## Introduction

This document outlines the test approach, design decisions, and instructions for setting up, running, and validating the tests for the eShopOnWeb application. The primary goal is to ensure that critical functionalities of the application are covered through a combination of API and UI automated tests. This ensures fast feedback, allowing the application to continue delivering value to users.



**Test Strategy and Approach**

**Objective**

The objective is to implement an automated testing solution that validates both the API and UI layers of the eShopOnWeb application. This combination of API and UI testing ensures thorough validation, covering end-to-end user flows and critical features.

**Testing Tools**

* **UI Tests:**
  + **Selenium WebDriver**: For browser interaction and simulating user actions.
  + **TestNG**: To structure, organize, and execute test cases.
  + **Page Object Model (POM)**: To promote maintainability and scalability in the test scripts.
* **API Tests:**
  + **RestAssured**: For sending HTTP requests and validating responses.
  + **TestNG**: For test execution and reporting.



**Scope of Testing**

**UI Testing**

1. **Login Flow**: Ensure users can log in with valid credentials and are redirected correctly.
2. **Catalog Search and Filtering**: Validate that users can filter products by brand and type.
3. **Checkout Process**: Verify that users can successfully add items to the basket, proceed to checkout, and confirm their order.

**API Testing**

1. **Authentication**: Validate the login API and token generation with both valid and invalid credentials.
2. **Catalog API**: Ensure catalog endpoints return products filtered by brand and type.
3. **Item Creation**: Test the ability to add new catalog items through API calls.



**Test Design**

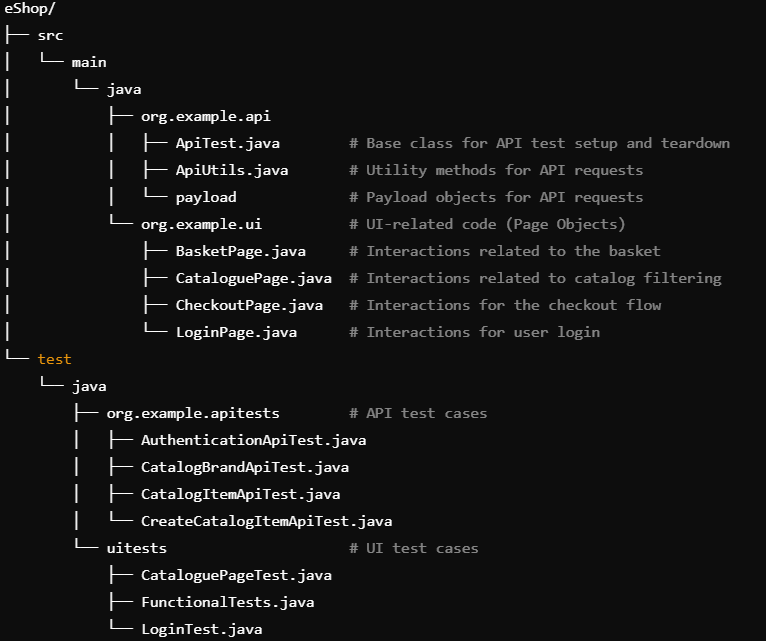
**Test Categories**

* **UI Tests**: Validate user interactions, workflows, and visual elements using Selenium and the Page Object Model (POM). This abstraction of UI elements and actions ensures maintainability and scalability.
* **API Tests**: Use RestAssured to validate the API endpoints, ensuring data integrity and response accuracy from the back-end services.



**Test Structure**

The project follows a modular and maintainable structure that separates API tests from UI tests and organizes pages, utilities, and test scripts into a clear hierarchy.



**Page Object Model (POM) for UI Tests**

POM is employed for UI tests to encapsulate each page's elements and actions in Java classes. This improves maintainability and readability by separating UI components and logic from test scripts.

* **Example**:
  + CataloguePage.java: Encapsulates catalog search filters, actions, and interactions.
  + CheckoutPage.java: Handles interactions related to the checkout flow.



**Test Groups**

The tests are grouped to allow flexibility in running specific types of tests:

* **API**: Includes all API-related tests.
* **UI**: Includes all UI-related tests.



**Setup Instructions**

1. **Prerequisites**:
   * Install Java, Maven, and any required browsers (Chrome/Firefox) for Selenium.
   * Download and configure a Selenium WebDriver for your browser.
   * Ensure the eShopOnWeb application is running locally or in a test environment.
2. **Clone the Project**:



1. **Build the Project**: Use Maven to build the project and download dependencies:



1. **Run Tests**:  
   mvn test  
   or   
   use an ide to run independent test
2. **View Reports :** View the reports in target folder created.



**Validation and Reporting**

1. **TestNG Reports**: After each test run, TestNG generates detailed reports indicating the status of each test case (pass/fail) and any related logs for failed test cases.
2. **Logs and Screenshots**: For UI tests, browser logs and screenshots are captured for any test failures to facilitate debugging.



**Conclusion**

The testing strategy and design outlined in this document aim to deliver a scalable, maintainable, and efficient automated testing solution for the eShopOnWeb application. By covering both API and UI layers, we ensure a robust test framework that delivers fast feedback and maintains the quality of the application over time.