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

Software Design Document

Version 0.1

03/08/2020

Document Control

Approval

The Guidance Team and the customer shall approve this document.

Document Change Control

|  |  |
| --- | --- |
| Initial Release: | 0.1 |
| Current Release: |  |
| Indicator of Last Page in Document: | & |
| Date of Last Review: |  |
| Date of Next Review: |  |
| 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. 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:

Hector Dozal

Victor Vargas

Eduardo Lara

Irvin Bosquez

Gerardo Armenta

Change Summary

The following table details changes made between versions of this document

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Modifier | Description |
| 0.1 | 03/08/2020 | Hector Dozal | Creation of Document |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[Document Control ii](#_Toc226283109)

[Approval ii](#_Toc226283110)

[Document Change Control ii](#_Toc226283111)

[Distribution List ii](#_Toc226283112)

[Change Summary ii](#_Toc226283113)

[1. Introduction 1](#_Toc226283114)

[1.1. Purpose and Intended Audience 1](#_Toc226283115)

[1.2. Scope of Product 1](#_Toc226283116)

[1.3. References 1](#_Toc226283120)

[1.4. Definitions, Acronyms, and Abbreviations 2](#_Toc226283121)

[1.4.1. Definitions 2](#_Toc226283122)

[1.4.2. Acronyms 2](#_Toc226283123)

[1.4.3. Abbreviations 2](#_Toc226283124)

[1.5. Overview 2](#_Toc226283125)

[1.5.1. Introduction 2](#_Toc226283122)

[1.5.2. Decomposition Description 2](#_Toc226283123)

[1.5.3. Detailed Description of Components 2](#_Toc226283124)

[1.5.4. Database 2](#_Toc226283124)

[2. Decomposition Description 3](#_Toc226283126)

[2.1. Scope 3](#_Toc226283127)

[2.2. Use 3](#_Toc226283128)

[2.3. Subsystem Description 3](#_Toc226283129)

[2.4. Hierarchy Graphs 4](#_Toc226283130)

[2.4.1. Database Manager Subsystem 4](#_Toc226283131)

[2.4.2. Database Manager (API) 4](#_Toc226283132)

[2.4.3. Unit Conversion 5](#_Toc226283133)

[2.4.4. Notification Manager 5](#_Toc226283134)

[3. Dependency Description 6](#_Toc226283135)

[3.1. Scope 6](#_Toc226283136)

[3.2. Use 6](#_Toc226283137)

[3.3. Collaboration Description 6](#_Toc226283138)

[4. Database 8](#_Toc226283139)

[4.1. Databasae Design 8](#_Toc226283140)

[4.2. Database Schema 11](#_Toc226283152)

# Introduction

Section 1 will introduce the concept of a software design document, it will provide a scope for the PICK project, give a comprehensive list of definitions, acronyms and abbreviations, and an overview of the entire document.

## Purpose and Intended Audience

Software design is the process in which requirements are translated into a concrete representation of the software incorporating classes, interfaces, and data necessary for the proper implementation. This document will serve as a central source and sink for documentation which will be used to aid both LSH when using the final completed product and us; Team 13 for documenting and implementing new features. As such the intended audience of this document encompasses the clients (Dr. Oscar Perez, Mr. Vincent Fonseca, Ms. Herandy Vazquez, Mr. Baltazar Santaella, Ms. Florencia Larsen,) Team 13 (Eduardo Lara, Irvin Bosques, Gerardo Armenta, Hector Dozal, and Victor Vargas,) and the UTEP guidance team who will be providing input and constructive criticism on the overall structure of this document.

## Scope of Product

We will be creating a PMR Insight Collective Knowledge (PICK) tool to facilitate the process of writing reports by the White Team (LSH) about the ability of the blue team to defend against cyber-attacks by the red team. This software’s primary goal will consist of facilitating the job of the white team by employing different tools including: Sorting by chronological order based on the date of ingestion, automatic creation of a graphs to better represent when an attack has occurred, creation of vectors to organize all log files that pertain to a singular event, as well as other requirements described within this document. Our system shall focus firstly on improving the analysis of log files, our system shall not use the internet in any way to ensure the security of the system. The success of this project will be determined by the benefits given to the White team upon using this system.

## References

[1] E. Tai-Ramirez & S. Roach, SRS\_v7. Internet: https://github.com/CS4311-spring-2020/pick-tool-team06-team-404/blob/master/doc/SRSv7.pdf, 2020 (Jan. 30, 2020).

## Definitions, Acronyms, and Abbreviations

### Definitions

|  |  |
| --- | --- |
| **Log Entry** | Splunk takes the validated log files and convert them into normalized data. The normalized data are called log entries. Users of the system can filter and edit log entries. |
| **Data Validation** | Data validation is the process of inspecting data in the cleansed log files based on predefined data validation rules. |
| **Data Cleansing** | Data cleansing is the removal of unwanted characters from uncleansed TMUX log file; removal of blank rows from uncleansed excel log file; and removal of blank lines from uncleansed log file. |

### Acronyms

|  |  |
| --- | --- |
| **SDD** | Software Design Document |
| **LSH** | Lethality, Survivability, and HSI Directorate |
| **PICK** | Prevent, Mitigate, and Recover (PMR) Insight Collective Knowledge System |
| **PMR** | Prevent, Mitigate, and Recover |
| **UTEP** | University of Texas at El Paso |
| **ERD** | Entity/Relationship diagram |

### Abbreviations

|  |  |
| --- | --- |
| **e.g.** | For example |
| **i.e.** | That is |
| **TBD** | To be determined |
|  |  |
|  |  |

## Overview

The Software Design Document is divided into 4 major sections each with various subsections. The 4 main sections of the SDD are the following:

### Introduction

This will introduce the entire SDD as well as all the different subsections within it. The main subsections in section 1 are the following: Purpose of the document, Scope of the product, References and Definitions, Acronyms, and Abbreviations.

### Decomposition Description

This section will describe different components that will be involved in the design of the system. A UML component diagram will also be provided in this section.

### Detailed Description of Components

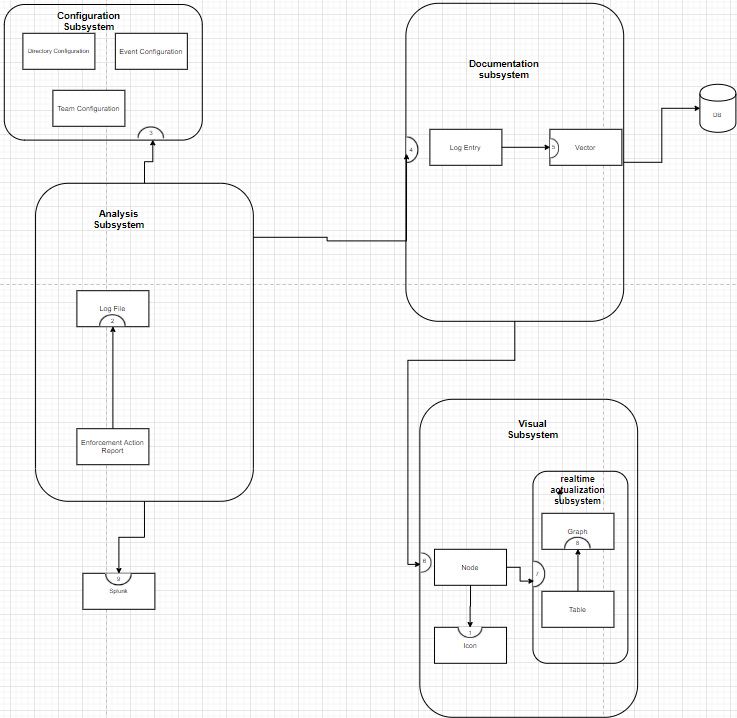
For each component or subsystem listed in section 3 a detailed description will also be provided. The description will also contain all the different contracts for those specific components.

### Database

In this section we will explore the database system that will be used to store information pertaining to the system. An entity-relationship schema will also be provided.

# Decomposition Description

<< Provide a description of how your component descriptions can be used by designers and maintainers identify major design entities, for purposes such as determining which entity is responsible for specific functions and tracing requirements to design entities. >>



## System Collaboration Diagram

<< Provide a UML Component Diagram or a Wirffs-Brock Collaboration Diagram. If this is a subsystem or part of a larger system, show the collaboration or component diagram for the entire system in a separate diagram first.

Show the major components or subsystems in this system and indicate collaborations between components. If useful, show the UML class diagram that indicates class hierarchies.>>

<< Provide a description of the way the system has been structured and the major divisions between the design entities. Subsystems and classes are referred as design entities >>

## Subsystem and Component Descriptions

<< Provide a description of subsystems and components in the diagrams in section 2.1. For each component or subsystem to be described in this document, give the name, describe the purpose, and list the contracts supported by this component (e.g., the subsystem contracts). Indicate where in the following detailed sections of this document each component is discussed. >>

This section will provide a description of each subsystem and component of the system, specified in the previous section. It will talk about main things of each subsystem indicate where the classes are described in following sections.

**Subsystem Name:** Configuration Subsystem

**Subsystem Purpose:** The purpose of this system is to contain all the configuration of the system for that current project.

**Subsystem Contract:** The details will be described in section 5.

**Subsystem Name:** Analysis Subsystem

**Subsystem Purpose:** The purpose of this system is to ingest, cleanse and validate log files and send them to Splunk class for uploading.

**Subsystem Contract:** Contracts 2, 3, 4 and 9. Will be descripted in detail in section 4.

**Subsystem Name:** Documentation Subsystem

**Subsystem Purpose:** The purpose of this subsystem is to provide the multiple configuration information for other subsystems including, directory, event and team configuration.

**Subsystem Contract:** Contract 5,6. Will be described in detail in section 6.

**Subsystem Name:** Realtime Actualization subsystem

**Subsystem Purpose:** The purpose of this system is to correlate the data between the table and graph

**Subsystem Contract:** Contract 8. Details will be given in Section.

**Subsystem Name:** Visual subsystem

**Subsystem Purpose:** The purpose of this system is to display the log entries in a visual representation with Nodes and Icons.

**Subsystem** Contract 1,7,8. Will be described in detail in section 3.

## Dependencies

<< describe how the component dependencies will impact development >>

This section will talk about the dependencies that the program has, by subsystem and how this can impact development.

The Splunk subsystem will depend on the Splunk Enterprise Software and Splunk Python SDK, this can influence the development of the system if the main Splunk software or the SDK are updated and deprecate any of their methods or make significant changes to the API. Any big change can lead to a delay on the development of the PICK system.

The Graph Subsystem, specifically the graph class will be dependent on the library that is chosen to provide that view. Given that Maltego does not have an SDK that delays the graphing aspect of the software until an appropriate external tool can be found. Same as with Splunk, if the chosen tool undergoes a great change, the development time will increase.

The Log subsystem and the Configuration subsystem will interact with a local database. The tool that is used for this will increase the development time, depending on its complexity and the changes the tool undergoes.

Python: 3.7

Splunk Enterprise and SDK Support: 8.0

# Detailed Description of Visual Subsystem

Each component or subsystem listed in section 2.2 are described with a detailed design description for clarification. They all include their respective classes with their responsibilities, collaborations, contracts, and protocols.

## Component Description

|  |  |
| --- | --- |
| **Component Name:**  Visual | **Purpose:**  It is the component that encompasses the graphical view of the data and the tabular view of the data. The data presented come from the log entries chosen by the user to be part of the vector. |
| **Classes:**   1. Table 2. Graph 3. Node 4. Icon 5. Vector | |

## Class Description Table

|  |  |
| --- | --- |
| **Class Name**: Table | |
| **Description:** Class that will show a visual representation of significant log entries in tabular format | |
| **Superclass**: Graph | |
| **Subclasses**: None | |
| **Private Responsibilities:**   * Knows:  1. Node’s relationships. 2. Node’s information.  * Can:  1. Change the visibility status of a node. 2. Order nodes in ascending and descending order 3. Update row information. | |
| **Contract:** 1. Gets node information from the Node class. | |
| **Responsibilities** | **Collaborations** |
| 1. Requests and displays information from a Node | Node |
| **Contract**: 2. Gets node information changes from the Graph class. | |
| **Responsibilities** | **Collaborations** |
| 1. Knows about the changes made in the Graph class. 2. Can update the information shown in the table based on the changes from the Graph class. | Graph |

### Contract 1

This contract is responsible for obtaining the necessary information to display in the table view.

Protocol: Get Node Information

Method Name: getNodeInfo()

Type of Output: List of Strings

Type of Input: List of Strings

Pre-Conditions: There must be a vector created. There must be log entries in the system.

Post-Conditions: Shows up to date node information.

Protocol: Edit Row

Method Name: editRow()

Type of Output: Void

Type of Input: List of String, List of String, Int. Row to be edited in integer, and two lists of strings, one representing the fields to change and one to represent the new data.

Pre-Conditions:

Post-Conditions: Row has been edited.

Protocol: Set Column Visibility

Method Name: ColumnShow()

Type of Output: None

Type of Input: None

Pre-Conditions:

Post-Conditions: The table show the colu

## Class Description Graph

|  |  |
| --- | --- |
| **Class Name**: Graph | |
| **Description:** Class that will show a visual representation of significant log entries in graphical format | |
| **Superclass**: None | |
| **Subclasses**: Table | |
| **Private Responsibilities**   * Knows:   1. Export Format   2. Orientation   3. Interval Units   4. Interval   5. Position of Nodes   6. Position of Relationships   7. The size of the current graph/table * Can:   1. Can add new nodes.      + Adds a new row to the node table.      + Adds a new node to the nodes configuration in the graph.   2. Can add relationships to nodes.      + Adds a new row in the relationship table.      + Adds a new relationship to the nodes configuration in the graph.   3. Can delete relationships.      + The row in the relationship table shall be removed.      + The relationship in the nodes configuration in the graph shall be removed.   4. Can display an error when the user attempts to delete a relationship when one is not selected.   5. Can delete a node.      + The selected node in the table shall be removed.      + The selected node in the nodes configuration in the graph shall be removed.   6. A display error is shown when the user attempts to delete a node when non is selected.   7. When filtering is complete,      + The node table shall display nodes that meet the criteria.      + The relationship table shall display relationships that meet the criteria.      + The graph shall display nodes that meet the criteria.   8. When the user selects to edit node the user will be able edit node contents   9. Export graph in selected format   10. Undo any changes to graph since last commit   11. Commit any changes made to graph and save it       + Change should be saved in temporary storage       + Change should be logged in change list   12. Change icon type of node   13. Do “Add Icon” operation   14. Do “Delete Icon” operation   15. Allow nodes to be repositioned in graph   16. Display nodes in graphical view   17. Display nodes in table view | |
| **Contract:** Get information from Node class | |
| **Responsibilities** | **Collaborations** |
| Requests Information of a Node instance. | Node |
| **Contract**: | |
| **Responsibilities** | **Collaborations** |
|  |  |

### Contract

Purpose of this contract is to obtain information from the Node class to display it in a graphical way.

Protocol: Get Node Information

Method Name: getNodeInfo()

Type of Output: List of Strings

Type of Input: List of Strings

Pre-Conditions: There must be a vector created. There must be log entries in the system.

Post-Conditions: Node data is received.

Protocol: Add Node

Method Name: addNode

Type of Output: Void, it applies changes to graph.

Type of Input: List of Strings, information to create the Node to be Displayed.

Pre-Conditions: The added node must be linked with a log entry.

Post-Conditions: A node is added to the graph.

Protocol: Change Icon

Method Name: iconChange

Type of Output: void, icon change is applied to a node.

Type of Input: Node, the node to which the change is going to be applied.

Pre-Conditions: Node change action is triggered.

Post-Conditions: Icon of a Node is changed.

Protocol:

Method Name:

Type of Output:

Type of Input:

Pre-Conditions:

Post-Conditions:

## Class Description Node

|  |  |
| --- | --- |
| **Class Name**: Node | |
| **Description:** Visual/Tabular representation of a log entry | |
| **Superclass**: Log Entry | |
| **Subclasses**: None | |
| **Private Responsibilities**   * Knows:   1. Node Visibility   2. At least one graph   3. Node Id Visibility   4. Node Name Visibility   5. Node Timestamp Visibility   6. Node Description Visibility   7. Log Entry Reference Visibility   8. Log Creator Visibility   9. Event Type Visibility   10. Icon Type Visibility   11. Source Visibility   12. Node Visibility   13. Relationship ID   14. Parent ID   15. Child ID   16. Label * Can:   1. Make a relationship between a parent and a child node.   2. Allow user to select visibility settings of a node      + Node Id Visibility      + Node Name Visibility      + Node Timestamp Visibility      + Node Description Visibility      + Log Entry Reference Visibility      + Log Creator Visibility      + Event Type Visibility      + Icon Type Visibility      + Source Visibility      + Node Visibility | |
| **Contract:** Icon Depicting Team Node | |
| **Responsibilities** | **Collaborations** |
| Can have an icon depicting the team the node belongs to | Icon |
| **Contract**: Log entry data | |
| **Responsibilities** | **Collaborations** |
| Gets all the information needed from the log entries to store in the node | Log |

### Contract

Purpose of this contract is to request the icon to display a certain information and image.

Protocol: Get Team

Method Name: getTeam()

Type of Output: Returns color of the team (e.g. Red, Blue, White)

Type of Input: None

Pre-Conditions: Request team color

Post-Conditions: Team color associated to node

Protocol: Display Team Icon

Method Name: displayTeamIcon()

Type of Output: None, displays icon referring to the team

Type of Input: None

Pre-Conditions: Node has a team associated to it

Post-Conditions: Node icon is displayed

### Contract

Purpose of this contract is to get the information from a log entry to store in a node

Protocol: Get log entry data

Method Name: getData()

Type of Output: Returns the information needed to be stored in a node

Type of Input: None

Pre-Conditions: There must be log entries in the system

Post-Conditions: Aides in the creation of a Node for the Graph

Protocol: Successfully creates a Node instance

Method Name: node()

Type of Output: Node

Type of Input: Gets log info from getData()

Pre-Conditions: Log information must be received from getData()

Post-Conditions: A node is generated

## Class Description Icon

|  |  |
| --- | --- |
| **Class Name**: Icon | |
| **Description:** Visual image of a node | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**   * Knows:   1. Icon Name   2. File Path * Can:   1. Store the attributes it knows | |
| **Contract:** Display Team Icon | |
| **Responsibilities** | **Collaborations** |
| Can display an icon on a node depending on which team the node corresponds to (e.g. Red, Blue, or White Team) | Node (8) |
| **Contract**: | |
| **Responsibilities** | **Collaborations** |
|  |  |

### Contract 1: Display Team Icon

Contract responsible for displaying an icon depending on the information provided by node.

Protocol: Get Image

Method Name: getImage

Type of Output: Image Resource, to represent the icon.

Type of Input: None

Pre-Conditions: Request a change of image.

Post-Conditions: Image displayed as icon.

## Class Description Vector

|  |  |
| --- | --- |
| **Class Name**: Vector | |
| **Description:** Objective that is defined by the user | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**   * Knows:   1. Vector Name   2. Vector Description * Can:   1. Save the changes to permanent storage   2. Enable the user to edit the selected log entry   3. Add significant log entries to a vector   4. Establish connection between the lead and the user   5. Allow user to do “Pull” operation   6. Allow user to do “Push” operation | |
| **Contract:** Requests Information from Database | |
| **Responsibilities** | **Collaborations** |
| Requests information from Database. | DB? |
| **Contract**: | |
| **Responsibilities** | **Collaborations** |
|  |  |

Table 4

### Contract

This contract is to requests or update information of a vector based on the database status.

Protocol: Set Vector Name

Method Name: setVectorName

Type of Output: Void, sets a property of the class

Type of Input: String, represents vector name

Pre-Conditions: name of vector not set.

Post-Conditions: Name of vector is set.

Protocol: Save to Database

Method Name: saveDB

Type of Output: Void, performs an action onto a database

Type of Input: Void, performs with fields in class.

Pre-Conditions: Save Vector action triggered.

Post-Conditions: Vector and information is saved to Database.

Protocol: Add Log Entry

Method Name: addLogEntry

Type of Output: Void, performs action on class fields

Type of Input: Log Entry, received an instance of the log entry to be added to the vector.

Pre-Conditions: Log Entry is not in Vector already

Post-Conditions: Log entry added to vector.

Protocol: Pull from Database

Method Name: pull

Type of Output: Void, modifications done on class

Type of Input: Void

Pre-Conditions: Updates are possible

Post-Conditions: Vector updated from database.

Protocol: Push to database

Method Name: push

Type of Output: Void

Type of Input: Void, information obtained from vector itself

Pre-Conditions: Changes made to Vector

Post-Conditions: Database updated.

# Detailed Description of Analysis Subsystem

|  |  |
| --- | --- |
| **Component Name:**  Data Processing | **Purpose:**  Handles the ingestion, cleansing, and validation of the files uploaded to PICK. Here the log files are generated prior to being sent to Splunk. |
| **Classes:**   1. Ingestion 2. Cleansing 3. Validation 4. Enforcement Action Report | |

## Class Description Log File

|  |  |
| --- | --- |
| **Class Name**: Log File | |
| **Description:** File from where entries are going to be extracted | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**   * Knows:   + Log File Name   + Cleansing Status   + Validation Status   + Ingestion Status   + Acknowledgement Status * Can:   + Data transformation operation of a log file   + Data cleansing operation of log file   + Data validation operation of log file   + Perform the data ingestion operation once data ingestion operation is complete   + Update ingested log file   + Do search operation by keyword   + Do different search mechanisms     - Logical searching     - Wildcard searching | |
| **Contract:** 1. Print Information | |
| **Responsibilities** | **Collaborations** |
| 1. Do something | B (3) |
| **Contract**: 2. Provide Some Attributes | |
| **Responsibilities** | **Collaborations** |
| 1. Know Something 2. Know another thing | X (7) |

Table 2

### Contract <contract 1 name>

## Class Description Enforcement Action Report

|  |  |
| --- | --- |
| **Class Name**: Enforcement Action Report | |
| **Description:** Provides insight to errors during cleansing and validation | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**   * Knows:   + Line Number   + Error Message * Can:   + Generate an enforcement action report and set the validation status of the log file to “fail” of data validation fails | |
| **Contract:** 1. Print Information | |
| **Responsibilities** | **Collaborations** |
| 1. Do something | B (3) |
| **Contract**: 2. Provide Some Attributes | |
| **Responsibilities** | **Collaborations** |
| 1. Know Something 2. Know another thing | X (7) |

Table 3

# Detailed Description of Configuration Subsystem

|  |  |
| --- | --- |
| **Component Name:**  Project Configuration | **Purpose:**  Anything that has to do with the system’s setting regarding the event configuration for log ingestion, path directories for the location of the original files to be uploaded to the system are managed here. |
| **Classes:**   1. Event Configuration 2. File Directory | |

## Class Description Event

<< For each class in this component, create a new Heading 2 subsection. In each subsection, describe the internal details of each design entity. Each class should have a class name, a description, super classes, and private responsibilities. >>

|  |  |
| --- | --- |
| **Class Name**: Event | |
| **Description:** Class that will contain all the event configuration for the current project | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**   * Knows:   + Event Name   + Event Description   + Event Start Timestamp   + Event End Timestamp   + Root Directory   + Red Team Folder   + White Team Folder   + Blue Team Folder   + Lead   + Lead’s IP Address   + Connection Established * Can:   + Check connection to lead conditions are true:     - Lead’s checkbox is unchecked     - Lead’s IP address is not empty     - The IP address of the local machine is not the same as lead’s IP address   + Display error for any undesired action for connection to lead conditions   + Structural check operation:     - The root directory shall contain three folders     - The names of the three folders shall match the red team folder name, blue team folder name, and white team folder name specified in the event configuration     - The name of the root directory shall not be editable once the event is saved   + Generate a root directory structure error | |
| **Contract:** 1. Print Information | |
| **Responsibilities** | **Collaborations** |
| 1. Do something | B (3) |
| **Contract**: 2. Provide Some Attributes | |
| **Responsibilities** | **Collaborations** |
| 1. Know Something 2. Know another thing | X (7) |

### Contract <contract 1 name>

<< For each contract supported by this class, create a new Heading 3 subsection. Give the detailed contract descriptions here. A contract description must have a contract identifier (used in the diagrams and in cross references), a contract name, a description, and a set of protocols. A protocol includes a list of responsibilities, method signatures to support those responsibilities, pre- and post-conditions, algorithm descriptions and collaborations. A collaboration must specify the contract supported by the service provider. >>

### Contract <contract 2 name>

### Contract <contract name>

# Detailed Description of Documentation Subsystem

## Class Description Log Entry

|  |  |
| --- | --- |
| **Class Name**: Log Entry | |
| **Description:** Represents a single piece of data in an event | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**   * Knows:   + Log Entry Number   + Log Entry Timestamp   + Log Entry Content   + Host   + Source   + Source Type   + Significant * Can:   + Mark a log entry as significant or not | |
| **Contract:** 1. Print Information | |
| **Responsibilities** | **Collaborations** |
| 1. Do something | B (3) |
| **Contract**: 2. Provide Some Attributes | |
| **Responsibilities** | **Collaborations** |
| 1. Know Something 2. Know another thing | X (7) |

Table 5

### Contract

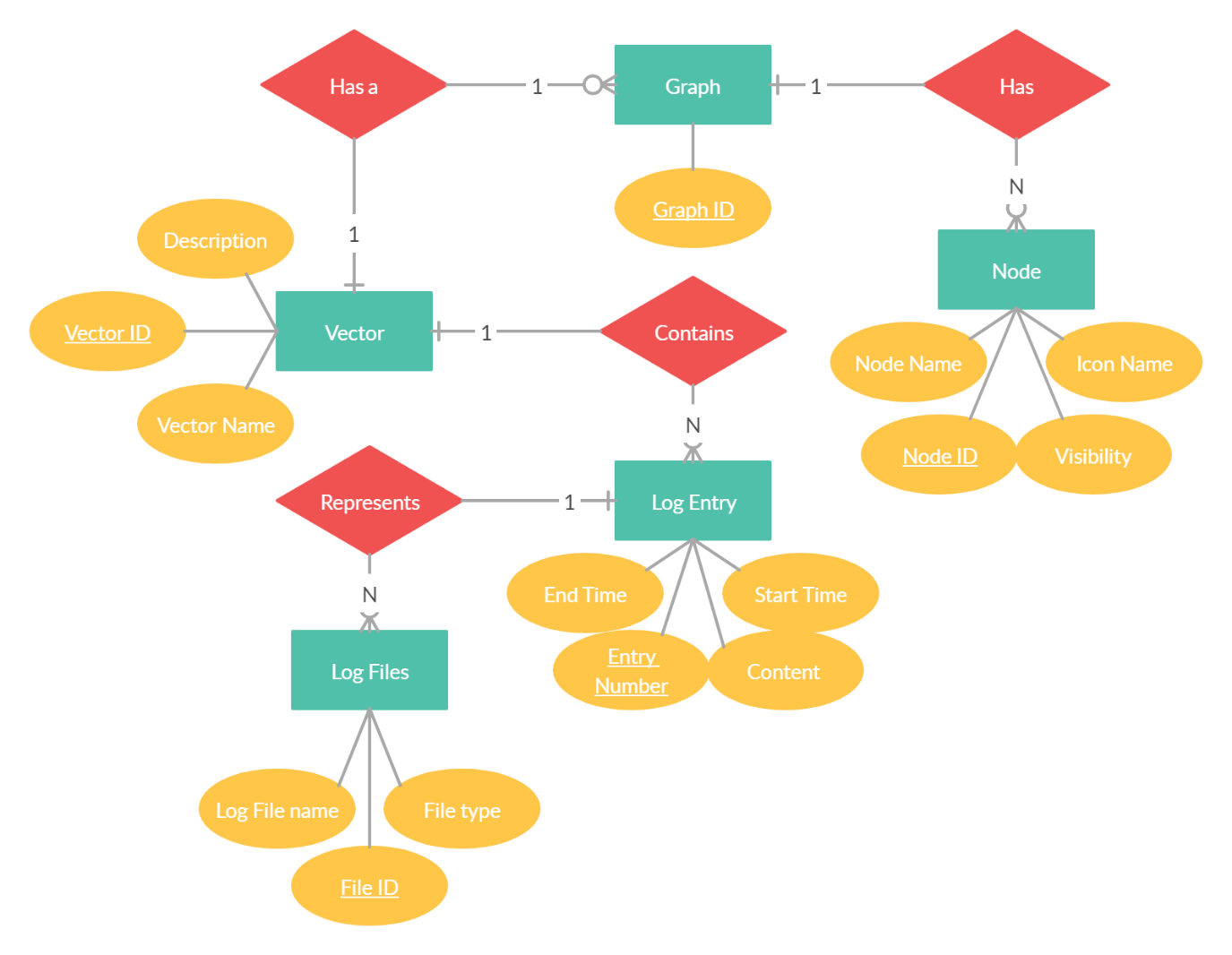
# Detailed Description of Splunk Class

|  |  |
| --- | --- |
| **Component Name:**  Splunk | **Purpose:**  This component is the connection between the PICK application and Splunk. This will allow for log files to be uploaded to Splunk for filtering and generating the log entries. |
| **Classes:**   1. Import and Export 2. Filtering | |

Table 1

# Database

## Database Schema



&