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

**Software Design Document**

Version 2.0

March 4, 2020

**Document Control**

**Approval**

The Guidance Team and the customer shall approve this document.

**Document Change Control**

|  |  |
| --- | --- |
| Initial Release: | Version 1.0 |
| Current Release: | Version 2.0 |
| Indicator of Last Page in Document: | $ |
| Date of Last Review: | May 7, 2020 |
| Date of Next Review: | May 14, 2020 |
| Target Date for Next Update: | May 20, 2020 |

**Distribution List**

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

Guidance Team Members:

Dr. Gates

Dr. Salamah

Dr. Roach

Elsa Tai Ramirez

Jake Lasley

Customer:

Mr. Vincent Fonseca

Mr. Baltazar Santaella

Ms. Herandy Vasquez

Ms. Florencia Larsen

Dr. Oscar Perez

Mr. Erick De Nava

Software Team Members:

Ana Zepada

Dima AbdelJaber

Ricardo Sanchez

Luis Ochoa

Scott Honaker

**Change Summary**

The following table details changes made between versions of this document

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Modifier | Description |
| 1.0 | 3/4/2020 | Spice Girls | Creation of Document |
| 1.1 | 3/31/2020 | Spice Girls | Completion of Protocols |
| 1.2 | 3/31/2020 | Spice Girls | Collaboration Diagram Update |
| 2.0 | 5/7/2020 | Spice Girls | Code Alignment |
|  |  |  |  |
|  |  |  |  |

Table of Contents

**Document Control** 2

Approval 2

Document Change Control 2

Distribution List 2

Change Summary 2

1. **Introduction 1**
   1. Purpose and Intended Audience 1
   2. Scope of Product 1
   3. References 1
   4. Definitions, Acronyms, and Abbreviations 1
      1. Definitions, Acronyms, and Abbreviations 1
   5. Overview 2
2. **Decomposition Description 3**
   1. System Collaboration Diagram 3
   2. System and Component Descriptions 3
      1. User Interaction Subsystem 3
      2. Graphing Subsystem 4
      3. File Storage Subsystem 4
      4. Log Ingestion Subsystem 4
   3. Dependencies 5
3. **Detailed Description User Interaction Subsystem 6**
   1. Component Description 6
   2. Class Description: Custom Icon Widget 6
   3. Class Description: Enforcement Action Report Widget 6
   4. Class Description: Export Configuration 7
   5. Class Description: Filter Configuration 7
   6. Class Description: Log File Configuration 8
   7. Class Description: Graph Operations 8
   8. Class Description: Pick Start Page 9
   9. Class Description: Main Window 9
   10. Class Description: Settings Window 11
4. **Detailed Description Graphing Subsystem 14**
   1. Component Description 14
   2. Class Description: Vector 14
   3. Class Description: Nodes 15
   4. Class Description: Icon 17
   5. Class Description: Relationship 18
5. **Detailed Description File Storage Subsystem 20**
   1. Component Description 20
   2. Class Description: Splunk Handler 20
   3. Class Description: DB Handler 21
6. **Detailed Description Log Ingestion Subsystem 23**
   1. Component Description 23
   2. Class Description: Log File 23
   3. Class Description: Log Entry 24
   4. Class Description: Log Cleanser 24
   5. Class Description: Log Validator 24
   6. Class Description: Transcription 24
   7. Class Description: Audio Transcription Interface 25
   8. Class Description: Image Transcription Interface 25
7. **Database Description 26**
   1. Data Schema 26

# Introduction

## Purpose and Intended Audience

The purpose of creating the software design document is to aid in the development of the design and structure of the system that the team will build. It gives guidance on the design. The SDD document shows how the system can be separated into components to simplify the implementation. The intended audience are the guidance team, the software engineering teams, and the clients: Mr. Vincent Fonseca, Mr. Baltazar Santaella, Ms. Herandy Vasquez, Ms. Florencia Larsen, Dr. Oscar Perez, and Mr. Erick De Nava.

## Scope of Product

PICK shall be a tool used by the white team analysts in order to efficiently sort through documents pertaining to adversarial assessments. These include computer log files and screenshots. These documents are then used to piece together an attack log to analyze the way in which the blue team responds to the red team’s attack. Without the tool, analysts are currently having to open up all the files that they wish to reference in their attack graphs. In addition, this system shall simplify the way in which data is filled for nodes in the attack graph. The ultimate goal of the system is to reduce the amount of time doing each analysis to approximately two weeks.

LSH recognizes the complexity and the time it takes to

analyze the applicable logs, observation notes, and other artifacts gathered from an adversarial assessment from the red, blue, and white teams and generate a report that presents the events that took place during the adversarial assessment. They want a system that would aid their analysts in correlating red team’s activities to

blue team’s responses and represent the events that took place during an adversarial assessment graphically.

UTEP and LSH are collaborating to develop Prevent, Mitigate, and Recover (PMR) Insight Collective Knowledge System (PICK) that will provide the ability to correlate red team’s activities to blue team’s responses and graphically represent the events that took place during an

adversarial assessment.

## References

[1] Tai Ramirez, Elsa, *Prevent, Mitigate, and Recover (PMR) Insight Collective Knowledge System (PICK)* [SRS] El Paso, TX: UTEP, 2020

## Definitions, Acronyms, and Abbreviations

### **Definitions**

|  |  |
| --- | --- |
| 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. |
| Data Validation | Data validation is the process of inspecting data in the cleansed log files based on predefined data validation rules. |
| Log Entry | Splunk takes the validated log files and converts them into normalized data.  The normalized data are called log entries.  Users of the system can filter and edit log entries. |
| Significant Log Entry | A log entry selected by the user and associated with a vector. The attributes are the same as for a log entry. The system stores significant log entries. Splunk stores log entries in the normalized data files. |
| Timestamp | Denotes time in hours:minutes, date in month:date:year, and section in am/pm. |
| Significant log entry | Denotes a log entry that is associated to at least one vector. |

### **A**c**ronyms**

|  |  |
| --- | --- |
| UTEP | The University of Texas at El Paso |
| LSH | The Lethality, Survivability, and HSI Directorate |
| SDD | Software Design Document |
| PICK | Prevent, Mitigate, and Recover (PMR) Insight Collective Knowledge |

### **A**b**breviations**

|  |  |
| --- | --- |
| e.g | For example |
| i.e | That is |

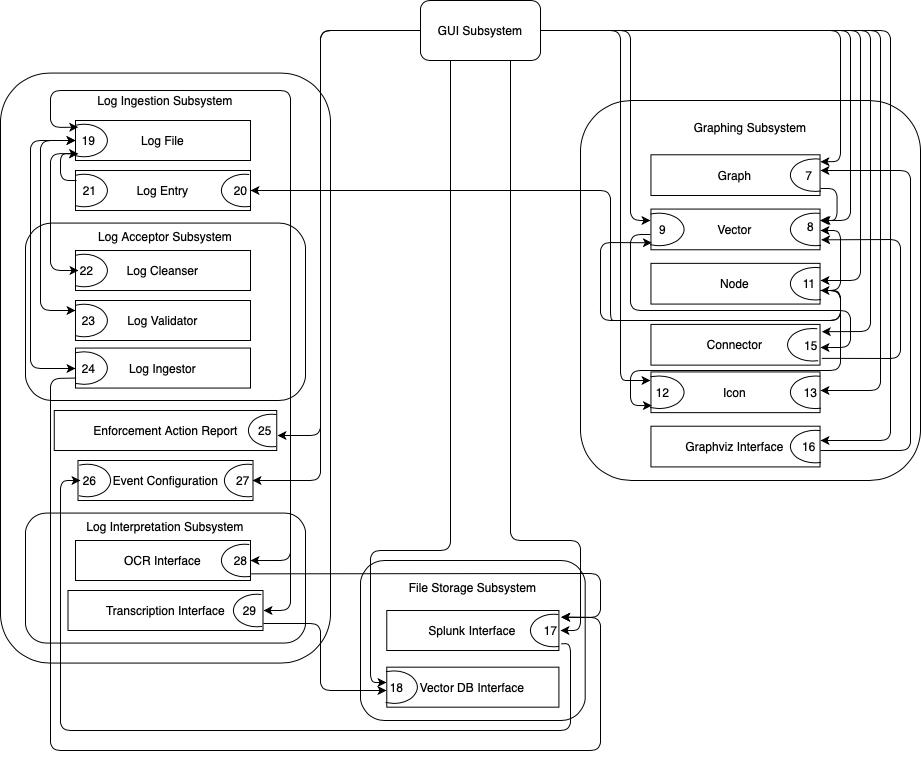
## Overview

The document is divided into six sections. The first section gives a description of the overall system and how all the components relate to each-other. The following four sections are detailed descriptions of subsections of the system. Each of the detailed descriptions of subsystems gives the subsystem name, its general description and classes. It also goes into describing the subsystem’s responsibilities and contracts. The database section shows the relational diagram of the database as well as the schema for the database.

# Decomposition Description

## System Collaboration Diagram

The PICK System will be divided as follows:



**Figure 1: System Collaboration Diagram**

## Subsystem and Component Descriptions

The following section will describe the subsystems of the system and the classes they contain.

### 2.2.1. User Interface Subsystem

The GUI subsystem will handle any input and output of the system. It will allow the user to alter the system through prompts.

The classes of the subsystem are:

* Custom Icon Widget
* Enforcement Action Report Widget
* Export Configuration
* Filter Configuration
* Log File Configuration
* Graph Operations
* Pick Start Page
* Main Window
* Settings Window

The contracts of the subsystem include:

* Display Icon Creator
* Display Enforcement Action Report
* Display Export Configuration
* Display Filter Configuration
* Display Log File Configuration
* Generate Json
* Open Correct Window
* Display Main Window
* Display Settings Window

### 2.2.2. Graphing Subsystem

The classes of the subsystem include:

* Vector
* Node
* Icon
* Relationship

The contracts of the subsystem include:

* Know Vector Components
* Change Vector Components
* Know Node Details
* Change Node Details
* Know Icon Components
* Know Relationship Components
* Change Relationship Components

### 2.2.3. File Storage Subsystem

The file storage subsystem has interfaces to the vector database and to Splunk.

The classes for the subsystem include:

* Splunk Interface
* Vector DB Interface

The contracts of the subsystem include:

* Implement Splunk
* DB Interaction

### 2.2.4. Log Ingestion Subsystem

The log ingestion subsystem allows the user to create an event. It will allow the user to designate directories, access log files, interpret the log files, and split the log files into log entries.

The classes for the subsystem include:

* Log File
* Log Entry
* Log Acceptor Subsystem
  + Log Cleanser
  + Log Validator
* Interpretation Subsystem
  + Transcription
  + Audio Transcription Interface
  + ImageTranscription Interface

The contracts of the subsystem include:

* Know File Attributes
* Know Entry Attributes
* Create Entry
* Cleanse Logs
* Validate Logs
* Convert Audio Logs, Video Logs, and Image Logs to Text Logs

## Dependencies

PICK is intended to run on Kali Linux and will be programmed in Python.

Log File will be dependent on an OCR and Transcription tool to convert visual and audio logs to text. It is also dependent on Splunk and a database that will function as file storage. Splunk will additionally aid in file filtering and searching.

The GUI will be using Graphviz to convert Graphs to attack graphs and timelines.

# Detailed Description User Interaction Subsystem

## Component Description

**Component name:** User Interaction Subsystem

**Purpose:** To allow the user to input information into the system and to view the state of the system.

**Classes:** User Interface

## 3.2. Class Description: Custom Icon Widget

|  |  |
| --- | --- |
| **Class:** Custom Icon Widget | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** | |
| **Contract 1:** Display Icon Creator | |
| **Responsibilities** | **Collaborations** |
| 1. Display new icon widget 2. Set icon image 3. return icon image | 1. Database Handler (xxxx) 2. Database Handler (xxxx) |

initUI()

**pre:** none

**post:** displays the new icon widget

image()

**pre:** calling icon must be a valid icon object

**post:** returns the image pertaining to the icon object from which it is called

setImage(String value)

**pre:** calling icon must be a valid icon object, value must be a valid image

**post:** sets the icon’s image to the new value

## 3.3. Class Description: Enforcement Action Report Widget

|  |  |
| --- | --- |
| **Class:** Enforcement Action Report Widget | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:**   1. Highlight line 2. Unhighlight 3. Highlight block 4. Update text changes | |
| **Contract 2:** Display Enforcement Action Report | |
| **Responsibilities** | **Collaborations** |
| 1. Display enforcement action report 2. Accept updates to Splunk | 1. Splunk handler (xxxx) |

initUI()

**pre:** none

**post:** displays the enforcement action report widget and input is used to ingest proper log files to Splunk

## 3.4. Class Description: Export Configuration

|  |  |
| --- | --- |
| **Class:** Export Configuration | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** | |
| **Contract 3:** Display Export Configuration | |
| **Responsibilities** | **Collaborations** |
| 1. Display export configuration GUI 2. Accept updates to export configuration |  |

initUI()

**pre:** none

**post:** displays the export configuration GUI and uses input to update export configuration which is stored on a local shelf

## 3.5. Class Description: Filter Configuration

|  |  |
| --- | --- |
| **Class:** Filter Configuration | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** | |
| **Contract 4:** Display Filter Configuration | |
| **Responsibilities** | **Collaborations** |
| 1. Display filter configuration GUI 2. Accept updates to filter configuration |  |

initUI()

**pre:** none

**post:** displays the filter configuration GUI and uses input to update filter configuration which is stored on a local shelf

## 3.6. Class Description: Log File Configuration

|  |  |
| --- | --- |
| **Class:** Log File Configuration | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** | |
| **Contract 5:** Display Log File Configuration | |
| **Responsibilities** | **Collaborations** |
| 1. Display export configuration GUI 2. Accept updates to export configuration |  |

initUI()

**pre:** none

**post:** displays the log file configuration GUI and uses input to update log file configuration which is stored on a local shelf

## 3.7. Class Description: Graph Operations

|  |  |
| --- | --- |
| **Class:** Graph Operations | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** | |
| **Contract 6:** Generate Json | |
| **Responsibilities** | **Collaborations** |
| 1. Parses vectors for Graphviz | 1. Vector |

graph\_from\_raw\_vector(Vector vector)

**pre:** vector must be a valid vector

**post:** generates a Json for use by Graphviz

## 3.8. Class Description: Pick Start Page

|  |  |
| --- | --- |
| **Class:** Pick Start Page | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** | |
| **Contract 27:** Open Correct Window | |
| **Responsibilities** | **Collaborations** |
| 1. Prompts user for what they want to do | 1. Main Window (7), Settings Window (8) |

pickStartPage()

**pre:** none

**post:** either settings view or main view is initialized depending on the user’s selection

## 3.8. Class Description: Main Window

|  |  |
| --- | --- |
| **Class:** Main Window GUI | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:**   1. Updates an entry from table 2. Determines what entry field is selected 3. Determines what node is selected 4. Determines what edge is selected 5. Determines what node is invoked 6. Determines what edge is invoked 7. Determines what node is removed 8. Determines what edge is removed | |
| **Contract 7:** Display Main Window | |
| **Responsibilities** | **Collaborations** |
| 1. Display graph 2. Removes node 3. Saves graph 4. Loads graph 5. Creates new graph 6. Adds node 7. Deletes nodes 8. Add edge 9. Delete edte 10. Updates views 11. Opens filter configuration 12. Opens settings 13. Populates search table 14. Populates table 15. Populates graph 16. Generates models | 1. Graph Operations 2. Database Handler, Graph Operations 3. Graphviz 4. Graphviz 5. Graph Operations 6. Node 7. Node 8. Relationships 9. Relationships 10. Graph Operations 11. Filter Configuration 12. Settings 13. Vector Database 14. Database Handler 15. Database Handler 16. Graph Operations\ |

save()

**pre:** none

**post:** saves a graph to a json wherever the user selects

new()

**pre:** none

**post:** generates a new graph the displays it

load()

**pre:** json file selected must be in the proper format

**post:** json file is displayed as a graph

save()

**pre:** none

**post:** saves a graph to a json wherever the user selects

addNode()

**pre:** none

**post:** blank node is made and added to the vector the graph in the main window represents

deleteNode()

**pre:** none

**post:** node is deleted and removed from the vector the graph in the main window represent

addEdge()

**pre:** none

**post:** blank edge is made and added to the vector the graph in the main window represents

deleteEdge()

**pre:** none

**post:** edge is deleted and removed from the vector the graph in the main window represent

updateViews(self)

**pre:** none

**post:** models generated, table, search table and graph populated

openExportConfig(self)

**pre:** none

**post:** opens export configuration

openFilterConfig(self)

**pre:** none

**post:** opens filter configuration

openSettings(self)

**pre:** none

**post:** opens settings

populateSearchTable(self)

**pre:** none

**post:** search table is populated with log entries

populateTable(self)

**pre:** none

**post:** table is populated with nodes and relationships

populateGraph(self)

**pre:** none

**post:** graph is populated with nodes and relationships

## 3.9. Class Description: Settings Window

|  |  |
| --- | --- |
| **Class:** Settings Window GUI | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** | |
| **Contract 8:** Display Settings Window | |
| **Responsibilities** | **Collaborations** |
| 1. Display settings panel 2. Add project directory paths 3. Add project name and description 4. Add project start and end times 5. Add splunk credentials 6. Save project settings 7. Cleanse project files 8. Transcribe audio/image files 9. Validate project files 10. Ingest project files 11. Add icons 12. Edit Icons 13. Delete Icons 14. Open main window | 1. Event Configuration 2. Event Configuration 3. Event Configuration 4. Splunk Handler 5. Log Cleanser 6. Transcription 7. Log Validator 8. Log Ingestor |

btn(int index)

**pre:** none

**post:** update the settings configuration stackview to the selected index

setDir(int index)

**pre:** none

**post:** opens a window to select a directory, then it fills the respective text input field (which is selected with the index parameter) with a selected directory

updateED()

**pre:** none

**post:** update event name and description on the settings database

updateDirLE()

**pre:** none

**post:** update directory path on the settings database

vaalidateTime()

**pre:** none

**post:** validate start and end time based on previous established criteria, if they are not valid, a dialog error message will pop-up and inform the user to fix them

validateRoot()

**pre:** the directory text input fields must contain non empty data

**post:** validate root directories based on previous established criteria, if they are not valid, a dialog error message will pop-up and inform the user to fix them

validateCredentials()

**pre:** the splunk text fields contain non empty data

**post:**  validate credentials and establish a connection to local splunk server, if they are not valid, a dialog error message will pop-up and inform the user to fix them

startCleanse()

**pre:** files are present in the directories assigned

**post:** if files contain blank lines, or unreadable lines, those will be deleted and the file will be updated, nothing will occur to the files if they do not contain any of those

startAudioTranscription()

**pre:** audio files are present in the directories assigned

**post:** audio files will be transcribed to text, a copy with a different extension will be created

startImageTranscription()

**pre:** image files are present in the directories assigned

**post:** image files will be transcribed to text, a copy with a different extension will be created

startValidation()

**pre:** directories must be valid, and existing files must be present

**post:** generate enforcement action report for the files validated

organizeDirectories()

**pre:** directories must be configured accordingly

**post:** organize the raw files and copies in separate directories

startIngestion()

**pre:** directories must be configured accordingly

**post:** initialize the data ingestion process

startDataPopulation()

**pre:** none

**post:** redirects entries from splunk to the entry database document

openMain()

**pre:** none

**post:** open main window screen and close settings window

populateIcons()

**pre:** at least one icon file is present in collection

**post:** populates tables with the icon name, filepath, and an image of the icon

changeInIconTable()

**pre:** row must exist before it can be selected

**post:** point current selection to a specific file in the icon configuration table

showEditButtons()

**pre:**  existing non-blank entry must be selected on the icon configuration table

**post:** displays and opens access to the edit and delete buttons

deleteIconConfigEdit()

**pre:** an existing icon must be selected

**post:** opens a dialog to ask for confirmation then proceeds to delete icon on collection

openIconConfigEdit()

**pre:** an existing icon must be selected

**post:** opens icon configuration window to configure fields of icon and update on icon collection

openIconConfigAdd()

**pre:** none

**post:** opens window configuration to add a new icon to icon collection

# Detailed Description Graphing Subsystem

## 4.1. Component Description

**Component Name:** Graphing Subsystem

**Purpose:** Knows about the graph and its components

**Classes:** Vector, Node, Relationships, Icon

## 4.2. Class Description: Vector

|  |  |
| --- | --- |
| **Class:** Vector | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** | |
| **Contract 9:** Know Vector Components | |
| **Responsibilities** | **Collaborations** |
| 1. Know vector name 2. Know vector time range 3. Know vector description 4. Know nodes belonging to vector 5. Know relationships belonging to vector | 1. DB Handler 2. DB Handler 3. DB Handler 4. DB Handler 5. DB Handler |
| **Contract 10:** Change Vector Components | |
| **Responsibilities** | **Collaborations** |
| 1. Change vector name 2. Change vector time range 3. Change vector description 4. Add nodes 5. Delete nodes 6. Add relationships 7. Delete relationships | 1. DB Handler 2. DB Handler 3. DB Handler 4. DB Handler 5. DB Handler 6. DB Handler 7. DB Handler |

changeVectorName(String name)

**pre:** none

**post:** if name does not belong to any other vector, vector name is changed to name; otherwise, no change occurs to vector

changeVectorTimeRange(Timestamp start, Timestamp end)

**pre:** none

**post:** vector time range is changed to that between start and end

changeVectorDescription(String description)

**pre:** none

**post:** vector description is changed to description

deleteNode(Node node)

**pre:** node must be within the vector

**post:** node is deleted to the vector; nothing else is changed

addRelationship(String name, Node source, Node destination)

**pre:** source, and destination must all be valid; source and destination are both within vector

**post:** a new relationship of called name will connect source and destination within vector

deleteRelationship(Relationship relationship)

**pre:** relationship and vector are valid; relationship is within vector

**post:** relationship is deleted and is no longer referenced by vector

## 4.3. Class Description: Nodes

|  |  |
| --- | --- |
| **Class:** Nodes | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** Knows the next node number in sequence, knows the node name, knows the node id, knows the node timestamp, knows the node’s related file path (if any) | |
| **Contract 11:** Know Node Details | |
| **Responsibilities** | **Collaborations** |
| 1. Knows node name 2. Knows node description 3. Knows node timestamp 4. Knows node icon 5. Knows related log file 6. Know node visibility 7. Know name visibility 8. Know id visibility 9. Know description visibility 10. Know log creator visibility 11. Know event type visibility 12. Know source visibility | 1. DB Handler 2. DB Handler 3. DB Handler 4. DB Handler 5. DB Handler 6. DB Handler 7. DB Handler 8. DB Handler 9. DB Handler 10. DB Handler 11. DB Handler 12. DB Handler |
| **Contract 11:** Change Node Details | |
| **Responsibilities** | **Collaborations** |
| 1. Create node 2. Change icons for nodes 3. Change name for nodes 4. Change description for nodes 5. Change node timestamp 6. Change node visibility 7. Change name for nodes 8. Change name visibility 9. Change id visibility 10. Change description visibility 11. Change log creator visibility 12. Change event type visibility 13. Change source visibility | 1. DB Handler 2. DB Handler 3. DB Handler 4. DB Handler 5. DB Handler 6. DB Handler 7. DB Handler 8. DB Handler 9. DB Handler 10. DB Handler 11. DB Handler 12. DB Handler 13. DB Handler |

Node(String name)

**pre:** none

**post:** log file created with node id being the next number in the sequence, timestamp being 00:00 00:00:0000, description left blank, and name as provided

Node(String name, LogFile file)

**pre:** log file must be valid

**post:** log file created with node id being the next number in the sequence, timestamp of log file, description of log file and name as provided

changeIcon(String name)

**pre:** name must be one of the names of icons already stored

**post:** the icon for the node changes to match the icon with the given name

changeName(String name)

**pre:** none

**post:** the name of the icon is changed to name

changeDescription(String name)

**pre:** none

**post:** the name of the icon is changed to name

changeTimestamp(Timestamp time)

**pre:** time must be a valid Timestamp

**post:** the time for the node changes to match the time provided

deleteNode()

**pre:** none

**post:** only given node deleted; node is deleted from the vector containing it

changeVisibility(boolean switch)

**pre:** none

**post:** the node becomes/stays visible if switch is true, the node becomes/stays invisible if switch is false

changeIDVisibility(boolean switch)

**pre:** none

**post:** the node id becomes/stays visible if switch is true, the node id becomes/stays invisible if switch is false

changeNameVisibility(boolean switch)

**pre:** none

**post:** the node name becomes/stays visible if switch is true, the node name becomes/stays invisible if switch is false

changeDescriptionVisibility(boolean switch)

**pre:** none

**post:** the node description becomes/stays visible if switch is true, the node description becomes/stays invisible if switch is false

changeLogEntryVisibility(boolean switch)

**pre:** none

**post:** the node log entry becomes/stays visible if switch is true, the node log entry becomes/stays invisible if switch is false

changeSourceVisibility(boolean switch)

**pre:** none

**post:** the node source becomes/stays visible if switch is true, the node source becomes/stays invisible if switch is false

changeLogCreatorVisibility(boolean switch)

**pre:** none

**post:** the node log creator becomes/stays visible if switch is true, the node log creator becomes/stays invisible if switch is false

changeEventTypeVisibility(boolean switch)

**pre:** none

**post:** the node event type becomes/stays visible if switch is true, the node event type becomes/stays invisible if switch is false

## 4.4. Class Description: Icon

|  |  |
| --- | --- |
| **Class:** Icon | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** | |
| **Contract 12:** Know Icon Components | |
| **Responsibilities** | **Collaborations** |
| 1. Know icon name 2. Know icon path | 1. DB Handler 2. DB Handler |
| **Contract 13:** Change Icon Components | |
| **Responsibilities** | **Collaborations** |
| 1. Create icon 2. Delete icon 3. Change icon name 4. Change icon path | 1. DB Handler 2. DB Handler 3. DB Handler 4. DB Handler |

createIcon(String name, String filePath)

**pre:** filePath must lead to an image

**post:** icon called name and path filepath is created

deleteIcon()

**pre:** none

**post:** icon is deleted, it is removed from nodes containing it

changeIconName(String newName)

**pre:** none

**post:** icon’s name is change to newName

changIconPath(String newPath)

**pre:** newPath must be contain an image file

**post:** icon’s path is changed to newPath

## 4.5. Class Description: Relationship

|  |  |
| --- | --- |
| **Class:** Relationship | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** | |
| **Contract 14:** Know Relationship Components | |
| **Responsibilities** | **Collaborations** |
| 1. Know relationship name 2. Know source node 3. Know destination node | 1. DB Handler 2. DB Handler 3. DB Handler |
| **Contract 15:** Change Relationship Components | |
| **Responsibilities** | **Collaborations** |
| 1. Create relationship 2. Change relationship name 3. Change source node 4. Change destination node 5. Delete relationship | 1. DB Handler 2. DB Handler 3. DB Handler 4. DB Handler 5. DB Handler |

createRelationship(Vector vector, String name, Node source, Node destination)

**pre:** none

**post:** relationship called name will connect source and destination within vector

changeRelationshipName(Relationship relationship, Name name)

**pre:** none

**post:** relationship name is changed to name

changeRelationshipSource(Vector vector, Relationship relationship, Node newSource)

**pre:** relationship and newSource must both be in vector

**post:** relationship’s new source node is newSource

changeRelationshipDestination(Vector vector, Relationship relationship, Node newDestination)

**pre:** relationship and newDestination must both be in vector

**post:** relationship’s new destination node is newDestination

deleteRelationship(Vector vector, Relationship relationship)

**pre:** none

**post:** relationship is deleted and removed from vector

# Detailed Description File Storage Subsystem

## 5.1. Component Description

**Component name:** File Storage Subsystem

**Purpose:** Persistently stores changes made to vectors, nodes, relationships, icons and graphs.

**Classes:** Splunk Interface, Vector DB Interface

## 5.2. Class Description: Splunk Handler

|  |  |
| --- | --- |
| **Class:** Splunk Handler | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** | |
| **Contract 16:** Implement Splunk | |
| **Responsibilities** | **Collaborations** |
| 1. get index from splunk 2. set index 3. upload files to splunk 4. add an index 5. delete an index 6. print users 7. print indexes 8. print jobs 9. create jobs 10. create a new user 11. add a directory 12. print inputs 13. download log entries from splunk 14. cleanse log files 15. Validate log files | 1. Event Configuration (26) |

get\_index(string index) returns index

**pre:** access to Splunk must be valid and index exists

**Post:** Index returned

set\_index(string name)

**pre:** access to Splunk must be valid

**Post:** index name is set

upload\_file(string index, string path)

**pre:** access to Splunk must be valid

**post:** Files are now within splunk

add\_index(string index)

**pre:** access to Splunk must be valid

**Post:** index is created

delete\_index(string index)

**pre:** access to Splunk must be valid

**Post:** index is deleted

print\_users() returns users

**pre:** access to Splunk must be valid

**Post:** users are printed

print\_indexes() returns indexes

**pre:** access to Splunk must be valid

**Post:** indexes are printed

print\_jobs() returns jobs

**pre:** access to Splunk must be valid

**Post:** jobs are printed

create\_jobs()

**pre:** access to Splunk must be valid

**Post:** jobs are created

create\_new\_user(string username, string password, string role, string fullname)

**pre:** access to Splunk must be valid

**Post:** user is created

add\_directory(string path)

**pre:** access to Splunk must be valid

**Post:** directory added

print\_inputs() returns inputs

**pre:** access to Splunk must be valid

**Post:** inputs are printed

download\_log\_files()

**pre:** access to Splunk must be valid

**Post:** log files downloaded as log entries

cleanse(string file)

**pre:** access to Splunk must be valid

**Post:** file is cleansed. All empty rows are removed

validate(string file, string event\_start, string event\_end)

**pre:** access to Splunk must be valid

**Post:** files are validated

## 5.3. Class Description: DB Handler

|  |  |
| --- | --- |
| **Class:** DB Handler | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** | |
| **Contract 17:** Log Entry Storage | |
| **Responsibilities** | **Collaborations** |
| 1. Retrieve log entries 2. Create log entry 3. Delete log entry 4. Change log entry content 5. Change log entry timestamp 6. Change log entry host 7. Change log entry source 8. Change log entry source type |  |
| **Contract 18:** Vector Storage | |
| **Responsibilities** | **Collaborations** |
| 1. Create vector 2. Retrieve vector’s name 3. Retrieve vector 4. Retrieve all vectors |  |

pushChanges(DBHandler database)

**pre:** database must be accessible

**post:** changes are pushed for approval

approveChanges(DBHandler database)

**pre:** database must be accessible and user must be a lead

**post:** changes are approved

pullChanges()

**pre:** database must be accessible

**post:** the system shall reflect changes on the main database

# Detailed Description Log Ingestion Subsystem

## 6.1. Component Description

**Component name:** Log Ingestion Subsystem

**Purpose:** Deals with the initial input of files into the system

**Classes:** Log File, Log Entry, Log Cleanser, Log Validator, Log Ingestor, Enforcement Action Report, Evet Configuration, OCR Interface, Transcription Interface

## 6.2. Class Description: Log File

|  |  |
| --- | --- |
| **Class:** Log File | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** | |
| **Contract 19:** Know File Attributes | |
| **Responsibilities** | **Collaborations** |
| 1. Know log file path 2. Know log file contents |  |

## 6.3. Class Description: Log Entry

|  |  |
| --- | --- |
| **Class:** Log Entry | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** | |
| **Contract 20:** Know Entry Attributes | |
| **Responsibilities** | **Collaborations** |
| 1. Know log file path 2. Know timestamp 3. Know log entry content 4. Know source |  |
| **Contract 21:** Create Entry | |
| **Responsibilities** | **Collaborations** |
| 1. Create log entry |  |

## 6.4. Class Description: Cleanser

|  |  |
| --- | --- |
| **Class:** Cleanser | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** | |
| **Contract 22:** Cleanse Logs | |
| **Responsibilities** | **Collaborations** |
| 1. Remove empty rows and columns |  |

cleanse(File file)

**pre:** file must be a valid log file

**post:** file is cleansed by removing empty rows and columns

## 6.5. Class Description: Validator

|  |  |
| --- | --- |
| **Class:** Log Validator | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** | |
| **Contract 23:** Validate Logs | |
| **Responsibilities** | **Collaborations** |
| 1. Check if log is in a given time range 2. Change validated status 3. Know validated status | 1. Log File (19) 2. Log File (19) 3. Log File (19) |

validate(LogFile file)

**pre:** file must be a valid log file which has been cleansed, validated status must be false

**post:** file is cleansed by checking if it si within a given time range and validated status becomes true

searchByPattern(line)

**pre:** file must be a valid log file which has been cleansed, validated status must be false

**post:** file is cleansed by checking if it si within a given time range and validated status becomes true

getEnforcementReport()

**pre:** file must be a valid log file which has been cleansed, validated status must be false

**post:** file is cleansed by checking if it si within a given time range and validated status becomes true

## 6.6. Class Description: Transcription

|  |  |
| --- | --- |
| **Class:** Transcription | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** | |
| **Contract 29:** Convert Audio Logs, Video Logs and Image Logs to Text Logs | |
| **Responsibilities** | **Collaborations** |
| 1. Convert audio logs to text logs 2. Convert video logs to text logs 3. Convert image logs to text logs | 1. Audio Transcription Interface 2. Audio Transcription Interface 3. Image Transcription Interface |

transcribeAudio(String filePath, fileDirectory)

**pre:** filePath must lead to a valid audio file

**post:** creates a file in fileDirectory containing the text version of the audio file.

## 6.7. Class Description: Audio Transcription Interface

|  |  |
| --- | --- |
| **Class:** Audio Transcription Interface | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** | |
| **Contract 29:** Convert Audio Logs to Text Logs | |
| **Responsibilities** | **Collaborations** |
| 1. Convert audio logs to text logs |  |

transcribeAudio(String filePath, fileDirectory)

**pre:** filePath must lead to a valid audio file

**post:** creates a file in fileDirectory containing the text version of the audio file.

## 6.8. Class Description: Image Transcription Interface

|  |  |
| --- | --- |
| **Class:** Image Transcription Interface | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** | |
| **Contract 29:** Convert Audio Logs to Text Logs | |
| **Responsibilities** | **Collaborations** |
| 1. Convert image logs to text logs |  |

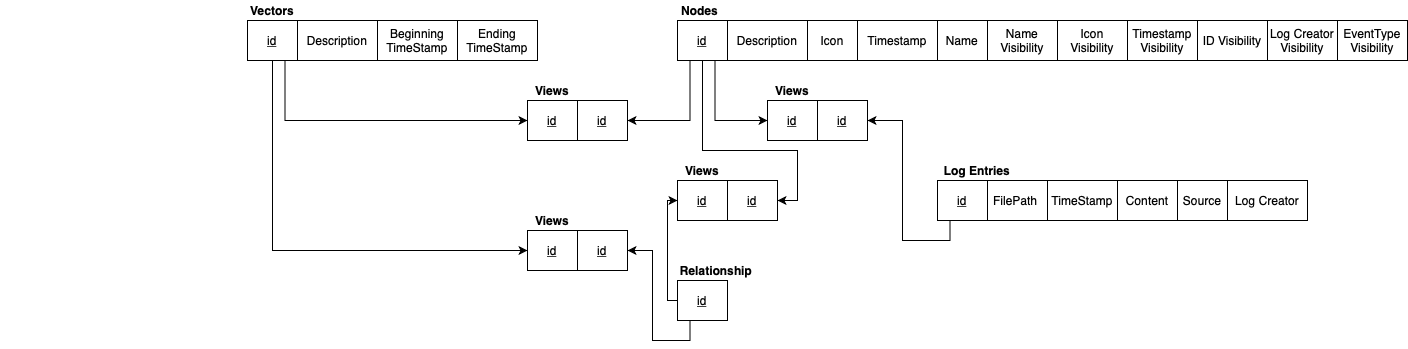
transcribeText(String filePath, fileDirectory)

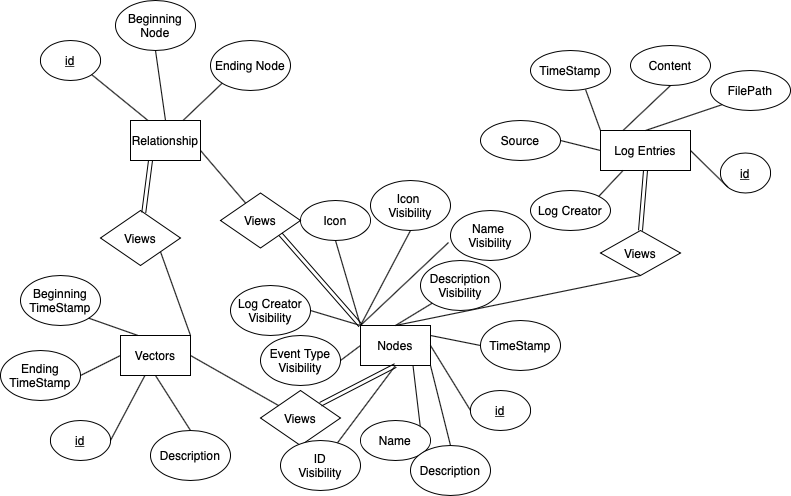
**pre:** filePath must lead to a valid image file

**post:** creates a file in fileDirectory containing the text version of the image file.

# Database

## Database Schema





$