Information

Name:	Project frml24113
Description:	Description of frml24113.
Version:	3.0
Date Created:	2021-12-13T21:02:48.903
Git Commit:	1e84b5100e09d9b6c5ea1b6c2ccee8957391beec
Git Tag:	ods-generated-v3.0-3.0-0b11-D
Git URL:	https://bitbucket/scm/ofi2004/ofi2004-release.git
OpenShift Cluster API URL:	https://openshift-sample
Created by Jenkins Job Name:	ofi2004-cd/ofi2004-cd-release-master
Created by Jenkins Build Number:	666

Combined Functional and Requirements Testing Plan for 'Project frml24113'

TABLE OF CONTENTS

1 PURPOSE

This document contains the integration and acceptance test as part of functional and requirements testing plan for Project frml24113. Details on the testing strategy and test coverage are defined. Integration testing is based on the functional and configuration specification when applicable, whereas Acceptance testing is based on the user requirements (functional and/or non-functional) of the system. The testing processes, details of test execution, review of test results, and resolution of failures are defined as well.

2 SCOPE

3 ROLES AND RESPONSIBILITIES

Role	Responsibility
Jenkins (Technical Role)	To run the verification of the system, execute tests if applicable and collect all the needed information.
Test Administrator	In case of full automation (and no further testcases) - N/A, otherwise Test Administrators will supervise the Administrator execution of the test by the Testers and will review the test cases.
Tester	In case of full automation (and no further testcases) - N/A, otherwise Testers will execute the test cases and document the results.
Developer	Writes tests.

4 LEVELS OF TESTING

The Testing Approach and Strategy is adapted to the Agile Development Methodologies applied in Platforms. This means that the former LeVA Development, Functional and Requirements Testing will be also covered but grouped in a different classification: Unit Testing, Installation Testing, Integration Testing and Acceptance Testing. Unit testing is performed during development by the development engineers and documented in the Development Test Plan (C-DTP) and Report (C-DTR).

Installation Testing is aimed at checking the successful installation and configuration, as well as updating or uninstalling the software. This level of testing is usually executed automatically and in Platforms it is part of the Installation Test Plan (C-IVP) and Report (C-IVR).

4.1 INTEGRATION TESTING

The objective of the Integration Testing level is to verify whether the combined Units work well together as a group. Integration testing is aimed at detecting the flaws in the interactions between the Units within a module, micro-services and/or systems.

In Platforms Integration Testing is part of the Combined Functional/Requirements Test Plan (C-CFTP) and Report (C-CFTR).

4.2 ACCEPTANCE TESTING

This is the last stage of the testing process, where the product is verified against the end user requirements (can be functional or non-functional ones) and for accuracy. Successfully performed acceptance testing is a prerequisite for the product release. This testing level focuses on overall system quality, for example, from content and UI (functional) to performance or security issues (non-functional).

Within an agile approach the Acceptance Criteria are well-defined upfront.

In Platforms Acceptance Testing is part of the Combined Functional/Requirements Test Plan (C-CFTP) and Report (C-CFTR).

As enunciated before, requirements and acceptant criteria can be functional and/or non-functional so Acceptance Testing can be split in two main groups: Functional Testing and Non-Functional Testing.

4.2.1 Functional Testing

Functional testing is a type of software testing in which the system is tested against the functional (user) requirements and specifications. Functional testing ensures that the (user) requirements or specifications are properly satisfied by the application. This type of testing is particularly concerned with the result of processing. It focuses on simulation of actual system usage but does not develop any system structure assumptions.

It is basically defined as a type of testing which verifies that each function of the software application works in conformance with the requirement and specification. This testing is not concerned about the source code of the application. Each functionality of the software

application is tested by providing appropriate test input, expecting the output and comparing the actual output with the expected output.

Some examples of functional testing types are: Unit Testing, Smoke Testing, Integration Testing, System Testing, Exploratory Testing, etc.

4.2.2 Non-Functional Testing

Non-functional testing is a type of software testing that is performed to verify the Non-Functional requirements of the application or system. It verifies whether the behavior of the system is as per the requirement or not. It tests all the aspects which are not tested in Functional testing.

Non-functional testing is defined as a type of software testing to check non-functional aspects of a software application. It is designed to test the readiness of a system as per non-functional parameters which are never addressed by Functional testing. Non-functional testing is as important as Functional testing.

Some examples of functional testing types are: Performance Testing, Load Testing Stress Testing, Security Testing, Scalability Testing, Compatibility Testing, Usability Testing, etc.

5 OPERATIONAL QUALIFICATION ACTIVITIES AND TRAINING

Before test execution the testers will receive formal training on:
* the testing process, including good documentation practice
* a basic end user training for System name
* an overview of the business processes supported by System name

The training will be documented and copies of the training records will be attached to the testing documentation package.

5.1 TEST PROCEDURE 1: VERIFICATION OF OPERATIONAL DOCUMENTATION

As part of the Integration Testing the following documentation will be verified for all relevant subjects listed in the Qualification Plan.

5.1.1 Test Procedure 1.1: SOPs and Working Instructions

Verify that approved SOPs and Working Instructions addressing and regarding the production environment, are in place. They must be approved and effective prior to release for production use.

5.1.2 Test Procedure 1.2: Manuals and other System Documentation

Verify that appropriate manuals and other system documentation exist for use in operating, maintaining, configuring, and/or troubleshooting of the system.

5.2 TEST PROCEDURE 2: VERIFICATION OF TRAINING DOCUMENTATION

List the applicable procedures in which the Integration Testing participants must be trained before executing their portions of the Functional Testing Plan. Describe when and how the training will take place.

Before Integration Testing can start, the Integration Testing participants have to be trained in
* the testing process
* the use of the <insert system name> system

Verify that an approved training plan exists, if justified, for all personnel involved in operating and maintaining the Infrastructure System.

6 INTEGRATION TESTING

6.1 PURPOSE OF INTEGRATION TESTING

The purpose of the integration testing is to verify the functional, non-functional and reliability between the modules that are integrated and work within the specifications as defined in the functional and/or configuration specifications.

6.2 SCOPE OF INTEGRATION TESTING

The System name is commercial off-the-shelf software which was configured according to the configuration specification. Since some user requirements could not be fulfilled via configuration and therefore have been implemented by in-house developed customizations.
* Integration testing will be performed to verify correct functionality of all customizations, i.e. functional specifications will be 100% covered by functional test cases.
* Configuration specification will not be tested on an integration level, but only implicitly in acceptance test cases.
* Standard product functionality will not be tested in integration testing since appropriate testing has already performed by the vendor. This has been verified in a vendor audit.

7 ACCEPTANCE TESTING

7.1 FUNCTIONAL TESTING

7.1.1 Purpose of Functional Testing

The purpose of the combined functional/requirements testing is to confirm that the computerized system is capable of performing or controlling the activities of the processes as intended according to the user requirements in a reproducible and effective way, while operating in its specified operating environment.

7.1.2 Scope of Functional Testing

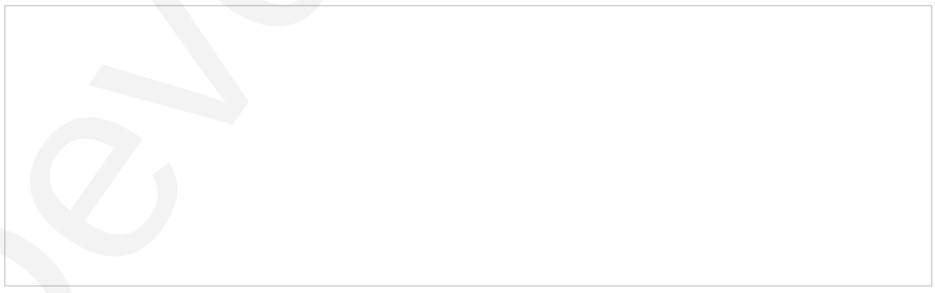
The System name is commercial off-the-shelf software which was configured according to the configuration specification. Since some user requirements could not be fulfilled via configuration and therefore have been implemented by in-house developed customizations.
* Functional testing will be performed to verify correct functionality of all customizations, i.e. functional specifications will be 100% covered by functional test cases.
* Configuration specification will not be tested on a functional level, but only implicitly in acceptance test cases.
* Standard product functionality will not be tested in functional testing since appropriate testing has already performed by the vendor. This has been verified in a vendor audit.

7.2 NON-FUNCTIONAL TESTING

7.2.1 Purpose of Non-Functional Testing

The purpose of Non-Functional testing is to check non-functional aspects (performance, usability, reliability, etc) of a software application. It is designed to test the readiness of a system as per non-functional parameters which are never addressed by functional testing.

7.2.2 Scope of Non-Functional Testing



8 TEST STRUCTURE AND EXECUTION

8.1 TEST CASES

Test cases are based on the user requirements, functional specifications and the business processes which are supported by the system. Test cases shall contain test data and expected outcomes against which observed outcomes may be compared. If test cases must be executed in a specific order this has to be defined.

The level of testing for the individual test cases will be defined based on the risk priority as determined in the risk assessment.

Risk Priority	Level of Testing
3	No testing required
2	Testing of functionality or requirements without challenge/boundary tests
1	Full test of requirement or functionality including challenge/boundary tests (e.g. invalid, borderline or missing input values, out-of-order execution of functions, disconnected interfaces etc.)

8.2 TEST EXECUTION

Test results shall be recorded in a way that independent reviewer can compare the documented acceptance criteria against the (written or captured) test evidence and determine whether the test results meet these criteria.

In the case that the test execution is fully automated:

Jenkins (Technical Role) shall:

- execute the code base test cases
- record the test results and evidence after the execution and include them in the XUnit file following Good Documentation Practices
- mark the test cases as a "Fail" or a "Pass"
- stop the test execution if one of the test cases has failed
- report back the test execution results to the Test Management Tool

As the execution is fully automated, as included in the section 3 "Roles and Responsibilities" the Tester and Test Administrator does not apply.

In the case that the test execution is not fully automated:

Testers shall:

- execute test cases
- record actual test results (e.g. "as per expected result" is not an actual result) and supporting evidence immediately and accurately following Good Documentation Practices
- pass or fail all test cases
- provide comments for all failed test cases
- sign and date each test in spaces provided after test execution
- label any test output or evidence (e.g., screenshots, printouts and any additional pages) with test case number and test step number. Sign and date the output. If pages have successive page numbers signing and dating the first or last page is sufficient.
- if any deviations from the test are encountered, follow the Test Case Failure and Problem Resolution (see section 10)

If a test case is executed by more than one person (tester), it is required that each tester signs (signature or initials and date) each test step for traceability purpose.

The Test Administrators shall:

- review the executed test cases and its attachments for completeness and correctness and sign for the review on the signature page
- for failed test cases add comments (including a final pass/fail evaluation and problem resolution) and sign and date the test case

Test execution and test result review must be independent, i.e. for any individual test case the Tester and the Test Administrator must be different individuals.

The training records of all testers should be verified prior to initiating testing.

9 TRACEABILITY MATRIX

The test coverage is captured in the traceability matrix which maps user requirements and functional specifications to test cases. The traceability matrix will be updated before test execution to include all functional and requirements test cases.

10 VALIDATION ENVIRONMENT

Description of the Environment of the system that will be used for test execution.

11 TEST CASE FAILURE AND PROBLEM RESOLUTION

A test case shall be failed if the observed outcome of a test differs in any way from the expected outcome identified in the test case data. If any step of a test case fails and cannot be resolved, then the entire test case fails. There may be many reasons for failure, some of which may not mean that the system itself has a flaw. It is, however, the Tester's responsibility to fail the test case and indicate in the comments box the failure or problem that occurred, including the test steps related to this discrepancy.

Reasons for test case failure may include:

- a bug in the system
- a failure related to the operating environment
- a mistake in the test case instructions or data
- a tester's error

Upon failing a test case, the Tester shall always contact the Test Administrator immediately to review the problem. The Test Administrator shall decide how to proceed, since test cases may build upon each other and a failure may cascade through several cases.

The Test Administrator will also record all discrepancies that occur during the test execution in a designated discrepancy log. The Test Administrator is responsible for determining failure resolutions and whether a failure represents an unacceptable flaw in the system. The Test Administrator will document the result of this determination in the discrepancy log.

The final evaluation of remaining risks and unresolved critical failures will be assessed in the validation summary report.

12 INTEGRATION / ACCEPTANCE TESTING DOCUMENTATION

The following documents will be created as a result of the execution of integration and acceptance testing as part of the functional and requirements testing:

- Traceability as defined in section 9
- All executed integration test cases
- All executed acceptance test cases
- Discrepancy log

13 DEFINITIONS AND ABBREVIATIONS

13.1 DEFINITIONS

Term	Definition
Jenkins	Build engine supplied by cloudbees - part of OpenDevStack (BI-IT-DEVSTACK)
xUnit	Unit testing framework, aggregaults across multiple languages

13.2 ABBREVIATIONS

Abbreviation	Meaning
ODS	OpenDevStack
EDP	Enterprise Development Platform

14 REFERENCE DOCUMENTS

- Risk Assessment (version BI-IT-DEVSTACK / 5-WIP)

15 DOCUMENT HISTORY

Version	Date	Author	Change Reference
1	See Summary of electronic document or signature page of printout.		Initial document version.
2	See Summary of electronic document or signature page of printout.		Modifications for project version '2.0'.
3	See Summary of electronic document or signature page of printout.		Modifications for project version '3.0'.
4	See Summary of electronic document or signature page of printout.		Modifications for project version '4.0'.
5	See Summary of electronic document or signature page of printout.		Modifications for project version '4.0'. This document version invalidates the changes done in document version '4'.

The following table provides extra history of the document.

Version	Date	Author	Reference
	See summary of electronic document or signature page of printout.		