Software Quality Assurance Plan

<Project Name>

<Department Name>

<Version Number>

Document history

Date	Version	Author	Reviewed by	Approved by	Description

Table of Contents

List of Tables

1. Introduction

1.1. Background

< Background of the Project>

1.2. Purpose/Objective

The main purpose of the Test Strategy Document for the proposed <Application Name> System is to document and define the "Software Quality Assurance Plan" in regards to the overall Project scope, including the following:

- High level Testing methodology for the project implementation
- Identification of the various Testing Phases, Types of Testing, Testing environments, Testing guidelines and Issue resolution
- Identification of the Testing activities/Deliverables
- Plan for Test execution
- Description as regards to participation and expectations for the test phases
- Approach to verify the work products/new system/all deliverables
- Resource requirements/allocations, including hardware/software, and staff responsibilities
- Requirements for environment and setup, for the test team
- High level business process scenarios that will be covered during testing
- An overview of the business flows, architecture, and testing infrastructure

The Testing activity will be carried over across the key phases as identified below:

- Test Planning
- Test Design and Development
- Functional and Regression Test Execution
- Defect Reporting and Management
- Test Summary and Sign-Off

The intended audience for the document is as follows:

- Project Business Sponsors
- Development Team
- QA Team
- IT Infrastructure and Network Services
- DBA (Database Administrator)

2. Scope of Work

Following identified functional requirements of the proposed <Application Name>, are considered to be in scope of testing:

<List of Modules to be tested>

3. Test Strategy

The strategy to be adopted for testing the proposed <Application Name> System is based on a phase-wise approach as detailed in the following sections.

3.1. Overall Test Strategy

<Mention actual project strategy. A typical Test Cycle will include the following steps:

- Step 1: Review of the requirements
- Step 2: Test Case preparation
- Step 3: Smoke Testing
- Step 4: System testing
- Step 5: Integration Testing
- Step 6: Regression Testing
- Step 7: Data Conversion Testing
- Step 8: Security and Access Control Testing
- Step 9: User Acceptance Testing(UAT)>

3.2. Testing Types

3.2.1. System Testing

The goal of System Testing is to verify proper data acceptance, processing, and retrieval, and the appropriate implementation of the business rules. This type of testing is based upon black box technique; that is verifying the application and its internal processes by interacting with the application via the Graphical User Interface (GUI), and analysing the output or results.

System Testing will include the following:

- User Interface Testing
- Functional Testing

3.2.1.1 User Interface Testing

User Interface (UI) testing will verify user's interaction with the software. In addition, UI testing ensures that the objects within the UI function as expected and conform to corporate or industry standards.

#	Item	Description
1	**Test Objective**	To verify the following: • Window objects and characteristics, such as menus, size, position, state, and focus conform to standards. • UI of the pages should be as per the requirements. It would include various things like look-and-feel of the page, navigation within page, navigation to reports, display of controls, font, colour, size etc. • UI of the Administrative screen should be as per the requirements of the UI guideline of the project. It would include look-and-feel of the page, navigation, font, colour, size etc.
2	**Technique**	Create tests for each screen, pages and reports to verify proper navigation.
3	**Environment**	Will be carried out in <Development/QA/Prod/UAT> environment.
4	**Entry Criteria**	• Unit testing is complete • UI checklist, System test plan and Test specifications are ready.
5	**Exit Criteria**	• All planned tests have been executed. • All identified defects have been addressed.
6	**Special Considerations**	Bug fixing effort and retesting effort will impact the total effort estimated for testing.

Table 1: User Interface Testing Details

3.2.1.2 Functional Testing

The goals of these tests are to verify business functionality of the application. Also included are the testing of the screens for data, navigation and functionalities of the application. Focus will be on testing the dashboard and all the reports for data, report formats and navigation.

#	Item	Description
1	**Test Objective**	Ensure proper target-of-test functionality, including navigation, data entry, processing, workflow and retrieval
2	**Technique**	This phase will be executed in each of the iteration. Execute each use-case flow, or function, using valid and invalid data, to verify the following: • The expected results occur when valid data is used. • The appropriate error or warning messages are displayed when invalid data is used. • Each business rule is properly applied • All the workflows will send notification to assignee as and when required. • Reports will be verified for correct data population • Additionally, Front end data will be verified through data verification from the database.

#	Item	Description
3	**Environment**	Will be carried out in <Development/QA/Prod/UAT> environment.
4	**Entry Criteria**	• Unit testing is complete. • System test plan is ready. • Data Plan is ready.
5	**Exit Criteria**	• All planned tests have been executed. • All identified defects have been addressed.
6	**Special Considerations**	• Testing team will verify the business rules and functions specified in the documents provided. • Any change of business rule and functions will have an impact on the test plan and testing cycle. • Bug fixing effort and retesting effort will impact the total effort estimated for testing.

Table 2: Functional Testing Details

3.2.2. Integration Testing

Integration Testing will be carried out at the end. This phase will concentrate on the integration of all of the modules. Data flow will be validated end-to-end to ensure smooth functioning of the system as a whole.

#	Item	Description
1	**Test Objective**	Ensures that the end product is working as expected with all the different modules integrated at the end.
2	**Technique**	Check for Interfacing points between each of the modules completed: • Verify for end-to-end data flow across modules • Verify the functional requirement across modules
3	**Environment**	Will be carried out in QA / Staging environment.
4	**Entry Criteria**	• Completion of the development and system Testing activities for all the modules • Integration Test points have been identified and Test Cases are prepared
5	**Exit Criteria**	• All the Test cases have been executed and Test coverage status is 100% • All the Test cases are passed with no severe bugs/issues.
6	**Special Considerations**	• Sufficient ramp up time required for the testers to understand the design and data flow. • A well-planned coordination is needed between teams to carry out such Testing. • Bug fixing effort and retesting effort will impact the total effort estimated for testing.

Table 3: Integration Testing Details

3.2.3. Regression Testing

Regression Testing would be carried out in the application in different iterations. A regression suite would be built for each of the module and Testing will be carried out to ensure that all the functionalities implemented earlier are as they were.

#	Item	Description
1	**Test Objective**	Ensure that the functionalities implemented in each of the iterations for different modules/functionalities remains intact after the implementation of subsequent modules.
2	**Technique**	Functionalities for each of the modules will be implemented in each of the iteration, as defined earlier. So it is imperative that modules implemented in the later iterations will be tested along with some of the functionalities from earlier iterations.
3	**Environment**	Will be carried out in QA / Stage environment.
4	**Entry Criteria**	• Functional Testing is complete for current module • Regression suite is prepared for earlier modules.
5	**Exit Criteria**	• All the Test cases are passed with no severe bugs/issues
6	**Special Considerations**	None

Table 4: Regression Testing Details

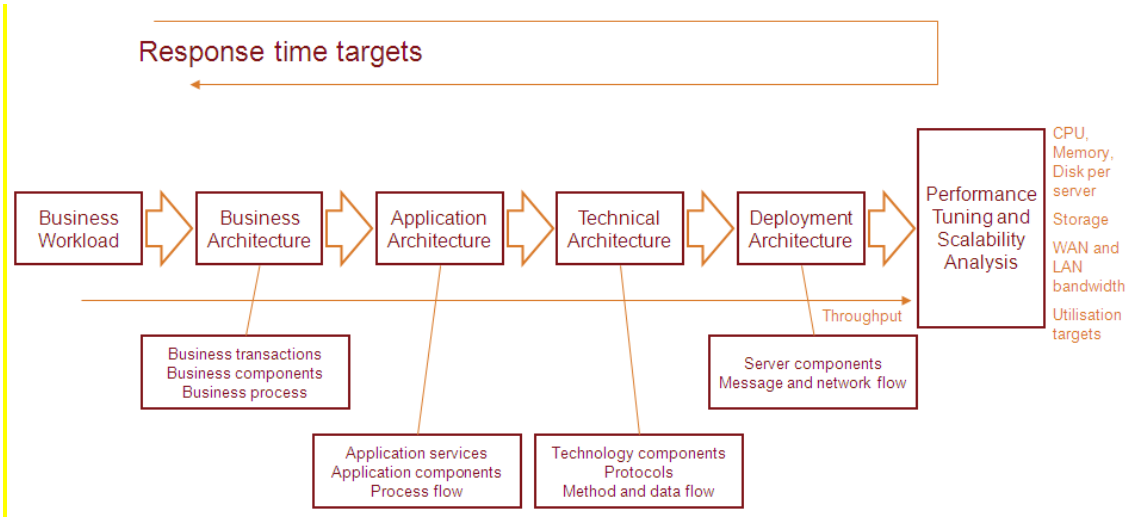
3.2.4. Data Conversion Testing

<Optional, it will be required for migration project or as per contract>

3.2.5. Performance Testing

<Optional, can be mandatory as per contract>

The standard top-down approach that follows along with the sequential steps involved for a typical performance/scalability testing engagement are depicted in the following diagram.



This approach is based on business perspective unlike bottom-up approach, which takes into consideration the technical perspective. In the top-down approach, focus is on response time for any transaction mix and user volume, irrespective of the Hits/sec or Throughput.

<Department name> decides on the approach depending on the business goal and objective for any client like system integrity, performance benchmarking, performance enhancement and performance problem diagnosing, which drives the perspective of conducting this particular test type.

3.2.6. Security and Access Control Testing

Security and Access Control Testing focus areas

3.2.7. User Acceptance Testing

User Acceptance Testing (UAT) focuses mainly on the functional requirements of the application and performed by the real users of the system.

#	Item	Description
1	**Test Objective**	To verify the following: • End system behaves in the desired way as specified in the specification requirement document
2	**Technique**	• Identify a set of comprehensive Test cases to be the candidate for UAT • Run the Test set to measure the coverage of User requirements
3	**Environment**	Will be carried out in Pre-production environment
4	**Entry Criteria**	• System Testing along with all other Testing is complete. • Test data specific to user and role are available to the testing team. • Users are identified to perform the Testing
5	**Exit Criteria**	• For each known actor type the appropriate function or data are available • All the Tests are completed with sufficient no of iterations

#	Item	Description
		• All Sev1 and Sev2 Defects are closed
6	**Special Considerations**	• Test data should be available specific to user/group level and role. • Bug fixing effort and retesting effort will impact the total effort estimated for testing

Table 5: User Acceptance Testing Details

3.3 Tools

The following tools will be employed for the Testing activities involved in the project:

#	Item	Tool	Vendor/In-house
1	Test Management	TFS	
2	Defect Tracking	DGT	
3	Version Control	TFS– SVN –	
4	Project Management	MPP	

Table 6: List of Tools for Testing Activities

3.4 Defect Management

Defect management would be taken care in Team Foundation Server (TFS). All the defects and issues during the Testing lifecycle would be logged in TFS. All the developers and Testers would be provided access to the respective Project in TFS accordingly to work on the Defects.

The workflow that would be followed for Defect Management is as follows:

1. Testers would typically log all the Defects and issues in TFS and assign it to particular Business Analyst/Dev Lead with a Status 'Open'. Any Requirement clarifications would need to be assigned to Business Analyst.
2. Developers will keep an eye on the assigned issues/defects each day. They will change the status to 'Work on Progress' once they start working.
3. The status would be changed to 'Fixed' upon fixing the Defects.
4. Testers would re-test the Defects and change the status as 'Closed' or 'Open' accordingly.

3.5 Environments

3.5.1. Environment Details

The details of the environment, where the testing will be conducted, are as follows:

#	Test Type	Environment
1	Smoke Testing	All
2	Functional System Testing	<Development/QA/Production/UAT/>

#	Test Type	Environment
3	Integration Testing	<Development/QA/Production/UAT/>
4	Regression Testing	<Development/QA/Production/UAT/>

Table 7: Environment Details

4. Resource

4.1 Roles & Responsibilities

The following table describes the Roles / Responsibilities, as regards to the testing activities, for the proposed <Application Name> system:

#	Item	Description
1	**Test Lead**	• Identifies, prioritizes, and implements testing activity • Responsibilities: - Overall review of system design , process - Create Test strategy and Test specification - Generate test plan - Generate test model - Test Execution - Report to upper management - Asses Project health - Mitigate Risks
2	**Test System Administrator**	• Ensures test environment and assets are managed and maintained. • Responsibilities: - Administer Test Management System - Install and Manage access to Test Systems
3	**Database Administrator, Database Manager**	• Ensures Test Data (database) environment and assets are managed and maintained • Responsibilities: - Administer test data (database)
4	**Tool Admin**	• Handles any issues related to Tools

Table 8: Roles/Responsibilities

5. Project Milestones

5.1 High Level Test Schedule

<As per project Charter>

5.2 Project Build / Release Plan

The Configuration Manager will be responsible for coordinating the migration of each build to each of the environments and ensuring that all documentation for each build is complete and stored in the appropriate directories. The configuration manager will also coordinate with external interfacing systems to reduce the risk of any downtime resulting from code incompatibility introduced as new builds are implemented.

5.3 High Level Build Schedule

<As per project Charter>

6. Deliverables

The list of deliverables is as follows:

- Test Strategy and Test Plans
    - Strategy document outlining the high level approach for testing, resources, timeline, environment etc., along with detailed Test plans for functional, unit and end to end process testing
- Test Cases
    - Document containing the set of conditions or variables under which a tester will examine all aspects including inputs and outputs of a system along with a detailed description of the steps that should be taken, the results that should be achieved, and other elements that should be identified
- Test Execution Logs
    - Document detailing the tests performed as per the Test Cases

7. Communication

The following is the proposed communication protocol for the proposed <Application Name> system testing:

- Meeting between test participants and stakeholder members associated with Functional Test to review test results and discuss next steps.
- All operational issues will be tracked by the functional test team. These would also be documented as part of the final test report.
- All functional changes for e.g. a change in a query that affect functionality will be validated and approved by the application development team.

- The changes that are made will be communicated to be verified and used as a checklist for functional regression testing.
- Weekly meeting to determine requirements fulfilment for tests planned in the following week.

8. Assumptions

The functional testing strategy for the proposed <Application Name> system has been created and documented based on the following assumptions:

#	Item	Assumptions
1	**Application**	• Each build should be made available to the QA at least one day prior to Test Execution as scheduled in plan, for one round of smoke testing before the final run of test execution • Every build should escort by a Build release plan containing the build contains in details
2	**Data and Database**	• The input data to be provided by the respective application teams • The database is equivalent to production in terms of data volume, hardware and software configuration • Application team to help functional test team write test cases for validating the correct execution of a given scenario and for profiling
3	**Schedule**	• Schedule to be altered by functional Test Lead in concurrence with the Project Plan

Table 9: Assumptions

9. Risk & Mitigation

The following are the typical risks and the mitigation from the proposed <Application Name> system functional testing perspective:

#	Item	Risks	Severity	Mitigation
1	**Application, Database, Infrastructure**	Ambiguous Requirement	High	Test Team members to discuss with the Development Team to understand the requirement and application whereabouts
		Inadequate mapping of Business to Functional	High	Test Team members to discuss with the Development Team to understand the requirement and application whereabouts
		Changes incorporated even after Requirement freezing	Medium	Impact of such functionalities to be kept out of scope
		Typical Data dependency	High	Typical data to be supplied prior test execution start

#	Item	Risks	Severity	Mitigation
		Environmental dependency	Medium	Version controlling need to be strictly maintained and monitored to prevent any violation
2	**Project Risks**	Test Failures	Medium	Use Friday nights and weekends to make up
		Test pre-requisites not ready	Medium	Execute other non-dependent test cases
		System Crashes during test	Medium	Action for Immediate restore and recovery activity
		Re-Test due to Reports not available	High	Some Key tests to be repeated to assess impact of reports

Table 10: Risk & Mitigation

10. Glossary of Terms

#	Term	Description
1	**Bug**	See Defect.
2	**Conversion Testing**	Confirms the accuracy of the conversion procedures needed to initially load the data into the system. Also validates the usage of the data during day-to-day production activity. Performed during the System test level.
3	**Defect/Bug**	The deviation of an actual result from the expected result during the application testing. A flaw in the software with potential to cause a failure which is raised by a tester and is meant for a developer to fix. If the defect cannot be resolved only by developers, then the item would be considered an issue. Defect Severities: **Severity 1: Critical** - Catastrophic defect that causes total failure of the software or unrecoverable data loss. There is no work around. In general, a severity 1 defect would prevent the product from being released. Examples: defects that cause the system to crash, massive performance degradation, data corruption, data loss, security violation or completely disrupt service. **Severity 2: High** - Defect results in severely impaired functionality. A work around may exist but its use is unsatisfactory. In general, you would not release the product with such a defect. Examples: operational error, data integrity, some performance degradation, loss of functionality (no workaround) **Severity 3: Medium** - Defect causes failure of non-critical aspects of the system. There is a reasonably satisfactory work around. The product may be released if the defect is documented, but the existence of the defect may cause customer dissatisfaction.

#	Term	Description
		Example: a non Client Financial Report is not recognizing an option correctly, but if a filter is set, the report can be generated with the proper output. **Severity 4: Low** - Defect of minor significance. A work around exists or, if not, the impairment is slight. Generally, the product could be released and most customers would be unaware of the defect's existence or only slightly dissatisfied. Example: A button or button set is slightly off center on a data screen, or the problem is purely cosmetic and not easily recognizable, minor problem, misspelling, UI layout, rare occurrence.
4	**End-to-End Testing**	Additional interface testing from the beginning to the end of a process including all upstream and downstream impacted systems that receive data, whether direct or indirect from the primary system. Performed during the System test level.
5	**Entry Criteria**	Metrics specifying the condition that must be met in order to begin testing at the next stage or level.
6	**Environment**	The collection of hardware, software, data and personnel that comprise a level of test.
7	**Exit Criteria**	Metrics specifying the conditions that must be met in order to promote a software product to the next stage or level.
8	**Functional Testing**	See System Testing.
9	**Integration Testing**	The objective of Integration Testing is to test the interaction of related data interface components in order to confirm that these components function properly when integrated together. This serves to identify and resolve major interface defects before starting System Testing. Typically conducted by the development team.
10	**Interfacing System**	Downstream or upstream system that may require change due to the primary system.
11	**Issue**	An issue can be any discrepancies, deficiencies or other abnormal system behaviour noted during the entire phase of the software development life cycle. Issues are not limited to hardware, database, network, and integration issues alone. Issues are often process related as well. An issue is anything that hampers the normal workflow during SDLC. Items are considered to be issues if they involve more than a developer to resolve. An issue can be raised by anyone related with the concerned application during any phase of the SLDC
12	**Primary System**	Core application being developed or modified.
13	**QA**	Quality Assurance

Table 11: Glossary of Terms