**8hourjobs**

**[Job searching and Posting site]**

Business Requirements Document

|  |  |
| --- | --- |
| Initiative Sponsor | Rahul / Ichip Solutions |
| Version | 0.1 |
| Status | Initial Draft |
| Date | 06/09/2015 |
| Author(s) | Raji Rudhrakuamr  Test Analyst |

Test Strategy

**Scope and Overview:**

The scope of this document is to create a shared understanding of overall targets to build a website for a person who is looking for some casual or temporary work. That website should be accessed by both jobseeker and employers.

This document consists of information like who is the reviewer and approver for this project. To defines the testing activities and phases to be carried out.

**Test Approach:**

Define testing process, level of testing, roles and responsibilities of every team member. For every [test type](http://www.softwaretestinghelp.com/category/types-of-testing/) defined in test plan (e.g. [unit](http://www.softwaretestinghelp.com/unit-testing/), integration, system, regression, [installation/uninstallation](http://www.softwaretestinghelp.com/software-installationuninstallation-testing/), usability, load, performance, and security testing) describe why it should be conducted along with details like when to start, test owner, responsibilities, testing approach and details of automation strategy and tool if applicable.

In test execution there are various activities like adding new defects, defect triage, defect assignments, re-testing, regression testing and finally test sign-off. You must define exact steps to be followed for each activity. You can follow same process which worked for you in your previous test cycles. A visio presentation of all these activities including number of testers and who will work on what activity is very helpful to quickly understand roles and responsibilities in the team.

E.g. defect management cycle – mention the process to log new defect. Where to log it, how to log new defects, what should be the defect status, who should do defect triage, whom to assign defects after triage etc.

Also define change management process. This includes defining change request submission, template to be used, and process to handle the request.

**Step #3 – Test Environment:**

**Test Environments strategy**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Description** | **Data Setup** | **Usage** |
| Development | This environment is local and specific to each developer/tester machine. It is based on the version/branch of source code being developed. Integration points are typically impersonated. | Data and configuration is populated through setup scripts. | Unit, Functional and Acceptance Tests. Test tools e.g. Xunit test tools (Nunit, Junit), Mocking tools. Source code management for version control |
| Integration | This environment supports continuous integration of code changes and execution of unit, functional and acceptance tests. Additionally, static code analysis is completed in this environment. | Data and configuration is populated through setup scripts. | Unit, Functional and Acceptance Tests. Static code analysis Continuous Integration tools e.g. Cruise control |
| Staging | This environment supports exploratory testing | Populated with post-analysis obfuscated production data | Exploratory testing |
| Production | Live environment | New instances will contain standard project reference data. Existing instances will have current data migrated into the environment | Production verification testing |

Test environment setup should outline information about number of environments and required setup for each environment. E.g. one test environment for functional test team and another for UAT team. Define number of users supported on each environment, access roles for each user, software and hardware requirements like operating system, memory, free disk space, number of systems etc.

Defining test data requirements is equally important. Provide clear instruction on how to [create test data](http://www.softwaretestinghelp.com/tips-to-design-test-data-before-executing-your-test-cases/) (either generate data or use production data by masking fields for privacy).

Define test data backup and restore strategy. Test environment database may run into problems due to unhandled conditions in the code. I remember the problems we faced on one of the projects when there was no database backup strategy defined and we lost whole data due to code issues. Backup and restore process should define who will take backups, when to take backup, what to include in backup, when to restore database, who will restore it and data masking steps to be followed if database is restored.

**Step #4 – Testing Tools:**

Define test management and automation tools required for test execution. For performance, load and security testing describe the test approach and tools required. Mention whether it is open source or commercial tool and how many users are supported on it and plan accordingly.

**Step #5 – Release Control:**

As mentioned in our last [UAT article](http://www.softwaretestinghelp.com/successful-user-acceptance-testing/), unplanned release cycle could result into different software versions on test and UAT environments. Release management plan with proper version history will ensure test execution of all modifications in that release.

E.g. set build management process which will answer – where new build should made available, where it should be deployed, when to get new build, from where to get the production build, who will give go, no-go signal for production release etc.

**Step #6 – Risk Analysis:**

List all risks that you envision.  Provide a clear plan to mitigate these risks and also a contingency plan in case if you see these risks in reality.

**Step #7 – Review and Approvals:**

When all these activities are defined in test strategy plan it needs to be reviewed for sign-off by all entities involved like project management, business team, development team and system administration (or environment management) team. Summary of review changes should be tracked at the begging of the document along with approver name, date and comment. Also it’s a living document meaning this should be continuously reviewed and updated with the testing process enhancements.

**Conclusion:**

Test strategy is not a piece of paper. It’s the reflection of whole QA activities in software testing life cycle. Refer this document time to time in test execution process and follow plan till the software release. When project nears the release date it’s fairly easy cut on testing activities by ignoring what you have defined in test strategy document. But it is advisable to discuss with your team whether or not cutting down on any particular activity will help for release without any potential risk of major issues post release.

Most of the agile teams cut down on writing strategy document as team focus is on test execution rather than documentation. But having a basic test strategy plan always help to clearly plan and mitigate risks involved in the project. Agile teams can capture and document all high level activities to complete test execution on time without any issues.

*I’m sure developing a good test strategy plan and committing to follow it will definitely improve testing process and quality of the software. It would be my pleasure if this article inspires you to write a test strategy plan for your project!*