Team Admin

Student Reports and Statistics

(SRAS)

Software Design Document

Prepared by:
Mai Nguyen
Whitner Reichman
Judy Gonzalez

Release Date
March 21, 2023

Table of Contents

Executive Summary

Student Reports and Statistics (SRAS) is an anonymous reporting and tracking system designed for college students. SRAS aims to provide transparent information and report tracking for students that is lacking in the annual safety reports conducted by Campus Safety, as well as the anonymous (and non-anonymous) support systems in place at the Title IX office. With SRAS, students can file a report anonymously, access school-wide data, and re-access every report they have submitted.

This document provides technical information regarding the software design of SRAS.

Document Versioning

Date	Owner	Comment
03/15/2023	Judy Gonzalez	Initial text and rough draft started. Updated the overall feature discussion.
03/15/2023	Mai Nguyen	Updated Feature Matrix BRD ID, feature discussion, and diagram. Removed Home component and added GUI component.
03/15/2023	Whitner Reichman	Updated Statistics, and Admin Access.
03/21/2023	Mai Nguyen	Added Report Component Command Design Pattern and Filter Component Decorator Design Pattern. Updated high architecture diagram.
03/21/2023	Judy Gonzalez	Updated the overall document.
03/21/2023	Whitner Reichman	Overall document check + Filter component (Identitytxt)

Feature Matrix

The feature matrix enumerates the technical features required to support each of the business requirements. The discussion section provides details regarding the constraints and functionality of the feature. The ids are used for traceability. Features that can be removed should strike-through the feature id and have a comment added to identify why this feature can be removed without impacting the BRD requested functionality.

Priority Codes:

H - High, a must have feature for the product to be viable and must be present for launch
M - Medium, a strongly desirable feature but product could launch without
L - Low, a feature that could be dropped if needed

ID	Pri	Feature Name	Comment	BRD ID
P.1	H	GUI	Java Swing	C.1, D.2, E.1, E.4
P.2	M	Action (submit, cancel, return home, etc.)		C.1, D.2, D.3, D.4, D.8, D.10, D.11
P.3	H	Language	Implementation in Java	C.1, C.2, C.3, T.1, D.1, D.2, D.3, D.4, D.5, D.6, D.7, D.8, D.9, D.10, D.11, D.12, D.13, E.1, E.2, E.3, E.4
L.1	L	Error Handling		E.1
L.2	M	Save to Database		D.1, D.2, D.3, D.5, D.6, D.7, D.8, D.9, D.10, D.12, E.2, T.2
M.1	M	~~View Database~~	Did not implement due to time constraint	D.3, D.8, T.2
M.2	M	Remove From Database		D.4, T.2
M.3	H	Generate ID		E.3
G.1	H	Generate Report		D.2, E.3, D.1, D.5, D.6, D.7, D.8 D.9, D.13, D.14
G.2	L	Filtered Search		E.4, D.5, D.6, D.8, T.2

Feature Discussion

P.1 - GUI

SRAS' anonymous reporting system has simple user interactions which can be easily operated in a GUI window. SRAS provides the GUI for navigating the application easily by directly interacting with clickable objects and text fields.

P.2 - Action

The program must support different actions taken within the GUI, such as going to the next page, canceling a report, returning home, and submitting a report. These actions are available to the user by prompts, buttons, and text fields so that they may use the application smoothly.

P.3 - Language

SRAS' must be usable with a number of different devices. Therefore, our group has decided to use JAVA instead of Python, JavaScript, C++, etc. because it has cross-platform support, meaning that it is able to move easily from one computer system to another once the code is written. This supports consistent GUI behavior across different platforms (i.e. Windows, Mac, and Linux).

L.1 - Error Handling

If a user inputs an unknown response when filling out a report or filtering through statistics, an error message will be displayed to inform the user that an error has occurred. The user is then given a message to try again with one of the available input options.

L.2 - Save to Database

An action initiated by the 'submit' button, that reads the user's answers and saves them to our SQL database, leaving blanks where fields are empty.

~~M.1 - View Database~~

~~'View Database' is a feature accessible via the 'Access Report' page. There will be an 'Admin' button somewhere on the page and will only display the database of reports up to the most recent entry, with all fields of text and multiple choice included, if a user has an entry ID when prompted to type one in.~~

M.2 - Remove from Database

The 'Remove from Database' feature allows for the user to delete a report if they decide to. A user can do this by re-accessing it and clicking the 'delete report' button. It will then be deleted from the database.

M.3 - Generate ID

When a user submits a report, the program will generate a unique ID for that report. The ID may consist of a combination of numbers. The ID will be associated with the report in the database system, along with the other relevant information such as the date, mental health impact, graduating class, and other optional information. To access a report, the user can select the ‘Access Report’ option from the main menu, and then enter the ID when prompted. The program will search for the report with the corresponding ID in the database, and then display the report information to the user.

G.1 - Generate Report

When the user selects the ‘File a Report’ option from the home page, the program opens up a new window where it prompts the user to enter information about the incident. The information that the user is prompted to enter includes the required and optional information. The submitted report will then be saved in the database.

G.2 - Filtered Search

The filtered search feature works by accessing the data in the program's database, and then applying the selected filters to the data. The filters may include options such as selecting a specific date range, location, or graduating class. Once the filters are selected, the program will display the relevant data based on the criteria.

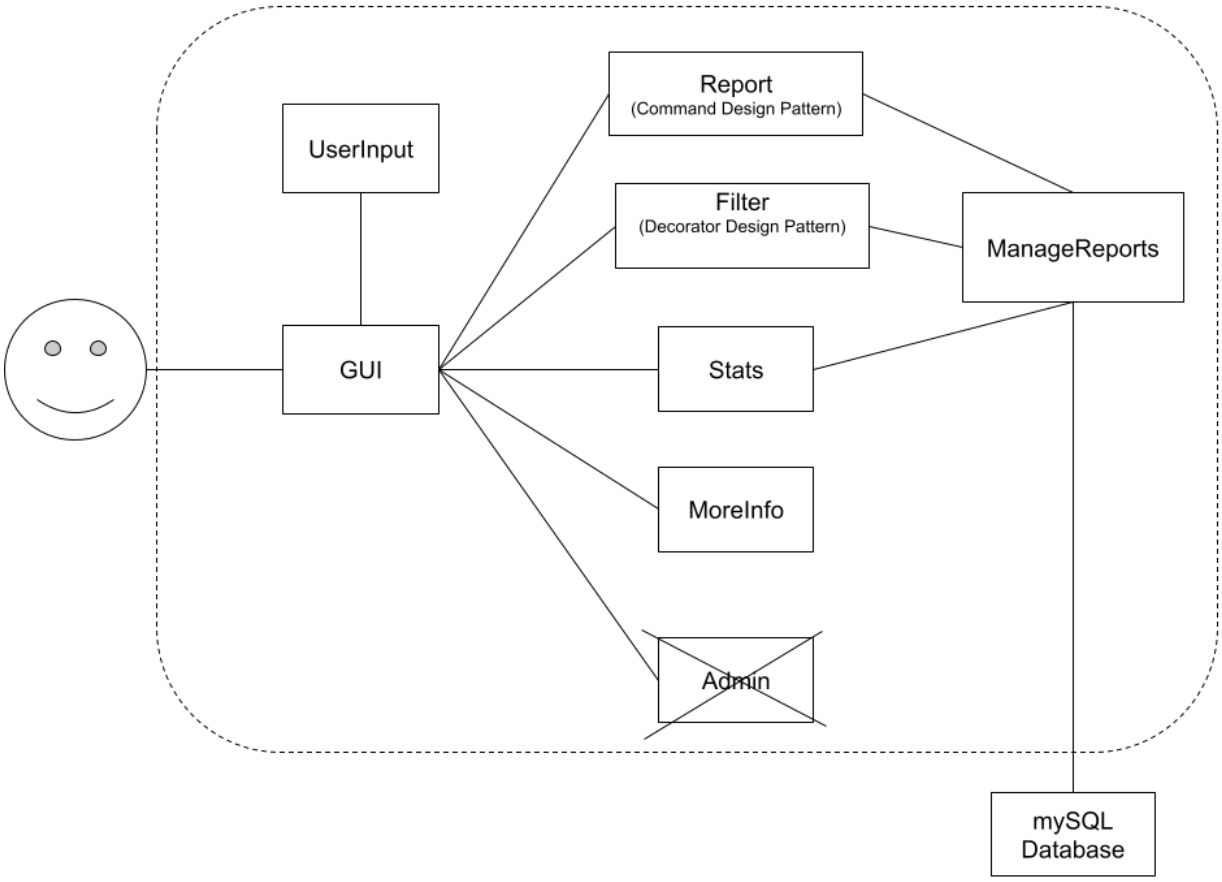
System Design

This section describes the system design in detail. An overview of the major system components and their roles will be provided, followed by more detailed discussion of components. Component discussion will reference the technical features the component helps satisfy.

Note: Design co-evolves with implementation. It is important to start with some sense of components and their interactions, but design docs should be updated during implementation to capture the actual system as-built.

Architecture Overview

High-level System Architecture



High-level Architecture Diagram

SRAS interacts with the host computer. The user and data file appear outside of the system boundary for SRAS. Inside of the SRAS system boundary are several components that interact with one another.

Major Components

GUI

P.1, P.2
The GUI component is the graphical representation of our program that the user will see and interact with after running our program. It allows user-friendly navigation where users can file a report, view statistics, get more information on SRAS, view additional resources, or re-access their report.

UserInput

P.2, L.1
The UserInput component processes the user input when filing or updating their report. It will support different scenarios like entering numbers or text, and it will also have error handling in case of user error. It will provide error messages that prevents the user from submitting or updating their report until the minimum requirements are met and provided information is in the expected format.

Report

G.1, P.2, L.2, M.1, M.3
The Report component is responsible for having a report object that has attributes pertaining to an incident reported by the user. It is also responsible for allowing the user to file a report, save a report to the database, access an existing report, update an existing report, and delete an existing report.

This component uses the Command Design Pattern to execute actions on the report, such as filing report, saving report, updating report, accessing report, and deleting report.

ManageReports

L.2, M.1, M.2
The ManageReports component is a connecting class that helps store and retrieve information from our SQL database.

Stats

P.2, M.1
The Stats component is responsible for showing all of the current statistics and an option for the user to filter through those statistics.

Filter

G.2
The Filter component is responsible for filtering the statistics based off of date, class, identity impact and/or identity text. The user specifies the criteria they want to filter by.

This component will use the Decorator Design Pattern to stack the criteria a user is filtering by.

MoreInfo

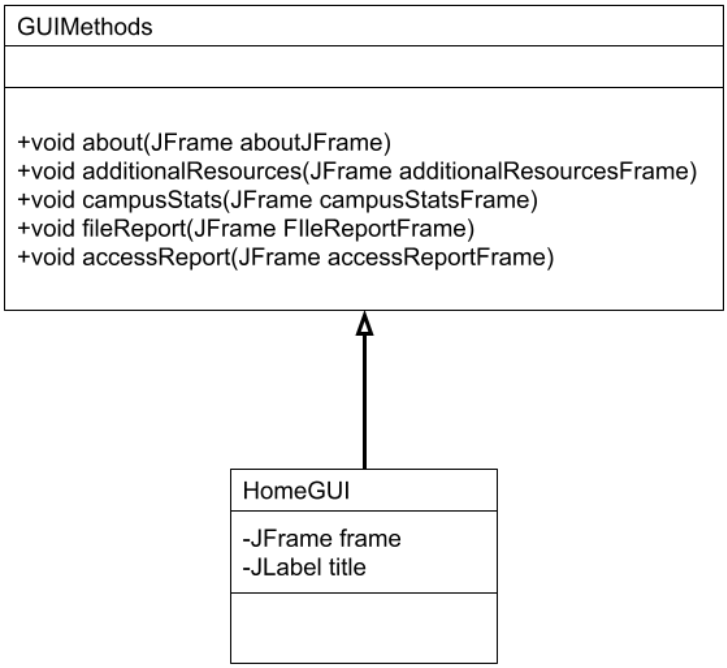
P.2
The MoreInfo component is responsible for displaying the ‘additional resources’ information available for a user (i.e. Campus Safety, the Title IV Office, the Counseling Center, etc.) or the ‘about’ information for our program.

~~Admin~~

~~M.1~~
~~The Admin Component is responsible for viewing all of the data recorded in the system. It should be available via the ‘Access Report’ Page using a specialized user ID.~~

Detail Design

GUI Component



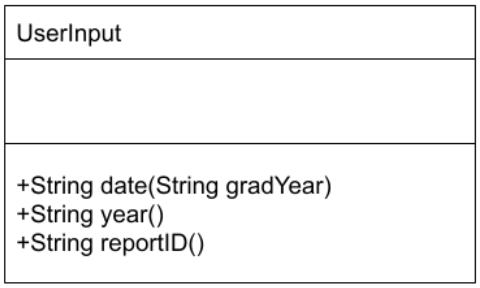
GUIMethods

GUIMethods contains methods that display different JFrames, one at a time, to graphically represent different pages of the application. Each frame is a page to be accessed from the home page menu. Each method displays a new frame with a specified size and layout manager, setting up the empty frame.

HomeGUI

HomeGUI encapsulates the logic for the entire GUI side of the application. The class provides a user-friendly interface to interact with the underlying system, which makes it easier for users to navigate the application's functionalities. It extends GUIMethods in order to use its methods to display the frame of each page of the application.

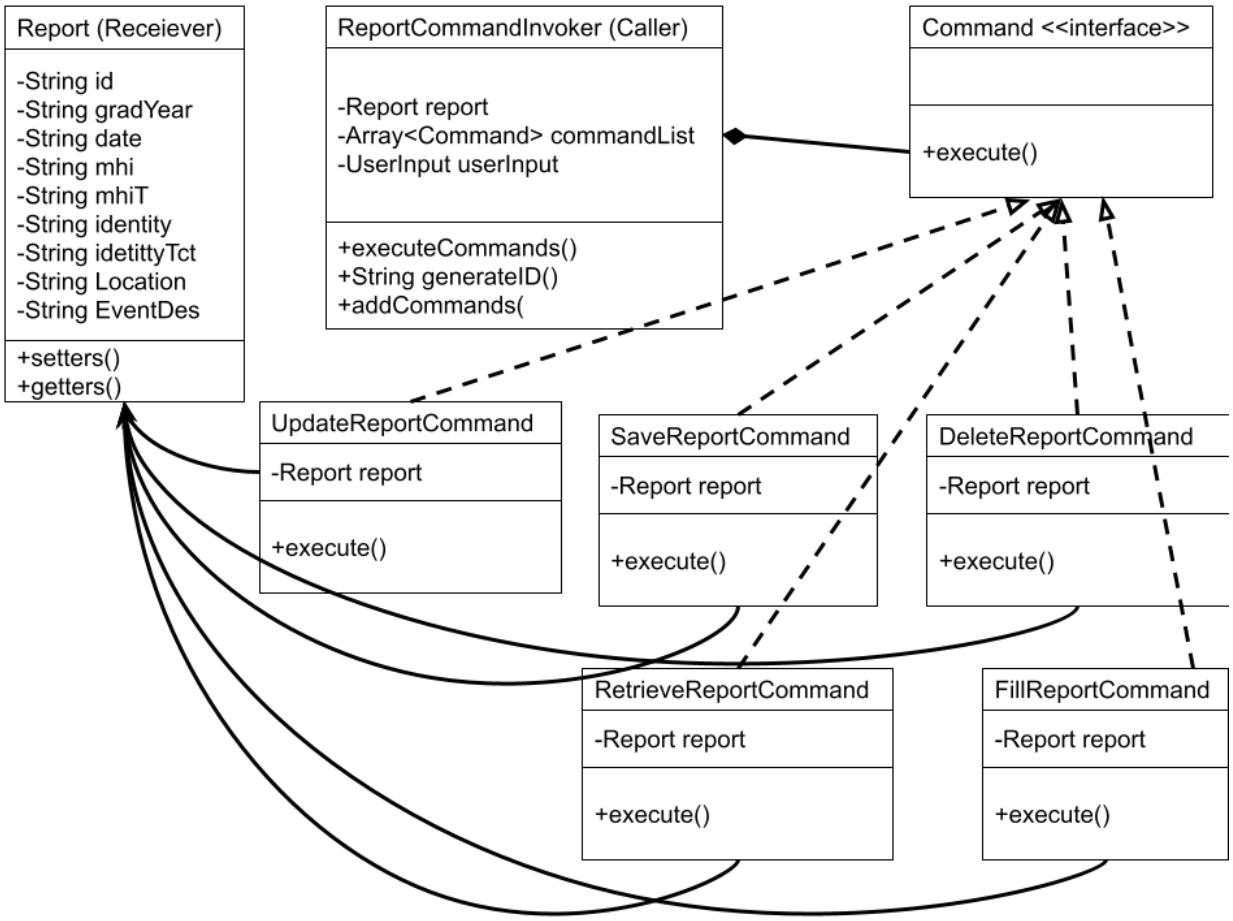
UserInput Component



UserInput

UserInput encapsulates the logic to ask for user input and make sure that the user enters prompted information in the correct format. Whenever user input is required, a method from this class is called corresponding to the prompt. If a user provides input in a format that is not expected, then an error message appears to provide the correct format and the user is put into a loop until the expected format is provided.

Report Component (Command Design Pattern)



Report

A report is the receiver in the command design pattern. It is made up of various information, which will be all String objects, and can be set and get by other concrete commands to get and set report information/attributes.

ReportCommandInvoker

ReportCommandInvoker is the caller in the command design pattern. It encapsulates the logic to add different concrete report commands into a command list and execute all of them by using the Command interface execute method. This class is used in homeGUI to file a new report, access a report, update a report, and delete a report.

Command

Interface class for commands that can be done to a report, all of its subclasses will implement the execute method.

SaveReportCommand

Implements the Command interface. It has an execute method that takes in a completed report and saves it to our SQL database.

DeleteReportCommand

Implements the Command interface. It has an execute method that takes in a completed report and deletes it from our SQL database.

RetrieveReportCommand

Implements the Command interface. It has an execute method that takes in a report ID, retrieves the corresponding report from our SQL database, and fills in a report with the information from our SQL database by joining two of our tables (i.e. ShortAnswers and LongAnswers).

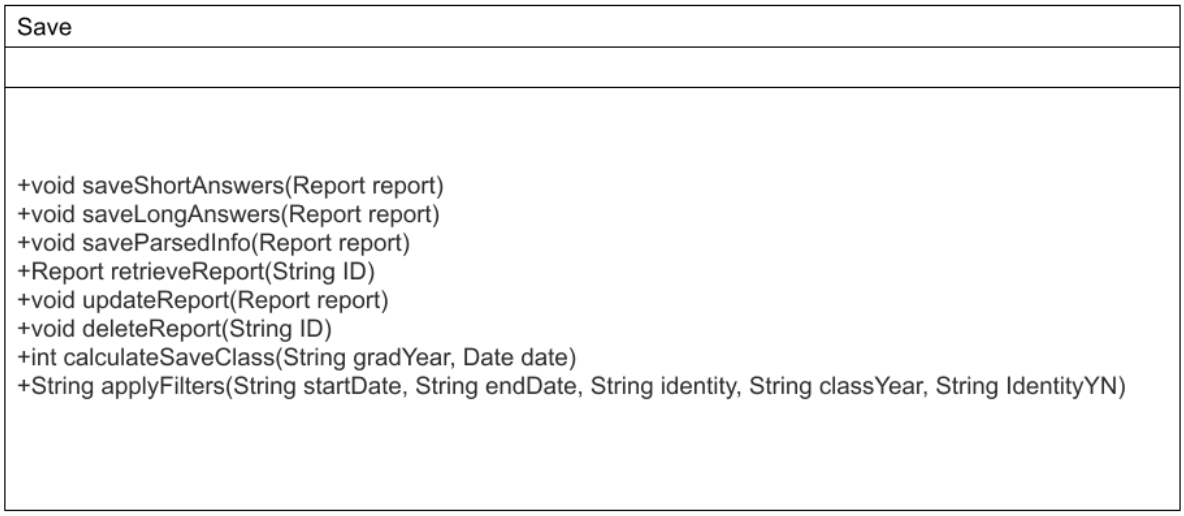
FillReportCommand

Implements the Command interface. It has an execute method that takes in all user inputs for report fields and fills out the report object attribute with the entered information.

UpdateReportCommand

Implements the Command interface. It has an execute method that takes in a completed report and deletes it from our SQL database.

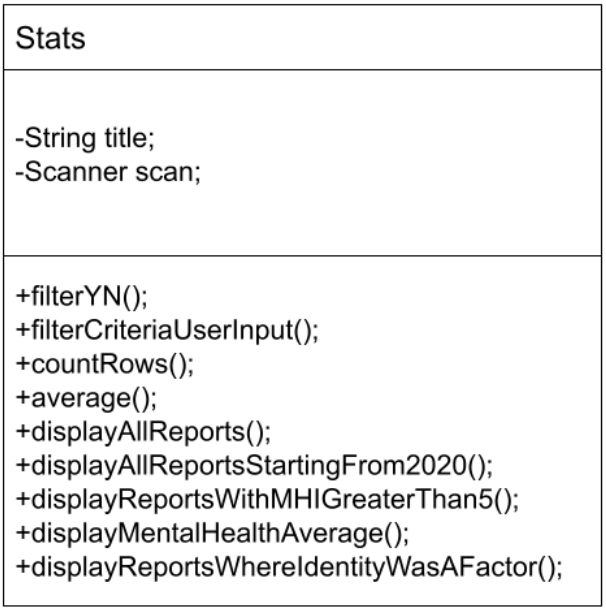
ManageReports Component



Save

This class has methods that act as data manager, interacting with our SQL database system. It contains a method to save a report to the database, retrieve report from the database using Report ID as the primary key, update report in the database, and delete report using Report ID as the primary key. It also take has a method that forms one mysql query to retrieve filtered data.

Stats Component



Stats

Stats is a class that will display default campus statistics and ask the user if they wish to further filter through our reports. They will then have the option to type in one of three filter options (i.e. date, class, identity, and identity text). It will then get the filtered statistics. There should also be additional GUI button selections for additional features based on the filtering classes.

Filter Component (Decorator Design Pattern)

Filter

Filter is the common interface that will be used by the specified Filter classes.

BaseFilter

BaseFilter class serves as the base class for all filters. It starts the mysql query syntax by selecting all from 2 tables: ParsedInfo table and ShortAnswers table. If no other filter is added on, then the query will return all data in the database.

AddOns

AddOns is an abstract class serves as the base class for all concrete filter decorator classes. It implements the same interface as BaseFilter, in which it decorates.

DateFilter

DateFilter is a class that will override what was stated in the Filter Interface to specifically filter through start date and end date of the user's input. It adds on to the base query or existing Filter type query by adding mysql syntax to the query to filter statistics between specific dates.

ClassFilter

ClassFilter is a class that will override what was stated in the Filter Interface to specifically filter through the user's input. It adds on to the base query or existing Filter type query by adding mysql syntax to the query to filter statistics with a given class.

IdentityYNFilter

IdentityYNFilter is a class that will override what was stated in the Filter Interface to specifically filter the user's input. It adds on to the base query or existing Filter type query by adding mysql syntax to the query to filter statistics with a given option for identity yes/no fields.

IdentityTxtFilter

IdentityTxtFilter is a class that will override what was stated in the Filter Interface to specifically filter through the user's input. It adds on to the base query or existing Filter type query by adding mysql syntax to the query to filter statistics by user text input.

MoreInfo Component

MoreInfo

MoreInfo is an abstract class that holds the basic information used in our 'additional resources' and 'about' pages. String attributes were assigned to the various headers and descriptions since that is how users will be able to read through them.

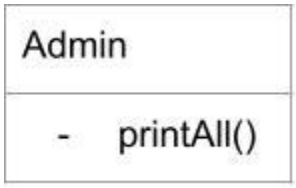
Additional Resources

Additional Resources extends MoreInfo and overwrites its original method. It should only display additional resources available for students.

About

About extends MoreInfo and overwrites its original method. It should only display general information about our program.

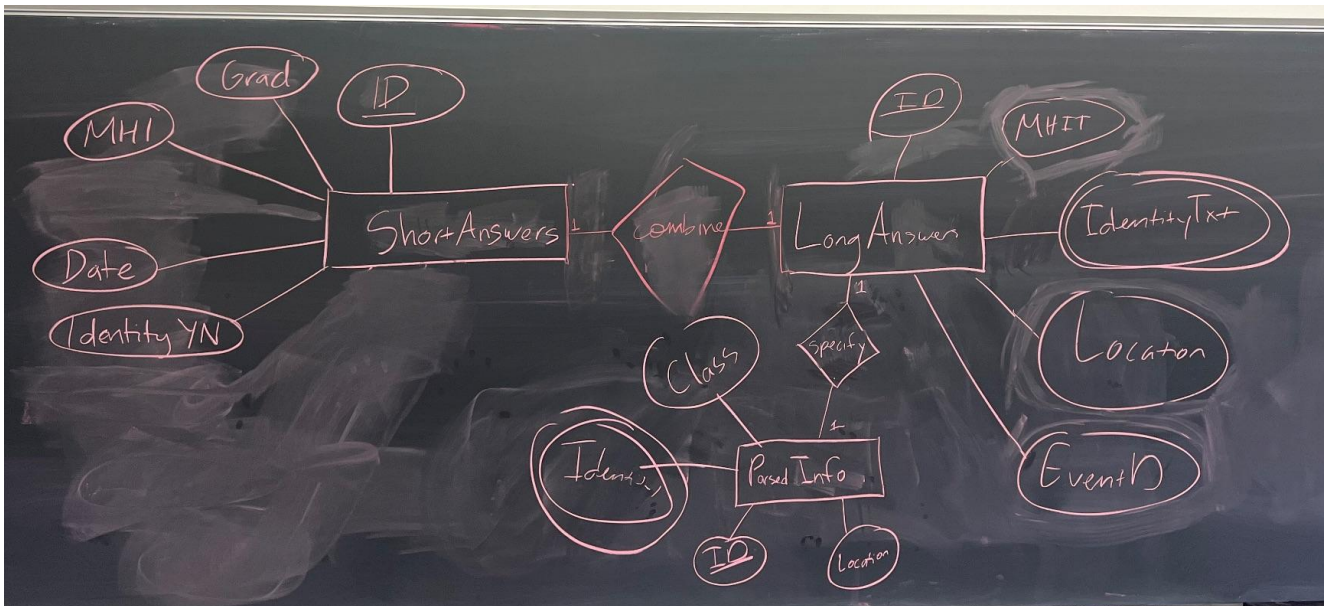
~~Admin Component~~



~~Admin~~

~~The Admin class is relatively simple, it contains a method to print all of the information stored in the csv document.~~

Entity-Relationship Schema diagrams



Snippets of database tables

```
mysql> show tables;
+----------------+
| Tables_in_sras |
+----------------+
| LongAnswers    |
| ParsedInfo     |
| ShortAnswers   |
+----------------+
3 rows in set (0.00 sec)

mysql> describe LongAnswers;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| ID          | varchar(10)  | NO   | PRI | NULL    |       |
| MHIT        | varchar(255) | YES  |     | NULL    |       |
| IdentityTxt | varchar(255) | YES  |     | NULL    |       |
| Location    | varchar(255) | YES  |     | NULL    |       |
| EventD      | varchar(255) | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)

mysql> describe ShortAnswers;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| ID         | varchar(10) | NO   | PRI | NULL    |       |
| Grad       | varchar(4)  | YES  |     | NULL    |       |
| MHI        | varchar(2)  | YES  |     | NULL    |       |
| Date       | date        | YES  |     | NULL    |       |
| IdentityYN | varchar(3)  | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)

mysql> describe ParsedInfo;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| ID       | varchar(10)  | NO   | PRI | NULL    |       |
| Location | varchar(255) | YES  |     | NULL    |       |
| Identity | varchar(255) | YES  |     | NULL    |       |
| Class    | int(11)      | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

```
mysql> select * from LongAnswers;
+------------+-------------+---------------+-------------+--------+
| ID         | MHIT        | IdentityTxt   | Location    | EventD |
+------------+-------------+---------------+-------------+--------+
| 8SXV92H56U | hello world |               |             |        |
| B9PLORLG0J |             |               |             |        |
| E92ZS7Q5P5 |             |               | worner      |        |
| N5GMVTRXI5 |             |               |             |        |
| O2GT841IBA |             | female        |             |        |
| O2S76356AF |             |               | hello world |        |
| Q93JP3V3S1 |             |               |             |        |
| QZAG2OOXDV |             |               |             |        |
| W2CB59GGTM |             |               |             |        |
| YJ2SGYUKO7 |             | female, asian |             |        |
+------------+-------------+---------------+-------------+--------+
10 rows in set (0.00 sec)

mysql> select * from ShortAnswers;
+------------+------+------+------------+------------+
| ID         | Grad | MHI  | Date       | IdentityYN |
+------------+------+------+------------+------------+
| 8SXV92H56U | 2024 | 9    | 2023-01-01 | Yes        |
| B9PLORLG0J | 2021 | 2    | 2020-01-01 | Yes        |
| E92ZS7Q5P5 | 2024 | 3    | 2020-01-01 | Yes        |
| N5GMVTRXI5 | 2024 | 7    | 2023-01-01 |            |
| O2GT841IBA | 2020 | 3    | 2021-01-20 |            |
| O2S76356AF | 2020 | 3    | 2020-01-01 |            |
| Q93JP3V3S1 | 2020 | 3    | 2020-01-01 |            |
| QZAG2OOXDV | 2024 | 3    | 2023-01-01 |            |
| W2CB59GGTM | 2021 | 2    | 2020-01-01 | Yes        |
| YJ2SGYUKO7 | 2024 | 2    | 2022-01-01 |            |
+------------+------+------+------------+------------+
10 rows in set (0.00 sec)

mysql> select * from ParsedInfo;
+------------+----------+--------------+-------+
| ID         | Location | Identity     | Class |
+------------+----------+--------------+-------+
| 8SXV92H56U | other    |              |     5 |
| B9PLORLG0J | other    |              |     1 |
| E92ZS7Q5P5 | worner   |              |  NULL |
| N5GMVTRXI5 | other    |              |     5 |
| O2GT841IBA | other    | Female       |  NULL |
| O2S76356AF | other    | TEST         |  NULL |
| Q93JP3V3S1 | other    |              |     0 |
| QZAG2OOXDV | other    |              |     1 |
| W2CB59GGTM | other    |              |     1 |
| YJ2SGYUKO7 | other    | Female Asian |  NULL |
+------------+----------+--------------+-------+
10 rows in set (0.00 sec)
```