|  |  |
| --- | --- |
| **Concrete Class: Vector Manager** | |
| **Responsibilities (Does):**   * Compare Vector DB differences between Lead and Non-Lead * Download Vector and Graph from Lead Vector DB to user’s Vector DB * Upload Vector and Graph from user’s Vector DB to Lead approval status page * Load Vector from Local Storage * Download copy (Saving) of Vector to Local Storage * Load history of changes made in the Vector DB | **Collaborators:**   * The Vector Manager class will collaborate with Event Configuration to gain access to its attributes * The Vector Manager class receives log entries attributes from Significant Log Entry class * The Vector Manager class receives Graph attributes from Graph class * The Vector Manager class receives Vector attributes from Vector class |

|  |  |
| --- | --- |
| **Concrete Class: Event Configuration** | |
| **Responsibilities (Knows):**   * Knows event name. * Knows event description. * Knows event start timestamp. * Knows event end timestamp. * Knows root directory. * Knows red team folder. * Knows white team folder. * Knows blue team folder. * Knows where the master vector DB is stored. * Knows lead’s IP address. * Knows the number of established connections to the host machine.   **Responsibilities (Does):**   * Check directory structure. | **Collaborators:**   * The Event Configuration is a client of the user manager class who tells it how many users are connected. |

|  |  |
| --- | --- |
| **Log File** | |
| **Responsibilities (Knows):**   * Knows the log file name. * Knows the cleansing status of a log file. * Knows the validation status of a log file. * Knows the ingestion status of a log file. * Knows the acknowledgement status of a log file. * Knows audio transcribed text in one- minute intervals. * Knows transcribed text from audio extracted from video in one-minute intervals. * Knows extracted text from images. * Knows extracted text from pdf files. * Knows start date, end date, start timestamp, and end timestamp. | **Collaborators**:   * A Log File class depends upon the directory attributes provided by the Event Configuration class |

|  |  |
| --- | --- |
| **Enforcement Action Report** | |
| **Responsibilities (Knows):**   * Knows where an error occurs in a log file. * Knows why a specific line in the log file failed the validation test. | **Collaborators**:   * The Enforcement Action Report receives the status attributes and transcriptions from the Log File class |

|  |  |
| --- | --- |
| **Vector** | |
| **Responsibilities (Knows):**   * Knows its name * Knows its description | **Collaborators**: |

|  |  |
| --- | --- |
| **Significant Log Entry** | |
| **Responsibilities (Knows):**   * Knows the log entry number. * Knows the timestamp. * Knows the log entry content. * Knows the name of the log file from which it originates. * Knows the path of the log file from which it originates. * Knows the host / IP address. * Knows the source type.   **Responsibilities (Does):**   * Associates a log entry to at least one vector. | **Collaborators:**   * The Significant Log Entry class uses in association the name attributes form the Vector class |

|  |  |
| --- | --- |
| **Node** | |
| **Responsibilities (Knows)**:   * Knows its ID * Knows its name * Knows its description * Knows its log entry reference * Knows its log creator * Knows its event type * Knows its icon type * Knows its source * Knows its visibility | **Collaborators**:   * The Node class selects log entry attributes from log entry class to categorize as a node * The Node class selects the type attribute from icon class to select a visual for the node |

|  |  |
| --- | --- |
| **Relationship** | |
| **Responsibilities (Knows)**:   * Know its relationship ID * Know its parent ID (source node of the relationship) * Knows its child ID. * Knows its label. | **Collaborators**:   * A relationship is a client of the node class because it makes references to the parent (source) and child (destination) nodes. |

|  |  |
| --- | --- |
| **Icon** | |
| **Responsibilities (Knows):**   * Knows the path to an image. * Knows the icon name.   **Responsibilities (Does):**   * Provides icon to the node class. | **Collaborators:** |

|  |  |
| --- | --- |
| **Graph** | |
| **Responsibilities (Does):**   * Adds nodes * Edit nodes * Delete nodes * Adds relationships * Edit relationships * Remove relationships * Change icons * Add icons * Move nodes * Delete icons * Handle errors when trying to delete relationships/nodes * Handles errors when trying to edit nodes/relationships * Filter nodes and relationships * Export an image of the graph * Undo changes * Save changes * Keep track of a change list   **Responsibilities (Knows)**   * Know Export format * Know Orientation * Know Interval units * Know Interval * Know Position of nodes * Know Position of relationships | **Collaborators**:   * Graph is a client of Node because it provides its attributes for display and modification purposes * Graph is a client of Relationship because the positions of the relationship will be used to place them on the graph. |

|  |  |
| --- | --- |
| **Log Entries Manager** | |
| **Responsibilities (Does):**   * Check for changes between original log file entries and log entries * Save updated log entries to the log files * Saves updated log entries to local storage. | **Collaborators:**   * The Log Entries Manager class selects the log files from Log File Class to compare any changes in their deltas |

|  |  |
| --- | --- |
| **User Manager** | |
| **Responsibilities (Does):**   * Responsible for managing the number of connected users. * Listens for incoming user connections to the server. * Updates event configuration if there is a change in the number of connected users. (E.g. A user closes the application). * Differentiates the IP addresses between Lead and Non - Lead User   **Responsibilities (Knows):**   * The current I.P. addresses connected to the system. * The number of users connected to the system. | **Collaborators:**   * The User Manager selects the IP address of users from the User class. |

|  |  |
| --- | --- |
| **User** | |
| **Responsibilities (Knows):**   * The IP address of the User * Knows if the User is a Lead for the | **Collaborators:** |

|  |  |
| --- | --- |
| **Validation (Splunk) Handler** | |
| **Responsibilities (Does):**   * Check that the log files contain a timestamp per line * Check that log file contain timestamps that are bounded by the start data, end date, start time, and end time specified in the event configuration. * If the log file is of type CVS and the originator of the log file is from the white team, check that the log file contains timestamps that are within Lower limit of the range and Upper limit of the range * Certify log files as validated log file. * Change status of log file to pass or fail. * Create log files with same source of ingested log files * Cleanse Log Files from unwanted characters if log file is of type TMUX * Cleanse Log Files from blank rows if the log file is of type CVS * Create cleanse certification for Log Files * Create log files with same source of ingested log files * Cleanse Log Files from unwanted characters if log file is of type TMUX * Cleanse Log Files from blank rows if the log file is of type CVS * Create cleanse certification for Log Files | **Collaborators:**   * The Validation Handler class selects attributes of log files from the Log File class to check timestamps syntax and the presence of a timestamp. Update Log Files where changes were made in the cleansing process * The Validation Handler class selects timestamp requirements from the Event Configuration class to select logs the correspond to that timeframe |

|  |  |
| --- | --- |
| **Transcription Handler** | |
| **Responsibilities (Does):**   * Transcribe audio Log Files into text * Translate Image Log Files into text * Transcribe Video Log Files into text | **Collaborators:**   * The Ingestion class the audio, image, and video files from the Log File class to convert into text Log Files. |

**PICK PMR Insights Collective Tool**

Software Design Document

<0.1>

<03/09/2020>

Document Control

Approval

The Guidance Team and the customer shall approve this document.

Document Change Control

|  |  |
| --- | --- |
| Initial Release: | 1.0.0 |
| Current Release: | 1.0.0 |
| Indicator of Last Page in Document: | 10 |
| Date of Last Review: | 3/8/20 |
| Date of Next Review: | 3/10/20 |
| 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:**

Jake Lasley

Steven Roach

**Clients:**

Dr. Oscar Perez

Vincent Fonseca

Herandy Denisse Vazquez

Baltazar Santaella

Florencia Larsen

Erick De Nava

**Software Team Members:**

Jay James

Adrian Sosa

Cristian Molina

Sergio Nogami

Jesus Gomez

Change Summary

The following table details changes made between versions of this document

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Modifier | Description |
| 1.0 | 3/8/20 | ALL | 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)

[1.2.1. Database 1](#_Toc226283117)

[1.2.2. Notification Manager 1](#_Toc226283118)

[1.2.3. Unit Conversion 1](#_Toc226283119)

[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)

[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. Detailed Design 8](#_Toc226283139)

[4.1. Scope 8](#_Toc226283140)

[4.2. Use 8](#_Toc226283141)

[4.3. Components 8](#_Toc226283142)

[4.3.1. Database Manger 8](#_Toc226283143)

[Scenario 1: Perform insert, update, select, or delete query. 8](#_Toc226283144)

[Scenario 1: Upload. 9](#_Toc226283145)

[Scenario 2: Download. 9](#_Toc226283146)

[4.3.2. Unit Conversion 9](#_Toc226283147)

[Scenario 1: Request conversion of units. 9](#_Toc226283148)

[4.3.3. Notification mails 10](#_Toc226283149)

[Scenario 1: Edit data. 10](#_Toc226283150)

[Scenario 2: Add new weather station. 10](#_Toc226283151)

[4.4. Database Schema 11](#_Toc226283152)

# Introduction

## Purpose and Intended Audience

The purpose of the Software Design Document (SDD) is to describe the protocols that will be used in the PICK PMR Insight Collective Knowledge system, as well as to identify the different components of the system, collaborations, responsibilities, and contracts. This document also defines how we shall delegate implementations to programmers and will show how we shall derive the tests for pre and post-conditions. The SDD gives us an improved perspective on the design of the system.

The intended audience for the SDD is the following:

**Guidance Team Members:**

Jake Lasley

Steven Roach

**Customer:**

Dr. Oscar Perez

Vincent Fonseca

Herandy Denisse Vazquez

Baltazar Santaella

Florencia Larsen

Erick De Nava

**Software Team Members:**

Jay James

Adrian Sosa

Cristian Molina

Sergio Nogami

Jesus Gomez

## Scope of Product

<< product description: This should give a brief summary of the product goals and scope. >>

The Lethality Survivability & Human Systems Integration Directorate (LSH) works with the Department of Defense (DOD) to perform operational testing on technology systems to provide secure, resilient capabilities in the expected operational environment. The LSH or White team will analyze the flow of events that occurred during adversarial attacks between the Red and Blue teams to validate what happened during the attack. The Red team performs the attacks and the Blue team tries to mitigate them. The scope of PICK PMR Insights Collective tool will cover the need of the LSH to draw relationships between the events that occurred between the Red and Blue teams. The PMR tool will not draw the correlation of events itself but rather ease the process of drawing the correlations for the analyst.

## References

[1] ￼Elsa Tai (2018), “Prevent, Mitigate, and Recover (PMR) Insight Collective Knowledge System (PICK)

Software Requirements Specification ” UTEP, Sept 2020.

[2] O. Perez et al, Requirements Definition Document, Lethality, Survivability and HSI Directorate, 2019.

[3] “Components and Containers in AWT”. Internet: [https://www.cs.utexas.edu/~mitra/csSpring2009/cs313/lectures/GUIComponents.html](about:blank), 2009 [Jan. 28, 2019]

## Definitions, Acronyms, and Abbreviations

Table 1 shall provide the definitions for terms that are used throughout this document and our project overall.

These are terms that were also mentioned in the course as well as the SRS document.

### Definitions

|  |  |
| --- | --- |
| **Class** | A representation of an object (i.e. a template definition of the methods and variables on an object) |
| **Responsibility** | What an object knows or does |
| **Contract** | Set of cohesive responsibilities |
| **Protocol** | Set of signatures for methods to be implemented |

### Acronyms

|  |  |
| --- | --- |
| **SDD** | Software Design Document |
| **PICK** | PMR Insight Collective Knowledge |
| **SRS** | Software Requirement Specification document |
|  |  |

### Abbreviations

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

## Overview

<< This is the document overview. Describe what is in the following sections >>

The SDD will describe the decomposition of the system, into its components and subsystems and identify collaborations, responsibilities, and contracts.

The SDD has the following sections:

* Decomposition Description
  + This section provides the System Collaboration Graph and a detailed description of the subsystems that Team 12 designed.
* Detailed Description of Subsystem Workspace
  + This section provides a detailed description of the component [TBA]

# 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. >>

Our system was designed following these steps: first, we scanned the SRS looking for nouns and verbs. This gave us an idea of what classes and responsibilities would be needed in the system and we came up with a mapping from nouns to classes and verbs to responsibilities. Then we decided what responsibilities would belong to which classes. Each class description gives a summary of what their responsibilities consist of and whether there were any instances where the classes needed help from other classes.

## 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. >>

Log Ingestion Subsystem

Splunk Integration Component

Graphing Subsystem

Graph integration component

## Dependencies

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

Dependencies on Splunk and Graphviz. Component dependencies will impact development by ensuring that these components of the system are able to be integrated in their current version.

# Detailed Description of Graphing Component

<< For each component or subsystem listed in section 2.2, add a section with a detailed design description of the component.>>

The graphing component integrates the graphing function into the system and provides the analyst with the ability to manipulate the graph to create a timeline of events.

## Component Description

<< The description will contain: The component name, the purpose of the component, and a list of classes contained in the component. If there are several classes, it may be useful to include a detailed component diagram or a UML class diagram. >>

## Class Description Graph Handler

|  |  |
| --- | --- |
| **Class Name**: Graph Handler |  |
| **Superclass**: None |  |
| **Subclasses**: None |  |
| **Private Responsibilities**   Know nodes   * Know vector |  |
| **Contract:** 1. Generate Graph |  |
| **Responsibilities** | **Collaborations** |
| 1. Integrate graphing software into the system 2. Display graph in system | Node ( 1 ) |

<< 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 Description Node

|  |  |
| --- | --- |
| **Class Name**: Node |  |
| **Superclass**: None |  |
| **Subclasses**: None |  |
| **Private Responsibilities**   Know node ID   * Know Visibility * Know Log Entry * Know position * Know Source * Know Event Type * Know Author |  |
| **Contract:** 1. Generate Graph |  |
| **Responsibilities** | **Collaborations** |
| 1. Display node in graph 2. Hold a relationship to another node | Graph Handler (2) |

### Contract Generate Graph

<< 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>

## Class Description <class 2 name>

### Contract <contract 1 name>

# Database

## Database Schema

<< if a database is needed, describe the data layout >>

ER DIAGRAM

<https://drive.google.com/file/d/1hPO-OHS65D5YCQtKQJFmyJR3MeXUeWYh/view?usp=sharing>

RELATIONAL SCHEMA

<https://drive.google.com/file/d/1mI51Xs-EInV2WXvlYDiX0kaL438pI0F_/view?usp=sharing>