# ISIT 324 Bonus Homework – Automate the BC site with Selenium

# **15 Bonus Points Possible**

For this assignment you’ll need to understand how to use the Selenium Web Driver for Chrome. You’ll also need to understand the rudiments of building a Page Object Model. The ride-along from the lecture will prove extremely helpful.

As a reference for basic Selenium functionality, a console app using most of the functions required for the homework, but without a page object model, is provided on the Assignment page.

In addition, you’ll find a solution that includes a starter class library that includes a Driver class as well as classes for two of the pages you’ll need to visit. It also includes a single unit test for your reference. Feel free to use this solution as your starting point.

## Instructions

Build the following unit tests for the Bellevue College web site (<https://www.bellevuecollege.edu/>). **You must construct and use a Page Object Model, per the “For Full Credit” section below.**

Put everything in a single TestClass named **HomePage\_Should.**

**Test 1: ReturnAboutPage\_WhenAboutClicked**

1. Start at home page
2. Click “About Us” in the top nav bar
3. Validate that you’ve landed on the About Us page.

**Test 2: ReturnListOfPages\_WhenEachMenuItemClicked()**

1. Start at home page
2. Click each menu item on the top nav bar (Home, Classes, Programs of Study, Enrollment, etc.) in turn, and then use the automated Back button to return to the home page.
3. Record the title of each page (you can determine which element to grab the title from) in a collection (List<string> is what I’d use.)
4. Validate that the collection you build during test execution matches an expected list of items.
5. **Note:** FluentAssertions are good for testing equality of collections.
6. **Note:** You’ll need a class for each page visited for full credit.

**Test 3: ReturnCorrectSearchResults\_WhenSearchingForClasses**

1. Parameterize test with three parameters:
   1. Expected return string
   2. Quarter
   3. Search string
2. Start at home page
3. Click “Classes” in the top nav bar
4. On the Classes page:
   1. Select the tab corresponding Quarter string that was passed in
   2. Fill the search box with the passed-in Search String
   3. Click the search button
   4. Check to see that the FIRST RESULT in the returned set of search results is what you expected.

Here’s the test data from number 4:

* 1. Set 1:
     1. Expected return string = “ISIT 324 Software Testing • 5 Cr.”
     2. Quarter = “Winter2020”
     3. Search string = “ISIT 324”
  2. Set 2
     1. Expected return string = “PROG 110 Introduction to Programming • 5 Cr.”
     2. Quarter = “Spring2020”
     3. Search string = “dennis minium”

**For full credit:**

Submission

* Send your entire solution as a .zip file.

Requirements

* Use MSTest V2 as your test framework.
* The unit tests must NOT use Selenium concepts directly. Everything Selenium-specific must be provided through the Page Object Model. That is, your unit tests must not declare or directly use any IWebDriver or IWebElement objects. If correctly written, your unit test .cs file will not require a using statement for anything starting with **using OpenQA.**
* Do not use the deprecated PageFactory class.
* Create a folder for the pages in your Page Object Model and create one class per page, e.g., “HomePage.cs” for the home page class and “ServicesPage.cs” for the Services page class.
* All properties in each page class must be private. You must provide public methods to return values and invoke methods.
* For parameterized tests, use standard MSTest V2 parameterized tests (i.e., use the DataTestMethod and DataRow attributes.)
* Follow the naming rules as best you can.
* Arrange your test methods using the “arrange, act, assert” pattern (including comments).
* You do **NOT** need to use the SUT alias for full credit. Doing so is pretty clunky in this instance.