**Instructions:**

Test Scenario:

1. Login using existing account

2. Perform a search on home page from a pool of key words given below

3. Identify an item from the result set that you can add to cart

4. Add the item to cart

5. Validate that item added is present in the cart and is the only item in the cart

Pre-Requisites:

To run this project we need the following:

* Selenium WebDriver for Java. It can be downloaded from <http://www.seleniumhq.org/download/>
* Chrome driver(WebDriver for Chrome). It can be downloaded from <https://sites.google.com/a/chromium.org/chromedriver/getting-started>

Details of the Project:

Based on analysis of the given scenario initially I have identified the reusable functions. The project is divided mainly into 3 parts. They are Reusable functions, Test Scenario and Object Repository.

* Reusable\_Functions.java file consists of different functions like LaunchApplication, Login, SearchItem, SelectItem, ValidateCartItem, etc. which can be used in different tests.
* TestScenario.java file consists of test steps that we need to execute to get the required result. It calls the reusable functions from the Reusable\_Functions.java file and execute the steps in order. It also has input variables and web element declarations we need to pass to the functions.
* Object Repository has all the objects we use to identify the web elements. This file has all the objects like xpath, id, name, and class etc. of different web elements grouped together at one place. We can access this file in the java files and utilize the objects instead of use them directly in the code. If there is change in any object we can edit it in the object repository file and we don’t need to modify the code.

Reusable\_Functions: It contains the following functions

**LaunchApplication**: This function takes two arguments, browser name and URL. This function initializes properties of browser based on the browser (Google chrome, Firefox, Internet explorer, safari, etc.) passed to it and launches the browser with the URL passed to it.

**Login:** This function logs into the system by taking the objects, username, password passed as arguments.

**SearchItem:** This function takes objects and item as arguments. It will search the item passed to it.

**SelectItem:** This function takes objects and ItemName as arguments. It utilizes objects in the list view to select an item from the results displayed, so initially the results are viewed in list view mode by clicking on the listview object. It collects all the objects of results in a list. It traverses through the list and finds the item requested by comparing the name of item passed with the name of the object derived from innerHTML. It clicks on the item and item page is displayed. It also collects the ‘data-item-id’ of the item which is used for verification.

**SelectItemQuantity:** This function takes objects and Quantity as arguments. It collects the dropdown list of quantity and selects a quantity passed to it from the list by comparing the visible text. It can select any number of units of an item.

**AddItemstoCart:** This function takes objects as arguments. It adds the item to the cart by clicking on the ‘Add cart’ button. If the cart button is not present it displays the message “out of stock”.

**ViewCartItems:** This function takes objects as arguments. It views the items in the cart after adding them by clicking on ‘View Cart’ button. The items in the cart are displayed.

**ValidateCartItem:** This function validates that item is added to the cart and only item present in the cart. It compares the ‘data-item-id’ derived in SelectItem function with the ‘data-us-tem-id’ of the item in the cart and the quantity passed initially to SelectItemQuantity function with the quanity in the cart. If they both match it prints validation is successful otherwise unsuccessful to the console.

Test Scenario: In it the test scenario mentioned above is executed in a sequential manner and at the end the item in the cart is validated. TestScenario.java extends Reuasble\_functions.java file to make use of the web driver and functions in it. It also contains 2 web element declarations viewcartiteminfo and viewcartquantity which are passed to ValidateCartItem. They give the ‘data-us-item-id’ and the quanity of the item in the cart.

Object Repository: Object Repository file has all the objects we use to identify the web elements in this project. We can access the contents in the file through the object ‘objects’ of ‘properties class with method ‘objects.load’. Objects is passed as argument to the reusable functions used in the test scenario to access the contents of object repository.

Test Data:

The following is the test data I have utilized:

BrowserName: Google Chrome

URL: https://www.walmart.com/account/login?returnUrl=%2Faccount%2F

Username: [vimalchandra.gorijala@sjsu.edu](mailto:vimalchandra.gorijala@sjsu.edu)

Password: sanjose

SerachItem: tv, dvd, socks, iphone

Item\_Input:

RCA LED55G55R120Q 55" 1080p 120Hz LED HDTV

VIZIO D43-C1 43" 1080p 120Hz Class LED HDTV

RCA LED32C45RQ 32" 1080p 60Hz Class LED (3.1" ultra-slim) HDTV

Hanes Men's 12 Pack No Show socks

Starter - Boys' Crew Socks, 20 Pairs

Fruit of the Loom Women's Crew Socks, 10 Pack

LG DP132 DVD Player

Verbatim DVD-R 4.7GB 16X AZO 50pk Spindle

Memorex DVD-RW Discs, 4.7GB, 4x, w/Slim Jewel Cases, Silver, 10/Pack

Technical Choices: I have used Selenium WebDriver with Java to automate the scenario. I have chosen it over other options because it supports many languages, multi browser, cross platform, integrates well with existing frameworks, has its own IDE, etc. Though watir web driver is also a good option but it is only limited to Ruby langauge and every browser requires a different library which is a drawback.

I have covered all the steps from the given scenario. The code I have implemented works for the search terms **tv**, **dvd** and **toys**. It works fine for most of the items in **socks** except for those which have color and size options associated to them. It also works fine for some of the items in **iphone** except for those which have color and capacity options associated to them. If I have time I would have implemented the select item functionality that works for all the search terms given.