TeamWork

Prevent, Mitigate, and Recover (PMR) Insight Collective Knowledge System (PICK)

Test plan

Version 1.9

4/28/2020

Document Control

Approval

The Guidance Team and the customer shall approve this document.

Document Change Control

Initial Release:	4/10/2020
Current Release:	4/10/2020
Indicator of Last Page in Document:	$
Date of Last Review:	4/16/2020
Date of Next Review:	4/21/2020
Target Date for Next Update:	4/28/2020

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
Elsa Tai Ramirez
Peter Hanson

Customer:
Dr. Oscar Perez
Vincent Fonseca
Herandy Denisse Vazquez
Baltazar Santaella
Florencia Larsen
Erick De Nava

Software Team Members:
Charlie Juarez
Miriam Juarez
Angelica Marquez
Andrew Munoz
Aaron Rodriguez

Change Summary

The following table details changes made between versions of this document

Version	Date	Modifier	Description
1.0	4/12/2020	Angelica Marquez	Document creation
1.1	4/14/2020	Andrew Munoz, Aaron Rodriguez,	3. Testing approach tables

		Angelica Marquez	
1.2	4/15/2020	Andrew Munoz	Section 1.1, section 1.4, section 1.6, section 6 (partial)
1.3	4/16/2020	Charlie Juarez	Section 4 test cases(TSI|TSV; partial completion), Section 6 description
1.4	4/16/2020	Angelica Marquez	Section 1.5
1.4.1	4/26/2020	Miriam Juarez, Charlie Juarez	Make changes according to Jake Lasley's notes from the meeting on 4/23/2020. Completed Section 1.3
1.5	4/27/2020	Aaron Rodriguez	Section 4 test cases(TSDM:1,2,3), Section 3 review, Section 7, Section 6
1.6	4/27/2020	Miriam Juarez	Section 2
1.7	4/27/2020	Charlie Juarez	Section 3 description, Section 4 test cases(TSC:1,2|TSI:1,2,3|TSV:1,2,3,4), Section 2 review
1.8	4/28/2020	Angelica Marquez	Section 4 test cases (TSFC:1,3,4,5,6,7) Section 5, Section 6 (Patial)
1.9	4/28/2020	Andrew Munos	Section 4 test cases (TSGC: 1,2,3,4)

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

1.1. Purpose

The purpose of creating the test plan document is to report the testing approach that will be used on the project. It will facilitate the task of testing and help organize the development and test effort to assess the system. The document contains the information on testing items and features, testing approach, test cases, and the test schedule to accomplish this purpose. The intended audience are the software engineers that will be implementing the test plan and the analysts that will be using the software, for future verification and maintenance.

1.2. Scope

The PMR Insight Collective Knowledge (PICK) System will facilitate the current node correlation process for analysts to complete their cyber security attack graphs. Currently, the analysis process takes months to complete and the system is needed to shorten the completion time to a more reasonable length. The system will save the analysts time with the automatization of log file processing along with filtering/searching functionality. The system’s attack graph creator component will further save analysts time and provide a flexible structure for analysts to work off of. The purpose of PICK is to retrieve and layout log information provided by the adversarial assessment. The software will ingest the log files to the analyst's configuration and allow the analyst to produce attack graphs containing the events that transpired to provide a report for LSH.

1.3. System Overview

The system will gather log files provided by analysts, which will be cleansed and validated according to the analysts criteria. After the files have been cleansed and validated, log entries will be derived from such files after which the log entries will be ingested into the system’s database. The log entries will now be manipulated by the system, therefore the analyst can manipulate the data created by different teams in a precise way. The data that is within the system can then be put into reports in table, or graph representations. From which the analyst will have the freedom to select what such reports will include, for example: date, time, actions, teams involved, location, and descriptions.

1.4. Suspension and Exit Criteria

The suspension criteria will be defined by the critical tests that will be conducted in the test cases, if any of these critical tests fail, the testing will be suspended for appropriate fixes. The exit criteria will be defined by the non-critical and critical tests combined, in which 100% of the critical tests must pass along with a percentage of 80% and above for non-critical tests.

1.5. Document Overview

The Test Plan encompases Test Suites which describes test cases, and the importance of the test. Defines test objectives, test criteria, resource planning, and a detailed summary of all resources required of complet it, including human, equipment, and material. In addition, the plan test environment with the schedule and estimation, for example project deadline and resources availability.

Section 2 talks about test items and features as it explains how the components; displays, functions, and features will be interacted with. Section 3 describes the approach used to test the system, in part this specifies the types of testing that will be performed to exercise the system functions and the sequence of events to stress the system. Afterwards, section 4 exposes the test cases for different available actions in the system, explaining its steps to recreate it as well as the expected results. Section 5, similar to section 4, explains interface testing which focuses on the test cases that interact between the user and the system. Moreover, section 6 establishes the schedules for the tests, who is responsible for it , and a description of the test. Finally, section 7 encloses the references for the test cases.

1.6. References

[1] S. Roach et al, Software Requirements Specification, Lethality, Survivability, and HSI Directorate (LSH), 2019.

2. Test Items and Features

Component Descriptions

Validation and Ingestion Subsystem:

Validation Component: In charge of validating all the log file data to ensure it is ready to continue to the ingestion process. This validation process is to ensure that the files can be uploaded to splunk which will create them into log entries. Contracts 1, 2, 3, 4, and 5 [Appendix A.1 - A.5].

Ingestion Component: Ensures that validated log files, which are now log entries, are stored within the PICK system's MongoDB database. A log file is considered fully ingested once it has been cleansed, validated, and turned into multiple log entries into the database. Contracts 3, 6, and 7 [Appendix A.3, A.6, A.7].

Project Initialization Subsystem:

Configuration Component: Composed of three classes, ProjectConfigWindow, Calendar Dialog and Event Configuration. Event Configuration purpose is to set the general description of the project that is being created. ProjectConfigWindow's purpose is to get the directories from which the raw file content will be extracted from. Calendar Dialog purpose is to set the time frame from which the raw files will be extracted from.

Startup Component: Delegate the direction the software will go based on the users actions; which are going to the project configuration or the navigation subsystem.

Navigation Subsystem:

Regex Component: In charge of the information retrieval process within the log entry content in the system (MongoDB). Contracts 6, 13, and 14 [Appendix A.6, A.13, A.14].

Graph Creation Subsystem:

Graph Configuration: Display a graph, along with being able to edit, and export it. Contract 20 [Appendix A.20].

Graph Export: Create a JPG image of the graph or a csv file to be exported.

Vector Node Modification Component: Handles the modification process of the vectors, nodes, and relationships. Contracts 15, 16, 17, and 18 [Appendix A.15 - A.18].

Classes description

Script Handler: Representation of the cleansing script used on the log files during the cleansing process. It has the responsibility of cleansing the log files that have been selected by removing unreadable/unnecessary characters. Contract 1 [Appendix A.1].

Log File: Representation of a file of any type (e.g. .txt, .jpeg, etc.), that have been ingested by the system. It has the responsibility of knowing its own file type, and knows all its own content. Contract 2 [Appendix A.2]

ValidationIngestionWindow: Visual representation of the handler of the validation and ingestion process of the log file data. The superclass will be ProjectConfigWindow, and will handle the transition to NavigationWindow. Contract 3 [Appendix A.3].

Enforcement Action Report: Represents generated report of each invalidated log file. This class will include the error(s) associated with the log file that makes it invalidated, and the analysts' reasons why a log file is invalidated. Contract 4 [Appendix A.4].

Description DirDialog: Handle the directory choosing function whenever the software requires it. The superclass will be ProjectConfigWindow as that is where all the information will be returning to once a directory has been chosen. Contract 5 [Appendix A.5].

Log Entry: Represents the breakdown parts of a log file once it is ingested into the PICK system. Log entries allow the analyst to see all log file data in a single type of format regardless of the log file filetype (i.e., PDF, PNG). Every Log Entry is the converted format of a log file in the PICK system. The Log Entry class also represents a piece of log file information that can be eventually stored in a Vector and Graph image output. Contract 6 [Appendix A.6].

ProjectConfigWindow: Visual representation of the different entries once an input is provided for the event title and event description. It is also responsible for displaying the directories from which the raw log files will be extracted from. Afterwards, it is aswell responsible for displaying the time range from which the selected raw log files will be retrieved. The superclass of ProjectConfigWindow is main. Contract 8- 10 [Appendix A.8-A.10].

CalendarDialog: Displays a calendar from which the user is able to select a date. Contract 11 [Appendix A.11].

DirDialog: This class is responsible for displaying a file chosen by the user from a directory. Contract 12 [Appendix A.12].

Main: Display the options to determine which window will be displayed next. This class offers the user the ability to choose from creating a new project and opening an existing one. Here, the new project is handled by the Configuration component and the Validation and Ingestion Subsystem, while opening an existing project is handled by the Navigation Subsystem.

Filter: Represents the chosen filtering requirements the user has requested to be processed. It has the responsibility of knowing and processing the filtering user request. Contract 13 [Appendix A.13].

NavigatorWindow: Visual representation of the handler of the log entry navigation/filtering process. It is responsible for displaying all the log entry content within the system, and provides filtering options to the user for navigation through the log entries. The superclass will be ValidationIngestionWindow, and will handle the transition to VectorTableWindow. Contract 14 [Appendix A.14].

Vector: Representation of sequence of actions built up by nodes that detail the happenings of an event. It is responsible for knowing its vector name, nodes, and vector description. Contract 15 [Appendix A.15].

Node: Significant event representation that was marked for a vector. It is responsible for knowing its node ID, name, timestamp, description, log entry reference, log creator, event type, icon type, source, node visibility, and also knows each of its attribute visibility: Boolean flags to be shown on the graph. Contract 16 [Appendix A.16].

Relationship: Visual representation of a connection between nodes in a vector which is shown on the graph. The relationship is responsible for knowing its relationship ID, parent ID, child ID, and its label. Contract 17 [Appendix A.17].

VectorTableWindow: Handler of the vector, node and relationship modification process. It is responsible for displaying all the vector, node, and relationship table views which the user can interact with to modify the data. The superclass will be NavigatorWindow, and will handle the transition back to GraphWindow and NavigatorWindow. Contract 18 [Appendix A.18].

Graph: Visual representation of a vector. It knows attributes of Orientation, Export Format, Interval Units, Interval, Position of Nodes, and Position of Relationships. Contract 19, 21 [Appendix A.19, A.21].

GraphWindow: It is responsible for displaying the graph, knowing the graph, and each one of the graph nodes. Contract 20 [Appendix A.20].

3. Testing Approach

The following tables describe the test suites that implement all the test cases in the rest of the test plan along with general information about each individual test case.

Table 1: Ingestion Test Plan

TEST SUITE Configuration		
Description of Test Suite	The following configuration test plan will be conducted in order to ensure the application can correctly process configurations on the project	
Test Case Identifier	**Objective**	**Criticality**
TSC OC1	Allow directory selection to retrieve log files.	Critical
TSC OC2	Specify the time range to selectively pick the correct log files to be ingested.	Critical

Table 2: Ingestion Test Plan

TEST SUITE Ingestion		
Description of Test Suite	The following ingestion test plan will be conducted in order to ensure the application does not result in failures in regards to the ingestion process.	
Test Case Identifier	**Objective**	**Criticality**
TSI OC1	Do not ingest log entries in the non-ingestable table.	Critical
TSI OC2	Ingest all log entries in the intestable table.	Critical
TSI OC3	Allow non-validated log entries to become ingestible if the user changes the non-validated entries to become validated.	Critical

Table 3: Validation Test Plan

TEST SUITE Cleansing/Validation		
Description of Test Suite	Data validation test plan will be performed to ensure correctness of inspecting data in the clansed log files based on predefined data validation rules.	
Test Case Identifier	**Objective**	**Criticality**
TSV OC1	The cleansing script is applied to the log files.	Critical
TSV OC2	Validate the log files specified in the validate table.	Critical
TSV OC3	Only validate files that the user pointed to in the configuration.	Critical
TSV OC4	Validate files that have passed the cleansing process.	Critical

Table 4: Filter Control Test Plan

TEST SUITE Filter Control	
Description of Test Suite	The filter control test plan will be performed to ensure the provided filtering components in the application are functional and correct.

Test Case Identifier	Objective	Criticality
TSFC OC1	Filter log entries in response to the appropriate filter inputs from the user.	Normal
TSFC OC2	Provide functional regex keyword search filter option.	Critical
TSFC OC3	Do not filter any log entries if the user has no filter inputs activated.	Normal
TSFC OC4	Ensure all filter inputs are accessible in every UI window they must exist in.	Critical
TSFC OC5	Test the Filter Control if log entries are filtered by the Team that was originated from.	Critical
TSFC OC6	Log entries are able to be filtered by time.	Critical
TSFC OC7	Log entries are able to be filtered by Location.	Critical

Table 5: Database Management Test Plan

TEST SUITE Database Management		
Description of Test Suite	The database management test plan will be conducted to guarantee the database is accurately managing the provided data.	
Test Case Identifier	**Objective**	**Criticality**
TSDM OC1	Create the database in response to the user using the application for the first time.	Critical
TSDM OC2	Update the appropriate db table in response to the user editing an item from that db table in the system.	Critical
TSDM OC3	Display UI tables with data from the appropriate db table data.	Critical
TSDM OC4	Output appropriate requested data from the db table.	Critical

Table 6: Graph Control Test Plan

TEST SUITE Graph Control		
Description of Test Suite	The Graph Control test plan will be performed to ensure the correct content display, and exportation of the graph	
Test Case Identifier	**Objective**	**Criticality**
TSGC OC1	Graph accurately displays the specified nodes and relationships .	Critical
TSGC OC2	Graph is exported to the format specified by the user.	Normal
TSGC OC3	Graph displays content associated with the specified vector.	Critical
TSGC OC4	Graph accurately displays the chosen visible attributes of a node.	Normal

4. Test Cases

4.1 Test TSC OC 1

Objective: Specify the Directory from which log files will be retrieved from.
Notes: This test assumes a new project is being created.

Test No.: 1		Current Status: Pending		
Test title: Directory Specification				
Testing approach: The tester must have 3 directories, all different from one another.				
STEP	OPERATOR ACTION	PURPOSE	EXPECTED RESULTS	COMMENTS
1	The tester presses the directory button [the one with the folder icon] on an entry requiring input.	Determine if the program can open a file chooser window.	The program should display a file chooser window from which only directories can be chosen.	
2	The tester selects one unique directory and presses the open button.	Determine if the program can retrieve the folder path of the chosen directory.	The program should close the file chooser window and display the folder path based on the tester's selection.	
3	Repeat steps 1 and 2 consecutively until all three folder paths have been specified.	Determine if the program can correctly retrieve all folder paths specified.	The program should display the folder paths, each of which is respective to the selection of the tester.	
Concluding Remarks:				
Testing Team: << List members of testing team and lead >>		Date Completed:		

4.2 Test TSI OC 2

Objective: Time range specification from which log files will be selected.
Notes: This test assumes a new project is being created.

Test No.: 2			Current Status: Pending	
Test title: Time range for log files				
Testing approach: The tester will specify the time range from which the log files are to be chosen.				
STEP	OPERATOR ACTION	PURPOSE	EXPECTED RESULTS	COMMENTS
1	The tester presses the “Select a start date.” button.	Determine if the program is able to display a calendar window.	The program should display a calendar window.	
2	The tester selects a date.	Determine if the program can retrieve the date specified from the tester.	The program should display the chosen date in the bottom left corner of the screen.	
3	The tester presses the “Confirm Date” button.	Determine if the program can retrieve the date specified and display it on the configuration window.	The program should close the calendar window and display the chosen date below the “Select a start date.” button.	
4	The tester presses the “Select a end date.” button.	Determine if the program is able to display a calendar window.	The program should display a calendar window.	
5	The tester selects a date.	Determine if the program can retrieve the date specified from the tester.	The program should display the chosen date in the bottom left corner of the screen.	
6	The tester presses the “Confirm Date” button.	Determine if the program can retrieve the date specified and display it on the configuration window.	The program should close the calendar window and display the chosen date below the “Select a end date.” button.	
Concluding Remarks:				

Testing Team: << List members of testing team and lead >>	Date Completed:

4.3 Test TSI OC 1

Objective: Non-Ingestable Log Entries to not be ingestible.
Notes: This test assumes nothing went wrong in the cleansing process.

Test No.: 3			Current Status: Pending	
Test title: Not ingesting Non-ingestable log entries				
Testing approach: The tester must have previously cleansed the log files which generate the log entries, in this case non-ingestable log entries.				
STEP	OPERATOR ACTION	PURPOSE	EXPECTED RESULTS	COMMENTS
1	The tester presses the “Ingest Log Entries and Continue to Navigator” button.	Determine if the program is able to ingest the log entries without the ones in the non-ingestable entries.	The program should take the ingestable log entries and ingest them into MongoDB. With the non-ingestable log entries not being put into MongoDB. In which the program will move on to the navigator window.	
Concluding Remarks:				
Testing Team: << List members of testing team and lead >>			Date Completed:	

4.4 Test TSI OC 2

Objective: Ingest Log Entries
Notes: This test assumes nothing went wrong in the cleansing process.

Test No.: 4	Current Status: Pending
Test title: Ingest the log entries specified in the ingestable list.	

Testing approach: The tester must have previously cleansed the log files which generate the log entries, in this case ingestable log entries.				
STEP 1	OPERATOR ACTION The tester presses the "Ingest Log Entries and Continue to Navigator" button.	PURPOSE Determine if the program is able to ingest the log entries in the ingestable list.	EXPECTED RESULTS The program should take the log entries in the ingestable list and ingest them into MongoDB. In which the program will move on to the navigator window.	COMMENTS
Concluding Remarks:				
Testing Team: << List members of testing team and lead >>		Date Completed:		

4.5 Test TSI OC 3

Objective: Non-Ingestable Log Entries become ingestible.
Notes: This test assumes nothing went wrong in the cleansing process.

Test No.: 5		Current Status: Pending		
Test title: Making non-ingestable log entries ingestible.				
Testing approach: The tester must have previously cleansed the log files which generate the log entries, in this case both ingestable and non-ingestable log entries.				
STEP 1	OPERATOR ACTION The tester presses the "Ingest Log Entries and Continue to Navigator" button.	PURPOSE Determine if the program is able to switch all non-ingestable log entries into ingestable log entries.	EXPECTED RESULTS The program should take the ingestable log entries and ingest them into MongoDB. With the non-ingestable log entries not being put into MongoDB In which the program will move on to the navigator window.	COMMENTS

2	The tester presses the “Ingest Log Entries and Continue to Navigator” button.	Determine if the program is able to ingest the log entries without the ones in the non-ingestable entries.	The program should take the ingestable log entries and ingest them into MongoDB. With the non-ingestable log entries not being put into MongoDB. In which the program will move on to the navigator window.	
Concluding Remarks:				
Testing Team: << List members of testing team and lead >>		Date Completed:		

4.6 Test TSV OC 1

Objective: Cleanse Log Files
Notes: This test assumes nothing went wrong in the configuration window and has provided all inputs necessary before moving on to the validation and ingestion window.

Test No.: 6			Current Status: Pending	
Test title: Cleanse the log files specified in the cleanse table.				
Testing approach: The tester must have previously selected a non empty red, blue, and white team folder from which the program can extract the log files to cleanse.				
STEP	OPERATOR ACTION	PURPOSE	EXPECTED RESULTS	COMMENTS
1	The tester presses the cleanse button first in the window.	Determine if the program is able to cleanse the log files on the cleanse table.	In the cleanse table the cleansed value should have changed from no to yes and in the validate table the cleansed values should have appeared on the validate table with the validated value as no.	

Concluding Remarks:	
Testing Team: << List members of testing team and lead >>	Date Completed:

4.7 Test TSV OC 2

Objective: Validate Log Files
Notes: This test assumes nothing went wrong in the cleansing process after pressing the cleanse button.

Test No.: 7			Current Status: Pending	
Test title: Validate the log files specified in the validate table.				
Testing approach: The tester must have previously pressed the cleanse button and have the cleansed files displayed on the validate table.				
STEP	OPERATOR ACTION	PURPOSE	EXPECTED RESULTS	COMMENTS
1	The tester presses the validate button.	Determine if the program is able to validate the log files on the validate table.	The system should prompt the user for their Splunk credentials.	
2	The tester correctly inputs their Splunk credentials.	Determine if the Splunk upload is possible.	The system should have uploaded the log files to splunk and should have returned the log entries derived from the log files. The log entries will then be displayed on the ingestable list. In the validate table the validated value should have changed from no to yes.	
Concluding Remarks:				

Testing Team: << List members of testing team and lead >>	Date Completed:

4.8 Test TSV OC 3

Objective: Validate Log Files within the time range given in the configuration
Notes: This test assumes nothing went wrong in the configuration process.

Test No.: 8	Current Status: Pending
Test title: Validate the log files specified in the validate table.	
Testing approach: The tester must have previously pressed the cleanse button and have the cleansed files displayed on the validate table.	

STEP	OPERATOR ACTION	PURPOSE	EXPECTED RESULTS	COMMENTS
1	The tester previously applied the time range in the configuration window.	Determine if the program is able to apply the time constraints to the log files to be validated.	The system should only display the log files that are within the time range in the cleanse/validation table.	
2	The tester presses the validate button.	Determine if the program is able to validate the log files on the validate table.	The system should prompt the user for their Splunk credentials.	
3	The tester correctly inputs their Splunk credentials.	Determine if the Splunk upload is possible.	The system should have uploaded the log files to splunk and should have returned the log entries derived from the log files. The log entries will then be displayed on the ingestable list. In the validate table the validated value should have changed from no to yes.	

Concluding Remarks:

Testing Team: << List members of testing team and lead >>	Date Completed:

4.9 Test TSV OC 4

Objective: Validate Log Files
Notes: This test is to be conducted after cleansing previous log files.

Test No.: 9	Current Status: Pending
Test title: Validate the log files specified in the validate table.	
Testing approach: The tester must have previously pressed the cleanse button before starting this test case.	

STEP	OPERATOR ACTION	PURPOSE	EXPECTED RESULTS	COMMENTS
1	The tester presses the validate button.	Determine if the program is able to validate the log files on the validate table that were created from the cleansing process.	The system should prompt the user for their Splunk credentials.	
2	The tester correctly inputs their Splunk credentials.	Determine if the Splunk upload is possible.	The system should have uploaded the log files to splunk and should have returned the log entries derived from the log files. The log entries will then be displayed on the ingestable list. In the validate table the validated value should have changed from no to yes.	

Concluding Remarks:	
Testing Team: << List members of testing team and lead >>	Date Completed:

4.10 Test TSFC OC1

Objective: Filter log entries in response to the appropriate filter inputs from the user.
Notes:

Test No.: 10	Current Status: Pending
Test title: Test Suite Filter Control, filter log entries from user inputs.	
Testing approach: Sample input test to see the proper working of the filter.	

STEP	OPERATOR ACTION	PURPOSE	EXPECTED RESULTS	COMMENTS
1	The tester clicks on Navigation view	Check if Navigation window can display filter control buttons	Navigation window will be displayed with filter control	
2	The tester randomly chose a combination of filters. For example “Blue team”	See if there the system is filtering the data.	Display filtered log Entries.	

Concluding remarks: This test case is only to observe that information is being filtered, individual filetinf options will be tested to be able to identified if individual filtering options work.	
Testing Team: Charlie Juarez Angelica Marquez - Lead Andrew Munoz Aaron Rodriguez	Date Completed: N/A

1.7. Test TSFC OC 2

Objective: Functional Regex

Notes: This test assumes there is a log entry with the keyword inside of it.

Test No.: 11			Current Status: Pending	
Test title: Test the regex keyword search filter option.				
Testing approach: The tester must be on the navigator window of an active project.				
STEP	OPERATOR ACTION	PURPOSE	EXPECTED RESULTS	COMMENTS
1	The tester provides a word that is guaranteed to be in a log entry in the regex bar.	Determine if the program is able to search and display all log entries with the keyword provided.	In the navigation table the log entries with the searched keyword are displayed.	
Concluding Remarks:				
Testing Team: Charlie Juarez Angelica Marquez - Lead Andrew Munoz Aaron Rodriguez			Date Completed:	

4.10 Test TSFC OC3

Objective: Do not filter any log entries if the user has no filter inputs activated.
Notes: This test assumes that the system has validated log entries to be displayed**.**

Test No.: 12			Current Status: Pending	
Test title: Test Suite Filter Control, no filter inputs activated				
Testing approach: Test the system by having a NULL input of not selecting any filtering options.				
STEP	OPERATOR ACTION	PURPOSE	EXPECTED RESULTS	COMMENTS
1	The tester clicks on Navigation view	Check if Navigation window can display filter control buttons	Navigation window will be displayed with filter control	

2	If there is a filter selected, uselect the filter.Tester has to make sure that no filter option is selected.	The purpose to deselect filters is to have no filter selected to be able to perform the test.	All log entries should be displayed on the table window.	If there are no entries the test fails.
Concluding remarks: The objective is to test the displaying result of the table window where there are not filters selected, this being all log entries displayed.				
Testing Team: Charlie Juarez Angelica Marquez - Lead Andrew Munoz Aaron Rodriguez	Date Completed: N/A			

4.10 Test TSFC OC4

Objective: Ensure all filter inputs are accessible in every UI window they must exist in.
Notes: UI windows expected to display are Navigation and Graph window.

Test No.: 13	Current Status: Pending			
Test title: Test Suite Filter Control filter inputs across all UI where they should exist.				
Testing approach: Test across every window where the filtering option should be available and working.				
STEP	OPERATOR ACTION	PURPOSE	EXPECTED RESULTS	COMMENTS
1	The tester clicks on Navigation view	Check if Navigation window can display filter control buttons	Navigation window will be displayed with filter control.	
2	The tester will click on Graph view.	Check if Graph window will display filter control buttons.	Graph UI window will display filter control buttons.	
Testing Team: Charlie Juarez Angelica Marquez - Lead Andrew Munoz Aaron Rodriguez	Date Completed: N/A			

4.10 Test TSFC OC5

Objective: Test the Filter Control if log entries are filtered by the Team that was originated from.
Notes: This test assumes that the system has validated log entries to be displayed from each team option.

Test No.: 14			Current Status: Pending	
Test title: Test Suite Filter Control Filter by Team.				
Testing approach: Test the Team functionality filter button. This test will consist of three subtests for each team, Team Blue, Team Red, and Team White.				
STEP	OPERATOR ACTION	PURPOSE	EXPECTED RESULTS	COMMENTS
1	The tester clicks on Navigation view	Check if Navigation window can display filter control buttons	Navigation window will be displayed with filter control	
2	The tester will select ”Blue” under “Team” option	Test if the button gets selected and if the log entries get filtered by the “Blue” team.	Navigation window will display log entries originated by the Blue Team	
3	The tester will unselect ”Blue” under “Team” option.	Have unselected filter options to test next button on Filter Control by Team	Display all log entries on Navigation Table Window	
4	The tester will select ”Red” under “Team” option	Test if the button gets selected. Also, test the filter option of Red team log entries.	Navigation window will display log entries originated by the Red Team.	
5	The tester will unselect ”Red” under “Team” option.	Have unselected filter options to test next button on Filter Control by Team	Display all log entries on Navigation Table Window	
6	The tester will select ”White” under “Team” option	Test if the button gets selected. Also, test the filter option of White team log entries.	Navigation window will display log entries originated by the White Team	
Concluding remarks: The objective is to test the option of filtering by Team.				

Testing Team: Charlie Juarez Angelica Marquez - Lead Andrew Munoz Aaron Rodriguez	Date Completed: N/A

4.10 Test TSFC OC6

Objective: Ensure that the Filter by Time correctly filters log entries by time.
Notes: This test assumes that the system has validated log entries with a time stamp.

Test No.: 15			Current Status: Pending	
Test title: Test Suite Filter Control log entries filtered by time.				
Testing approach: Test a time range to filter log entries.				
STEP	OPERATOR ACTION	PURPOSE	EXPECTED RESULTS	COMMENTS
1	The tester clicks on Navigation view	Check if Navigation window can display filter control buttons	Navigation window will be displayed with filter control.	
2	The tester will click on “Start Time“ under “Time” and will select the start day and time of the range.	Input start time.	The UI will let the Tester select the information on “Start Time” in day time format and be able to input.	The Tester will have to have some knowledge of the Time stamps on the log entries to be able to select an appropriate Start time for the range.
3	The tester will click on “End Time“ under “Time” and will select the day and end time of the range.	Input start time.	The UI will let the Tester select the information on “Start change” and be able to input.	The Tester will have to have some knowledge of the Time stamps on the log entries to be able to select an appropriate End time for the range.
Testing Team: Charlie Juarez Angelica Marquez - Lead Andrew Munoz Aaron Rodriguez			Date Completed: N/A	

4.10 Test TSFC OC7

Objective: Log entries are able to be filtered by “Location”

Notes: This test assumes that the system has validated log entries to be displayed with Location information.

Test No.: 16			Current Status: Pending	
Test title: Test Suite Filter Control, filtered by Location				
Testing approach: Test the filter by "Location" on the Navigation window Filter Control Buttons by observing the output after selecting the filtered option.				
STEP	OPERATOR ACTION	PURPOSE	EXPECTED RESULTS	COMMENTS
1	The tester clicks on Navigation view	Check if Navigation window can display filter control buttons	Navigation window will be displayed with filter control "Location". The "Location" search bar will include only options from log entries.	
2	The tester selects any location under "Location"	The purpose is to test if the selected location filters log data.	Log entries with selected location should be displayed on Navigation window.	
Concluding remarks:				
Testing Team: Charlie Juarez Angelica Marquez - Lead Andrew Munoz Aaron Rodriguez			Date Completed: N/A	

4.1 Test TSDM OC1

Objective: Create the database in response to the user using the application for the first time.
Notes: Database being used is MongoDB

Test No.: TSDM OC1	Current Status: Pending
Test title: Database Creation Test	

Testing approach: This test will be conducted using the PICK system running on a Kali Linux machine. Behavior is observed on the Kali Linux screen and on the db data.				
STEP	OPERATOR ACTION	PURPOSE	EXPECTED RESULTS	COMMENTS
1	The tester executes the PICK application for the first time, meaning the first instance of that PICK application.	Determine that the PICK system can work in first time execution.	The PICK system should function normally when being initially executed.	
2	The tester reaches the Ingestion/Validation window.	This is the first time the database will be used in the system, so reaching this window is critical.	The Ingestion/Validation window should function normally.	
3	The tester checks that no database errors are in the Ingestion/Validation window.	Determine that the database exists.	No database errors should come up in this window, as the database should exist.	
Concluding Remarks: With this test, we can know that the database is created when the analyst will use an instance of the PICK system for the first time for a project.				
Testing Team: Charlie Juarez Angelica Marquez - Lead Andrew Munoz Aaron Rodriguez	Date Completed: N/A			

4.8 Test TSDM OC2

Objective: Update the appropriate db table in response to the user editing an item from that db table in the system.
Notes: Database being used is MongoDB

Test No.: TSDM OC2	Current Status: Pending
Test title: Database Table Update Test	
Testing approach: This test will be conducted using the PICK system running on a Kali Linux machine. Behavior is observed on the Kali Linux screen and on the db data.	

STEP	OPERATOR ACTION	PURPOSE	EXPECTED RESULTS	COMMENTS
1	The tester changes data from an item in the log entry UI table.	Determine if the log entry db table is properly updated when the log entry UI table is updated.	The changed data in the log entry UI table should also be altered in the log entry db table.	
2	The tester changes data from an item in the vector UI table.	Determine if the vector db table is properly updated when the vector UI table is updated.	The changed data in the vector UI table should also be altered in the vector db table.	
3	The tester changes data from an item in the graph UI view.	Determine if the graph db table is properly updated when the graph UI view is updated.	The changed data in the graph UI view should also be altered in the graph db table.	
4	The tester changes data from an item in the node UI table.	Determine if the node db table is properly updated when the node UI table is updated.	The changed data in the node UI table should also be altered in the node db table.	
5	The tester changes data from an item in the time filter UI list.	Determine if the time filter db table is properly updated when the time filter UI list is updated.	The changed data in the time filter UI list should also be altered in the time filter db table.	

Concluding Remarks: With this test, we can know that every database table/collection works and is updated when the appropriate UI view related to a table/collection is updated.	
Testing Team: Charlie Juarez Angelica Marquez - Lead Andrew Munoz Aaron Rodriguez	Date Completed: N/A

4.9 Test TSDM OC3

Objective: Display UI tables with data from the appropriate db table data.
Notes: Database being used is MongoDB

Test No.: TSDM OC3	Current Status: Pending
Test title: UI Update From Database Test	

Testing approach: This test will be conducted using the PICK system running on a Kali Linux machine. Behavior is observed on the Kali Linux screen and on the db data.				
STEP	OPERATOR ACTION	PURPOSE	EXPECTED RESULTS	COMMENTS
1	The tester views the log entry UI table with dummy data in the log entry db table.	Determine if the log entry UI table is correctly reading/querying data from the log entry db table.	The log entry UI table should be displaying all the data exactly as stored in the log entry db table.	
2	The tester views the vector UI table with dummy data in the vector db table.	Determine if the vector UI table is correctly reading/querying data from the vector db table.	The vector UI table should be displaying all the data exactly as stored in the vector db table.	
3	The tester views the graph UI view with dummy data in the graph db table.	Determine if the graph UI view is correctly reading/querying data from the graph db table.	The graph UI view should be displaying all the data exactly as stored in the graph db table.	
4	The tester views the node UI table with dummy data in the node db table.	Determine if the node UI table is correctly reading/querying data from the node db table.	The node UI table should be displaying all the data exactly as stored in the node db table.	
5	The tester views the time filter UI list with dummy data in the time filter db table.	Determine if the time filter UI table is correctly reading/querying data from the time filter db table.	The time filter UI list should be displaying all the data exactly as stored in the time filter db table.	
Concluding Remarks: With this test, we can know that every UI view related to a database table/collection is receiving its data from the appropriate table/collection.				
Testing Team: Charlie Juarez Angelica Marquez - Lead Andrew Munoz Aaron Rodriguez	Date Completed: N/A			

4.10 Test TSDM OC4

Objective: Output appropriate queried data from the db table.

Notes: Database being used is MongoDB

Test No.: TSDM OC4			Current Status: Pending	
Test title: Database Table Query Test				
Testing approach: This test will be conducted using the PICK system running on a Kali Linux machine. Behavior is observed on the Kali Linux screen and on the db data.				
STEP	OPERATOR ACTION	PURPOSE	EXPECTED RESULTS	COMMENTS
1	The tester enters a valid regex search in the regex search bar.	Determine if the regex search functionality is working correctly.	The log entry table should reduce its data to only the data that is true for the given regex search.	
2	The tester compares the returned result with what the MongoDB Compass Community application would return.	Determine if the correct regex search is being used in the PICK system to query data from the db.	The log entry tables reduced data should match what MongoDB Compass Community would return.	
3	The tester checks the actual query statement that was used to perform the regex search in the regex search bar.	Determine if the query statement used in the db is valid and will be valid for future queries. Make sure the query was not correct by chance, but by purposeful programming.	The query statement should have correct syntax to ensure all future query statements will have correct syntax.	
Concluding Remarks: With this test, we can know that correct query statements are being generated by the PICK system to query data from the db.				
Testing Team: Charlie Juarez Angelica Marquez - Lead Andrew Munoz Aaron Rodriguez			Date Completed: N/A	

4.11 Test TSGC OC1

Objective: Graph accurately displays the specified nodes and relationships.

Test No.: TSGC OC1			Current Status: Pending	
Test title: Graph node/relationship test				
Testing approach: This test will be conducted using the PICK system running on a Kali Linux machine. Behavior is observed on the Kali Linux screen.				
STEP 1	OPERATOR ACTION The tester selects "view graph" button after specifying nodes and their relationships.	PURPOSE Determine if the Graph displays the nodes and relationships.	EXPECTED RESULTS The graph is displayed with nodes and relationships.	
2	The tester checks if all the specified nodes are displayed accurately along with all the specified relationships. Relationship lines from parents to children must have arrow pointing to child.	Determine if the graph nodes and relationships are accurately displayed.	The nodes and relationships match to what was specified. Relationship lines from parents to children must have arrow pointing to child.	
Concluding Remarks:				
Testing Team: Charlie Juarez Angelica Marquez - Lead Andrew Munoz Aaron Rodriguez			Date Completed: N/A	

4.12 Test TSGC OC2

Objective: Graph is exported to the format specified by the user.

Test No.: TSGC OC2	Current Status: Pending
Test title: Graph Export Test	

Testing approach: This test will be conducted using the PICK system running on a Kali Linux machine. Behavior is observed on the Kali Linux screen.				
STEP 1	OPERATOR ACTION The tester selects "export graph" in the graph view window.	PURPOSE Initiate export graph process.	EXPECTED RESULTS The export graph dialog is displayed with option of exporting as XML.	
2	The tester selects to export as XML and specifies file location and name.	Determine if XML file of graph is exported.	XML file of the graph is exported to specified folder with specified name.	
3	The tester checks the exported file to see if it is an XML file.	Determine if the graph was exported in the specified XML format.	The exported file is in XML format with ".xml" filename extension.	
Concluding Remarks:				
Testing Team: Charlie Juarez Angelica Marquez - Lead Andrew Munoz Aaron Rodriguez	Date Completed: N/A			

4.13 Test TSGC OC3

Objective: Graph displays content associated with the specified vector.

Test No.: TSGC OC3	Current Status: Pending
Test title: Graph display associated vector test	
Testing approach: This test will be conducted using the PICK system running on a Kali Linux machine. Behavior is observed on the Kali Linux screen.	

STEP	OPERATOR ACTION	PURPOSE	EXPECTED RESULTS	COMMENTS
1	The tester selects “view graph” button after specifying nodes and their relationships within a vector in Vector table view.	Determine if the Graph displays the nodes and relationships.	The graph is displayed with nodes and relationships.	
2	The tester checks if all the specified nodes are within the vector.	Determine if the graph content of nodes and relationships are associated with the specified vector	The nodes and relationships match to what is within the specified vector.	
Concluding Remarks:				
Testing Team: Charlie Juarez Angelica Marquez - Lead Andrew Munoz Aaron Rodriguez		Date Completed: N/A		

4.14 Test TSGC OC4

Objective: Graph accurately displays the chosen visible attributes of a node.

Test No.: TSGC OC4	Current Status: Pending
Test title: Graph display associated vector test	
Testing approach: This test will be conducted using the PICK system running on a Kali Linux machine. Behavior is observed on the Kali Linux screen.	

STEP	OPERATOR ACTION	PURPOSE	EXPECTED RESULTS	COMMENTS
1	The tester selects “view graph” button after specifying the visible	Determine if the Graph displays the specified nodes and relationships.	The graph is displayed with nodes and relationships.	

	attributes of a node within Vector table view.			
2	The tester checks if all the selected visible node attributes appear on the graph.	Determine if the graph accurately displays the node's specified attributes.	The graph displays all the selected visible node attributes and does not display any selected hidden node attributes.	
Concluding Remarks:				
Testing Team: Charlie Juarez Angelica Marquez - Lead Andrew Munoz Aaron Rodriguez		Date Completed: N/A		

5. User Interface Testing

5 Test User Interface 1

Objective: Test flexibility of navigation

Test No.: Interface 1	Current Status: Pending
Test title: Test flexibility of navigation.	
Testing approach: Test if the user can change windows from Ingestion, to Navigation, to Graph.	

STEP	OPERATOR ACTION	PURPOSE	EXPECTED RESULTS	COMMENTS
1	The tester opens the system.	Determine if "Project Configuration" Window is display.	The program should display "Project configuration"	
2	Input "Events" and Log Entries, select "Configuration Complete"	Determine if there is a consistent transition between windows.	The System should display "Cleansing Overview" with Cleansing and Validation overview	
3	Select "Navigation" button	Determine if there is a consistent transition between windows.	The System should display the "Navigation" window.	Notice that filter buttons should be easily accessible at "Navigation".
4	Select "Graph" view	Determine if there is a consistent transition between windows.	The System should display the "Graph" window.	
5	Select "Navigation" button	Determine if "Navigation" window has an easy transition to "Graph" window.	The System should display the "Navigation" window.	

Concluding Remarks:	
Testing Team: Charlie Juarez Angelica Marquez - Lead Andrew Munoz Aaron Rodriguez	Date Completed: N/A

6. Test Schedule

The following table contains the testing order in which the tests will be completed. The table contains the task and date on the left column for when and what to be completed. Along with the respective people who will be responsible for completing the task, appended by a description of the test responsibility.

Task and date	People	Description
TSC OC1-2	Charlie Juarez	Test TSC OC1 and OC2, test directory selection to retrieve log files.
TSI OC1-OC3	Charlie Juarez	Test TSI OC1 to OC3, test ingestion of log entries
TSV OC1-OC4 5/2/2020	Charlie Juarez, Aaron Rodriguez	Test TSC OC1 to OC4, test the cleansing of the script, and validation of the log files
TSFC OC1-OC7 5/3/2020	Mriam Juarez, Angelica Marquz	Test TSFC OC1 to OC7, test filter control, by team, time, and location
TSDM OC1-4 5/3/2020	Aaron Rodriguez, Angelica Marquez	Test Database Management Test Suite (TSDM OC1-4)
TSGC OC1-2 5/4/2020	Andrew Munoz, Charlie Juarez	Test TSG OC1 and OC2, the graph displaying nodes and relationships, and the graph export format.
TSG OC3-4 5/4/2020	Aaron Rodriguez, Angelica Marquez	Test TSG OC3 and OC4, the graph displaying specified vector, and the graph visible attributes.

7. Other Sections

N/A

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. >>

A.

1. Contract Run Script

Contract number: 1
Contract name: run script
Contract description: runs the cleansing script on the given log file.
Protocol 1:
    Signature: runScript(filePath)
        input filePath- log file directory path to run script on, type string.
        output - none
    Pre-Cond: the given filePath is the correct directory path and exists.
    Post-Cond: the cleansing script is executed on a given log file.

2. Contract Log Ingestion

Contract number: 2
Contract name: Log Ingestion
Contract description: Given a log file, the class will return the given log type, the log content and if there are any ingestion errors.
    Protocol 1:
        Signature: logIngest(anyTypeLog)
            input anyTypeLog - Log file of any type.
            output fileType - log file type (i.e. csv, jpeg, pdf, etc…)
            output logContent - data information from the log file.
            output ingestionErrors - information of an error if it exists.
        PreCond: Log file has to exist.
            Post-Cond: Log file stays unchanged.

3. Contract Ingest Data Selection

Contract number: 3
Name: Ingest Data Selection
Contract description:
    Protocol 1:
        Signature: ingestDataSelection(validLogFile)
            input validLogFile- files that are validated.
        output: none
        Pre-Condn: the given LogFile has to be previously validated.
        Post-Condition: LogFiles are now LogEntries.

4. Contract Create Error Report

Contract number: 4
Contract name: Create error report
Contract description: Creates an enforcement action report at the detection of an error in a log file.
    Protocol: 1
        Signature: createErrorReport(logFileInfo, errorType)

input logFileInfo: Log file information to display in the enforcement action report, so that the analyst can know which log file is invalidated and must be changed or approved as is.
input errorType: The error type(s) associated with the invalidated log file. The error type(s) enables the analyst to see which validation rules the associated log file broke.
output: Enforcement Action Report
Pre-Cond: Log file must not be null and error type must be a valid error for generating an enforcement action report.
Post-Cond: Log file must remain unchanged.

5. Contract Directory Chooser

Contract number: 5
Contract name: Directory Chooser
Contract description: Display a file chooser window that will select the directory for log data.
Protocol 1:
Signature: dir_dialog(self, dialogtitle)
self-identify method as an instance of the class DirDialog
dialogtitle- name for the dialog that will be created.
output - none
Pre-Cond: a dialogtitle has been given
Post-Cond: the instance saves the file directory path

6. Contract Save Log Entry

Contract number: 6
Name: Save Log Entry
Contract description: Saves the log entry into the MongoDB
Protocol 1:
Signature: store_logentry(logentry)
logentry: the Log Entry to store in the MongoDB
Output: None
Pre-Condition: Log Entry cannot be null and the log entry table in MongoDB cannot be null
Post -Condition: Log Entry is stored in the MongoDB table log entry

7. Contract Create Log Entry DB Table

Contract number: 7
Name: Create Log Entry DB Table
Contract description: Creates a table in the MongoDB called log entry where all Log Entry instances will be stored.
Protocol 1:
Signature: create_table_logentry()
Output: a boolean that is True if the DB table creation was successful and false if it was not successful
Pre-Condition: The MongoDB must not be null

8. Display Directory

Contract number: 8
Contract description: Display the directory chosen
Protocol 1:
Signature: on_pathdir_button_clicked(self)
Output: None
Pre-Condition: A directory has been chosen by DirDialog.
Post -Condition: Display the directory that was chosen.

9. Display Start Date

Contract number: 9
Contract description: Display start date
Protocol 1:
Signature: on_startdate_button_clicked(self)
Output: None
Pre-Condition: A date has been chosen by CalendarDialog.
Post -Condition: Display the date that has been chosen.

10. Display End Date

Contract number: 10
Contract description: Display end date
Protocol 1:
Signature: on_enddate_button_clicked(self)
Output: None
Pre-Condition: A date has been chosen by CalendarDialog.
Post -Condition: Display the date that has been chosen.

11. Select Date

Contract number: 11
Contract description: Select date
Protocol 1:
Signature: on_confirm_button_clicked(self)
Output: None
Pre-Condition: A date has been selected in CalendarDialog.
Post -Condition: Set the date picked for its caller to retrieve from.

12. Select Directory

Contract number: 12
Contract description: Select directory
Protocol 1:
Signature: dir_dialog(self, dialogtitle)
Output: filename
Pre-Condition: A date has been selected in DirDialog.
Post -Condition: Set the directory picked for its caller to retrieve from.

13. Process Filter Request

Contract number: 13
Contract Name: Process Filter Request
Contract Description: Processes the user's filter request on the log entries.
Protocol 1:
Signature: ProcessFilter(filterRequest)
filterRequest: list of the user's requested filtering requirements.
Output: list of all log entries which pass the filtering requirements.
Pre-Condition: At lease one log entry exists in the system.
Post-Condition: The current full list of log entries is filtered to the user's request.

14. Display Results

Contract number: 14

Contract Name: Display Results
Contract Description: Displays the user’s filter/search request results.
Protocol 1:
Signature: ResultsButtonClicked(self)
Output: None
Pre-Condition: At least one filtering request has been chosen by the user.
Post-Condition: Display the resulting list of log entries from the filtering request.

15. Create Vector

Contract number: 15
Contract Name: Create Vector
Contract Description: Creates a vector.
Protocol 1:
Signature: createVector(Nodes, Name ,Description)
Nodes: list of the vector’s nodes.
Name: String of the vector name.
Description: String of vector’s description.
Output: none
Pre-Condition: Vector name is unique, and vector is comprised of at least one significant log entry.
Post-Condition: Vector with given attributes is created.
Protocol 2:
Signature: createVector(Nodes, Name)
Nodes: list of the vector’s nodes.
Name: String of the vector name.
Output: none
Pre-Condition: Vector name is unique, and vector is comprised of at least one significant log entry.
Post-Condition: Vector with given attributes is created and the vector description is an empty string.

16. Create Node

Contract number: 16
Contract Name: Create Node
Contract Description: Creates Node
Protocol 1:
Signature: createNode(id, name, time, description, ref, creator, event, icon, source, visibl, visFlags)
ID: String of the Nodes’s unique ID
Name: String of the node name.
Time: Timestamp of when the log entry took place.
Description: String of nodes description
Ref: The node’s log entry reference
Creator: String of node’s creator type showing the team who created the log (White, blue, or red).
Event: String of node’s event type showing the team of who executed the activity (White, blue, or red)
Icon: String of node’s icon type showing path to an image used to reflect the nature of the activity.
Source: String of the node’s source.
Visibl: Boolean of the node’s visibility, stating if the node is visible on the graph.
visFlag: Boolean list of size 9 determining the visibility of each node attribute on the graph as follows – id, name, time, description, , ref, creator, event, icon, and source
Output: none
Pre-Condition: The id is unique and the given creator and event are either “white”, ”red”, or ”blue”. Time is the timestamp in zulu time format HH:MM MM/DD/YY AM/PM.
Post-Condition: The node is created with its given attributes.
Protocol 2:
Signature: createNode(id, name, time, description, creator, event, icon, visibl, visFlags)

ID: String of the Nodes's unique ID
Name: String of the node name.
Time: Timestamp of when the log entry took place.
Description: String of nodes description
Creator: String of node's creator type showing the team who created the log (White, blue, or red).
Event: String of node's event type showing the team of who executed the activity (White, blue, or red)
Icon: String of node's icon type showing path to an image used to reflect the nature of the activity.
Visibl: Boolean of the node's visibility, stating if the node is visible on the graph.
visFlag: Boolean list of size 9 determining the visibility of each node attribute on the graph as follows – id, name, time, description, , ref, creator, event, icon, and source
Output: none
Pre-Condition: The id is unique and the given creator and event are either "white", "red", or "blue". Time is the timestamp in zulu time format HH:MM MM/DD/YY AM/PM.
Post-Condition: The node is created with its given attributes and both ref and source are left empty.

Protocol 3:
Signature: createNode(id, name, time, description , creator, event, icon, source, visibl, visFlags)
ID: String of the Nodes's unique ID
Name: String of the node name.
Time: Timestamp of when the log entry took place.
Description: String of nodes description
Creator: String of node's creator type showing the team who created the log (White, blue, or red).
Event: String of node's event type showing the team of who executed the activity (White, blue, or red)
Icon: String of node's icon type showing path to an image used to reflect the nature of the activity.
Source: String of the node's source.
Visibl: Boolean of the node's visibility, stating if the node is visible on the graph.
visFlag: Boolean list of size 9 determining the visibility of each node attribute on the graph as follows – id, name, time, description, , ref, creator, event, icon, and source
Output: none
Pre-Condition: The id is unique and the given creator and event are either "white", "red", or "blue". Time is the timestamp in zulu time format HH:MM MM/DD/YY AM/PM.
Post-Condition: The node is created with its given attributes and ref is left empty.

Protocol 4:
Signature: createNode(id, name, time, description, ref, creator, event, icon, visibl, visFlags)
ID: String of the Nodes's unique ID
Name: String of the node name.
Time: Timestamp of when the log entry took place.
Description: String of nodes description
Ref: The node's log entry reference
Creator: String of node's creator type showing the team who created the log (White, blue, or red).
Event: String of node's event type showing the team of who executed the activity (White, blue, or red)
Icon: String of node's icon type showing path to an image used to reflect the nature of the activity.
Visibl: Boolean of the node's visibility, stating if the node is visible on the graph.
visFlag: Boolean list of size 9 determining the visibility of each node attribute on the graph as follows – id, name, time, description, , ref, creator, event, icon, and source
Output: none
Pre-Condition: The id is unique and the given creator and event are either "white", "red", or "blue". Time is the timestamp in zulu time format HH:MM MM/DD/YY AM/PM.
Post-Condition: The node is created with its given attributes and source is left empty.

17. Create Relationship

Contract number: 17
Contract Name: Create Relationship
Contract Description: Creates a Relationship

Protocol 1:

Signature: createRelationship(id, parent, child, label)
id: String of the relationship's unique ID.
Parent: String of the parent ID, node ID of the source node of the relationship.
Child: String of the child ID, node ID of the destination node of the relationship.
Label: String of the label, a description of the relationship between the source and destination nodes.
Output: none.
Pre-Condition: Parent ID cannot be the same as child ID and relationship ID must be unique.
Post-Condition: The relationship is created with the given attributes.

18. Display Updated Table View

Contract number: 18
Contract Name: Display Updated Table View
Contract Description: Displays the user's filter/search request results.
Protocol 1:

Signature: updateTableView(self)
Output: None
Pre-Condition: At least one modification has been made by the user.
Post-Condition: Display the updated table views with the modification made.

19. Create Graph

Contract number: 19
Contract Name: Create Graph
Contract Description: Creates Graph
Protocol 1:

Signature: createGraph(vectorOrientation, nodesPosition, relationPosition, interval, intervalUnits)
vectorOrientation: String with orientation, ID of the node to which it points to.
nodesPosition: String with node position by grid.
relationPosition: String of ID related nodes.
interval: String with interval of marks on a timeline.
intervalUnits: seconds, minutes, hours, days, weeks.
Output: graph
Pre-Condition: Information on data base.
Post-Condition: Graph is created with relationships.

20. Display Graph Window

Contract number: 20
Contract Name: Display Graph Window
Contract Description: Display Graph Window
Protocol 1:

Signature: displayGraphWindow(graph)
graph: Structure with graph information.
Output: graph displayed in window.
Pre-Condition: A pre-build graph to be displayed.
Post-Condition: Graph is created with relationships.

21. Export Graph

Contract number: 21
Contract Name: Export Graph
Contract Description: Export Graph
Protocol 1:

Signature: exportGraph(graph, exportFormat)
graph: Structure with graph information.
exportFormat: String with export format information, options are JPG, csv.
Output: JPG image.
Pre-Condition: A pre-build graph to be displayed.
Post-Condition: None

$$$$$$