OpenCart Application Test Strategy

PREPARED FOR

Pramod Dutta

TheTestingAcademy

PREPARED BY

Fernando Garzon

My Portfolio

Summary

Introduction

This document outlines the test strategy for the OpenCart web application. The objective is to ensure that this eCommerce application functions currently, processes user data in a secure and accurate way, and meets usability standards.

Objectives

The testing process aims to deliver a fully functional eCommerce application that fulfils the required standards and exceeds performance expectations. Our primary goals are to identify and resolve defects, ensure data validation, and confirm that the payment process is being done in a secure way.

Test Levels

Testing will be conducted at multiple levels:

- **Unit Testing**: Focused on individual features to ensure they function correctly in isolation.
- **Integration Testing**: Ensures that the pages work together as expected and the data submission process is seamless.

Test Types

The following test types will be conducted:

- **Functional Testing**: Validates that the application functions according to the specified requirements and the system delivers what is expected.
- **Regression Testing**: Ensures that recent changes have not adversely affected existing functionality.
- **Smoke Testing**: Conducted to verify that the basic functionalities of the form are working as expected after each new build.

Test Techniques

We will utilise automated black-box testing techniques. This will allow us to focus on user perspective application behaviour without considering the internal code structure.

Test Deliverables

The following deliverables will be produced during the testing process:

- **Test Plans**: Outline the scope, approach, resources, and schedule of testing activities.
- **Test Cases**: Detailed steps to verify specific functionalities of the application..
- **Test Automated Scripts**: Scripts developed for automated testing using the Robot Framework and Selenium library.
- **Test Reports**: Summarise the results of testing activities, including any identified defects and their status.

Test Environment

The testing environment will consist of:

- A computer with an updated operating system, connected to the internet with decent bandwidth.
- Browsers to be tested include Chrome, Firefox, and Edge to guarantee compatibility.
- Automation scripts will be written in Python using the Robot Framework and Selenium library.
- Pabot library will be used to run parallel tests.
- Jenkins servers will be used to control the test builds and generate the reports.

Test Schedule

A detailed test schedule will be developed, giving the timeline for each phase of the testing process. The schedule will include timeframes for each test type.

Test Scenarios

The tests scenarios planned are the following:

1. **Store Page Functions**:
    a. Users can access different category pages and visualise its contents.
    b. Users can compare two or more products using the Product Comparison feature.
    c. Users can add items to their shopping cart.
2. **LogIn and Registration**:
    a. **Registration Page**:
        i. Users can register new accounts.
        ii. Page only accepts valid data.
        iii. Privacy Policy is shown.
    b. **LogIn Page**:
        i. Users can use their newly created account to LogIn.
        ii. Log In Page only accepts valid registered account credentials.
3. **Cart and Checkout**:
    a. **Cart**:
        i. Users can access the floating Cart Preview from the Store Page.
        ii. Can remove items using the floating Cart Preview.
        iii. Can access the full Cart Page.
        iv. Can change item quantity and price updates automatically.
    b. **Checkout**:
        i. LogIn is requested for not logged in users.
        ii. Accepts only valid payment information.
        iii. Order is confirmed.

Risk and Mitigation

Potential risks and their mitigation strategies include:

- **Compatibility Issues**: Testing across multiple browsers (Chrome, Firefox, Edge) to ensure consistent behaviour.
- **Data Validation Issues**: Implementing thorough validation checks to ensure data accuracy.
- **Automation Script Failures**: Continuous integration with Jenkins to detect and address failures early.

Conclusion

This test strategy aims to ensure that the application is robust, reliable, and ready for deployment. By following this strategy, we aim to identify and resolve defects early, maintain high quality, and meet user expectations.