# Solution to Technical Assessment

**Task:** To develop a web automation framework in order to test a scenario of logging in to an e-commerce website as a user and adding a product to their wishlist.

**Application under test:** testing http://automationpractice.com/

**Solution link:** https://github.com/sbhat6/SampleWebAutomation

**Solution by:** Subrahmanya Bhat, QA Engineer

**Scenario in detail:**

1. Launch website
2. Log in as an existing user
3. Go to 'Dresses' section
4. Select a product of interest
5. On product details page, click on 'Add to wishlist' button

Note: All the test cases to meet all the acceptance criteria are not automated. This framework is just an illustration of my ability to develop web automation framework from scratch using standard practices and advanced open source tools and technologies.

**Tools used (with version):**

Selenium - 3.141.59
Java (JDK) - 8.0-151
TestNG- 6.14.3
Maven- 3.5.4
Log4J- 1.2.17
Extent Reports- 3.1.5
Eclipse IDE- Photon 4.8
Windows- 10
Google Chrome- 78.x

**Implementation:**

1. Maven project was created on Eclipse.
2. All dependencies (external jar files) were installed through pom.xml file.
3. Packages and classes were added as per Page Object Standard.
4. Webdriver instance was created separately in order to achieve code reusability.
5. Data driven approach was used to read the common variables such as browser name, driver paths, user email, user password etc from an external '.properties' file.
6. A separate class was created to read the variables from '.properties' file was written under utilities package.
7. Element repositories & corresponding methods were added to different page classes required for the scenario.
8. TestNG xml file was created containing Listener & test suite details.
9. TestNG assertions were added wherever required.
10. Java Try-Catch error handling technique were added wherever required.

11. To generate test reports, Extent Reports plugin was used. A separate TestNG Listener class created under utilities package to perform actions on test start, on test pass, on test failure etc. Listener information was added to TestNG's xml file.
12. Finally, appropriate comments have been put into the code wherever desired.

**How to run/evaluate?**

1. Do a git pull request from my repository- https://github.com/sbhat6/SampleWebAutomation
2. Import the project in your Eclipse IDE environment.
3. Go to InputData>inputdata.properties and update valid username, user email and user password.
   Note: The existing username and password are valid and the same can be used without changing anything.
   Note: Note: All the drivers (Chrome, Firefox & IE) are already included in the project and their relative paths are handled in the code. You do not need to download and update driver path.
4. Run the TestScenario1.java class as TestNG test.
5. Alternatively, run the automationstore.xml file as TestNG test.
6. Alternatively, if you have installed Maven on your PC, go to project directory on Windows and run the command "mvn clean install".
7. Verify the test result on website & Eclipse console.
8. View the test report inside Test-reports folder of the project.

**Enhancements:**
1. The rest of the test cases of the given user story can be further automated.
2. This project can be further integrated with Jenkins to run it outside Eclipse & Maven and to integrate with CI/CD.