Version Draft

Affinity Solutions

June 18, 2020

Version History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Document Version** | **Document Revision History** | **Author/Reviser** |
| 2020/06/18 | Draft | CaMS Test Strategy document | Anneline van der Merwe |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Document Authorisation

|  |  |
| --- | --- |
| Approved By (Sign) |  |
| Name (Capacity) | Julien Van der Pluym (CTO) |
| Date Approved |  |
| Approved By (Sign) |  |
| Name (Capacity) |  |
| Date Approved |  |
| Approved By (Sign) |  |
| Name (Capacity) |  |
| Date Approved |  |
| Approved By (Sign) |  |
| Name (Capacity) |  |
| Date Approved |  |

STAKEHOLDERS LIST

|  |  |  |
| --- | --- | --- |
| **Name** | **Title** | **Category of Stakeholder**  **(Refer to Note Below)** |
| Julien Van der Pluym | CTO | A |
| Anneline van der Merwe | Senior Automation Tester | R |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**NOTE**

Categories of level of responsibility as determined in **(Document Name**).

**R** Document Author Name must appear as the author of the document

**A** Accountable Name must appear as an authorizer in the document

**C** Must be consulted Name must appear in the RTS of the document

**I** Must be informed Name must appear in the Distribution List of the document

DISTRIBUTION INSTRUCTIONS

|  |
| --- |
| All stakeholders |
| SharePoint |

REFERENCE DOCUMENTATION

|  |
| --- |
| Cams Architectural Design Vision Document |
|  |

Definitions and Acronyms

|  |  |
| --- | --- |
| **Definitions and Acronyms** | **Description** |
| Black box Testing | Is also a behaviour testing, method in which the internal structure/design/implementation of the item being tested is not known to the tester. |
| Component Test | It is a kind of white box testing. So, component testing looks for bugs and verifies the functioning of the modules/programs which are separately testable. |
| Exploratory Testing | Exploratory Testing is defined as simultaneous learning, test design and test execution. It is an approach to testing that values the tester as an integral part of the test process and shares the same values as the Agile Manifesto: Individuals and interactions over processes and tools. |
| Functional Tests | It is a type of software testing whereby the system is tested against the functional requirements/specifications. Functions (or features) are tested by feeding them input and examining the output. ... Determine the output based on the function's specifications. Execute the test case. |
| Scenario | A usage scenario, or scenario for short, describes a real-world example of how one or more people or organizations interact with a system. They describe the steps, events, and/or actions which occur during the interaction. ... Usage scenarios are applied in several development processes, often in different ways. |
| Story Tests | A user story is a tool used in Agile software development to capture a description of a software feature from an end user perspective. The user story describes the type of user, what they want and why. ... A user story can be considered a starting point to a conversation that establishes the real product requirement. |
| Stress Testing | Testing that verifies the stability & reliability of the system. This test mainly measures the system on its robustness and error handling capabilities under extremely heavy load conditions. Stress Testing is done to make sure that the system would not crash under crunch situations. |
| Test Case | A specific set of test data along with expected results for a particular test objective. |
| Test Coverage | Describes how much of a system has been tested. |
| Test Design | Describes how a feature or function shall be tested. |
| Test Plan | Test Strategy describes the scope, approach, resources and schedule for the testing activities of the project. This includes defining what will be tested, who will perform testing, how testing will be managed, and the associated risks and contingencies. |
| Test Procedure | Describes the steps for executing a set of test cases and analysing their results. |
| Test Script | A test script in software testing is a set of instructions that will be performed on the system under test to test that the system functions as expected. There are various means for executing test scripts. Manual testing. These are more commonly called test cases. |
| Usability Testing | Is a way to see how easy to use something is by testing it with real users. Users are asked to complete tasks, typically while they are being observed by a researcher, to see where they encounter problems and experience confusion. |
| Use Case | Describes a sequence of interactions between a system and an external actor that results in the actor accomplishing a task that provides benefit to someone. An actor is a person or other entity external to the software system being specified who interacts with the system to accomplish tasks. Different actors often correspond to different user classes, or roles, identified from the customer community that will use the product. |
| User Acceptance Testing | Is a phase of software development in which the software is tested in the "real world" by the intended audience |
| White box Testing | Software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester. |

[1. Overview 8](#_Toc43736181)

[2. Introduction to Agile 8](#_Toc43736182)

[3. Guiding standards 8](#_Toc43736183)

[4. Requirements strategy 8](#_Toc43736184)

[5. Quality and Test Objectives 8](#_Toc43736185)

[6. In Scope 9](#_Toc43736186)

[7. Out of Scope 9](#_Toc43736187)

[8. Test Levels 10](#_Toc43736188)

[9. Test design strategy 11](#_Toc43736189)

[10. Test Environment strategy 12](#_Toc43736190)

[11. Test Execution Strategy 12](#_Toc43736191)

[12. Test Data Management Strategy 13](#_Toc43736192)

[13. Test Automation Strategy 13](#_Toc43736193)

[14. Test Management 13](#_Toc43736194)

[15. Risks and assumptions 14](#_Toc43736195)

[16. Defect Management Strategy 14](#_Toc43736196)

[17. Installation/Configuration 14](#_Toc43736197)

# Overview

This Test Strategy describes the integration and system tests that will be conducted on the CaMS (Case Management Software) cloud-based platform. Provides sets of web apps and exposes services for data manipulation. The purpose of assembling the architectural prototype was to test feasibility and performance of the selected architecture. It is critical that all system and subsystem interfaces be tested as well as system stress testing at this early stage. The interfaces between the following subsystems will be tested

# Introduction to Agile

* Iterative approach to software development
* Highly collaborative
* Guiding principles for example continuous design improvement.

# Guiding standards

* Shared responsibility
* Data management
* Test management
* Test automation

# Requirements strategy

* [Active Stakeholder Participation](http://www.ambysoft.com/essays/agileTesting.html#ActiveStakeholderParticipation)
* [Functional Requirements Management](http://www.ambysoft.com/essays/agileTesting.html#FunctionalRequirementsManagement)
* [Initial Requirements Envisioning](http://www.ambysoft.com/essays/agileTesting.html#InitialEnvisioning)
* [Iteration Modelling](http://www.ambysoft.com/essays/agileTesting.html#IterationModeling)
* [Just in Time (JIT) Model Storming](http://www.ambysoft.com/essays/agileTesting.html#ModelStorming)
* [Non-Functional Requirements Management](http://www.ambysoft.com/essays/agileTesting.html#NonFunctionalRequirements)
* [Who is Doing This?](http://www.ambysoft.com/essays/agileTesting.html#RequirementsDoingThis)
* [The Implications for Testing](http://www.ambysoft.com/essays/agileTesting.html#ImplicationsForTesting)

# Quality and Test Objectives

* Accuracy (Identify if the test was working 100%) – Add priority
* Integrity of data (secure, variables must be encrypted) – Add priority
* Maintainability (Easy to add features) – Add priority
* Availability (Set Target)
* Inter-operability (To be able to work on Web browsers)

Stress Testing (Test with the max possible users, requests, Error handling, security threats, memory leaks, check the stability of the systems).

System Performance indicators as per CaMS Architectural Design Vision Document:

* System should handle 200 concurrent users
* Page rendering time takes maximum 1 second (for pages that do not render >= 100 records)
* Email/SMS notification time takes maximum 1 minute

# In Scope

* Automated Unit Testing
* Code Analysis (Static and Dynamic)
* Integration Testing
* Automated Feature and functional testing
* Data conversion testing
* System testing
* Automated Security testing
* Environment Testing
* Automated Stress
* Availability testing
* Automated Regression Testing
* Acceptance testing

# Out of Scope

**???????**

# **Test Levels**

Functional Tests

Examples

Story Tests

Prototypes

Simulation

Exploratory Testing

Scenarios

Usability Testing

User Acceptance Testing (UAT)

Alpha/Beta Testing

Unit Tests

Component Test

Stress Testing

Security Testing

Q1

Q2

Q3

Q4

**Business Facing**

**Supporting the Team**

**Technology Facing**

**Critique Product**

**Agile Quadrant 1**

The internal code quality is the main focus in this quadrant, and it consists of test cases which are technology driven and are implemented to support the team, it includes

* Unit Tests
* Component Test

**Agile Quadrant 2**

It contains test cases that are business driven and are implemented to support the team.  This Quadrant focuses on the requirements. The kind of test performed in this phase is:

* Testing of examples of possible scenarios and workflows
* Testing of User experience such as prototypes
* Pair testing

**Agile Quadrant 3**

This quadrant provides feedback to quadrants one and two.  The test cases can be used as the basis to perform automation testing.  In this quadrant, many rounds of iteration reviews are carried out which builds confidence in the product.  The kind of testing done in this quadrant is

* Usability Testing
* Exploratory Testing
* Pair testing with customers
* Collaborative testing
* User acceptance testing

**Agile Quadrant 4**

This quadrant concentrates on the non-functional requirements such as performance, security, stability, etc.  With the help of this quadrant, the application is made to deliver the non-functional qualities and expected value.

* Non-functional tests such as stress testing
* Security testing with respect to authentication and hacking
* Infrastructure testing
* Data migration testing
* Scalability testing

# Test design strategy

* Specification based techniques
  + **Equivalence Partitioning**
  + **Boundary Value Analysis**
  + **Decision Tables**
  + **State Transitioning**
* Structure based techniques
  + Statement testing (White box approach)
  + Decision testing/branch testing (White box approach)
  + Condition testing (True of False – To get a 100% coverage)
  + Multiple condition testing
  + Path testing
* Experience base techniques
  + Error Guessing
  + Checklist Based Testing
  + Exploratory Testing
  + Attack Testing (Interaction with external interfaces, applications, system and external elements

# Test Environment strategy

* QA Environment
  + Unit Testing
  + Functional Testing
  + Acceptance Testing
* Integration Environment
  + Unit Testing
  + Functional Testing
  + Acceptance Testing
  + Static code Analysis
  + Continues Integration Testing
  + Stress Testing
* Staging Environment
  + Exploratory Testing
  + Automated Testing
* Production Environment
  + Production verification Testing

# Test Execution Strategy

* Agile testing must be iterative
* Testers cannot rely on having complete specification
* Testers should be flexible
* They need to be independent and independently empowered in order to be effective
* Tester should be generalizing specialists
* Testers should be prepared to work closely with developers
* Should focus on value added activities
* Focus on What and Not How to test
* Testers should be embedded in the agile team
* Flexible to contribute in any way they can
* Testers must have a wide range of skills with one or more specialties
* There must be shorter feedback cycles
* Need to focus on sufficient and straightforward situations
* Need to focus on exploratory testing
* Specify the meaning of “Done” for example when activities or tasks performed during the system development can be considered complete
* Define when to continue or stop testing before delivering the system to the customer. Specify which evaluation criteria is to be used for example time, coverage and quality and how it will be used.

# Test Data Management Strategy

* Consists of 2 components: Creational strategy and Cleanup Strategy

# Test Automation Strategy

* Use a planned approach
* Increase the quality of test automation code
* Automate stable and high priority test cases first

# Test Management

* Test Management in Agile is self-directed and not responsible to lead the testers or a test team. In this case Jira will be used.

# Risks and assumptions

* Any risks must be logged immediately, and the owner identified should take immediate action.

# Defect Management Strategy

* Defect classification (Critical, High, Medium, or Low)
* Defect life cycle
* Capture information using the bug work item type
* Triage bugs by assigning a priority
* Update bug status throughout the bug lifecycle
* Monitor bug assignments and trends

# Installation/Configuration

|  |  |
| --- | --- |
| Operating System | Windows 10 |
| Testing Tools | Visual Studio 2019  Selenium  C#  JMeter  Azure DevOps |