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

**Software Design Document**

**3.0**

**05/09/2020**

**Document Control**

**Approval**

The Guidance Team and the customer shall approve this document.

**Document Change Control**

|  |  |
| --- | --- |
| Initial Release: | 1.0 |
| Current Release: | 2.5 |
| Indicator of Last Page in Document: | $$ |
| Date of Last Review: | 2/27/20 |
| Date of Next Review: | 2/28/20 |
| Target Date for Next Update: | 2/28/20 |

**Distribution List**

This following list of people shall receive a copy of this document every time a new version of this document becomes available:

Guidance Team Members: Dr. Roach, Jake Lasley

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

Software Team Members: Ian Hudson, Diego Garcia, Wan Koo, Ricardo Pineda, Jesus Yael Ramos Favela

**Change Summary**

The following table details changes made between versions of this document

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Modifier | Description |
| 1.0 |  |  | Creation of Document |
| 1.1 | 2/22/20 | Ian Hudson | Added component descriptions and contracts for Vector, Node, and Log Entry |
| 1.2 | 2/25/20 | Jesus Ramos | Added component descriptions and contracts for Log Entry and Ingestion |
| 1.3 | 2/25/20 | Diego Garcia | Added component descriptions and contracts for VectorDB and EAR |
| 1.4 | 2/25/20 | Jesus Ramos | Added component descriptions and contracts for Icon, Connection, and Graph |
| 1.5 | 2/26/20 | Wan Koo | Added component descriptions and contracts for Event Configuration as well as added to all other components |
| 1.6 | 2/26/20 | Ricardo Pineda | Uploaded and completed collaboration Diagram |
| 1.7 | 2/26/20 | Ian Hudson | Cleaned up document and prepped for submission |
| 2.0 | 3/25/20 | Wan Koo | Added Basic Database description |
| 2.1 | 3/25/20 | Jesus Ramos | Corrected Sections 2 to 3.2 based on feedback |
| 2.2 | 3/28/20 | Diego Garcia | Corrected sections 3.3-3.6 based on feedback |
| 2.3 | 3/30/20 | Ricardo Pineda | Corrected sections 3.7 - 3.11 based on feedback |
| 2.4 | 3/30/20 | Wan Koo | Made adjustments to contracts and protocols |
| 2.5 | 3/30/20 | Ian Hudson | Added ER Diagram |
| 2.6 | 05/09/2020 | Wan Koo | Fixed filtering and ear classes |
| 2.7 | 05/09/2020 | Jesus Ramos | Added Splunk based on code |
| 2.8 | 05/09/2020 | Diego Garcia | Made adjustments to transcription section |
| 2.9 | 05/09/2020 | Ricardo Pineda | Refactored OCR section based off code |
| 3.0 | 05/09/2020 | Ian Hudson | Final corrections based off source code |

Table of Contents

[**DOCUMENT CONTROL II**](#_30j0zll)

[Approval ii](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.3as4poj)

[Document Change Control ii](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.1pxezwc)

[Distribution List ii](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.49x2ik5)

[Change Summary ii](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.2p2csry)

[**1.** **INTRODUCTION 1**](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.147n2zr)

[1.1. Purpose and Intended Audience 1](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.3o7alnk)

[1.2. Scope of Product 1](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.23ckvvd)

[*1.2.1.* *Database 1*](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.ihv636)

[*1.2.2.* *Notification Manager 1*](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.32hioqz)

[*1.2.3.* *Unit Conversion 1*](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.1hmsyys)

[1.3. References 1](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.41mghml)

[1.4.](#_17dp8vu) Definitions, Acronyms, and Abbreviations 2

[*1.4.1.* *Definitions 2*](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.2grqrue)

[*1.4.2.* *Acronyms 2*](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.vx1227)

[*1.4.3.* *Abbreviations 2*](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.3fwokq0)

[1.5. Overview 2](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.1v1yuxt)

[**2.** **DECOMPOSITION DESCRIPTION 3**](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.4f1mdlm)

[2.1. Scope 3](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.2u6wntf)

[2.2. Use 3](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.19c6y18)

[2.3. Subsystem Description 3](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.3tbugp1)

[2.4. Hierarchy Graphs 4](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.28h4qwu)

[*2.4.1.* *Database Manager Subsystem 4*](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.nmf14n)

[*2.4.2.* *Database Manager (API) 4*](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.37m2jsg)

[*2.4.3.* *Unit Conversion 5*](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.1mrcu09)

[*2.4.4.* *Notification Manager 5*](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.46r0co2)

[**3.** **DEPENDENCY DESCRIPTION 6**](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.2lwamvv)

[3.1. Scope 6](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.111kx3o)

[3.2. Use 6](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.3l18frh)

[3.3. Collaboration Description 6](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.206ipza)

[**4.** **DETAILED DESIGN 8**](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.4k668n3)

[4.1. Scope 8](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.2zbgiuw)

[4.2. Use 8](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.1egqt2p)

[4.3. Components 8](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.3ygebqi)

[*4.3.1.* *Database Manger 8*](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.2dlolyb)

[*Scenario 1: Perform insert, update, select, or delete query. 8*](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.sqyw64)

[*Scenario 1: Upload. 9*](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.3cqmetx)

[*Scenario 2: Download. 9*](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.1rvwp1q)

[*4.3.2.* *Unit Conversion 9*](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.4bvk7pj)

[*Scenario 1: Request conversion of units. 9*](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.2r0uhxc)

[*4.3.3.* *Notification mails 10*](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.1664s55)

[*Scenario 1: Edit data. 10*](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.3q5sasy)

[*Scenario 2: Add new weather station. 10*](https://docs.google.com/document/d/1Nn2qxiCT3Icsn7SvZHuwXz_UbskGojhEoXB6x9V6bF0/edit#heading=h.25b2l0r)

[4.4.](#_qsh70q) Database Schema 11

# **Introduction**

## **Purpose and Intended Audience**

The purpose of creating the Software Design Document (SDD) is to provide a fully enough description of the Prevent, Mitigate, and Recover (PMR) Insight Collective Knowledge System (PICK) system, that will allow comprehensive system design. This design will allow the software development and implementation with a complete understanding of what is to be built and how it is expected to be built. The SDD will identify all the subsystems, components and classes pertaining to the PICK system. This document will help the team analyze more in-depth how the design of the system was chosen as well as leading the team as how it shall be implemented.

The intended audiences for the SDD includes:

Guidance Team Members:

Dr. Roach

Jake Lasley

Customers:

Dr. Oscar Perez

Vincent Foncesa

Herandy Denisse Vazquez

Baltazar Santaella

Florencia Larsen

Erick De Nava

Software Team Members:

Ian Hudson – System Analyst

Diego Garcia – System Architect Analyst

Wan Koo – V&V Supervisor

Ricardo Pineda – Lead Designer

Jesus Yael Ramos Favela – Lead Programmer

## **Scope of Product**

The Adversarial Assessment (AA) is to characterize the operational effects to critical missions caused by threat-representative cyber activity against a unit trained and equipped with a combat system, as well as the effectiveness of defensive capabilities. These threats could be any electronic data exchange that provides an opportunity for a cyber threat to deny, degrade, disrupt, destroy, deceive, or manipulate information critical to military operations. It is important to have all systems that are acquired by the Department of Defense (DOD) need to have strong cybersecurity defenses. The Red Teams will keep logs of cyber-attacks they perform on a combat system. As Red Teams are performing cyber-attacks on the combat system, the Blue team will be responding to these attacks could include operators, maintainers, operational cyber/network defenders, end users, network and/or system administrators, help desk personnel. Blue Team as they are responding to cyber-attacks will be generating log data, either from a Host Based Security System, or other intrusion detection/prevention systems or any chat logs or emails sent, or reports generated. Lethality, Survivability & Human System Integration (LSH) will observe the Red and Blue teams as they are conducting operations. LSH’s objective will be to take observational notes of both teams during the AA as well as collect all applicable logs and artifacts and represent the truth of what happened. Once the AA is complete LSH will take all data collected and create a report that presents the events that happened. The report should show the cyber-attacks the Red Team performed and the actions or inactions the Blue Team took as a result. This software should be able to ingest many types of logs and easily allow an analyst to search for relevant logs, notes, artifacts so the analyst can associate these logs together. Finally, the system will graphically represent what the analyst has associated, thereby representing what happened during the AA.

## **References**

[1] E. Tai Ramirez., “Prevent, Mitigate, and Recover (PMR) Insight Collective Knowledge System (PICK)”, SRS, El Paso, TX USA, February 2020.

## **Definitions, Acronyms, and Abbreviations**

The definitions in this section are given in the context of the product being developed. The intention is to assist the user in their understanding of the document.

### **Definitions**

Table1: Definition of terms used in the report

|  |  |
| --- | --- |
| Contract | Set of Cohesive classes that collaborate among themselves to assist a set of contracts. |
| Pre-Condition | Capture the conditions that must be true in order for the method to execute correctly. |
| Post-Condition | Must clearly state what is true when the method completes execution. |
| Subsystem | Set of cohesive classes that collaborate among themselves to assist a set of contracts. |

### **Acronyms**

This section lists the acronyms used in the document and their associated definitions.

Table 2: Acronyms

|  |  |
| --- | --- |
| DB | Database |
| GUI | Graphical User Interface |
| PICK |  |
| SDD | Software Design Document |
| UTEP | University of Texas at El Paso |
| EAR | Enforcement Action Report |

### **Abbreviations**

This section provides a list of used abbreviations and their associated definitions.

Table 3: Abbreviations

|  |  |
| --- | --- |
| e.g. | For example |
| i.e. | That is |
| CRUD | Create Retrieve Update Delete |
| VDB | Vector Database |

## **Overview**

The Software Design Document comprises the following sections: Decomposition Description, Detailed Description of Component and Database.

The Decomposition Description provides a description of how the component descriptions can be used by designers and maintainers. It will identify major design entities, for purposes such as determining which entity is responsible for specific functions and tracing requirements to design entities.

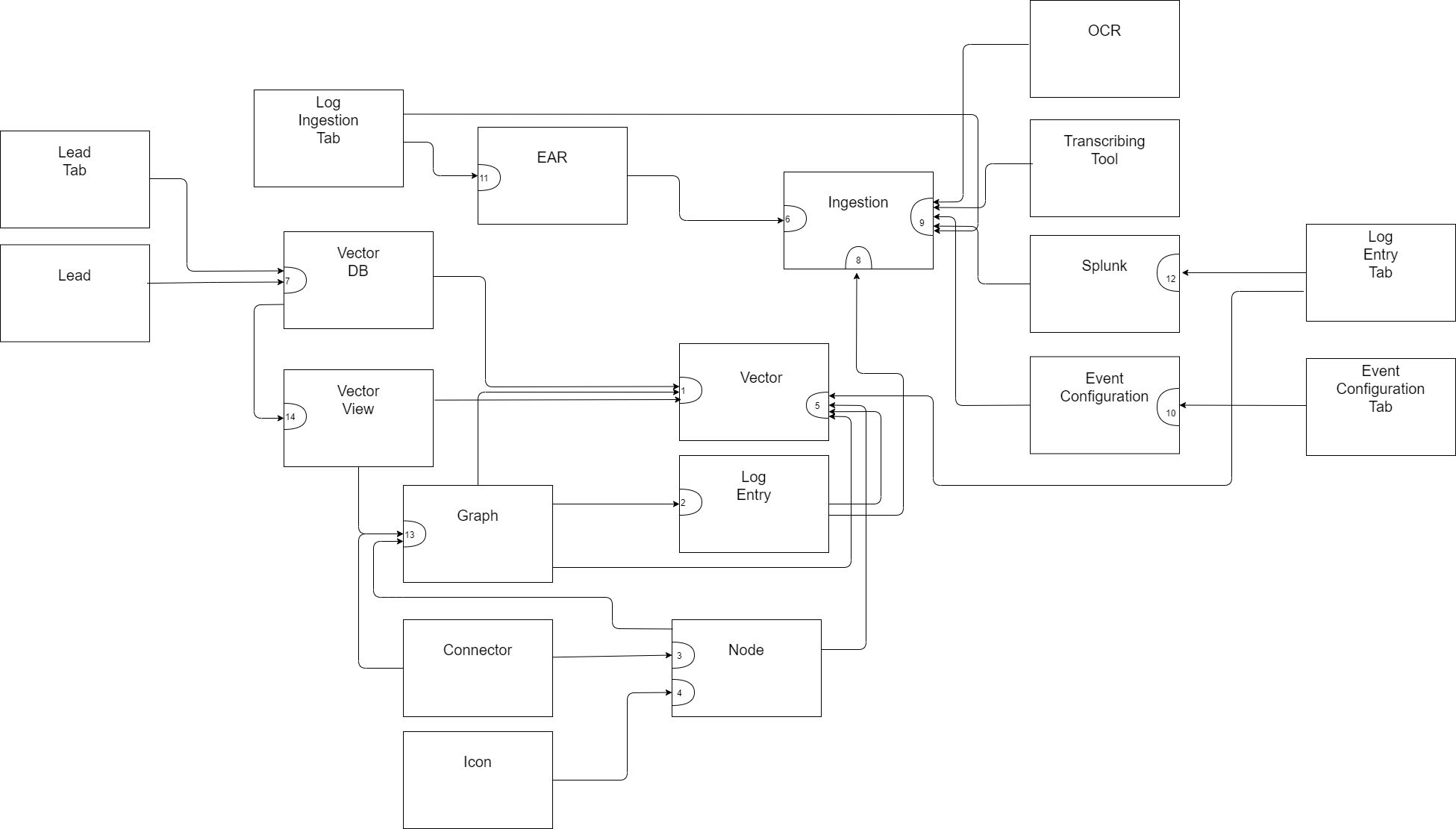
The Detailed Description of Component will provide a section with a detailed design description of each of the components listed in section 2.2, Subsystem and Component Description.

The Database section will describe the database schema and layout required by the system.

# **Decomposition Description**

Our system was designed using the following steps: first, we identified a set of classes, then we determined their responsibilities, and finally their collaborations. Each class description gives a description of the responsibilities that the class should have, as well as contains a list of contracts with their respective methods or functions that relate to those responsibilities. Each contract has a cohesive set of responsibilities it will implement and a list of the other contracts they will interact with to fulfill their task. Each contract is numbered and named so that it will be easier to identify and trace which contracts are interacting together. Once all the classes were properly identified, they were assembled into subsystems. These subsystems enclose groups of classes that collaborate with one another in order to support a set of contracts.

## **System Collaboration Diagram**



## **Subsystem and Component Descriptions**

## **Dependencies**

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

# **Detailed Description of Component <name>**

The purpose of the Tab Subsystem is to generate tab views.

## **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: Vector**

|  |  |
| --- | --- |
| **Class Name:** Vector | |
| **Description:** Responsible for vector management (creation, deletion, etc). | |
| **Contracts:**  **1. CRUD Operations**  - Create Vector  - Delete Vector  - Know template file name    **2. Associate to Vector**  **-** Associate log entries to vectors    **Private Responsibilities:**  - Know log entries  - Know graph | **Collaborations:**    - Log Entry (3.3.1) (CRUD Operations)  - Node (3.4.2) (Associate to Vector)  - Graph |

### **Contract: Create Vector**

This contract is responsible for the creation of vectors.

Protocol: Vector(startDate, endDate, creator, description, name)

Pre-condition: User must create a vector

Post-condition: The system shall return the newly created vector.

Description: This method is used to create a vector.

* + 1. **Contract: Delete Vector**

This contract is responsible for the deletion of vectors.

Protocol: deleteVector(vector)

Pre-condition: User deletes a vector

Post-condition: The system shall return the list of vectors excluding the deleted one.

Description: This method is used to delete a vector.

### **Contract: Associate to Vector**

This contract is responsible for associating a log entry to a vector to mark it as significant.

Protocol: addLogEntry(logEntry)

Pre-condition: User associates a log entry to a vector

Post-condition: The system shall return a vector with the log entries that the user has associated with it.

Description: This method is used to associate log entries to the desired vector.

## **Class Description: Log Entry**

|  |  |
| --- | --- |
| **Class Name:** Log Entry | |
| **Description:** Responsible for log entries in the system | |
| **Contracts:**  **2. Associate to Vector**  **-** Associate log entries to vectors    **3. Create Log Entries**  - Creates log entries from the ingested files    **4. Filter Log Entries**  -Filters imported log entries depending on specified  category    **Private Responsibilities:**  - Retrieving log entries  - Updating log entries  - Deleting log entries | **Collaborations:**    - Ingestion (3.11.1)  - Vector (3.2.3 Associate to vector) |

### **Contract: Create Log Entries**

This contract is responsible for the creation of log entries after the ingestion of the log files.

Protocol: createLogEntries()

Pre-condition: User must have ingested log files in the system.

Post-condition: The system shall return the log entries of the log files that were selected.

Description: This method is used to make log entries from log files.

**3.3.2 Contract: Filter Log Entries**

This contract is responsible for filtering the log entries depending on the specified criteria of the user.

Protocol: filter(logEntryList)

Pre-condition: There must exist log entries.

Post-condition: The system shall return the newly filtered log entry list.

Description: This method is used to filter the current log entries by the way of the users specified need (name, description, time, team, and searching).

## **Class Description: Node**

|  |  |
| --- | --- |
| **Class Name:** Node | |
| **Description:** Responsible for the nodes in the graph of the system | |
| **Contracts:**  **2. Associate Vector**  **-** Associate log entries to vectors    **4. Add Icon to Node**  - Adds icon to the node  - Delete Vector  - Know template file name    **5. Connect Node**  **-** Connect node to connector class    **Private Responsibilities:**  - Change visibility  - Know node information | **Collaborations:**    - Connector |

### **Contract: Add icon to node**

This contract is responsible for adding the icons to nodes on the graph.

Protocol: attachIcon(node)

Pre-condition: There must be pre-existing icons to attach to the graph.

Post-condition: The system shall return the node showing the correct icon on the graph.

Description: Add a type of icon to a no within the graph.

* + 1. **Contract: Connect Node**

This contract is responsible for the connection of the nodes.

Protocol: connectNodes(node1, node2)

Pre-condition: There must be 2 nodes that are on the graph.

Post-Condition: The system shall return a connector between the 2 nodes that were specified.

Description: Creates a relationship between two nodes after connecting them to one another.

## **Class Description: Graph**

|  |  |
| --- | --- |
| **Class Name:** Graph | |
| **Description:** Responsible for all graph functionality (exportation of graph, moving nodes around) | |
| **Contracts:**    **Private Responsibilities:**  - Export graph  - Position node  - Know connections | **Collaborations:**  - Vector (3.2.1 Create Vector) (3.2.3 Delete Vector)  - Log Entry (3.3.1 Create log entry) |

### 

## **Class Description: Connector**

|  |  |
| --- | --- |
| **Class Name:** Connector | |
| **Description:** Responsible for connecting 2 nodes together in the graph | |
| **Contracts:**  **5. Connect node**  **-** Connect node to connector class | **Collaborations:**  - Node (3.4.2 Connect node) |

## **Class Description: Icon**

|  |  |
| --- | --- |
| **Class Name:** Icon | |
| **Description:** Icon is a graphical representation of a node | |
| **Contracts:**  **6. Associate to node:**  - Attach icon to node    **Private Responsibilities:**  - Know icons  - Add icons  - Delete icon  - Edit icon | **Collaborations:**    - Node (3.4.1 Add icon to node) |

### **Contract Associate to node:**

This contract is responsible for providing icons to nodes.

Protocol: attachIcon(node, icon)

Pre-condition: There must be a node associated with a vector.

Post-Condition: The system shall return the node showing the icon on the graph.

Description: Provides a specific icon to a certain node.

## **Class Description: Event Configuration**

|  |  |
| --- | --- |
| **Class Name:** Event Configuration | |
| **Description:** Provides initial vector list, the path of the root directory, and the time range for log entries | |
| **Contracts:**  **7. Validating directory:**  - Checks for valid root directory    **Private Responsibilities:**  - Stores event configuration information | **Collaborations:**    - Vector |

### **Contract Validating directory:**

This contract is responsible for checking that the root directory is valid(has three sub-folders ).

Protocol: checkRoot(rootDirectory)

Pre-condition: Must be provided with an valid existing path of the root directory.

Post-Condition: Root directory is saved in the system for log ingestion.

Description: Checks if the root directory has all three valid folders.

## **Class Description: Enforcement Action Report(EAR)**

|  |  |
| --- | --- |
| **Class Name: Enforcement Action Report** | |
| **Description:** Generate enforcement action reports when invalid logs are sent | |
| **Contracts:**  **8. CRUD Operations**  - Create EAR  - Delete EAR    **9. Refresh EAR Table**  **-** Refreshes table with current invalid logs.    **Private Responsibilities:**  - Stores EAR | **Collaborations:**    - Ingestion(3.11.1 Ingest Log Files) |

### **Contract Create EAR:**

This contract is responsible for providing enforcement action report when ingesting logs are invalid.

Protocol: EAR(name, time, desc, team, error)

Pre-condition: Log file being ingested is failed in validation.

Post-Condition: Prompt analyst the ingestion error.

Description: Enforcement action report is created with a list (if applicable) of log file(s) that are invalid for the system.

### **Contract Delete EAR:**

This contract is responsible for deleting enforcement action report after viewing invalid log files.

Protocol: deleteEAR()

Pre-condition: User deletes enforcement action report.

Post-Condition: There is nothing showing related to enforcement action report.

Description: Deletes the enforcement action report after viewing and accepting invalid log files shown.

**3.9.3. Contract Refresh EAR Table**

This contract is responsible for refreshing the EAR table with the current invalid log files.

Protocol: refreshEARTable(table, earList)

Pre-condition: There must exist invalid log files that were put through validation.

Post-condition: A table will be returned with all the invalid log files.

Description: Refreshes the table on the UI with the invalid log files shown.

## **Class Description: Vector DB**

|  |  |
| --- | --- |
| **Class Name:** Vector DB | |
| **Description:** In charge of storing any changes done in the workspace to be approved/deny by the lead | |
| **Contracts:**  **9. Update vector changes:**  - Push VDB Changes  - Pull VDB Changes    **10. Approving:**  - Approve VDB Changes    **Private Responsibilities:**  - Commit changes | **Collaborations:**    - Vector  - Lead |

### **Contract Approving:**

This contract is responsible for storing all changes approved by the lead analyst

Protocol: approveChange(changeRequest)

Pre-condition: Changes done in the workspace have to be approved by the lead.

Post-Condition: Update VDB for analyst to pull changes.

Description: Lead analyst will either approve or deny changes that have been made to vector database.

* + 1. **Contrat Update VDB changes:**

This contract is responsible for pushing/ pulling any changes made to the VDB

Protocol: pushChanges(vectorDatabase), pullChanges(vectorDatabase)

Pre-condition: Changes have to be done to the VDB.

Post-Condition: Changes shall be sent to the lead to be approved.

Description: Pushes and pull changes made to the vector database after lead analyst has approved them.

## **Class Description: Ingestion**

|  |  |
| --- | --- |
| **Class Name:** Ingestion | |
| **Description:** Responsible of ingesting log files from root directory | |
| **Contracts:**  **11. Validation of logs:**  - Checks if log files are valid to ingest    **12.Ingest log file:**  -Ingest audio files  -Ingest image files  -Ingest log files    **3. Create log entries:**  -Creates log entries    **Private Responsibilities:**  - Cleanse log files | **Collaborations:**    - Log Entries(3.3.1 Create Log Entries)  -Enforcement Action Report(3.9.1 Create EAR)  - OCR  - Transcription  - Splunk  - Event Configuration(3.8.1 Check Valid Root Directory) |

### **Contract: Ingest Log Files**

This contract is responsible for all processes centered around ingestions of log files and the eventual production of Log Entries

Protocol: IngestLogFiles(logFileList)

Pre-condition: Log Files are placed into one of the three folders in the root directory

Post-condition: The system shall return Log Entries into the Log Entry Table

Description: This method is used to ingest all log files into the system as well as produce all Enforcement Action Reports and Splunk Uploads

## **Class Description: Splunk**

|  |  |
| --- | --- |
| **Class Name:** Splunk | |
| **Description:** For uploading and exporting from Splunk | |
| **Contracts:**  **13. Upload to Splunk**   * Upload log entries to splunk     **10. Export from Splunk:**  - Exports the log entries from Splunk index | **Collaborations:**    - Log Entry |

3.12.1 Contract: Upload to Splunk

This contract is responsible for uploading log entries to Splunk

Protocol: splunk\_upload()

Pre-condition: Log entries exist.

Post-condition: This method shall upload all log entries into the Splunk index.

Description: The method shall upload the log entries it has validated and upload them to the Splunk index.

3.12.2 Contract: Export from Splunk

This contract is responsible for exporting the Splunk index

Protocol: splunk\_export()

Pre-condition: There must be an index within Splunk with log entries.

Post-condition: Log entry index from Splunk shall be exported.

Description: The method shall export the specified index from the Splunk database.

## **Class Description: OCR**

|  |  |
| --- | --- |
| **Class Name:** OCR | |
| **Description:** Detects if a file is an image file and puts it into the Tesseract to return a .txt file | |
| **Contracts:**  **14. Translate to .txt**  -Translates the image file to a .txt file | **Collaborations:**    - Log Entry |

3.13.1 Contract: Translate to .txt

This contract is to translate image files to .txt files.

Protocol: imagetotext(path)

Pre-condition: There must be an image file being imported.

Post-condition: The method will return a .txt file with the data residing within the image.

Description: The method shall translate an image to text.

## **Class Description: Transcription**

|  |  |
| --- | --- |
| **Class Name:** Transcription | |
| **Description:** For transcribing audio to .txt files | |
| **Contracts:**  **15. Translate to .txt**  -Translates an audio file to a .txt file | **Collaborations:**    - Ingestion |

3.14.1 Contract: Translate to .txt

This contract is to translate audio files to .txt files

Protocol: transcribe(path)

Pre-condition: There must be an audio file being imported

Post-condition: The method will return a .txt file with the data from the audio

Description: The method shall transcribe audio to text.

# **Database**

## **Database Purpose**

The purpose of the database within PICK is to ensure the persistence of data within the system that does not exist inside of splunk. With this being said data that falls in this category within the system consists of Event Configuration Information, Vector Definitions and subsequent information, Log Entry information, Node information, Enforcement Action Report information, and Log File metadata and status information. All other relevant data can be cached based on the event at hand.

## **Entity Relationship Diagram**

