Software Test Automation Engineer – Homework Write-up

Brandon Davis

Test Framework Setup:

I completed this project using a Visual Studio C# Console Application. I added the following extension to Visual Studio before beginning the project:

- NUnit3 Test Adapter

Within my console project, I also used the following references, added using the NuGet Package Manager:

- WebDriverChromeDriver v2.10.0
- Nunit v 3.2.0
- Selenium.WebDriver v2.53.0
- Selenium.Support v2.53.0

Design Approach

Using the NUnit test framework, I designed a series of unit tests that I ran using the Test Explorer. I setup each test for each case using the same [SetUp] attribute that initializes the WebDriver and opens the URL.

Each [Test] attribute utilized the Page Object Model to establish a relationship with each page to be used for the test. First, I created a page object and found all of the elements under test. Next, I wrote to the elements, read back their value, and submitted the entries. For all Text and button inputs, I searched by name or id. To locate Cost and Duration, I searched using TagName <b>, which represents bold text. They are the only two items specified as <b>. To open the calendar and select the date, I also used TagName to find links or bold-faced text. For some of the links with TagName<a>, I sorted through them by the attribute “href” and matched a string pattern such as “IncYear”.

I organized the code in the following classes within the project namespace:

- Program: Main code for executing tests.
- ParkingCalcPageObject: Using the Page Object Model, this class defines all methods to interact with the main Parking Calculator web page.
- CalendarPageObject: This class defines all methods to interact with the Calendar pop-up window.
- SetMethods: Extended C# methods for writing the values of the various element types
- GetMethods: Extended C# methods for reading the values of the various element types
- TestProperties: Defines WebDriver, Enums, and other elements used in the test.

All details of the tests are sent to the test output using Console.WriteLine(). It may also be beneficial to only output test details in the instance of a fail, as some tests can be designed to "pass quietly". However, since this code is to be turned in and evaluated, I included a detailed output for both pass and fail cases.

To indicate a pass or fail, I used Assert.Fail() to make it clear to the user that the web page did not pass the test. In TestProperties.cs I defined passFail as a Boolean and initialized it as true in [SetUp]. As soon as one value fails, the automated test sets passFail to false, and it cannot be rewritten to true until the test has been completed. The Assert.Fail() is not executed until Teardown, so that all elements can be tested. I used this design approach because I wanted the entire test to run even if there was a failure at the beginning. In Test Cases 6 and 7, I run multiple iterations of a test. If there are 4 fails out of the 48 iterations, I would like to see them all at the same time in the test output, rather than having the test stop once the first fail happens.

Description of Test Cases

Test Case 1-3
Tests the pages according to the MINDBODY project specifications.

NOTE: All of my 5 test cases result as fails, because in each case I found a bug in the web page.

Test Case 4
Summary: *Here I am testing to see what will happen if all text entries are cleared and left blank when submitted. I set the pass/fail criteria under the assumption that the form should result in an error.*
Details:
Navigate to http://adam.goucher.ca/parkcalc/index.php
Select **Economy Parking**
Clear the time and date in the **Choose Entry Date and Time** section
Clear the time and date in the **Choose Exit Date and Time** section
Click **Calculate**
Check that Cost returns a string containing **ERROR!**

Test Case 5
Summary: *Testing for an error if the Entry Time is later than the Exit Time on the same day.*
Details:
Navigate to http://adam.goucher.ca/parkcalc/index.php
Select **Long-Term Garage Parking**
Enter **3:00** and **03/19/1991** in the **Choose Entry Date and Time** section
Select the **PM** option in the **Choose Entry Date and Time** section
Enter **3:00** and **03/19/1991** in the **Choose Exit Date and Time** section
Select **AM** option in the **Choose Exit Date and Time** section
Click **Calculate**
Check that Cost returns a string containing **ERROR!**

Test Case 6
Summary: *Validating the military time conversion that takes place in the Entry Time and Exit Time text boxes after clicking 'Calculate'. The pass/fail criteria is set under the assumption that midnight is 0:00 hours, since the radio button re-initializes to AM regardless of the input.*
Details:
Navigate to http://adam.goucher.ca/parkcalc/index.php
Select **Short-Term Parking**
Set Entry and Exit Dates to **03/19/1991**
For all hours in a 24-hour day, enter the standard time in both **Entry Time** and **Exit Time**
Click **Calculate** with each entry
Check that the military time equivalent is present and **correct** for each hour.

Test Case 7
Summary: *Testing to see if the form will handle military time as the input. The pass/fail criteria is set assuming that the military time conversion that takes place after clicking 'Calculate' will match the military time input, whether AM or PM is selected.*
Details:
Navigate to http://adam.goucher.ca/parkcalc/index.php
Select **Short-Term Parking**
Set Entry and Exit Dates to **02/01/2011**
For all hours in a 24-hour day, enter the military time in both **Entry Time** and **Exit Time**
Test all 24 hours with AM Selected
Test all 24 hours with PM Selected
Click **Calculate** with each entry
Check that the military time is present and matches the original entry for each hour.

Test Case 8
Summary: *Testing for an error if the Date value is set to a non-valid date. The pass/fail criteria is set assuming that an invalid date will result in an error.*
Details:
Navigate to http://adam.goucher.ca/parkcalc/index.php
Select **Long-Term Garage Parking**
Enter **3:00** and **hello** in the **Choose Entry Date and Time** section
Select the **AM** option in the **Choose Entry Date and Time** section
Enter **3:00** and **hello** in the **Choose Exit Date and Time** section
Select **AM** option in the **Choose Exit Date and Time** section
Click **Calculate**
Check that Cost returns a string containing **ERROR!**

Bugs Found
Test Case 4: If all fields are left blank, no error is shown. The blank times are calculated as 12:00AM, which is problematic, because if the user only enters one time and leaves the other blank, the page will still output a cost. The same is true with the date. The page selects a default date when the box is left blank. There is potential for incorrect cost calculation.

Test Case 5: If the Entry Time is later than the Exit Time, there isn't always an error. Test Case 5 tested this in Long-Term Garage parking and the cost was calculated as $0 with a duration of (-1 Days, 13 Hours, 0 Minutes). However, I manually tested this in other lots, and an error was shown.

Test Case 6: The military time equivalent is not always accurate. The issue is with 12:00AM and 12:00PM. When **Calculate** is selected, the radio buttons are reset to **AM**, and 12:00 stays at 12:00. So 12:00 AM and 12:00 PM both show up as 12:00 AM after calculation. If true military time is shown, 12:00 AM should be 0:00 AM.

Test Case 7: The parking calculator accepts some military time inputs, but not all. This test was run from the mindset of someone thinking in terms of military time and disregarding AM/PM radio buttons. With AM set the calculator works well, besides 11:00AM and 12:00AM. But if PM were selected, 13:00 hours results in 25:00 hours. This is an invalid input. To avoid confusion, the page should not allow military time inputs and give an error for any value above 12:59.

Test Case 8: The page does not result in an error for invalid date entries. This is a problem, because if someone typed the date incorrectly and included an incorrect character, the page may give an incorrect price.

Other Bugs Found:

- The Lot Drop-Down Menu resets to "Short-Term Parking" every time that **Calculate** is selected.
- When typing invalid times into the Entry and Exit Times, no error is shown.
- No error is shown when the day is beyond what exists in the month. For example, entry date 2/31/2014 and exit date 3/6/2014 are calculated as 3 days. There should be an error.