­Prevent, Mitigate, and Recover (PMR) Insight

Collective Knowledge System (PICK)

Software Design Document

Version 1.7

05/09/2020

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 |
| 1.6 | 3/30/2020 | Elizabeth | Completed Sec.4 |
| 1.7 | 5/07/2020 | Antoine | Updated section 2.2.3, 2.2.4 |
| 1.8 | 05/09/20 | Yamel | Updated Sections |
| 1.9 | 05/9/2020 | Elizabeth | Updated Sec. 4 , Sec 3.3 Sec 2.3 |
| 2.0 | 05/09/2020 | Manuel | Added section 3.6 & 3.7 |
| 2.1 | 05/09/2020 | Antoine | Updated 3.4.1 – 3.4.7, 3.5, 3.8, updated PICK System graph |

Table of Contents

[Document Control ii](#_Toc39963149)

[Approval ii](#_Toc39963150)

[Document Change Control ii](#_Toc39963151)

[Distribution List ii](#_Toc39963152)

[Change Summary ii](#_Toc39963153)

[1. Introduction 1](#_Toc39963154)

[1.1. Purpose and Intended Audience 1](#_Toc39963155)

[1.2. Scope of Product 1](#_Toc39963156)

[1.3. References 1](#_Toc39963157)

[1.4. Definitions, Acronyms, and Abbreviations 1](#_Toc39963158)

[1.4.1. Definitions 1](#_Toc39963159)

[1.4.2. Acronyms 2](#_Toc39963160)

[1.4.3. Abbreviations 2](#_Toc39963161)

[1.5. Overview 2](#_Toc39963162)

[2. Decomposition Description 3](#_Toc39963163)

[2.1. System Collaboration Diagram 3](#_Toc39963164)

[2.2. Subsystem and Component Descriptions 3](#_Toc39963165)

[2.2.1. Database Subsystem 3](#_Toc39963166)

[2.2.2. Ingestion Subsystem 3](#_Toc39963167)

[2.2.3. GUI Subsystem 3](#_Toc39963168)

[2.2.4. Vector Subsystem 4](#_Toc39963169)

[2.2.5. Event Configuration Component 4](#_Toc39963170)

[2.2.6. Log File Component 4](#_Toc39963171)

[2.3. Dependencies 4](#_Toc39963172)

[3. Detailed Description of Components 5](#_Toc39963173)

[3.1. Database Subsystem 5](#_Toc39963174)

[3.1.1. DB Façade 5](#_Toc39963175)

[3.2. Ingestion Subsystem 7](#_Toc39963176)

[3.2.1. Ingestion 7](#_Toc39963177)

[3.2.2. Enforcement Action Report 8](#_Toc39963178)

[3.2.3. Splunk Façade 9](#_Toc39963179)

[3.2.4. Cleanser 10](#_Toc39963180)

[3.3. Log File Component 10](#_Toc39963181)

[3.4. Graphical User Interface Subsystem 12](#_Toc39963182)

[3.4.1. Main Window 12](#_Toc39963183)

[3.4.2. Generic Window 12](#_Toc39963184)

[3.4.3. Generic Frame 13](#_Toc39963185)

[3.4.4. Event Configuration Frame 14](#_Toc39963186)

[3.4.5. Vector Database Frame 14](#_Toc39963187)

[3.4.6. Vector Information Frame 15](#_Toc39963188)

[3.4.7. Vector Frame 16](#_Toc39963189)

[3.5. Event Configuration Component 16](#_Toc39963190)

[3.6. Graph Subsystem 17](#_Toc39963191)

[3.6.1. Graph Window 17](#_Toc39963192)

[3.6.2. Graphics Scene 18](#_Toc39963193)

[3.6.3. Scene 18](#_Toc39963194)

[3.6.4. Graphics View 19](#_Toc39963195)

[3.6.5. Node 19](#_Toc39963196)

[3.6.6. Graphic Node 20](#_Toc39963197)

[3.6.7. Node Content Widget 21](#_Toc39963198)

[3.6.8. Socket 21](#_Toc39963199)

[3.6.9. Graphic Socket 22](#_Toc39963200)

[3.6.10. Node Edge 22](#_Toc39963201)

[3.6.11. Graphics Edge 23](#_Toc39963202)

[3.6.12. Graphics Edge Direct 24](#_Toc39963203)

[3.6.13. Serializable 24](#_Toc39963204)

[3.7. Vector Facade 25](#_Toc39963205)

[4. Database 26](#_Toc39963206)

[4.1. Data Model Design 26](#_Toc39963207)

[4.1.1. Data Schema Design 26](#_Toc39963208)

[% 27](#_Toc39963209)

# 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

No abbreviations found in the document.

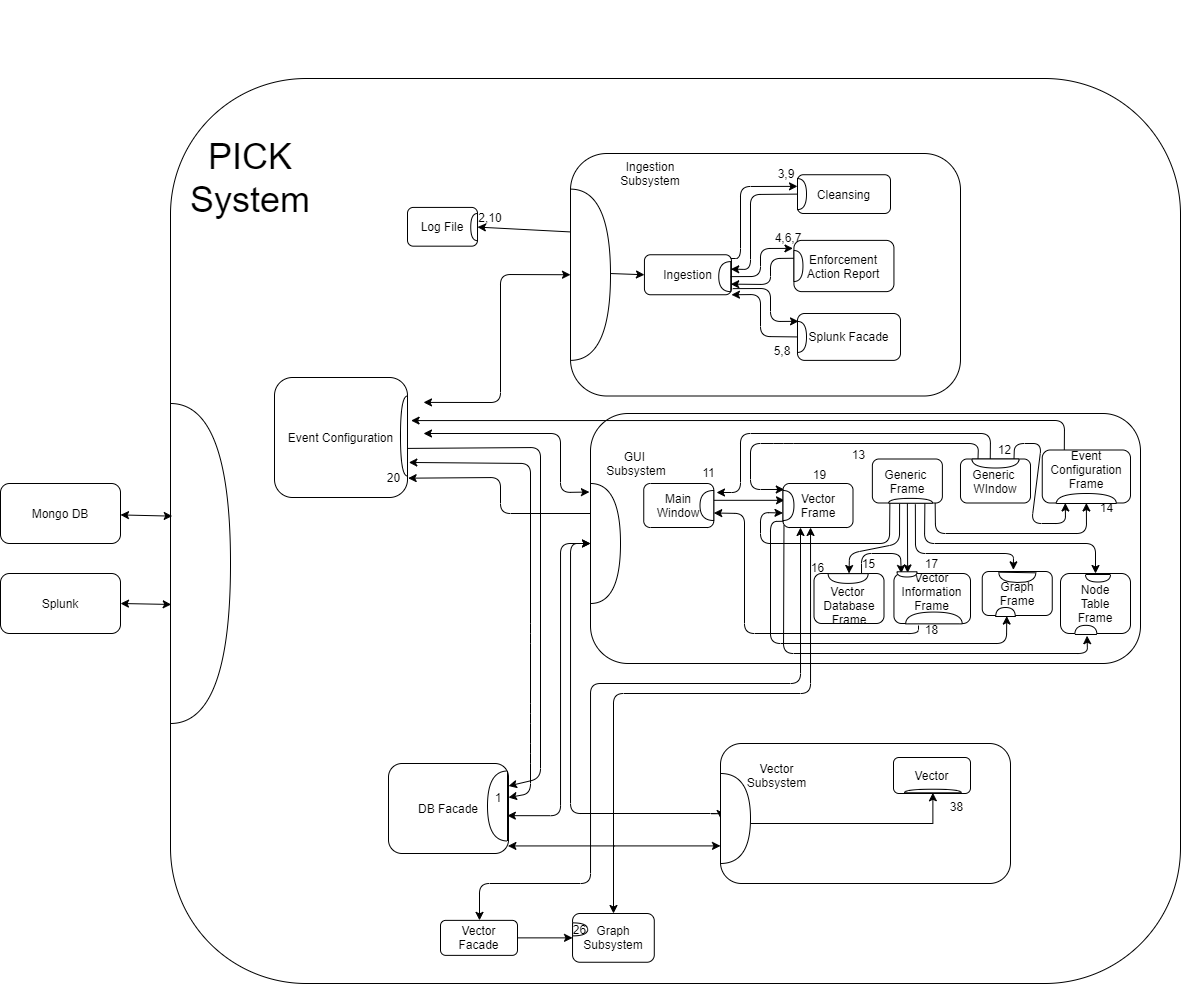
## Overview

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.

### Database Subsystem

This subsystem will be further discussed in 3.1, 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 |

### Ingestion Subsystem

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

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

### GUI Subsystem

This subsystem will be further discussed in section 3.4.

|  |
| --- |
| **Subsystem:** GUI Subsystem |
| **Classes:** MainWindow, GenericWindow, GenericFrame, EventConfigurationFrame, VectorDatabaseFrame, VectorInformationFrame, VectorFrame |
| **Description:** Responsible for displaying system content and getting user input. |
| **Contracts:**   1. Display Main Window    1. Server: Main Window 2. Control Window    1. Server: Generic Window 3. Set Frame Layout    1. Server: Generic Frame 4. Save Event Configuration    1. Server: Event Configuration Frame 5. Manage Vectors    1. Server: Vector Database Frame 6. Display Vector Database    1. Server: Vector Database Frame 7. Update Vector Information    1. Server: Vector Information Frame 8. Display Vector Information    1. Server: Vector Information Frame 9. Display Vector Workflow    1. Server: Vector Frame |

### Vector Subsystem

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

|  |
| --- |
| **Subsystem:** Vector Subsystem |
| **Classes:** Vector, Graph |
| **Description:** Responsible for modeling a vector and its information as data |
| **Contracts:**   1. Manage Vector    1. Server: Vector |

### Event Configuration Component

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

### Log File Component

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

### Graph Component

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

## Dependencies

The system will have two main dependencies, Mongo DB and Splunk. The Splunk API is used with our system to parse and retrieve log entries into our system, thus Splunk is a requirement that must be downloaded as well in order to use our system. The download can be found in splunk.com. We depend on understanding these components in order to deliver an optimize system. As a storing mechanism MongoDB is used to save the objects within the system and be able to maintain the updated content from the GUI to the DB and the vice versa. In order to meet this requirement, research on a python driver for MongoDB will be necessary to develop the database feature in the system.

# Detailed Description of Components

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

## 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:** 1. Query to Manipulate DB | |
| **Contract Description**: | |
| **Contract Protocols:**   * **add\_object** (data, collectionname) returns None   + **Purpose:** Adds objects’ content into a specific collection in the DB.   + **Pre-Condition:**     - The data represents the content of a log file, event configuration, vector, significant log entry, graph, node, icon, node visibility, or relationship object formatted in a dictionary that will be manipulated in the DB.     - The collectionname is a String specifying in which collection to search to add the data.     - The DB connection is established.   + **Post-Condition:**     - The object will be saved into the DB along with its content, the DB doesn’t return anything.     - An acknowledgment is printed out to confirm that the data was added into the DB   + **Invariants:** None * **del\_object** (key, value, collectionname) returns None   + **Purpose:** Deletes objects’ content from the DB.   + **Pre-Condition:**     - The key is a String specifying the attribute of the document that will be searched on to find the right document to delete.     - The value is the String data that will be used to find a doc matching that value.     - The collectionname is a String specifying in which collection to search to delete the doc.     - The DB connection is established.   **Post-Condition:**   * + - The content of a vector, node, graph or any other object in the system will be deleted from the DB along with its content.     - An acknowledgment is printed out to confirm that the data was deleted from the DB   + **Invariants:** None * **search\_object** (key, value, collectionname) returns result   + - **Purpose:** Searches for an object stored in the DB with matching searching criteria. * **Pre-Condition:**    + The key is a String specifying the attribute that will be used to search for the document.   + The value is the String data that will be used to find the doc matching that value.   + The collectionname is a String specifying which collections to search for the document.   + The DB connection is established. * **Post-Condition:** * If the collectionname is in “Event Configuration” then only find that one document inside that collection. * If the document is not in the “Event Configuration” collection, then the key and value along with the collectionname will be sent to the DB to search the content in the DB. * The DB will return the result matching the search criteria. * **Invariants:** None * **update\_object** (objectid, objectdata, collectionname) returns None * **Purpose:** Updates one or many object's attributes within a document in a specified collection in the DB. * **Pre-Condition:** * The objectid is a String specifying a document’s ID, assigned by the DB, to be searched in the DB. * The object data represents the content of a log file, event configuration, vector, significant log entry, graph, node, icon, node visibility, or relationship object formatted in a dictionary that contains the change(s) to be updated in the DB. * The collectionname is a String specifying which collection to search to update the document. * The DB connection is established. * **Post-Condition:** * The id will be used to set the new changes from the object data into the document in the DB. * An acknowledgment is printed out to confirm that the data was updated. * **Invariants:** None * **get\_vector\_list** () returns vectorlist * **Purpose:** Retrieves all the vectors in DB and inserts them into a list to keep track of them in the system. * **Pre-Condition:** * The DB connection is established * **Post- Condition:** * Appends the vectors from the DB to a list. * Returns the vectorlist. * **Invariants:** None * **check\_db** () returns None * **Purpose:** Detects user’s operating system to provide the MongoDB executable path to connect to MongoDB when the PICK system starts. * **Pre-Condition:** * The DB connection is established * **Post- Condition:** * Checks the count of all the documents inside every collection in the DB. * Appends the vectors from the DB to a list. * Returns the vectorlist. * **Invariants:** Collection is not in the DB. * **connect\_to\_db** () returns None * **Purpose:** Detects user’s operating system to provide the MongoDB executable path to connect to MongoDB when the PICK system starts. * **Pre-Condition:** MongoDB is installed in the user’s machine. * **Post- Condition:** * Detects the user’s operating system. * Assigns an executing path based on operating system. * **Invariants:** None | |
| **Responsibilities** | **Collaborators:** |
| * Run query to add objects to the DB * Run query to delete objects in the DB * Run query to search for objects in the DB * Run query to update objects in the DB * Run query to check if the DB is empty. * Runs MongoDB executable file | * Event Configuration * Vector * Graph |

## Ingestion Subsystem

The following section will depict and describe how the Ingestion subsystem. 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, acts like a façade for the ingestion class. | | |
| **Private Responsibilities**   * Know logfile | | |
| **Responsibilities:** | **Collaborators:** | |
| * Know path * Scan for files | * Event Config () * Log File () | |
| **Contract:**  2. Load & Save log files | | |
| **Contract Description:** Creates log files objects | | |
| **Contract Protocols:**   * populate\_LogFiles(path)   + **Purpose:** Populates logfiles, from the directory given   + **Pre-conditions:**     - Path must be given     - Takes in a log file object for its parameter   + **Post-conditions:**     - Creates log file objects per file found in path, saves current logfiles in a list to continue the ingestion process.   + **Invariants:** None | | |
| **Responsibilities:** | **Collaborators:** | |
| * Sends logfiles to be cleansed | * Cleansing () | |
| **Contract:**  3. Sends Logfiles to be cleansed | | |
| **Contract Description:** Cleanses the logfiles by calling the cleanser class | | |
| **Contract Protocols:**   * cleanse\_LogFiles(logfile\_list)   + **Purpose:** Sends logfiles to be cleansed & updates their cleansing status to true   + **Pre-conditions:**     - Logfile list must be given     - List must contain log file objects   + **Post-conditions:**     - Updates cleansing status on log files in logfile list.     - Creates new folder for cleansed files, so originals are untouched   + **Invariants:** None | | |
| **Responsibilities:** | | **Collaborators:** |
| * Validates logfiles | | Enforcement Action Report () |
| **Contract:**  4. Validates logfiles | | |
| **Contract Description:** Sends logfiles to Enforcement Action Report to confirm validation | | |
| **Contract Protocols:**   * validate\_LogFiles(logfile\_list)   + **Purpose:** Sends logfiles to be validated & updates their validation status to true   + **Pre-conditions:**     - Logfile list must be given     - List must contain log file objects   + **Post-conditions:**     - Updates validation status on log files in logfile list.     - Returns error message if logfile is found to be invalid   + **Invariants:** None * force\_validate(logfile)   + **Purpose:** Changes validation status of logfiles if analyst wishes to validate them even if they have errors   + **Pre-conditions:**     - Logfile must be given   + **Post-conditions:**     - Updates validation status on log files from false to true.   + **Invariants:** None | | |
| **Responsibilities:** | | **Collaborators:** |
| * Sends log files to Splunk | | Splunk Facade () |
| **Contract:**  5. Sends Logfiles to Splunk | | |
| **Contract Description:** Sends logfiles to Splunk to be parsed and uploaded | | |
| **Contract Protocols:**   * upload\_logfiles\_to\_splunk(logfile\_list)   + **Purpose:** Sends logfiles to be uploaded to Splunk so it can be parsed into entries   + **Pre-conditions:**     - Logfile list must be given     - List must contain log file objects     - User must know username & password for splunk   + **Post-conditions:**     - Updates ingestion status on log files in logfile list.     - Uploads logfiles to Splunk & returns parsed entries   + **Invariants:** None | | |

### Enforcement Action Report

|  |  |
| --- | --- |
| **Class Name**: Enforcement Action Report | |
| **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:** 6. Configure relative log file with error | |
| **Responsibilities**   * Know log file | **Collaborators:**  Ingestion () |
| **Contract:**  7.Configure errors | |
| **Contract Description**: Configures validation error if found in the logfile | |
| **Contract Protocols:**   * check file(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 * has\_date(row, other\_words=True)   + **Purpose:** Find out where the given row in the logfile has a date, returns True if id does, otherwise False. Helper function for check\_file.   + **Pre-condition**     - Must be given a string or row of the logfile   + **Post-condition**     - Returns True or False depending if a date is found in the given string/row.   + **Invariants**     - None | |

### Splunk Façade

|  |  |
| --- | --- |
| **Class Name:** Splunk Façade | |
| **Superclass:** None | |
| **Subclass:** None | |
| **Description:** Class that facilitates 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 | |
| **Responsibilities:** | **Collaborators:** |
| * Reference to logfile | Ingestion () |
| **Contract: 8.** Provide Splunk Usability | |
| **Contract Description:** | |
| **Contract Protocol:**   * get\_credentials() returns credentials to be used   + **Purpose:** Return Splunk credentials to be used so it can be used by other functions.   + **Pre-conditions:**     - None   + **Post-conditions:**     - Systems knows Splunk username and password to use for uploading and retrieving   + **Invariants:**      - None * upload (logfile, username, password)   + **Purpose:** Upload files to Splunk to be parsed   + **Pre-conditions:**     - Credentials are correct to connect to Splunk     - Reference to logfile     - There is a connection established with Splunk   + **Post-conditions:**     - Logfiles are successfully uploaded to Splunk & parsed   + **Invariants:**      - None * get\_log\_entries (username, password)   + **Purpose:** Retrieve parsed log entries from Splunk   + **Pre-conditions:**     - Credentials are correct to connect to Splunk     - There is a connection established with Splunk     - There are log entries in Splunk   + **Post-conditions:**     - Logfiles are successfully retrieved   + **Invariants:**      - None | |

### 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 & Non-ASCII characters | |
| **Responsibilities:** | **Collaborators:** |
| * Reference to logfile | Ingestion () |
| **Contract:** 9. Cleanse log files | |
| **Contract Description:** Cleanses log files from non-ascii characters and extra spacing | |
| **Contract Protocol:**   * Cleanse (logfile) returns a cleansed logfile in a created directory “Cleansed”   + **Purpose:** Cleanses log files from non-ascii characters and extra spacing   + **Pre-conditions:**     - Reference to logfile is given   + **Post-conditions:**     - Log files contents are cleansed   + **Invariants:**      - None * is\_ascii (string s) returns only characters in string that are non-ascii   + **Purpose:** Helper function for cleanse class to determine whether logfile has non-ascii characters   + **Pre-conditions:**     - String/row of logfile is provided   + **Post-conditions:**     - Log files contents are cleansed of non-ascii characters   + **Invariants:**      - None | |

## 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 | |
| **Responsibilities:** | **Collaborators:** |
| * Reference to logfile | Ingestion () |
| **Contract: 10.** Manage log file statuses | |
| **Contract Description:** Manage log file statuses | |
| **Contract Protocol:**   * setCleansingStatus (boolean b) return void   + **Purpose:** Set the cleansing status of the logfile object   + **Pre-conditions:**     - Log file exists     - Log file is cleansed     - Boolean value is given   + **Post-conditions:**     - Log file cleansing status is set to b.   + **Invariants:**      - None * getCleansingStatus () return Boolean   + **Purpose:** Return current cleansing status of an individual log file   + **Pre-conditions:**     - Log file exists   + **Post-conditions:**     - Returns the cleansing status   + **Invariants:**      - None * setValidationStatus (boolean b) return void   + **Purpose:** Change Boolean attribute of a log file   + **Pre-conditions:**     - Log file exists     - Boolean value is given   + **Post-conditions:**     - Validation status for the log file is set to b.   + **Invariants:**      - None * getValidationStatus () return Boolean   + **Purpose:** Return the current status of validation for a log file   + **Pre-conditions:**     - Log file exists   + **Post-conditions:**     - Returns the validation status   + **Invariants:** None * setIngestionStatus (boolean b) return void   + **Purpose:** To change Boolean attribute of ingestion for log file   + **Pre-conditions:**     - Boolean value is given     - Log file exists   + **Post-conditions:**     - Ingestion status of log file is set to b.   + **Invariants:**      - None * setIngestionStatus () return Boolean   + **Purpose:** Return the current status of ingestion for a log file   + **Pre-conditions:**     - log file exists   + **Post-conditions:**     - Returns the ingestion status   + **Invariants:**      - None * setAcknoledgemetnSatus (Boolean b) return void   + **Purpose:** Change the validation status for the log file to be accepted into the system   + **Pre-conditions:**     - Boolean value given     - Log file exists   + **Post-conditions:**     - The log file status is validated     - The log file is ingested   + **Invariants:** None | |

## 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.

### Main Window

|  |  |
| --- | --- |
| **Class Name**: MainWindow | |
| **Superclass**: QMainWindow | |
| **Subclasses**: None | |
| **Description:** Display the main point of interaction between the user and the PICK system | |
| **Private Responsibilities:**   * Get input from user | |
| **Contract:** 11. Display Main Window | |
| **Contract Description:** Displays main GUI elements | |
| **Contract Protocols:**   * init\_UI() return none   + **Purpose:** Instantiates the main window with current data   + **Pre-conditions:**      - Project instance has been created   + **Post-conditions:**      - Main window is displayed with information from database   + **Invariants:** None * update\_data() return none   + **Purpose:** Updates the information shown on the user interface   + **Pre-conditions:**     - Model data has sent signal of change   + **Post-conditions:**     - Main window is updated with new model data   + **Invariants:**     - None | |
| **Responsibilities** | **Collaborators** |
| * Display toolbar * Display vector frame * Display table of ingested log entries | * VectorFrame * GenericWindow |

### Generic Window

|  |  |
| --- | --- |
| **Class Name**: Generic Window | |
| **Superclass**: QDialog | |
| **Subclasses**: None | |
| **Description:** Display frames as pop-up windows | |
| **Private Responsibilities:**   * None | |
| **Contract:** 12. Control Window | |
| **Contract Description:** Controls window state and layout | |
| **Contract Protocols:**   * closeEvent() return none   + **Purpose:** Closes the window displaying the frame and notifies of closure   + **Pre-conditions:**      - Window has been instantiated     - Pending data has been saved or deleted by frame   + **Post-conditions:**      - Window is closed   + **Invariants:** None * show() return none   + **Purpose:** Displays the window   + **Pre-conditions:**      - Window has been instantiated     - Frame has been instantiated   + **Post-conditions:**      - Window is shown on screen   + **Invariants:** None * addWidget(QWidget widget) return none   + **Purpose:** Adds the given widget to the layout of the window   + **Pre-conditions:**      - Widget has been instantiated     - Layout type has been declared   + **Post-conditions:**      - Widget is part of the layout     - Widget is shown   + **Invariants:** None | |
| **Responsibilities** | **Collaborators** |
| * Show window with frame * Emit window close signal | * VectorDatabaseFrame * EventConfigurationFrame * MainWindow |

### Generic Frame

|  |  |
| --- | --- |
| **Class Name**: GenericFrame | |
| **Superclass**: QFrame | |
| **Subclasses**: None | |
| **Description:** Structures data in a frame | |
| **Private Responsibilities:**   * Hold QWidgets and QObjects | |
| **Contract:** 13. Set Frame Layout | |
| **Contract Description:** Sets the layout of the frame | |
| **Contract Protocols:**   * addWidget(QWidget widget) return none   + **Purpose:** Adds the given widget to the layout   + **Pre-conditions:**      - Widget has been instantiated     - Layout type has been declared   + **Post-conditions:**      - Widget is part of the layout     - Widget is shown   + **Invariants:** None | |
| **Responsibilities** | **Collaborators** |
| * Display frame | * VectorDatabaseFrame * EventConfigurationFrame |

### Event Configuration Frame

|  |  |
| --- | --- |
| **Class Name**: EventConfigurationFrame | |
| **Superclass**: GenericFrame | |
| **Subclasses**: None | |
| **Description:** Handles displaying and editing of event configuration information | |
| **Private Responsibilities:**   * Display event configuration information * Get input from user | |
| **Contract:** 14. Save Event Configuration | |
| **Contract Description:** Sets the layout of the frame | |
| **Contract Protocols:**   * save\_forms() return none   + **Purpose:** Updates event configuration information from data in fields   + **Pre-conditions:**      - Event Configuration has been instantiated     - Event Configuration information has been altered   + **Post-conditions:**      - Event Configuration information is updated in database     - Frame is updated with new information   + **Invariants:** None | |
| **Responsibilities** | **Collaborators** |
| * Update event configuration information | * VectorDatabaseFrame * EventConfigurationFrame |

### Vector Database Frame

|  |  |  |
| --- | --- | --- |
| **Class Name**: VectorDatabaseFrame | | |
| **Superclass**: GenericFrame | | |
| **Subclasses**: None | | |
| **Description:** Display list of all vectors and their information | | |
| **Private Responsibilities:**   * Get input from user | | |
| **Contract:** 15. Manage Vectors | | |
| **Contract Description: A**dds, or deletes vectors from the system | | |
| **Contract Protocols:**   * Add\_vector() return none   + **Purpose:** Creates new vector with default information   + **Pre-conditions:**      - User input request to add vector   + **Post-conditions:**      - Vector is created and stored in database   + **Invariants:** None * delete\_vector() return none   + **Purpose:** Deletes selected vector   + **Pre-conditions:**      - User input request to delete vector   + **Post-conditions:**      - Vector is deleted from database   + **Invariants:** None | | |
| **Responsibilities** | | **Collaborators** |
| * Add vector to system * Delete vector from system | | * DBFaçade (1) * Vector |
| **Contract: 16.** Display Vector Database | | |
| **Contract Description:** Displays list of vectors and their information | | |
| **Contract Protocols:**   * Init\_list() return none   + **Purpose:** Initializes list of vectors   + **Pre-conditions:**      - There is at least one vector in the database   + **Post-conditions:**      - List of vectors is populated     - VectorInformationFrame is instantiated   + **Invariants:** None | | |
| **Responsibilities** | **Collaborators** | |
| * Display list of vectors * Display vector information frame | * DBFaçade (1) * VectorInformationFrame(1) | |

### Vector Information Frame

|  |  |  |
| --- | --- | --- |
| **Class Name**: VectorInformationFrame | | |
| **Superclass**: GenericFrame | | |
| **Subclasses**: None | | |
| **Description:** Displays information of a vector and can edit the information | | |
| **Private Responsibilities:**   * Get input from user | | |
| **Contract:** 17. Update Vector Information | | |
| **Contract Description:** Edits the information from the vector this frame is attributed to | | |
| **Contract Protocols:**   * update\_vector() return none   + **Purpose:** Changes the attributes of the currently selected vector and saves changes to database   + **Pre-conditions:**      - Vector is selected     - Vector is in database   + **Post-conditions:**      - Vector information is changed in database   + **Invariants:** None | | |
| **Responsibilities** | | **Collaborators** |
| * Edit vector information | | * DBFaçade (1) * Vector * VectorDatabaseFrame(1) |
| **Contract: 18.** Display Vector Information | | |
| **Contract Description:** Displays information of given vector | | |
| **Contract Protocols:**   * update\_frame(QListItem vector) return none   + **Purpose:** Sets given vector as the vector to act upon, and displays its information in the fields   + **Pre-conditions:**      - Vector is derived from currently selected vector in VectorDatabaseFrame   + **Post-conditions:**      - Vector is set as current vector to act upon     - Vector information is displayed in text fields   + **Invariants:** None | | |
| **Responsibilities** | **Collaborators** | |
| * Display list of vectors * Display vector information frame | * DBFaçade (17) * VectorInformationFrame(1) | |

### Vector Frame

|  |  |
| --- | --- |
| **Class Name**: VectorFrame | |
| **Superclass**: GenericFrame | |
| **Subclasses**: None | |
| **Description:** Organize vector information to be displayed | |
| **Private Responsibilities:**   * Get input from user * Display vector nodes, associated log entries, node relationship, and graph information | |
| **Contract:** 19. Display Vector Workflow | |
| **Contract Description:** Arrange vector information into tabs, one tab per vector | |
| **Contract Protocols:**   * populate\_tabs() return null   + **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 | * MainWindow * NodeTableFrame * GraphFrame * DBFaçade (17) |

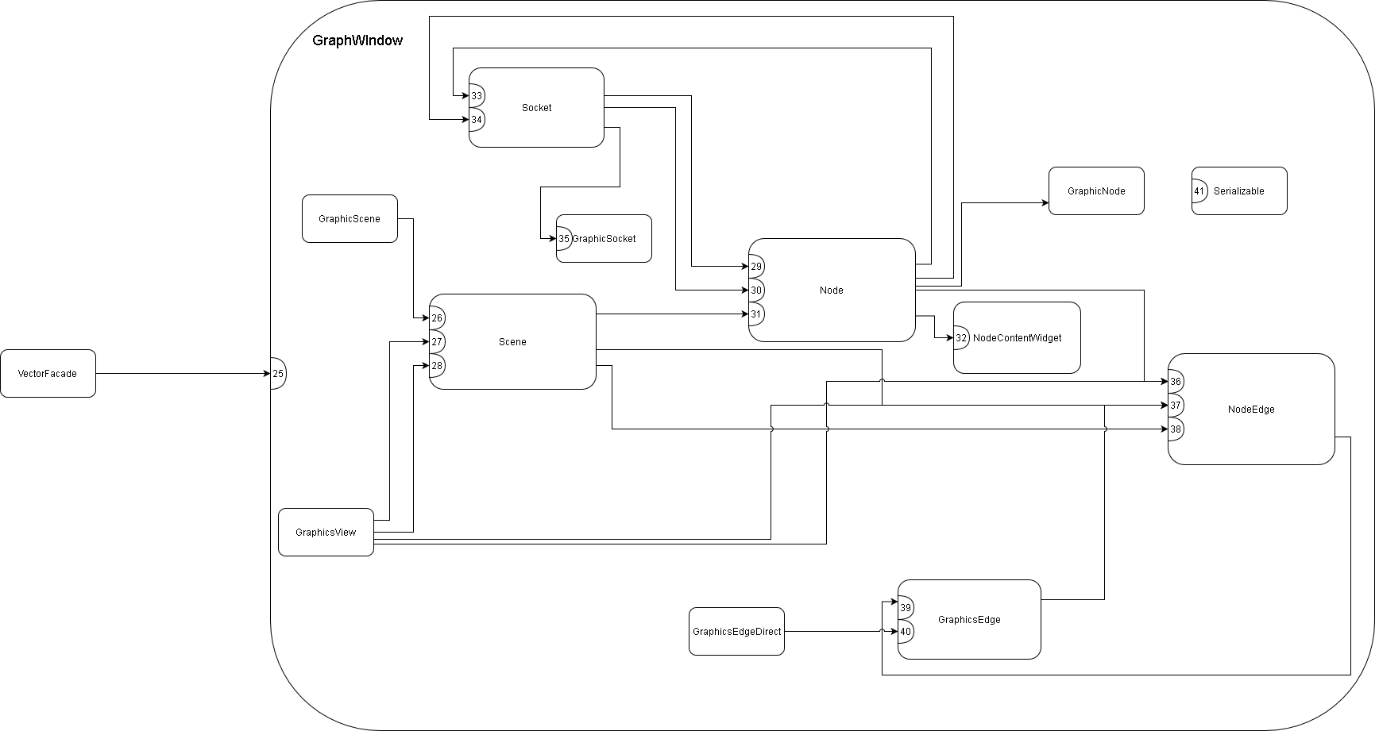
## Event Configuration Component

This component manages the event configuration model. When instantiated, if it doesn’t exist in the database already, it will take the parameters given and create one with the given information. If it is, it simply instantiates the object with the information stored.

|  |  |
| --- | --- |
| **Class Name**: Event Configuration | |
| **Superclass**: QObject | |
| **Subclasses:** None | |
| **Description:** Store event configuration details | |
| **Private Responsibilities:** None | |
| **Contract:** 20. Update Event Configuration | |
| **Contract Description:** Updates event configuration information with changes from instantiated object | |
| **Contract Protocols:**   * update() return None   + **Purpose:** Update event configuration information in database   + **Pre-conditions:**      - Event Configuration is instantiated     - Event Configuration already exists in the database   + **Post-conditions:**      - Event Configuration information is updated in database   + **Invariants:** None | |
| **Responsibilities** | **Collaborators** |
| * Update event configuration details | * Event Configuration Frame |

## Graph Subsystem

Graph section encapsulates all the files needed to create relationships and nodes in a graph in the directory titled “Graph”. The collaboration diagram of the graph will be shown here because it is treated as a component and it is complicated.



### Graph Window

|  |  |
| --- | --- |
| **Class Name:** GraphWindow | |
| **Superclass:** Qwidget | |
| **Subclass:** None | |
| **Description:** | |
| **Private Responsibilities**   * Set size of layout * Set layout * Set Scene and Graphics view | |
| **Responsibilities:** | **Collaborators:** |
|  | * Scene * GraphicsView |
| **Contract: 21.** Add Node | |
| **Contract Description:** Create a node with a dictionary | |
| **Contract Protocol:**   * addNode(self, \*\*kwargs): void   + **Purpose:** Take in a dictionary and create a node   + **Pre-conditions:** GraphWIndow needs to be instanciated, and dictionary must be passed   + **Post-conditions:**      - A node is created and displayed on the graph with the information from the dictionary   + **Invariants:** None | |
| **Responsibilities:** | **Collaborators:** |
| * Create a node | Node |

### Graphics Scene

|  |
| --- |
| **Class Name:** GraphicsScene |
| **Superclass:** QGraphicsScene |
| **Subclass:** None |
| **Description:** Creates background |
| **Private Responsibilities**   * Set graphics scene size * Draw background |

### Scene

|  |  |
| --- | --- |
| **Class Name:** Scene | |
| **Superclass:** Serializable | |
| **Subclass:** None | |
| **Description:** | |
| **Private Responsibilities**   * Keep a list of nodes and edges | |
| **Contract: 22.**Update Node and Edge lists | |
| **Contract Description:** Functions used to update list of nodes and edges | |
| **Contract Protocol:**   * addNode(self, node): void   + **Purpose:** Append node to list in class   + **Pre-conditions:** List of nodes has to exist, node exists   + **Post-conditions:** A node was appended to the node list of the class   + **Invariants:** None * removeNode(self, node): void   + **Purpose:** Remove node to list in class   + **Pre-conditions:** List of nodes has to contain the node to be deleted   + **Post-conditions:** Node specified was removed form node list   + **Invariants:** None * addEdge(self, edge): void   + **Purpose:** Append edge to list in class   + **Pre-conditions:** List of edges has to exist, edge exists   + **Post-conditions:** Specified edge was appended to the edge list of the class   + **Invariants:** None * removeEdge(self, edge): void   + **Purpose:** Remove edge from edge list   + **Pre-conditions:** Edge list has to contain the specified edge   + **Post-conditions:** Edge specified was removed   + **Invariants:** None | |
| **Responsibilities:** | **Collaborators:** |
| * Add nodes to scene node list * Remove nodes from scene node list * Add edges to scene edge list * Remove edges from edges list | GraphicsScene |
| **Contract: 23.**Save To File | |
| **Contract Description:** Create file to save dictionary of the information of edges and their relationships and nodes and their content. | |
| **Contract Protocol:**   * saveToFile(self, filename): void   + **Purpose:** Write to filename a dictionary with node and edge information   + **Pre-conditions:** Graph needs to be running.   + **Post-conditions:**     - A file in the directory gets created with the filename provided and all the information in nodes and edges   + **Invariants:** None | |
| **Responsibilities:** | **Collaborators:** |
| * Save scene information, nodes and edges | GraphicsView |
| **Contract: 24.**Serialize | |
| **Contract Description:** Returns information from the scene, nodes and edges as dictionary | |
| **Contract Protocol:**   * serieialize(self): return dictionary   + **Purpose:** Return as dictionary information from the scene to be saved into a file   + **Pre-conditions:** Graph needs to be running and crtl+s keys need to be pressed   + **Post-conditions:** Dictionary with the following will be returned:     - **Id**     - **Scene\_width**     - **Scene\_height**     - **Nodes**     - **edges**   + **Invariants:** None | |
| **Responsibilities:** | **Collaborators:** |
| * Format information for saving | GraphicsView |

### Graphics View

|  |  |
| --- | --- |
| **Class Name:** GraphicsView | |
| **Superclass:** QGraphicsView | |
| **Subclass:** None | |
| **Description:** Gives the ability to give movement logic to nodes and objects and constantly updating screen | |
| **Private Responsibilities**   * Set graphic scene * Setting policy to hide scroll bars and rendering images smoother * Handle default click and move event * Know what graphic item is being clicked or selected * Know what keys are being pressed * Click and drag ability with middle button on graph | |
| **Responsibilities:** | **Collaborators:** |
|  | * Graphics Node * Edge * GraphicSocket * Scene |

### Node

|  |  |
| --- | --- |
| **Class Name:** Node | |
| **Superclass:** Serializable | |
| **Subclass:** None | |
| **Description:** None | |
| **Private Responsibilities**   * Know scene * Know Information of node | |
| **Contract: 25.**Get Socekt Position | |
| **Contract Description:** Return socket position relative to node | |
| **Contract Protocol:**   * getSocketPosition(self, socket\_x\_position, socket\_y\_position): return [x, y]   + **Purpose:** Set socket position relative to node depending if socket is Top, or Bottom   + **Pre-conditions:** Node has to be created   + **Post-conditions:** Position of socket is returned as a list   + **Invariants:** None | |
| **Responsibilities:** | **Collaborators:** |
|  | * NodeEdge * Socket |
| **Contract: 26.**Update Connected Edges | |
| **Contract Description:** None | |
| **Contract Protocol:**   * updateConnectedEdges(self):   + **Purpose:** Update coordinate position of where edge is connected   + **Pre-conditions:** Edge needs to be connected to a socket   + **Post-conditions:** Coordinate position of Edge gets updated   + **Invariants:** None | |
| **Responsibilities:** | **Collaborators:** |
| * None | Socket |
| **Contract: 27.**Serialize | |
| **Contract Description:** Create dictionary o f node information | |
| **Contract Protocol:**   * serialize(self): return dictionary   + **Purpose:** Format node information in dictionary   + **Pre-conditions:** Node has to exist   + **Post-conditions:** Dictionary with following informationis returned:     - **Id**     - **Name**     - **Pos x**     - **Pos y**     - **Parent**     - **Child**     - **Content**   + **Invariants:** None | |
| **Responsibilities:** | **Collaborators:** |
| * Know node position, id, contents, parent and child | Scene |

### Graphic Node

|  |  |
| --- | --- |
| **Class Name:** GraphicNode | |
| **Superclass:** QGraphicsItem | |
| **Subclass:** None | |
| **Description:** Creation of a node graphically | |
| **Private Responsibilities**   * Set dimetions of a node * Overwrite paint function to draw node * Set properties of movement of the node | |
| **Responsibilities:** | **Collaborators:**  Node |

### Node Content Widget

|  |  |
| --- | --- |
| **Class Name:** NodeContentWidget | |
| **Superclass:** QWidget, Serializable | |
| **Subclass:** None | |
| **Description:** Display node information from nodes dictionary | |
| **Private Responsibilities**   * Create labels and edit lines * Set layout * Display information of a node | |
| **Responsibilities:** | **Collaborators:** |
|  | Node |
| **Contract: 28.**Node Information | |
| **Contract Description:** Funtion when called to return a dictionary of the node’s content | |
| **Contract Protocol:**   * serialize(self): dictionary   + **Purpose:** Return dictionary of the information displayed of the node   + **Pre-conditions:** Node exists with information   + **Post-conditions:** Dictionary with information of node is returned   + **Invariants:** None | |
| **Responsibilities:** | **Collaborators:** |
|  | Node |

### Socket

|  |  |
| --- | --- |
| **Class Name:** Socket | |
| **Superclass:** Serializable | |
| **Subclass:** None | |
| **Description:** | |
| **Private Responsibilities**   * Create graphic socket * Keep track if socket has an edge | |
| **Contract: 29.**Get Socket Position | |
| **Contract Description:** Get socket position with x and y coordinates within a node | |
| **Contract Protocol:**   * getSocketPositon(self): [x, y]   + **Purpose:** Get socket position   + **Pre-conditions:** Node has to exist   + **Post-conditions:** List comprised of x and y coordinates of a socket on a node   + **Invariants:** None | |
| **Responsibilities:** | **Collaborators:** |
|  | Node |
| **Contract: 30.**Serialize | |
| **Contract Description:** When called return information of a socket as a dictionary | |
| **Contract Protocol:**   * serialize(self): dictionary   + **Purpose:** Return information for sockets of a node as a dictionary   + **Pre-conditions:** Node has to exist   + **Post-conditions:** Dictionary with information of nodes socket is returned in a dictionary   + **Invariants:** None | |
| **Responsibilities:** | **Collaborators:** |
|  | Node |

### Graphic Socket

|  |  |
| --- | --- |
| **Class Name:** GraphicsSocket | |
| **Superclass:** QGraphicsItem | |
| **Subclass:** None | |
| **Description:** Set settings and draw graphic socket on node | |
| **Private Responsibilities**   * Graphically draw socket * Set bounding rectangle to socket | |
| **Responsibilities:** | **Collaborators:** |
|  | Socket |
| **Contract: 31.**Set Socket Position | |
| **Contract Description:** Over writes parent function to set position to a QGraphicItem | |
| **Contract Protocol:**   * setPos(\*[x, y]): void   + **Purpose:** Set position within node to graphically draw socket   + **Pre-conditions:** Graphics socket is instantiated   + **Post-conditions:** Socket is drawn within a node in the positions specified   + **Invariants:** | |
| **Responsibilities:** | **Collaborators:** |
|  | Socket |

### Node Edge

|  |  |
| --- | --- |
| **Class Name:** NodeEdge | |
| **Superclass:** Serializable | |
| **Subclass:** None | |
| **Description:** Calculate coordinate positions for edges to draw delete and update socket and edges | |
| **Private Responsibilities**   * Know label * Create GraphicSocket | |
| **Contract: 32.**Update Positions | |
| **Contract Description:** Depending on socket position of the node, set coordinates for edge to start drawing and stop | |
| **Contract Protocol:**   * updatePositions(self)   + **Purpose:** Update coordinate position of socket to draw edge form position of socket on graphic node   + **Pre-conditions:** Node needs to exist, click and drag on socket   + **Post-conditions:** Coordinates get updated   + **Invariants:** None | |
| **Responsibilities:** | **Collaborators:** |
|  | * GraphicsView * Node |
| **Contract: 33.**Remove socket/edge | |
| **Contract Description:** Remove socket information from the edge and delete edge | |
| **Contract Protocol:**   * remove\_from\_socket(self): void   + **Purpose:** Make socket None   + **Pre-conditions:** There needs to an established edge from socket   + **Post-conditions:** information from edge was removed from socket   + **Invariants:** None * remove(self): void   + **Purpose:** rRemove edge from socket, from scenem and make None   + **Pre-conditions:** There needs to be an established edge from socket   + **Post-conditions:** Edge is removed graphically and socket and edge information updated   + **Invariants:** None | |
| **Responsibilities:** | **Collaborators:** |
|  | * Scene * GraphicsEge * GraphicsView |
| **Contract: 34.**Serialize | |
| **Contract Description:** Information from the class instance get put on a dictionary | |
| **Contract Protocol:**   * serialize(self): return Dictionary   + **Purpose:** Return edge information in dictionary format   + **Pre-conditions:** Edge needs to exist   + **Post-conditions:** Dicrionary with the following information is returned:     - **Id**     - **Label**     - **Start**     - **End**   + **Invariants:** None | |
| **Responsibilities:** | **Collaborators:** |
|  | Scene |

### Graphics Edge

|  |  |
| --- | --- |
| **Class Name:** GraphicsEdge | |
| **Superclass:** QGraphicsPathItem | |
| **Subclass:** GraphicsEdgeDirect | |
| **Description:** | |
| **Private Responsibilities**   * Set settings for drawing line * Overwirtes mpaint method | |
| **Contract: 35.**Source and destination coordinates | |
| **Contract Description:** Sets position source of where to begin to draw a line to the end position | |
| **Contract Protocol:**   * setSource(self, x, y): void   + **Purpose:** Set coordinate points to start drawing an edge   + **Pre-conditions:** A socket needs to exist, be clicked, and dragged   + **Post-conditions:** Variable is set with list containing x, y   + **Invariants:** None * setDestination(self, x, y)   + **Purpose:** Set coordinate point to finish drawing an edge   + **Pre-conditions:** A socket must have been clicked and dragged   + **Post-conditions:** Variable is set with list containing x, y   + **Invariants:** None | |
| **Responsibilities:** | **Collaborators:** |
|  | NodeEdge |
| **Contract**: 36.Update Path to paint | |
| **Contract Description:** Set coordinate position to start drawing an edge and set ending coordinate position | |
| **Contract Protocol:**   * updatePath(self): raise NotImplemented   + **Purpose:** Set style of line to draw by making it abstract   + **Pre-conditions:** Coordinates for starting and ending have to be set, needs to be overwritten   + **Post-conditions:** Edge is drawn with the style implemented form the coordinates given   + **Invariants:** None | |
| **Responsibilities:** | **Collaborators:** |
| * Draw Edges form coordinates | GraphicEdgeDirect |

### Graphics Edge Direct

|  |  |
| --- | --- |
| **Class Name:** GraphicsEdgeDirect | |
| **Superclass:** GraphicsEdge | |
| **Subclass:** None | |
| **Description:** Sets type of edge to draw from source to destination coordinates | |
| **Private Responsibilities**   * Sets type of line to draw from staring to ending coordinates | |
| **Responsibilities:** | **Collaborators:** |
| * Know edge type * Know Coordinates | GraphicsEdge |

### Serializable

|  |  |
| --- | --- |
| **Class Name:** Serializable | |
| **Superclass:** None | |
| **Subclass:** Scene, Node, Node Edge, Socket, NodeContentWidget | |
| **Description:** Abstract class need to implement serialize | |
| **Private Responsibilities**   * Know id | |
| **Contract: 37.**Serialize Parent | |
| **Contract Description:** Serialize function needs to be implemented by children | |
| **Contract Protocol:**   * serialize(self): raise NotImplemented   + **Purpose:** Abstract to be overwritten by children   + **Pre-conditions:** Class needs to be inherited   + **Post-conditions:** Children will return different types of dictionaries   + **Invariants:** None | |
| **Responsibilities:** | **Collaborators:** |
|  | * Scene * Node * NodeEdge * Socket * NodeCOntentWidget |

## Vector Façade Component

|  |  |
| --- | --- |
| **Class Name:** VectorFacade | |
| **Superclass:** None | |
| **Subclass:** None | |
| **Description:** Class that initializes a graph for a vector | |
| **Private Responsibilities**   * Know name and description * Create graph with same name and description | |
| **Responsibilities:** | **Collaborators:** |
| * Create Graph | GraphWindow |

## Vector Component

This component manages the vector model.

|  |  |
| --- | --- |
| **Class Name**: Vector | |
| **Superclass**: QObject | |
| **Subclasses:** None | |
| **Description:** Store vector information | |
| **Private Responsibilities:** None | |
| **Contract:** 38. Manage Vector | |
| **Contract Description:** Adds a vector or updates a vector in the database | |
| **Contract Protocols:**   * update() return None   + **Purpose:** Update vector information in database   + **Pre-conditions:**      - Vector is instantiated     - Vector already exists in the database   + **Post-conditions:**      - Vector information is updated in database   + **Invariants:** None * add() return None   + **Purpose:** Adds an instantiated vector into the database   + **Pre-conditions:**      - Vector is instantiated   + **Post-conditions:**      - Vector is added to the database with a unique name   + **Invariants:** None | |
| **Responsibilities** | **Collaborators** |
| * Create vector * Update vector | * Vector Frame * Graph Frame * Vector Information Frame * Vector Database Frame |

# 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 configuration 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 2 demonstrates the structure of the information stored within the database. The way the data is stored is through three collections, “Event Configuration”, “Vector” and “Graph.” When the system creates an Event Configuration, Vector, Graph, the database saves the objects in the database and a document ID gets assigned to each object. That ID is then used to pull the objects to the system and display the contents currently from the DB. Once the user makes any updates within those objects in the system the updates are reflected in the database.

The “Vector” collection contains documents comprised of a document ID, name, description, a graph ID and its significant log entries. The graph ID stores the reference of its corresponding graph, and the significant log entries subdocument contains a list of all the significant log entries associated to that vector.

The “Graph” collection contains documents compromised of a document ID, export format, orientation, interval units, interval, a list of nodes within the graph and a list of relationships.

The “Event Configuration” collection contains documents compromised of a document ID, event name, description, event start and end time, root directory, red, blue, white team folders, the lead, lead IP address, and connection status.

A screenshot of a cell phone

Description automatically generated

**Figure 2:** MongoDB Document Schema Design

## %