[Insert image and/or partner logos as relevant]

OPENLMIS QUALITY ASSURANCE PLAN

[INSERT NAME OF COUNTRY, DATE]

Table of Contents

ACRONYMS & ABBREVIATIONS

eLMIS	Electronic logistics management information system
MOH	Ministry of Health
OpenLMIS	Open Logistics Management Information System
QA	Quality Assurance
UAT	User Acceptance Testing
UI	User Interface

1. INTRODUCTION

The purpose of this document is to outline the test plan and strategy for testing the OpenLMIS application during the development and release process.

1.1 BACKGROUND

[insert relevant background information here] such as:
- program funders and partners working on the implementation
-program efforts that have taken place so far
-primary objectives of the program
-implementation timeline

1.2 OBJECTIVES

The OpenLMIS quality assurance (QA) plan outlines the testing strategies, tools, and processes used to ensure complete validation that OpenLMIS configuration and development meets the business and software requirements for this program with several objectives:

- Ensure the system is stable and performant
- Ensure configuration of master and reference data is accurate and as intended
- Confirm the functionality of country-specific changes and customizations
- Avoid/minimize regressions
- Identify and resolve issues prior to release

The implementation team will achieve these quality objectives by:

- Verifying that software requirements are complete and accurate
- Performing detailed test planning
- Identifying standards and test procedures that will be used in the project
- Identifying responsibilities for testing tasks within the project
- Preparing and documenting test data, test scenarios and test cases
- Regression testing to validate that unchanged functionality has not been affected by new code
- Managing the defect tracking process
- Providing test metrics / test summary reports
- Ensuring that the application is certified for launch in the production environment
- Defining requirements for Go / No Go

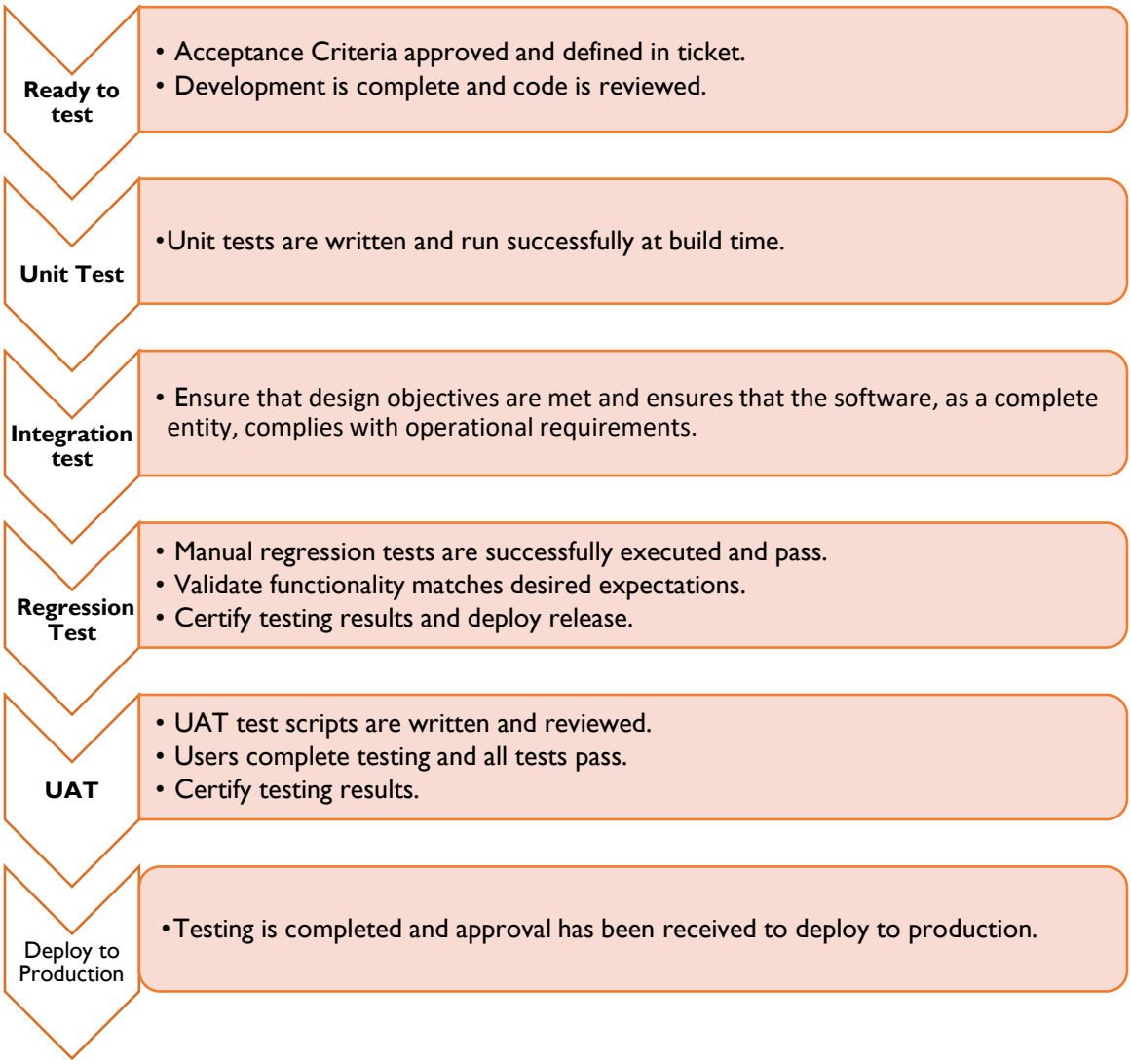
2. APPROACH

The approach for testing is divided into five areas: unit testing, integration testing, performance testing, regression testing and user acceptance testing. During each sprint cycle the QA will perform manual testing using Zephyr test cases and test cycles (see section 3 for further detail on testing tools). Manual Regression testing is performed every other sprint as needed and as part of the release process. Automated testing including unit testing and integration testing are written and executed within the development process.

2.1 TESTING STAGES

Testing is completed in stages as shown in Figure 1. During the development process, the cycle includes test case creation and ends with regression testing for a release. Once all development has been completed the final steps are completed in UAT to certify a release for deployment to production.

Figure 1. Testing Stages



2.2 UNIT TESTING

Unit testing involves the testing of individual components or functions (units) of the software. Developers create unit tests to verify new functionality as a standard part of the development process. The unit test coverage is expected to reach 80-100% and unit tests are included as part of the acceptance criteria in development tickets.

2.3 INTEGRATION TESTING

The purpose of integration testing is to ensure that design objectives are met and ensures that the software, as a complete entity, complies with operational requirements. Developers add integration tests as new cross-service functionality is introduced during the development process. Within the context of OpenLMIS' development, integration tests aim to confirm that independently developed units of the system work together properly. For example, a test might

verify whether a non-responsive call made from OpenLMIS' stock-management service to its authentication service is handled gracefully by the former. These integration tests are defined in code and automatically run on a central Jenkins server dedicated to this implementation as part of an integration or deployment job.

2.4 REGRESSION TESTING

The purpose of regression testing is to ensure that any new changes implemented within a sprint cycle have not negatively impacted previously tested functionality. Two approaches will be used for regression testing: previously existing integration tests and manual regression testing. The QA tester will conduct manual testing for functional regressions during the release process. When new functionality is added the QA tester must determine which tests to include as regression tests. Test cases are considered for regression when the changes impact the core functionality or feature of the software, cover complex test scenarios, or are test cases that frequently fail or identify bugs. In Zephyr, the QA tester must label regression test cases with "Regression" and assign them to each release's test cycle.

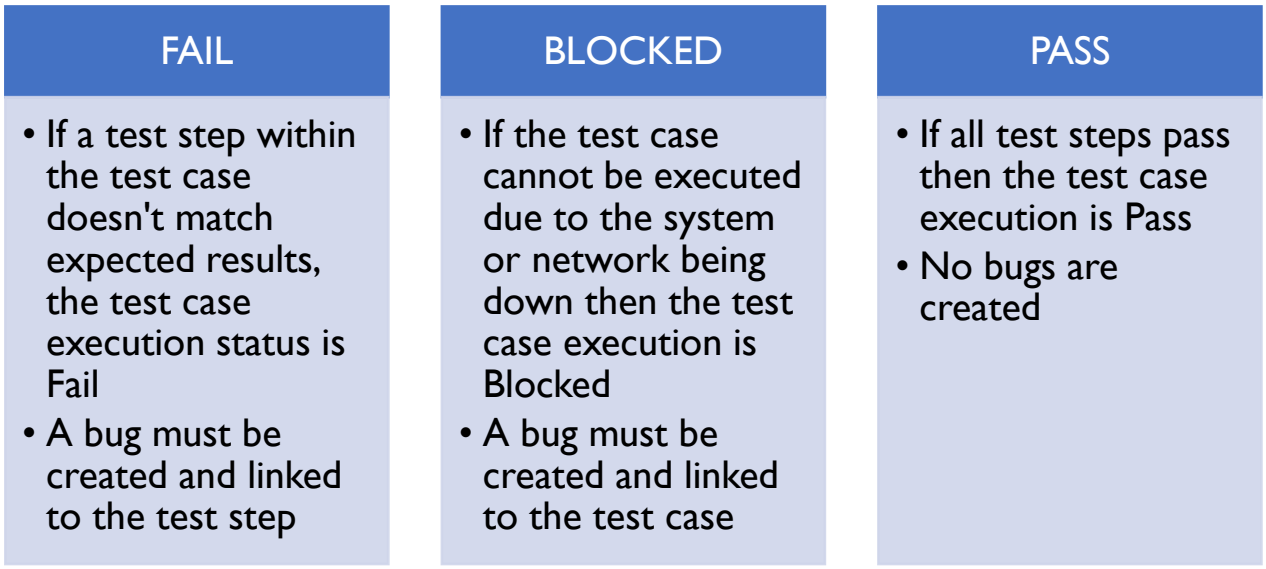
2.5 USER ACCEPTANCE TESTING

The purpose of User Acceptance Testing (UAT) is to validate that the development completed by the project team satisfies the acceptance criteria of the customer. UAT ensures that customers' requirements have been met and all components are included in the deployed release.

2.6 MANUAL TESTING PASS/FAIL STATUS

When manual tests are executed, a Zephyr test case status is assigned to indicate the results of the testing. Test cases can have many statuses during the test process, with the most important statuses for release testing being Pass, Fail, or Blocked. A test case is assigned the "Pass" status when all steps are successfully executed and the results match the expected results described in the test case. A test case has a status of "Fail" when the test case has one or more steps that do not match the expected results. When a test step is marked as "Fail" during the testing process, the QA tester will log a bug to report the failure and expected results or, in some cases, link the test step to an existing bug, if the issue causing the failure is already logged. If for some reason the step cannot be executed, such as the system is down and the user cannot start executing a test, then the status would be "Blocked". These statuses are assigned by the tester in all Zephyr manual test cases executed for the release process and UAT.

Figure 2. Test Case Execution Status



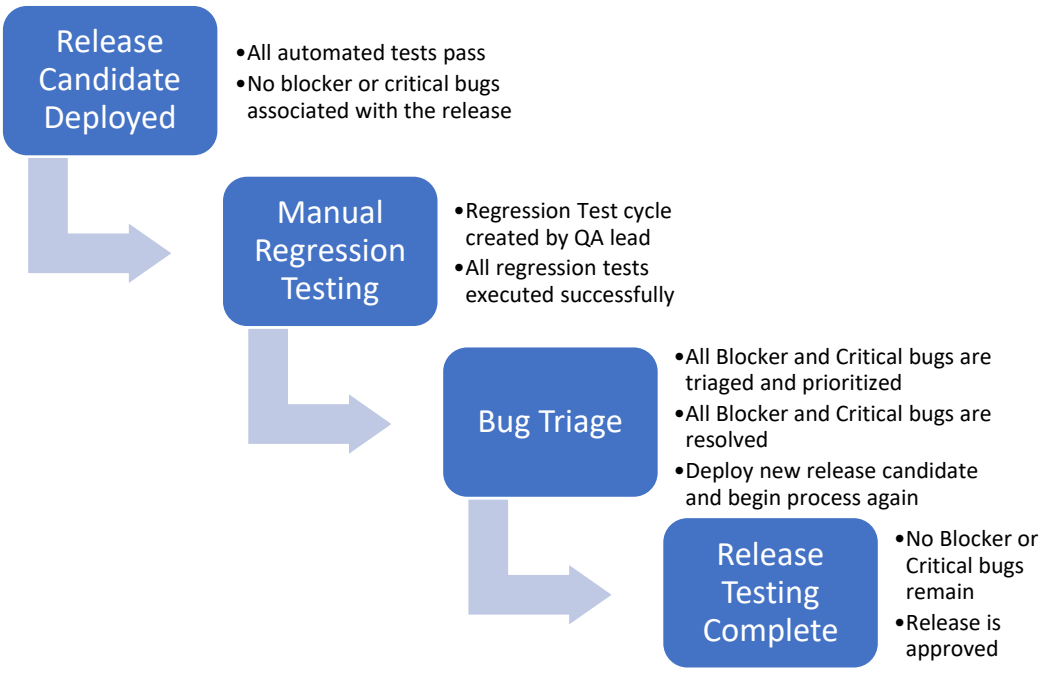
2.6.1 Suspension criteria

Manual testing during the release process will be suspended for a component if one or more bugs with a priority of Blocker or Critical are found. The team have created a Ticket Prioritization page on the OpenLMIS wiki for detail on how priority is assigned. The project team and other stakeholders will review the priority and status of all bugs daily to determine whether manual testing should be suspended. A release candidate should not be released with unresolved Blocker or Critical bugs unless reviewed and approved for release by the project team and other stakeholders.

2.6.2 Resumption Criteria

Manual release testing will only resume when all blocker or critical bugs have been fixed and tested by the QA team. If the blocker and critical bug fixes include changes to multiple components or many new changes were introduced then a full manual regression test cycle is required before releasing. If the bug fixes are focused on a specific service then it is acceptable to resume testing for only the service. To resume manual testing, a new release test cycle for the updated release candidate is then created by the QA lead, and all necessary test cases are added to the cycle. The process will repeat until no blocker or critical bugs are found and all tests pass.

Figure 3. Testing Resumption Process



2.6.3 Approval Criteria

A release is certified to deploy to production once all automated tests pass, all manual tests pass, and signoff from QA lead and program manager are received.

2.7 TEST DELIVERABLES

Test deliverables include:

- Test data/demo data
- Test cases (test scripts/scenarios)
- Test Cycle Execution reports for each development cycle
- Bug report for each release test cycle

2.8 TESTING TASKS

Testing tasks for the QA lead or QA tester include:

- Updates to existing test cases or creating new test cases as new functionality is introduced
- Routine auditing of test cases to ensure test cases contain the appropriate test steps and expected results
- Selection of test cases for a release cycle and executing test cases for complete coverage
- Reporting bugs and communicating bug status during a release
- Communication the status of test cycle execution

Testing tasks for the development team include:

- Creation of unit and integration tests as required during development

- Creation of test data

2.9 RESPONSIBILITIES

Testing responsibilities are outlined in Table 1.

Table 1. Testing Roles & Responsibilities

Individual / Team	Responsibilities
QA Team	• The QA lead is responsible for creating the test plan and signing off on each test cycle execution. • The QA tester is responsible for writing and executing test cases within the development process, as well as creating and executing the test cycle, and reporting bugs. • The QA tester will generate testing reports as defined in the test deliverables.
Development team	The development team is responsible for writing and maintaining automated tests, and resolving any bugs identified by the project team that must be fixed. Developers may also assist in testing when additional resources are needed.
Users	Users are responsible for participating in the UAT and reporting bugs as they are identified.
Project team	• The project team which includes the program manager, development lead, and QA lead are responsible for prioritizing development tickets assigned to a sprint and prioritizing bugs.

2.10 SCHEDULE

Table 2 outlines the schedule for each type of testing that occurs throughout the testing process (see sections 2.1 through 2.6 for details). The scheduled outlined here is a generalized schedule, based on best practices and expected testing needs. The specific schedule and duration of testing may vary based on project timelines and/or quantity or degree of changes included in a release (i.e. a greater number of changes and/or significant changes may require more testing that fewer, simpler, changes). A specific test plan and schedule should be defined for each release.

Table 2. Test Schedule

Testing type	Timeline	Dependencies
Unit tests	• Ongoing as part of development	n/a

Integration tests	• Ongoing as part of development	n/a
Performance tests	• Included in regression testing	• Set up of performance testing environment • Test data
Regression tests	• Major release – 2 weeks • Minor release – 3 days	• Updated test cases • Test data
UAT	• 4 weeks prior to release	• UAT testing scenarios • Test data • Test credentials & materials

3. TESTING ENVIRONMENT & TOOLS

3.1 TESTING ENVIRONMENT

The test environment that will be used for the project is: http://test.ao.openlmis.org/. The test environment hardware and most configuration (but not necessarily software) must match the production environment while maintaining the latest code. Test users are created as needed during the development process to ensure full coverage for each test case. The users list is maintained on the QA Users page of the project Confluence space. The test environment will be refreshed before every major release or as directed by the QA team as needed. Any person involved in testing will need access to the JIRA project for entering bugs and Zephyr (see section 3.2) for executing the test cases.

3.1.1 Hardware, Software, Network

This section includes a list of types of supported devices/browsers prioritized for manual testing.

Past versions of OpenLMIS have officially supported Firefox. OpenLMIS version 3.x, prioritizes the support of Chrome because of global trends, its developer tools and auto-updating nature.

For the testing of OpenLMIS, our browser version priorities are:

- Chrome: The latest version;
- Firefox: The latest version.

The next most widely-used browser is Internet Explorer but we will not conduct specific testing or bug fixes for Internet Explorer compatibility in OpenLMIS, as this is not an OpenLMIS Core supported browser.

The operating systems on which we should test are:

- Windows 7
- Windows 10

For development and testing environments our specifications are:

OpenLMIS Application Server

- CPU: Dual-core 2.4 GHz Intel Xeon® E5-2676 v3 (Haswell).
- RAM: 8 GB.
- Hard drive capacity: 86 GB, (54 GB currently used).

Database Server

- CPU: Dual-core 2.5GHz Intel Xeon.
- RAM: 8 GB.
- Hard drive capacity: 100 GB, (27 GB currently used).

OpenLMIS Reporting Server

- Preferred OS: Ubuntu 16.04.2 LTS or newer
- CPU: Quad-core 2.5GHz
- RAM: 16 GB
- Hard drive capacity: 100 GB

3.2 TESTING TOOLS

The Zephyr extension for JIRA will be used to track all testing tasks and provide reporting of testing status. The process for the creation of test cases, test cycles, and reporting will follow the OpenLMIS Core team's documented standards and best practices (see details in the OpenLMIS Testing Guide). Links to key pages for a specific country implementation Zephyr and JIRA pages includes some of the following examples:

- Test Summary
- Test Cases
- Test Cycle
- Test metrics
- OpenLMIS-Angola backlog

4. RISKS, DEPENDENCIES, & ASSUMPTIONS

Risks, dependencies, and assumptions that may affect the occurrence, quality, and completion of the testing described in this document are identified in the following sections.

4.1 RISKS

- Updates from the OpenLMIS Core team may affect country-specific development and customization if changes are not contributed back to the Core product
- Limited French/Spanish/Portuguese language skills will limit the effectiveness of testing to validate translations (this depends on the country where they deployment is taking place)

- Variations or changes in supply chain processes the system supports may reduce the comprehensiveness of testing
- If requirements are changed or added it could prevent of delay the creation/updating of test cases and completion of testing processes

4.2 ASSUMPTIONS & DEPENDENCIES

- Realistic, appropriate, and sufficiently complete demo/testing data for manual test case and UAT execution
- Realistic QA user credentials
- Validation of Portuguese localization depends upon stakeholders and end-users
- UAT is dependent upon user availability to complete tests
- Regression and release testing are dependent upon the availability of the system, hardware, and connectivity
- Creation of unit and integration tests during development is required to complete all types of testing

5. ADDITIONAL TESTING GUIDELINES

In addition to the testing stages described in section 2 other testing, such as performance testing, should be conducted to support the overall maintenance of the system. Additional testing does not necessarily need to be completed with every release.

5.1 PERFORMANCE TESTING

The purpose of performance testing is to establish that OpenLMIS's configuration and any changes made maintain, at a minimum, the last-mile performance characteristics provided in OpenLMIS v3.5.

Performance testing is not required for every release but must be routinely scheduled as development is completed to prevent a decline in overall system performance. Examples of when to conduct performance testing are; as extra functionality is added, a large number of products are added, increased number of users in the system, or a general increased use of the application. Additionally, any development completed with the specific aim to improve performance, testing must be completed to verify that the changes accomplish the desired improvements.