**HOW THE CODE WORKS**

**DATABASES**

**CSVReader**

CSVReader is used to read and parse through the data from a CSV file. It is used primarily in the MainWindow.

**QSortListIO**

QSortList is used to read and write data from and into the file. It is used primarily in the MainWindow.

**RecordsManager**

The RecordsManager is responsible for creating records using data from the CSV files. These records are used by the system to create various types of representations of the information. It is also used by the TreeModel.

**DATAMODEL**

**TreeModel**

TreeModel uses RecordsManager to construct a linked list given the data that’s to be in it. This is then passed to the GUI. Within TreeModel, TreeItem is used as a pointer to the first record. MainWindow uses the TreeModel to determine the correct information to display. In addition to being a subclass of QAbstractItemModel, it is used by GranFundinTreeModel, PresentationTreeModel, PublicationTreeModel, and TeachingTreeModel.

**TreeItem**

TreeItem is used by TreeModel to store information for the records. A linked list is created which stores the data provided in an efficient manner.

**GUI**

**MainWindow**

As a subclass of QMainWindow, MainWindow inherits the properties of the superclass. This allows it to access the QT library features, which allows for consistency in the interface. MainWindow’s purpose is to set up the user interface. Specifically, this means that it controls the window’s core functionality, including refreshing, loading files, handling errors, filtering, and displaying information through various different mediums.

**Main**

This class serves as the driver for the program, where it runs the application and MainWindow. If the test variable is true, it runs the tests and outputs the results to the log.

**CustomSort**

As a subclass of QDialog, CustomSort inherits the properties of the superclass. This allows it to take on properties which allow for integration with other GUI components. In addition, the CustomSort class allows the user to customize the data they want for representation. This is called in MainWindow when the method new\_sort\_clicked() is called for each Tree item.

**ErrorEditDialog**

As a subclass of QDialog, ErrorEditDialog inherits the properties of the superclass. This allows it to take on properties which allow for integration with other GUI components. In addition, the ErrorEditDialog class is intended to provide an opportunity to discard or edit mandatory fields which are missing.

**PieChartWidget**

As a subclass of QWidget, PieChartWidget inherits the properties of the superclass. This allows it to access the QT library features, which allows for consistency in the interface. Specifically, PieChartWidget helps visualize the data stored in the TreeModel.

**QCustomPlot**

Defines the properties of the interface, with details such as the pen and plotting. These are then used to plot the data onto graphs, which are used in MainWindow.

**STRENGTHS AND WEAKNESSES**

**STRENGTHS**

**Customizability**

The application allows the user to determine their own sort order and how the data is presented. This allows the data to be catered to what the user feels most appropriate.

**Usability**

The GUI is easy to use, and the user inputs are clearly defined. This means that the user will easily be able to use the program easily without much initial guidance, allowing the administration to focus less on training, and more on achieving results.

**Efficiency**

Records are stored in the tree using a linked list, which is more efficient than other alternatives such as an array.

**Functionality**

When the application is run, all classes work as intended. This allows for successful execution of the program.

**Cohesion**

The methods are encapsulated in each class. Specifically, the functionalities are embedded in a class and accessed through its methods.

**Maintainability**

Looking at the structure of the classes, they are fairly generalized, which will allow for easy enhancement through modifications to the code.

**Reusability**

In addition, the classes included are suitable for other applications if the client desires so. This is because classes such as TreeItem and QSortListIO are easily transferrable to other problems if needed. This is because TreeItem stores information within a tree, while QSortListIO reads and writes a QList to a given filename.

**WEAKNESSES**

**Duplication in code**

In MainWindow, the code introduces four of every UI element, which represents one for each of the data types. In the GUI, there is a button for each: presentation, funding, teaching, and publications. This results in four times the amount of code, because the code is repeated for each of them. Specifically, each UI element has its own “on\_clicked” or “on\_toggle”. This could be improved by using cases instead of repeating the code for each UI element.

**Comments**

While comments are well documented for the majority of classes, PieChartWidget has no comments, which makes auditing more difficult, and reduces ease of use in future modifications.