­­­PMR Insight Collective K­­­­nowledge (PICK)

Software Design Document

Version 3.5

5/3/2020

Document Control

Approval

The Guidance Team and the customer shall approve this document.

Document Change Control

|  |  |
| --- | --- |
| Initial Release: | 1.0 |
| Current Release: | 3.5 |
| Indicator of Last Page in Document: | $$$ |
| Date of Last Review: | 5/6/2020 |
| Date of Next Review: | 5/7/2020 |
| Target Date for Next Update: |  |

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. Steven Roach

Jake Lasley

Customer: Mr. Baltazar Santaella

Ms. Herandy Vasquez

Mr. Vincent Fonseca

Ms. Florencia Larsen

Mr. Eric de Nava

Software Team Members: Itzel Rivas

Eder Rodriguez

Jose Gallardo

Gilbert Alvarez

Nusrat Sarmin

Change Summary

The following table details changes made between versions of this document

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Modifier | Description |
| 1.0 | 2/27/2020 | Gilbert Alvarez | Creation of Document |
| 1.1 | 2/28/2020 | Gilbert Alvarez | Purpose and Intended Audience |
| 1.1.1 | 2/28/2020 | Itzel Rivas | Scope and helped with purpose |
| 1.2 | 2/29/2020 | Jose Gallardo | Definitions, Acronyms, Abreviation |
| 1.2.1 | 2/29/2020 | Nusrat Sarmin | Definitions |
| 1.3 | 3/4/2020 | Gilbert Alvarez | Overview and V1 of Wriffs-Brock diagram. |
| 1.3.1 | 3/4/2020 | Eder Rodriguez | Modified Wriffs-Brock diagram |
| 1.4 | 3/5/2020 | Gilbert, Eder, Itzel, Jose, Nusrat | Worked on improving CRC |
| 1.4.1 | 3/5/2020 | Gilbert Alvarez | Subsystem and comment descriptions |
| 1.5 | 3/5/2020 | Jose Gallardo | Detailed Description of components (user subsystem) |
| 1.5.1 | 3/5/20 | Eder Rodriguez | Edited System Component descriptions |
| 1.6 | 3/6/2020 | Eder Rodriguez | Described Dependencies |
| 1.7 | 3/7/2020 | Gilbert Alvarez | Modified Wriffs-Brock diagram and added description |
| 1.8 | 3/7/2020 | Itzel Rivas | 3.3 User interface Subsystem |
| 1.8.1 | 3/7/2020 | Itzel Rivas | 3.4 Ingestion/Validation/Cleansing Subsystem |
| 1.9 | 3/8/2020 | Jose Gallardo | Contracts for Ingestion/Validation/Cleansing subsystem |
| 1.9.1 | 3/8/2020 | Gilbert Alvarez | Modify diagram again to have appropriate contract numbers, fixed contracts of classes in section 3 |
| 1.9.2 | 3/8/2020 | Gilbert Alvarez | Created Database Schema for Vector DB |
| 2.0 | 3/25/2020 | Gilbert Alvarez | Fixed Wriffs-Brock diagram |
| 2.1 | 3/26/2020 | Gilbert Alvarez,  Eder Rodriguez,  Itzel Rivas,  Jose Gallardo | reviewed section 3.0 protocols for subsystems  -Processes  -UI  -Managers |
| 2.2 | 3/29/2020 | Gilbert Alverez | Continued to fix section 3.0 |
| 2.3 | 3/29/2020 | Eder Rodriguez | Database schema (crow’s foot notation) |
| 3.0 | 4/30/2020 | Eder Rodriguez, Gilbert Alvarez | Worked on Collaboration Diagram |
| 3.1 | 4/30/2020 | Jose Gallardo | Worked on protocols for Managers |
| 3.2 | 5/3/2020 | Itzel Rivas | Worked on protocols for UI subsystem |
| 3.3 | 5/4/2020 | Gilbert Alvarez | Reworked section 2.2 |
| 3.4 | 5/4/2020 | Eder Rodriguez | Worked on protocols for Processes, Database |
| 3.5 | 5/7/2020 | Gilbert Alvarez | Reworked protocols for Models and reviewed entire SDD |

Table of Contents

[Document Control ii](#_Toc39913541)

[Approval ii](#_Toc39913542)

[Document Change Control ii](#_Toc39913543)

[Distribution List ii](#_Toc39913544)

[Change Summary ii](#_Toc39913545)

[1. Introduction 1](#_Toc39913546)

[1.1. Purpose and Intended Audience 1](#_Toc39913547)

[1.2. Scope of Product 1](#_Toc39913548)

[1.3. References 1](#_Toc39913549)

[1.4. Definitions, Acronyms, and Abbreviations 1](#_Toc39913550)

[1.4.1. Definitions 1](#_Toc39913551)

[1.4.2. Acronyms 2](#_Toc39913552)

[1.4.3. Abbreviations 2](#_Toc39913553)

[1.5. Overview 2](#_Toc39913554)

[2. Decomposition Description 4](#_Toc39913555)

[2.1. System Collaboration Diagram 4](#_Toc39913556)

[2.2. Subsystem and Component Descriptions 4](#_Toc39913557)

[2.2.1. Models 4](#_Toc39913558)

[2.2.2. Managers 4](#_Toc39913559)

[2.2.3. Processes 5](#_Toc39913560)

[2.2.4. Tools 5](#_Toc39913561)

[2.2.5. UI 5](#_Toc39913562)

[2.2.6. Database 6](#_Toc39913563)

[2.3. Dependencies 6](#_Toc39913564)

[3. Detailed Description of Component 7](#_Toc39913565)

[3.1. Models 7](#_Toc39913566)

[3.1.1. Node 7](#_Toc39913567)

[3.1.2. Log Entry 9](#_Toc39913568)

[3.1.3. Event Config 10](#_Toc39913569)

[3.1.4. Vector 14](#_Toc39913570)

[3.1.5. Log File 14](#_Toc39913571)

[3.2. Managers 15](#_Toc39913572)

[3.2.1. Event Config Manager 15](#_Toc39913573)

[3.2.2. Log File Manager 17](#_Toc39913574)

[3.2.3. Log Entry Manager 18](#_Toc39913575)

[3.2.4. Vector Manager 19](#_Toc39913576)

[3.2.5. Node Manager 20](#_Toc39913577)

[3.3. Processes 21](#_Toc39913578)

[3.3.1. Cleansing 21](#_Toc39913579)

[3.3.2. Ingestion 22](#_Toc39913580)

[3.4. Tools 23](#_Toc39913581)

[3.4.1. Transcription Tool 23](#_Toc39913582)

[3.4.2. Optical Character Reader 23](#_Toc39913583)

[3.4.3. Splunk 23](#_Toc39913584)

[3.5. UI 23](#_Toc39913585)

[3.5.1. Team Configuration UI 23](#_Toc39913586)

[3.5.2. Vector Configuration UI 24](#_Toc39913587)

[3.5.3. Directory Configuration UI 24](#_Toc39913588)

[3.5.4. Event Configuration UI 25](#_Toc39913589)

[3.5.5. Action Report View 25](#_Toc39913590)

[3.5.6. Processing View 25](#_Toc39913591)

[3.5.7. Analysis View 26](#_Toc39913592)

[3.5.8. Graph 27](#_Toc39913593)

[3.6. Database 28](#_Toc39913594)

[3.6.1. Database 28](#_Toc39913595)

[4. Database 29](#_Toc39913596)

[4.1. Database Schema 29](#_Toc39913597)

[5. Appendix 30](#_Toc39913598)

[5.1. Notes 30](#_Toc39913599)

# Introduction

## Purpose and Intended Audience

The purpose of the Software Design Document (SDD) is to identify collaborations and subsystems within the PMR Insight Collective Knowledge (PICK) software to be able to design interactions within the system. The SDD describes what is to be built and is intended to help the development team understand what is expected to be built. Sections within the SDD include the system and subsystem collaboration graphs, component descriptions, descriptions of the classes within the components, contracts between classes, and description of database schemas. The SDD focuses on describing contracts of classes which are client server relationships that occur in order to fulfill a client’s responsibilities, and protocols of classes that state each method in a class, parameters of the methods, return value of the method, purpose of the method and pre and post conditions of a method.

## Scope of Product

This Software Design Document focuses on the core functionality of the PICK system and its main components, which is to streamline the job of an analyst to create a report from log files submitted by the defense and attacking teams. The system will utilize outside libraries and pre-existing systems, described on section 2.3, to achieve artifact ingestion, validation, cleansing and creating a visual representation of significant events for the analyst to see. Our goal is to be able to aid analysts in the creation of a report from any form of artifacts containing information of an attack response situation.

## References

[1] 08c\_SDD\_Outline.pdf. “System Design Document Outline”. 18, Feb 2020. https://piazza.com/class\_profile/get\_resource/k55o3l3qykt5ez/k6qs06oynxn2wt

[3] 08aContracts.pdf. “Contracts”. 18 Fed, 2020. https://piazza.com/class\_profile/get\_resource/k55o3l3qykt5ez/k6qs05yplls2v2

[4] 08b\_Protocols.pdf. “Protocols”. 18 Feb, 2020. https://piazza.com/class\_profile/get\_resource/k55o3l3qykt5ez/k6qs0681en62vv

[5] 09Subsystems.pdf. “Subsystems”. 10 Mar, 2020. https://piazza.com/class\_profile/get\_resource/k55o3l3qykt5ez/k6qs06wo8d92xi

[6] Hasneeza. “Example of Software Design Document (SDD).” UniMAP Portal, 26 Apr. 2012, portal.unimap.edu.my/portal/page/portal30/Lecture Notes/KEJURUTERAAN\_KOMPUTER/Semester 2 Sidang Akademik 20112012/EKT420 Software Engineering/Example of Software Design Document (SDD).

[7] 07aCollaborations.pdf. “Collaborations”. 13, Feb 2020. https://piazza.com/class\_profile/get\_resource/k55o3l3qykt5ez/k6i6k6zcief3kz

## Definitions, Acronyms, and Abbreviations

### Definitions

|  |  |
| --- | --- |
| **Lead Analyst** | The analyst responsible for the oversight of work being performed by the other analysts, some other responsibilities include delegating work, and approving work |
| **Analyst** | A user that is authorized to work PICK system, they must connect to the Lead analysts' machine to gain access to the log files and databases. Some of their responsibilities include editing, creating, deleting log entries/nodes, and pushing work to the Lead analyst for approval upon synchronization. |
| **Log File** | A file that records events from a team users' machine. |
| **Log Entry** | A file that records events from a team users' machine. |
| **Log Creator** | Node attribute that states what team created the log. |
| **Node** | Visual representation of log entry on a vector graph. |
| **Node name** | Node attribute, unique name of a node. |
| **Start Date** | Start date and time of Assessment |
| **End Date** | End date and time of Assessment |
| **Timestamp** | Time the activity occurred. Recorded in Zulu time. |
| **Node ID** | Unique numerical ID created for nodes. |

### Acronyms

|  |  |
| --- | --- |
| **PICK** | PMR Insight Collective Knowledge |
| **PMR** | Prevent, Mitigate, Recover |
| **SDD** | Software Design Document |
| **UML** | Unified Modeling Language |
| **OCR** | Optical Character Reader |

### Abbreviations

|  |  |
| --- | --- |
| **DB** | Database |
| **config** | Configuration |
| **e.g.** | For example, |
| **i.e.** | in other words, |

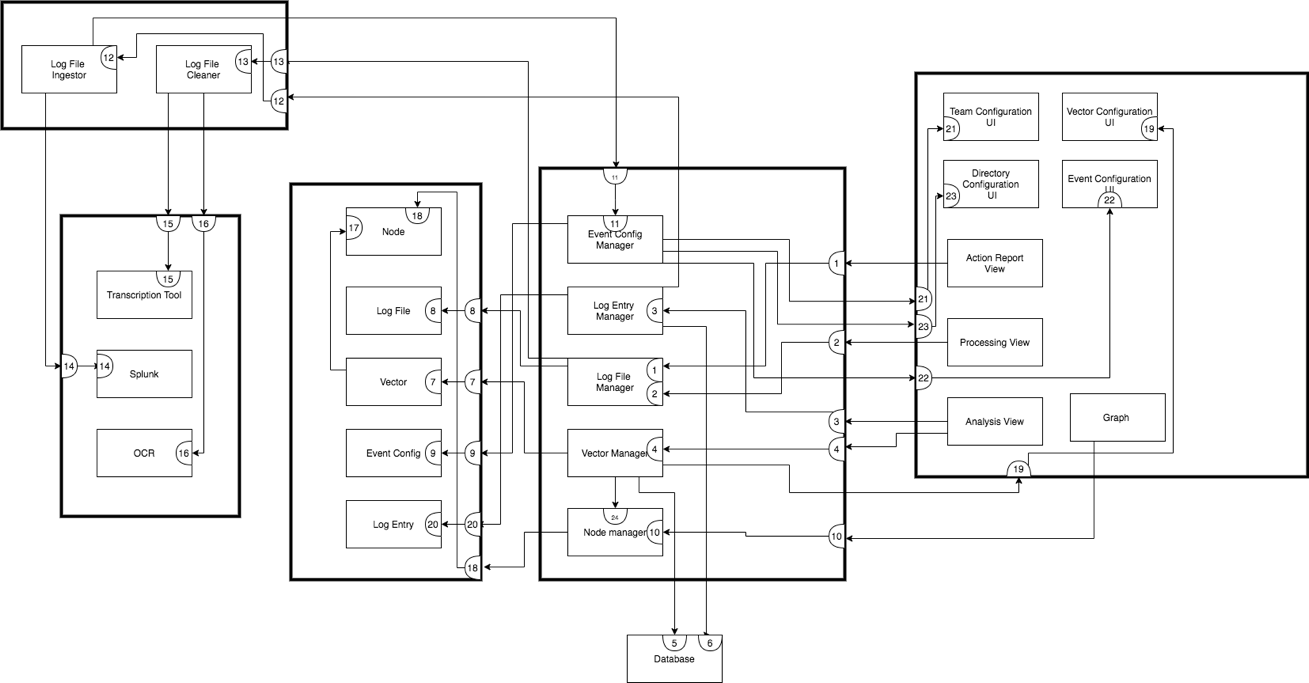
## Overview

The following section describes the components and the interactions of the subsystems in the PICK system using Wriffs-Brocks diagram. Wriffs-Brock diagram utilizes rounded squares to indicate classes and semicircles with numbers and arrows pointing from one class to another to indicate a contract between two classes. Subsystems on the Wriffs-Brock diagram are indicated with a larger rectangular container surrounding classes. This section also describes the design choices used for dividing design entities.

# Decomposition Description

This section describes the subsystems of the system and the collaboration between the components.

## System Collaboration Diagram



*figure 1.0*

The system can be broken down into 6 major subsystems. UI, Models, Tools, Managers, Processes and Database.

## Subsystem and Component Descriptions

### Models

**Description:** This subsystem focuses on the data attributes held by objects in the system. The classes that are part of the models subsystem are event config, log entry, log file, node and vector.

**Purpose:** To access data of the objects, such as getting a vectors name and description to display on the UI. To update an objects data, such as updating the ingestion, cleansing, and validation status of log files. To create objects containing data, such as creating log entries that hold their name, time, content, source and sources type.

**Contracts:**

* 17 (described in section 3.1.4)

### Managers

**Description:** This subsystem allows for the interaction between the models and the rest of the system, such as the UI subsystem and the Processes subsystem. Its classes include log entry manager, log file manager, event config manager and vector manager.

**Purpose:** To allow the system to interact with the models.

**Contracts:**

* 5 (described in section 3.2.4)
* 6 (described in section 3.2.3)
* 7 (described in section 3.2.4)
* 8 (described in section 3.2.2)
* 9 (described in section 3.2.1)
* 12 (described in section 3.2.3)
* 13 (described in section 3.2.2)
* 18 (described in section 3.2.5)
* 19 (described in section 3.2.4)
* 20 (described in section 3.2.3)
* 21 (described in section 3.2.1)
* 22 (described in section 3.2.1)
* 23 (described in section 3.2.1)

### Processes

**Description:** The processes subsystem gathers log files from the system to cleanse, validate and ingest them back into the system as log entries. Classes associated with this subsystem include Log File Ingestor and Log File Cleaner.

**Purpose:** Log File Cleaner removes empty lines from log files and interacts with the OCR and Transcription tool to transcribe image and audio files to text files. The Log file Ingestor then calls Splunk sends the log files for ingestion then receives the results for validation.

**Contracts:**

* 11 (described in section 3.3.2)
* 14 (described in section 3.3.2)
* 15 (described in section 3.3.1)
* 16 (described in section 3.3.1)

### Tools

**Description:** The tools subsystem is comprised of Splunk, OCR and transcription tool. The system interacts with tools through the processes class. After a log file has been cleansed and validated, the system calls the tools to transform the data to be ingestible. The OCR tool is used to transcribe image files to text files that can be used by Splunk. The transcription tool transcribes audio files to text files that can be used by Splunk.

**Purpose:** To transform log files into log entries, and transcribe log files that are images or audio files to text files that can be ingested by Splunk.

**Contracts:**

* The tools don’t call other methods from other classes and don’t collaborate with classes, instead other classes collaborate with the tools to fulfill their responsibilities.

### UI

**Description:** This subsystem is used to interact with the user by displaying the ingestion, cleansing and validation process, displaying the graph view and log entry view, display the action report view and asking the user for input. Classes that are a part of this subsystem are the directory configuration widget, event configuration widget, team configuration widget, vector configuration widget, analysis view, action report view, and processing view, graph manager, graph generator, and graph widget.

**Purpose:** To allow the user to interact with the system by taking input and providing a visual representation of log entries and nodes on a graph view, list of vectors and their log entries, and the process of log files being ingested, cleansed and validated.

**Contracts:**

* 1 (described in section 3.5.5)
* 2 (described in section 3.5.5)
* 3 (described in section 3.5.7)
* 4 (described in section 3.5.7)
* 10 (described in section 3.5.8)

### Database

**Description:** A database will be used to store information about the log files, vectors and log entries. Using a database will allow the user to quit the program and return at a later time to continue working on the assessment without losing the work they have performed.A mongo database will be used to store vectors and log entries created during the assessment.

**Purpose:** To store information of the log files, vectors and log entries. For vectors this includes storing its name and description, for log entries their number, source, source type, time stamp, creator and information of the log entry (content) are stored. For the log entry its cleansing, validation and ingestion statuses are stored along with its path.

**Contracts:**

* The database subsystem is called by the vector manager and log entry manager but does not collaborate with any component.

## Dependencies

The system runs using Python version 3 making the code portable with the ability to be able to run on the client's choice of operating system (Kali Linux). The graph, its nodes and its connections are drawn by utilizing the external library QGraphViz. The UI component uses an external library PyQt5 to draw user interface elements and create views such as the event configuration widget, team configuration widget, vector configuration widget, directory configuration widget, processing view, analysis view and action report view. External tools that are used by the system include Splunk and pytesseract to read an image and create a text file from recognizable text. Vector and log entry data are stored on a Mongo DB data base, this is achieved by using the python library pymongo.

# Detailed Description of Component

## Models

### Node

Represents a significant log entry and can be visually seen on a graph of a specific vector. This class collaborates with the log entry class to retrieve information about the log entry (seen in signature).

*Table 1.0*

|  |  |
| --- | --- |
| **Class Name**: Node | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**   1. Knows its ID number. 2. Knows its name. 3. Knows its time stamp. 4. Knows its description. 5. Knows its creator. 6. Knows its event type. 7. Knows its icon. 8. Knows its source. 9. Knows its visibility status. | |
| **Contract:** called by 18. (see section 3.2.5) and called by 17. (see section 3.1.4) | |
| **Responsibilities** | **Collaborations** |
|  |  |

**-Method Name:** setLogFilePath(path n)

**-Return Type:** void

**-Input & Output parameters:**

- **n:** string

-**Description of parameters:** the full directory path of the log file

**-Purpose:** set the path of the log file on the log file reference

**-Pre-Conditions:** user set the root directory location with the event config UI, there exists a log file

**-Post-Conditions:** the log file’s path attribute is updated to match the location of the log file

**-Method Name:** getLogFilePath()

**-Return Type:** void

**-Input & Output parameters:**

-**Description of parameters:**

**-Purpose:** to get the path of the log file

**-Pre-Conditions:** the log file has a path attribute set

**-Post-Conditions:** returns a string of the log file path attribute.

**-Method Name:** setID(id i)

**-Return Type:** void

**-Input & Output parameters:**

**-id:** integer

-**Description of parameters:** id is the unique id number assigned to the node

**-Purpose:** to set an id number to a node

**-Pre-Conditions:** a node must exist by being created by a log entry or by a user creating node from the GUI

**-Post-Conditions:** the attribute of id has been set to the node

**-Method Name:** getID()

**-Return Type:** integer

**-Input & Output parameters:**

-**Description of parameters:**

**-Purpose:** to retrieve the id attribute of a node

**-Pre-Conditions:** a node must exist and has an id attribute set

**-Post-Conditions:** the unique id of the node is returned

**-Method Name:** setTime(time t)

**-Return Type:** void

**-Input & Output parameters:**

**-t:** string

-**Description of parameters:** string is the corresponding time stamp of the node from the log entry or from user input

**-Purpose:** to set a time stamp to the node

**-Pre-Conditions:** a node must exist by being created by a log entry or by a user creating node from the GUI

**-Post-Conditions:** a string attribute of time has been set to a node

**-Method Name:** getTime()

**-Return Type:** void

**-Input & Output parameters:**

-**Description of parameters:**

**-Purpose:** to retrieve the time attribute of a node

**-Pre-Conditions:** a node must exist and has a time attribute set

**-Post-Conditions:** a string of the nodes time is returned

**-Method Name:** setDescription(description d)

**-Return Type:** void

**-Input & Output parameters:**

**-d:** string

-**Description of parameters:** string is the corresponding description of the node from the log entry or from user input

**-Purpose:** to set a description attribute to a node

**-Pre-Conditions:** a node must exist by being created by a log entry or by a user creating node from the GUI

**-Post-Conditions:** a string attribute of the description has been set to a node

**-Method Name:** getDescripotion()

**-Return Type:** string

**-Input & Output parameters:**

-**Description of parameters:**

**-Purpose:** to retrieve the description attribute of a node

**-Pre-Conditions:** a node must exist and have a description attribute set

**-Post-Conditions:** a string of the node’s description is returned

**-Method Name:** setIcon(icon ic)

**-Return Type:** void

**-Input & Output parameters:**

**-ic:** string

-**Description of parameters:** string leading to the icon

**-Purpose:** to set an icon to a node on a graph

**-Pre-Conditions:** a node must exist by being created by a log entry or by a user creating node from the GUI and the user has assigned to an icon to the node

**-Post-Conditions:** the icon attribute is set to a node

**-Method Name:** getIcon()

**-Return Type:** string

**-Input & Output parameters:**

-**Description of parameters:**

**-Purpose:** to retrieve the path to icon the node is associated with

**-Pre-Conditions:** a node must be created using the GUI by a user or by associated a log entry to a vector

**-Post-Conditions:** an icon representing a node appears on the graph view

**-Method Name:** setVisability(vis v)

**-Return Type:** void

**-Input & Output parameters:**

**-v:** boolean

-**Description of parameters:** true if the node is visible on the graph, false if the node is not visible on the graph

**-Purpose:** to toggle a node to be seen on the graph or to be hidden

**-Pre-Conditions:** a node must exist by being created by a log entry or by a user creating node from the GUI and the user has changed its visibility

**-Post-Conditions:** visibility attribute is set to the corresponding node

**-Method Name:** getIcon()

**-Return Type:** boolean

**-Input & Output parameters:**

-**Description of parameters:**

**-Purpose:** to hide or see a node on the graph view

**-Pre-Conditions:** a visibility attribute of the node has been set

**-Post-Conditions:** node is hidden on the graph if false or is seen on the graph if true

### Log Entry

This class holds information on a log entry object and can be accessed by other classes through the log entry manager class.

*Table 1.1*

|  |  |
| --- | --- |
| **Class Name**: Log Entry | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**  1. Knows its number.  2. Knows its time stamp.  3. Knows its content.  4. Knows its log file.  5. Knows its log file type. | |
| **Contract:** called by 20. (see section 3.2.3) | |
| **Responsibilities** | **Collaborations** |
|  |  |

**-Method Name:** setNumber(number n)

**-Return Type:** void

**-Input & Output parameters:**

**-n:** integer

-**Description of parameters:** integer attribute for a log entry

**-Purpose:** to set a number to a log entry

**-Pre-Conditions:** a log file must have been ingested into the system

**-Post-Conditions:** an integer number has been set to the node that is referenced

**-Method Name:** getNumber()

**-Return Type:** integer

**-Input & Output parameters:**

-**Description of parameters:** a reference to a node object

**-Purpose:** to retrievethe log entry number

**-Pre-Conditions:** a log file must have been ingested into the system and number attribute was set to it

**-Post-Conditions:** an integer number attribute for a log entry is returned

**-Method Name:** setTime(timestamp t)

**-Return Type:** void

**-Input & Output parameters:**

- **t:** string

-**Description of parameters:** time attribute from log entry in the zulu time, format “YYYY-MM-DD” for date and “HH:MM:SS” for time

**-Purpose:** to set a timestamp attribute to a log entry, which will be used to connect two nodes together on the graph view.

**-Pre-Conditions:** a log file has been ingested by Splunk into the system, there were no issues in cleansing, validation or ingestion

**-Post-Conditions:** the attribute of timestamp is set on a log entry

**-Method Name:** getTime()

**-Return Type:** string

**-Input & Output parameters:**

-**Description of parameters:**

**-Purpose:** to retrieve the timestamp of a log entry stored as “YYYY-MM-DD” for the date and “HH:MM:SS” for the time

**-Pre-Conditions:** a timestamp attribute must have been set for the log entry

**-Post-Conditions:** a string is returned representing the log entry’s timestamp attribute

**-Method Name:** setContent(content c)

**-Return Type:** void

**-Input & Output parameters:**

- **c:** string

-**Description of parameters:** description of the log entry stored as a string

**-Purpose:** to set the content attribute to a log entry, the content describes the single event the log entry represents taken from the log file

**-Pre-Conditions:** a log file has been ingested by Splunk into the system, there were no issues in cleansing, validation or ingestion

**-Post-Conditions:** the attribute of content is set on a log entry

**-Method Name:** getContent()

**-Return Type:** string

**-Input & Output parameters:**

-**Description of parameters:**

**-Purpose:** to retrieve the content of the log description for the user to see on the log entry table on the analysis view

**-Pre-Conditions:** a content attribute must have been set for the log entry

**-Post-Conditions:** a string is returned representing the log entry’s content attribute

### Event Config

This class holds the information of the current event based on user input from the UI. Information in class can be accessed by other classes through the event manager class.

*Table 1.2*

|  |  |
| --- | --- |
| **Class Name**: Event Config | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**  1. Knows its name.  2. Knows its description.  3. Knows its start time.  4. Knows its end time.  5. Knows the root directory location.  6. Knows the red team folder location.  7. Knows the blue team folder location.  8. Knows the white team folder location.  9. Knows who the lead analyst is.  10. Knows the lead analysts IP address.  11. Knows the analysts connected to the leads machine. | |
| **Contract:** called by 9, 21. (see section 3.2.1) | |
| **Responsibilities** | **Collaborations** |
|  |  |

**-Method Name:** setName(name n)

**-Return Type:** void

**-Input & Output parameters:**

- **n:** string

-**Description of parameters:** string input for event received from user input from event config UI

**-Purpose:** to set a name attribute for the event configuration

**-Pre-Conditions:** user has input name of event on the event config UI and has started the project

**-Post-Conditions:** the name attribute for an event has been assigned to the string from the user input

**-Method Name:** getName()

**-Return Type:** string

**-Input & Output parameters:**

-**Description of parameters:**

**-Purpose:** to retrieve the name attribute set to the event

**-Pre-Conditions:** user has named the event in the event configuration UI and the name has been set to the event

**-Post-Conditions:** the name attribute of the event has been returned as a string

**-Method Name:** setDescription(desc d)

**-Return Type:** void

**-Input & Output parameters:**

- **d:** string

-**Description of parameters:** a string describing the event

**-Purpose:** to set a description of the event

**-Pre-Conditions:** user has input a description of the event in the event configuration UI and has started the project

**-Post-Conditions:** the description attribute for the event has been set

**-Method Name:** getDescription()

**-Return Type:** string

**-Input & Output parameters:**

-**Description of parameters:**

**-Purpose:** to retrieve the description of the event

**-Pre-Conditions:** user has input a description for the event in the event configuration UI and the description has been set to the event

**-Post-Conditions:** a string of the description attribute for the event is returned

**-Method Name:** setStart(startDateTime s)

**-Return Type:**

**-Input & Output parameters:**

- **s:** string

-**Description of parameters:** start time for the event in format “YYYY-MM-DD HH:MM:SS”

**-Purpose:** to set a start time and date for the event, this will be used to sort out relevant log files that fall only within the time frame of the event

**-Pre-Conditions:** user has set a start ti

“, µme on the event configuration UI and pressed the save button

**-Post-Conditions:** a start time in the format “YYYY-MM-DD HH:MM:SS” in zulu time is set to the event

**-Method Name:** getStartTime()

**-Return Type:** string

**-Input & Output parameters:**

-**Description of parameters:**

**-Purpose:** to retrieve the start time attribute of the event to use in validation process

**-Pre-Conditions:** the start time attribute has been set by user input from the event configuration UI

**-Post-Conditions:** a string for the attribute start time is returned for the event configuration

**-Method Name:** setRoot(root r)

**-Return Type:** void

**-Input & Output parameters:**

**-r:** string

-**Description of parameters:** string of root directory path

**-Purpose:** store the location of all log files to be used for use in the system

**-Pre-Conditions:** a user has set a location from the GUI to the path of the root directory containing all log files

**-Post-Conditions:** a string for the root directory attribute has been set for the event configuration

**-Method Name:** getRoot()

**-Return Type:** string

**-Input & Output parameters:**

-**Description of parameters:**

**-Purpose:** retrieve the location of all log files to be used for use in the system

**-Pre-Conditions:** a user has set a location from the GUI to the path of the root directory containing all log files and has been set to the event

**-Post-Conditions:** a string for the root directory attribute is returned

**-Method Name:** setRed(redDir rd)

**-Return Type:** void

**-Input & Output parameters:**

**-rd:** string

-**Description of parameters:** string of red team directory path

**-Purpose:** store the location of all log files from the red team to be used for use in the system

**-Pre-Conditions:** a user has set a location from the GUI to the path of the red team’s directory containing all log files

**-Post-Conditions:** a string for the red team’s directory attribute has been set for the event configuration

**-Method Name:** getRed()

**-Return Type:** string

**-Input & Output parameters:**

-**Description of parameters:**

**-Purpose:** retrieve the location of all log files from the red team to be used for use in the system

**-Pre-Conditions:** a user has set a location from the GUI to the path of the red team’s directory containing all log files and has been set to the event

**-Post-Conditions:** a string for the red team’s directory attribute is returned

**-Method Name:** setBlue(blueDir b)

**-Return Type:** void

**-Input & Output parameters:**

**-b:** string

-**Description of parameters:** string of blue team’s directory path

**-Purpose:** store the location of all log files from the blue team to be used for use in the system

**-Pre-Conditions:** a user has set a location from the GUI to the path of the blue team’s directory containing all log files

**-Post-Conditions:** a string for the blue team’s directory attribute has been set for the event configuration

**-Method Name:** getBlue()

**-Return Type:** string

**-Input & Output parameters:**

-**Description of parameters:**

**-Purpose:** retrieve the location of all log files from the blue team to be used for use in the system

**-Pre-Conditions:** a user has set a location from the GUI to the path of the blue team’s directory containing all log files and has been set to the event

**-Post-Conditions:** a string for the blue team’s directory attribute is returned

**-Method Name:** setWhite(whiteDir w)

**-Return Type:** void

**-Input & Output parameters:**

**-w:** string

-**Description of parameters:** string of white team’s directory path

**-Purpose:** store the location of all log files from the white team to be used for use in the system

**-Pre-Conditions:** a user has set a location from the GUI to the path of the white team’s directory containing all log files

**-Post-Conditions:** a string for the white team’s directory attribute has been set for the event configuration

**-Method Name:** getWhite()

**-Return Type:** string

**-Input & Output parameters:**

-**Description of parameters:**

**-Purpose:** retrieve the location of all log files from the white team to be used for use in the system

**-Pre-Conditions:** a user has set a location from the GUI to the path of the white team’s directory containing all log files and has been set to the event

**-Post-Conditions:** a string for the white team’s directory attribute is returned

**-Method Name:** setWhite(leader l)

**-Return Type:** void

**-Input & Output parameters:**

**-l:** boolean

-**Description of parameters:** Boolean that is set true if the user is lead analyst and false if the user is not the lead

**-Purpose:** to allow a user to be the lead analyst of the assessment

**-Pre-Conditions:** a user has checked the option to mark them as the lead analyst on the team configuration UI.

**-Post-Conditions:** a Boolean value is set to true if a user checks if they are the lead

**-Method Name:** setIP(ipaddress i)

**-Return Type:** void

**-Input & Output parameters:**

**-i:** string

-**Description of parameters:** a string value to be used to set the lead analysts IP address

**-Purpose:** to allow the lead analyst to set their IP to allow others to connect to them and for analysts to connect to the lead analysts machine

**-Pre-Conditions:** a user has selected the option they are the lead analysts and have entered their IP address, an analyst has entered the lead analysts IP address from the team configuration UI

**-Post-Conditions:** a string value has been set to the event configuration that is the lead’s IP address.

**-Method Name:** getIP()

**-Return Type:** string

**-Input & Output parameters:**

-**Description of parameters:**

**-Purpose:** to retrieve the lead analyst IP address set to the event configuration and allow others to connect to the lead

**-Pre-Conditions:** the IP address of the lead has been set to the event configuration

**-Post-Conditions:** a string for the IP address attribute of the event configuration has been returned

### Vector

This class is used to associate log entries and nodes to a specific vector based on the description or association of the user by using the UI. It holds a list of log entries and nodes and can be accessed by other classes through the vector manager class.

*Table 1.3*

|  |  |
| --- | --- |
| **Class Name**: Vector | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**  1. Knows its name.  2. Knows its description.  3. Knows the nodes associated to it. | |
| **Contract:** 17. Associate nodes to vector | |
| **Responsibilities** | **Collaborations** |
| 1. add nodes to a vector | Nodes(1) |

**Contract 17**

**-Method Name:** addNode()

**-Return Type:** void

**-Input & Output parameters:**

-**Description of parameters:** a reference of a node object

**-Purpose:** create an association between nodes created from log entries and user created nodes to a vector.

**-Pre-Conditions:** a node has been created from a log entry or the user has created a new node.

**-Post-Conditions:** the node object is appended to a list of nodes that is associated to a vector.

### Log File

This class is used to hold information on log files.

*Table 1.4*

|  |  |
| --- | --- |
| **Class Name**: Log File | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**  1. Knows its name.  2. Knows its cleansing status.  3. Knows its validation status.  4. Knows its ingestion status.  5. Knows its acknowledgement status.  6. Knows its file path.  7. Knows what type of file it is. | |
| **Contract:** called by 8. (see section 3.2.2) | |
| **Responsibilities** | **Collaborations** |
|  |  |

## Managers

### Event Config Manager

This class handles getting input from the user for the event from the UI. It also allows other classes to access information of the event config class through it.

*Table 2.0*

|  |  |
| --- | --- |
| **Class Name**: Event Config Manager | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**  1. Store name of event.  2. Store description of event.  3. Store start date and time of event.  4. Store end date and time of event.  5. Store root directory location.  6. Store red directory location.  7. Store blue directory location.  8. Store while directory location.  9. Return instance of an event.  10. Stores the lead analyst of the event.  11. Stores the IP address of the lead’s machine.  12. Stores the connections to the leads machine. | |
| **Contract:** 9. Create event | |
| **Responsibilities** | **Collaborations** |
| 1. Store an event from name, description, start and end time.  2. Store the lead, IP address and connections associated with the event.  3. Store the root directory location, as well as red, blue, white team’s folder location. | Event Config (1,2,3) |
| **Contract:** 21. Get team configuration user input from UI. | |
| **Responsibilities** | **Collaborations** |
| 1. Get event information. | Event Configuration UI(1) |
| **Contract:** 22. Get event configuration user input from UI. | |
| **Responsibilities** | **Collaborations** |
| 2. Get team configuration. | Team Configuration UI(1) |
| **Contract:** 23. Get directory location information user input from UI. | |
| **Responsibilities** | **Collaborations** |
| 1. Get directory information | Directory Configuration UI(1) |

**Contract 9**

**-Method Name:** setEventAttributes(name n, description d, startTime s, endTime e)

**-Return Type:** void

**-Input & Output parameters:**

- **n:** string input

- **d:** string input

- **s:** string input

- **e:** string input

-**Description of parameters:** user input strings from UI text fields. Strings for name, description, start time and end time.

**-Purpose:** store the attribute of the current event

**-Pre-Conditions:** user has filled in and saved from the event configuration UI.

**-Post-Conditions:** name, description, start and end time of an event have been set to an event object.

**-Method Name:** setTeamAttributes(lead x, ip a, connection c)

**-Return Type:** void

**-Input & Output parameters:**

- **x:** boolean

- **a:** string

- **c:** list

-**Description of parameters:** Boolean set to the lead analysts’ machine, IP addressed saved as a string of the leads machine, list of machines connected to leads machine.

**-Purpose:** to save information of the team of the current assessment.

**-Pre-Conditions:** lead analyst entered IP address and specified they were the lead on the event configuration GUI, and there were users who connected to the leads machine.

**-Post-Conditions:** the lead their IP and connections to the lead’s machine have been set to an event object.

**-Method Name:** setDirAttributes(rootdir r, rdir rd, bdir b, wdir w)

**-Return Type:** void

**-Input & Output parameters:**

- **r:** string input

- **rd:** string input

- **b:** string input

- **w:** string input

-**Description of parameters:** string of the file paths of the root directory, red team folder location, blue team folder location, white team folder location.

**-Purpose:** to save information of the directory and folders containing the log files of the red, blue and white team.

**-Pre-Conditions:** the user must input the locations in the GUI by browsing or typing, file string path must lead to correct directory.

**-Post-Conditions:** the directory locations (file paths) have been set to an event object.

**Contract 21**

**-Method Name:** getTeamInfo(lead x, ip a, connection c)

**-Return Type:** void

**-Input & Output parameters:**

- **x:** boolean

- **a:** string

- **c:** list

-**Description of parameters:** Boolean set to the lead analysts’ machine, IP addressed saved as a string of the leads machine, list of machines connected to leads machine.

**-Purpose:** to get information of the team of the current assessment from the GUI.

**-Pre-Conditions:** the user must start a new project and enter information in the team input fields.

**-Post-Conditions:** the attributes for the lead, lead IP and connections are retrieved from the UI and ready to be set to the event config

**Contract 22**

**-Method Name:** getEventInfo(name n, description d, startTime s, endTime e)

**-Return Type:** void

**-Input & Output parameters:**

- **n:** string input

- **d:** string input

- **s:** string input

- **e:** string input

-**Description of parameters:** user input strings from UI text fields. Strings for name, description, start time and end time.

**-Purpose:** to get information of the event from the GUI

**-Pre-Conditions:** the user must start a new project and enter information in the event configuration input fields.

**-Post-Conditions:** the attributes for the name, description, start date/time and end date/time are retrieved from the UI and ready to be set to the event config

**Contract 23**

**-Method Name:** getDirectoryInfo(rootdir r, rdir rd, bdir b, wdir w)

**-Return Type:** void

**-Input & Output parameters:**

- **r:** string input

- **rd:** string input

- **b:** string input

- **w:** string input

-**Description of parameters:** user inputs string of the file paths of the root directory, red team folder location, blue team folder location, white team folder location.

**-Purpose:** to get information of the directory paths from the GUI

**-Pre-Conditions:** the user must start a new project and enter information in the directory configuration input fields.

**-Post-Conditions:** the attributes for the root directory, red, blue and white directory are retrieved from the UI and ready to be set to the event config

### Log File Manager

*Table 2.1*

|  |  |
| --- | --- |
| **Class Name**: Log File Manager | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**  1. Knows log file information.  2. Updates log file cleansing status.  3. Updates log file ingestion status.  4. Updates log file validation status. | |
| **Contract:** 8. Get log file | |
| **Responsibilities** | **Collaborations** |
| 1. Get information of a log file. | Log File(1) |
| **Contract:** 13. Get log file cleansing status | |
| **Responsibilities** | **Collaborations** |
| 1. Update the cleansing status of a log file  2. Update the validation status of a log file | Log File Cleaner(1)  Log File Cleaner(2) |

**Contract 8**

**-Method Name:** getLogFile()

**-Return Type:** void

**-Input & Output parameters:**

-**Description of parameters:** the reference to a log file would allow the manager to retrieve information of the log such as its name, path, type of file, cleansing, validation and its ingestion status.

**-Purpose:** to allow the ingestion, cleansing, validation processes and action report to retrieve information of the log file.

**-Pre-Conditions:** there must be a valid directory that contains the logs and the use has specified their location.

**-Post-Conditions:** a log file object is returned.

**Contract 13**

**-Method Name:** updateCleansingStatus(name n, status s)

**-Return Type:** void

**-Input & Output parameters:**

- **n:** string

- **s:** boolean

-**Description of parameters:** the name of the logfile and the current cleansing status of the log file

**-Purpose:** to set the cleansing status of a log file to true if it has been cleansed successfully

**-Pre-Conditions:** a log file is found from the root directory provided by the user and has not yet been processed or has been processed already

**-Post-Conditions:** sets the cleansing status of a log to true if it has been cleansed successful otherwise it is set to false if the cleansing process failed

**-Method Name:** updateValidationStatus(name n, status s)

**-Return Type:** void

**-Input & Output parameters:**

- **n:** string

- **s:** boolean

-**Description of parameters:** the name of the logfile and the current validation status of the log file

**-Purpose:** to check whether a log file has been validated yet by the system

**-Pre-Conditions:** the log file must exist and have been cleansed by the system in order to proceed to validation

**-Post-Conditions:** sets the validation status of a log file to true if it has been successfully validated or false if the log file has failed validation

### Log Entry Manager

*Table 2.2*

|  |  |
| --- | --- |
| **Class Name**: Log Entry Manager | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**  1. Knows log entry information.  2. Add log entry to database. | |
| **Contract:** 12. Add log entry | |
| **Responsibilities** | **Collaborations** |
| 1. Add a log entry. | Log File Ingestor (1) |
| **Contract:** 20. Get log entry’s information | |
| **Responsibilities** | **Collaborations** |
| 1. Return information of a log entry. | Log Entry(1) |
| **Contract:** 6. Store log entries information in Database | |
| **Responsibilities** | **Collaborations** |
| 1. Store Vector in Database | Database(1) |

**Contract 12**

**-Method Name:** addLogEntry(name n, timeStamp t, content c, source s, soureType st)

**-Return Type:** void

**-Input & Output parameters:**

- **n:** string

- **t:** string

- **c:** string

- **s:** string

- **st:** string

-**Description of parameters:** name, timestamp, content of a log file, what log file the entry came from and the source type of the log file

**-Purpose:** to add a log entry that has completed the cleansing, validation and ingestion process to the system

**-Pre-Conditions:** a log file has to have been processed by Splunk then ingested

**-Post-Conditions:** a log entry is appended to a list of log entries that have finished being cleansed, validated and ingested

**Contract 20**

**-Method Name:** getEntry(logEntry e)

**-Return Type:** void

**-Input & Output parameters:**

- **e:** reference to a log entry object

-**Description of parameters:** the reference to a log entry to retrieve information on log entries that have been created

**-Purpose:** to retrieve the attributes of a log entry this includes its log entry number, path, creator, source, source type and content.

**-Pre-Conditions:** log files have been ingested, validated and cleansed and added a list of existing log entries.

**-Post-Conditions:** returns log entry reference

**Contract 6**

**-Method Name:** addEntryToDatabase(logEntry e)

**-Return Type:** void

**-Input & Output parameters:**

- **e:** reference to a log entry object

-**Description of parameters:** the reference to a log entry to retrieve information on log entries that have been created

**-Purpose:** to store the attributes of the log entry on the database

**-Pre-Conditions:** log files have been ingested, validated and cleansed

**-Post-Conditions:** the attributes of the log entry being stored are saved to the data base, this includes the log entry number, path, creator, source, source type and content.

### Vector Manager

*Table 2.3*

|  |  |
| --- | --- |
| **Class Name**: Vector Manager | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**  1. Knows vector information.  2. Updates a vector.  3. Deletes a vector.  4. Checks if a vector exists.  5. Creates vector.  6. Adds a vector to a database. | |
| **Contract:** 7. Update Vector information | |
| **Responsibilities** | **Collaborations** |
| 1. Update information of a vector | Vector(1) |
| **Contract:** 19. Get user input from UI | |
| **Responsibilities** | **Collaborations** |
| 1. Receive input from user to create vector | Event Configuration UI(1) |
| **Contract:** 5. Get Vector from database | |
| **Responsibilities** | **Collaborations** |
| 1. Get list of vectors in the database  2. Add vector to database. | Database(1)  Database(2) |
| **Contract:** 24. Associate log entry to node | |
| **Responsibilities** | **Collaborations** |
| 1. Add nodes to a list for every log entry associated to a vector | Node Manager(1) |

**Contract 5**

**-Method Name:** getVectors(vectors v)

**-Return Type:** list

**-Input & Output parameters:**

- **v:** reference to vector list

-**Description of parameters:** a list of vector objects that contain the vectors name and description

**-Purpose:** to retrieve a list of vectors associate to the current assessment and display them in the UI

**-Pre-Conditions:** The system is connected to the database and there is at least one vector stored.

**-Post-Conditions:** a list containing the vectors currently stored in the data base is returned.

**Contract 7**

**-Method Name:** updateVector(lead x, name n, description d)

**-Return Type:** void

**-Input & Output parameters:**

- **x:** reference to vector

- **n:** string

- **d:** string

-**Description of parameters:** reference to a vector that’s going to be edited, name and description of edits of vector.

**-Purpose:** to update the name and description of a vector that already exists.

**-Pre-Conditions:** a vector must exist and have a name and description.

**-Post-Conditions:** the name and description attributes of the vector are changed

**Contract 19**

**-Method Name:** getVectorInformation(name n, description d)

**-Return Type:** void

**-Input & Output parameters:**

- **n:** string input

- **d:** string input

-**Description of parameters:** name and description of the vector being created from user input.

**-Purpose:** to get information from the user from the GUI.

**-Pre-Conditions:** the user must start a new project and input data to create a new vector in the GUI, or the user may create new vectors anytime using the edit vector configuration

**-Post-Conditions:** the name and description attributes of the vector are retuned

**Contract 24**

**-Method Name:** associateLogEntry(logEntry e, vector v)

**-Return Type:** void

**-Input & Output parameters:**

- **n:** reference to a log entry

- **d:** reference to a vector

-**Description of parameters:** reference to a vector and a reference to the log entry that will be associated to the vector

**-Purpose:** to create and display a node for every log entry associated to the vector.

**-Pre-Conditions:** a vector must exist, and log entries must be cleansed, validated and ingested into the system

**-Post-Conditions:** log entries are added to a list describing its association to the vector, meaning the log entry is part of the vector

### Node Manager

*Table 2.4*

|  |  |
| --- | --- |
| **Class Name**: Node Manger | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**  1. Create a node.  2. Know node information. | |
| **Contract:** 18. Create a node | |
| **Responsibilities** | **Collaborations** |
| 1. create a new node from log entries or user input | Node(1) |

**Contract 18**

**-Method Name:** addNode(id i, name n, time t, description d, logCreator c, icon im, source s, visibility v)

**-Return Type:** node object

**-Input & Output parameters:**

- **i:** integer

- **n:** string

- **t:** string

- **d:** string

- **c:** string

- **im:** string

- **s:** string

- **s:** boolean

-**Description of parameters:** the nodes unique id, name of the node, timestamp of the node, description of the node, entity that created the node, path to icon the node will displayed as, log file the node references, and true if the node is visible on the graph false if the node it not visible.

**-Purpose**: to create a node object from a log entry or from user input.

**-Pre-Conditions:** log entry must be associated with a vector and must be marked significantor node has been created by the user from the GUI.

**-Post-Conditions:** a reference to a node object is returned containing all the attributes as described above.

## Processes

### Cleansing

A process that takes in log files and removed empty lines and unwanted characters. It receives text files from the log files found in the directories and text files returned from the OCR and transcription tool.

*Table 3.0*

|  |  |
| --- | --- |
| **Class Name**: Log File Cleaner | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**  1. Cleanse Log files. | |
| **Contract:** 15. Transcribe audio file to text file | |
| **Responsibilities** | **Collaborations** |
| 1. Calls the OCR to transcribe text from the image file to a text file | OCR(1) |
| **Contract:** 16. Transcribe image file to text file | |
| **Responsibilities** | **Collaborations** |
| 1. Calls the transcription tool to transcribe text file from audio file | Transcription Tool(1) |

**Contract 15**

**-Method Name:** transcribeAudioLogFile(logfile e)

**-Return Type:** text file

**-Input & Output parameters:**

- **e:** reference to an audio log file

-**Description of parameters:** image of log file.

**-Purpose:** to transcribe an audio file to a text file to be processed by Splunk.

**-Pre-Conditions:** audio file must exist on a blue, red or white team directory and fall in the time range of the event.

**-Post-Conditions:** the audio file is transcribed into a text file.

**Contract 16**

**-Method Name:** transcribeImageLogFile(logfile e)

**-Return Type:** text file

**-Input & Output parameters:**

- **e:** reference to an image log file

-**Description of parameters:** image of log file.

**-Purpose:** to transcribe an image file to a text file to be processed by Splunk.

**-Pre-Conditions:** image file must exist on a blue, red or white team directory and fall in the time range of the event.

**-Post-Conditions:** the image file is transcribed into a text file.

### Ingestion

Process used to turn log files into log entries and return the log entries to the system from Splunk.

*Table 3.1*

|  |  |
| --- | --- |
| **Class Name**: Log File Ingestor | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**  1. Validate time stamps from time range.  2. Retrieve Log files that have been processed by Splunk. | |
| **Contract:** 11. Get time range from event configuration manager. | |
| **Responsibilities** | **Collaborations** |
| 1. Get start and end time for event. | Event Configuration Manager(1) |
| **Contract:** 14. Feed new log entries to log entry manager. | |
| **Responsibilities** | **Collaborations** |
| 1. Get new log entries. | Splunk(1) |

**Contract 11**

**-Method Name:** getTimeRange(startTime s, endTime e)

**-Return Type:** void

**-Input & Output parameters:**

- **s:** string input

- **e:** string input

-**Description of parameters:** start date/time and end date/time for the current event.

**-Purpose:** used to filter log files that are in the time range of the current event.

**-Pre-Conditions:** log files must have been cleansed and then processed by Splunk.

**-Post-Conditions:** none

**Contract 14**

**-Method Name:** setNewLogEntry(logEntry e)

**-Return Type:** void

**-Input & Output parameters:**

- **e:** reference to a log entry object

-**Description of parameters:** new log entry object after a log file has been processed by Splunk.

**-Purpose:** to add new log entries that are associated to the event and are within time range.

**-Pre-Conditions:** log entries must have been processed by Splunk and ingestion process is given a log entry object.

**-Post-Conditions:** a log entry object has been added to the current assessment.

## Tools

### Transcription Tool

External tool used to transcribe text from an audio file. The tool returns a text file.

*Table 4.0*

|  |  |
| --- | --- |
| **Class Name**: Optical Character Reader (OCR) | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**  1. Transcribe from image files to text files. | |
| **Contract:** called by 16. (see section 3.3.1) | |
| **Responsibilities** | **Collaborations** |
|  |  |

### Optical Character Reader

External tool used to read text from an image and create a text file.

*Table 4.1*

|  |  |
| --- | --- |
| **Class Name**: Transcription Tool | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**  1. Transcribe from audio to text files. | |
| **Contract:** called by 15. (see section 3.3.1) | |
| **Responsibilities** | **Collaborations** |
|  |  |

### Splunk

External tool used to validate and ingest log files saved as text files.

*Table 4.2*

|  |  |
| --- | --- |
| **Class Name**: Splunk | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**  1. Process text log files. | |
| **Contract:** called by 14. (see section 3.3.2) | |
| **Responsibilities** | **Collaborations** |
|  |  |

## UI

### Team Configuration UI

This dialog is used to ask the user for the IP address of the lead analysist if the user isn’t a lead analyst. For indicating the lead analyst, a user would check the check box for lead analyst option, then enter their IP address. The list of connections connected to the lead can be seen in a table on this view.

*Table 5.0*

|  |  |
| --- | --- |
| **Class Name**: Team Configuration UI | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**  1. Display a checkbox for the lead analyst to choose.  2. Display a “lead IP” label next to check box.  3. Display a “lead IP address” label.  4. Display a text input field for the lead and other analysts to input the leads IP address.  5. Display a “connections” label.  6. Display a table view to show the connections to the lead.  7. Display a “save” button to allow the user to save inputs. | |
| **Contract:** called by 21. (see section 3.2.1) | |
| **Responsibilities** | **Collaborations** |
|  |  |

### Vector Configuration UI

Dialog used to ask the user for input of a vectors name and description. Users can add, delete and edit a vector from this dialog.

*Table 5.1*

|  |  |
| --- | --- |
| **Class Name**: Vector Configuration UI | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**  1. Display a table view of vectors that have been created.  2. Display a “delete vector” button.  3. Display an “edit vector” button.  4. Display and “add vector” button | |
| **Contract:** called by 19. (see section 3.2.4) | |
| **Responsibilities** | **Collaborations** |
|  |  |

### Directory Configuration UI

Dialog where a user indicates the path to the root directory of the team folders containing the log files. User input for the location of the red, blue and white team is also asked of the user to enter.

*Table 5.2*

|  |  |
| --- | --- |
| **Class Name**: Directory Configuration UI | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**  1. Display a “root directory” label.  2. Display a “red team folder” label.  3. Display a “blue team folder” label.  4. Display a “white team folder” label.  5. Display input fields for appropriate labels.  6. Display “browse” button next to each input field.  7. Display a “save” button. | |
| **Contract:** called by 23. (see section 3.2.1) | |
| **Responsibilities** | **Collaborations** |
|  |  |

### Event Configuration UI

This dialog is meant for the user to input data such as the current events name, description, start and end date and time.

*Table 5.3*

|  |  |
| --- | --- |
| **Class Name**: Event Configuration UI | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**  1. Display a “event name” label.  2. Display a “event description” label.  3. Display a “start date and time” label.  4. Display a “end date and time” label.  5. Display input fields for appropriate labels.  6. Display a “save” button. | |
| **Contract:** called by 21. (see section 3.2.1) | |
| **Responsibilities** | **Collaborations** |
|  |  |

### Action Report View

View to display log files and the process, they have failed to pass.

*Table 5.4*

|  |  |
| --- | --- |
| **Class Name**: Action Report View | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**  1. Display log files that have failed validation, cleaning or the ingestion process. | |
| **Contract:** 1. Display log files that failed processing | |
| **Responsibilities** | **Collaborations** |
| 1. Get log files that have failed cleansing, validation, or ingestion | Log File Manager(1) |

**Contract 1**

**-Method Name:** getLogFile()

**-Return Type:** log file object

**-Input & Output parameters:**

-**Description of parameters:**

**-Purpose:** to return and display log files that have failed cleansing, validation and ingestion process.

**-Pre-Conditions:** log file has been cleansed but failed or validated but failed or ingested but failed.

**-Post-Conditions:** reference to a log file that has failed one process is displayed in the action report view.

### Processing View

View that is used to display a log file and cleansing, validation ingestion status. It displays in progress while the file is being or has yet to be processed by appropriate process. The view shall display “successful” if the log file has been cleansed, validated or ingested without fail; and display “failed” if the opposite.

*Table 5.5*

|  |  |
| --- | --- |
| **Class Name**: Processing View | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**  1. Display a table for the log files that are being cleansed, validated, and ingested. | |
| **Contract:** 2. Get Log file process status | |
| **Responsibilities** | **Collaborations** |
| 1. Retrieve the status of cleansing, validation and ingestion of log files | Log File Manager(1) |

**Contract 2**

**-Method Name:** getLogFileCleansingStatus()

**-Return Type:** Boolean

**-Input & Output parameters:**

-**Description of parameters:**

**-Purpose:** to display the current cleansing status of a log file on the processing view.

**-Pre-Conditions:** a log file must exist in a red, blue or white file directory.

**-Post-Conditions:** the cleansing status of the log files is returned and if it hasn’t been processed it displays “in progress” if it has successfully been processed display “success” and otherwise display “failed”

**-Method Name:** getLogFileValidationStatus()

**-Return Type:** Boolean

**-Input & Output parameters:**

-**Description of parameters:**

**-Purpose:** to display the current validation status of a log file on the processing view.

**-Pre-Conditions:** a log file must exist in a red, blue or white file directory.

**-Post-Conditions:** the validation status of the log files is returned and if it hasn’t been processed it displays “in progress” if it has successfully been processed display “success” and otherwise display “failed”

**-Method Name:** getLogFileIngestionStatus()

**-Return Type:** Boolean

**-Input & Output parameters:**

-**Description of parameters:**

**-Purpose:** to display the current ingestion status of a log file on the processing view.

**-Pre-Conditions:** a log file must exist in a red, blue or white file directory.

**-Post-Conditions:** the ingestion status of the log files is returned and if it hasn’t been processed it displays “in progress” if it has successfully been processed display “success” and otherwise display “failed”

### Analysis View

View used to display a vector and its associated log entries in a table as well as a graph view of it log entries.

*Table 5.6*

|  |  |
| --- | --- |
| **Class Name**: Analysis View | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**  1. Display log entry list on a table.  2. Display current vector and its log entries. | |
| **Contract:** 3. Get log files | |
| **Responsibilities** | **Collaborations** |
| 1. Get the log files associated to the current vector. | Log Entry Manager(1) |
| **Contract:** 4. Display log files | |
| **Responsibilities** | **Collaborations** |
| 1. Get the list of vectors to display.  2. Display the current vector. | Vector(1,2) |

**Contract 3**

**-Method Name:** getLogEntry()

**-Return Type:**

**-Input & Output parameters:**

-**Description of parameters:**

**-Purpose:** to get the log entries associated to the current vector displayed on the analysis view.

**-Pre-Conditions:** a vector exists and has associated log entries to it.

**-Post-Conditions:** log entries are displayed on a table to the associated vector on the window.

**Contract 4**

**-Method Name:** getVectors()

**-Return Type:** vector list

**-Input & Output parameters:**

-**Description of parameters:**

**-Purpose:** to display the vectors that have been created on the vector table.

**-Pre-Conditions:** a vector must exist in the database or has been created by the user with the GUI.

**-Post-Conditions:** a list of vectors for the current assessment are returned.

**-Method Name:** getCurrentVector()

**-Return Type:** vector object

**-Input & Output parameters:**

-**Description of parameters:**

**-Purpose:** to display the vector selected by the user from the list of vectors on the GUI.

**-Pre-Conditions:** a vector must exist in the database or has been created by the user with the GUI and has been chosen from a drop down on the GUI analysis view.

**-Post-Conditions:** the vector selected is displayed on the GUI.

### Graph

Displays nodes associated to a vector on a graph view. It shows the correlation of nodes between each other using connections. A user can create nodes not associated to log entries from this view as well.

*Table 5.7*

|  |  |
| --- | --- |
| **Class Name**: Graph | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**  1. Generate Graph.  2. Display Graph.  3. Get Information about nodes. | |
| **Contract:** 10. Generate node on graph | |
| **Responsibilities** | **Collaborations** |
| 1. Add node to the graph.  2. Add a connection from one node to another on the graph.  3. Display node on graph view. | Node Manager(1,2,3) |

**Contract 10**

**-Method Name:** getNodes(Vector v, node n)

**-Return Type:** void

**-Input & Output parameters:**

**- v: a** reference to the vector selected on the graph view

**- n:** reference to a node associated to the current vector selected on the graph view

-**Description of parameters:** the vector reference is selected when the user changes vectors on the analysis view, a node reference associated with the current vector.

**-Purpose:** to display a node on the graph view.

**-Pre-Conditions:** a node must exist from a log entry, visibility is set to true, or a node is created by the user from the GUI.

**-Post-Conditions:** a node is displayed on the graph view.

**-Method Name:** addConnection(vector v, node n1, node n2)

**-Return Type:** void

**-Input & Output parameters:**

**-v: reference to the current vector selected on the graph view**

**-n1: is a reference to the starting node to be connected.**

**-n2: reference to the second node that connects to the first node.**

-**Description of parameters:**

**-Purpose:** to draw an edge or “connection” from one node to another and visually display it on the graph view.

**-Pre-Conditions:** user must be on the graph view, at least two node objects exist on the current vector selected.

**-Post-Conditions:** a connection is displayed on the graph view connecting one node to another node.

## Database

### Database

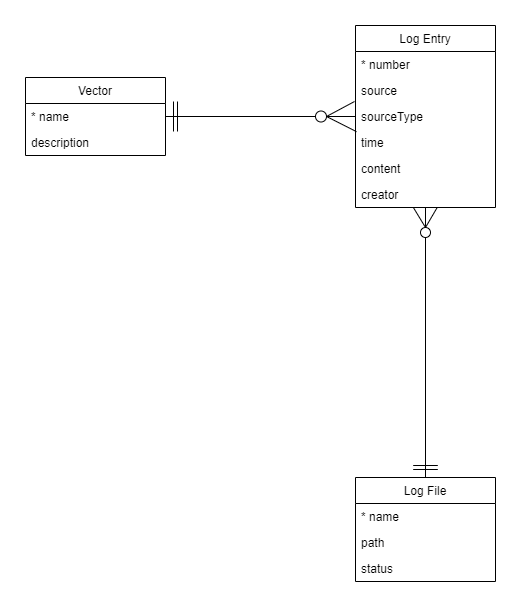
Mongo database used to store vector configuration information and their log entries as well as log files for the current assesment.

*Table 6.0*

|  |  |
| --- | --- |
| **Class Name**: Database | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**  1. Store vector configuration data.  2. Store log entry data. | |
| **Contract:** called by5 and 6 (see section 3.2.3, 3.2.4) | |
| **Responsibilities** | **Collaborations** |
|  |  |

# Database

## Database Schema



# Appendix

## Notes

-SDD will not match code.

-There’s duplication of data. This is true with the contracts between the managers and the models where the managers get attribute from the models with functions that get some attributes at once.

$$$