**Test Plan for E-commerce Automation with Playwright & Cucumber**

**1. Introduction**

The purpose of this test plan is to outline the automated testing strategy for the e-commerce website, https://ecommerce-playground.lambdatest.io/. The automation will be implemented using **Playwright** for browser automation and **Cucumber** for behavior-driven development (BDD). The test suite will cover key user flows such as registration, login, product search, and checkout.

**2. Objectives**

* Ensure the e-commerce platform's main functionality works correctly across browsers.
* Automate critical user flows like registration, login, product search, cart functionality, and checkout process.
* Provide comprehensive reporting and logs for each test run.
* Detect any regressions or failures in the application when new changes are made.

**3. Scope**

* **Automated Functional Testing**: Test core functionalities such as login, registration, product search, adding items to the cart, and completing checkout.
* **Cross-Browser Testing**: Tests will run on Chromium, Firefox, and Webkit using Playwright.
* **Data-Driven Testing**: Use test data from JSON files for user registrations and product details.
* **Report Generation**: Generate HTML reports using Cucumber to visualize the results of test executions.

**4. Test Environment**

* **Browser**: Chromium, Firefox, Webkit
* **Operating System**: Windows, macOS, and Linux (Cross-platform compatibility)
* **Test Framework**: Playwright
* **BDD Framework**: Cucumber (with Gherkin syntax)
* **Programming Language**: JavaScript
* **Reporting Tool**: Cucumber Reports (HTML format)

**5. Test Scope**

**Modules to be Tested**

1. **User Registration**
   * Verify that the user can successfully register with valid details.
   * Verify validation error messages when incorrect details are entered (e.g., invalid email, weak password).
2. **User Login**
   * Verify that a user can log in with valid credentials.
   * Verify the system shows an error message when login is attempted with incorrect credentials.
3. **Product Search**
   * Verify that users can search for products by name or category.
   * Verify the correct results are displayed when a search is conducted.
4. **Product Details & Add to Cart**
   * Verify that the user can view product details (e.g., price, description).
   * Verify that users can add products to the cart.
5. **Cart and Checkout**
   * Verify that items can be added to the cart and quantities can be updated.
   * Verify that the user can proceed to checkout and complete a purchase (optional for testing).
   * Verify order summary details are correct at the checkout.

**Test Scenarios**

1. **Registration**:
   * Scenario: User registers with valid details.
   * Scenario: User tries to register with an invalid email address.
   * Scenario: User registers with a weak password.
2. **Login**:
   * Scenario: User logs in with valid credentials.
   * Scenario: User logs in with invalid credentials (wrong username/password).
3. **Product Search**:
   * Scenario: User searches for a product by name.
   * Scenario: User filters products by category and verifies the results.
4. **Add to Cart**:
   * Scenario: User adds a product to the cart.
   * Scenario: User updates the quantity of a product in the cart.
5. **Checkout**:
   * Scenario: User proceeds to checkout with the items in the cart.
   * Scenario: User enters billing details and completes the purchase.

**6. Testing Approach**

* **Playwright** will be used for browser automation and interaction with the website.
* **Cucumber** with **Gherkin syntax** will be used to define test scenarios.
* **Step Definitions**: Each step in the Gherkin scenarios will be mapped to Playwright actions (e.g., filling out forms, clicking buttons).
* **Test Data**: The tests will use data from JSON files for user credentials, product names, and other dynamic inputs.
* **Parallel Execution**: The tests will be run in parallel on different browsers (Chromium, Firefox, WebKit) to ensure cross-browser compatibility.

**7. Test Deliverables**

* **Test Report**: After each test run, an HTML report will be generated showing the test results (pass/fail status) for each scenario.
* **Logs**: Detailed logs of each action and step will be provided in case of failures.
* **Test Artifacts**: Feature files, step definitions, and test data will be part of the project repository.

**8. Test Schedule**

* **Phase 1**: Setup and Environment Configuration (1 day)
* **Phase 2**: Test Case Development and Initial Testing (3 days)
* **Phase 3**: Cross-browser Testing and Report Generation (2 days)
* **Phase 4**: Final Reporting and Documentation (1 day)

**9. Risk & Mitigation**

* **Risk**: Changes in the website UI might break locators used in tests.
  + **Mitigation**: Use dynamic locators based on class names, ids, or text, and keep the locator strategy flexible.
* **Risk**: Tests may fail due to network issues or slow server response time.
  + **Mitigation**: Implement retries for flaky tests and monitor network performance.

**10. Tools and Technologies**

* **Playwright**: For browser automation.
* **Cucumber**: For BDD-style test scenarios.
* **Node.js**: For running JavaScript-based test scripts.
* **Mocha**: As the test runner for executing the Cucumber steps.
* **Chai**: Assertion library for test validations.

**11. Test Execution Strategy**

The tests will be executed using **Cucumber.js**, which integrates seamlessly with **Playwright** for browser automation. The feature files will be written in Gherkin syntax and executed with the following command:

npx cucumber-js

Tests will be executed in a headless mode (no visible browser) for efficiency, but can be run in non-headless mode for debugging purposes.

**12. Conclusion**

This test plan defines the approach for testing the core functionalities of the e-commerce platform using Playwright and Cucumber. The goal is to automate the testing of key user flows to ensure a smooth user experience, while also verifying the platform's stability and responsiveness across different browsers.