, re-enter password, address line 1 and line 2, city, zip code and state in the code. The execution has been done in command line prompt as CasperJs supports headless browser execution. The command used for execution is casperjs register.js, where register.js is the file name with .js extension.

Captured screenshot:

New User Signup!

bindu
bindu@gmail.com
.....
.....
E 10th street
Address Line 2 (Optional)
greenville
27858
North Carolina
Signup

Figure 11: CasperJs Screenshot1

The code has been written in register.js file to capture the screenshot in the middle of the execution so, that we can view how the execution process has been done or it can be used for later reference.

Login, add to cart and logout functionalities:

Execution: casperjs login.js

```
Command Prompt
C:\Users\kiran\Desktop\soccer casperjs>casperjs login.js
[info] [phantom] Starting...
[info] [phantom] Running suite: 25 steps
[debug] [phantom] opening url: http://localhost/soccer/, HTTP GET
[debug] [phantom] Navigation requested: url=http://localhost/soccer/, type=Other, willNavigate=true, isMainFrame=true
[debug] [phantom] url changed to "http://localhost/soccer/"
[debug] [phantom] Successfully injected Casper client-side utilities
[debug] [phantom] start page is loaded
[info] [phantom] Step anonymous 3/25 http://localhost/soccer/ <HTTP 200>
[debug] [phantom] Mouse event 'mousedown' on selector: a[href='login.php']
[debug] [phantom] Mouse event 'mouseup' on selector: a[href='login.php']
[debug] [phantom] Mouse event 'click' on selector: a[href='login.php']
[debug] [phantom] Navigation requested: url=http://localhost/soccer/login.php, type=LinkClicked, willNavigate=true, isMainFrame=true
[info] [phantom] Step anonymous 3/25: done in 1983ms.
[debug] [phantom] url changed to "http://localhost/soccer/login.php"
[debug] [phantom] Successfully injected Casper client-side utilities
[info] [phantom] Step anonymous 4/25 http://localhost/soccer/login.php <HTTP 200>
[info] [remote] attempting to fetch form element from selector: 'div.login-form'
```

Figure 12: CasperJs execution 1

To test the login, add to cart and logout functionalities in the soccer web application, a JavaScript code has been written in CasperJs in Notepad++ which automatically navigates to the login page and enter the mentioned username and password in the code. If the user has been successfully login, the website redirects the user to the home page. In the home page, the code has been written to select a product and change the quantity of that product and add that product to the cart. The execution has been done in command line prompt as CasperJs supports headless browser execution. The command used for execution is `casperjs register.js`, where `register.js` is the file name with `.js` extension. Screenshots has been captured to view the execution.

Captured images:

Login to your account

bindu

Login

Figure 13: CasperJs screenshot 2

The code has been written in `login.js` file to capture the screenshot in the middle of the execution. Through this screenshot we can see that the code has entered given credentials.



 Cart Welcome, bindu!logout

Your Cart

Arsenal Away Men's Jersey				
(M)		\$99.00	-	
6	+	\$594.00		Remove
Arsenal Home Men's Jersey				
(L)		\$99.00	-	
1	+	\$99.00		Remove
Grand Total: \$693.00				

Figure 14: CasperJs screenshot 3

The code has been written in login.js file to capture the screenshot in the middle of the execution. Through this screenshot we can see that the products have been added to the cart by changing the quantity of one product and the quantity of the second product is default.



 Cart Welcome, bindu!logout


Your Cart

Arsenal Away Men's Jersey				
(M)		\$99.00	-	
5	+	\$495.00		Remove
Arsenal Home Men's Jersey				
(L)		\$99.00	-	
1	+	\$99.00		Remove
Grand Total: \$594.00				

Figure 15: CasperJs screenshot 4

The code has been written in login.js file to capture the screenshot in the middle of the execution. Through this screenshot we can see that the quantity of one of the product has been decremented using minus sign beside the quantity of that product.



 Cart Welcome, bindu!logout

Your Cart

Arsenal Home Men's Jersey				
(L)		\$99.00	-	
1	+	\$99.00		Remove
Grand Total: \$99.00				

Figure 16: CasperJs screenshot 5

The code has been written in login.js file to capture the screenshot in the middle of the execution. Through this screenshot we can see that one of the product has been removed by clicking on remove.

2.5.1.2: Sahi:

Register functionality:

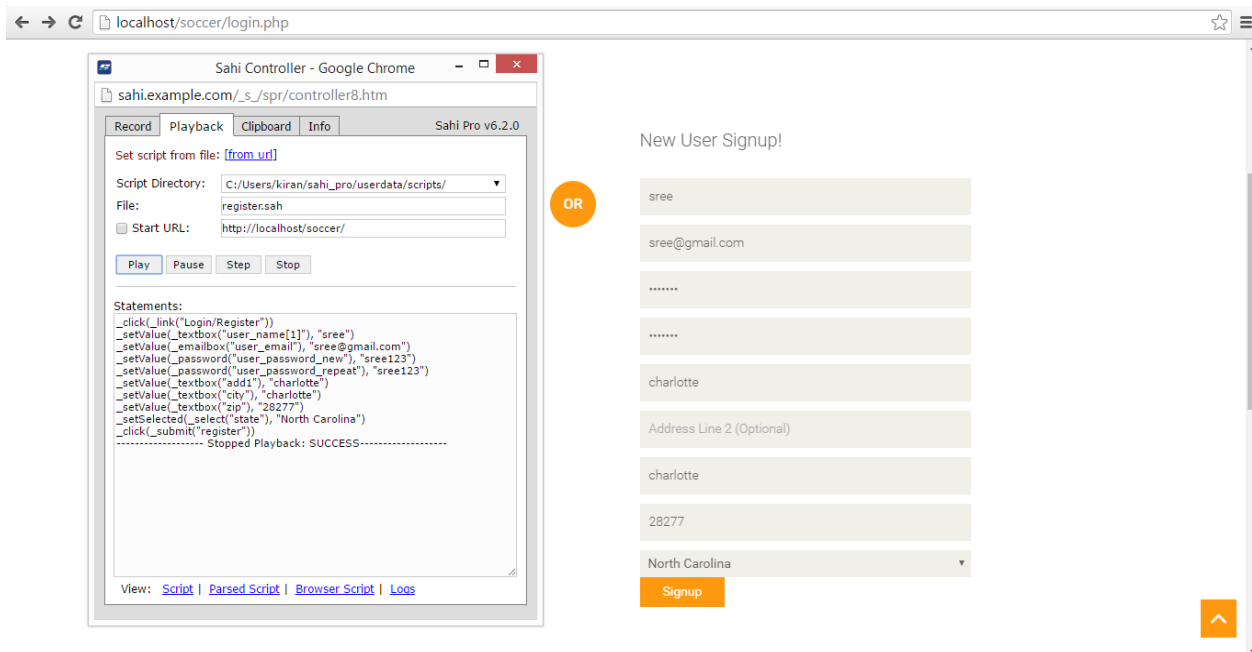


Figure 17: Sahi register

To test the register functionality in the soccer web application, sahi has recorded the process that have to be done by tester and automatically generated a sahi script which automatically navigates to the register page and enter the mentioned username, email, password, re-enter password, address line 1 and line 2, city, zip code and state in the code. The execution can be viewed by selecting the file register.sah where register is the file name and .sah is the extension to save sahi script and clicking on play option. The script will start execution and the screenshot has been captured.

Login/Add to cart/ Logout functionalities:

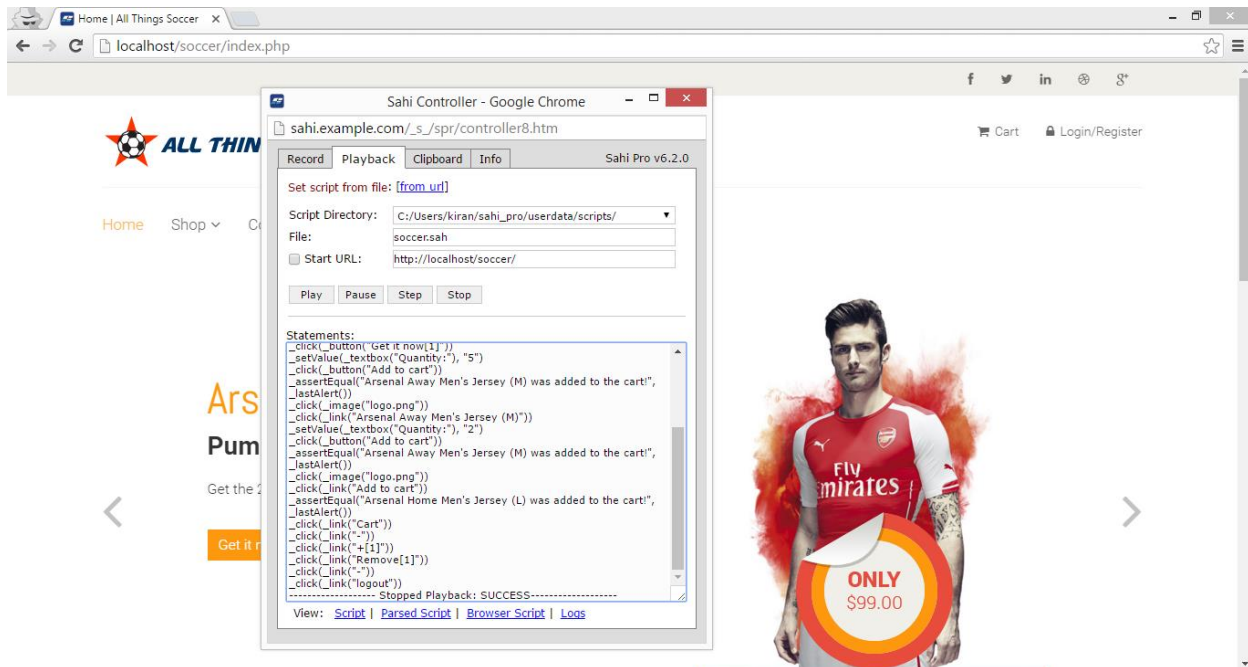


Figure 18: Sahi login

To test the login, add to cart and logout functionality in the soccer web application, sahi has recorded the process that have done by tester and automatically generated a sahi script which automatically navigates to the login page and enter the mentioned username and password in the code. If the user has been successfully login, the website redirects the user to the home page. In the home page, in the home page, the product has been selected and the quantity of that product is changed and add that product to the cart and logouts the user. The execution can be viewed by selecting the file soccer.sah where soccer is the file name and .sah is the extension to save sahi script and clicking on play option. The script will start execution and the screenshot has been captured.

Logs:

Root | Suite Report | Test Cases Report | **Script Report**

Script Name: soccer.sah | Auto Refresh

Test	Total Steps	Failures	Errors	Success Rate	Time Taken (ms)	Node	Load	Browser
soccer.sah	28	0	0	100 %	9247	localhost:9999	0	chrome

Report Id: soccer_chrome__7fc4be970328f0421108c170f836c25f6c9c | Compare Logs

Starting script [Expand All](#) [Collapse All](#)

```
_click(_link("Login/Register")) [1162 ms] [08:27:07.520 PM]
_setValue(_textbox("user_name"), "sree") [1068 ms] [08:27:08.588 PM]
_setValue(_password("user_password"), "sree12") [154 ms] [08:27:08.742 PM]
_click(_submit("login")) [239 ms] [08:27:08.981 PM]
_click(_link("Go back to the Login Page")) [424 ms] [08:27:09.405 PM]
_setValue(_textbox("user_name"), "sree") [893 ms] [08:27:10.298 PM]
_setValue(_password("user_password"), "sree123") [159 ms] [08:27:10.457 PM]
_click(_div("row[3]")) [246 ms] [08:27:10.703 PM]
_click(_submit("login")) [141 ms] [08:27:10.844 PM]
_click(_listitem(14)) [582 ms] [08:27:11.426 PM]
_click(_button("Get it now[1]")) [170 ms] [08:27:11.596 PM]
_setValue(_textbox("Quantity:"), "5") [506 ms] [08:27:12.102 PM]
_click(_button("Add to cart")) [241 ms] [08:27:12.343 PM]
_assertEqual("Arsenal Away Men's Jersey (M) was added to the cart!", _lastAlert()) [151 ms] [08:27:12.494 PM]
_click(_image("logo.png")) [130 ms] [08:27:12.624 PM]
_click(_link("Arsenal Away Men's Jersey (M)")) [379 ms] [08:27:13.003 PM]
_setValue(_textbox("Quantity:"), "2") [360 ms] [08:27:13.363 PM]
_click(_button("Add to cart")) [229 ms] [08:27:13.592 PM]
```

Figure 19: Sahi logs

PASS/FAIL logs have been generated automatically, where we can clearly view the pass log and fail log. In the log the script has also been projected with the time taken to execute each and every step. In the Fail log, it will be clearly shown at what step the execution has been failed.

2.5.1.3: Selenium IDE:

Register functionality:

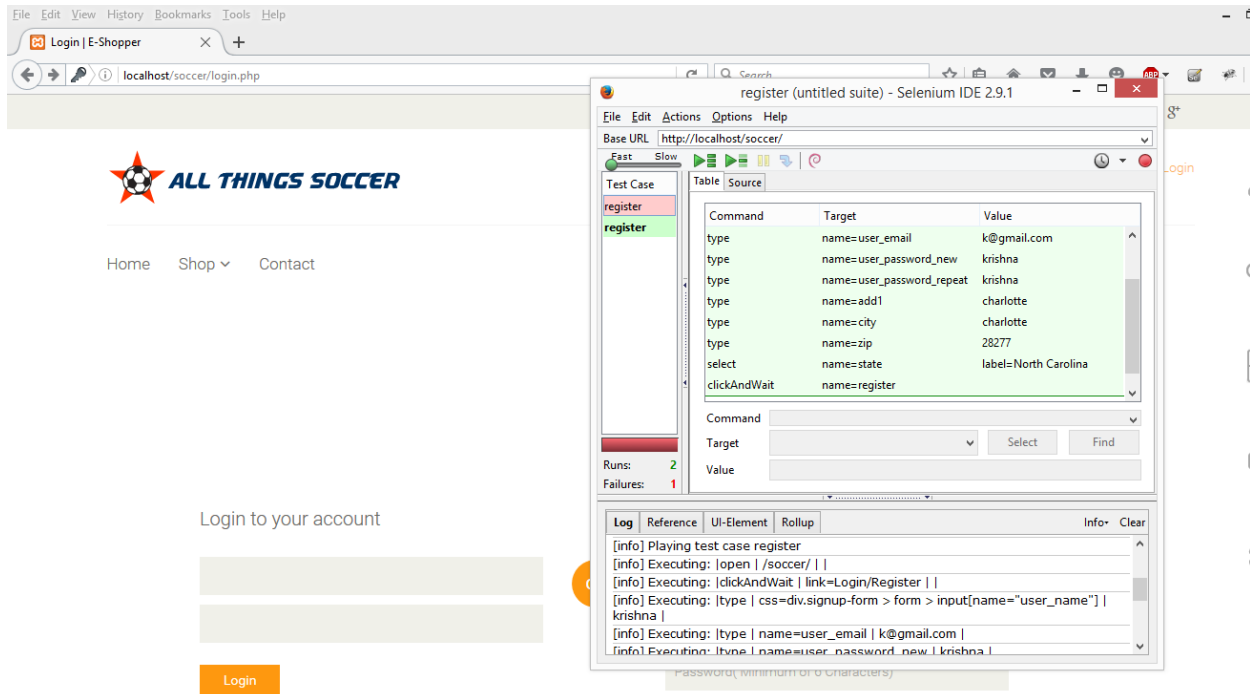


Figure 20: Selenium IDE Register

To test the register functionality in the soccer web application, selenium IDE has recorded the process that have to be done by tester in Firefox and automatically generated a script which automatically navigates to the register page and enter the mentioned username, email, password, re-enter password, address line 1 and line 2, city, zip code and state in the code. The execution can be viewed by selecting the register test case and clicking on the above play button.

Login/add to cart/Logout functionalities:

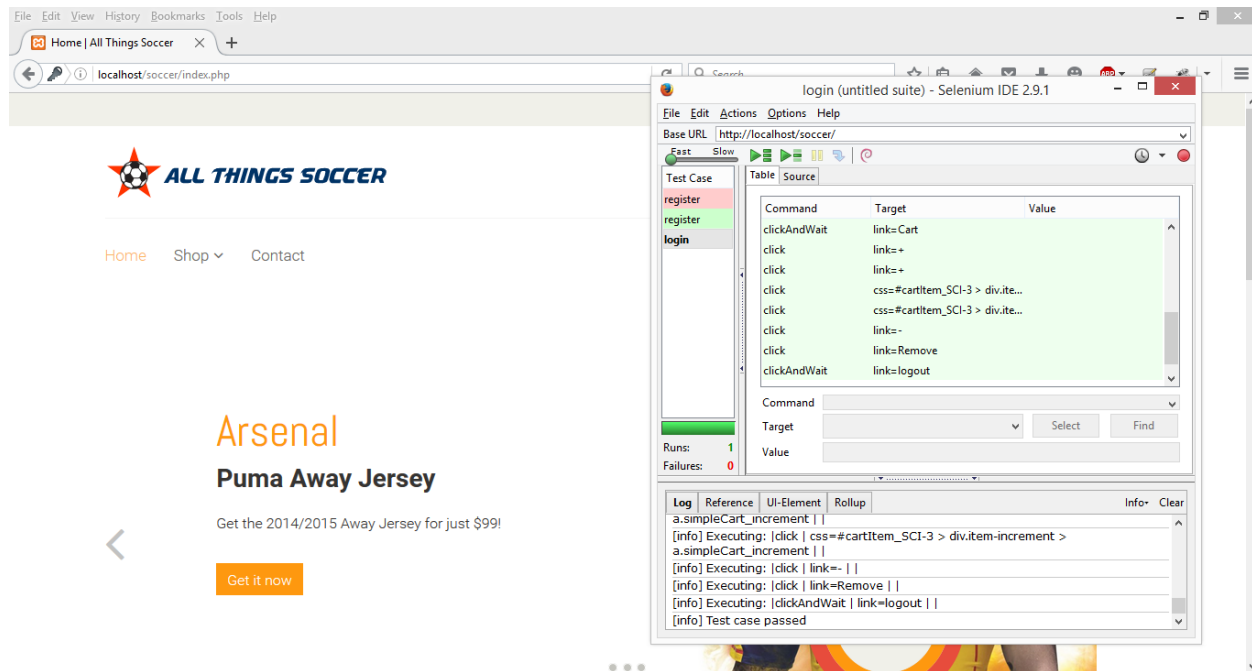


Figure 21: Selenium IDE Login

To test the login, add to cart and logout functionality in the soccer web application, Selenium IDE has recorded the process that have done by tester in Firefox and automatically generated a script which automatically navigates to the login page and enter the mentioned username and password in the code. If the user has been successfully login, the website redirects the user to the home page. In the home page, the product has been selected and the quantity of that product is changed and add that product to the cart and logouts the user. The execution can be viewed by selecting the login test case and clicking the above play button.

2.5.1.4: Selenium WebDriver:

Register functionality:

The screenshot displays a web interface for user registration. On the left, there is a login section with two empty text input fields and an orange 'Login' button below them. In the center, an orange circle with the text 'OR' indicates an alternative path. To the right of this is the registration form, which consists of several text input fields: 'sindu' for the username, 's@gmail.com' for the email, two fields with masked characters (dots) for the password and its confirmation, 'E 10th Street' for the address line 1, an empty field for 'Address Line 2 (Optional)', 'greenville' for the city, '27858' for the zip code, and a dropdown menu for the state currently set to 'North Carolina'. An orange 'Signup' button is located at the bottom of the registration form. A vertical scrollbar is visible on the far right edge of the page.

Figure 22: Selenium 2.0 Register

To test the register functionality in the soccer web application, a Java code has been written in Selenium in Eclipse which automatically navigates to the register page and enter the mentioned username, email, password, re-enter password, address line 1 and line 2, city, zip code and state in the code. The execution can be done by opening the script and clicking run option in eclipse and the process is viewed in the browser.

Login/logout functionality:

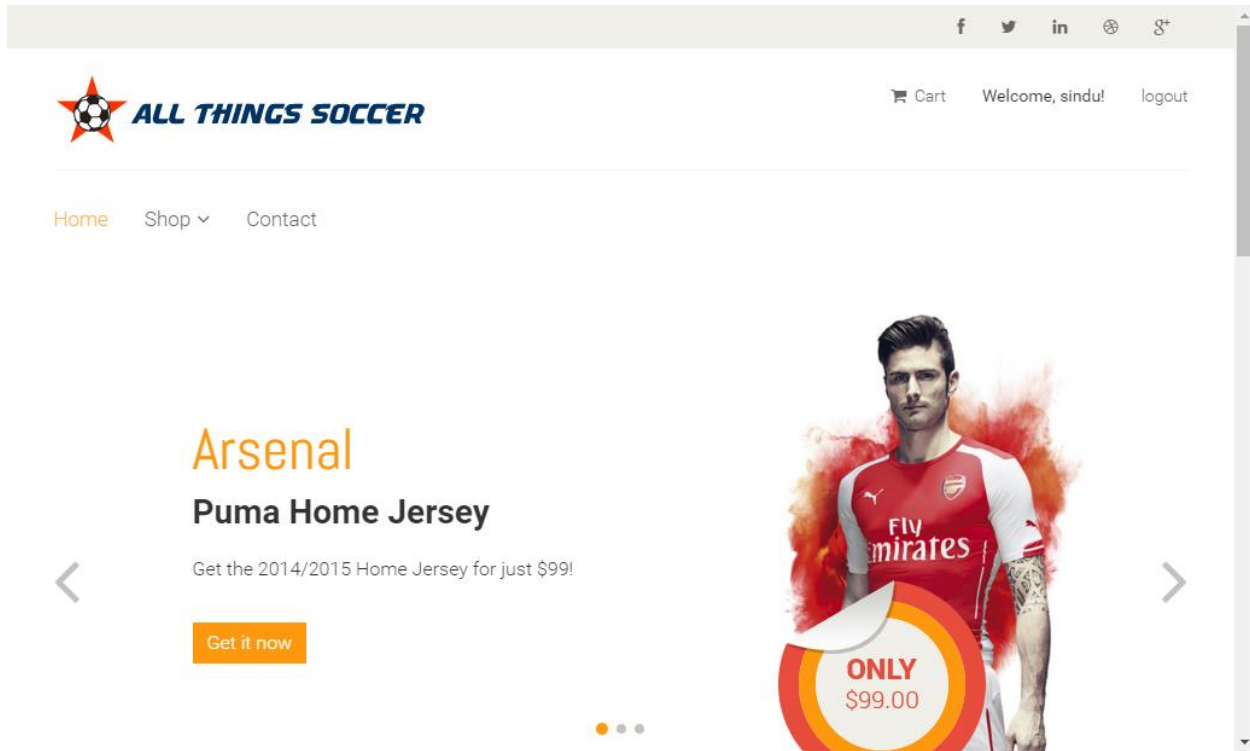


Figure 23: Selenium 2.0 login

To test the login and logout functionality in the soccer web application, a Java code has been written in Selenium in Eclipse which automatically navigates to the login page and enter the mentioned username and password in the code. If the user has been successfully login, the website redirects the user to the home page. Then the user can logout. The execution can be done by opening the script and clicking run option in eclipse and the process is viewed in the browser.

Add to cart functionality:

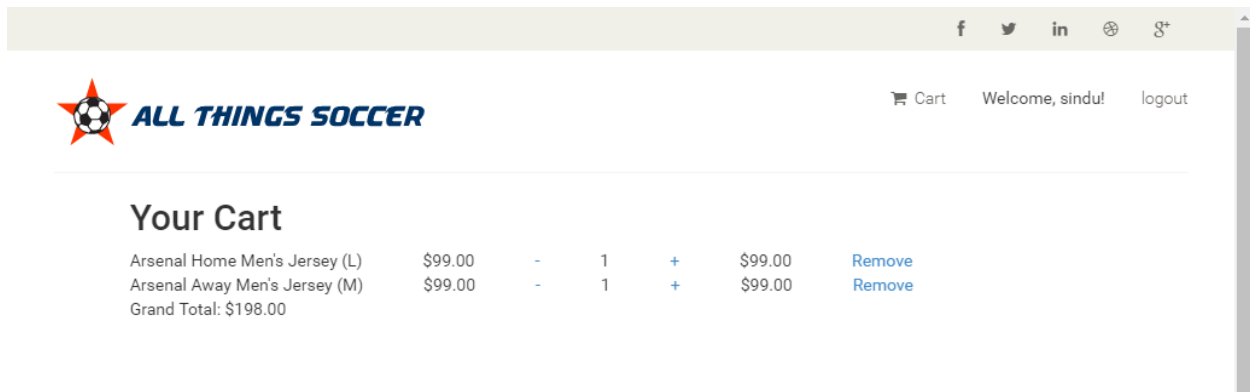


Figure 24: Selenium 2.0 Add to cart


To test the add to cart functionality in the soccer web application, a Java code has been written in Selenium in Eclipse which automatically navigates to the login page and enter the mentioned username and password in the code. If the user has been successfully login, the website redirects the user to the home page. In the home page, the code has been written to select a product and change the quantity of that product and add that product to the cart. The execution can be done by opening the script and clicking run option in eclipse and the process is viewed in the browser.

2.5.2: Failure Tests:

The soccer web application failed some test cases. Below are the logs for failed test cases.

2.5.2.1: Test case 1:

In the soccer home page, when the user wants to view the products according to sportswear category, user clicks on the brands in the sportswear category the page gets refresh and instead of displaying the products according to the condition the all products are being displayed.

Starting script [Expand All](#) [Collapse All](#) 

_click(_link("Sportswear")) [1996 ms] [11:34:34.442 PM]

_click(_link("Nike")) [192 ms] [11:34:34.634 PM]

_assertEqual("\$99 Arsenal Home Men's Jersey (L) Add to cart", _getText(_div("\$99 Arsenal Home Men's Jersey (L) Add to cart")))

_click(_listItem("Adidas")) [126 ms] [11:34:35.123 PM]

_click(_link("Adidas")) [140 ms] [11:34:35.263 PM]

_assertEqual("\$99 Arsenal Home Men's Jersey (L) Add to cart", _getText(_div("\$45 Chelsea Home Men's Jersey (M) Add to cart")))

[Assertion Failed]

Expected: "\$99 Arsenal Home Men's Jersey (L) Add to cart"

Actual: "\$45 Chelsea Home Men's Jersey (M) Add to cart"

at: (C:\Users\kiran\sahi_pro\userdata\scripts\bug1.sah&n=9)

[+] onScriptFailureDefault([object])

_click(_section("slider")) [72 ms] [11:34:50.766 PM]

_click(_link("Puma")) [117 ms] [11:34:50.883 PM]

_assertEqual("\$99 Arsenal Home Men's Jersey (L) Add to cart", _getText(_div("\$99 Arsenal Alternate Men's Jersey (L) Add to cart")))

[Assertion Failed]

Expected: "\$99 Arsenal Home Men's Jersey (L) Add to cart"

Actual: "\$99 Arsenal Alternate Men's Jersey (L) Add to cart"

at: (C:\Users\kiran\sahi_pro\userdata\scripts\bug1.sah&n=13)

[+] onScriptFailureDefault([object])

Stopping script

Figure 25: Test case 1

2.5.2.2: Test case 2:

In the soccer home page, when the user wants to view the products according to MENS category, when the user clicks on the brands in the MENS category the web page gets refresh and instead of displaying the products according to the condition all products are being displayed.

Starting script [Expand All](#) [Collapse All](#) ☐

```

_click(_link("Mens"))    [1108 ms] [11:36:11.040 PM]
_click(_link("Puma[1]")) [218 ms] [11:36:11.258 PM]
_assertEqual("$99 Arsenal Home Men's Jersey (L) Add to cart", _getText(_div("$99 Arsenal Alternate Men's Jersey (L) Add to cart")))
[Assertion Failed]
Expected: "$99 Arsenal Home Men's Jersey (L) Add to cart"
Actual: "$99 Arsenal Alternate Men's Jersey (L) Add to cart"
at: (C:\Users\kiran\sahi_pro\userdata\scripts\bug2.sah&n=5)

[+] onScriptFailureDefault([object])
_click(_link("Adidas[1]")) [85 ms] [11:36:26.119 PM]
_assertEqual("$99 Arsenal Home Men's Jersey (L) Add to cart", _getText(_div("$45 Chelsea Home Men's Jersey (M) Add to cart")))
[Assertion Failed]
Expected: "$99 Arsenal Home Men's Jersey (L) Add to cart"
Actual: "$45 Chelsea Home Men's Jersey (M) Add to cart"
at: (C:\Users\kiran\sahi_pro\userdata\scripts\bug2.sah&n=8)

[+] onScriptFailureDefault([object])
_click(_link("Nike[1]")) [58 ms] [11:36:41.404 PM]
_click(_link("Nike")) [122 ms] [11:36:41.526 PM]
_assertEqual("$99 Arsenal Home Men's Jersey (L) Add to cart", _getText(_div("$99 Arsenal Home Men's Jersey (L) Add to cart")))
Stopping script

```

Figure 26: Test case 2

2.5.2.3: Test case 3:

In the soccer home page, when the user wants to view the products according to WOMENS category, when the user clicks on the brands in the WOMENS category the web page gets refresh and instead of displaying the products according to the condition all products are being displayed.

Starting script [Expand All](#) [Collapse All](#) ☐

```

_click(_link("Womens")) [1714 ms] [11:36:50.089 PM]
_click(_link("Adidas[2]")) [186 ms] [11:36:50.275 PM]
_assertEqual("$99 Arsenal Home Men's Jersey (L) Add to cart", _getText(_div("$45 Chelsea Home Men's Jersey (M) Add to cart")))
[Assertion Failed]
Expected: "$99 Arsenal Home Men's Jersey (L) Add to cart"
Actual: "$45 Chelsea Home Men's Jersey (M) Add to cart"
at: (C:\Users\kiran\sahi_pro\userdata\scripts\bug3.sah&n=6)

[+] onScriptFailureDefault([object])
Stopping script

```

Figure 27: Test case 3

2.5.2.4: Test case 4:

In the soccer home page, when the user wants to view the products according to Brands, when the user clicks on the brands displayed in Brands category, the web page gets refresh and instead of displaying the products according to the condition all products are being displayed.

Starting script [Expand All](#) [Collapse All](#) ☐

```
_assertEqual("3", _count("_image", "Item Alt", _link("Home")))  
[Assertion Failed]  
Expected:"3"  
Actual:6  
at: (C:\Users\kiran\sahi_pro\userdata\scripts\bug4.sah&n=5)
```

[+] onScriptFailureDefault([object])

```
_assertEqual("3", _count("_image", "Item Alt", _link("Home")))  
[Assertion Failed]  
Expected:"3"  
Actual:6  
at: (C:\Users\kiran\sahi_pro\userdata\scripts\bug4.sah&n=8)
```

[+] onScriptFailureDefault([object])

```
_assertEqual("1", _count("_image", "Item Alt", _link("Home")))  
[Assertion Failed]  
Expected:"1"  
Actual:6  
at: (C:\Users\kiran\sahi_pro\userdata\scripts\bug4.sah&n=11)
```

[+] onScriptFailureDefault([object])

Stopping script

Figure 28: Test case 4

2.5.2.5: Test case 5:

In the soccer home page, when the user wants to view the products according to price range, the web page instead of displaying the products according to the condition all products are being displayed.

Starting script [Expand All](#) [Collapse All](#) ☐

```
_sahi.setServerVarForFetch("__lastValue__1461814828308", _getText(_heading2("$252")));
```

```
Error: The parameter passed to _getText was not found on the browser  
at: (C:\Users\kiran\sahi_pro\userdata\scripts\bug5.sah&n=3)
```

[+] **onScriptErrorDefault([object])**

Stopping script

Figure 29: Test case 5

Chapter 3: Conclusion

- Automated testing is an efficient and effective process for large web projects which saves both money and time.
- A web application has been tested using three automated testing tools and success and failure test cases have been found.
- Each automated testing tool has its own advantages and disadvantages. It depends on the individuals to choose which tool works for them well.
- CasperJs testing tool is useful if the individual has JavaScript knowledge.
- Sahi is a commercial tool, which is useful for large applications and the individuals need to have SahiScript knowledge.
- Through my evaluation, I found that Selenium works for every individual as it supports many programming languages and also supports record and playback features on Firefox where the scripts can be reusable and run on any browser using WebDriver.

References

Selenium IDE: http://www.seleniumhq.org/docs/02_selenium_ide.jsp.

Selenium WebDriver: http://www.seleniumhq.org/docs/03_webdriver.jsp.

CasperJs: <http://docs.casperjs.org/en/latest/quickstart.html>.

Sahi: <https://www.thoughtworks.com/insights/blog/introduction-sahi-part-1>.

A. Holmes and M. Kellogg. Automating functional tests using Selenium. In Proceedings of AGILE 2006, pages 270-275. IEEE, 2006.

Andreas Bruns, Andreas Kornstädt, Dennis Wichmann, "Web Application Tests with Selenium", IEEE Software, vol.26, no. 5, pp. 88-91, September/October 2009, doi:10.1109/MS.2009.144

<https://sahipro.com/docs/introduction/index.html>.

Appendix: code

CasperJs:

Register

```
var casper = require('casper').create({  
    verbose: true,  
    logLevel: "debug",  
});  
  
var x = require('casper').selectXPath;  
casper.start('http://localhost/soccer/');
```

```
//REGISTER
casper.then(function() {
    casper.click("a[href='login.php']");});
casper.then(function() {
    //fill form
    this.fill('div.signup-form', {
        user_name : 'sindu',
        user_email : 'sindu@gmail.com',
        user_password_new : 'sindu11',
        user_password_repeat : 'sindu11',
        add1 : 'E 10th street',
        city : 'greenville',
        zip : '27858',
        state : 'North Carolina'
    }, true); });
casper.then(function() { this.capture('register.png');});
casper.then(function() {
    casper.click("button[name='register']");});
casper.run();
```

Login/Add to cart/Logout

```
var casper = require('casper').create({
    verbose: true,
    logLevel: "debug",
});
casper.start('http://localhost/soccer/');
//LOGIN
```

```

casper.then(function() {
    casper.click("a[href='login.php']");});
casper.then(function() {
    //fill form
    this.fill('div.login-form', {
        user_name : 'sindu',
        user_password : 'sindu11'
    }, true); });
casper.then(function() {
    this.waitForSelector('.btn.btn-default', function() {
        this.click('.btn.btn-default');
    });});
casper.then(function() {
    this.capture('login.png');    });
//ITEM
casper.then(function() {
    casper.click("a[href='product-details.php?listing_id=2']");});
casper.then(function() {
    this.waitForSelector('.item_quantity', function() {
        this.sendKeys('.item_quantity', '2'); }); });
casper.then(function() {
    this.waitForSelector('.item_add.btn.btn-fefault.cart', function() {
        this.click('.item_add.btn.btn-fefault.cart'); }); });
casper.waitForAlert(function(response) {
    this.echo("Alert received: " + response.data);
    return true; });
casper.then(function() {

```

```

        this.capture('item.png');    });
casper.then(function() {
    this.waitForSelector('.logo.pull-left>a>img', function() {
        this.click('.logo.pull-left>a>img'); }); });
casper.then(function() {
    this.capture('home.png');});
casper.then(function() {
    this.waitForSelector('.item_add.btn.btn-default', function() {
        this.click('.item_add.btn.btn-default'); }); });
casper.waitForAlert(function(response) {
    this.echo("Alert received: " + response.data);
    return true;});
//CART
casper.then(function() {
    casper.click("a[href='cart.php']");});
casper.then(function() {
    this.waitForSelector('.simpleCart_increment', function() {
        this.click('.simpleCart_increment'); }); });
casper.then(function() {
    this.waitForSelector('.simpleCart_increment', function() {
        this.click('.simpleCart_increment'); }); });
casper.then(function() {
    this.capture('increcart.png'); });
casper.then(function() {
    this.waitForSelector('.simpleCart_decrement', function() {
        this.click('.simpleCart_decrement'); }); });
casper.then(function() {

```

```

        this.capture('cart.png');    });
casper.then(function() {
    this.waitForSelector('.simpleCart_remove', function() {
        this.click('.simpleCart_remove');    });});
casper.then(function() {
    this.capture('finalcart.png');    });
//LOGOUT
casper.then(function() {
    casper.click("a[href='login.class.php?logout=true']");});
casper.then(function() {
    this.capture('logout.png');    });
casper.run();

```

Failure Test cases:

1.

```

var casper = require('casper').create({
    verbose: true,
    logLevel: "debug",
});
casper.start('http://localhost/soccer/');
casper.then(function() {
    casper.click("a[href='#sportswear']");});
casper.then(function() {

```

```

this.waitForSelector('.panel-body>ul>li>a', function() {
this.click('.panel-body>ul>li>a'); }); });
casper.then(function() {
this.test.assertEquals("(Chelsea Home Men's Jersey (M))", "(Arsenal Home Men's Jersey (L))" ,
"true");
this.exit();
});
casper.run();

```

```

2. var casper = require('casper').create({
  verbose: true,
  logLevel: "debug",
});
casper.start('http://localhost/soccer/');
casper.then(function() {
casper.click("a[href='#mens']");});
casper.then(function() {
this.waitForSelector('.panel-body>ul>li>a', function() {
this.click('.panel-body>ul>li>a'); }); });
casper.then(function() {
this.test.assertEquals("(Chelsea Home Men's Jersey (M))", "(Arsenal Home Men's Jersey (L))" ,
"true");
this.exit();
});
casper.run();

```

3.

```

var casper = require('casper').create({
  verbose: true,
  logLevel: "debug",
});
casper.start('http://localhost/soccer/');
casper.then(function() {
  casper.click("a[href='#womens']");});
casper.then(function() {
  this.waitForSelector('.panel-body>ul>li>a', function() {
    this.click('.panel-body>ul>li>a'); }); });
casper.then(function() {
  this.test.assertEquals("(Chelsea Home Men's Jersey (M))", "(Arsenal Home Men's Jersey (L))" ,
    "true");
  this.exit();
});
casper.run();

```

4.

```

var casper = require('casper').create({
  verbose: true,
  logLevel: "debug",
});
casper.start('http://localhost/soccer/');
casper.then(function() {
  this.waitForSelector('.brands_products>h2', function() {
    this.click('.brands_products>h2'); }); });
casper.then(function() {

```



```

this.waitForSelector('.nav.nav-pills.nav-stacked>li>a', function() {
this.click('.nav.nav-pills.nav-stacked>li>a'); }); });

casper.then(function() {
this.test.assertEquals("(Chelsea Home Men's Jersey (M))", "(Arsenal Home Men's Jersey (L))" ,
"true");
this.exit();
});
casper.run();

```

5.

```

var casper = require('casper').create({
  verbose: true,
  logLevel: "debug",
});
casper.start('http://localhost/soccer/');

```

```

casper.then(function() {
this.waitForSelector('.price-range>h2', function() {
this.click('.price-range>h2'); }); });
casper.then(function() {
this.waitForSelector('.tooltip-inner', function() {
this.echo("250 : 450"); }); });
casper.then(function() {
this.test.assertEquals("$250", "$99" , "true");
this.exit();
});

```

```
casper.run();
```

SAHI

Register

```
_click(_link("Login/Register"));
_setValue(_textbox("user_name[1]"), "sree");
_setValue(_emailbox("user_email"), "sree@gmail.com");
_setValue(_password("user_password_new"), "sree123");
_setValue(_password("user_password_repeat"), "sree123");
_setValue(_textbox("add1"), "charlotte");
_setValue(_textbox("city"), "charlotte");
_setValue(_textbox("zip"), "28277");
_setSelected(_select("state"), "North Carolina");
_click(_submit("register"));
```

Login/Add to cart/Logout

```
_click(_link("Login/Register"));
_setValue(_textbox("user_name"), "sree");
_setValue(_password("user_password"), "sree12");
_click(_submit("login"));
_click(_link("Go back to the Login Page"));
_setValue(_textbox("user_name"), "sree");
_setValue(_password("user_password"), "sree123");
_click(_div("row[3]"));
_click(_submit("login"));
```

```

_click(_listItem(14));
_click(_button("Get it now[1]"));
_setValue(_textbox("Quantity:"), "5");
_click(_button("Add to cart"));
_assertEqual("Arsenal Away Men's Jersey (M) was added to the cart!", _lastAlert());
_click(_image("logo.png"));
_click(_link("Arsenal Away Men's Jersey (M)"));
_setValue(_textbox("Quantity:"), "2");
_click(_button("Add to cart"));
_assertEqual("Arsenal Away Men's Jersey (M) was added to the cart!", _lastAlert());
_click(_image("logo.png"));
_click(_link("Add to cart"));
_assertEqual("Arsenal Home Men's Jersey (L) was added to the cart!", _lastAlert());
_click(_link("Cart"));
_click(_link("-"));
_click(_link("+[1]"));
_click(_link("Remove[1]"));
_click(_link("-"));
_click(_link("logout"));

```

Failure Test Cases:

1.

```

_getText(_link("Sportswear"));
_click(_link("Sportswear"));
_click(_link("Nike"));
_getText(_div("$99 Arsenal Home Men's Jersey (L) Add to cart"))

```

```

_assertEqual("$99 Arsenal Home Men's Jersey (L) Add to cart", _getText(_div("$99 Arsenal
Home Men's Jersey (L) Add to cart")));

_click(_listItem("Adidas"));

_click(_link("Adidas"));

_getText(_div("$45 Chelsea Home Men's Jersey (M) Add to cart"));

_assertEqual("$99 Arsenal Home Men's Jersey (L) Add to cart", _getText(_div("$45 Chelsea
Home Men's Jersey (M) Add to cart")));

_click(_section("slider"));

_click(_link("Puma"));

_getText(_div("$99 Arsenal Alternate Men's Jersey (L) Add to cart"))

_assertEqual("$99 Arsenal Home Men's Jersey (L) Add to cart", _getText(_div("$99 Arsenal
Alternate Men's Jersey (L) Add to cart")));

```

2.

```

_getText(_link("Mens"))

_click(_link("Mens"));

_click(_link("Puma[1]"));

_getText(_div("$99 Arsenal Alternate Men's Jersey (L) Add to cart"))

_assertEqual("$99 Arsenal Home Men's Jersey (L) Add to cart", _getText(_div("$99 Arsenal
Alternate Men's Jersey (L) Add to cart")));

_click(_link("Adidas[1]"));

_getText(_div("$45 Chelsea Home Men's Jersey (M) Add to cart"))

_assertEqual("$99 Arsenal Home Men's Jersey (L) Add to cart", _getText(_div("$45 Chelsea
Home Men's Jersey (M) Add to cart")));

_click(_link("Nike[1]"));

_click(_link("Nike"));

_getText(_div("$99 Arsenal Home Men's Jersey (L) Add to cart"))

_assertEqual("$99 Arsenal Home Men's Jersey (L) Add to cart", _getText(_div("$99 Arsenal
Home Men's Jersey (L) Add to cart")));

```

3.

```
_count("_link","Arsenal Home Men's Jersey (L)",_link("Home"));
_getText(_link("Womens"));
_click(_link("Womens"));
_click(_link("Adidas[2]"));
_getText(_div("$45 Chelsea Home Men's Jersey (M) Add to cart"))
_assertEqual("$99 Arsenal Home Men's Jersey (L) Add to cart", _getText(_div("$45 Chelsea
Home Men's Jersey (M) Add to cart"));
```

4.

```
_count("_image","Item Alt",_link("Home"));
_getText(_heading2("Brands"))
_getText(_link("(3)Nike"));
_count("_image","Item Alt",_link("Home"));
_assertEqual("3", _count("_image","Item Alt",_link("Home")));
_getText(_link("(3)Puma"));
_count("_image","Item Alt",_link("Home"));
_assertEqual("3", _count("_image","Item Alt",_link("Home")));
_getText(_link("(1)Adidas"));
_count("_image","Item Alt",_link("Home"));
_assertEqual("1", _count("_image","Item Alt",_link("Home")));
```

5.

```
_getText(_heading2("Price Range"))
```

```
_getText(_div("250 : 450"))  
_getText(_heading2("$252"));
```

Selenium:

Register/Login/Add to cart/Logout

```
package soccer;  
  
import java.io.File;  
  
import java.io.IOException;  
  
import java.util.concurrent.TimeUnit;  
  
import org.apache.commons.io.FileUtils;  
  
import org.openqa.selenium.By;  
  
import org.openqa.selenium.OutputType;  
  
import org.openqa.selenium.TakesScreenshot;  
  
import org.openqa.selenium.WebDriver;  
  
import org.openqa.selenium.chrome.ChromeDriver;  
  
import org.openqa.selenium.support.ui.Select;  
  
public class Register {  
  
    public static void main(String[] args) throws IOException {  
  
        System.setProperty("webdriver.chrome.driver",C:\\Users\\kiran\\Desktop\\Selenium\\servers\\  
        \\chromedriver.exe");  
  
        WebDriver driver=new ChromeDriver();  
  
        driver.get("http://localhost/soccer/");  
  
        driver.manage().timeouts().implicitlyWait(50, TimeUnit.SECONDS);  
  
        //register  
  
        driver.findElement(By.xpath(".*[@id='header']/div[2]/div/div/div[2]/div/ul/li[2]/a")).click();  
  
        driver.findElement(By.xpath(".*[@id='form']/div/div/div[3]/div/form/input[1]")).sendKeys("si  
ndu");
```

```

driver.findElement(By.xpath(".*[@id='form']/div/div/div[3]/div/form/input[2]")).sendKeys("@gmail.com");

driver.findElement(By.xpath(".*[@id='form']/div/div/div[3]/div/form/input[3]")).sendKeys("sindu11");

driver.manage().timeouts().implicitlyWait(50, TimeUnit.SECONDS);

driver.findElement(By.xpath(".*[@id='form']/div/div/div[3]/div/form/input[4]")).sendKeys("sindu11");

driver.findElement(By.xpath(".*[@id='form']/div/div/div[3]/div/form/input[5]")).sendKeys("E 10th Street");

driver.findElement(By.xpath(".*[@id='form']/div/div/div[3]/div/form/input[7]")).sendKeys("greenville");

driver.findElement(By.xpath(".*[@id='form']/div/div/div[3]/div/form/input[8]")).sendKeys("27858");

Select dropdown= new
Select(driver.findElement(By.xpath(".*[@id='form']/div/div/div[3]/div/form/select")));

dropdown.selectByVisibleText("North Carolina");

File srcFile=((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);

FileUtils.copyFile(srcFile, new File("C:\\Users\\kiran\\Desktop\\register.png"));

driver.findElement(By.xpath(".*[@id='form']/div/div/div[3]/div/form/button")).click();

driver.get("http://localhost/soccer/");

//login

driver.findElement(By.xpath(".*[@id='header']/div[2]/div/div/div[2]/div/ul/li[2]/a")).click();

driver.findElement(By.xpath(".*[@id='login_input_username']")).sendKeys("sindu");

driver.findElement(By.xpath(".*[@id='login_input_password']")).sendKeys("sindu11");

driver.findElement(By.xpath(".*[@id='form']/div/div/div[1]/div/form/button")).click();

WebDriverWait w1= new WebDriverWait(driver,20);

w1.until(ExpectedConditions.visibilityOfElementLocated(By.xpath(".*[@id='header']/div[2]/div/div/div[1]/div[1]/a/img")));

driver.findElement(By.xpath(".*[@id='header']/div[2]/div/div/div[1]/div[1]/a/img")).click();

```

```

driver.findElement(By.cssSelector(".item_add.btn.btn-default")).click();

System.out.println(driver.switchTo().alert().getText());

driver.switchTo().alert().accept();

driver.findElement(By.xpath(".*[@id='header']/div[2]/div/div/div[2]/div/ul/li[1]/a")).click();
int i=1;

while(i<5){

driver.findElement(By.cssSelector(".simpleCart_increment")).click();

i++; }

int j=1;

while(j<3){

driver.findElement(By.cssSelector(".simpleCart_decrement")).click();

j++; }

driver.findElement(By.xpath(".*[@id='header']/div[2]/div/div/div[2]/div/ul/li[3]/a")).click();} }

```

Failure Test Cases:

1.

```

package soccer;

import org.junit.Assert;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.interactions.Actions;

public class Bugs {

public static void main(String[] args) throws InterruptedException {

System.setProperty("webdriver.chrome.driver",
"C:\\Users\\kiran\\Desktop\\Selenium\\servers\\chromedriver.exe");

```



```

WebDriver driver=new ChromeDriver();

driver.get("http://localhost/soccer/");


String s3 = driver.findElement(By.cssSelector(".price-range>h2")).getText();

System.out.println(s3);

System.out.println("Before Price Range");

driver.findElement(By.xpath("html/body/section[2]/div/div/div[1]/div/div[3]/div/div/div[1]"
)).click();

String s2 = driver.findElement(By.cssSelector(".tooltip-inner")).getText();

System.out.println(s2);

WebElement sliderA = driver.findElement(By.cssSelector(".slider-handle.round.left-
round"));

WebElement sliderB = driver.findElement(By.cssSelector(".slider-handle.round.left-
round"));

Actions action = new Actions(driver);

for (int i = 1; i < 2; i++)

{action.dragAndDropBy(sliderA, 50, 0).build().perform();

    Thread.sleep(300);

    action.dragAndDropBy(sliderB, 50, 0).build().perform();

    Thread.sleep(300);

}

System.out.println("After Price Range");

driver.findElement(By.xpath("html/body/section[2]/div/div/div[1]/div/div[3]/div/div/div[1]"
)).click();

String s5 = driver.findElement(By.cssSelector(".tooltip-inner")).getText();

    System.out.println(s5);

Assert.assertEquals ("$450", driver.findElement(By.cssSelector(".item_price")).getText());

}

```

```
}
```

2.

```
public class Bug2 {

    public static void main(String[] args) {

        System.setProperty("webdriver.chrome.driver",
"C:\\Users\\kiran\\Desktop\\Selenium\\servers\\chromedriver.exe");
        WebDriver driver=new ChromeDriver();
        driver.get("http://localhost/soccer/");
        String s6 = driver.findElement(By.cssSelector(".panel-title")).getText();
        System.out.println(s6);
        driver.findElement(By.xpath(".*[@id='accordian']/div[1]/div[1]/h4")).click();
        driver.findElement(By.xpath(".*[@id='sportswear']/div/ul/li[1]/a")).click();
        driver.findElement(By.xpath("html/body/section[2]/div/div/div[2]/div[1]/div[1]/div/div/div/div
/h2")).click();
        System.out.println("NIKE");
        String s4 = driver.findElement(By.cssSelector(".item_name>a")).getText();
        System.out.println(s4);
        System.out.println(s6);
        System.out.println("ADIDAS");
        driver.findElement(By.xpath(".*[@id='sportswear']/div/ul/li[3]/a")).click();
        Assert.assertEquals ("Chelsea Home Men's Jersey (M)",
        driver.findElement(By.cssSelector(".item_name>a")).getText());
        System.out.println(s6);
        System.out.println("PUMA");
        driver.findElement(By.xpath(".*[@id='sportswear']/div/ul/li[4]/a")).click();
        Assert.assertEquals ("Arsenal Alternate Men's Jersey (L)",
        driver.findElement(By.cssSelector(".item_name>a")).getText());
    }
}
```

3.

```
package soccer;
import org.junit.Assert;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
```

```

public class Bug3 {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver",
"C:\\Users\\kiran\\Desktop\\Selenium\\servers\\chromedriver.exe");
        WebDriver driver=new ChromeDriver();
        driver.get("http://localhost/soccer/");
        String s7 = driver.findElement(By.cssSelector(".left-sidebar>h2")).getText();
        System.out.println(s7);
        String s6 = driver.findElement(By.xpath(".*[@id='accordian']/div[2]/div[1]/h4")).getText();
        System.out.println(s6);
        driver.findElement(By.xpath(".*[@id='accordian']/div[2]/div[1]/h4/a/span")).click();
        driver.findElement(By.xpath(".*[@id='mens']/div/ul/li[1]/a")).click();
        System.out.println("NIKE");
        driver.findElement(By.xpath(".*[@id='mens']/div/ul/li[3]/a")).click();
        String s4 = driver.findElement(By.cssSelector(".item_name>a")).getText();
        System.out.println(s4);
        System.out.println(s6);
        System.out.println("PUMA");
        Assert.assertEquals ("Arsenal Alternate Men's Jersey (L)",
        driver.findElement(By.cssSelector(".item_name>a")).getText());
        System.out.println(s6);
        System.out.println("ADIDAS");
        driver.findElement(By.xpath(".*[@id='mens']/div/ul/li[2]/a")).click();
        Assert.assertEquals ("Chelsea Home Men's Jersey (M)",
        driver.findElement(By.cssSelector(".item_name>a")).getText());
    }
}

```

4.

```

package soccer;

import org.junit.Assert;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class Bug4 {

    public static void main(String[] args) {

```

```

System.setProperty("webdriver.chrome.driver",
"C:\\Users\\kiran\\Desktop\\Selenium\\servers\\chromedriver.exe");

WebDriver driver=new ChromeDriver();

driver.get("http://localhost/soccer/");

String s7 = driver.findElement(By.cssSelector(".left-sidebar>h2")).getText();

System.out.println(s7);

String s6 = driver.findElement(By.xpath(".*[@id='accordian']/div[3]/div[1]/h4")).getText();

System.out.println(s6);

driver.findElement(By.xpath(".*[@id='accordian']/div[3]/div[1]/h4/a/span/i")).click();

driver.findElement(By.xpath(".*[@id='womens']/div/ul/li[1]/a")).click();

System.out.println("PUMA");

Assert.assertEquals ("Chelsea Away Women's Jersey (M)",
driver.findElement(By.cssSelector(".item_name>a")).getText());

}

}

```

5.

```

package soccer;

import org.junit.Assert;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

public class Bug5 {

    public static void main(String[] args) {

        System.setProperty("webdriver.chrome.driver",
"C:\\Users\\kiran\\Desktop\\Selenium\\servers\\chromedriver.exe");

        WebDriver driver=new ChromeDriver();

        driver.get("http://localhost/soccer/");
    }
}

```

```

String s1 =
driver.findElement(By.xpath("html/body/section[2]/div/div/div[1]/div/div[2]/h2")).getText();

System.out.println(s1);


System.out.println("Count");

String s2 =
driver.findElement(By.xpath("html/body/section[2]/div/div/div[1]/div/div[2]/div/ul/li[1]/a")).ge
tText();

System.out.println(s2);

driver.findElement(By.xpath("html/body/section[2]/div/div/div[1]/div/div[2]/div/ul/li[1]/a")).cli
ck();

Assert.assertEquals ("3", driver.findElements(By.cssSelector(".item_name>a")).size());

System.out.println("Count");

String s3 =
driver.findElement(By.xpath("html/body/section[2]/div/div/div[1]/div/div[2]/div/ul/li[2]/a")).ge
tText();

System.out.println(s3);

driver.findElement(By.xpath("html/body/section[2]/div/div/div[1]/div/div[2]/div/ul/li[2]/a")).cli
ck();

System.out.println("total number of items present in the page");

Assert.assertEquals ("3", driver.findElements(By.cssSelector(".item_name>a")).size());

System.out.println("Count");

String s4 =
driver.findElement(By.xpath("html/body/section[2]/div/div/div[1]/div/div[2]/div/ul/li[3]/a")).ge
tText();

System.out.println(s4);

driver.findElement(By.xpath("html/body/section[2]/div/div/div[1]/div/div[2]/div/ul/li[3]/a")).cli
ck();

System.out.println("total number of items present in the page");

Assert.assertEquals ("1", driver.findElements(By.cssSelector(".item_name>a")).size());

```

```
System.out.println("Count");

String s5 =
driver.findElement(By.xpath("html/body/section[2]/div/div/div[1]/div/div[2]/div/ul/li[4]/a")).ge
tText();

System.out.println(s5);

driver.findElement(By.xpath("html/body/section[2]/div/div/div[1]/div/div[2]/div/ul/li[4]/a")).cli
ck();

System.out.println("total number of items present in the page");

Assert.assertEquals ("1", driver.findElements(By.cssSelector(".item_name>a")).size());
}
}
```