Test Strategy Document for Ecommerce Website Automated Testing

1. Introduction This Test Strategy Document provides an overview of the approach, resources, and schedule for the automated testing of the Ecommerce website. The primary objective is to ensure the website's functionality, usability, and stability through systematic testing.

2. Scope of Testing The testing will focus on the following key areas:

- User Account Management: Registration, login.
- Product Search: Search functionality, filters, and search results accuracy
- Shopping Cart: Adding, updating, and removing items
- Checkout Process: Address selection, payment method, and order confirmation

3. Test Approach

- Automated Functional Testing: To validate that all website functionalities work as expected.
- Automated Regression Testing: To ensure that new changes or updates do not break existing functionalities.
- Data-Driven Testing: To test scenarios with different sets of input data for comprehensive coverage.

4. Tools and Technologies

- Programming Language: Java
- Selenium WebDriver: For browser automation
- Test Framework: Cucumber
- Build Management: Maven

5. Test Environment

- Browsers: Chrome
- Operating Systems: Windows 11
- Java Version: JDK 11

6. Test Data Management Test data will be created and maintained separately for different test scenarios. This may include user credentials, product names, and other relevant data for testing.

7. Test Deliverables

- Test Strategy Document
- Test Plan
- Automated Test Scripts
- Test Execution Reports
- Defect Reports

8. Risk Analysis Potential risks include website updates during testing, which may require updates to the test scripts. A process for monitoring and updating test scripts will be established to mitigate this risk.

9. Reporting and Metrics Test results will be documented in test execution reports, highlighting the number of tests passed, failed, and skipped, along with any defects found.

Test Plan for Ecommerce Website Automated Testing

1. Test Plan Introduction This Test Plan outlines the objectives, scope, approach, resources, and schedule for the automated testing of the Ecommerce website.

2. Test Objectives

- To ensure that all key functionalities of the Ecommerce website are working as expected.
- To identify and report any defects or issues in the website.

3. Test Scope As outlined in the above Test Strategy.

4. Test Approach As outlined in the Test Strategy.

5. Resources

- Hardware: Lenovo Laptop
- Software: Eclipse IDE Version: 2023-09 (4.29.0)

6. Entry and Exit Criteria

- Entry Criteria: Conditions that must be met before testing begins (e.g., test environment setup, test data ready)
- Exit Criteria: Conditions that must be met to conclude testing (e.g., all test cases executed, all critical defects fixed)

7. Defect Management Process for logging, tracking, and resolving defects found during testing.

8. Test Closure Activities for concluding the testing phase, including test summary report preparation and lessons learned.

User Registration:

1. **TC_User_001**: Register a New User with Valid Details
    - **Preconditions**: User is not logged in.
    - **Steps**: Go to the registration page, enter all required information with a new email address, and submit.
    - **Expected Result**: User account is created and user is logged in.
2. **TC_User_002**: Attempt Registration with an Existing Email
    - **Preconditions**: User is not logged in.
    - **Steps**: Go to the registration page, enter all required information with an existing email address, and submit.
    - **Expected Result**: Registration fails with a message that the email is already in use.

Product Search:

1. **TC_Search_001**: Search for a Product Using Valid Keywords
    - **Preconditions**: User is on the homepage.
    - **Steps**: Enter a popular product name into the search bar and initiate search.
    - **Expected Result**: List of products matching the keywords is displayed.
2. **TC_Search_002**: Search for a Non-existent Product
    - **Preconditions**: User is on the homepage.
    - **Steps**: Enter a nonsensical or very unlikely string into the search bar and initiate search.
    - **Expected Result**: Message stating no results found is displayed.

Adding Items to the Cart:

1. **TC_Cart_001**: Add a Single Item to the Cart
   - **Preconditions**: User has searched for a product.
   - **Steps**: Select a product from the search results and add it to the cart.
   - **Expected Result**: The item is added to the cart, and the cart count is updated.
2. **TC_Cart_002**: Add Multiple Quantities of an Item to the Cart
   - **Preconditions**: User has searched for a product.
   - **Steps**: Select a product, choose a quantity, and add it to the cart.
   - **Expected Result**: The correct quantity of the item is reflected in the cart.

Checkout Process:

1. **TC_Checkout_001**: Checkout with a Single Item
   - **Preconditions**: User has added an item to the cart.
   - **Steps**: Proceed to checkout, fill in or confirm the address and payment details, and complete the purchase.
   - **Expected Result**: Order is placed successfully, and confirmation is displayed.
2. **TC_Checkout_002**: Checkout with Multiple Items
   - **Preconditions**: User has added multiple items to the cart.
   - **Steps**: Proceed to checkout, fill in or confirm the address and payment details, and complete the purchase.
   - **Expected Result**: Order is placed successfully with all items included, and confirmation is displayed.

Order Management:

1. **TC_Order_001**: View Order History
   - **Preconditions**: User has placed orders in the past and is logged in.
   - **Steps**: Navigate to the 'Your Orders' section.
   - **Expected Result**: User can view all past orders with their statuses.
2. **TC_Order_002**: Cancel an Order
   - **Preconditions**: User has a recently placed order that is not yet shipped and is logged in.
   - **Steps**: Navigate to the 'Your Orders' section, select an order, and cancel it.
   - **Expected Result**: Order is canceled, and user receives a confirmation message.

Test Automation Framework: Selenium

Why Selenium? Selenium is a popular and widely-used open-source framework for automating web browsers. It supports multiple programming languages such as Java, Python, C#, Ruby, and JavaScript, allowing testers to write test scripts in the language they are most comfortable with. Selenium can automate actions in web browsers and can be used for both functional and regression testing.

Some of the reasons for choosing Selenium include:

1. **Cross-Browser Compatibility:** Selenium supports all major browsers like Chrome, Firefox, Safari, Internet Explorer, and Edge. This allows for testing across different browsers to ensure consistent behavior.
2. **Multiple Language Support:** As mentioned, Selenium supports various programming languages, providing flexibility in choosing the language that best fits the project or team's skill set.
3. **Community and Ecosystem:** Being open-source and widely used, Selenium has a large community of contributors and users. This community provides a wealth of knowledge, support, and plugins/extensions that can enhance the functionality of Selenium.
4. **Integration with Other Tools:** Selenium can be easily integrated with other tools like TestNG, JUnit for testing, and Maven, Gradle for build management. It also integrates well with continuous integration tools like Jenkins, which helps in setting up CI/CD pipelines.
5. **Grid for Parallel Execution:** Selenium Grid allows for running tests in parallel across different machines and browsers, which can significantly reduce the time required for executing test suites.

Framework Architecture: Selenium provides several components to support web browser automation:

1. **Selenium WebDriver:** The core component that provides APIs for browser interactions. It directly communicates with the browser at the OS level, offering control over browser sessions, DOM interactions, and user interactions with web elements.
2. **Selenium IDE:** An integrated development environment for Selenium scripts. It is a browser extension that provides a record-and-playback tool for authoring tests without needing to learn a test scripting language.

3. **Selenium Grid:** A tool used to run tests in parallel across different machines and browsers. It helps in scaling the test execution environment by distributing test commands to multiple instances of WebDriver.

4. **Selenium Remote Control (RC):** An older component of Selenium that is now deprecated in favor of WebDriver. It was used to control browsers for testing purposes.

A typical Selenium WebDriver setup involves writing test scripts using the WebDriver APIs to interact with web elements. These scripts are then executed in the desired browser(s) using WebDriver's browser-specific drivers (e.g., ChromeDriver for Google Chrome, GeckoDriver for Firefox). When using Selenium Grid, these tests can be executed in parallel on different machines and browsers, controlled by a central hub that distributes the tests to the appropriate nodes.

Overall, Selenium provides a flexible and powerful platform for automated testing of web applications, with support for various languages, browsers, and integrations.