Contents

[**Scope** 1](#_Toc148564789)

[**End to End testing** 2](#_Toc148564790)

[**1.** **Adding products to cart, login, checkout and Order Validation** 2](#_Toc148564791)

[Test Approach: 2](#_Toc148564792)

[Test Plans: 2](#_Toc148564793)

[Test Data: 3](#_Toc148564794)

[**2.** **Insert, Update and Delete Items as Admin** 3](#_Toc148564795)

[Test Approach: 3](#_Toc148564796)

[Test Plans: 4](#_Toc148564797)

[**3.** **Fetch Item Details using API and Validate in Product Page** 5](#_Toc148564798)

[Test Approach: 5](#_Toc148564799)

[Test Plans: 5](#_Toc148564800)

[**Unit Testing** 6](#_Toc148564801)

[**1.** **Validate Filter Functionality** 6](#_Toc148564802)

[Test Approach: 6](#_Toc148564803)

[Test Plan: 6](#_Toc148564804)

[Test Data: 6](#_Toc148564805)

[**2.** **Validate Login Functionality** 7](#_Toc148564806)

[Test Approach: 7](#_Toc148564807)

[Test Environment 8](#_Toc148564808)

[Revision History 8](#_Toc148564809)

# **Scope**

We are testing an e-commerce site **e-ShopOnWeb** as part of a technical assessment. Below are the type of testing conducted and Test cases selected to cover this. All major workflows like add to cart, checkout, item manipulation, login functionality are considered bases on importance.

Selenium is one of the best automation tool for testing web applications across different browsers. Selenium can automate a variety of test types including end-to-end testing, regression testing, and integration testing. It can also be integrated with CI/CD pipeline easily. Selenium supports multiple languages and I’m going ahead with Java as binding language.

We have used Rest Assured for API interactions and validations along with Selenium Java for Website validations. Rest Assured is also flexible in asserting the response as Parsing and validating XML and JSON can be done easily in REST Assured. You can integrate the REST Assured with other java libraries like TestNG and JUnit as a test framework.

Based on type of testing, we have used multiple test approaches. Here we are mainly considering End to End testing (mainly workflow validation and integration testing) and Unit testing (Functionality validation). Below are the scenarios considered for each approaches:

# **End to End testing**

## **Adding products to cart, login, checkout and Order Validation**

For an e-commerce site, adding products to cart, checkout and order validation is considered the main workflow and requirement from business perspective. There are numerous calculations involved when adding multiple products in different quantities and tracking them manually in basket, checkout page and order details section each time could be difficult and prone to human error. This is an ideal candidate for automation as this should not break and should be validated every sprint as a regression test.

Considering the complexity and length of the workflow and code quality parameter such as ease to maintain, we can utilize TDD (Test Driven) approach here. Implementing TestNG helps us to modularize the end to end workflow into smaller and meaningful test methods.

These tests are also ideal for regression testing as it cover the entire functionality and complex calculations. Using TestNG to develop them and adding them in an xml file will help to run regression in a later point of time.

### Test Approach:

* We can use a TDD Framework with TestNG and Selenium Java for E2E testing.
* Test Data can be stored in Excel. Each row representing one End to End Testing.
* In TestNG Classes, test methods could test small unit tests sequentially depending on the results of previous test methods completing an end to end scenario using DependsOnMethod annotation.

### Test Plans:

#### Workflow 1: Filter for products, Add Items to Cart, Checkout and validate order creation.

Scenario 1: Filter Products- In Product Page, products should be filtered based on Brand and Type. Product names and prices of filtered products should be validated against expected value.

Scenario 2: Add to Cart- In Product Page, desired products should be added to basket. In Basket product names and prices of selected products should be validated. Desired Quantities of each product should be selected and updated total price of each product should be validated against calculated value. Total Cart Amount should be validated against calculated value. Total Quantity in basket should be validated.

Total Product Price = Price of Product \* Quantity of Product

Total Cart Amount = SUM (Total product Prices)

Total Basket Quantity = SUM (Quantity of Products)

Scenario 3: Checkout and Login - In Basket Page, after selection of products, should click Checkout button to be redirected to login page. After provide valid credentials and logging in, User should be navigated to Review Page. Product details and basket count should be validated against expected value. Should click Pay Now button to complete checkout. Validate completion message.

Scenario 4: Order Details Validations - In My Orders page of user, validate that completed order details is present. Compare the order details with the expected value.

Test Data: Sample values to consider during testing:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No.** | **Products** | **Product Price** | **Required Quantities** | **Total Product Price** | **Total Quantity** | **Total** |
| 1  2 | Cup<T> White Mug Roslyn Red Sheet | $12.00 $ 8.50 | 2 3 | $ 24.00 $ 25.50 | 5 | 49.5 |

## **Insert, Update and Delete Items as Admin**

Another critical scenarios for testing in an e-commerce site is, the Items that are available for selection, and how they are added, updated and deleted. These actions are performed by Admin and hence valid Authorization is also required to be validated. These functionalities can be performed using Web Application and API. Here I’m prioritizing API validations as it is faster and also form the base of these functions in website. This is also an ideal candidate for automation as this should not break and should be validated every sprint as a regression test.

We need to perform API Testing of these scenarios and validate if these activities are reflecting in Web Application as well. As we are validating both API and Web Application, I have decided to use Rest Assured for API interactions and validations along with Selenium Java for Website validations. Rest Assured is also flexible in asserting the response as Parsing and validating XML and JSON can be done easily in REST Assured. You can integrate the REST Assured with other java libraries like TestNG and JUnit as a test framework.

Considering the complexity and length of the workflow and code quality parameter such as ease to maintain, we can utilize TDD (Test Driven) approach here. Implementing TestNG helps us to modularize the end to end workflow into smaller and meaningful test methods.

These tests are also ideal for regression testing as it cover the entire functionality and complex calculations and integrated testing of API and Web Application are also included. Using TestNG to develop them and adding them in an xml file will help to run regression in a later point of time.

### Test Approach:

* We can use a TDD Framework with TestNG and Selenium Java for E2E testing.
* Rest Assured can be used in combination to validate API interactions.
* Bearer Token and Item Details for validations can be fetched from API using Rest Assured.
* Test Data can be stored in Excel. Each row representing one End to End Testing.
* In TestNG Classes, test methods could test small unit tests sequentially depending on the results of previous test methods completing an end to end scenario using DependsOnMethod annotation.

### Test Plans:

Scenario 1: Perform API Authentication as Admin and Fetch the Bearer Token.

Username : [admin@microsoft.com](mailto:admin@microsoft.com), password : Pass@word1

Scenario 2: Create a new item in API by sending a POST request using bearer token from above and below details

Endpoint <https://localhost:5099/api/catalog-items>

Content type JSON,

Request Body { "catalogBrandId": 2, "catalogTypeId": 1, "description": "Test Item", "name": "Test Item timestamp ", "pictureUri": "", "pictureBase64": "", "pictureName": "", "price": 10 }

And Validate status code and details in response.

Scenario 3: Validate the newly created item in Website by logging in as Admin and navigating to Admin page.

Scenario 4: Update the new item in API by sending a PUT request using bearer token from above and below details

Endpoint <https://localhost:5099/api/catalog-items>

Content type JSON,

Request Body { "id": Itemid, "catalogBrandId": 2, "catalogTypeId": 1, "description": "Test Item", "name": "Updated Test Item timestamp", "pictureUri": "", "pictureBase64": "", "pictureName": "", "price": 10 }

And Validate status code and details in response.

Scenario 5: Fetch the updated item in API by sending a Get request using and below details

Endpoint [https://localhost:5099/api/catalog-items /](https://localhost:5099/api/catalog-items%20/) Itemid

And validate the fetched details.

Scenario 6: Validate the updated item details in Website by logging in as Admin and navigating to Admin page.

Scenario 7: Delete the updated item in API by sending a DELETE request using and below details

Endpoint [https://localhost:5099/api/catalog-items /](https://localhost:5099/api/catalog-items%20/) Itemid

And Validate status code and status in response.

## **Fetch Item Details using API and Validate in Product Page**

Validating the accuracy of Item details in product page also amount to valid scenarios for testing. The details of item can be fetched through APIs and should be compared with the details provided in Web Application. Since it is an integration testing it is also an ideal candidate for automation as manually validating it is cumbersome.

As we are validating both API and Web Application, I have decided to use Rest Assured for API interactions and validations along with Selenium Java for Website validations. Considering the data from different methods are required we can adopt a TDD method using TestNG. We can modularize the end to end workflow into smaller and meaningful test methods and use DependsOnMethod to perform these tests is order. Data fetched from one method that is required for other test methods can be stored in Public Class variables.

### Test Approach:

* We can use a TDD Framework with TestNG and Selenium Java.
* Rest Assured can be used in combination to validate API interactions.
* Test Data can be stored in Excel. Each row representing one End to End Testing.
* In TestNG Classes, test methods could test small unit tests sequentially depending on the results of previous test methods completing an end to end scenario using DependsOnMethod annotation.

### Test Plans:

Scenario 1: Fetch the item details using API by sending a Get request using and below details

Endpoint [https://localhost:5099/api/catalog-items /](https://localhost:5099/api/catalog-items%20/) ItemId

And validate the details in response and store it in a global variable Item.

Scenario 2: Fetch the Bands using API by sending a Get request using and below details

Endpoint <https://localhost:5099/api/catalog-brands>

And validate the Brand list in response. Fetch the Brand with ID BrandID of item and store it in a global variable brandName.

Scenario 3: Fetch the Types using API by sending a Get request using and below details

Endpoint <https://localhost:5099/api/catalog-types>

And validate the Type list in response. Fetch the Type with ID TypeID of item and store it in a global variable typeName.

Scenario 4: In Product Page of Website, Filter page with brandName and typeName and validate if item with ItemId is present. Validate other details of item such as price and description in comparison to the item details from API.

Test Data

|  |  |
| --- | --- |
| Item ID | 1 |
| Item Name | .NET Bot Black Sweatshirt |
| Descrition | .NET Bot Black Sweatshirt |
| Price | 19.5 |
| BrandID | 1, .Net |
| TypeID | 2, T-shirt |

# **Unit Testing**

## **Validate Filter Functionality**

After validating the major workflows we can test each Functionality in detail. One such functionality is Filter, which help us to filter products based on Brand and Type. The items can be filtered in multiple combinations to fetch different items. Validating this manually would be time consuming and difficult, making this a good candidate for automation.

For testing such specific functionality, BDD can be used, which verify that each function of the software application operates in conformance with the requirement specification. It also help us to validate multiple scenarios including negative testing. Each feature file can be considered for validating a functionality, providing multiple scenarios and negative testing.

### Test Approach:

* We can implement BDD Framework with Cucumber, JUnit and Selenium Java for unit testing.
* Scenario Outline can be used for data driven testing where same test has to be done with multiple sets of data.
* Test Data can be stored in Excel. Each row representing one Unit Testing.

### Test Plan:

Scenario 1: Filter Validation - Multiple combinations and Brand and Type should be used to filter products in In Product Page. Product names and prices of filtered products in each combination should be validated against expected values.

Test Data: Sample values to consider during testing

|  |  |  |  |
| --- | --- | --- | --- |
| Brand Option | Type Option | Products List | Product Prices |
| .NET | Mug | .NET Black & White Mug | $ 8.50 |
| Other | Mug | Cup<T> White Mug | $ 12.00 |
| All | Sheet | Roslyn Red Sheet,  .NET Foundation Sheet,  Cup<T> Sheet | $ 8.50,  $ 12.00,  $ 8.50 |

## **Validate Login Functionality**

Another functionality with multiple scenarios to consider is Login and Authentication, Were both positive and negative scenarios to be tested. This need to be validate in both API and Web Application. Positive scenario here is straightforward. But we have to validate that login is possible only if valid credentials are provided. Here for negative testing we have to consider multiple scenarios were invalid or no credentials are provided and we should validate the error message communicate enough to the user to understand what went wrong. In API testing, validate Bearer token is created for only valid credentials.

For testing such functionality, BDD can be used, which also help us to validate multiple scenarios including negative testing. Using BDD help users to understand distinction between each scenario very clearly. For such validations we can also use background keyword, to keep common steps together. Rule keyword can be used to have different backgrounds in one feature file. Here we have different background for API and Web Testing. Rest Assured can be used to perform API Interactions.

### Test Approach:

* We can implement BDD Framework with Cucumber, JUnit and Selenium Java for unit testing.
* Background keyword can be used to keep common steps together.
* Rule Keyword can be used to test scenarios of different backgrounds in same feature.
* Rest Assured can be used to test API scenarios.
* Test Data can be stored in Excel. Each row representing one Unit Testing.

#### Test Plan:

Scenario 1: User try to login without username and password. Validate if user gets email not present error message and password not present error message.

Scenario 2: User try to login without username. Validate if user gets user should get email not present error messages

Scenario 3: User try to login without password. Validate if user gets password not present error message.

Scenario 4: User try to login with invalid email ID. Validate if user gets invalid login attempt error messages.

Scenario 5: User try to login with invalid password. Validate if user gets invalid login attempt error messages

Scenario 6: User try to login with correct username and password. Validate User is logged in

|  |  |
| --- | --- |
| **loginEmail** | **loginPassword** |
| [demouser@microsoft.com](mailto:demouser@microsoft.com) | [Pass@word1](mailto:Pass@word1) |

Scenario 7: Validate Authentication using valid credentials in API by sending a POST request using and below details

Endpoint <https://localhost:5099/api/authenticate>

Content type JSON,

Request Body { "username": "demouser@microsoft.com", "password": "Pass@word1" }

And Validate status code and in response, validate bearer token is created and result is true.

Scenario 8: Validate Authentication using invalid credentials in API by sending a POST request using and below details

Endpoint <https://localhost:5099/api/authenticate>

Content type JSON,

Request Body {"username": "demouser@microsoft.com", "password": "Password1"}

And Validate status code and in response, validate bearer token is not created and result is false.

# Test Environment

For web testing: https://localhost:5001

For API testing: <http://localhost:5099/api>

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Author | Description |
| 18/10/2023 | 1.1 | Soniya Don | Updated Test Approach based on feedback, Added API Scenarios |
| 04/10/2023 | 1.0 | Soniya Don | Created Test Approach for e-shopOnWeb |