**1. Introduction**

* **Purpose:**

The main objective of the test strategy is to identify, categorize and prioritize the functionalities that need to be explored and tested in the <https://ecommerce-playground.lambdatest.io/> webstore.

By understanding the main purpose of the website, a successful and robust point of reference can be created. Following the steps described in this document, an acceptable balance of user satisfaction, legislative compliance and owner requirements can be met, thus ensuring a competitive product.

* **Scope:**

A set of main functionalities consisting of but not limited to is currently determined to be as follows: payment processing, search functionalities, order confirmations, cart usability, product catalog functionality, account management and security and general navigation of the website.

**2. Test Approach**

* **Testing Types:**

To cover the important functionalities, several testing types are to be used:

* + An initial exploratory testing session

A quick look around the site by the tester, to determine the natural flow for a client, helps with prioritization of subsequent testing.

* + Functional testing

Testing the outcome of user interaction with the selected modules. Ensuring an uninterrupted shopping experience on the platform.

* + End-to-end

Automating the most frequent flows for easy smoke and regression testing after subsequent updates or feature changes.

* **Test Techniques:**
  + State transition: applicable for the shopping cart, products on the page. Used to verify consistency when the products are added to the cart, filtered, or otherwise manipulated by the user.
  + BVA: Applied to fields expected to be filled out by the user, in checkout, search, login and register forms.
  + Equivalence partitioning: Further verifying data validity where needed.

**3. Test Environment**

* **Tools:** Considering the size of the task, a basic framework and POMs are to be created and utilized in the testing. Selenium WebDriver in conjunction with Java is to be used for the creation of the automated tests.

**4. Risks and Contingencies**

* **Potential risks:**
  + False positives or negatives
  + Incomplete coverage
  + Maintenance of tests
    - Creating a modular framework from the beginning of the automation process with robust page models, allowing for easy and quick remodeling or upgrading of tests can be used to mitigate the risks of poorly performing tests.
    - Well-established object models can help with the addition of tests where coverage is found to be lacking.
    - By combining the contingencies in the points above, a quality maintenance workflow can be established within a brief time.
  + Lack of expertise
  + Resource restraints
    - These can be avoided by utilizing proper code reviews by higher staff and assisting and guiding the tester/s adequately and in time.

**7. Metrics and Reporting**

* **Metrics:**
  + Pass/Fail rate
  + Execution time
* **Reporting:**
  + Test execution reports can be generated with every run and compiled into a final report
  + UX report can be created following the exploratory testing, highlighting issues with the usability of the store
  + Test coverage report created after finalizing the test suites

**8. Sign-off Criteria**

* The current test strategy can be accepted as completed after management has approved the test suites.
* The main functionalities identified above are covered sufficiently.
* The automated tests are created and are available for implementation in the development pipeline.
* There are no major defects affecting the usability of the product or the successful running of the tests