**Table of Contents**

* [Purpose And Objective](#2bk92fy8xx68)
* [Test Principle](#rh898v4r7ynq)
* [Test Strategy](#wc9kup3vtyx0) 
  + [Test Methodology](#jub3pf88gjhp)
  + [User Story/ Sprint planning](#cr7474e96ye9)
  + [Acceptance Criteria](#5pwk1s1d5b27)
* [Text Execution Using Test Pyramid](#3q7eku41kg53)
* [Test Coverage](#mum2rsxzf14)
  + [Types of Testing](#mum2rsxzf14)
  + [Specific Tests for Omni Channel Capabilities](#d7s8ypktragk)
  + [Test Approach for Different Markets/ Regions](#imk2i4khb5m4)
  + [Test quadrants](#hx8y9y5c3jnl)
  + [Test Execution flow in TDD](#yv9ury5k1esh)
* [Test Environments](#2ibmemz1ku0d)
* [Tools/ Framework for Testing](#hxbwjk3r7akn)
* [Test Management](#8w29gnez53im)
* [Issue/Defect Management](#7vi1h85wsto9)
* [Scope , Timelines and Milestones](#hrxxmljt04xe)
  + [Resource Allocation](#yvh1ofxd99g4)
* [Risks and Mitigation Plans](#lyoqkfe185t7)
* [Test Coverage Analysis](#qg5qcqp2nosg)
* [Key Metrics and Reporting](#k4712ygebqdm)
* [Done Criteria](#qdipiu169gl)

**Purpose And Objective**

* This document covers the Test Strategy we will be following to test any project developed using the Spree Ecommerce App.
* The purpose is to deliver a bug free , fully functional project / Application which is developed using the Spree Ecommerce platform.
* Once the test strategy is in place, the project can be expanded with limited scope ( timeframe) and delivered with high quality at a low maintenance cost.
* Solutions/ test approach for Omni channel capability is detailed in the test strategy.
* Also to quickly expand the scope to different markets the test approach is highlighted.

**Test Principle**

The mission of the testing across all stages / phases for an Ecommerce application is based on the following principles:

* Plan early to facilitate starting tests on time and staying on schedule
* Test early. It is far less costly to fix errors earlier in the systems development life cycle than later
* Establish entry and exit criteria to ensure that the objectives of each testing stage / phase are met
* Define and document separate test suites for testing the application in different channels(ex: mobile browser, mobile Apps, IOS and android applications)
* A common base test to be used to test across all regions and markets.
* Separate test suites to be created for already existing markets ( ex: US , CA , UK etc) which can be easily extensible when the app is extended to other markets in future.
* This can include both manual and automation tests depending on the feasiblitly.

**Test Strategy**

The Test Strategy provides a comprehensive documentation reference guide to be used throughout the testing effort for any Spree Ecommerce Application.

**Test Methodology**:

**Agile** : The test methodology that we are proposing to follow is Agile.

**Scrum**: Scrum is an Agile methodology in which the Project is divided into User Stories and Each Story is worked upon in a Sprint.

A Sprint will be typically 3 or 2 weeks in scope.

In Agile, QA is the responsibility of everyone, not only the testers. QA is all the activities we do to ensure correct quality during the development of new products.

This means, the entire project is divided into a set of user stories and then each story is worked upon by the project team.

**User Story / Sprint Planning:**

* In each story workshop, everyone in the team learns about the details of the stories so developers and QA know the scope of the work.
* Everybody should have the same understanding of what the story is about.
* Developers should have a good understanding of the technical details that are involved in delivering the story,
* QA should know how the story will be tested and if there are any impediments to test the stories.
* In the sprint planning meetings, the estimates given for a story should include the testing effort as well and not just coding effort. QA (manual and automation) must also be present in the sprint planning meetings to provide an estimate for testing of the story.
* Each story will have an acceptance criteria.

Example user stories for a Spree Ecommerce application,

* Develop a login page using the spree commerce app, which has the facility to both sign in and register.
* A home page with categories to select different products in each categories.

User Stories to be created separately for ,

1. Testing for cross browser and cross device testing ( QA Sprint needs to be planned and thought out thoroughly on what needs to be covered in these tests)
2. When an App is expanded to a particular region, we need to be clear on what user stories need to be picked up and tested upon to release it faster and defect free.
3. Cross border trading can be thought of as a user Story and specific Acceptance criteria can be fixed for those stories.

Assumptions and deliverables in each stories should be pointed out clearly.

**Acceptance Criteria:**

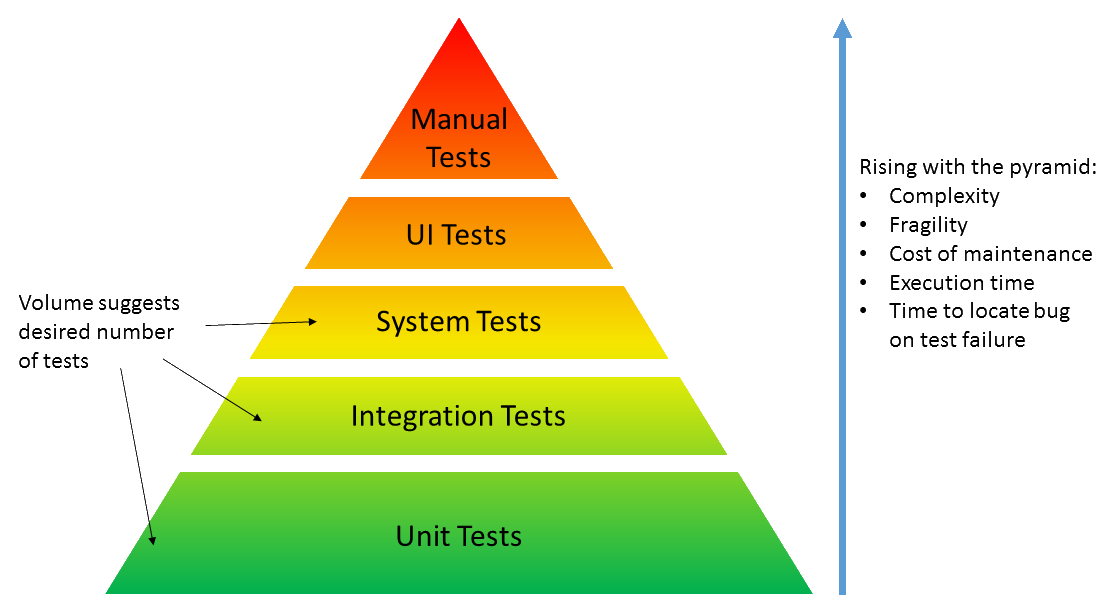
* Each of the User stories must contain acceptance criteria. This is possibly the most important element which encourages communication with different members of the team.
* Acceptance criteria should be written at the same time the user story is created and should be embedded within the body of the story. All acceptance criteria should be testable.
* Each Acceptance Criteria should have a number of Acceptance Tests presented as scenarios.

**Text Execution Using Test Pyramid**

Each Story in the Spree Ecommerce Application will follow what is called a “ Test Pyramid “ for testing the entire Story/ feature and deliver quick and quality product.

The approach to be followed here is TDD - Test Driven Development , where in the tests are developed even before the Development of code , so the safety net is already built.

This helps in quick finding and closure of defects early in the Story lifecycle.

****

**Test** Coverage

**Types of Tests to be carried out for a Spree Ecommerce Application:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Type | Test Level | Who | What | When | Where | How |
| Functional Testing | Unit Testing | Dev | All new code + re-factoring of legacy code as well as Javascript unit Testing | As soon as the code is developed | Dev Box | Checked in to CI /CD pipeline as well  Automated using Unit, PHPUnit etc |
|  | API/Service Testing | Dev, QA | All new APIs involved in the project to be tested individually | Parallel automation of APIs can be done during API development cycle | QA | Checked into CI/CD pipeline  Automated using TestNG , REST Assured frameworks, Manual testing using Postman for tests that cannot be automated |
|  | Functional UI Testing | QA | UI tests on the Application | Parallel Automation of UI pages during Development | QA | Automation using Selenium.  Manual testing of the flows that cannot be / need not be automated. |
|  | Integration Testing | QA | All the components are integrated and tested together end to end | Once the components are ready and deployed in an integrated environment | Integrated Environment ( staging) | Mostly integrated tests are carried out manually. A minimal set of integrated tests can be automated and checked into CI pipeline |
| Acceptance Testing | Acceptance criteria Test | QA, BA, Dev | The acceptance criteria tests for each story is tested | During Dev to QA handover | Dev box , QA Environment | Mostly Manual as it covers a minimal set of tests for a single user story |
| Regression Testing | Full Regression test | QA | Scenario Testing, User flows and typical User Journeys | After the functional tests are certified and story or project moves to Integration environment | Integration Environment | The pack should have a set of automated User flow tests that can be run parallel and easy to maintain. |
|  | Smoke Test | QA | Quick check of main functionalities ( Typically P1s and P2s) | After each build (once the builds are stable and there are checkins for bug fixes) | Integration Environment | A small set of Smoke test to be automated and run through CICD pipeline. Should not take long time to give feedback |
| UAT (User Acceptance Tests) | Exploratory Testing | QA, BA, PO ( Can conduct bug bash with different Users) | Exploratory testing should focus on user scenarios and should find bugs that automation misses. | Once regression starts , the UAT can also start in parallel or the Team may start it once regression completes | Integration Environment | Manual testing of the user flow and different actions an end customer can perform.  The aim of UAT is to ensure that the developed features make business sense and helpful to customers.This should also cover scenarios for different markets ( localisation and globalization ) |
| Security Testing | Integration Testing | Security Engineer / QA | Security Tests should check for basic security vulnerabilities. | In each stage ( QA, Integration and UAT ) | QA and Integration Environment | Few Security standards are tested across all stages (UI, API and integration). Can be covered in Unit tests and Service Tests in automation |
| Performance Testing |  | Performance Engineer / QA | Performance Tests should check performance metrics on each deploy to ensure no performance degradation | Once the story is in QA, and is stable in QA environment | QA and Integration Environment | Tools like JMeter can be used to measure the performance parameters |
| Cross Browser Testing | UI Testing | QA | UI responsiveness in different Browsers and OS.This will include testing across multiple operating systems on both Android and iOS platforms | Once the story is in QA, and is stable in QA environment | QA and Integration environment | High level functional test cases are executed on different browsers having latest version –  Chrome, Firefox, Safari & Edge/IE.  Application as a whole is continually validated from usability point of view. |
| Cross Device Testing | UI Testing | QA | UI responsiveness and load in different devices like mobile devices, mobile apps etc | Once the story is in QA, and is stable in QA environment | In QA and Integration Environment | Manual testing of few high level User flows in all the mobile apps and devices.Cross device testing would be performed across different devices – desktops, Tablets & mobile phones. |
| Accessibility testing | UI Testing | QA | Are the pages / UI accessibility friendly | In QA | Integration Environment | Accessibility parameters to be tested in the web pages manually |

Specific Tests for Omni Channel Capabilities:

For Omni Channel capabilities, the tests to be done are,

1. Cross Browser testing - Cover all the Tier 1 browsers and OS combinations. This can be achieved using a small set of Functional UI tests run in parallel across different browsers and OS , using a GRID ( Selenium Grid).
2. Cross Device Testing - Testing across devices including desktop, laptop, mobile and tablets with OS such as Windows 10, IOS and Android Apps.

Most companies have a Mobile Software Lab with most of the top tier Devices where the UI can be tested using few regression tests manually.

These tests can be automated to run in CI CD, but at least once these needs to be run manually and validated for any UI error.

1. The feedback of these tests should be easily available and validated after each new build , to certify the UI.
2. We can have a separate QA to run these tests if the resource allocation can be done.

Test Approach for Different Markets/ Regions:

To make sure the App is compatible for different regions and even new markets once expanded, the main tests to be covered are,

1. Localization testing - testing in different countries with different languages
2. Globalization testing - testing in different countries with same language setting
3. Cross border testing - switching between one market to another

Take few regions , for example if the App is designed for US and then expanded to other countries like UK , Canada etc, few sample tests ( may be a regression suite ) should be run across these countries and validated. Users to be created in those countries.

Likewise, if the App can be expanded to Markets with different languages, the suite should be robust enough to change only few assertion points ( which involves localization language changes to those specific regions) and quickly run and validated.

The assertion points which are language specific should be kept in seperate files for each markets and validated across the specific market.

Along with testing localisation with one market/ region, cross border testing also needs to be carried out, like one User from one country buys / purchases from another country. How does the shipping cost work, what is the price currency etc.

We should have localization QAs ( ex if we are testing in Chinese we need to have a localized language QA to validate the key features once) to do a regression testing to make sure the language specific tests are passing.

The resource allocation should be done in such a way that we get their time as and when needed. This QA can be from the client side as well. In such case , we can get the test cases well ahead from them and run , before it passes on to the client to validate.

The above points should be clearly called out in User stories and validated with Regression suite run specifically for a set of cases run for Cross Border trading.

Test Quadrants:

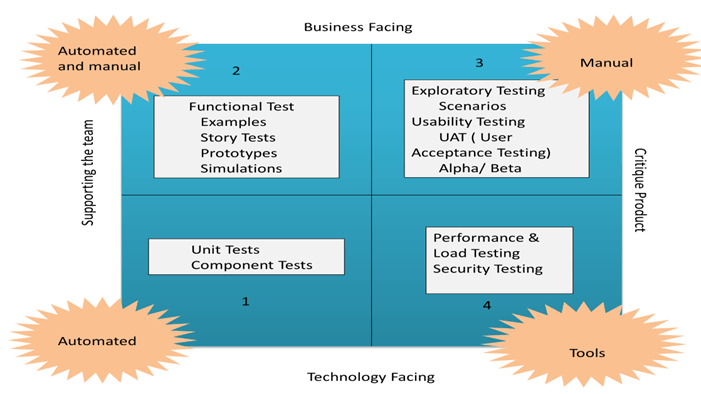
The agile testing quadrants separate the whole process in four Quadrants and help to understand how agile testing is performed,

Quadrant 1 : Unit testing , Component testing , API testing

Quadrant 2: Functional UI Manual tests, Functional UI automated tests, Tests with mocks / Stubs/ Simulations

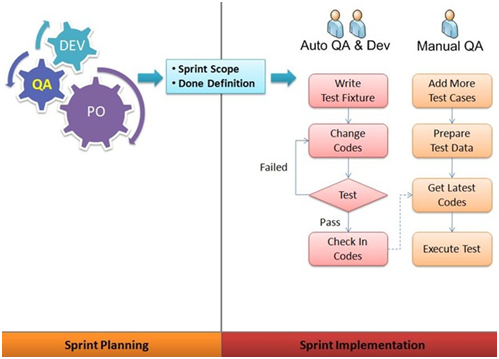
Quadrant 3: Exploratory testing , UAT, Accessibility tests , Cross browser, cross Device testing

Quadrant 4: Performance tests, Load tests, Security Tests wrt Authentication, Hacking etc.



Cross browser and cross device testing can come in both Quadrant 3 and quadrant 2 if they can be automated. But its always better to run them manually once, so they should surely be covered under quadrant 3.

Cross border testing and localization testing will also be covered under quadrant 3, extensively but a small test of P1 P2s for these tests can be added to the regression suite as well to validate them after each build through Continuous Integration (CI/CD pipeline).

Test execution Flow in a TDD Approach:

Test Environments

|  |  |
| --- | --- |
| Environment Type | Link to Access |
| DEV | <<Dev Link>> |
| QA | <<QA>> |
| INTEGRATION/ STAGING | <<Staging Environment>> |
| PRE-PROD | <<Pre-prod Link>> |
| PROD | <<Prod Link>> |

Mocks/ Stubs and Third Party Integration Environments:

* In addition to the above environments, plan needs to be done for any third party integration that needs to be facilitated for testing.
* If the third party integration can happen only at the later stage in the cycle/ sprint, there should be stubs or mocks in place to leverage the integration capability for carrying out integration testing.
* The mock/ stub should be designed well in advance before the Dev to QA happens , so that Integration testing can happen quickly and easily.

Example of a Mock in our application is : Payment gateway which needs to be integrated for testing.

Also while doing End to end testing, if the underlying APIs are not ready we can mock them as well for testing the UI flows.

Though a Mock or Stub is used, it is always a good idea to test by integrating directly with the third party once before UAT to avoid any last minute surprises.

For Different markets / regions, the local payment and shipping options needs to be integrated.

If those third party apps are not available , its advisable to mock them well in advance and test out well in the early phase of the user Story rather than to wait till the Third party integration happens.

<<Mock And Stubs link for Spree Ecommerce>>

Tools/ Framework for Testing

Manual Testing:

* Service / API tests - PostMan, SOAP UI
* Functional UI tests - Web Browsers

Automation Frameworks:

* Appium - For Mobile devices Automation
* Selenium - For UI Automation
* Rest Assured - for REST API testing

TestNG - Test Automation tool and reporting tool

Maven - Build tool to facilitate CI/ CD pipeline

Git - Version Control System to manage automation scripts and enable central storage of all tests

<<Link to the GitHub repo for this project>>

Test Management

Test Data:

1. Test Data for automation tests as well as manual tests can be prepared once the story/ Sprint starts and the Dev is in progress. This will facilitate quick test execution once the dev to QA happens.

2. Test Data to be got from the client ( for any integration testing ) , should be clearly communicated and the format of the Data to be documented.

3. The client data is to be validated against the standard format and approved well before the Dev to QA happens and before the project is in text execution phase.

For Spree Ecommerce App, the test data includes and not limited to,

* Registered users on the Application
* The different categories to be tested
* The products to be tested
* Users with different types of items in their cart
* Users in different sites / markets are to be tested along with normal users
* Items/ Products listed in different markets/ regions are also included in test data

These kind of test data can be prepared through automation in earlier, and can be executed when the test execution phase happens.

<<Test Data repository Link>>

Maintenance of tests:

* Automated tests are to be checked in and maintained in a Central Version control system. The proposed version control for Spree eCommerce App is GIT.
* Manual tests ( Acceptance Tests, Functional UI tests which cannot be automated etc) to be maintained in Excel sheets and checked in along with the project automation test suite to maintain a single point of lookup for the entire project documents.
* The tests are to be updated frequently as and when a new defect arises because of an unidentified / leaked test.

Issue/Defect Management

* Issue Management is done using JIRA.
* Tag the issues to the User Story the issue is currently raised on.
* If the User Story is complete, then the issue goes to the backlog and is picked in the next sprint.
* The Issue Management Process should be repeated at regular intervals throughout the duration of the project or program.
* In a program context, you may find that it makes sense to increase the frequency at which you run the Issue Management Process as your program nears completion, even running the process twice daily to ensure issues are being closed off quickly and effectively as time runs out.

<<Link to User Stories for Spree Ecommerce in JIRA>>

Scope , Timelines and Milestones

|  |  |  |  |
| --- | --- | --- | --- |
| User Story / Task | Weeks/ Sprints | Resources allocated | Milestone / Date of Release |
| User story1 <<Jira Link>> |  |  |  |
| User Story 2 |  |  |  |
| Integration Testing |  |  |  |
| UAT |  |  |  |
| Pre - Production Testing |  |  |  |
| Done Criteria |  |  |  |

For Each User Story, the milestones can be further fixed for each of the different tests as and when the sprint starts.

Resource allocation:

* No of QA resources required for the project
* No of QA for each sprint
* No of Automation engineers needed
* Any special needs for a specific tool expertise needed or not, to be called out

Risks and Mitigation plans

* Delay in QA Environment Readiness might impact QA activities and estimates, which could lead to delay in delivery
* Unavailability of Resource Expertise & Specialisation in planned automation tool might impact QA estimation
* Failure to identify complex functionalities and time required to develop those functionalities would impact QA estimates & project delivery
* Unexpected project scope expansions would impact current QA efforts and additional efforts would be estimated for the same

RCA ( Root Cause Analysis):

Each Defect with a Severity and Priority of 1 or 2 will have to undergo the RCA process to avoid leakage of those defects in further cycles.

Root : Where the Defect has been caught

Cause: What caused the defect and where it got leaked

Analysis: How did the defect leak and where and how it could have been prevented.

Once the RCA is done, the particular test can be added to the corresponding test suite for regression of further User Stories.

Test Coverage Analysis

Test coverage measures the amount of testing performed by a set of test.

It will include gathering information about which parts of a program are actually executed when running the test suite in order to determine which branches of conditional statements have been taken.

What Test Coverage does -

* Finding area of a requirement not implemented by a set of test cases
* Helps to create additional test cases to increase coverage
* Identifying a quantitative measure of test coverage, which is an indirect method for quality check
* Identifying meaningless test cases that do not increase coverage.

Test Coverage will be accomplished -

* Test coverage can be done by exercising the static review techniques like peer reviews, inspections and walk-through
* By transforming the ad-hoc defects into executable test cases
* At code level or unit test level, test coverage can be achieved by availing the automated code coverage or unit test coverage tools like Junit Coverage.
* Functional test coverage can be done with the help of proper test management tools

Key metrics and reporting

* # of Test Cases
* # of Manual Test Cases
* # of Automated Test Cases
* # of Passed Manual Test Cases
* # of Passed Automated Test Cases
* # of Failed Test Cases
* # of Test Cases Blocked
* Percentage TCs Passed

Other Metrics can include,

* Defects metrics ( No of Defects logged, no of P1s, No of defects open at each checkpoint etc)
* Test Coverage - Manual Test coverage, Automation test coverage , Automation test pass percentage
* Defect Leakage percentage
* Environment downtime

Automation Reports can include, ( These report to be published for each of the automation suite running in Pipeline and then collated)

* Type of test
* Total number of Tests
* No of tests passed
* Test pass percentage

Done Criteria

A Definition of Done drives the quality of work and is used to assess when a User Story has been completed.

"Done means every task under the User Story has been completed and any work created is attached to the User Story so the Product Owner can review it and make sure it meets his or her expectations."

Example of a Done Criteria for Spree eCommerce can be:

Any User is successfully able to login and do a purchase and can see the purchased product in his Order Page.

Each user Story ( and the project ) should have a done criteria and the activities are performed at the end and the story is marked done.

Once all the above activities are completed and no issues found, a story is Done!