TEST STRATEGY - PLAN

Group #7

Table of Contents

1. Purpose of this document

This document has been created in order to help team have a good overview on all aspects of testing activities which we will cope with in the whole project timeline. Since test planning for agile is sprint dependent, at the end of this document we will sketch out the details of generic test plan for each sprint.

All of these details are to keep the project development process running consistently: catching up with other activities in each iteration, reducing overheads, delivering releases on time, easily dealing with frequent changes or risks caused by project's demands, etc., and in short to ensure the final outcome is a working software.

2. Guiding standards

Principles	Description
Team responsibility	- In testing context (in other contexts as well), each team member is responsible for the quality of their work and also their team members' work. Specifically, one will have to do two separate things, the first thing is working on a specific set of test tasks assigned to him as planed in each sprint (iteration), and the second is reviewing the work done by others. The whole team is responsible for the team's work as a partial contribution to the cluster and the class project.
Automation	- Make use of automation tools and continuous integration. - Never carry out testing activities manually, unless it's really needed.
Test code management	- All test code just as other code must follows ruby and rails conventions.
Test data management	- All of test data must be generated with careful consideration and stored in appropriate places
Test configuration management	- All configurations for testing must be saved in one place – easy to look up and well organized with timestamp attached.

3. Test scope

The project is intended to be developed in Ruby on Rails framework as a web-based application with principles of Software as a Service for distribution model. We are also employing Agile development model for our project. Therefore, test activities will basically tie to testing techniques, strategies and other associated aspects that are compatible with the mentioned context.

4. Test approach

Type	Definitions	Test tools
Unit	Unit testing is a testing approach in which individual units or elementary modules of the system are tested for their supposed functionalities.	Rspec, Rails built-in unit tests for model, HardMock, Mocha, Rspec mocking library
Integration	In integration testing, individual modules are combined together as groups of components. These groups then are tested. Integration testing is carried out after unit testing.	Cucumber, Cabybara, Selenium, Rails built-in integration tests for action flows
Functional	Functional testing is used to validate that the system functions according to requirements of customers	Rspec, Rails built-in functional tests for controllers
Acceptance	Tests based on specification provided by customer to check if the system fits customer's needs	Rspec and Capybara or Selenium
Performance	Performance tests is a kind of integration tests, evaluating underneath performances of the system such as process time, memory, etc.	Apache Bench

5. Test environments

- Rails' support
  Simulation environment for HTTP requests is provided by Rails. Accordingly we can conduct our tests without the presence of a browser.

  The developed Rails application and the tests for it need a database to interact with. Rails provide a separate configuration for testing in *config/database.yml* file. Setting up database environment should be done at the beginning stage of the project development.

  Rails also have other built-in mechanisms to help us with testing. For each rails application, they provide a test folder which contains other logical sub-folders inside. We will develop our tests in this organized environment.

- Database: Oracle, Postgres, MySQl, sqlite3, etc.
- Web server**:** WEBrick, Apache, etc.
- Operating system: Linux mint (Petra), Windows 7/8, Mac OS, etc.

6. Test data management

Rails provides us test environment whose separate database is configured in config/database.yml. This test database enables us to manage our test data and operates testing operations. What we need to do is following the conventions guided by Rails documentation, and keeps track of what we do in this testing regard.

We can use:

- fixtures to manage test database with predefined data.
- randexp to generate things like names, emails addresses and urls.
- part of Production data as a source of test data for various types of test.
- the whole of Production data as source of test data for performance test

7. Defect management

7.1 Defect prioritization

Defect prioritization is based on scope and severity of defects.

Defect Scope

Ranking	Description
4	Defect has adverse impacts on a wide range of users or system functionalities
3	Defect has adverse impacts on a moderate range of users or system functionalities
2	Defects has adverse impacts on a small range of users or system functionalities
1	Defects has an adverse impacts on a minima range of users or system functionalities

Defect Severity

Ranking	Description

5	Data lost or corrupted, system crashes
4	Critical feature is not available without a workaround
3	Critical feature is not available with a realistic workaround
2	Secondary feature is not available with a realistic workaround
1	Other features are not available with a simple workaround

Priority

Scope\Severity	1	2	3	4	5
1	1	2	3	4	5
2	2	4	6	8	10
3	3	6	9	12	15
4	4	8	12	16	20

7.3. Defect resolution

20	Critical, must have immediate actions to fix or relieve the situation inform all parties involved to get around and handle the problems.
12-16	Serious, quickly set up time for working on the defects, inform al responsible people.
6-10	Moderate, plan time to fix problems
1-5	Low, plan time to fix problems

8. Test plan for a sprint

8.1 Test planning execution in a sprint

In each iteration session, we will allocate time for test planning. We make use of the following check-list to keep track current testing needs and make sure a sufficient testing coverage and test quality. The check-list after filled up with answers will be translated to workable tasks to put on the sprint backlog.

Check-list:

What	- needs to test and not to test - scope of test - test goals - sprint goals
Who	- will test different parts of the developed functionalities, features o behaviors - in scrum, normally have 1 - 3 testers (ideally 2 to have someone t discuss and plan with) - who will test cross-functional or characteristics involving other tean (this should have more planning).
Where	- test environment needed, and when needed - required configuration in test environment - look ahead to coming sprint to prepare new features test (tools, test data, preparation and/or configurations)
Why	- specific areas needs to be tested - other areas needs not to be covered
When	- when different testing tasks are started in current iteration - How often tests are tested (regression test) - When to start cross-functional tests with other team in cluster
How	- Test approaches? - Test methods, techniques, test data, tools?
Perquisites	- Prerequisites needed to start a specific test task? For examples: Expert knowledge, training, skills needed? or any dependency?
Dependencies	- Relationship, dependencies to: Functions, code modules, system, external system, tools, technology, other tasks constraints, test types.
Risks	- Most risks - Least risks - Potential risk related to the features about to be tested
Priorities	- Parts of higher/lower priorities for testing?
Time	- Time available for test activities

	- Time needed for test activities - Deadline for delivering the testing outcome <Definition of "Done"?>

After the backlog for test tasks is done with the fully answered check list, we will assign each member to a specific subset of tasks in the test backlog, and that member will be assigned a reviewer (another one in team) who will validate the work done by that member.

8.2 Test documentation

Tester must create a short summary of what he has done after finishing his set of testing tasks, including things associated with test design, test data, procedures, environment, defects detected, defect resolution and any configuration.

After the validation of his work, important information such as configuration, test data info, etc. will be recorded in appropriate places and live in there through the lifecycle of the project.

9. References

Adrian, S 2011, 'Agile test strategy template', viewed 10 June 2014, http://ennova.com.au/blog/2011/05/agile-test-strategy.

Anders, C 2011, 'Test planning in Agile process', viewed 4 June 2014, http://agiletestenea.wordpress.com/2011/06/01/test-planning-in-agile-projects/.

Michael, L 2010, 'A simple agile defect management process', viewed 12 June 2014, http://michaellant.com/2010/05/25/a-simple-agile-defect-management-process/.