We Showed Up

Prevent, Mitigate, and Recover (PMR) Insight Collective Knowledge System (PICK)
Test plan
Version <1.0>
04/13/20

Document Control

Approval

The Guidance Team and the customer shall approve this document.

Document Change Control

Initial Release:	1.0
Current Release:	1.0
Indicator of Last Page in Document:	*
Date of Last Review:	04/13/20
Date of Next Review:	04/14/20
Target Date for Next Update:	04/14/20

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:

Customer:

Software Team Members:

Change Summary

The following table details changes made between versions of this document

Version	Date	Modifier	Description
0.1	4/13/2020	Diego Rincon	Added initial test case T1 in Section 4
0.2	4/13/2020	Ricardo Alvarez	Added initial version of Section 2
0.3	4/14/2020	Jessica Redekop	Began creating Test Suites for Section 3.
0.4	4/14/20	Matthew Iglesias	Section 1.1 – 1.6: Introduction
0.5	4/15/20	Daniela Garcia	Added second test case T2 in section 4

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

1. Introduction

The overview of the PMR Insight Collective Knowledge (PICK) tool test plan follows within the following sub-sections.

1.1. Purpose

The purpose of a test plan document is to fundamentally describe, analyze and apply the necessary strategies for testing, scheduling and deliver the appropriate resources for adequate testing. The project is carefully designed and implemented to meet the client's needs, therefore, requires a tedious test plan. These include but are not limited to the testing of ingestion process, but the transcription, validation and log cleansing.

1.2. Scope

The scope of the project is based upon the current version of this test plan document, currently at version <INSERT MOST CURRENT VERSION>.

1.3. System Overview

The PICK tool is based on the client's needs, which the testing plan is to approach the system accordingly: The log ingestion process is access and ingested given on the set root directory, importing a varied file format in which to be imported. Before the system can move onto the next process of log entry ingestion, it must go through the appropriate transcriber module (depending on format) to be readable in the system. The next step involves cleansing the transcribed file with readable text, which removes unwanted characters which may interfere with correctly ingesting the log entries to tables and vectors. We want to make sure the testing approach involves a variety of appropriate file formats to be imported and cleansed to meet system needs.

1.4. Suspension and Exit Criteria

The suspension and exit criteria are two distinct methods implemented in the testing plan.

- **Suspension Criteria**: met during testing, which the cycle is temporarily suspended until the criteria is resolved.
    - All test cases will be executed if there are no suspension criteria
    - Test cases shall be suspended if 60% of the test cases fail
        - Strictly includes ingestion and validation process
- **Exit Criteria**: denotes when testing stops, it targets the results of the test and are necessary before we proceed with the next phase in development.
    - All critical tests must pass at a 100% rate

1.5. Document Overview

The Test Plan is indoctrinated with labeled sections for the remainder of this document:

Section 2 – Includes test items and features to be tested in the system
Section 3 – Includes the approach to test the system's functions
Section 4 – Includes documentation to applying testing methods to the system
Section 5 – Includes interaction between the user and the system

Section 6 – Includes test scheduling, order in testing phases for the system
Section 7 – Includes other forms of testing such as test deliverables of the system
Section 8 – Includes appendix of any output medium from the tested system

1.6. References

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

2. Test Items and Features

<< 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. >>

Due to the nature of the PICK Tool and the ongoing update of SPLUNK (April 2020), the items to be tested included classes, functions, methods and components in general ranging from networking, graphing and interaction, UI with SPLUNK. All of the classes that hold some sort of intelligence (provide any type of functionality) should be tested considering that this is a new release of the software altogether.

Classes to be tested along with the relevant methods include:

1. Ingestion
2. Validator
3. SPLUNKFacade
4. Cleanser
5. TableManager
6. UI
7. AudioTranscriber
8. ImageTranscriber
9. MongoDBFacade
10. Network

Features to be tested (with relevant methods and classes they pertain to):

1. Create Event – this is concerned with the event creation and initial setup of it:
    a. UI: display_new_event()
    b. UI: create_event_button_triggered()
    c. SPLUNKFacade: create_index(index_name)
    d. SPLUNKFacade: get_index_list()
    e. SPLUNKFacade: validate_user_info()
    f. MongoDBFacade: add_event()
    g. MongoDBFacade: add_vector()
    h. Network: set_lead()
2. Connect to Event – concerned with how an analyst that is not the lead will connect to an external event setup by another analyst:
    a. UI: connect_button_trigerred()
    b. UI: display_open_event()
    c. Network: get_event_list()
    d. Network: connect_user()
    e. MongoDBFacade: get_event(event_name)
    f. MongoDBFacade: get_vectort(vector_name)
3. Ingest directories into database – encompasses all the ingestion chores that shall be done for the log entries to be gathered from raw log files into the event:
    a. Ingestion: get_files_from_directory(root_path, white_team_folder, red_team_folder, blue_team_folder)
    b. Ingestion: ingest_directory_into_splunk(event_config)
    c. Ingestion: validate_files(log_files)
    d. Ingestion: validate_file_anyway(log_file)
    e. Validator: validate_file(log_file, start_time, end_time)
    f. Cleanser: cleanse_log_file(log_file)
    g. AudioTranscriber: transcribe_audio_file(log_file)
    h. ImageTranscriber: transcribe_image_file(log_file)
    i. SPLUNKFacade: add_file_to_index(log_file)
    j. SPLUNKFacade: add_directory_monitor(folder_path)
    k. MongDBFacade: add_log_file(log_file)

   l. TableManager: populate_log_entry_table()
   m. TableManager: populate_log_file_table()
   n. TableManager: populate_enforcement_action_report_table()
4. Recurrent Update of Entries – the functionality for recurrent refreshing of available entries to the user:
   a. SPLUNKFacade: refresh_log_entries()
   b. Ingestion: delta_found_triger()
   c. SplunkFacade: edit_log_entry(log_entry_id)
   d. SplunkFacade: remove_log_entry(log_entry_id)
5. Search and Filter – functionality for searching and filtering through the log entries of the event:
   a. UI: filter_search_triggered()
   b. SPLUNKFacade: search_in_index(index, search_arguments)
   c. SPLUNKFacade: refresh_log_entries()
   d. TableManager: populate_log_entry_table()
6. Manage Tables (General) – Interaction between the user and data from the tables, including log entries, nodes, vectors, log files and relationships:
   a. UI log_entry_table_clicked()
   b. UI: log_file_table_clicked()
   c. UI: enforcement_action_report_table_clicked()
   d. UI: vector_table_clicked()
   e. UI: relationship_table_clicked()
   f. UI: display_vector_list(vector_list)
   g. UI: display_long_description(long_description)
   h. TableManager: populate_log_entry_table(log_entries)
   i. TableManager: populate_log_files_table(log_files)
   j. TableManager: populate_vector_table(vector_list)
   k. TableManager: populate_nodes_table(nodes)
   l. TableManager: populate_relationship_table(relationships)
   m. TableManager: export_csv_from_table(table, folder_path, filename)
   n. SplunkFacade: remove_log_entry(log_entry_id)
   o. SplunkFacade: edit_log_entry(log_entry_id, field)
   p. MongoDBFacade: add_node(node)
   q. MongoDBFacade: remove_node(node_id)
   r. MongoDBFacade: edit_node(node_id, field)
   s. MongoDBFacade: add_relationship(relationship)
   t. MongoDBFacade: remove_relationship(relationship_id)
   u. MongoDBFacade: edit_relationship(relationship_id, field)
   v. GraphInterface: update_graph()
7. Graphing – functionality concerned with the visual displaying and exporting of the graph:
   a. GraphInterface: display_graph(graph)
   b. GraphInterface: update_graph(graph)
   c. GraphInterface: export_graph(graph)
   d. UI: tick_triggered()
   e. MongoDBFacade: get_graph()
   f. MongoDBFacade: get_nodes()
   g. MongoDBFacade: get_vector()
8. Version Control – networking methods used for vcs and signaling:
   a. Network: connect_to_lead(lead_ip)
   b. Network: push_change(change_request, analyst_id)
   c. Network: accept_change(change_request, analyst_id)
   d. Network: reject_change(change_request, analyst_id)

3. Testing Approach

<<Describe the approach to be used to the test the system. This description includes specifying the types of tests to be performed, e.g., tests designed to exercise system functions one by one; tests designed to exercise sequences of functions that approximate operational use of the system; tests designed to stress the system to its design and requirements limits. The description lists the specific tests to be performed, but does not give the test steps. For each of these tests, give it a name and specify its objective. Label the criticality of the test cases. >>

Table 1: Event Information

TEST SUITE Event		
Description of Test Suite	**This test suite will cover the tests appropriate to operational functionalities of creating an event.**	
Test Case Identifier	**Objective**	**Criticality**
TS1	**Test that an event can be created and added to splunk.**	**Critical**
TS2	**Test that event start date < event end date.**	**Critical**
TS2	**Test that the root folder of the event has 3 distinct folders: "Red", "White", "Blue"**	**Critical**
TS3	**Test that a "red folder" is selected when inserting a "red directory path"**	**Critical**
TS3	**Test that a "blue folder" is selected when inserting a "blue directory path"**	**Critical**
TS3	**Test that a "white folder" is selected when inserting a "white directory path"**	**Critical**

Table 3: Ingestion

TEST SUITE Ingestion		
Description of Test Suite	**This test suite will cover the tests appropriate to the functional requirements of the ingestion process.**	
Test Case Identifier	**Objective**	**Criticality**
TS1	**Test addition of new Event to SPLUNK.**	**Critical**
TS2	**Test for audio file transcribing ability.**	**Critical**
TS3	**Test for image file transcribing ability.**	**Critical**
TS4	**Test for cleansing of non-alphabetical and non-punctuation characters.**	**Critical**
TS5	**Test to validate timestamps within a certain range.**	**Critical**
TS6	**Test to advise the analyst for invalid files.**	**Critical**
TS7	**Test the ability to appeal a non-valid file.**	**Critical**

Table 6: Graph

TEST SUITE Graph	
Description of Test	**This test suite will cover the tests appropriate to connect the lead and analyst to**

Suite	the system and allocates exclusive functionalities.	
Test Case Identifier	**Objective**	**Criticality**
TS1	**Test adding a new node to the graph not connected to a log file**	**Critical**
TS2	**Test adding a new node to the graph that is connected to a log file**	**Critical**
TS3	**Test connecting two nodes with a relationship**	**Critical**
TS4	**Test adding information to an existing node**	**Critical**
TS5	**Test changing the icon of a node**	**Critical**

Table 4: Version Control

TEST SUITE Event Creation		
Description of Test Suite	**This test suite will cover the tests appropriate to the version control portion of the application.**	
Test Case Identifier	**Objective**	**Criticality**

Table 5: Network

TEST SUITE Event Creation		
Description of Test Suite	**This test suite will cover the tests appropriate to connect the lead and analyst to the system and allocates exclusive functionalities.**	
Test Case Identifier	**Objective**	**Criticality**
TS4	**Test to only allow leads to create events.**	**Critical**
TS5	**Test to connect analyst to lead.**	**Critical**
	Test to reject connection if no lead is selected.	**Critical**
	Test to close server when lead closes connection.	

4. Test Ingestion

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

4.1. Test TS1

Objective: Add a new index to Splunk
Notes: <<This area provides general notes concerning the test procedure. Such notes might include comments on how to execute the test procedure, an estimate of the test duration, the requirements of the procedure tests, or a statement of resources needed for this test.>>

Test No.: TS1			Current Status: Pending	
Test title: Test addition of new index in Splunk.				
Testing approach: The following test is to be conducted in the console application of Linux Ubuntu through the VMWare application. Alternatively, the test can be performed in the console application of Kali Linux through VMWare as well, or the command prompt application in Windows. Steps for this test case will be described in detail in the OPERATION ACTION column below.				
STEP	OPERATOR ACTION	PURPOSE	EXEPCTED RESULTS	COMMENTS
1	Click on the application icon in the menu of the main page of Linux. Type “console” in the search bar and select the console icon. Alternatively, if Windows is used, type “Command Prompt” in the main search bar located at the left-bottom corner of the screen and select the “Command Prompt” option when results are displayed.	Open the console, which will be used to open the PICK application.	The console window opens and is displayed in the screen.	

2	Use the command cd in the console to move to the directory where the PICK application is located, (i.e., cd Git/pick-splunk-implementation/src).	Reach the directory where the PICK application is located.	The directory where all the PICK application files are located is reached.	To ensure that directory where the PICK application is located has been reached, the command “ls” can be used in the console to list all the files in the current directory. If a file called “mainExec.py” appears in the list, the correct directory has been reached.
3	Write the command “python mainExec.py” in the console application to open the PICK application	Open the PICK application.	A prompt in the console appears asking for a username and a password	The username is … The password is …
4	Type the username and password, provided at the comments of step 3, into the prompt of the console.	Open the PICK application. Connect to Splunk.	A prompt appears in the console informing that the log files have been initialized, followed by another pop up with a window called PMR Insight Collective Knowledge (The PICK application), showing the contents of the Event tab	

5	Click on the “File” tab at the left-top corner of the PICK application window and select the option “new”.	Open Event-configuration window.	A window called “Dialog” with header “Event Configuration – Create New” pops up at the center of the screen.	
6	In the Dialog window, click on the text box below the label “Event Name” and write “test1”. Change the date on the spin box below the title “Event End Timestamp” to 1/2/2000 12:00AM.	Write the name of the new index to be added to Splunk. Change the Event End Timestamp to avoid timestamp conflicts.	The letters “test1” are displayed in the text box below Event Name. The spin box below Event End Timestamp displays the date and time 1/2/2000 12:00AM.	
7	In the text box below the label “Description” write the message “new index addition test”.	Write the description of the event.	The previously written message is displayed in the text box below Description.	

8	Click the button at the right below the Description textbox called "Save Event".	Add index test1 to Splunk.	At the left below the Description text box a label will appear with the following message: "Event test1 added".	End of test.
Concluding Remarks:				
Testing Team: Daniela, Diego, Jessica, Matthew, Ricardo			Date Completed:	

4.2. Test TS2

Objective: Test for audio file transcribing ability
Notes: <<This area provides general notes concerning the test procedure. Such notes might include comments on how to execute the test procedure, an estimate of the test duration, the requirements of the procedure tests, or a statement of resources needed for this test.>>

Test No.: TS2	Current Status: Pending
Test title: Test for audio file transcribing ability	
Testing approach: The following test is to be conducted in the console application of Linux Ubuntu through the VMWare application. Alternatively, the test can be performed in the console application of Kali Linux through VMWare as well, or the command prompt application in Windows. Steps for this test case will be described in detail in the OPERATION ACTION column below.	

STEP	OPERATOR ACTION	PURPOSE	EXEPCTED RESULTS	COMMENTS
1	Click on the application icon in the menu of the main page of Linux. Type “console” in the search bar and select the console icon. Alternatively, if Windows is used, type “Command Prompt” in the main search bar located at the left-bottom corner of the screen and select the “Command Prompt” option when results are displayed.	Open the console, which will be used to open the PICK application.	The console window opens and is displayed in the screen.	
2	Use the command cd in the console to move to the directory where the PICK application is located, (i.e., cd Git/pick-splunk-implementation/src).	Reach the directory where the PICK application is located.	The directory where all the PICK application files are located is reached.	To ensure that directory where the PICK application is located has been reached, the command “ls” can be used in the console to list all the files in the current directory. If a file called “mainExec.py” appears in the list, the correct directory has been reached.
3	Write the command “python mainExec.py” in the console application to open the PICK application	Open the PICK application.	Ensures that all libraries needed to run the project have been successfully installed if the GUI appears	If a library has not been installed the command line will show a prompt explaining what is missing

4	Create a new event and select a root directory to be ingested.	Selects files that will be searched through to see if there are any audio files that need to be transcribed	If there are any audio files in the selected directory, the audio transcriber should create a copy of these files and send them to the file transcriber API	If there are no audio files the audio transcribing step will be skipped
5	In the log entry table, if there were any audio files there should appear log entries with the audio file name as their source	Make sure the audio files were transcribed successfully	Log entries should appear in the log entry table with the name of the audio file as their source.	If the audio file was not transcribed successfully, the errors will appear in the Enforcement Action Report table where they can be revised and validated.
Concluding Remarks: A test to make sure audio files in directories can be properly transcribed before passing them on to cleansing and validating.				
Testing Team: Daniela, Diego, Jessica, Matthew, Ricardo		Date Completed:		

5. 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.
>>

6. 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

7. 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

>>

8. 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. >>