­Prevent, Mitigate, and Recover (PMR) Insight

Collective Knowledge System (PICK)

Software Design Document

Version 1.7

3/29/2020

Document Control

Approval

The Guidance Team and the customer shall approve this document.

Document Change Control

|  |  |
| --- | --- |
| Initial Release: | 0.1 |
| Current Release: | 1.0 |
| Indicator of Last Page in Document: | % |
| Date of Last Review: | 3/29/20 |
| Date of Next Review: | 3/30/20 |
| Target Date for Next Update: | 3/30/20 |

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. Gates

Dr. Salamah

Dr. Roach

Steven Roach

Jake Lasley

Customer:

Dr. Oscar Perez

Vincent Fonseca

Herandy Denisse Vazquez

Baltazar Santaella

Florencia Larsen

Erick De Nava

Software Team Members:

Abel Rodriguez

Elizabeth Barragan

Jose A. Leon Cordero

Yamel E. Hernandez

Manuel Delgado

Change Summary

The following table details changes made between versions of this document

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Modifier | Description |
| 1.0 |  |  | Creation of Document |
| 1.1 | 03-06-2020 | Manuel | Completed Document Control Details and Section 1.2 |
| 1.2 | 3/7/20 | Yamel | Sec 1.5 & Sec 3 Intro |
| 1.3 | 3/9/2020 | Antoine | Added 3.1 |
| 1.3 | 3/9/20 | Yamel | Filled in Sec 1.4, Sec 2 Ingestion Subsystem, & Sec 3 Ingestion Subsystem |
| 1.3 | 03-09-2020 | Manuel | Added section 3.4, intro to section 2, section 2.3 |
| 1.3 | 3/9/20 | Elizabeth | Added full Sec. 4, Sec 3.5 |
| 1.4 | 3/29/20 | Antione, Manuel & Yamel | Filled out sections |
| 1.5 | 3/30/2020 | Elizabeth, Antoine, Yamel & Manuel | Fixing up comments, filling out contracts, filling out subsystem descriptions, adding diagrams |
| 1.5 | 3/30/2020 | Elizabeth | Completed Sec.4 |

Table of Contents

[Document Control ii](#_Toc36490054)

[Approval ii](#_Toc36490055)

[Document Change Control ii](#_Toc36490056)

[Distribution List ii](#_Toc36490057)

[Change Summary ii](#_Toc36490058)

[1. Introduction 1](#_Toc36490059)

[1.1. Purpose and Intended Audience 1](#_Toc36490060)

[1.2. Scope of Product 1](#_Toc36490061)

[1.3. References 1](#_Toc36490062)

[1.4. Definitions, Acronyms, and Abbreviations 1](#_Toc36490063)

[1.4.1. Definitions 1](#_Toc36490064)

[1.4.2. Acronyms 2](#_Toc36490065)

[1.4.3. Abbreviations 2](#_Toc36490066)

[1.5. Overview 2](#_Toc36490067)

[2. Decomposition Description 3](#_Toc36490068)

[2.1. System Collaboration Diagram 3](#_Toc36490069)

[2.2. Subsystem and Component Descriptions 3](#_Toc36490070)

[2.2.1. GUI Subsystem 3](#_Toc36490071)

[2.2.2. Ingestion Subsystem 3](#_Toc36490072)

[2.2.3. Database Subsystem 4](#_Toc36490073)

[2.2.4. Version Control Subsystem 4](#_Toc36490074)

[2.2.5. Vector Subsystem 4](#_Toc36490075)

[2.2.6. Networking Component 4](#_Toc36490076)

[2.2.7. Event Configuration Component 4](#_Toc36490077)

[2.2.8. Signal Controller Component 4](#_Toc36490078)

[2.2.9. Log File Component 4](#_Toc36490079)

[2.3. Dependencies 5](#_Toc36490080)

[3. Detailed Description of Components 6](#_Toc36490081)

[3.1. Graphical User Interface Subsystem 6](#_Toc36490082)

[3.1.1. GUI Façade 6](#_Toc36490083)

[3.1.2. GUI Main Window 7](#_Toc36490084)

[3.1.3. GUI Vector Frame 7](#_Toc36490085)

[3.2. Ingestion Subsystem 9](#_Toc36490086)

[3.2.1. Ingestion 9](#_Toc36490087)

[3.2.2. Error 9](#_Toc36490088)

[3.2.3. Enforcement Action Report 10](#_Toc36490089)

[3.2.4. OCR Façade 10](#_Toc36490090)

[3.2.5. Audio Transcriber Façade 11](#_Toc36490091)

[3.2.6. Splunk Façade 12](#_Toc36490092)

[3.2.7. Cleanser 12](#_Toc36490093)

[3.3. Database Subsystem 13](#_Toc36490094)

[3.3.1. DB Façade 13](#_Toc36490095)

[3.4. Version Control subsystem 15](#_Toc36490096)

[3.4.1. Version Control Façade 15](#_Toc36490097)

[3.4.2. Network History 16](#_Toc36490098)

[3.4.3. History 17](#_Toc36490099)

[3.5. Vector Subsystem 18](#_Toc36490100)

[3.6. Networking Component 18](#_Toc36490101)

[3.7. Event Configuration Component 21](#_Toc36490102)

[3.8. Signal Controller Component 24](#_Toc36490103)

[3.9. Log File Component 25](#_Toc36490104)

[4. Database 28](#_Toc36490105)

[4.1. Data Model Design 28](#_Toc36490106)

[4.1.1. Data Model Design 28](#_Toc36490107)

[% 29](#_Toc36490108)

# Introduction

## Purpose and Intended Audience

The purpose of this document is to inform future users about, collaborations, and protocols of each of the PICK system’s subsystems and components for future changes to the system. The document will go over the descriptions of each subsystem and component, along with the classes’ descriptions associated with them. The collaborations will depict and explain which set of classes and contracts, what data is sent to each class through the contracts, and the result of the collaboration. Lastly, protocols will be an in-depth explanation of each method call in the system that will allow the maintainer to understand how to call a method, the inside workings of the that method, and the expected result outcome of the method.

## Scope of Product

The Lethality, Survivability, and HSI Directorate (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.

The University of Texas at El Paso (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] 08c\_SDD\_Outline.pdf

[2] 09Subsystems.pdf

[3]“Entity–relationshipmodel,”Wikipedia,09-Mar-2020.[Online].Available: https://en.wikipedia.org/wiki/Entity–relationship\_model. [Accessed: 10-Mar-2020].

[4] SDD\_detailed\_component\_table\_template.docx

## 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. |

### Acronyms

|  |  |
| --- | --- |
| **PICK** | Prevent, Mitigate, and Recover (PMR) Insight Collective Knowledge System |
| **UTEP** | University of Texas at El Paso |
| **LSH** | Lethality, Survivability, and HSI Directorate |
| **PMR** | Prevent, Mitigate, and Recover |
| **UI** | User Interface |
| **GUI** | Graphical User Interface |
| **OCR** | Optical Character Reader |

### Abbreviations

|  |  |
| --- | --- |
|  |  |
|  |  |

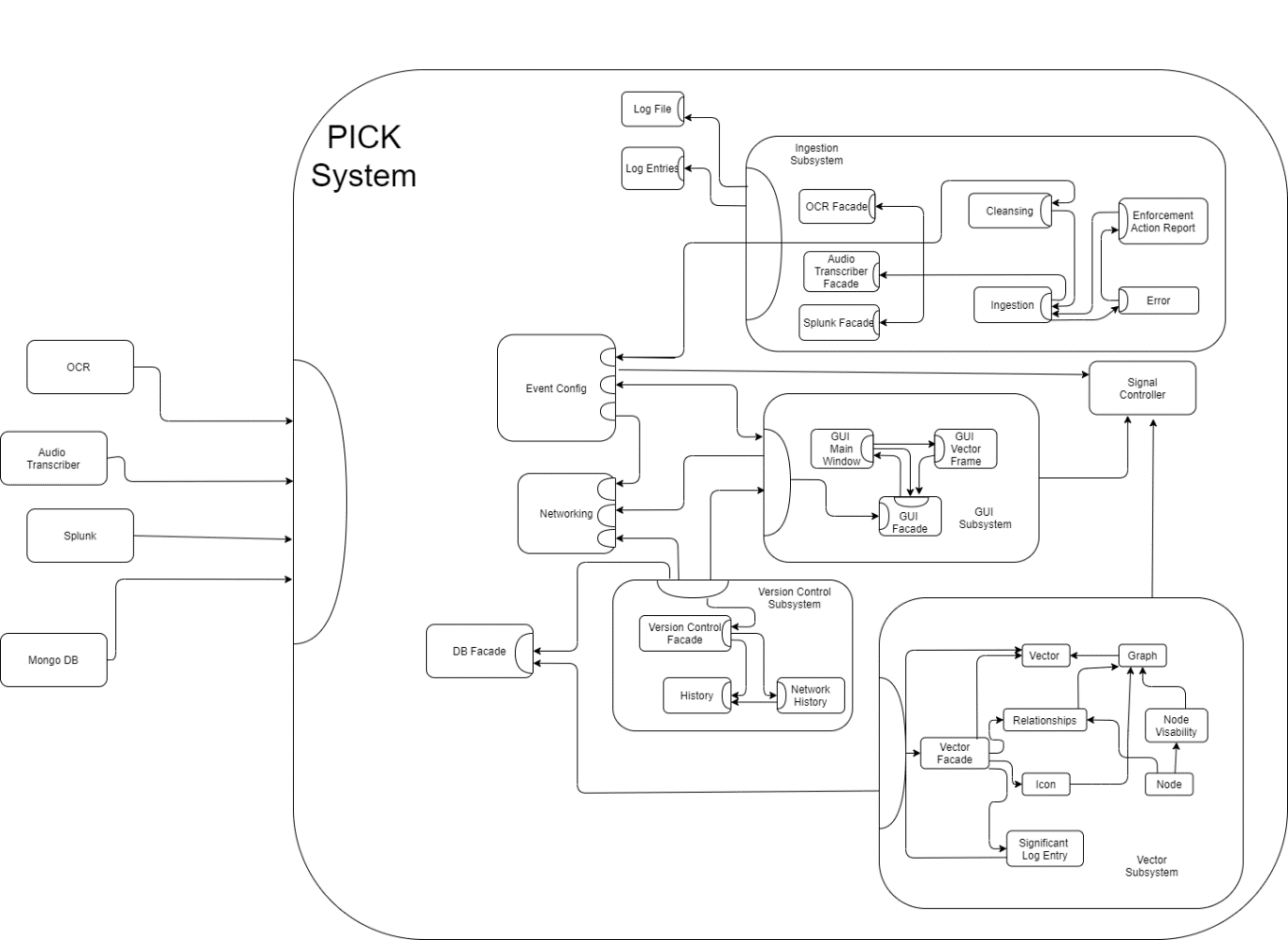
## Overview

The document will encompass a detail description of our overall system. Section 2 will discuss how we decomposed our system into subsystems and components, it will describe their purpose and who they collaborate with. Section 3 will discuss the details of the components, how they fit into the subsystems, their design, as well as their classes, contracts and protocols. Lastly section 4 will discuss how a database will be incorporated in the system and how it will be used by the system from its schema.

# Decomposition Description

The overall descriptions for each component will describe how within the system, subsystems interact with each other. It will abstractly and concisely inform the reader what functions the specific component oversees displaying.

## System Collaboration Diagram



## Subsystem and Component Descriptions

This section demonstrates the subsystem along with its contracts and protocols at a high level of abstraction, for further details each section contains a reference to its corresponding subsystem with more details regarding its contracts and protocols in section 3.

### GUI Subsystem

This subsystem will be further discussed in 3.1, it will include their protocols and responsibilities.

|  |
| --- |
| **Subsystem:** GUI Subsystem |
| **Classes:** GUI Façade, GUI Main Window, & GUI Vector Frame |
| **Description:** Responsible for displaying system content and getting user input. |
| **Contracts:**   1. Get User Input    1. Server: GUI Façade 2. Display    1. Server: GUI Main Window |

### Ingestion Subsystem

This subsystem will be further discussed in 3.2, it will include their protocols and responsibilities.

|  |
| --- |
| **Subsystem:** Ingestion Subsystem |
| **Classes:** Ingestion, Error, Enforcement Action Report, OCR Façade, Audio Transcriber Façade, Splunk Façade & Cleanser, |
| **Description:** Responsible for ingesting log file contents into the system |
| **Contracts:**   1. Scan for Changes 2. Validate Files 3. Ingest 4. Configure Error 5. Know Relative Log File Error 6. Provide Errors 7. Provide OCR Usability 8. Provide Audio Transcriber Usability 9. Search 10. Check Delta 11. Load Log Files 12. Save Log Files |

### Database Subsystem

This subsystem will be further discussed in 3.3, it will include its protocols and responsibilities.

|  |
| --- |
| **Subsystem:** Database Subsystem |
| **Classes:** DB Façade |
| **Description:** Responsible for interacting with the DB Management system. |
| **Contracts:** |
| 1. Query to Manipulate DB |

### Version Control Subsystem

This subsystem will be further discussed in 3.4, it will include its protocols and responsibilities.

|  |
| --- |
| **Subsystem:** Version Control Subsystem |
| **Classes:** Version Control Façade, Network History, History |
| **Description:** Responsible for managing the changes in the system. |
| **Contracts:** |
| 1. Send to Store Changes 2. Manage Network Changes 3. Manage Local Changes |

### Vector Subsystem

This subsystem will be further discussed in 3.5, it will include its protocols and responsibilities.

|  |
| --- |
| **Subsystem:** Vector Subsystem |
| **Classes:** |
| **Description:** |
| **Contracts:** |
|  |

### Networking Component

This component will be further discussed in 3.6, it will include its protocols and responsibilities.

### Event Configuration Component

This component will be further discussed in 3.7, it will include its protocols and responsibilities.

### Signal Controller Component

This component will be further discussed in 3.8, it will include its protocols and responsibilities.

### Log File Component

This component will be further discussed in 3.9, it will include its protocols and responsibilities.

## Dependencies

Components for optical character reader and audio transcriber have not yet been looked at individually therefore it impacts the time it will take to complete a detailed collaboration diagram. To take advantage of all the capabilities Splunk has to offer, there needs to be a considerable amount of time spent on understanding the parsing methods and the configuration files Splunk uses internally. We depend on understanding these components in order to deliver an optimize system.

# Detailed Description of Components

The following section will describe the system’s components with their functionalities and contracts with other components.

## Graphical User Interface Subsystem

This section describes the GUI subsystem of the PICK system. The purpose of the GUI subsystem is to present the PICK system to the user and provide a way for the user to interact with the system.

### GUI Façade

|  |  |
| --- | --- |
| **Class Name**: GUI Façade | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Description:** Manage interactions between components of the subsystem and other subsystems or external components | |
| **Private Responsibilities:** None | |
| **Contract:** 1. Get User Input | |
| **Contract Description:** Distribute data from window frames to models | |
| **Contract Protocols:**   * getInput(String input) return null   + **Purpose:** Gets input from text fields and sends it to respective model   + **Pre-conditions:**     - **input:** Is not null or empty     - Instance of respective model should not be null   + **Post-conditions:**     - Model is updated with input data   + **Invariants:** None | |
| **Responsibilities** | **Collaborators** |
| * Distribute input from window frames to other subsystems or components | * GUI Main Window |
|  |  |
| **Contract:** 2. Get System Data | |
| **Contract Description:** Distribute model data to window frames | |
| **Contract Protocols:**   * getData(String type) return Set   + **Purpose:** Return set of data needed by the window frame   + **Pre-conditions:**     - **type:** Is model type requested     - **type:** Is not null or empty   + **Post-conditions:**     - Return set of all data of type requested   + **Invariants:** None * update(String signal) return null   + **Purpose:** Given a signal, updates currently visible windows and frames that display data related to the signal   + **Pre-conditions:**     - **signal:** Is not null or empty     - Window frame that displays data related to the signal is currently visible   + **Post-conditions:**      - Windows are refreshed with updated model data   + **Invariants:** None | |
| **Responsibilities** | **Collaborators** |
| * Distribute model data to window frames * Update window frames based on controller signals | * Event Configuration * Vector * Node * Graph * Significant Log Entry * Node Visibility * Relationship * Icon * Splunk Façade * Log File |

### GUI Main Window

|  |  |
| --- | --- |
| **Class Name**: GUI Main Window | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Description:** Display the main point of interaction between the user and the PICK system | |
| **Private Responsibilities:**   * Display tool bar * Display menu bar * Read input from user * Know which vector frame is active | |
| **Contract:** 3. Display | |
| **Contract Description:** | |
| **Contract Protocols:**   * display() return null   + **Purpose:** Instantiates the main window with current data   + **Pre-conditions:**      - Model data to be displayed has been instantiated   + **Post-conditions:**      - Main window is displayed   + **Invariants:** None | |
| **Responsibilities** | **Collaborators** |
| * Display table of log entries * Display vector frame of active vectors * Display number of connections established | * GUI Façade * GUI Vector Frame |

### GUI Vector Frame

|  |  |
| --- | --- |
| **Class Name**: GUI Vector Frame | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Description:** Organize vector information to be displayed | |
| **Private Responsibilities:**   * None | |
| **Contract:** 4. Get Vector Frame | |
| **Contract Description:** Arrange vector information | |
| **Contract Protocols:**   * setupVectorFrame(Object vector) return self   + **Purpose:** set up information for a vector in a frame   + **Pre-conditions:**      - **vector:** Represents a single vector     - **vector:** Is not null or empty   + **Post-conditions:**      - Returns instantiated vector frame to be placed in a window   + **Invariants:** None | |
| **Responsibilities** | **Collaborators** |
| * Display table of nodes associated to vector * Display table of significant log entries * Display graph associated to vector | * GUI Façade |

## Ingestion Subsystem

The following section will depict and describe how the Ingestion subsystem, which will have three subsystems within itself, Splunk, OCR, & Audio Transcriber. The purpose of the ingestion subsystem is to provide a method for the user to ingest files into the system.

### Ingestion

|  |  |
| --- | --- |
| **Class Name**: Ingestion | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Description:** Ingests log files into the system for manipulation | |
| **Private Responsibilities**   * Know log file format | |
| **Contract:** 5. Scan for changes | |
| **Responsibilities** | **Collaborators:** |
| * Scan for changes to log files | * Splunk Façade (5) |
| **Contract:** 6. Validate Files | |
| **Responsibilities:** | **Collaborators:** |
| * Validate log files | * Log File (3) |
| **Contract:**  7. Ingest | |
| **Contract Description:** | |
| **Contract Protocols:**   * ingest(logfile)   + **Purpose:** Ingest cleansed log files into the system, so they analyst can manipulate   + **Pre-conditions:**     - Log file directory must be validated and verified, and log files must be flagged cleansed.     - Takes in a log file object for its parameter     - Logfile: is flagged ready for splunk   + **Post-conditions:**     - Log files are converted to log entries and stored to the database   + **Invariants:** None * identifyFormat(logfile)   + **Purpose:** Flag the type of format the log file is in, so it can be sent to the proper API.   + **Pre-condition**     - Takes a log file object for its parameter     - Logfile: is unflagged for its format   + **Post-condition**     - Log files are flagged according to their format   + **Invariants**     - None | |
| **Responsibilities:** | **Collaborators:** |
| * Know validation parameters * Ingest log files | * Event Configuration (1) * OCR Façade (1), Audio Transcriber Façade (1), Splunk Façade (1-3), Cleansing (3-5) |

### Error

|  |  |
| --- | --- |
| **Class Name**: Error | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Description:** Provides information on errors that are found that prevent the files from getting ingested. | |
| **Private Responsibilities**   * Know error message * Know line number error | |
| **Contract:** 8. Configure Error | |
| **Contract Description:** | |
| **Contract Protocols:**   * configureError(logfile)   + **Purpose:** Know which line(s) the error(s) reside in and be able to identify what type of errors they are, to know which error message corresponds to it.   + **Pre-condition**     - Reference to log file     - Log file object parameter     - Logfile: flagged for having issues ingesting   + **Post-condition**     - Error line number & proper error message by the usage of a try and catch.   + **Invariants**     - None | |
| **Responsibilities** | **Collaborators:** |
|  |  |

### Enforcement Action Report

|  |  |
| --- | --- |
| **Class Name**: Enforcement Action Report | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Description:** Provides a detailed report why some files can’t be ingested | |
| **Private Responsibilities** | |
| **Contract:** 9. Know relative log file | |
| **Responsibilities**  Know log file | **Collaborators:**  Ingestion (4) |
| **Contract:**  10. Provide errors | |
| **Contract Description:** | |
| **Contract Protocols:**   * findErrors(logfile)   + **Purpose:** Give detailed information to the user of what the error(s) are and where they are.   + **Pre-condition**     - Reference to log file & error parameters (from try-catch)     - Logfile: flagged for having issues ingesting   + **Post-condition**     - Reference where the error(s) are & display the error message(s) to the user   + **Invariants**     - None | |
| **Responsibilities:** | **Collaborators:** |
| * Know error(s) of log file | * Error (1,2) |

### OCR Façade

|  |  |
| --- | --- |
| **Class Name**: OCR Facade | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Description:** Serves as an entry point to interact with the OCR API | |
| **Private Responsibilities**   * Leverages OCR API to ingest OCR file types * Receives parsed text-based files | |
| **Contract:** 11. Provide OCR Usability | |
| **Contract Description:** | |
| **Contract Protocols:**   * sendToOCR(logfile)   + **Purpose:** Serve as an interface to interact with the OCR API   + **Pre-condition**     - Log files must be flagged as type OCR (images, pdfs, etc.)     - Takes in a log file object for its parameter     - Logfile: is flagged OCR type & unparsed   + **Post-condition**     - Sends OCR log files to the OCR API & receives (parsed) text-based log files from the OCR API   + **Invariants**     - None | |
| **Responsibilities** | **Collaborators:** |
|  |  |

### Audio Transcriber Façade

|  |  |
| --- | --- |
| **Class Name**: Audio Transcriber Facade | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Description:** Serves as an entry point to interact with the Audio Transcriber API | |
| **Private Responsibilities**   * Leverages Audio Transcriber API to ingest audio file types * Receives parsed text-based files | |
| **Contract:** 12. Provide Audio Transcriber Usability | |
| **Contract Description:** | |
| **Contract Protocols:**   * sendATInfo (logfile)   + **Purpose:** Serve as an interface to interact with the Audio Transcriber API   + **Pre-conditions:**     - Log files must be flagged as type Audio (mp3, mp4, wav, etc.)     - Takes in a log file object for its parameter     - Logfile: is flagged audio type & unparsed   + **Post-condition**     - Sends audio log files to the Audio Transcriber API & receives a (parsed) text-based log files from the Audio Transcriber API   + **Invariants**     - None | |
| **Responsibilities** | **Collaborators:** |
|  |  |

### Splunk Façade

|  |  |
| --- | --- |
| **Class Name:** Splunk Façade | |
| **Superclass:** None | |
| **Subclass:** None | |
| **Description:** Class that defines all interactions between the system and Splunk | |
| **Private Responsibilities**   * Know how to interact with Splunk * Sign in * Parse files * Upload files into Splunk * Retrieve log entries | |
| **Contract:** 13.Search | |
| **Contract Description:** System will return log entries that matched in Splunk to the user’s input | |
| **Contract Protocol:**   * searchLogEntries (String input) return Dictionary   + **Purpose:** Return only log entries that match what the user Is searching for.   + **Pre-conditions:**     - Input: Is information received through the gui to use as parameters in order to use as parameters in JSON     - There is a connection established with Splunk     - There are log entries in Splunk     - Inputted parameters are existent within the log entries in Splunk   + **Post-conditions:**     - If log entries that match the input parameters are existent, are returned as a dictionary else it returns null   + **Invariants:**      - None | |
| **Responsibilities:** | **Collaborators:** |
| * Can search through log entries * Send them back | GUI Façade, DB Façade |
| **Contract:** 14.Check Delta | |
| **Contract Description:** Checking for any changes in the predefined directories | |
| **Contract Protocol:**   * checkForDelta (Event Configuration object ) return void   + **Purpose:** Splunk will check if the directories where the raw log files are had changes   + **Pre-conditions:**     - An Event Configuration object must exist with a Map of directories   + **Post-conditions:**     - Splunk processes any log files that are new or changed if any are detected else does nothing     - Log files are cleansed, validated, and processed by Splunk   + **Invariants:**      - None | |
| **Responsibilities:** | **Collaborators:** |
| * Check for deltas from predefined directories | DB Façade |

### Cleanser

|  |  |
| --- | --- |
| **Class Name:** Cleanser | |
| **Superclass:** None | |
| **Subclass:** None | |
| **Description:** Class that will interact with raw log files and cleanse them and upload them to Splunk | |
| **Private Responsibilities**   * Removes extra spacing | |
| **Contract:** 15. Load log files | |
| **Contract Description:** Need to know where the log files are and open them | |
| **Contract Protocol:**   * open (Event Configuration object) returns   + **Purpose:** Open log files read into buffer   + **Pre-conditions:**     - D Readable log files must be saved in predefined directories     - Predefined directory attributes received from Event Configuration   + **Post-conditions:**     - Contents of the log files have been read in order to be able to cleanse the log files   + **Invariants:**      - None | |
| **Responsibilities:** | **Collaborators:** |
| * Get log files from predefined directories | DB Façade |
| **Contract:** 16. Save log files | |
| **Contract Description:** Ones the log files have been cleansed connect to Splunk to upload | |
| **Contract Protocols:**   * toSplunk (String log file) returns void   + **Purpose:** Use the Splunk façade to upload cleansed log files   + **Pre-conditions:**     - A log file needs to be validated and cleansed in order to be uploaded into Splunk.   + **Post-conditions:**     - The cleansed file will be saved inside Splunk   + **Invariants:**      - None | |
| **Responsibilities:** | **Collaborators:** |
| * Save cleansed log files to Splunk | Splunk Façade |

## Database Subsystem

This section describes the Database subsystem for the PICK system. The purpose of the Database Subsystem is to save permanently significant log entries and configuration parameters submitted by each analyst.

### DB Façade

|  |  |
| --- | --- |
| **Class Name**: DB Façade | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**  Know database connection details | |
| **Contract:** 17. Query to Manipulate DB | |
| **Contract Description**: | |
| **Contract Protocols:**   * def add (object) returns None   + **Purpose:** Adds objects into the DB.   + **Pre-Condition:**     - An object represents a log file, event configuration, vector, significant log entry, graph, node, icon, node visibility, or relationship object formatted in JSONthat will be manipulated in the DB.   + **Post-Condition:**     - The object will be saved into the DB along with its content, the DB doesn’t return anything.   + **Invariants:** None * def edit (object) returns None   + **Purpose:** Edits the object in the DB.   + **Pre-Condition**      - An object represents a log file, event configuration, vector, significant log entry, graph, node, icon, node visibility, or relationship object formatted in JSONthat will be manipulated in the DB.   + **Post-Condition**     - The specified data will be deleted in the DB, the DB doesn’t return anything.   + **Invariants:** None * def delete (object) returns None   + **Purpose:** Deletes an object/object’s content from the DB.   + **Pre-Condition:**     - An object represents a log file, event configuration, vector, significant log entry, graph, node, icon, node visibility, or relationship object formatted in JSONthat will be manipulated in the DB.   + **Post-Condition:**      - The specified object/object’s content will be deleted from the DB, the DB doesn’t return anything.   + **Invariants:** None * def filter (object, filter) return filteredResult * **Purpose:** Filters object’s attributes to return filtered data to the component/subsystem that requested the data. * **Pre-Condition:**   + An object represents a log file, event configuration, vector, significant log entry, graph, node, icon, node visibility, or relationship object formatted in JSON.   + The filter given as a string, represents the object’s attribute that will be specified to retrieve certain content from the object.   + The filter should be an existing object’s attribute. * **Post-Condition:** * The specified filter will be applied to the object in the DB, the DB will return the filtered content. * **Invariants:** None * def search (object, search\_parameter) returns info   + - **Purpose:** Searches data from the DB with specified criteria to be sent back to the component/subsystem that requested it. * **Pre-Condition:**    + An object represents a log file, event configuration, vector, significant log entry, graph, node, icon, node visibility, or relationship object formatted in JSON that will be manipulated in the DB.   + The search parameter given as a string, represents the specified value that will be used to search for content in the DB. * **Post-Condition:** * The search value will be sent to the DB to search the content in the DB, the DB will return the content matching the search criteria. * **Invariants:** None * def update (object, update\_filter, update\_data) returns None * **Purpose:** Updates specific fields of an object in the DB. * **Pre-Condition:** * The object represents a log file, event configuration, vector, significant log entry, graph, node, icon, node visibility, or relationship object formatted in JSON that will be manipulated in the DB. * The update filter given as a string represents the attribute of the object that will be updated in the DB. * The update data given as a JSON object, represents the actual data that it will be updated to in the DB. * **Post-Condition:** * The data to be updated will be sent to the DB and saved, the DB will not return anything. * **Invariants:** None | |
| **Responsibilities** | **Collaborators:** |
| * Run queries to add objects to the DB * Run queries to edit objects in the DB * Run queries to delete objects in the DB * Run queries to filter objects in the DB * Run queries to search for objects in the DB * Run queries to update objects in the DB |  |

## Version Control subsystem

This section describes the Version Control subsystem for the PICK system. The purpose of the Version Control Subsystem is to interact with analyst and the information flowing through the network in order to let the lead analyst decide what changes to keep. How the updates to each analyst connected will be updated.

### Version Control Façade

|  |  |
| --- | --- |
| **Class Name**: Version Control Façade | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**   * Know the instance of Network History and History classes | |
| **Contract:** 18. Send to Store Changes | |
| **Contract Description:** | |
| **Contract Protocols:**   * def send\_to\_network\_history (object) returns None   + **Purpose:** Delegates the changes occurring over the network to the Network History class.   + **Pre-Condition:** * The object represents a log file, event configuration, vector, significant log entry, graph, node, icon, node visibility, or relationship object formatted in JSON that will be stored.   + **Post-Condition:** * The object along with the network changes will be sent to the Network History class.   + **Invariants:** * None * **def** send\_to\_history (object) returns None   + **Purpose:** Delegate the local changes to the History class.   + **Pre-Condition:**     - The object represents a log file, event configuration, vector, significant log entry, graph, node, icon, node visibility, or relationship object formatted in JSON that will be stored.   + **Post-Condition:**     - The object with the local changes will be sent to the History class to be stored. * **Invariants:** None | |
| **Responsibilities:** | **Collaborators:** |
| * Delegate the changes occurring over the network to the Network History class. * Delegate the local changes to the History class. | Network History  History Class |

### Network History

|  |  |
| --- | --- |
| **Class Name**: Network History | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**  None | |
| **Contract:** 19. Manage Network Changes | |
| **Contract Description:** | |
| **Contract Protocols:**   * def store\_network\_changes (object) returns None   + - **Purpose:** Stores changes occurring over the network.     - **Pre-Conditions:** * The object represents a log file, event configuration, vector, significant log entry, graph, node, icon, node visibility, or relationship object formatted in JSON that will be stored.   + - **Post-Conditions:** * The object with the changes will be stored until approved by the lead analyst   + - **Invariants:** None * **def** send\_changes (object) returns approvedChanges   + - **Purpose:** Removes any changes from the stack once they are approved by the lead analyst and then send the changes to the History class to be saved as local changes.     - **Pre-Conditions:** * The object represents a log file, event configuration, significant log entry, graph, node, icon, node visibility, or relationship object formatted in JSON that will be sent to be stored as local changes.   + - **Post-Condition:** * The approved changes represent a network change in JSON format, that were approved by the lead analyst and is now considered a local change. * The approved changes will be removed from the stack. * Before removing changes from the stack, it should not be empty.   + - **Invariants:** None | |
| **Responsibilities** | **Collaborators:** |
| * Store changes occurring over the network * Send network changes to the History class once the changes are approved/committed | History |

### History

|  |  |
| --- | --- |
| **Class Name**: History | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**   * Undo and Redo local changes | |
| **Contract:** 20. Manage Local Changes | |
| **Contract Description:** | |
| **Contract Protocols:** | |
| * def store\_local\_changes (object) returns None * **Purpose:** Stores local changes. * **Pre-Conditions:** * The object represents a log file, event configuration, vector, significant log entry, graph, node, icon, node visibility, or relationship object formatted in JSON that will be stored. * **Post-Conditions:** * The object with the changes will be stored in a stack as a state   + - **Invariants:** None * def save (object) returns None * **Purpose:** Removes any changes from the stack and then send the changes to the DB Façade. * **Pre-Condition:** * The object represents a log file, event configuration, vector, significant log entry, graph, node, icon, node visibility, or relationship object formatted in JSON that will be stored in the DB. * **Post-Condition:** * The local changes, JSON formatted, are sent to the DB Façade to be stored in the DB. * Before removing changes from the stack, it should not be empty. * **Invariants:** None | |
| **Responsibilities:** | **Collaborators:** |
| * Stores local changes in a memento * Send/ receive local changes to the DB | DB Façade |

## Vector Subsystem

This section describes the Vector subsystem for the PICK system. The purpose of the Vector Subsystem is to handle all the system’s model-based classes that are contained in a vector. The subsystem concerns itself with handling all network and local requests that will cause any add, delete, or edit methods to any of the vector’s models.

|  |  |
| --- | --- |
| **Class Name**: Vector Façade | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**   * Handles incoming input data associated to Vector subsystem * Delegates the input data to the appropriate method of the associated class within the Vector Subsystem | |
| **Contract:** Delegate | |
| **Responsibilities** | **Collaborators:** |
| 1. Knows Vector 2. Knows Significant Log Entry 3. Knows Node 4. Knows Node Visibility 5. Knows Icon 6. Knows Relationship 7. Knows Graph | Vector  Significant Log Entry  Node  Node Visibility  Icon  Relationship  Graph |

|  |  |
| --- | --- |
| **Class Name**: Vector | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**   1. Know vector name 2. Know vector description 3. Change vector details 4. Add/remove vector 5. Add/remove significant log entries | |
| **Contract:** Send Query | |
| **Responsibilities** | **Collaborators:** |
| 1. Delegate the request of a component or subsystem to the DB Manager class. | DB Manager (2-6) |
| **Contract:** Receive Query | |
| **Responsibilities:** | **Collaborators:** |
| Delegate the information received from the | DB Manager (5) |

|  |  |
| --- | --- |
| **Class Name**: Significant Log Entry | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**   * Knows log entry number * Knows log entry timestamp * Knows log entry content * Knows host * Knows source * Knows source type | |
| **Contract:** Create Significant Log Entry (or) Associate Log Entry To Vector | |
| **Responsibilities** | **Collaborators:** |
|  |  |
| **Contract:** Edit Significant Log Entry | |
| **Responsibilities:** | **Collaborators:** |
|  |  |

|  |  |
| --- | --- |
| **Class Name**: Graph | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**   * Know export format * Know graphical orientation * Know interval units * Know set of nodes * Know set of relationships * Know current interval * Know position of nodes * Know position of relationships | |
| **Contract:** Edit Graph | |
| **Responsibilities** | **Collaborators:** |
|  | DB |
| **Contract:** Export Graph | |
| **Responsibilities:** | **Collaborators:** |
|  | DB |

|  |  |
| --- | --- |
| **Class Name**: Node | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**   * Know node ID * Know node name | |
| **Contract:** Create Node | |
| **Responsibilities** | **Collaborators:** |
|  |  |
| **Contract:** Edit Node | |
| **Responsibilities:** | **Collaborators:** |
|  |  |
| **Contract:** Delete Node | |
| **Responsibilities:** | **Collaborators:** |
|  |  |

|  |  |
| --- | --- |
| **Class Name**: Node Visibility | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**   * Know node’s ID visibility * Know node’s name visibility * Know node’s timestamp visibility * Know node’s description visibility * Know node’s log entry reference visibility * Know node’s log creator visibility * Know node’s event type visibility * Know node’s icon type visibility * Know node’s source visibility * Know node’s visibility | |
| **Contract:** Edit Node Visibility Property | |
| **Responsibilities** | **Collaborators:** |
| 1. Delegate the request of a component or subsystem to the DB Manager class. |  |

|  |  |
| --- | --- |
| **Class Name**: Relationships | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**   * Know relationship ID | |
| **Contract:** Add Relationship | |
| **Responsibilities** | **Collaborators:** |
|  |  |
| **Contract:** Edit Relationship | |
| **Responsibilities:** | **Collaborators:** |
|  |  |
| **Contract:** Delete Relationship | |
| **Responsibilities:** | **Collaborators:** |
|  |  |

|  |  |
| --- | --- |
| **Class Name**: Icon | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Private Responsibilities**   1. Know icon file path 2. Know icon name 3. Add/delete Icon | |
| **Contract:** Add Icon | |
| **Responsibilities** | **Collaborators:** |
|  |  |
| **Contract:** Delete Icon | |
| **Responsibilities:** | **Collaborators:** |
| D it. |  |

## Networking Component

|  |  |
| --- | --- |
| **Class Name**: Networking | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Description:** Enable interaction with other users of the PICK system over a local network | |
| **Private Responsibilities:** None | |
| **Contract:** 21. Process Network Data |  |
| **Contract Description:** Processes changes received from the network and labels them as network changes | |
| **Contract Protocols:**   * processInfo(String data) returns object   + **Purpose:** Builds objects from data received over the network   + **Pre-conditions:**     - **data:** Is representation of log file, event configuration, vector, significant log entry, graph, node, icon, node visibility, or relationship object formatted in JSON     - **data:** Is not null     - There is a connection established to a server or at least 1 client   + **Post-conditions:**     - If object is instantiated successfully, return object. Else, returns null.   + **Invariants:** None * storeNetworkChange(Object object) returns Map   + **Purpose:** Classify object as change received from the network   + **Pre-conditions:**     - **object:** Is output from processInfo     - **object:** Is not null   + **Post-conditions:**      - Return Map with network change identifier as key and object as value   + **Invariants:** None | |
| **Responsibilities** | **Collaborators** |
| * Receive project information from local network | * Log File * Event Configuration * Vector * Significant Log Entry * Graph * Node * Icon * Node Visibility * Relationship |
|  |  |
| **Contract:** 22.Push |  |
| **Contract Description:** Pushes local data to client(s) or server | |
| **Contract Protocols:**   * push(String data, Socket s) returns String   + **Purpose:** Send object formatted in JSON over the network   + **Pre-conditions:**      - **data:** Is representation of log file, event configuration, vector, significant log entry, graph, node, icon, node visibility, or relationship object encoded as JSON     - **data:** Is not null     - **s:** Is not null or closed   + **Post-conditions:**      - If data is sent successfully, then return success identifier. Else, return failure identifier   + **Invariants:** None | |
| **Responsibilities** | **Collaborators** |
| * Send project information over local network | * Version Control Façade |
|  |  |
| **Contract:** Connect |  |
| **Contract Description:** Handles connections with other instances of PICK | |
| **Contract Protocols:**   * connectToServer(String hostIP) return Socket   + **Purpose:** Establish a connection with a server   + **Pre-conditions:**      - Server is listening for connection requests   + **Post-conditions:**      - If connection is established successfully, start new Thread running the changeListener protocol with the server socket as a parameter and return the server socket. Else, return null   + **Invariants:** None * networkListener(Socket s) return String   + **Purpose:** Listen for data coming over the network   + **Pre-conditions:**     - **s:** Is not null or closed   + **Post-conditions:**      - If s is closed successfully and if instance is not a lead instance, the instance is turned into a lead instance, and return success identifier     - If s is closed successfully and if instance is lead instance, return success identifier     - If s is not closed successfully, return failure identifier   + **Invariants:** None * connectionListener(String hostIP) return String   + **Purpose:** Listen for incoming connection requests   + **Pre-conditions:**     - Lead is working online     - Number of connections is less than 20     - Instance is lead instance   + **Post-conditions:**      - Number of connections is equal to 20     - Return identifier noting that there can’t be more connections established   + **Invariants:** None * numConnections(Socket server) return Integer   + **Purpose:** determine number of connections established to project lead   + **Pre-conditions:**      - **server:** Is not null or closed   + **Post-conditions:**     - If current instance is lead instance, count and return number of clients     - If current instance is not the lead instance, return result of calling the protocol on the server instance   + **Invariants:** None * disconnect() return String   + **Purpose:** Disconnect local instance from all remote instances   + **Pre-conditions:**     - Local socket is not null or closed   + **Post-conditions:**     - Close client or server sockets, close local socket, return success identifier   + **Invariants:** None | |
| **Responsibilities** | **Collaborators** |
| * Establish connection to lead analyst’s instance of PICK * Accept connections from other instances of PICK * Know connection status * Know system’s IP address | * Event Configuration * GUI Façade |

## Event Configuration Component

|  |  |
| --- | --- |
| **Class Name**: Event Configuration | |
| **Superclass**: None | |
| **Subclasses:** None | |
| **Description:** Store investigation event details | |
| **Private Responsibilities:** None | |
| **Contract:** 23. Manage Event Information | |
| **Contract Description:** Returns values and processes changes of Event Configuration name and description | |
| **Contract Protocols:**   * getEventInformation() return Dictionary   + **Purpose:** Return Event Configuration name and description attributes   + **Pre-conditions:**      - Event Configuration is instantiated     - Event Configuration name is not null     - Event Configuration description is not null   + **Post-conditions:**      - Return Event Configuration name and description in a dictionary   + **Invariants:** None * setEventName(String name) return void   + **Purpose:** Set the Event Configuration name attribute   + **Pre-conditions:**      - **name:** Does not contain illegal characters     - **name:** Is not null or empty     - Event Configuration is instantiated   + **Post-conditions:**      - Event Configuration name attribute is equal to name   + **Invariants:** None * setEventDescription(String description) return void   + **Purpose:** Set the Event Configuration description attribute   + **Pre-conditions:**      - **description:** Does not contain illegal characters     - **description:** Is not null or empty     - Event Configuration is instantiated   + **Post-conditions:**      - Event Configuration description attribute is equal to description   + **Invariants:** None | |
| **Responsibilities** | **Collaborators** |
| * Know event name * Know event description | * GUI Façade |
|  |  |
| **Contract:** 24. Use Event Time Frame | |
| **Contract Description:** Returns values and processes changes of Event Configuration start and end timestamps | |
| **Contract Protocols:**   * getEventTimeFrame() return Dictionary   + **Purpose:** Return Event Configuration start timestamp and end timestamp attributes   + **Pre-conditions:**      - Event Configuration object is instantiated     - Event Configuration start timestamp is not null     - Event Configuration end timestamp is not null   + **Post-conditions:**      - Return Event Configuration start timestamp and end timestamp attributes in a dictionary   + **Invariants:** None * setEventStartTimeStamp(String startTimeStamp) return void   + **Purpose:** Set the Event Configuration start timestamp attribute   + **Pre-conditions:**      - **startTimeStamp:** Does not contain illegal characters and is formatted correctly     - **startTimeStamp:** Is not null or empty     - Event Configuration object is instantiated   + **Post-conditions:**      - Event Configuration start timestamp attribute is equal to startTimeStamp   + **Invariants:** None * setEvenEndTimeStamp(String endTimeStamp) return void   + **Purpose:** Set the Event Configuration end timestamp attribute   + **Pre-conditions:**      - **endTimeStamp:** Does not contain illegal characters and is formatted correctly     - **endTimeStamp:** Is not null or empty     - Event Configuration object is instantiated   + **Post-conditions:**      - Event Configuration end timestamp attribute is equal to endTimeStamp   + **Invariants:** None | |
| **Responsibilities** | **Collaborators** |
| * Know event start timestamp * Know event end timestamp | * GUI Façade |
|  |  |
| **Contract:** 25. Manage File Directories | |
| **Contract Description:** Returns values and processes changes of Event Configuration file directories | |
| **Contract Protocols:**   * getFileDirectories() return Map   + **Purpose:** Return Event Configuration root, red team folder, white team folder, and blue team folder directory attributes   + **Pre-conditions:**      - Event Configuration root directory attribute is not null or empty     - Event Configuration red team folder directory is not null or empty     - Event Configuration blue team folder directory is not null or empty     - Event Configuration white team folder directory is not null or empty     - Event Configuration object is instantiated   + **Post-conditions:**      - Return Event Configuration root, red team folder, white team folder, and blue team folder directory attributes in a map where “root”, “red”, “white”, and “blue” are the keys, respectively   + **Invariants:** None * setRootDirectory(String rootDirectory) return void   + **Purpose:** Set the Event Configuration root directory attribute   + **Pre-conditions:**      - **rootDirectory:** Is formatted correctly and does not contain illegal characters     - **rootDirectory:** Is not null or empty     - Event Configuration object is instantiated   + **Post-conditions:**      - Event Configuration root directory attribute is equal to rootDirectory   + **Invariants:** None * setRedDirectory(String redDirectory) return void   + **Purpose:** Set the Event Configuration red directory attribute   + **Pre-conditions:**      - **redDirectory:** Is formatted correctly and does not contain illegal characters     - **redDirectory:** Is not null or empty     - Event Configuration object is instantiated   + **Post-conditions:**      - Event Configuration red directory attribute is equal to redDirectory   + **Invariants:** None * setblueDirectory(String blueDirectory) return void   + **Purpose:** Set the Event Configuration blue directory attribute   + **Pre-conditions:**      - **blueDirectory:** Is formatted correctly and does not contain illegal characters     - **blueDirectory:** Is not null or empty     - Event Configuration object is instantiated   + **Post-conditions:**      - Event Configuration blue directory attribute is equal to blueDirectory   + **Invariants:** None * setWhiteDirectory(String whiteDirectory) return void   + **Purpose:** Set the Event Configuration white directory attribute   + **Pre-conditions:**      - **whiteDirectory:** Is formatted correctly and does not contain illegal characters     - **whiteDirectory:** Is not null or empty     - Event Configuration object is instantiated   + **Post-conditions:**      - Event Configuration white directory attribute is equal to whiteDirectory   + **Invariants:** None | |
| **Responsibilities** | **Collaborators** |
| * Know root directory * Know red team folder directory * Know white team folder directory * Know blue team folder directory | * GUI Façade |
|  |  |
| **Contract:** 26. Manage Lead IP | |
| **Contract Description:** Returns values and processes changes of Event Configuration Lead IP address attribute | |
| **Contract Protocols:**   * getHostIP() return String   + **Purpose:** Return Event Configuration lead analyst IP address attribute   + **Pre-conditions:**      - Event Configuration lead analyst IP address attribute is not null     - Event Configuration object is instantiated   + **Post-conditions:**      - Return Event Configuration lead analyst IP address attribute   + **Invariants:** None * setHostIP(String hostIP) return void   + **Purpose:** Set the Event Configuration lead analyst IP address attribute   + **Pre-conditions:**      - **hostIP:** Is formatted correctly and does not contain illegal characters     - **hostIP:** Is not null or empty     - Event Configuration object is instantiated   + **Post-conditions:**      - Event Configuration lead analyst IP address attribute is equal to hostIP   + **Invariants:** None * isLeadInsance() return boolean   + **Purpose:** Return whether current instance is a lead instance or not   + **Pre-conditions:**     - Event Configuration object is instantiated   + **Post-conditions:**     - Return current value of Event Configuration Lead attribute | |
| **Responsibilities** | **Collaborators** |
| * Know the lead analyst’s IP address * Know if current instance is lead analyst’s instance | * GUI Façade * Networking |
|  |  |
| **Contract:** 27. Get Connection Count | |
| **Contract Description:** Returns value of Event Configuration Connection Count attribute | |
| **Contract Protocols:**   * getConnectionCount() return Integer   + **Purpose:** Return the number of connections established to project lead   + **Pre-conditions:**      - Event Configuration object is instantiated   + **Post-conditions:**      - Return current number of connections established to project lead   + **Invariants:** None | |
| **Responsibilities** | **Collaborators** |
| * Know number of connections established | * Networking |

## Signal Controller Component

|  |  |
| --- | --- |
| **Class Name**: Signal Controller | |
| **Superclass**: None | |
| **Subclasses**: None | |
| **Description:** Acts as third party between model and user interfaces by signaling model changes to user interfaces | |
| **Private Responsibilities:**   * None | |
| **Contract:** 28. Get Model Changes | |
| **Contract Description:** Allows user interfaces to subscribe to model changes and provides method to broadcast model changes | |
| **Contract Protocols:**   * subscribe(Object object) return void   + **Purpose:** Register object as client to model change signals   + **Pre-condition:**      - **object:** Is instantiated user interface   + **Post-condition:**      - User interface reference is stored   + **Invariants:** None * broadcastChange(String identifier) return void   + **Purpose:** Broadcast update signal to all subscribers   + **Pre-conditions:**      - **identifier:** Is not null     - There is at least 1 subscriber   + **Post-conditions:**      - Subscribers receive signal to update models with respect to the received identifier   + **Invariants:** None | |
| **Responsibilities** | **Collaborators** |
| * Know when model information is changed * Transmit signal of a model’s change to subscribers | * Version Control Façade * GUI Façade |

## Log File Component

|  |  |
| --- | --- |
| **Class Name:** Log File | |
| **Superclass:** None | |
| **Subclass:** None | |
| **Description:** Log is to keep track of the statuses of each log file that gets into ingested | |
| **Private Responsibilities**   * Keep a list of Boolean parameters for each log file in the system * Know name of log file | |
| **Contract:** 29. Cleansing status | |
| **Contract Description:** Keeps track of the cleansing status | |
| **Contract Protocol:**   * setCleansingStatus (String name, Boolean bool) return void   + **Purpose:** Set the cleansing status   + **Pre-conditions:**     - name: name of a log file     - bool: Boolean     - Log file exists     - Log file is cleansed   + **Post-conditions:**     - Log file specified cleansing status attribute is true   + **Invariants:**      - None * getCleansingStatus (String name) return Boolean   + **Purpose:** Return current cleansing status of an individual log file   + **Pre-conditions:**     - name: Of type string, representing the name of a log file     - Log file in parameters exists   + **Post-conditions:**     - A Boolean is returned indication whether the log file is cleansed or not   + **Invariants:**      - None | |
| **Responsibilities:** | **Collaborators:** |
| * Know cleansing status of log file | Ingestion |
| **Contract:** 30. Validation status | |
| **Contract Description:** Keeps track of the validation status | |
| **Contract Protocol:**   * setValidationStatus (String name, Boolean bool) return void   + **Purpose:** Change Boolean attribute of a log file   + **Pre-conditions:**     - name: of type string, indicating the log file to change     - bool: Boolean to what the log file attribute should change to     - Log file exists   + **Post-conditions:**     - I Validation status for the log file provided is changed to true   + **Invariants:**      - None * getValidationStatus (String Name) return Boolean   + **Purpose:** Return the current status of validation for a log file   + **Pre-conditions:**     - name: of type string     - Log file exists   + **Post-conditions:**     - A Boolean is returned indicating whether the log file has been validated or not   + **Invariants:**      - None | |
| **Responsibilities:** | **Collaborators:** |
| * Know validation status of file | * Ingestion |
| **Contract:** 31. Ingestion status | |
| **Contract Description:** keeps track of validation status | |
| **Contract Protocol:**   * setIngestionStatus (String name, Boolean bool) return void   + **Purpose:** To change Boolean attribute of ingestion for log file   + **Pre-conditions:**     - name: Type string, name of a log file.     - bool: Boolean     - The log file passed in the parameters exists   + **Post-conditions:**     - The Boolean attribute for the log file is now true   + **Invariants:**      - None * setIngestionStatus (String name) return Boolean   + **Purpose:** Return the current status of ingestion for a log file   + **Pre-conditions:**     - name: log file must exist   + **Post-conditions:**     - The Boolean indicating the status of ingestion for that log file is returned   + **Invariants:**      - None | |
| **Responsibilities:** | **Collaborators:** |
| * Know ingestion status of file | * Ingestion |
| **Contract:** 32. Acknowledgement status | |
| **Contract Description:** “Force” validates files**,** changing its status | |
| **Contract Protocol:**   * setAcknoledgemetnSatus (String name) return void   + **Purpose:** Change the validation status for the log file to be accepted into the system   + **Pre-conditions:**     - name:     - The lg file needs to exist   + **Post-conditions:**     - The log file status is validated     - The log file is ingested   + **Invariants:**      - None | |
| **Responsibilities:** | **Collaborators:** |
| * Know Acknowledgement status of file | * Enforcement Action Report |

# Database

## Data Model Design

This section describes and demonstrates the use of a database in our system. In our system it will be necessary to store log files, significant log entries, vectors, nodes, graphs, event configurations and other major objects in our system. The storing mechanism will facilitate data retrieval, data filtering, and any data modification that will serve our system. The database will be utilized to retrieve the stored contents for an analyst to manipulate and create visualizations through our system’s interface. The data received from our system will vary in a way that some log files will be of types that currently might be known and be expected in our database but in the future unknown types of data will need to be accepted and stored. Due to this constraint the use of a non-SQL database fits the design of the PICK system. Section 4.1.1. further discusses the way the data is stored in the database and the relations between the data.

### Data Schema Design

The PICK system will need to handle massive amounts of data, making MongoDB fit for the development of the system. Due to its schema-less structure the data will be stored in a flexible manner, rather than having normalization in tables and having strict fields filled out before the data is accepted in a table, MongoDB allows for data to be accepted regardless of the field being missing without any normalization. Figure # demonstrates the storing of information inside the database. The way the data is stored is through four collections containing documents with subdocuments facilitating the way the data is searched and updated in the DB. The main collections consist of the Vector, Node, Relationships, and Event Configuration objects. When a vector is pulled from the database its graph along with its nodes and relationships references will also be pulled. If a specific node needs to be searched the node will then be referenced from the Node collection. The nodes will then contain the necessary attributes inside the main document as well as subdocuments. Since majority of the data is contained in the Node collection such as, Log entry reference, Icon details, Log file details, and its Node visibility the retrieval of that data can be searched faster. When a specific node needs to be searched and filtered within a vector, the node will be pulled by selecting the desired node inside the vector document, rather than looking for it in the Node collection with all the node documents. Fast data retrieval and data modifications are very crucial for the PICK system.

A screenshot of a cell phone

Description automatically generated

**Figure #:** MongoDB Document Schema Design

## %