# **Framework Details**

Repository URL: <https://github.com/instanda-technicaltest/eShopOnWeb>/Tests

## Steps to Run Tests

### Prerequisite:

1. Maven must be installed in system.
2. Assuming that my branch is merged with main branch, Clone the application repository and pull for the latest code.

Note: I couldn’t run Docker in my system due to my system‘s technical limitation. I found a workaround in the shared files, which I used to run the application.

### Steps:

1. Open Command Prompt, navigate to src/PublicAPI folder.
2. Run command: dotnet run from there.
3. Change directory to src/Web folder.
4. Run command: dotnet run –launch-profile Web
5. Change Directory to tests/Technical Test
6. Run command : mvn test

## Framework Configuration

1. Test Automation Framework – Selenium
2. Design Pattern – Page Object Model
3. Language - Java
4. Unit Testing Framework – Junit – for unit testing, TestNG – for end to end testing.
5. BDD Framework Tool - Cucumber
6. Build tool - Maven
7. Data - from excel using Apache POI jar
8. Version Control tool – GitHub

## Framework Flow

### TDD Framework flow

TestNG class with individual test methods performing unit tests sequentially. TestNG classes can be run individually as TestNG Test.

Test -> Flow -> Pages -> User Interactions -> HTML Report

Test Configurations in tests.BaseConfig file which is extended by Test classes.

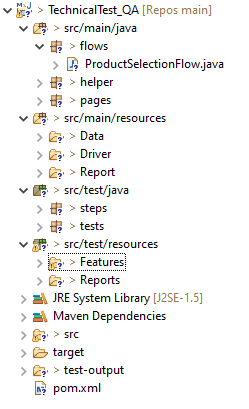
### BDD Framework flow

Cucumber Features with multiple scenarios for each unit testing. All feature files can be executed by running RunTest.Java class as Junit Test

Test ->Features -> Step Definitions -> Pages -> User Interactions -> HTML Report

Test Configurations in steps.Hooks which is extended by Step Definition classes.

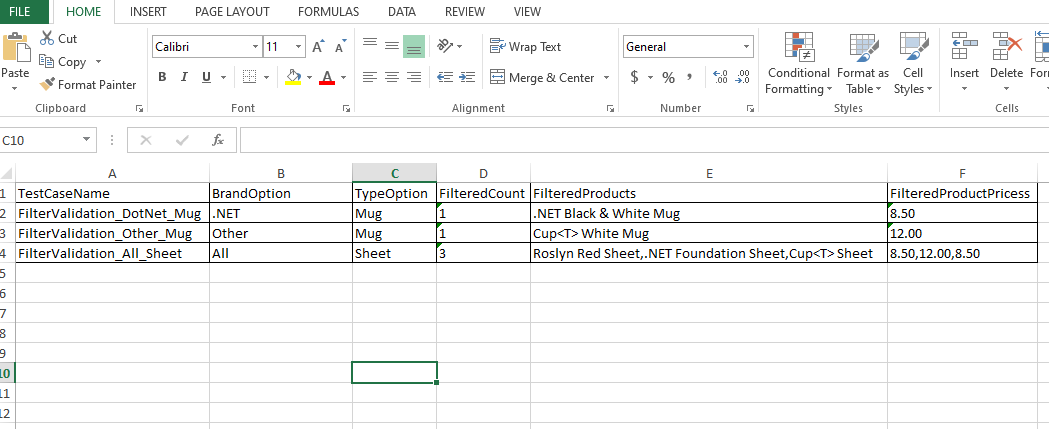
## Framework Structure:



## Framework Highlights

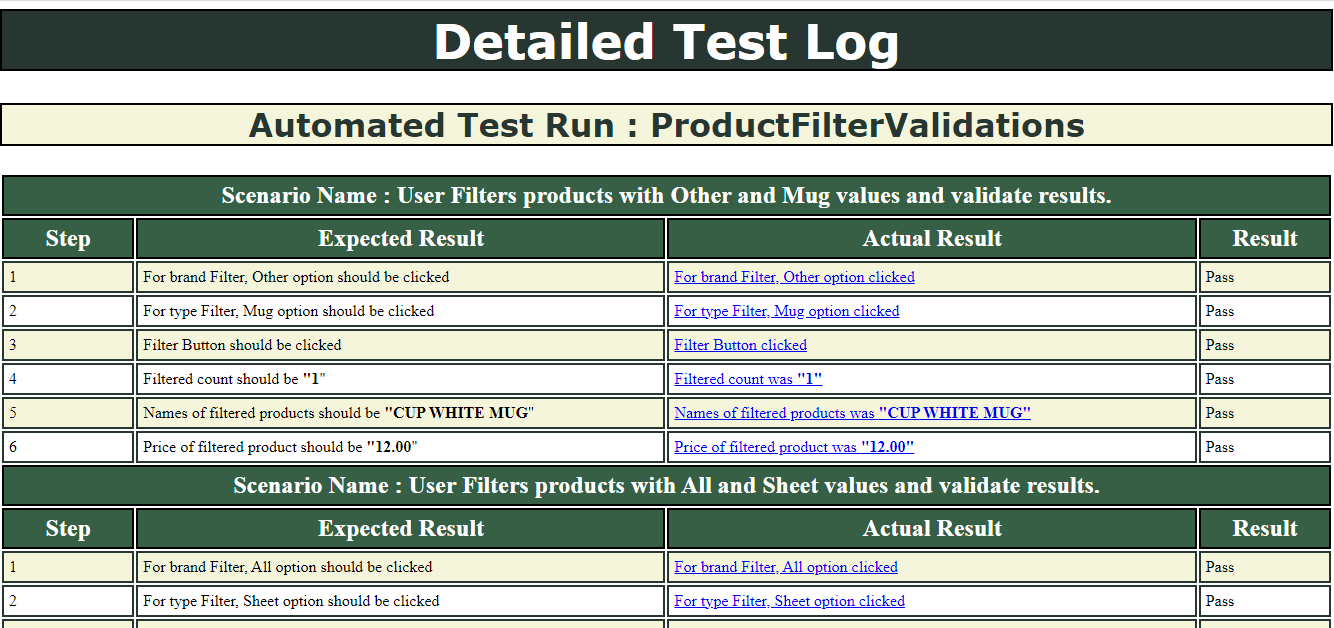
### Test Data

Test Data fetched from Excel (location: **src/main/resources/data/TestData.xls**) having rows for each scenario. For End to End scenario refer sheet Data, and for unit testing scenario refer webFeatures sheet.

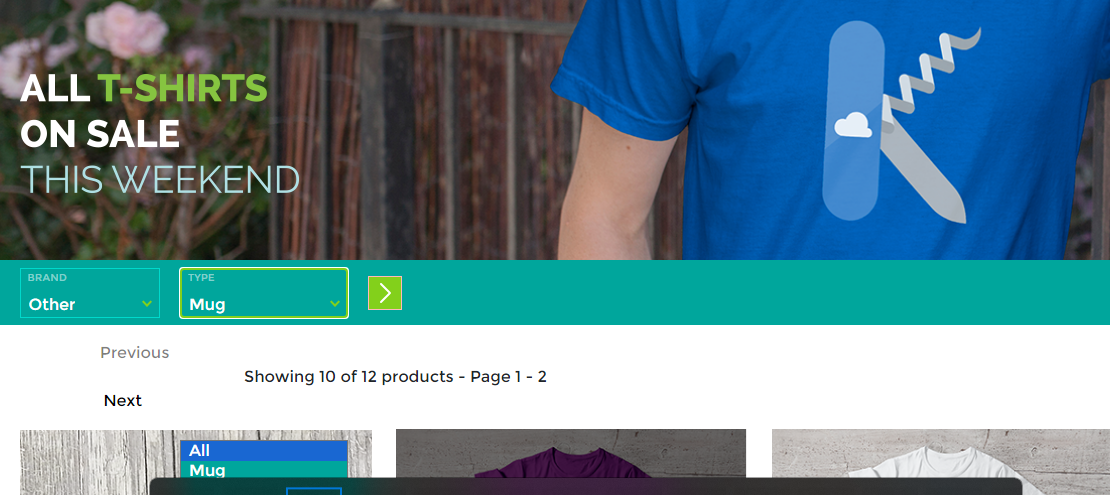


### Custom Logs and Validations

Custom Reports generation (location : **src/test/resources/reports**) provides detailed report of all the steps performed with link to screens of each respective steps. For each feature in unit testing and each TestNG class a separate log file will be created in the file created for each execution.



In Report Actual Results are hyperlink to respective screenshots.



### User Interactions

User Interactions – using custom helper class (location : src/main/java/ helper/UserInteractions) by implementing custom methods using Exception handling, explicit waits, timeouts, fail safes and reporting.

