JOHN BRYCE
תלמד הי-טק. זה עובד!
a matrix company

# Java & Test Automation Course – Final Project

This project is intended for individual work and implementation by each student separately. Each student should submit his own project by the end of the course.

## Test Plan

1. Choose a website for which you will define a set of at least 10 test cases, which you will later automate, using Selenium WebDriver.
2. Before beginning to work on automation code, you need to write a test plan document with a detailed step-by-step description of the test cases to be performed on your selected web application.
3. All test cases that you define must be "interesting". An "interesting" test case is one that implements a real-life scenario / use-case / business flow, where the user performs a series of actions on the website and then we verify/validate that a certain expected event(s) happened.
   *Example*: Fill out a form and leave one of the fields empty. Verify that the expected error message is shown on the page.
4. Test cases should be written in a very detailed manner, which would allow somebody who is not familiar with the system under test to follow the test steps and execute the whole test manually. *Example:*

**Test 003 – Shopping cart counter**

| | Step | Expected Result |
|---|---|---|
| 1 | Browse to amazon.com landing page | |
| 2 | Write a random search term in the top search bar and click the search button | Transferred to search results page |
| 3 | Click the title of the first item in the list of search results | Transferred to the product details page |
| 4 | Click "Add to Cart" button | Transferred to "Customers who bought <XYZ> also bought these items…" |
| 5 | Verify the "Cart" icon on the top right of the page shows the correct number of products added to the cart. | Cart counter value should be 1 |

*Note:* each test should have a unique ID / number and a name.

## Automation

1. Create an automation project for the test cases you defined. The project should be implemented as a **Java Maven** project, with dependencies for: **Selenium WebDriver**, **TestNG** and a library for **HTML reports**. Any other library that might be needed for your project should be provided as a Maven dependency as well.
2. All tests should employ the **Page Object** design pattern.
3. Use proper Java naming conventions as learned throughout the course.
4. When mapping web elements with XPath or CSS selectors, **DON'T** use auto-generated expressions. You must calculate these expressions yourself, so they will be as concise as possible.
5. TestNG features:
    a. Organize tests into groups.
    b. At least one of the test cases should use the data provider mechanism.
    c. Create at least one TestNG test suite XML file, which allows to execute a group of tests together.
6. Tests should NOT rely on hard-coded values where configurable test parameters are more applicable. For *example*: don't use hard-coded strings for username and password values (to login to the website). Instead, such values should be read from a separate configuration file, provided in one of the following formats: properties file / CSV / XML / JSON / Excel.
7. Tests should produce a detailed, informative and organized test report that documents all of the following:
    a. Each operation performed by Selenium (such as a click)
    b. When a new test step begins, print it's description
    c. For each assertion, print whether it was successful or failed.

    Use one of following HTML report libraries: Allure, Extent, Difido.
8. If a test fails, a screenshot should be added to the generated HTML report.
9. Inside the test methods, insert comments that detail which test step is currently implemented (copy from the test plan document).
10. Projects should be hosted on GitHub. To submit the project for check – send the URL of your GitHub repository.