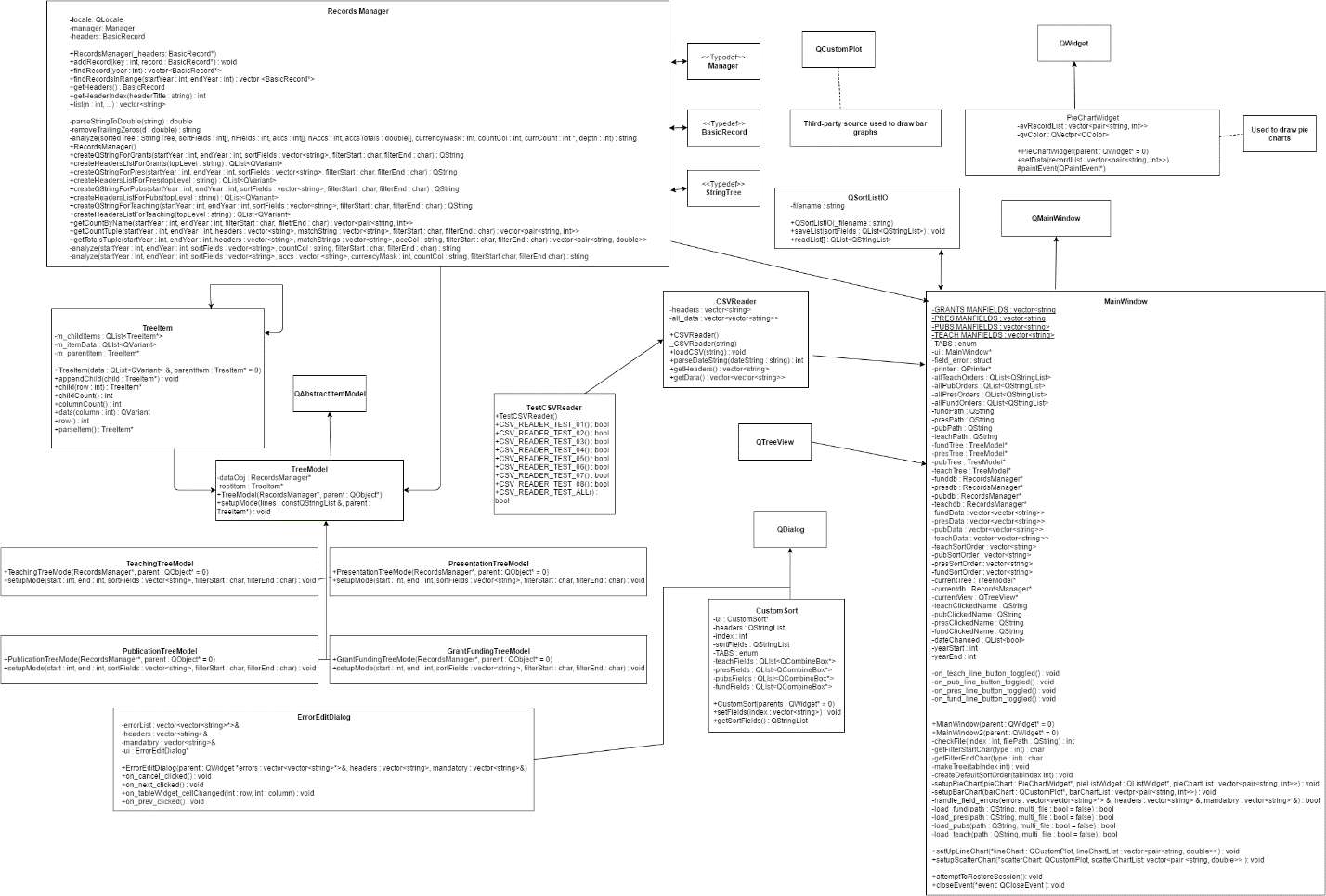
Case Diagram Description

Above is an updated version of the Peachy Galaxy use case diagram for team Lynx. A user is anyone who runs the program, namely managers and faculty. One main use case is “Navigate Dashboard”, and the other is “Visualize Data”. “Navigate Dashboard” refers to the user’s interaction with the system UI. Through this, they are able to sort data and apply filters. “Visualize Data” refers to the various ways the system can output the data utilized. The user has the options sort the data and apply filters, as well as print the file and export data to a PDF. Both of these uses also utilize “Load Data Files”, which represents the user opening CSV files and processing their data through the system. This is also where the system processes errors. Using new methods, the user may also navigate through the various errors encountered.



Class Diagram Description

The system utilizes the original classes from team Peach. Several methods have been added to modify the system. The methods “on\_cancel\_clicked”, “on\_next\_clicked”, “on\_tableWidget\_cellChanged”, andand “on\_prev\_clicked” have been added to “ErrorEditDialog”. class. These methods were added to the class to allow the user to navigate through errors identified in the system and fill in missing data, in accordance with the customer requests. Additionally, “setupLineChart” was added to the “MainWindow” class. This method allows the user to visualize the input data in the form of a line graph, in accordance with the customer’s request to have another way to visualize the data. This method also makes use of “on\_pub\_line\_button\_toggled”, “on\_pres\_line\_button\_toggled”, “on\_teach\_line\_button\_toggled”, and “on\_fund\_line\_button\_toggled”, to allow the user to choose which data is visualized. The method “attemptToRestoreSession” is used to retrieve a previous session, which would have been saved during a “closeEvent”, activated when the user ends a session, to store data from that session. “setUpLineChart” and “setupScatterChart” are both used to produce alternative graph types as requested by the user.



User Clicks Load Button Sequence Diagram

This diagram remains the same as the one used by team Peach. It illustrates the scenario, initiated by a user, to load a file, and the steps taken to do so. The initial action is the user clicks the “load Data Button” from the “MainWindow”. This button exists in four instances, one for each type of data. The process remains the same for each type, but uses the versions of each method for each respective type. The system then loads the file data and checks it to match the required type using the method “checkFile”. The final steps are handled by a “RecordsManager”, created by the main window. The system then retrieves the data and sorts it by adding it to a “TreeModel” of the associated data type. The “analyze” methods is then called to order the data, which is then output back to the “MainWindow”.



User Fills In Data Sequence Diagram

This action is initiated from the “MainWindow” when the user Loads a data set. Before it is added to the “MainWindow” The program checks for missing entries that are required. If missing entries are found, a “QMessageBox” is created requesting if the user would like to edit the entries. If the user selects “Yes”, the following actions occur. First a new “ErrorEditDialogBox” is created and the “MainWindow” passes on the data set. The “ErrorEditDialogBox” then cleans the allocated memory. A “QWidget” is then generated to hold a “QWidgetTableItem”, which are populated from the passed data set. All potential entries are then editable from the “ErrorEditDialogBox”. From here the user can use “on\_nextclicks” and “on\_prev\_clicked” to navigate through the widget items and edit the data. As the data is navigated, the “on\_tableWidget\_cellChanges” update the widget items. The user can then use “on\_save\_clicked” to save the data and return to the main window, or”on\_cancel\_clicked” to navigate back to the “MainWindow” without changing the data.



User Creates New Sort Order Sequence Diagram

The action is initiated from the “MainWindow” when a user clicks the button to create a new sort order. This begins an event to create a new sort order, each specific to the data type for the open window. A “CustomSort” object is then created. This then creates new “QWidget” from which the user can input their desired sort parameters and name their new sort type. The fields within the widget are then set, after which a “QStringList” object is created and populated with the fields. The “QStringList” is then transferred back to be stored inside a variable of the “MainWindow” using the “addItem” method. A “QSortListIO” object is then created so that the new sort may be saved to the “MainWindow” using the “saveList” method. Lastly, the “CustomSort” object is deleted.



Package Diagram

The package diagram remains the same as team Peach’s package diagram. There is still only one type of relationship, dependency, denoted by the solid arrows. Each arrow begins at the source and ends at the target. The Graphical User Interface package represents the classes required for user interaction such as the mainwindow. They are dependent on the other classes as they take input and transfer it to be converted to output and displayed by the other classes. The Database Model package represents the TreeModel classes and RecordsManager. These classes aid in processing the data into the desired format. The DataStructure Model packages represents the classes that are used to store and model the program’s data., taken from the classes in Database Model. Finally, the standard QT and C++ packages represent the standard librairies.

