­Prevent, Mitigate, and Recover (PMR) Insight

Collective Knowledge System (PICK)

Test plan

Version <1.0>

4/16/2020

Document Control

Approval

The Guidance Team and the customer shall approve this document.

Document Change Control

|  |  |
| --- | --- |
| Initial Release: |  |
| Current Release: |  |
| Indicator of Last Page in Document: | \* |
| Date of Last Review: | 4/16/2020 |
| Date of Next Review: | 4/18/2020 |
| Target Date for Next Update: | 4/22/2020 |

Distribution List

This following list of people shall receive a copy of this document every time a new version of this document becomes available:

Guidance Team Members:

Dr. Gates

Dr. Salamah

Dr. Roach

Steven Roach

Jake Lasley

Customer:

Dr. Oscar Perez

Vincent Fonseca

Herandy Denisse Vazquez

Baltazar Santaella

Florencia Larsen

Erick De Nava

Software Team Members:

Elizabeth Barragan

Jose A. Leon Cordero

Yamel E. Hernandez

Manuel Delgado

Change Summary

The following table details changes made between versions of this document

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Modifier | Description |
| 1.0 | 4/16/2020 | Manuel | Creation of Document & Document Control |
| 1.1 | 4/16/2020 | Elizabeth | Added a few database test cases (Sec 3) and Test Suite DB3 (Sec 4) |
| 1.2 | 4/16/20 | Yamel | Added section 1.5, added Ingestion test cases to section 3, & Ingestion Test Suit in Section 4 |
| 1.2 | 4/16/2020 | Manuel | Added Vector Subsys to section 3 as well as intro |

Note: The template presented in this document was taken from:

Donaldson, S., and S. Siegel, *Successful Software Development*. Upper Saddle River, NJ: Prentice Hall, 2001, pp. 321-323.

Note: The template presented in this document was taken from: Donaldson, S., and S. Siegel, *Successful Software Development*. Upper Saddle River, NJ: Prentice Hall, 2001, pp. 321-323 and modified by Humberto Mendoza and Steve Roach.

Supplementary information is from:

Pfleeger, S. *Software Engineering, Theory and Practice*. Upper Saddle River, NJ: Prentice Hall, 1998, p. 365.

Table of Contents

[Document Control ii](#_Toc22915465)

[Approval ii](#_Toc22915466)

[Document Change Control ii](#_Toc22915467)

[Distribution List ii](#_Toc22915468)

[Change Summary ii](#_Toc22915469)

[1. Introduction 1](#_Toc22915470)

[1.1. Purpose 1](#_Toc22915471)

[1.2. Scope 1](#_Toc22915472)

[1.3. System Overview 1](#_Toc22915473)

[1.4. Suspension and Exit Criteria 1](#_Toc22915474)

[1.5. Document Overview 1](#_Toc22915475)

[1.6. References 1](#_Toc22915476)

[2. Test Items and Features 2](#_Toc22915477)

[3. Testing Approach 3](#_Toc22915478)

[4. Test XX 4](#_Toc22915479)

[4.1. Test <<test id>> 4](#_Toc22915480)

[5. User Interface Testing 5](#_Toc22915481)

[6. Test Schedule 6](#_Toc22915482)

[7. Other Sections 7](#_Toc22915483)

[8. Appendix 8](#_Toc22915484)

# Introduction

## Purpose

<< Identify the project and stipulate the test plan purpose by indicating what the document contains (e.g., organizational responsibilities, test approach, test schedule. There are generally four different types of test plans: project test plan that describes the overall strategy for testing; the system test plan that describes the system from the customer’s point of view; integration test plan that describes integration of units and subsystems; unit test plan that describes modules or classes. This section needs to identify which of these this document is.>>

## Scope

<<Specify the project software releases/versions encompassed by the plan. >>

## System Overview

<<Describe the system to be exercised by the testing approach specified in the plan. This overview serves to identify aspects of the system operation that will be the focus of the plan’s testing approach. This should align with the systems overview of other documents in the project.>>

## Suspension and Exit Criteria

<< “suspension criteria” describes when we suspend testing, to be resumed at a later time. For example, if 40% of the test cases fail, or if any of the critical test cases fail. If there are no suspension criteria, indicate that all tests cases will be executed. “Exit criteria” indicates when testing stops. This could be based on run rate (number of test cases run divided by number of test cases specified) or pass rate (number of test cases passed divided by number of test cases run, or test cases passed divided by number of test cases specified). Nominally, we expect to run all of the specified tests. We want the pass rate to be high. We might specify that all critical tests must pass, and 90% of the non-critical must pass. In general, we want this to be high. >>

## Document Overview

<<Describe the remainder of the document.>>

This document outlines how we will implement and test our system. Section 2 will describe the test items and features that will be tested of those items, which will include a table or a class diagram for visualization. Section 3 will discuss how we will approach testing the items and features from section 2 in tabular format. The tables will list the type of test, it’s objective, and the level of criticality of the test. Section 4 will document the tests in a table per test. The table will have the test number, its status, title, approach, who performed the test, and when the test was concluded. Section 5 will discuss the testing of the interactions between the user and the system. Section 6 will list and describe the test schedule in a table, information on it will include a date, who will be performing the test and a short description. Section 7 will list other sections important to the document that have not been discussed. Section 8, the Appendix, will depict and illustrate outputs for analysis.

## References

<<List all the references applicable to the test plan. Generally, this includes project standards, SRS, SDD, and a product assurance plan.>>

[1]

[2]

[3]

# Test Items and Features

In this section, components, classes, functions and features to be tested are defined. Additionally, we define those which are not to be tested.

## Components to be Tested

The following is a list of the components and their functions and features that will be tested:

* Real-World Objects
  + Data transformation
  + Instantiation
  + JSON representation
* Splunk
* Signal Controller
  + Signals handling of all models
* Database
* Ingestion Subsystem
* Graphical User Interface
  + Window frame adjustment
  + Button events
  + Input processing
  + Model representation
  + Error handling
  + Default security settings
* Vector Subsystem
  + Graph
  + Node Table

## Components not to be Tested

<< This section describes the test items (e.g., components, classes, functions or methods) and the features to be tested. It may also list features not to be tested. A class diagram is useful. A table of features is useful. >>

The following is a list of features that will not be tested:

* Version Control
* Undo/Redo action
* Audio transcriber
* OCR
* Networking

# Testing Approach

The section demonstrates in a table format the types of test that will be performed on the system. The table provides a description for types of tests such as but not limited to exercising system functions, approximating operational use, or stressing the system to find out limitations. Each component or subsystem to be tested has a description of the type of test, test case identifier for each objective and its importance level.

Table 1: Database Test Suite

|  |  |  |
| --- | --- | --- |
| **TEST SUITE DATABASE** | | |
| **Description of Test Suite** | **This test suite has test cases for the functionality of the Database subsystem** | |
| **Test Case Identifier** | **Objective** | **Criticality** |
| DB 1 | Have the objects mapped with the DB Document Structure | Critical |
| DB 2 | Validation of the Insert/Delete/Update trigger functionality reflecting the system | Critical |
| DB 3 | Update an object’s attribute with wrong input | Normal |
| DB 4 | The acceptance of a Null object in the DB | Critical |

Table 2: Ingestion Test Suite

|  |  |  |
| --- | --- | --- |
| **TEST SUITE INGESTION** | | |
| **Description of Test Suite** | **This test suite will have appropriate test cases for the functionality of the ingestion subsystem and process** | |
| **Test Case Identifier** | **Objective** | **Criticality** |
| Ing 1 | Test for cleansing for additional spaces, lines, & binary data | Critical |
| Ing 2 | Test parsing in Splunk | Critical |
| Ing 3 | Test for error messages in enforcement action report & error class | Critical |
| Ing 4 | Test ingestion process to make sure log files get flagged for the state that they are in | Critical |

Table 3: Signal Controller Test Suite

|  |  |  |
| --- | --- | --- |
| **TEST SUITE SIGNAL CONTROLLER** | | |
| **Description of Test Suite** | **This test suite defines test cases for probing the functionality of the signal controllers** | |
| **Test Case Identifier** | **Objective** | **Criticality** |
| SC 1 | Test the functionality of vector model change signals | Critical |
| SC 2 | Test the functionality of significant log entry model change signals | Critical |
| SC 3 | Test the functionality of relationship model change signals | Critical |
| SC 4 | Test the functionality of node model change signals | Critical |
| SC 5 | Test the functionality of log file model change signals | Critical |
| SC6 | Test the functionality of icon model change signals | Critical |
| SC 7 | Test the functionality of graph model change signals | Critical |
| SC 8 | Test the functionality of event configuration model change signals | Critical |

Table 4: Vector Subsystem Test Suite

|  |  |  |
| --- | --- | --- |
| **TEST SUITE VECTOR SUBSYSTEM** | | |
| **Description of Test Suite** | **This test suite defines test cases for probing the creation and functionality of the Vector Subsystem** | |
| **Test Case Identifier** | **Objective** | **Criticality** |
| VS1 | Test creation of a Graph for a Vector | Critical |
| VS2 | Test creation of a Significant Log Entry | Critical |
| VS3 | Test creation of a Node for a Significant Log Entry | Critical |
| VS4 | Test creation of a Relationship within two Nodes inside a Graph | Critical |
| VS5 | Test functionality of movement for a Node in a Graph | Normal |
| VS6 | Test functionality of movement for a Relationship in a Graph | Normal |
| VS7 | Test deletion of a Node and Significant Log Entry from a Vector | Critical |
| VS8 | Test editing correlation between a Node and a Significant Log Entry | Critical |

Table 5: Event Configuration Test Suite

|  |  |  |
| --- | --- | --- |
| **TEST SUITE EVENT CONFIGURATION** | | |
| **Description of Test Suite** | **This test suite defines test cases for probing data transformation and representation of the event configuration model** | |
| **Test Case Identifier** | **Objective** | **Criticality** |
| EC 1 | Test event configuration instantiation with user input data | Critical |
| EC 2 | Test event configuration instantiation with JSON result from database query | Critical |
| EC 3 | Test event configuration JSON interpretation | Critical |
| EC 4 | Test event configuration object interpretation |  |
| EC 5 | Test event configuration attribute modification | Critical |
| EC 6 | Test saving event configuration data to database | Critical |
| EC 7 | Test event configuration attribute valid values | Critical |

# Test XX

<<The purpose of this section is to:

* document test input, specific test procedures, and outcomes.
* establish test methods,
* explain the nature and extent of each test >>

<< for each test case, complete the following: >>

## Test DB 3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test No.: DB3 | | | | Current Status: Failed | | |
| Test title: Update an object’s attribute with wrong input in the database | | | | | | |
| Testing approach: This test will be conducted by utilizing the Event Configuration class that will contain it’s given attributes, updating one of the object’s attribute will be reflected in the database GUI tool. | | | | | | |
| STEP  1 | OPERATOR ACTION  Begin test by making sure an Event Configuration object is stored in a document in the database within the Event Configuration collection.  Open up MongoDB Compass, be connected to the correct DB instance, select the Event Configuration Collection, view the contents of the document that will be updated. | PURPOSE  Initial Condition | | | EXEPCTED RESULTS  An Event Configuration object is stored in a document within the Event Configuration Collection. | COMMENTS |
| 2 | Invoke the Event Configuration class through a testing file that initializes the object’s attributes, change the data to assign to an attribute through the parameters.  Go into the Event Configuration class, specify the attribute that will be updated, and the data that the attribute will be modified to. | Invokes the update function within the class by calling the “update” function through the DB Façade. | | | The attribute is found in the database and updates the data to what was specified. | The attribute to be updated was misspelled and therefore nonexistent in the database document. |
|  | Run the testing file to update the document. | Final Condition | | | The update is reflected in MongoDB compass in the appropriate document and appropriate attribute field. |  |
| Concluding Remarks:  A new attribute field is added in the document with the data that was specified, due to providing a misspelled attribute that was not existent in the document in the database | | | | | | |
| Testing Team:  Elizabeth Barragan | | | Date Completed:  04/15/20 | | | |

## Test Ing 1

**Objective:** The objective of this test is to confirm whether the system performs cleansing properly and if not to determine which functionality of it fails.

**Notes:** Ingestion subsystem is not completely implemented yet. Practice data must be created specifically to point out the functionality of cleansing. At least 3 files must be created for this test. Test duration should be within 5 minutes.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test No.: Ing 1 | | | | Current Status: Failed | | |
| Test title: Test for cleansing for additional spaces, lines, & binary data. | | | | | | |
| Testing approach: This test will be performed by setting up practice data to ingest into the system | | | | | | |
| STEP | OPERATOR ACTION  Begin test by creating practice data and by saving it in the uncleansed directory  Have one file to have extra lines, another to have extra spaces, and the last one to have binary data | PURPOSE  Initial condition to properly test | | | EXEPCTED RESULTS  The system accepts the dummy files and begins process, so the user can make sure that conditions are met for cleansing | COMMENTS |
|  | Start the cleansing process by running the system | This is so you can make sure the system will accept the files and begin cleansing | | | The system begins the cleansing process without errors | If errors are found in this phase, the system doesn’t accept any files at all and is considered a failure. |
|  | After the system begins cleansing it lets the user know it is done cleansing. At this stage the user needs to check Cleansed directory made from the cleansing process. | This is to make sure that the system properly cleanses the files, and you know exactly which functionality lacks. | | | The system cleanses the files without any issues, and this can be seen in the cleansed folder. | An error occurs here, the file with the extra lines is cleansed properly along with the one with extra spacing, but the file with the binary data isn’t cleansed properly and is seen in the cleansed folder. |
| Concluding Remarks:  Test doesn’t meet the clients’ cleansing requirements by not cleansing binary data in files, therefore making it a failure. | | | | | | |
| Testing Team:  Yamel Hernandez | | | Date Completed:  04/16/20 | | | |

# User Interface Testing

<<This section focuses on the interaction between the user and the system. For testing the user interface, consider the following traits:

* Consistent terminology, shortcut keys, menu selections, and presentation
* Correct language, spelling, and grammar.
* Flexibility in navigation between windows and interface elements.
* Error handling that will inform user of critical operations.
* Follows standards and guidelines such as placement of scroll bars, windows, and menu items.

This section could be integrated into Section 4.

>>

# Test Schedule

<< Specify the schedule for testing activities. A table with the order and completion dates of the tests is useful. The table below might be useful.>>

|  |  |  |
| --- | --- | --- |
| **Task and date** | **People** | **Description** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Other Sections

<< Other sections that may appear in a test plan (but not required for this course) are:

* Test Management Requirements: how testing is to be managed; a delineation of responsibilities of each project organization involved with testing
* Staffing and training needs: delineate the responsibilities of those individuals who are to perform the testing, level of skill required, and training to be provided
* Environmental Requirements: describe the hardware (including communication and network equipment) needed to support testing; describe configuration of hardware components on which software and database to be tested are to operate.
* Software Requirements: describe the software needed to support testing; include the software code and databases that are object of the testing. Also include software tools such as compilers, CASE instruments and simulators that are needed to model the user’s operational environment.
* Risk and contingencies
* Cost: include an estimate of costs.
* Approvals
* Test Deliverables

>>

# Appendix

<< possibly more readable to put the expected output here and refer to it in the previous sections. Might also provide explicit directions for analysis of output, if it’s easier to read as an appendix or if analysis is post execution. >>

\*