Test Plan for E-commerce Platform Automation

1. Introduction

This test plan defines the scope, approach, resources, and schedule for the automated testing of the core functionalities of the e-commerce platform. The platform is built using Selenium WebDriver for browser automation, Cucumber for Behavior-Driven Development (BDD), and TestNG for test execution and reporting. The goal is to ensure that the e-commerce platform performs reliably in all critical user-facing functionalities.

Key Features to be Tested:

- Add Product to Cart
- Remove Product from Cart
- Increase/Decrease Product Quantity in Cart
- User Registration, Order Placement, and Checkout
- Search Product

2. Objective

The primary objective of this test plan is to validate that the core functionalities of the e-commerce platform work as expected, ensuring a seamless and reliable user experience. The automation tests will cover the functionality, user flow, and critical operations of the platform, including:

- Product cart operations (add, remove, update quantity)
- User registration and checkout
- Product search
- Order placement and confirmation

3. Test Scope

In-Scope:

- Add Product to Cart: Automating the process of adding products from the product listing to the cart.
- Remove Product from Cart: Verifying that products can be removed from the cart.
- Increase/Decrease Product Quantity: Ensuring the correct reflection of cart quantity changes and corresponding price updates.
- Search Product: Automating the product search functionality and validating search results.
- User Registration, Place Order, and Checkout: Automating the user registration process, adding products to the cart, proceeding to checkout, and successfully completing the order.

Out-of-Scope:

- Non-functional testing (e.g., performance, load testing).
- Advanced UI/UX validations (e.g., design aesthetics, animations).
- Third-party integrations (payment gateways, etc.).

4. Test Approach

Test Tools:

- Selenium WebDriver: For browser automation and interaction with the website.
- Cucumber: For defining test scenarios in Gherkin syntax (BDD).
- TestNG: For managing test execution and reporting.
- Java Faker: For generating dynamic test data.
- Maven: For managing project dependencies.
- Page Object Model (POM): To separate the test logic from the UI elements for maintainability and reusability.

Test Strategy:

- Page Object Model (POM): Implement a Page Object Model design pattern to separate the UI elements and actions for maintainable and reusable test scripts.
- BDD with Cucumber: Define test scenarios using Gherkin (Given-When-Then), which provides a high-level description of the test scenarios and enhances collaboration between technical and non-technical stakeholders.
- Test Execution: Tests will be executed in a sequential manner via TestNG, integrated with Cucumber to provide a readable report of test execution.

Test Data:

Test data will be generated dynamically using Java Faker for fields like:

- User Details (name, email, address, etc.)
- Payment Details (card number, expiry date, etc.)
- For the Add to Cart and Remove from Cart scenarios, a set of predefined product IDs will be used to simulate user interactions.

5. Test Scenarios

5.1 Add Product to Cart

- Scenario: Successfully add two products to the cart and verify their details.

- Given the user accesses the home page.
- When the user clicks the "Add to Cart" button for two products.
- Then both products should appear in the cart with correct price, quantity, and total.

5.2 Verify Product Quantity in Cart

- Scenario: Verify that increasing the product quantity reflects correctly in the cart.
  - Given the user has added a product to the cart.
  - When the user increases the product quantity in the cart.
  - Then the cart should display the updated quantity and total price.

5.3 Remove Product from Cart

- Scenario: Verify that a product can be removed from the cart.
  - Given the user has products in the cart.
  - When the user clicks the "Remove" button on a product.
  - Then the product should be removed from the cart, and the total should update accordingly.

5.4 Search Product

- Scenario: Successfully search for a product using the search bar.
  - Given the user accesses the home page.
  - When the user enters a product name in the search bar.
  - Then the matching products should be displayed.

5.5 User Registration, Place Order, Checkout

- Scenario: User registers, places an order, and completes the checkout process.
  - Given the user is not logged in.
  - When the user registers a new account, adds products to the cart, and proceeds to checkout.
  - Then the user should successfully place an order and receive an order confirmation.

6. Test Environment

- Browser: Google Chrome (latest version), Firefox (for cross-browser testing).
- Operating System: Windows 10/11.
- Test Automation Tools:
  - Selenium WebDriver (for browser automation)
  - Cucumber (for Behavior-Driven Development)

- TestNG (for test execution and reporting)
- Maven (for dependency management)
- Java Faker (for dynamic test data generation)

7. Execution Strategy

- TestNG Execution: Tests will be executed using TestNG with Cucumber integrated for step definition mapping.
- Sequential Execution: The tests will be executed in a sequential manner, ensuring that each feature is validated independently.
- Environment Setup: Test data and preconditions will be established before each test execution, including clearing the cart and resetting session cookies.
- Post-conditions: After each test, the browser will be closed to ensure isolation between tests.

8. Risk and Mitigation

Risk 1: Changes in the UI elements (IDs, class names) could break the scripts.

- Mitigation: Use robust locators (e.g., XPath, CSS selectors) and collaborate with the development team to ensure UI changes are communicated.

Risk 2: Test data inconsistency (e.g., hardcoded test data or duplicate data) causing conflicts.

- Mitigation: Use Java Faker to generate unique and dynamic test data on each run.

Risk 3: Cross-browser compatibility issues (e.g., rendering differences between Chrome and Firefox).

- Mitigation: Execute tests across multiple browsers to ensure compatibility.

Risk 4: Test failures due to network issues (e.g., slow server response during order placement).

- Mitigation: Implement retry mechanisms for flaky tests and ensure test stability by running tests during off-peak hours if necessary.

9. Defect Management

- Any defects identified during test execution will be logged and tracked in JIRA.
- Defects will be prioritized based on their impact on functionality and the user experience.
- After defect resolution, the corresponding tests will be re-executed to validate the fix.

10. Reporting and Metrics

- TestNG Reports: Will generate execution reports with detailed information on each test's execution status (Passed/Failed), time taken, and logs.

- Cucumber Reports: Provide readable, high-level results in Gherkin format, summarizing the Given-When-Then steps executed and their status.

11. Deliverables

- Automated test scripts (Cucumber feature files, Selenium WebDriver scripts, TestNG configurations).
- Test data (for both valid and invalid inputs).
- Test execution reports (TestNG and Cucumber reports).
- Defect logs and status updates.

12. Conclusion

This test plan outlines a comprehensive approach to automating the testing of an e-commerce platform. By using industry-standard frameworks such as Page Object Model (POM), BDD with Cucumber, and incorporating best practices for test execution and risk management, the plan ensures that all core functionalities are validated for reliability and performance. The test coverage will provide stakeholders with clear insights into the platform's stability and readiness for production.