<Project Name>

**Test Strategy**

**<Document Version><Date>**

**<Prepared by>**

Table of Contents

[1. Introduction: 5](#_Toc10981113)

[1.1 Purpose: 5](#_Toc10981114)

[1.2 Objective: 5](#_Toc10981115)

[1.3 Scope: 5](#_Toc10981116)

[1.4 Testing Principles: 5](#_Toc10981117)

[2. Test Strategy: 6](#_Toc10981118)

[2.1 Testing Methodology: 6](#_Toc10981119)

[2.2 Test Plans Development: 7](#_Toc10981120)

[2.3 Master Test Plan: 7](#_Toc10981121)

[2.4 Detailed Test Plan: 7](#_Toc10981122)

[2.5 Test Design & Preparation: 7](#_Toc10981123)

[2.6 Test Summary Report: 9](#_Toc10981124)

[3. Testing Approach 10](#_Toc10981125)

[3.1 Testing Objectives 10](#_Toc10981126)

[3.2 Testing Framework: 10](#_Toc10981127)

[3.3 Test Execution: 11](#_Toc10981128)

[3.4 Testing Challenges: 11](#_Toc10981129)

[4. Risk, Assumptions, Dependencies and Escalation Process 12](#_Toc10981130)

[5. Automated Testing Tools: 13](#_Toc10981131)

[5.1 Test Management: 13](#_Toc10981132)

[5.1.1 Test Management Tool: 14](#_Toc10981133)

[5.2 Defect Management Tool: 14](#_Toc10981134)

[5.3 Performance Testing Tool: 15](#_Toc10981135)

[5.4 Licensing and Installation: 15](#_Toc10981136)

[6. Test Environment Strategy 16](#_Toc10981137)

[6.1 Test Environment Management and Provisioning 16](#_Toc10981138)

[7. Test Data Strategy 17](#_Toc10981139)

[7.1 Test Data Strategy Objectives 17](#_Toc10981140)

[7.2 Test Data Management 17](#_Toc10981141)

[7.2.1 Types of Test Data 17](#_Toc10981142)

[8. Testing Controls & Procedures 18](#_Toc10981143)

[8.1 Entry and Exit Criteria 18](#_Toc10981144)

[8.2 Defect Management 18](#_Toc10981145)

[8.2.1 Defect Management Severity and Priority Code Definitions 18](#_Toc10981146)

[9. Test Design Approach 19](#_Toc10981147)

[9.1 Test Case Priorities 19](#_Toc10981148)

[9.2 Positive and Negative Test Cases 19](#_Toc10981149)

[9.3 Test Case Review Process 19](#_Toc10981150)

[10. Test Execution Approach 19](#_Toc10981151)

[10.1 Continuous Integration Testing Approach 19](#_Toc10981152)

[11. Testing Team roles and Responsibility and Governance 20](#_Toc10981153)

[11.1 Roles and Responsibilities 20](#_Toc10981154)

[11.2 Testing Governance 20](#_Toc10981155)

[11.2.1 Testing Meeting 20](#_Toc10981156)

[11.2.2 Change Management 20](#_Toc10981157)

**Document Control**

**Approvers**

|  |  |  |
| --- | --- | --- |
| **Name** | **Responsibility** | **Approval** |
|  |  |  |
|  |  |  |
|  |  |  |

**Author**

|  |  |
| --- | --- |
| **Name** | **Responsibility** |
|  |  |

**Reviewers**

|  |  |  |
| --- | --- | --- |
| **Name** | **Role** | **Level of Review** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Introduction:

< It will be best to provide a brief narrative introduction to the product or service offering. It should include the product history, reasons for this introduction and the changes, expected outcome of

the changes, who might use it and the benefits of them using the new or enhanced product>

## Purpose:

This Test Strategy have the purpose to define the overall approach that will be taken by the Testing Team when they will deliver these testing services to all the projects within the business.

The document will help to clarify the testing activities, roles, responsibilities, processes and practises to be used across the projects. Where a project’s testing needs, may deviate from what is covered by this Test Strategy, the exceptions will be detailed in the Test Plan.

## Objective:

This document will define the *<Project Name>* Test Strategy and approach of incremental testing stages which are required to ensure the acceptance of the solution to be delivered. It covers all the phases and releases. The foundation of the test and acceptance processes will be based on *<Project Name>* existing Guidance. These processes will need to be enhanced to embrace and achieve the acceptance conditions and criteria as mentioned within the document.

## Scope:

The Test Strategy will cover the below aspects:

* Identifying the different types of testing which will be undertaken throughout the lifecycle of development till live operations.
* Details of every ongoing testing of service enhancement and changes.
* The scope of each type of testing to be taken place.
* Targeting how common expectations and testing standards are to be achieved for all types of testing
* The high level technical, resource and environmental requirements which will be required.
* The key testing and quality assurance procedures that will be required.

## Testing Principles:

* Testing shows presence of defect.
* Exhaustive testing is not practically possible, instead risks and priorities can be a good option.
* Early testing practices should be adopted.
* Defect clustering.
* Pesticide paradox, new and different test cases should be written to find more defects.
* Testing is requirement dependent; each requirement demands its own ways of testing.

# Test Strategy:

## Testing Methodology:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Functional Testing** | **Test Objective** | **Techniques** | **Completion Criteria** | **Special Considerations** |
| **Unit Testing** |  |  |  |  |
| **Integration Testing** |  |  |  |  |
| **System Testing** |  |  |  |  |
| **Sanity Testing** |  |  |  |  |
| **Smoke Testing** |  |  |  |  |
| **Interface Testing** |  |  |  |  |
| **Regression Testing** |  |  |  |  |
| **Beta/Acceptance Testing** |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Non-Functional Testing** | **Test Objective** | **Techniques** | **Completion Criteria** | **Special Considerations** |
| **Performance Testing** |  |  |  |  |
| **Load testing** |  |  |  |  |
| **Stress testing** |  |  |  |  |
| **Volume testing** |  |  |  |  |
| **Security testing** |  |  |  |  |
| **Compatibility testing** |  |  |  |  |
| **Installation Testing** |  |  |  |  |
| **Recovery testing** |  |  |  |  |
| **Reliability Testing** |  |  |  |  |
| **Usability Testing** |  |  |  |  |
| **Compliance Testing** |  |  |  |  |
| **Localization Testing** |  |  |  |  |

## Test Plans Development:

< Include your text here>

## Master Test Plan:

< Include your text here>

## Detailed Test Plan:

<Include your text here>

## Test Design & Preparation:

Test design is basically an act of creating and writing test suites for testing a software. The test design component addresses the need to define the number of tests to be performed, the ways those testing will be approached i.e. (paths, functions, test conditions) that needs to be exercised. Test design standards need to be defined and followed. Automated tests should to be reusable, repeatable, and maintainable.

**Table 1: Test Design Procedure**:

|  |  |
| --- | --- |
| **Steps #** | **Description** |
| 1 | **Test Architecture Review**  The testing team reviews the test architecture in order to identify the testing techniques that apply. |
| 2 | **Test Procedure Definition (Development Level)**  A test procedure definition is made at the development test level, identifying the test procedure series that applies for the various design components and testing techniques. |
| 3 | **Test Procedure Definition (System Level)**  A test procedure definition is made at the system test level, identifying the test procedure series that applies for the various testing techniques. |
| 4 | **Test Procedure Design Standards**  Design standards are adopted with a unique naming convention that distinguishes the test procedures on the project from test procedures developed in the past or on other projects. |
| 5 | **Manual Versus Automated Tests**  Test procedures will be represented as being either performed manually or as part of an automated test. |
| 6 | **Test Procedures Flagged for Detailed Design**  Test procedures that stand out as more enlightened are flagged. These test procedures are further defined as a part of detailed test design. |
| 7 | **Detailed Design**  Those test procedures which are flagged as part of step 7 are designed in further detail within a detailed test design file or document. Test procedure detailed design may consist of pseudo-code of algorithms, preliminary test step definition, or pseudo-code of test automation programs. |
| 8 | **Test Data Mapping**  Test Data Mapping is Test procedure matrix which is modified to reflect test data requirements for each test procedure. |

**Table 2: Development/Testing Relationship**

|  |  |  |
| --- | --- | --- |
| **Phase** | **Development Process** | **Testing Process** |
| Module/Unit Development | Design module from requirements provided. | Perform test planning and test environment setup. |
|  | Code module. | Create test design and develop test data. |
|  | Debug module. | Write every test scripts or record test scenario using module. |
|  | Unit test module. | Debug the automated test scripts by running against module. Use tools that support unit testing. |
|  | Correct defects. | Re-run automated test script to regression test as defects are corrected. |
|  | Conduct performance testing. | Verify that system is adaptable and will meet performance requirements. |
| Integration | Build system by connecting modules.  Integration-test connected modules.  Review trouble reports. | Combine unit test scripts and add new test scripts that demonstrate module interconnectivity. Use test tools that supports automated integration testing. |
|  | Correct defects and update defect status. | Re-run automated test script as part of regression testing, as defects are corrected. |
|  | Continued performance testing activities. | Verify that system is adaptable and will meet performance requirements. |
| System Testing | Review trouble reports. | Integrate automated test scripts into system-level test procedures where possible and develop additional system-level test procedures. Execute system test and record test results. |
|  | Correct defects and update defect status. | Re-run automated test script as part of regression test as defects are corrected. |
| Acceptance Testing | Review incident reports. | Perform subset of system testing as part of demonstration of user acceptance test. |
|  | Correct defects. | Re-run automated test script as part of regression test as defects are corrected. |

## Test Summary Report:

Testsummaryreportis a formal document that outlines the results of all testing efforts for a testing cycle of a project / module or a sub module. Generally, Test Leads or Test Managers prepare this document at the end of testing cycle. Some Test Managers prepares it at the end of project. As one of the most important deliverable, the significance of test summary report is immense in [software development life cycle (SDLC)](http://www.professionalqa.com/software-development-life-cycle).

this report consists of all the necessary information related to the process of software testing as well as the results delivered by it. Other features of test summary report are:

Provides information about different activities performed during the testing process.

* Encapsulates the results of all testing efforts.
* Composed by the Test Leads or Test Managers.
* Test managers prepare this document at the end of the testing project.
* This includes details about the number of test cases executed, passed, failed, and blocked along with the comments provided by the testers.

|  |  |  |  |
| --- | --- | --- | --- |
| **Summary Assessment** | **Total number of Test Cases** | **Total Planned %** | **Comments** |
| **Test Cases Run** |  |  |  |
| **Test Cases Planned** |  |  |  |
| **Test Cases Reviewed** |  |  |  |
| **Test Cases Failed** |  |  |  |
| **Test Cases Passed** |  |  |  |
| **Test Cases to be Run** |  |  |  |
| **Test Cases Held** |  |  |  |

# Testing Approach

## Testing Objectives

Executing of test strategy for a project is known as "test approach". The test approach is usually defined in all test plans and test designs. Test approach refers to the onset of various project activities such as planning the testing process, selecting the designs, defining the entry and exit criteria etc. An efficient testing strategy should contain different types of QA testing that include manual, automated and exploratory testing. All of these bestow effectively tighten the release cycles and lower risks. Three types of QA testing must be included in a robust testing strategy which are unit tests to validate trivial components, integration tests to validate the compatibility of the components and functional tests to validate end-user scenarios.

There are two types Test Approach:

* Preventive Approach: An approach in which the test design process is initiated as early as possible in order to find and fix the defects before the build is created i.e. before the commencement of software development.
* Reactive Approach: An approach in which the testing is not started until the completion of design and coding i.e. after software development.

There are many strategies that a project can adopt depending on the context and some of them are:

* Dynamic and heuristic approaches.
* Consultative approaches.
* Model-based approach that uses statistical information about failure rates.
* Approaches based on risk-based testing where the entire development takes place based on the risk.
* Methodical approach, which is based on failures.
* Standard-compliant approach specified by industry-specific standards.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Testing Approaches** | **Test Objective** | **Techniques** | **Completion Criteria** | **Special Considerations** |
| **Exploratory Approach** |  |  |  |  |
| **Black Box Testing** |  |  |  |  |
| **White Box Testing** |  |  |  |  |
| **Grey Box Testing** |  |  |  |  |
| **Ad-Hoc Testing** |  |  |  |  |
| **Mutation Testing** |  |  |  |  |

## Testing Framework:

A Testing Framework is a set of guidelines or rules which are used for creating and designing Test Cases. A framework is formed of a combination of practices and tools that are designed to help QA professionals to test more efficiently and accurately.

There are different types of Testing Frameworks:

* Linear Scripting Framework
* Modular Testing Framework
* Data Driven Testing Framework
* Keyword Driven Testing Framework>
* Hybrid Testing Framework
* Behaviour Driven Development Framework

|  |  |  |
| --- | --- | --- |
| **Serial #** | **Framework** | **Requirements** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## Test Execution:

Test execution starts when the entry criteria has been satisfied. Test manager must ensure that the test execution starts only when entry criteria has been satisfied in order to avoid any unnecessary defects and delays in testing.

When test execution starts, the test analysts initiates the execution of the test scripts based on [test strategy](https://www.softwaretestingmentor.com/test-deliverables/what-is-test-strategy/) allowed in the project.

|  |  |
| --- | --- |
| **Serial #** | **Entry Criteria** |
|  |  |
|  |  |
|  |  |

## Testing Challenges:

Software Testing has a lot of challenges both in Manual Testing as well as in Automation Testing**.** As the software quality assurance prospect is rapidly growing and facing notable changes, QA and testing professionals are facing new challenges related to better understanding, managing, and adopting different types of QA testing. The extended use of agile testing has urged organizations to change their [development and QA strategies](http://techarcis.com/2915-2progressive-test-automation-to-expedite-the-testing-process/) to fulfil their customers’ needs.

Few of these challenges are highlighted below:

* Testing the complete application
* Misunderstanding of company processes
* Relationship with Developers
* [Regression Testing](https://www.softwaretestinghelp.com/regression-testing-tools-and-methods/)
* Lack of [skilled Testers](https://www.softwaretestinghelp.com/is-software-testers-job-really-low-profile-job/)
* [Testing always under time constraint](https://www.softwaretestinghelp.com/what-if-there-isnt-enough-time-for-thorough-testing/)
* Which tests to execute first
* Understanding the requirements
* [Automation Testing](https://www.softwaretestinghelp.com/10-tips-you-should-read-before-automating-your-testing-work/)
* The decision to stop the testing
* One test team under multiple projects
* Reusable Test scripts
* Testers focusing on finding easy bugs
* To cope with attenuation
* Increasing adoption of Open-Source
* Develop skills for Agile and DevOps
* Learn Coding and work along with Developers
* Adopt and Accept Test Automation over Manual Testing
* Dealing with the rapid changes in Web & Mobile Technologies
* Think beyond Testing, get logical
* **Inadequate Test Coverage**
* Code breakage Accidentally due to frequent Builds
* Early detection of Defects
* Inadequate API Testing
* Performance Gridlock

# Risk, Assumptions, Dependencies and Escalation Process

<The following table lists down the risk, its impact and the mitigation plan for the risk:>

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk Description** | **Impact** | **Mitigation Plan** | **Escalation Process** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Automated Testing Tools:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Serial #** | **Automation Tool** | **Version** | **Scripting language** | **Programming Skills** | **Requirement** | **Reason** |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

## Test Management:

An effective test management process should lead to one common target to deliver high quality products that meet customer demands. To achieve this goal, it’s crucial that teams work within a proven process framework. Rivalry among software suppliers to develop the best products in the little amount of time is leading to an increasing need for highly developed test management processes. As test teams work with development teams to deliver finished products within compact deadlines, the test management process has become the focus of attention. Ultimately, with the right focus, effective test management processes will not only increase customer satisfaction but will lead to smoother and more compatible projects.

**Test management activities**

Effective test management is parted down into several phases that includes planning, creation, execution and tracking.

**Test Planning:** The planning stage involves developing the overall direction and motive of the test phase, including the specifics of why, when and where to test. Tests are created when there is a test motivator present, such as a specific requirement that must be approved. What needs to be tested is parted down into multiple test cases, while the question of where to test is usually decided by documenting the required software and hardware arrangements. Once these details have been confirmed, the determination of when to test is made by tracking the test iterations, cycles or time period.

**Test Authoring:** In the authoring stage, the steps that are required to complete a given test are expressed, in order to answer the question of how a test will be conducted. In short, this process, is about describing generic test cases which are then parted down into thorough test steps. These steps can then be evolved as either manual or automated test scripts. For many teams this information is showcased and preserved in a test management tool.

**Test Execution:** During execution stage, the test cases are run in logical sets which are usually cited to as a test suite. Tests are run against a known arrangement of the software/hardware under test and/or against a test environment. It is important to record the configuration for the intension of reconstitute the tests at a later stage.

**Test Tracking:** Depending on the test management tools used by the team, test results are logged. These results are then displayed and outlined via dashboard or test metrics tool. Tracking is a necessity in the testing process, as quality metrics are required in order to productively track how the test effort is progressing, and to measure the quality of the system or application.

**Test Reporting:** Test reporting, gives you the capability to evaluate testing efforts and liaise test results to other interested parties. The objective of this is to establish the current status for project testing, but to also provide details about the overall quality of the application or system. Where a test management application is used to track the test process reporting is usually praised by project dashboards.

### Test Management Tool:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Manage-ment Tools** | **Version** | **Integration of Automation Testing** | **Flexibility** | **Real Time Reporting** | **Easy Tracking** | **Scalability** | **Ensured Security** | **Tracking Capability** |
| QTest |  |  | ✓ |  | ✓ |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

## Defect Management Tool:

Defect management process includes: Identification, Categorization, Prioritization, Assignment, Resolution, Verification, Closure, Management Reporting.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Defect Management Tools 🡪**  **Features 🡫** | Jira |  |  |  |
| **Version** |  |  |  |  |
| **Accessible from anywhere and anytime** | ✓ |  |  |  |
| **Ability to submit Defects from various sources easily** |  |  |  |  |
| **Customized Workflow** |  |  |  |  |
| **Integration with other Tools** |  |  |  |  |
| **Dashboard** |  |  |  |  |
| **Reporting Capabilities** |  |  |  |  |
| **Email Notifications** |  |  |  |  |
| **File Attachments** |  |  |  |  |
| **Support for a wide range of Custom Fields** |  |  |  |  |
| **Support for Escalation** |  |  |  |  |

## Performance Testing Tool:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Performance Testing Tools 🡪** | Jira |  |  |  |
| **Features 🡫** |  |  |  |  |
| **Version** |  |  |  |  |
| **URL Testing** | ✓ |  |  |  |
| **End User Experience** |  |  |  |  |
| **Monitoring** |  |  |  |  |
| **API Testing** |  |  |  |  |
| **Scalability** |  |  |  |  |
| **Real-Time Reporting** |  |  |  |  |
| **Monitoring Integrations** |  |  |  |  |
| **CI Integrations** |  |  |  |  |
| **Script Recording** |  |  |  |  |
| **Network Emulation** |  |  |  |  |
| **Testing for Developers** |  |  |  |  |

## Licensing and Installation:

A software license is a legal mechanism granting consumers to use or redistribute software. Without the license agreement, using the software would originate a contravene copyright law. The license agreement will describe to the end-user how they can use the software. All software should be legally licensed prior to installation. Where software licensing becomes confusing is in the different variance of licenses and the rights associated to the various licenses.

|  |  |
| --- | --- |
| **Software** |  |
| **Version** |  |
| **Type of Licence** |  |
| **Term of Licence** |  |
| **Licence Restrictions** |  |
| **Liability** |  |
| **Third Part Data Users** |  |
| **Business Days/Hours** |  |
| **Pricing** |  |
| **Termination Date** |  |
| **Installation Guide** |  |

# Test Environment Strategy

## Test Environment Management and Provisioning

Types of Enviroment in Software lifecycles include:

* The Analysis and Design environment
* The Development environment
* The Common Build environment
* The Testing Environment. It includes:
* The Systems Integration Testing environment
* The User Acceptance Testing environment

The importance of Environment Management and Provisioning is due to following reasons:

**Less Risk in Releases and Easier Fixes**

In some cases, a bad quality product is the result of inadequate test environments management - or a complete lack of test environments management. More the gap between the test environment and the production environment, the chances of delivering a defective product becomes higher, and the difficult it becomes to detect a problem back from its root cause. Multiple environments will differ in terms of their configuration, but they need to be managed within the frame of a single cloud management platform for utmost compatibility.

**More Stable and Productive IT Expenditures**

Deficient management of IT assets leads to a general lack of knowledge regarding the current state of each environment and a limited capacity to foretell appropriate resource allocation. This can create displeasing surprise budget spike and cause expensive delays as time is lost trying to identify and enact solutions.

**Improved Organization and Separation of Roles**

Using multiple test environments, its a great way to assure that you can clearly separate roles and limit access. Each environment can now be specified security protocols, best practices and procedures, allowing users for better enhancement of each environment for its specific purpose within your operations. In this scenario, different groups can test new features without depending on each other or having to wait until the previous group is finished with their test.

**Reliability and Dependability**

Using only one test environment increases the fear of downtime and interventions that can cause significant loss of profit. Using multiple test environments allows companies to regularly deliver a user experience that contemplate well on the reliability of their business. It also protects businesses against loss of data which is one of the problems experienced by environments.

# Test Data Strategy

## Test Data Strategy Objectives

* It should be started by discovering and understanding Test Data
* Subset production data from multiple data sources
* Mask or de-identify sensitive Test Data
* Refresh Test Data
* Automate Test Data result comparisons

## Test Data Management

Test data management is the process of creating realistic test data for nonproduction purposes such as development, testing, training or QA. Research shows that projects cancelled due to poor data quality are 15 percent more costly than successful projects of the same size and type. A better test data management strategy not only ensures greater development and testing efficiencies, but helps organizations identify and correct defects early in the development process, when they are cheapest and easiest to fix.

Basically, Test data management included two major activities: Test data preparation and Test data usage.

### Types of Test Data

Types of Test Data includes:

1. Valid: The most common or obvious type of data that should work.
2. Valid Extreme: Extreme, Unexpected and unusual data. Ex: Lowest and Highest (Checking the limits but it should work)
3. Invalid Data: For negative scenarios, data that should fail.
4. Invalid Extreme: Data that is at the edge of failure and is nearly acceptable.
5. Erroneous: Data that is the wrong data type.

**Test Data Preparation Approach:**

|  |  |  |
| --- | --- | --- |
| **Methods** | **Pros** | **Cons** |
| **Cloning Production Databases** |  |  |
| **Generating Synthetic Test Data** |  |  |
| **Subsetting production databases** |  |  |

# Testing Controls & Procedures

## Entry and Exit Criteria

|  |  |  |
| --- | --- | --- |
|  | **Activity** | **Responsibility** |
| **Entry Criteria** |  |  |
|  |  |
| **Exit Criteria** |  |  |
|  |  |

## Defect Management

After executing the test cases when a tester, sometimes he come across the test result which is contrary to expected result. This inequality in the test result is referred as a Software Defect. These defects or variance are known by different names in different organization like issues, problem, bug or incidents. These defects are manged using Defect Management Tools. Every organization use Defect Management Tool as per their convenience.

**Defect management process**

**Identification:** This is the first step in Defect Management, a tester team identify the defects and sometimes the customer also finds them. But it should be detected as earlier as possible.

**Categorization:** Once the defect is raised in the defect management tool, it is passed to the responsible person who checks it and categorize it as what type of defect it is and then it is moved to the next step.

**Prioritization:** According to the severity of the defect. These defects are given priorities for fixing it accordingly.

**Assignment:** After the priority of the defect set it is assigned to a developer to fix it.

**Resolution:** Developer reproduce the defect and fixes the defect, in the same place from where it was identified.

**Verification:** As soon as the defect is fixed, the developing team verifies that defect is fixed and working in the best way or not.

**Closure:** Once a defect is fixed and resolved it is marked as Closed.

**Management Reporting:** This is provided at regular intervals which helps the management keep a track of work.

### Defect Management Severity and Priority Code Definitions

|  |  |
| --- | --- |
| **Severity** | In Software Testing, Defect severity can be classified into four class: **Critical**: This class of defect specify complete closure of the process, nothing can proceed further. **Major**: This class of defect is highly severe defect and crashes the system. However, certain parts of the system remain functional. **Medium**: This class of defect causes some unwanted behaviour, but the system is still functional. **Low**: This class of defect won't cause any major break-down of the system. |
| **Priority** | The priority code indicates the impact of this defect on the project: **H** = High (extremely important, the project cannot be successful without this defect being resolved) **M** = Medium (important to project success, but a work around exists) **L** = Low (desirable, but with little effect to project success if the defect is not fixed) |

# Test Design Approach

## Test Case Priorities

Test cases need to be created with the following four priority levels:

* **1 - Urgent** - Test Case which Covers the most functionalities of the system. System should not move to next testing phases without executing these test cases.
* **2 - High** - Test Case which covers frequently used functionalities of the system. These test cases need to be executed before the system moves into production
* **3 - Medium** - Test Cases which covers with less frequently used functionalities of the system. These test cases can be executed even after the production implementation with the proper acceptance with customer
* **4 - Low** - Test cases which covers the rarely used functionalities or less important. Even if we do not execute these test cases it would not create any major issues in production

## Positive and Negative Test Cases

**Positive Test Cases**

Test Cases which are used to validate the non-error path or function is called positive test cases.

**Negative Test Cases**

Test Cases which are used to validate the Error Path or function is called negative test cases

## Test Case Review Process

Test cases need to be reviewed internally by the team and externally by the Business analysts, Respective Development team. Test Cases need to be uploaded in Quality centre after receives sign off from all the stake holders.

# Test Execution Approach

“Fixing defects in production is 10 times costlier that fixing a defect in the development phase”. To identify the Defect in the Development phase, so it’s better to start the test execution early in the life cycle by doing continuous integration testing followed by integration testing and System Testing.

## Continuous Integration Testing Approach

The Term “Continuous Integration Testing” is common in IT Testing Practice and it is a general Testing approach. Continuous Integration Testing is used to reduce the defects hit ratio in the later stage of testing phases.

# Testing Team roles and Responsibility and Governance

## Roles and Responsibilities

Following table represent the roles and responsibilities of everyone:

|  |  |  |
| --- | --- | --- |
| **Name** | **Role** | **Responsibility** |
|  |  |  |
|  |  |  |
|  |  |  |

## Testing Governance

### Testing Meeting

<Elaborate the test meeting schedule here>

### Change Management

<Any requirement change will be managed here>