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

**Software Design Document**

Version 1.0

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 1.0 |
| Indicator of Last Page in Document: | $ |
| Date of Last Review: | March 4, 2020 |
| Date of Next Review: | March 14, 2020 |
| Target Date for Next Update: | March 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 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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: User Interface 6
4. **Detailed Description Graphing Subsystem 13**
   1. Component Description 13
   2. Class Description: Graph 13
   3. Class Description: Vector 15
   4. Class Description: Nodes 16
   5. Class Description: Icon 17
   6. Class Description: Connector 18
   7. Class Description: Graphviz Interface 19
5. **Detailed Description File Storage Subsystem 20**
   1. Component Description 20
   2. Class Description: Splunk Interface 20
   3. Class Description: Vector DB Interface 20
6. **Detailed Description Log Ingestion Subsystem 22**
   1. Component Description 22
   2. Class Description: Log File 22
   3. Class Description: Log Entry 22
   4. Class Description: Log Cleanser 23
   5. Class Description: Log Validator 23
   6. Class Description: Log Ingestor 24
   7. Class Description: Enforcement Action Report 24
   8. Class Description: Event Configuration 24
   9. Class Description: OCR Interface 26
   10. Class Description: Transcription Interface 26
7. **Database Description 27**
   1. Data Schema 27

# 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 convert 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 |

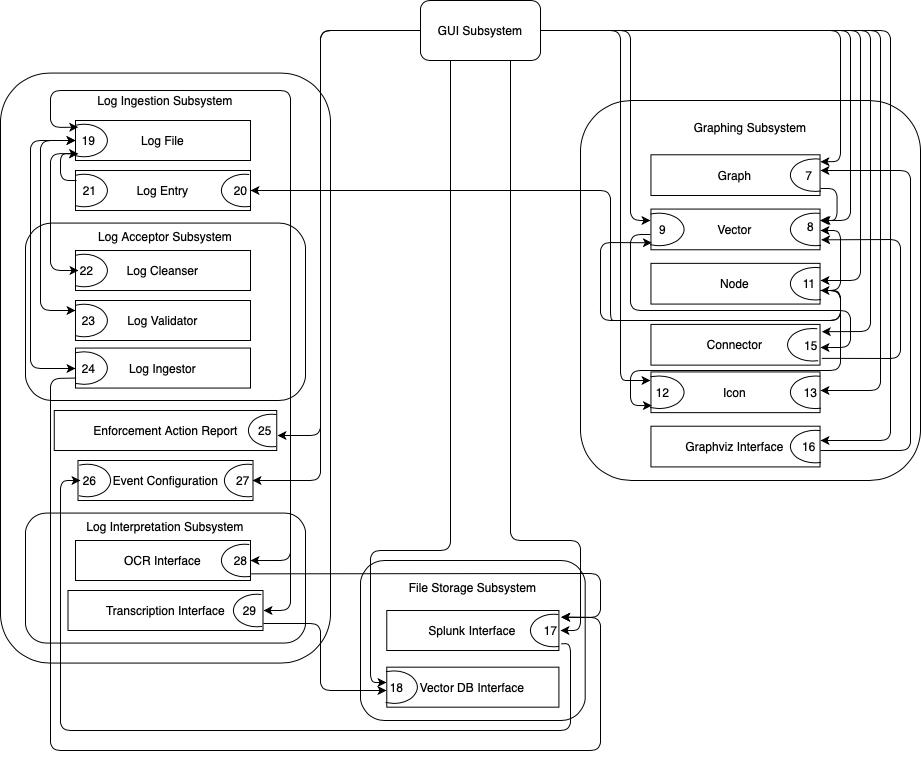
## Overview

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 class of the subsystem is:

* User Interface

The contracts of the subsystem include:

* Graph Interaction
* Vector Interaction
* Node Interaction
* Icon Interaction
* Event Creation
* Data Storage Interaction

### 2.2.2. Graphing Subsystem

The classes of the subsystem include:

* Graph
* Vector
* Node
* Icon
* Connector
* Graphviz Interface

The contracts of the subsystem include:

* Graph Settings
* Know Vector Components
* Change Vector Components
* Generate CSV
* Know Node Details
* Change Node Details
* Know Icon Components
* Know Connector Components
* Change Connector Components
* Implement Graphviz

### 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
  + Log Ingestor
* Enforcement Action Report
* Event Configuration
* Interpretation Subsystem
  + OCR Interface
  + Transcription Interface

## 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

## Class Description: User Interface

|  |  |
| --- | --- |
| **Class:** User Interface | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** | |
| **Contract 1:** Graph Interaction | |
| **Responsibilities** | **Collaborations** |
| 1. Prompt to show/hide node 2. Prompt to change icon 3. Prompt to show/hide node name 4. Prompt to show/hide node id 5. Prompt to show/hide node description 6. Prompt to show/hide node timestamp 7. Prompt to show/hide log entry reference 8. Prompt to show/hide log creator 9. Prompt to show/hide event type 10. Prompt to show/hide source 11. Prompt to change orientation 12. Prompt to change interval units 13. Prompt to change interval 14. Display attack graph 15. Display timeline 16. Display table 17. Export CSV of the graph 18. Export PDF of the graph | 1. Graph (7) Graphviz Interface (16) Icon (12) 2. Graph (7) Graphviz Interface (16) Icon (12) 3. Graph (7) Graphviz Interface (16) Icon (12) 4. Vector (8, 30) 5. Graph (7) Graphviz Interface (16) Icon (12) |
| **Contract 2:** Vector Interaction | |
| **Responsibilities** | **Collaborations** |
| 1. Prompt to delete node from vector 2. Prompt to filter through log entries 3. Prompt to search through log entries 4. Prompt to change vector name 5. Prompt to change vector description 6. Prompt to change vector time range 7. Prompt to delete vector 8. Prompt to change connector name 9. Prompt to change connector parent node 10. Prompt to change connector child node 11. Prompt to add connector 12. Prompt to delete connector | 1. Vector (9) 2. Splunk Interface (17) 3. Splunk Interface (17) 4. Vector (9) 5. Vector (9) 6. Vector (9) 7. Vector (9) 8. Connector (15) 9. Vector (9) Connector (15) 10. Vector (9) Connector (15) 11. Vector (9) Connector (15) 12. Vector (9) Connector (15) |
| **Contract 3:** Node Interaction | |
| **Responsibilities** | **Collaborations** |
| 1. Prompt user to create node from log file 2. Prompt user to create blank node 3. Prompt user to change node name 4. Prompt user to change node description 5. Prompt user to change node timestamp 6. Prompt user to change node source 7. Prompt user to delete node | 1. Node (11) Log Entry (20) 2. Node (11) 3. Node (11) 4. Node (11) 5. Node (11) 6. Node (11) 7. Vector (8) Node (11) |
| **Contract 4:** Icon Interaction | |
| **Responsibilities** | **Collaborations** |
| 1. Prompt user to create icon 2. Prompt user to delete icon 3. Prompt user to change icon name 4. Prompt user to change icon path | 1. Icon (13) 2. Icon (13) 3. Icon (13) 4. Icon (13) |
| **Contract 5:** Event Creation | |
| **Responsibilities** | **Collaborations** |
| 1. Prompt user to name event 2. Prompt user to add event description 3. Prompt user to select time range 4. Prompt user to select root directory 5. Prompt user to select blue team folder 6. Prompt user to select red team folder 7. Prompt user to select white team folder | 1. Event Configuration (27) 2. Event Configuration (27) 3. Event Configuration (27) 4. Event Configuration (27) 5. Event Configuration (27) 6. Event Configuration (27) 7. Event Configuration (27) |
| **Contract 6:** Data Storage Interaction | |
| **Responsibilities** | **Collaborations** |
| 1. Push changes to vector database 2. Pull changes from vector database 3. Approve changes to vector database | 1. Vector DB Interface (18) 2. Vector DB Interface (18) 3. Vector DB Interface (18) |

turnNodeVisibilityOn()

**pre:** nodeVisibility must be off

**post:** displays the node and applicable information (if turned on)

turnNodeVisibilityOff()

**pre:** nodeVisibility must be on

**post:** hides the entire node

changeIcon(String iconName)

**pre:** iconName must be a valid icon name

**post:** displays the changed node icon

turnNodeNameVisibilityOn()

**pre:** nodeNameVisibility must be off

**post:** displays the node with name

turnNodeNameVisibilityOff()

**pre:** nodeNameVisibility must be on

**post:** displays the node without name

turnNodeIDVisibilityOn()

**pre:** nodeNameVisibility must be off

**post:** displays the node with node id

turnNodeIDVisibilityOff()

**pre:** nodeIDVisibility must be on

**post:** displays the node without node id

turnNodeDescriptionVisibilityOn()

**pre:** nodeDescriptionVisibility must be off

**post:** displays the node with description

turnNodeDescriptionVisibilityOff()

**pre:** nodeDescriptionVisibility must be on

**post:** displays the node without description

turnNodeTimestampVisibilityOn()

**pre:** nodeTimestampVisibility must be off

**post:** displays the node with timestamp

turnNodeDTimestampVisibilityOff()

**pre:** nodeTimestampVisibility must be on

**post:** displays the node without timestamp

turnNodeLogEntryVisibilityOn()

**pre:** nodeLogEntryVisibility must be off

**post:** displays the node with log entry

turnNodeLogEntryVisibilityOff()

**pre:** nodeLogEntryVisibility must be on

**post:** displays the nod without log entry

turnNodeLogCreatorVisibilityOn()

**pre:** nodeLogCreatorVisibility must be off

**post:** displays the node with log creator

turnNodeLogCreatorVisibilityOff()

**pre:** nodeLogCreatorVisibility must be on

**post:**displays the node without log creator

turnNodeEventTypeVisibilityOn()

**pre:** nodeEventTypeVisibility must be off

**post:** displays the node with event type

turnNodeEventTypeVisibilityOff()

**pre:** nodeEventTypeVisibility must be on

**post:** displays the node without event type

turnNodeSourceVisibilityOn()

**pre:** nodeSourceVisibility must be off

**post:** displays the node with source

turnNodeSourceVisibilityOff()

**pre:** nodeSourceVisibility must be on

**post:** displays the node without source

turnLandscapeOrientation()

**pre:** orientation must be portrait

**post:** displays the change in orientation

turnPortraitOrientation()

**pre:** orientation must be landscape

**post:** displays the change in orientation

changeIntervalUnits(String unit)

**pre:** none

**post:** displays error if unit is not a valid unit; otherwise, displays the change in interval units

changeIntervals(int interval)

**pre:** none

**post:** requests new entry if interval is not positive; otherwise, displays the change in intervals

dispayAttackGraph(Graph graph)

**pre:** none

**post:** displays error if graph is invalid; otherwise, displays the graph that is returned

displayTimeline(Graph graph)

**pre:** none

**post:** displays error if graph is invalid; otherwise, displays the graph that is returned

displayTable(Graph graph)

**pre:** none

**post:** displays error if graph is invalid; otherwise, displays the graph that is returned

exportCSV(Vector vector)

**pre:** none

**post:** displays error if vector is invalid; otherwise, exports the String returned

exportPDF(Graph graph)

**pre:** none

**post:**  error is displayed if graph is invalid; otherwise, exports the pdf file

deleteNode(Vector vector, Node node)

**pre:** node must be valid

**post:** the node is deleted, vector no longer references it and removes all connectors with it as a parent or child

filterLogEntries(Timestamp start, Timestamp stop)

**pre:** start must be before stop

**post:** the filtered results are displayed

filterLogEntries(String team)

**pre:** none

**post:** error is displayed if team is not “red”, “blue”, or “white”; otherwise, the filtered results are displayed

searchLogEntries(String keyword)

**pre:** none

**post:** the results fitting the description are displayed

changeVectorName(Vector vector, String name)

**pre:** none

**post:** error is displayed if vector is invalid; otherwise, the changed vector name is displayed

changeVectorDescription(Vector vector, String description)

**pre:** none

**post:** error is displayed if vector is invalid; otherwise, the vector description is changed

changeVectorTimeRange(Vector vector, Timestamp start, Timestamp end)

**pre:** none

**post:** error is displayed if vector, start and end are not valid and start is not before end; otherwise, the vector time range is changed to be between start and end

deleteVector(Vector vector)

**pre:** none

**post:** error is displayed if vector is invalid; otherwise, vector is deleted

changeConnectorName(Connector connector, Name name)

**pre:** none

**post:** error displayed if connector is not valid; otherwise, connector name is changed to name

changeConnectorParent(Vector vector, Connector connector, Node newParent)

**pre:** none

**post:** error is displayed if either vector, connector or newParent are invalid or if connector or newParent are not within vector; otherwise, connector’s new parent node is newParent

changeConnectorChild(Vector vector, Connector connector, Node newChild)

**pre:** none

**post:**  error is displayed if either vector, connector, or newChild are invalid or connector or newChild are not within vector; otherwise, connector’s new child node is newChild

addConnector(Vector vector, String name, Node parent, Node child)

**pre:** none

**post:** error is displayed if either vector, parent, or child are invalid or if either parent or child do not exist within vector; otherwise, a new connector of called name will connect parent and child within vector

deleteConnection(Vector vector, Connector connector)

**pre:** none

**post:** error is displayed if either connector or vector are invalid or connector is not within vector; otherwise, connector is deleted and is no longer referenced by vector

createNode(String node)

**pre:** none

**post:** node called name and id as next number in sequence is created, all other fields are left blank; error is displayed if node does not exist

createNode(String name, LogFile file)

**pre:** none

**post:** node called name and id as next number in sequence is created, all other fields are decided according to the log file; error is displayed if log file does not exist

changeNodeName(Node node, String name)

**pre:** none

**post:** error is displayed if node does not exist; otherwise, node’s name is changed to name; nothing else is changed about node

changeNodeDescription(Node node, String description)

**pre:** none

**post:** error is displayed if node does not exist; otherwise, node’s description is changed to description; nothing else is changed about node

changeNodeTimestamp(Node node, Timestamp timestamp)

**pre:** none

**post:** error is displayed if node does not exist; otherwise, node’s timestamp is changed to timestamp; nothing else is changed about node

changeNodeSource(Node node, String source)

**pre:** none

**post:** error is displayed if node does not exist; otherwise, node’s source is changed to source; nothing else is changed about node

deleteNode(Node node)

**pre:** none

**post:** error is displayed if node does not exist; otherwise, node is deleted; it is removed from the vector containing it

createIcon(String name, String filePath)

**pre:** none

**post:** error is displayed if filePath does not lead to a valid image file; otherwise, icon called name with path filePath is created

deleteIcon(Icon icon)

**pre:** none

**post:** error is displayed if icon does not exist; otherwise, icon is deleted; nothing else changes

changeIconName(Icon icon, String newName)

**pre:** none

**post:** error is displayed if icon does not exist; otherwise, icon’s name is changed to newName

changeIconPath(Icon icon, String newPath)

**pre:** none

**post:** error is displayed if icon does not exist or filePath is not the path to a valid image file; otherwise, icon’s name is changed to newName

nameEvent(Event event, String name)

**pre:** none

**post:** error is displayed if event is invalid; otherwise, event name is set to name

eventDescription(Event event, String description)

**pre:** none

**post:** error is displayed if event is invalid; otherwise, event description is set to description

eventTimeRange(Event event, Timestamp start, Timestamp end)

**pre:** none

**post:** error is displayed if event, start, or end is invalid or start is not before end; otetherse, time range is set between start and finish

eventRootDirectory(Event event, String path)

**pre:** none

**post:** error is displayed if event or path is invalid; otherwise, root directory is set to path

eventBlueFolder(Event event, String path)

**pre:** none

**post:** error is displayed if event or path is invalid; otherwise, blue folder is set to path

eventWhiteFolder(Event event, String path)

**pre:** none

**post:** error is displayed if event or path is invalid; otherwise, white folder is set to path

eventRedFolder(Event event, String path)

**pre:** none

**post:** error is displayed if event or path is invalid; otherwise, red folder is set to path

pushChanges(VectorDBInterface database)

**pre:** none

**post:** error is displayed if database cannot be reached; otherwise, changes are pushed for approval

approveChanges(VectorDBInterface database)

**pre:** none

**post:** error is displayed if database cannot be reached or if user is not a lead; otherwise changes are approved

pullChanges(VectorDBInterface database)

**pre:** none

**post:** error is displayed if database cannot be reached; otherwise the system shall reflect changes on the main database

# Detailed Description Graphing Subsystem

## 4.1. Component Description

**Component Name:** Graphing Subsystem

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

**Classes:** Graph, Graphviz Interface, Vector, Node, Connectors, Icon

## 4.2. Class Description: Graph

|  |  |
| --- | --- |
| **Class:** Graph | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** | |
| **Contract 7:** Graph Settings | |
| **Responsibilities** | **Collaborations** |
| 1. Know related vector 2. Know node visibility 3. Know name visibility 4. Know id visibility 5. Know description visibility 6. Know node timestamp 7. Know orientation 8. Know interval units 9. Know interval 10. Know log entry visibility 11. Know log creator visibility 12. Know event type visibility 13. Know icon type visibility 14. Know source visibility 15. Change node visibility 16. Change name for nodes 17. Change id visibility 18. Change description visibility 19. Change node timestamp 20. Change orientation 21. Change interval units 22. Change interval 23. Change log entry visibility 24. Change log creator visibility 25. Change event type visibility 26. Change icon type visibility 27. Change source visibility | 1. Vector (8) |

changeVisibility(boolean switch)

**pre:** none

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

changeName(String name)

**pre:** none

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

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

changeOrientation(String orientation)

**pre:** orientation must be “Portrait” or “Landscape”

**post:** the orientation becomes/stays in landscape if orientation is landscape, the orientation becomes/stays in portrait if orientation is portrait

changeIntervalUnits(String units)

**pre:** units must be a valid unit

**post:** interval units are changed

changeIntervals(int intervals)

**pre:** interval must be a positive number

**post:** interval size is changed

## 4.3. Class Description: Vector

|  |  |
| --- | --- |
| **Class:** Vector | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** | |
| **Contract 8:** 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 connectors belonging to vector |  |
| **Contract 9:** Change Vector Components | |
| **Responsibilities** | **Collaborations** |
| 1. Change vector name 2. Change vector time range 3. Change vector description 4. Delete nodes 5. Add connectors 6. Delete connectors | 1. Node (11) 2. Connectors (15) 3. Connectors (15) |
| **Contract 30:** Generate CSV | |
| **Responsibilities** | **Collaborations** |
| 1. Generate CSV |  |

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

addConnector(String name, Node parent, Node child)

**pre:** parent, and child must all be valid; parent and child are both within vector

**post:** a new connector of called name will connect parent and child within vector

deleteConnection(Connector connector)

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

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

generateCSV() returns String CSV

**pre:** none

**post:** a CSV String is returned

## 4.4. 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 10:** Know Node Details | |
| **Responsibilities** | **Collaborations** |
| 1. Knows node name 2. Knows node description 3. Knows node timestamp 4. Knows related log file |  |
| **Contract 11:** Change Node Details | |
| **Responsibilities** | **Collaborations** |
| 1. Create node 2. Change icons for nodes 3. Change name for nodes 4. Change node timestamp 5. Delete node | 1. Log Entry (20) 2. Icon (12) 3. Vector (8) Vector (9) |

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

## 4.5. Class Description: Icon

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

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

## 4.6. Class Description: Connector

|  |  |
| --- | --- |
| **Class:** Connector | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** | |
| **Contract 14:** Know Connector Components | |
| **Responsibilities** | **Collaborations** |
| 1. Know connection name 2. Know parent node 3. Know child node |  |
| **Contract 15:** Change Connector Components | |
| **Responsibilities** | **Collaborations** |
| 1. Create connection 2. Change connection name 3. Change parent node 4. Change child node 5. Delete connection | 1. Vector (8) 2. Vector (8) 3. Vector (8) 4. Vector (8) |

createConnector(Vector vector, String name, Node parent, Node child)

**pre:** none

**post:** connector called name will connect parent and child within vector

changeConnectorName(Connector connector, Name name)

**pre:** none

**post:** connector name is changed to name

changeConnectorParent(Vector vector, Connector connector, Node newParent)

**pre:** connector and newParent must both be in vector

**post:** connector’s new parent node is newParent

changeConnectorChild(Vector vector, Connector connector, Node newChild)

**pre:** connector and newChild must both be in vector

**post:** connector’s new child node is newChild

deleteConnection(Vector vector, Connector connector)

**pre:** none

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

## 4.7. Class Description: Graphviz Interface

|  |  |
| --- | --- |
| **Class:** Graphviz Interface | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** | |
| **Contract 16:** Implement Graphviz | |
| **Responsibilities** | **Collaborations** |
| 1. Implement Graphviz 2. Export PDF | 1. Graph (7) 2. Graph (7) |

implementGraphviz(Graph graph, String type) returns int[][] Image

**pre:** graph must be valid, type must be either “Attack Graph”, “Timeline”, or “Table”

**post:** returns Image represents the vector with the conditions stored in graph in the format specified by type

exportPDF(Graph graph) returns a pdfImage

**pre:** graph must be valid

**post:** returns Image represents the vector with the conditions stored in graph

# Detailed Description File Storage Subsystem

## 5.1. Component Description

**Component name:** File Storage Subsystem

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

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

## 5.2. Class Description: Splunk Interface

|  |  |
| --- | --- |
| **Class:** Splunk Interface | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** | |
| **Contract 17:** Implement Splunk | |
| **Responsibilities** | **Collaborations** |
| 1. Pull log files from Splunk 2. Filter using Splunk | 1. Event Configuration (26) |

pullFiles()

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

**post:** Splunk files are now within the root directory

filterLogEntries(Timestamp start, Timestamp end) returns LogEntry[] list

**pre:** none

**post:** list contains the log entries in splunk reduced to those fitting within the timestamps

filterLogEntries(String team) returns LogEntry[] list

**pre:** none

**post:** list contains the log entries in splunk reduced to those originating from team

## 5.3. Class Description: Vector DB Interface

|  |  |
| --- | --- |
| **Class:** Vector DB Interface | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** | |
| **Contract 18:** DB Interaction | |
| **Responsibilities** | **Collaborations** |
| 1. Pull updates to vectors and components from DB 2. Push updates to vectors and components from DB 3. Approve updates to vectors and components from DB |  |

pushChanges(VectorDBInterface database)

**pre:** database must be accessible

**post:** changes are pushed for approval

approveChanges(VectorDBInterface 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 3. Know cleansing status 4. Know validation status 5. Know ingestion status | 1. Splunk Interface (17) OCR Interface (28) Transcription Interface (29) 2. Log Cleanser (22) 3. Log Validator (23) 4. Log Ingestor (24) |

## 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. Divide log file | 1. Log File (19) |

divideLogFil(LogFile file)

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

**post:** one or many log entries is made from the log file

## 6.4. Class Description: Log Cleanser

|  |  |
| --- | --- |
| **Class:** Log Cleanser | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** | |
| **Contract 22:** Cleanse Logs | |
| **Responsibilities** | **Collaborations** |
| 1. Remove empty rows and columns 2. Change cleansed status 3. Know cleansed status |  |

cleanseLog(LogFile file)

**pre:** file must be a valid log file, cleansed status must be false

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

## 6.5. Class Description: Log 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) |

validateLog(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

## 6.6. Class Description: Log Ingestor

|  |  |
| --- | --- |
| **Class:** Log Ingestor | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** | |
| **Contract 24:** Ingest Logs | |
| **Responsibilities** | **Collaborations** |
| 1. Take files into the system 2. Change ingested status 3. Know ingested status | 1. Log File (19) Splunk Interface (17) 2. Log File (19) 3. Log File (19) |

ingestLog(LogFile file)

**pre:** file must be a valid log file that has been cleansed and validated but not ingested

**post:** file is ingested by copying it within the system, ingested status becomes true

## 6.7. Class Description: Enforcement Action Report

|  |  |
| --- | --- |
| **Class:** Enforcement Action Report | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** | |
| **Contract 25:** Know Failed Logs | |
| **Responsibilities** | **Collaborations** |
| 1. Know logs that do pass the validation 2. Know logs that do not pass the validation | 1. Log Validator (23) 2. Log Validator (23) |

## 6.8. Class Description: Event Configuration

|  |  |
| --- | --- |
| **Class:** Event Configuration | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** | |
| **Contract 26:** Know Event Attributes | |
| **Responsibilities** | **Collaborations** |
| 1. Know event name 2. Know event description 3. Know event time range 4. Know root directory 5. Know red team directory 6. Know blue team directory 7. Know white team directory |  |
| **Contract 27:** Change Event Description | |
| **Responsibilities** | **Collaborations** |
| 1. Change event name 2. Change event description 3. Change event time range 4. Change root directory 5. Change red team directory 6. Change blue team directory 7. Change white team directory |  |

nameEvent(String name)

**pre:** none

**post:** event name is set to name

eventDescription(String description)

**pre:** none

**post:** event description is set to description

eventTimeRange(Timestamp start, Timestamp end)

**pre:** start and end must be valid timestamps and start must be before end

**post:** time range is set between start and finish

eventRootDirectory(String path)

**pre:** path must be valid

**post:** root directory is set to path

eventBlueFolder(String path)

**pre:** path must be valid

**post:** blue folder is set to path

eventWhiteFolder(String path)

**pre:** path must be valid

**post:** white folder is set to path

eventRedFolder(String path)

**pre:** path must be valid

**post:** red folder is set to path

## 6.9. Class Description: OCR Interface

|  |  |
| --- | --- |
| **Class:** OCR Interface | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** | |
| **Contract 28:** Convert Visual Logs to Text Logs | |
| **Responsibilities** | **Collaborations** |
| 1. Convert visual logs to text logs | 1. Splunk Interface (17) Log File (19) |

visualToText(ImageFile image) returns LogFile file

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

**post:** returns file which contains the content of image in a text format

## 6.10. Class Description: Transcription Interface

|  |  |
| --- | --- |
| **Class:** Transcription Interface | |
| **Superclass:** | |
| **Subclasses:** | |
| **Private Responsibilities:** | |
| **Contract 29:** Convert Audio Logs to Text Logs | |
| **Responsibilities** | **Collaborations** |
| 1. Convert audio logs to text logs | 1. Splunk Interface (17) Log File (19) |

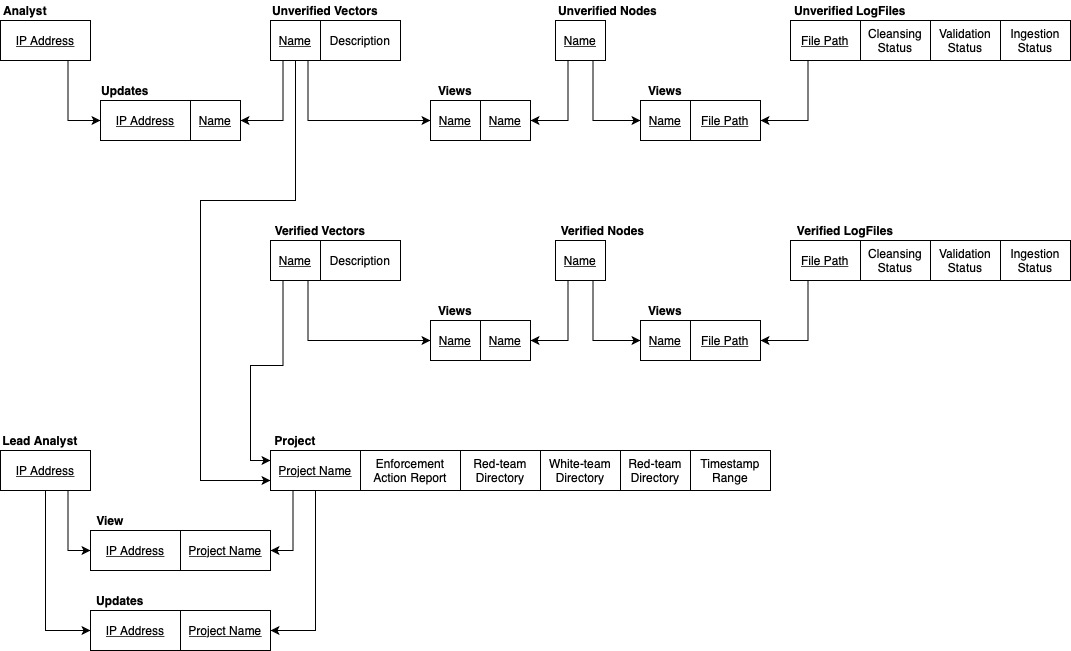
audioToText(AudioFile audio) returns LogFile file

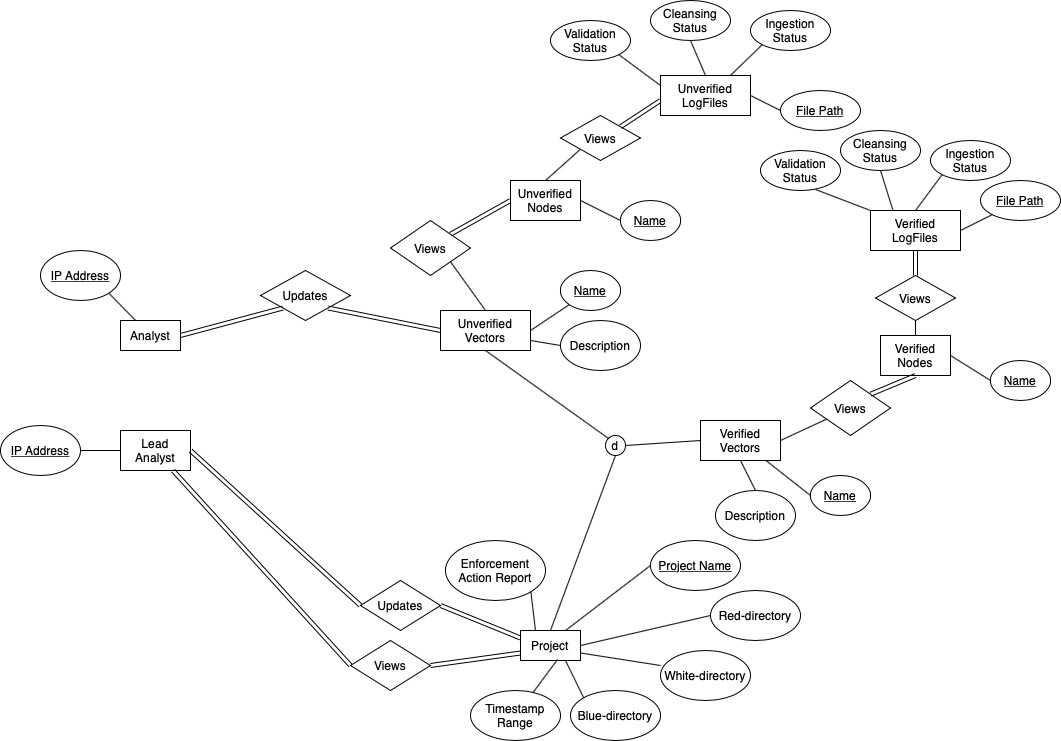
**pre:** audio must be a valid audio or video file

**post:** returns file which contains the content of audio in a text format

# Database

## Database Schema





$