Spree Ecommerce Test Strategy (Agile)

Table of Contents

1. Introduction 3

1.1 Overview 3

2. Objectives of Test Strategy 3

3. Scope and Limitations 4

3.1 Scope 4

3.2 Limitations 4

4. Test Approach 4

4.1 Scope 4

4.2 Out of Scope 4

4.3 Manual test process 4

4.4 Test Types 4

4.4.1 Unit Testing 4

4.4.2 API/Service Testing 5

4.4.3 Integration Testing 5

4.4.4 Smoke/Acceptance Testing 5

4.4.5 Functional Testing 5

4.4.6 UI Testing 5

4.4.7 Regression Testing 5

4.4.8 Cross Browser/Cross Device Testing 5

4.4.8 Usability Testing 5

4.4.9 Exploratory Testing 5

4.4.10 Accessibility Testing 6

4.4.11 Localizability Testing 6

4.4.12 Compliance Testing 6

4.4.13 Security Testing 6

4.4.14 Performance Testing 6

4.4.15 User Acceptance Testing 6

4.5 Test Levels 6

4.6 Automation Test Process 6

5. Test Environment 6

6. Defect Life Cycle: 7

7. Resource: 7

8. Automation Tools 7

9. Assumptions 8

10. Control Procedure 8

11. Deliverables 8

12. Risks and mitigation plans 8

• Introduction

1.1 Overview

Spree is an e-commerce application developed with Spree Ecommerce opensource framework. Spree helps user to do shopping. User can search for products, add to cart and buy the products. User can create an account by registering in the application. Spree Ecommerce also has plans to enter new markets/regions and build omni-channel capability.

• Objectives of Test Strategy

The objectives of the Agile test strategy are to do the following:

• Provide a framework for testing within the development lifecycle so that the effort stays focused and on schedule.

• Identify the tasks necessary to prepare for and to conduct release-level manual and automated tests.

• Provide recommendations on usage of appropriate tools, test approach, automation framework, defect management process.

• Scope and Limitations

3.1 Scope

Includes manual and automation testing of backlog sprints till current sprint.

3.2 Limitations

Automation of third party tools like payment gateways cannot be automated.

• Test Approach

4.1 Scope

• Test the user stories at hand

• Regress the previous stories and features

• Ensure environment coverage

4.2 Out of Scope

Third party payment tools used in application cannot be automated.

4.3 Manual test process

• Manual test cases will be drafted for each user story within the sprint

• Testcase will be reviewed before they are executed

• Any defects found during testing will be raised and assigned to developer

• Test Report will be sent to all the stake holders along with defects found during the testing phase at the end of each sprint

4.4 Test Types

4.4.1 Unit Testing

Automated tests on code level which run every night or after new code is added. Unit tests should be written for both functional and UI classes. Unit tests should also test boundary values and edge cases. Unit tests increase the safety net of the application. Test Driven Development(TDD) approach is used. Code coverage tools can be used to validate the effectiveness. This will be performed by the developer. However QA could be part of unit testing as well. Bugs found are fixed immediately.

4.4.2 API/Service Testing

Perform API testing or service testing to test code earlier in the cycle instead of relying only on “black box” testing. API testing is done to ensure communication between components are working.

4.4.3 Integration Testing

Testing interactions between different modules of the system. Integration testing will be formed on different modules of the application in each sprint. Any defects found during the testing will be raised and assigned to the respective developer.

4.4.4 Smoke/Acceptance Testing

Smoke/Acceptance test suite contains only high-level functionality to make sure the application is stable enough for further development or testing.

4.4.5 Functional Testing

Functional testing of the application is performed in each sprint and defects will be raised. Defects raised will be assigned to respective developer who has developed the application feature/module.

4.4.6 UI Testing

Testing done to verify the UI of different pages in the application. Defects raised will be assigned to respective developer who has developed the application feature/module.

4.4.7 Regression Testing

Tests to detect side effects from changes to the system. Regression testing will be performed after each sprint and defect will be reported.

4.4.8 Cross Browser/Cross Device Testing

Testing needs to be done across different supported browsers and supported devices(Mobile devices, tablets etc…). Cross browser testing should cover the main platforms like Linux, Windows, Mac etc..

4.4.8 Usability Testing

Testing of the application to ensure that the intended users of a system can carry out their tasks efficiently, effectively, and satisfactorily.

4.4.9 Exploratory Testing

Exploratory testing will be performed keeping in mind the user actions.

4.4.10 Accessibility Testing

Testing that will be done to ensure that the product/component is compliant with accessibility standards. Testing will complete with the submission of a compliance report.

4.4.11 Localizability Testing

Localizability consists of internationalization (i18n) and localization (l10n) testing. Once the content is translated, it is important to do content verification to ensure material is properly translated as per the language and culture.

4.4.12 Compliance Testing

Since this product involves handling financial data and personal data, testing for compliance with PCI and GDPR need to be performed. Chosen 3rd party payment providers shall adhere to PCI compliance as well.

4.4.13 Security Testing

Security Tests should check for basic security vulnerabilities derived from OWASP.

4.4.14 Performance Testing

Performance tests should cover tests to test response time performance. Load testing should be done. Scalability analysis should be done.

4.4.15 User Acceptance Testing

User Acceptance Tests are tests to confirm the built product is what was expected and that it meets user’s expectations.

4.5 Test Levels

We follow the **Test Pyramid** model. We should focus on producing many more low-level unit tests than high-level UI-based automated and manual tests, including exploratory and usability testing.

4.6 Automation Test Process

• Test cases will be identified in each sprint which can be automated and the same will be automated

• Regression suite covering the critical functionalities of the application from Sprint-1 to Sprint-N will be identified

• Regression suite identified in step-2 will be automated and test scripts will be added after each sprint

• Regression suite will be executed in each sprint to check the health of the application

• Automated scripts will be executed as part of Continuous Integration process

• Test Environment

**TYPES OF TESTING**

**AUTOMATED/MANUAL**

**ENVIRONMENT**

**EXECUTED BY**

Unit Tests

Automated

Development

CI

API/Service Tests

Both

Development/QA env

QA

Integration Tests

Automated

Development

CI

Smoke/Acceptance Tests

Automated

All env

CI

Functional Tests

Both

QA env

Manual - QA,  
Automation - CI

UI Tests

Both

QA env

Manual - QA,  
Automation - CI

Regression Tests

Both

QA/Regression env

Manual - QA,  
Automation - CI

Cross Browser/Cross Device Tests

Both

QA/Regression env

Manual - QA,  
Automation - CI

Usability Tests

Manual

UAT Env

QA/BA + Business Users

Exploratory Tests

Manual

QA env

QA

Accessibility Tests

Both

QA env

QA

Localizability Tests

Both

QA env

QA

Compliance Tests

Both

QA env

QA

Security Tests

Both

Dev env, QA env, Pre-prod/UAT env

QA

Performance Tests

Both

Dev env, QA env, Pre-prod/UAT env

Dev, QA

User Acceptance Tests

Manual

UAT Env

Business users

• Defect Life Cycle:

All the Defects/Suggestions will be raised in Jira. Report can be pulled out anytime to see the status of any defect. Defect report will be broadcasted to all the stake holders.

• Resource:

Four QA resources will be available to test the project.

• Automation Tools

**Intellij** open source development tool will be used for developing the scripts. Scripts will be written in **Java** using **Selenium Webdriver**. **TestNG** will be used as reporting tool. **Junit** will be used as a unit test framework. **Postman** & **RestAssured** will be used for API testing. **Git** will be used for configuration management

• Assumptions

Manual test cases will be used as reference while automating the features.

• Control Procedure

• Automated features will be delivered sprint wise

• Every item discussed on call will be documented

• Deadline of testing might change based on complexity of the feature being tested

• Test Documents – Test Plan, Test Strategy, Test Case Document, Automated Scripts, reports etc. will be uploaded in GitHub.

• QA team will be responsible for managing the test documents in GitHub.

• Deliverables

• Manual TCD

• Automated Scripts.

• Test execution report.

• Manual TCD

• Automated Scripts.

• Test execution report.

• Risks and mitigation plans

Risk

Mitigation plan

Environment unavailability

Use containerized deployment (such as docker) to enable quick spin up of new environments

Resource unavailability

Weekly meetings to assess resource capacity vs requirements