

**CS3307**  
**2016 First Semester**  
**Peachy Galaxy Phase 2**  
**Team Lynx**  
**Haitian Yu, Claire Robichon, Yusi Xu, Ryan Vansteenkiste, Aboudi Absi,**  
**Matthew Dudycz, Anthony Giugno**  
**Professor Nazim Madhavji**

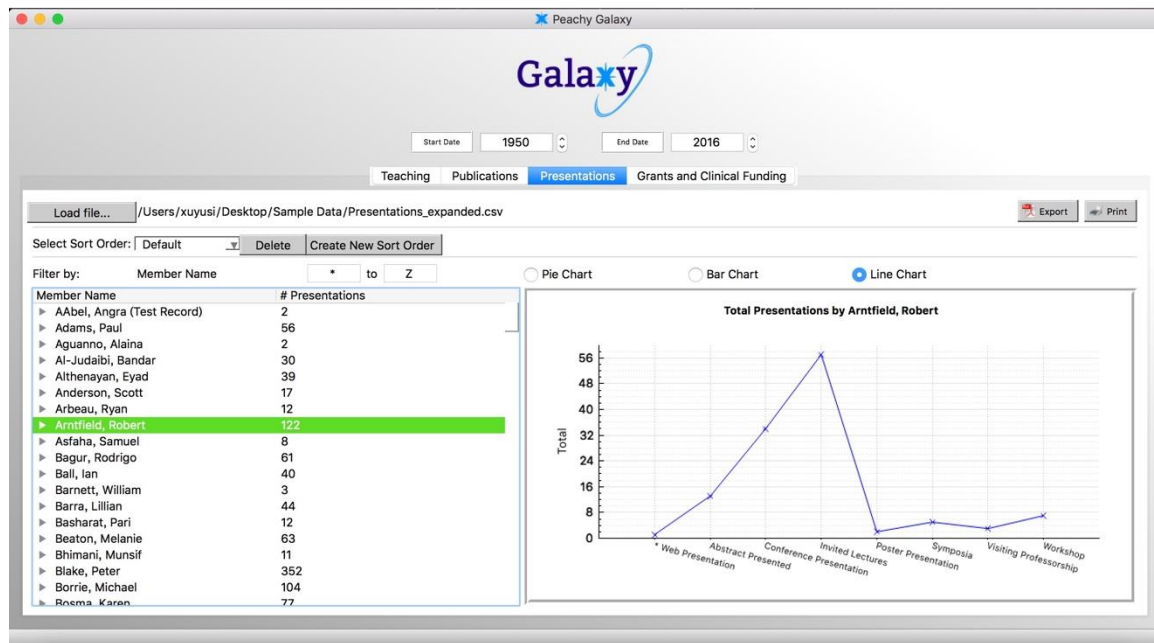
**ID:** MandatoryRequirement2

**Description:** Add a line chart and scatterplot in reporting options

**Origin:** Yusi Xu, Claire Robichon

**Supporting example:**

When we load the file for the Presentations and click on the member name, for example “Arntfield, Robert” here, we choose the graph type “Line Chart” and it will show like following.



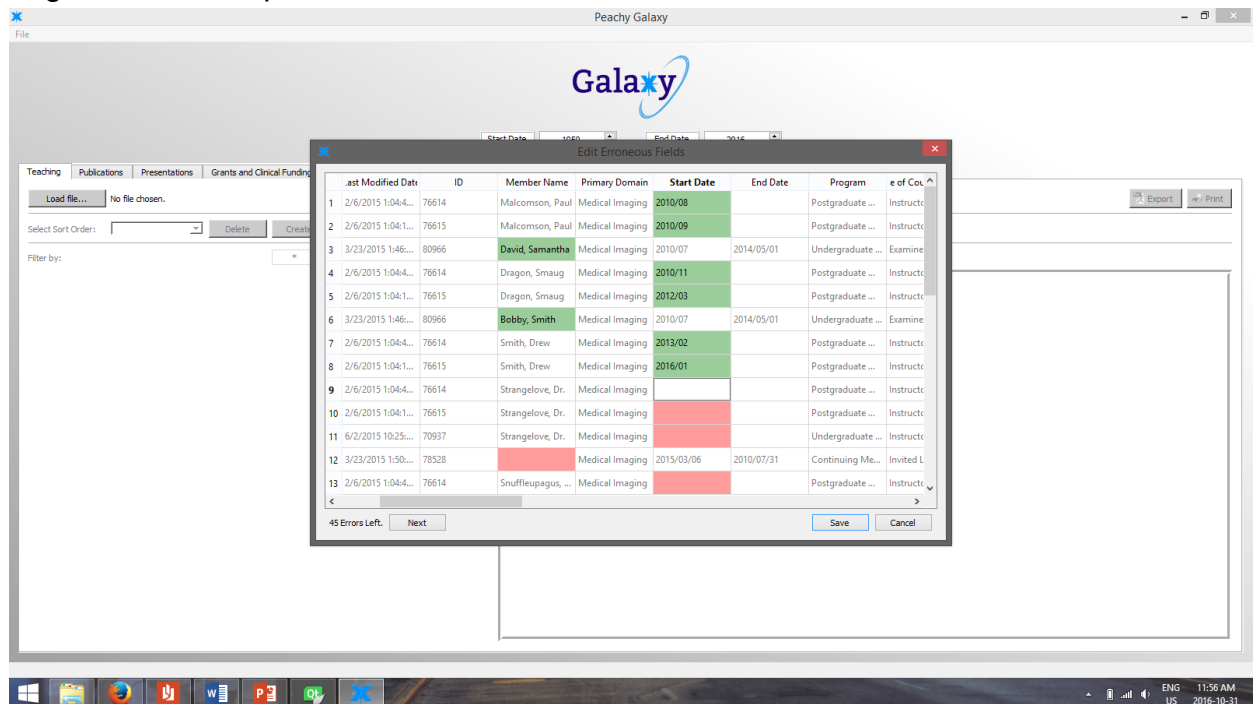
## ID: MandatoryRequirement9

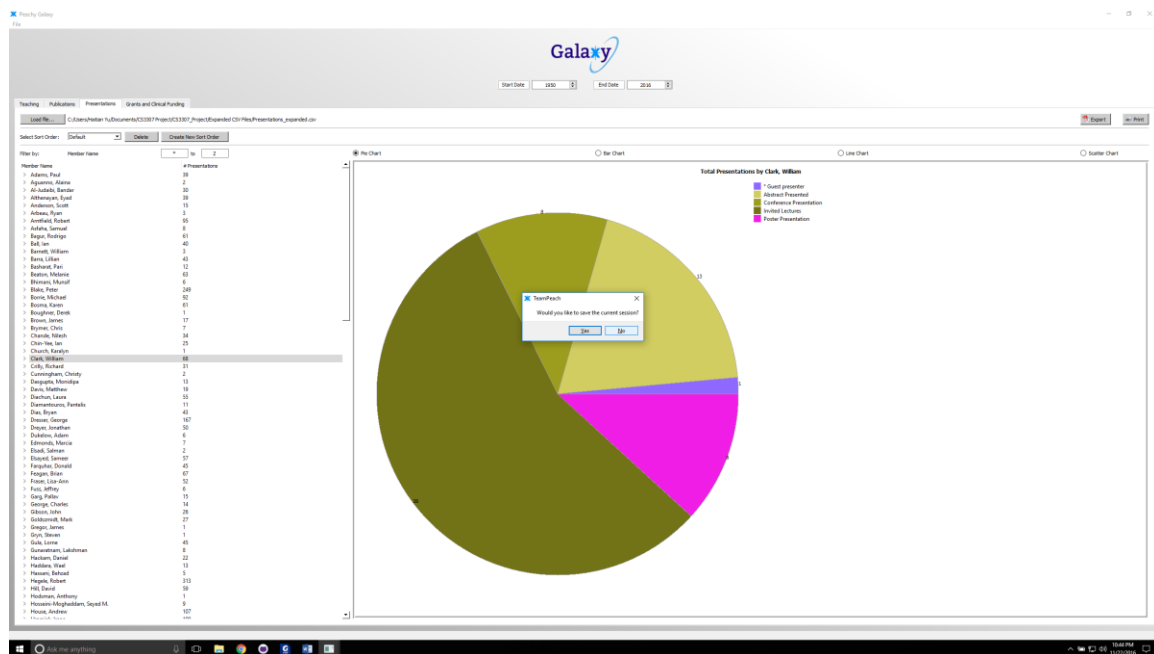
### Description: Navigation of Erroneous Entries (Mandatory Requirement)

When the CSV file is loaded for the first time, the error dialog box should report total number of errors and "Find Next/Prev" button which jumps on to the next/prev error. As the errors are fixed, the error count should go down. After all the errors are fixed, "Find Next/Prev" should be disabled.

**Origin:** Code created by Anthony Giugno

**Supporting Example:** An additional feature for user experiences purposes is also shown in the image below, which is the background color for the error cells turning to green when a specific error has been fixed.





**ID:** MandatoryRequirement4

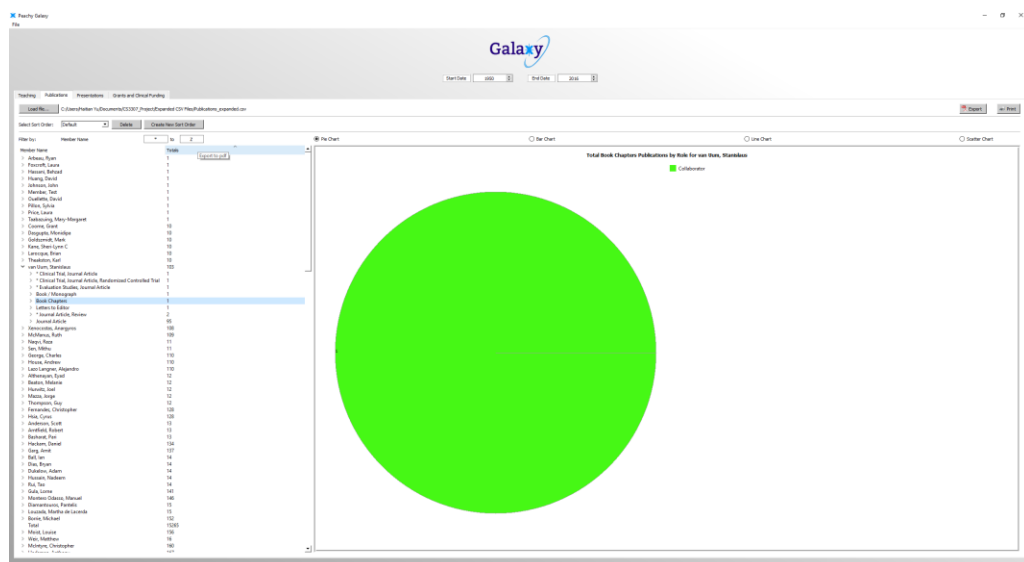
**Description:** Adding the ability to sort by the member's division

Before, in the drop down menu, there was only the option to select sort order. We have added the option to sort by member's division.

**Origin:** Code created by Aboudi

**Supporting Example:** INCOMPLETE

### Supporting Example:



**ID:** MandatoryRequirement6

**Description:** Fix all errors so finished product is error free

**Origin:** Coded by all members of the team

**Supporting Example:** INCOMPLETE





### Supporting Example:

File Galaxy

Galaxy

Start Date: 2018 End Date: 2018

Tasking Publications Presentations Grants and Clinical Funding

Select New... C:\Users\henry\Documents\Galaxy\Projects\GSE67\_Proteo\Expanded GDE\HepaPublications\_expanded.gal [Export] [Go Back]

Select Sort Order: Default Details Create New Sort Order

Filter By: Header Name + - 2

Header Name	
> Addit. Large (Not Rec'd)	2
> Adams, Paul	300
> Aggarwal, Akane	3
> Al-Labadi, Bashar	17
> Bork-Charney	2
> Published Abstract	20
> Athanasios, Eyal	12
> Constantinou	1
> Internal Article	6
> Published Abstract	5
> Anderson, Sam	13
> Internal Article	1
> Published Abstract	12
> Adkins, Ryan	11
> Internal Article	1
> Books Edited	1
> Internal Article	10
> Published Abstract	1
> Hestrich, William	1
> Adkins, Samuel	28
> Rajar, George	15
> Bat, Ian	14
> Barrett, William	9
> Berni, Lillian	28
> Barakat, Paul	12
> Beatin, Indraneel	12
> Bergh, Lynn	6
> Bhawan, Manish	6
> Bala, Peter	279
> Burns, Michael	152
> Burns, Tapan	31
> Bangalore, David	166
> Brown, James	69
> Bryant, Luke	21
> Chandra, Nikhil	57
> Chin, Ian	209
> Chen, Lawrence	89
> Choi, William	109
> Coombe, Grant	19
> Coff, Richard	197
> Dargatzis, Vasil	4
> Deshpande, Monalisa	10
> Davis, Matthew	23
> DeChato, Laura	27
> Demantowski, Patricia	13
> Diaz, Bryan	14
> Denier, George	75
> Diogo, Jonathan	39
> Dubois, Adam	18
> Edwards, Monica	32
> Edwards, Lester	12
> Fenecher, Donald	40
> Ferguson, Brock	402
> Fernandez, Christopher	128
> Ferrel, Yves	90
> Foxcroft, Laura	1
> Fraser, Lisa Ann	20
> Gang, Anne	127
> Garcia, Roberto	1

View: Pie Chart | Bar Chart | Line Chart | Scatter Chart

Total Journal Article Publications by Role for Athanasios, Eyal

Legend: Contributor (Orange), Collaborator (Green), Principal Author (Grey)

Role	Count (Approximate)
Contributor	60
Collaborator	30
Principal Author	10

## Team Lynx – Test Cases

Team peach specified 7 software requirements in their documentation for Peachy Galaxy:

- Loading data from file
- Error Processing
- Applying filters
- Using a custom sort order
- Visualizing data
- Printing to file
- Exporting to PDF

Team Lynx added the features below to the Peachy Galaxy:

- Saving application state
- Two new ways to visualize data (graphs)
- Column Sorting

Based on these requirements, we created test cases for various possible user test cases documented below. Test cases were developed in two methods: first, programmatically writing test cases using the QTest Library; second, manually using the GUI to replicate users actions. All programmatically written test cases can be found in the /test folder of the Peachy Galaxy project, and the 'unit to test row' in the following charts will provide the name of the method which was called in that test case.

Test cases that:

Testcase ID: 023, 021 (Both minor user experience issues)

<b>Test Case ID:</b>	001
<b>Unit to test:</b>	QSORT_LIST_01
<b>Requirements:</b>	3.1.1 Final System Delivery (Loading data from file)
<b>Assumptions:</b>	QSortListIO will catch and throw exception when given an invalid location.
<b>Test Data:</b>	"NO FILE"

<b>Steps to be executed:</b>	Pass string to sorter, then call method .readList() and store output as a QList
<b>Expected Result:</b>	QSortListIO success
<b>Actual Result:</b>	Debugger error
<b>Pass/Fail:</b>	Fail
<b>Comments:</b>	QSortListIO should contain a catch for invalid strings passed to the class.
<b>Solution:</b>	Design a try/catch statments for invalid files locations.

<b>Test Case ID:</b>	002
<b>Unit to test:</b>	QSORT_LIST_02
<b>Requirements:</b>	3.1.1 Final System Delivery (Loading data from file)
<b>Assumptions:</b>	QSortListIO will catch and throw exception when given an invalid location.
<b>Test Data:</b>	NULL
<b>Steps to be executed:</b>	Pass string to sorter, then call method .readList() and store output as a QList
<b>Expected Result:</b>	QSortListIO success
<b>Actual Result:</b>	Debugger error
<b>Pass/Fail:</b>	Fail
<b>Comments:</b>	QSortListIO should contain a catch for invalid strings passed to the class.
<b>Solution:</b>	Design a try/catch statments for invalid files locations.

<b>Test Case ID:</b>	003
<b>Unit to test:</b>	QSORT_LIST_03
<b>Requirements:</b>	3.1.1 Final System Delivery (Loading data from file)
<b>Assumptions:</b>	QSortListIO will catch and throw exception when given an invalid location.
<b>Test Data:</b>	Grants_expanded.csv
<b>Steps to be executed:</b>	Pass string to sorter, then call method .readList() and store output as a QList
<b>Expected Result:</b>	QSortListIO success
<b>Actual Result:</b>	Success
<b>Pass/Fail:</b>	Pass
<b>Comments:</b>	QSortListIO should contain a catch for invalid strings passed to the class.
<b>Solution:</b>	N/A

<b>Test Case ID:</b>	004
<b>Unit to test:</b>	QSORT_LIST_04
<b>Requirements:</b>	3.1.1 Final System Delivery (Loading data from file)
<b>Assumptions:</b>	QSortListIO will catch and throw exception when given an invalid location.
<b>Test Data:</b>	test_file.txt
<b>Steps to be executed:</b>	Create text file, Pass string to sorter, then call method .readList() and store output as a QList
<b>Expected Result:</b>	QSortListIO success
<b>Actual Result:</b>	Success
<b>Pass/Fail:</b>	Pass

<b>Comments:</b>	N/A
<b>Solution:</b>	N/A

<b>Test Case ID:</b>	005
<b>Unit to test:</b>	QCSV_READER_TEST_09
<b>Requirements:</b>	3.1.1 Final System Delivery (Loading data from file)
<b>Assumptions:</b>	CSVReader will catch and throw exception when given an invalid format
<b>Test Data:</b>	STAR 2015 Acuity STAR Western CS intro V2.pptx
<b>Steps to be executed:</b>	Pass string of location to CSVReader, then use method. getData and pass it to vector, then check size to validate that it was accepted
<b>Expected Result:</b>	CVSReader catch invalid file
<b>Actual Result:</b>	Failed. CVSReader did not check file format
<b>Pass/Fail:</b>	Fail
<b>Comments:</b>	CSVReader should not be reliant upon the user the enter a .csv file. There should be a try catch so the program does not crash
<b>Solution:</b>	Design a simple check to validate CSV files.

<b>Test Case ID:</b>	006
<b>Unit to test:</b>	Loading incorrect file from GUI
<b>Requirements:</b>	3.1.1 Final System Delivery (Loading data from file)
<b>Assumptions:</b>	Error thrown on non CVS
<b>Test Data:</b>	Teaching_test.docx
<b>Steps to be</b>	1) Select a tab

<b>executed:</b>	2) Click 'load file' 3) Pick an invalid CVS type 4) Wait for program response
<b>Expected Result:</b>	Program will not load invalid data
<b>Actual Result:</b>	Program did not load invalid file.
<b>Pass/Fail:</b>	Pass
<b>Comments:</b>	N/A
<b>Solution:</b>	N/A

<b>Test Case ID:</b>	007
<b>Unit to test:</b>	Loading incorrect tab types
<b>Requirements:</b>	3.1.1 Final System Delivery (Loading data from file)
<b>Assumptions:</b>	Error thrown on wrong CSV tab type
<b>Test Data:</b>	Grants_expanded.csv, GrantsClinicalFunding_sample.csv, Presentation_expanded.csv, Presentation_sample.csv, Program_Teaching_expanded.csv, Publication_expanded.csv, Publications_sample.csv, Teaching_sample.csv
<b>Steps to be executed:</b>	5) Select a tab 6) Click 'load file' 7) Pick an invalid CVS type 8) Wait for program response
<b>Expected Result:</b>	Program will not load invalid tab type data
<b>Actual Result:</b>	Program did not load incorrect CVS file type.
<b>Pass/Fail:</b>	Pass
<b>Comments:</b>	N/A
<b>Solution:</b>	N/A

<b>Test Case ID:</b>	008
<b>Unit to test:</b>	Clicking Edit Button
<b>Requirements:</b>	3.1.2 Final System Delivery (Error Proccessing)
<b>Assumptions:</b>	When clicking the Edit Button, Peachy galaxy should prompt me with the EditFields dialog.
<b>Test Data:</b>	Program_Teaching_expanded.csv,
<b>Steps to be executed:</b>	1) Load file 2) Click edit button
<b>Expected Result:</b>	EditField dialog box will open
<b>Actual Result:</b>	EditField dialog box will open
<b>Pass/Fail:</b>	Pass
<b>Comments:</b>	N/A
<b>Solution:</b>	N/A

<b>Test Case ID:</b>	009
<b>Unit to test:</b>	Clicking Export Button
<b>Requirements:</b>	- 3.1.8 Final System Delivery (Exporting to PDF)
<b>Assumptions:</b>	When clicking the Export Button Peachy galaxy should prompt me with the “Export File” dialog.
<b>Test Data:</b>	Program_Teaching_expanded.csv,
<b>Steps to be executed:</b>	1) Load file 2) Click export button
<b>Expected Result:</b>	“Export File” dialog box will open
<b>Actual Result:</b>	“Export File” dialog box will open
<b>Pass/Fail:</b>	Pass

<b>Comments:</b>	N/A
<b>Solution:</b>	N/A

<b>Test Case ID:</b>	010
<b>Unit to test:</b>	Clicking Print Button
<b>Requirements:</b>	3.1.7 Final System Delivery (Printing to file)
<b>Assumptions:</b>	When clicking the Print Button Peachy galaxy should prompt me with the "Print" dialog.
<b>Test Data:</b>	Program_Teaching_expanded.csv,
<b>Steps to be executed:</b>	1) Load file 2) Click print button
<b>Expected Result:</b>	"Print" dialog box will open
<b>Actual Result:</b>	"Print" dialog box will open
<b>Pass/Fail:</b>	Pass
<b>Comments:</b>	N/A
<b>Solution:</b>	N/A

<b>Test Case ID:</b>	011
<b>Unit to test:</b>	Clicking User Data
<b>Requirements:</b>	- 3.1.6 Final System Delivery (Visualizing data)
<b>Assumptions:</b>	When clicking a Member Name, their Data is visualized on the currently selected graph.
<b>Test Data:</b>	Program_Teaching_expanded.csv, Lavi, Shahar
<b>Steps to be executed:</b>	1) Load file 2) Click Lavi, Shahar



<b>Expected Result:</b>	Data is modeled on Pie Chart
<b>Actual Result:</b>	Data is modeled on Pie Chart
<b>Pass/Fail:</b>	Pass
<b>Comments:</b>	N/A
<b>Solution:</b>	N/A

<b>Test Case ID:</b>	012
<b>Unit to test:</b>	Clicking Scatter Chart Button
<b>Requirements:</b>	- 3.1.6 Final System Delivery (Visualizing data)
<b>Assumptions:</b>	When a member name is selected and the Scatter Chart Button is selected, a scatter chart is produced
<b>Test Data:</b>	Program_Teaching_expanded.csv, Lavi, Shahar
<b>Steps to be executed:</b>	<ol style="list-style-type: none"> <li>1) Load file</li> <li>2) Click Lavi, Shahar</li> <li>3) Click Scatter Chart Button</li> </ol>
<b>Expected Result:</b>	Data is modeled on Scatter Chart
<b>Actual Result:</b>	Data is modeled on Scatter Chart
<b>Pass/Fail:</b>	Pass
<b>Comments:</b>	N/A
<b>Solution:</b>	N/A

<b>Test Case ID:</b>	013
<b>Unit to test:</b>	Clicking Line Chart Button
<b>Requirements:</b>	- 3.1.6 Final System Delivery (Visualizing data)
<b>Assumptions:</b>	When a member name is selected and the Line Chart Button is selected, a line chart is produced

<b>Test Data:</b>	Program_Teaching_expanded.csv, Lavi, Shahr
<b>Steps to be executed:</b>	1) Load file 2) Click Lavi, Shahr 3) Click Line Chart Button
<b>Expected Result:</b>	Data is modeled on Line Chart
<b>Actual Result:</b>	Data is modeled on Line Chart
<b>Pass/Fail:</b>	Pass
<b>Comments:</b>	N/A
<b>Solution:</b>	N/A

<b>Test Case ID:</b>	014
<b>Unit to test:</b>	Entering g to h for Filter by: Member Name
<b>Requirements:</b>	- 3.1.4 Final System Delivery (Applying filters)
<b>Assumptions:</b>	When characters are entered into the fields for Filtering by: Member Name, all names between the characters are displayed.
<b>Test Data:</b>	Program_Teaching_expanded.csv,
<b>Steps to be executed:</b>	1) Load file 2) Enter g into first box and h into next box
<b>Expected Result:</b>	All entries between g and h are displayed
<b>Actual Result:</b>	All entries between g and h are displayed
<b>Pass/Fail:</b>	Pass
<b>Comments:</b>	N/A
<b>Solution:</b>	N/A

<b>Test Case ID:</b>	015
<b>Unit to test:</b>	Entering g to f for Filter by: Member Name

<b>Requirements:</b>	- 3.1.4 Final System Delivery (Applying filters)
<b>Assumptions:</b>	When characters are entered into the fields for Filtering by: Member Name, all names between the characters are displayed.
<b>Test Data:</b>	Program_Teaching_expanded.csv,
<b>Steps to be executed:</b>	1) Load file 2) Enter g into first box and h into next box
<b>Expected Result:</b>	No entries are displayed
<b>Actual Result:</b>	No entries are displayed
<b>Pass/Fail:</b>	Pass
<b>Comments:</b>	N/A
<b>Solution:</b>	N/A

<b>Test Case ID:</b>	016
<b>Unit to test:</b>	Entering 1 to 9 for Filter by: Member Name
<b>Requirements:</b>	- 3.1.4 Final System Delivery (Applying filters)
<b>Assumptions:</b>	When characters are entered into the fields for Filtering by: Member Name, all names between the characters are displayed.
<b>Test Data:</b>	Program_Teaching_expanded.csv,
<b>Steps to be executed:</b>	1) Load file 2) Enter g into first box and h into next box
<b>Expected Result:</b>	All entries are displayed
<b>Actual Result:</b>	All entries are displayed
<b>Pass/Fail:</b>	Pass
<b>Comments:</b>	N/A
<b>Solution:</b>	N/A

<b>Test Case ID:</b>	017
<b>Unit to test:</b>	Entering a to z for Filter by: Member Name
<b>Requirements:</b>	- 3.1.4 Final System Delivery (Applying filters)
<b>Assumptions:</b>	When characters are entered into the fields for Filtering by: Member Name, all names between the characters are displayed.
<b>Test Data:</b>	Program_Teaching_expanded.csv,
<b>Steps to be executed:</b>	<ol style="list-style-type: none"> <li>1) Load file</li> <li>2) Enter g into first box and h into next box</li> </ol>
<b>Expected Result:</b>	All entries are displayed
<b>Actual Result:</b>	Entries starting with lower case are displayed last
<b>Pass/Fail:</b>	Fail
<b>Comments:</b>	As the box parameters are not case sensitized, the program should display the names in alphabetical order regardless of case to avoid confusion.
<b>Solution:</b>	Convert names to lower case before filtering.

<b>Test Case ID:</b>	018
<b>Unit to test:</b>	Entering 1999 to 2015 for filtering by date
<b>Requirements:</b>	- 3.1.4 Final System Delivery (Applying filters)
<b>Assumptions:</b>	When characters are entered into the fields for filtering by date, all names between the dates are displayed.
<b>Test Data:</b>	Program_Teaching_expanded.csv,
<b>Steps to be executed:</b>	<ol style="list-style-type: none"> <li>1) Load file</li> <li>2) Enter g into first box and h into next box</li> </ol>
<b>Expected Result:</b>	All entries between the dates are displayed
<b>Actual Result:</b>	All entries between the dates are displayed

<b>Pass/Fail:</b>	Pass
<b>Comments:</b>	N/A
<b>Solution:</b>	N/A

<b>Test Case ID:</b>	019
<b>Unit to test:</b>	Entering 2015 to 2014 for filtering by date
<b>Requirements:</b>	- 3.1.4 Final System Delivery (Applying filters)
<b>Assumptions:</b>	When characters are entered into the fields for filtering by date, all names between the dates are displayed.
<b>Test Data:</b>	Program_Teaching_expanded.csv,
<b>Steps to be executed:</b>	1) Load file 2) Enter g into first box and h into next box
<b>Expected Result:</b>	All entries between the dates are displayed, as the last date cannot be entered
<b>Actual Result:</b>	All entries between the dates are displayed, as the last date cannot be entered
<b>Pass/Fail:</b>	Pass
<b>Comments:</b>	N/A
<b>Solution:</b>	N/A

<b>Test Case ID:</b>	020
<b>Unit to test:</b>	Print Button
<b>Requirements:</b>	- 3.1.7 Print to File
<b>Assumptions:</b>	First print button will open printdialog, second print button will print the selected graph.
<b>Test Data:</b>	Program_Teaching_expanded.csv,
<b>Steps to be</b>	1) Load file

<b>executed:</b>	2) Select data 3) Press first print button 4) Press print in the dialog box 5) File prints
<b>Expected Result:</b>	File prints with desired information.
<b>Actual Result:</b>	File prints with desired information.
<b>Pass/Fail:</b>	Pass
<b>Comments:</b>	N/A
<b>Solution:</b>	N/A

<b>Test Case ID:</b>	021
<b>Unit to test:</b>	Print Button when no data is selected
<b>Requirements:</b>	- 3.1.4 Print to File
<b>Assumptions:</b>	Peachy galaxy will not let you print if you do not have any data selected.
<b>Test Data:</b>	Program_Teaching_expanded.csv,
<b>Steps to be executed:</b>	1) Load file 2) Press first print button 3) Press print in the dialog box 4) File prints
<b>Expected Result:</b>	File will not allow user to print
<b>Actual Result:</b>	File prints blank page.
<b>Pass/Fail:</b>	Fail
<b>Comments:</b>	Program should not allow user to print blank page
<b>Solution:</b>	If there is no data selected, disable to print button.

<b>Test Case ID:</b>	022
----------------------	-----

<b>Unit to test:</b>	Exporting data
<b>Requirements:</b>	- 3.1.8 Exporting to PDF
<b>Assumptions:</b>	Peachy galaxy will export your selected data
<b>Test Data:</b>	Program_Teaching_expanded.csv,
<b>Steps to be executed:</b>	<ol style="list-style-type: none"> <li>1) Load file</li> <li>2) Select data</li> <li>3) Press export button</li> <li>4) PDF saves</li> </ol>
<b>Expected Result:</b>	File will export and pdf
<b>Actual Result:</b>	File exports as pdf
<b>Pass/Fail:</b>	Pass
<b>Comments:</b>	N/A
<b>Solution:</b>	N/A

<b>Test Case ID:</b>	023
<b>Unit to test:</b>	Exporting data
<b>Requirements:</b>	- 3.1.8 Exporting to PDF
<b>Assumptions:</b>	Peachy galaxy will not export to pdf if no data is selected
<b>Test Data:</b>	Program_Teaching_expanded.csv,
<b>Steps to be executed:</b>	<ol style="list-style-type: none"> <li>1) Load file</li> <li>2) Press export button</li> <li>3) Export data</li> </ol>
<b>Expected Result:</b>	File will not allow user to export blank file
<b>Actual Result:</b>	File exports
<b>Pass/Fail:</b>	Fail
<b>Comments:</b>	Program should not allow user to export blank file

<b>Solution:</b>	If there is no data selected, disable the export button.
------------------	--

<b>Test Case ID:</b>	024
<b>Unit to test:</b>	Saving users state
<b>Requirements:</b>	- Saving users state
<b>Assumptions:</b>	Peachy galaxy will save users state when prompt accepted
<b>Test Data:</b>	Program_Teaching_expanded.csv
<b>Steps to be executed:</b>	<ol style="list-style-type: none"> <li>1) Load file</li> <li>2) Select data</li> <li>3) Exit program, and press yes to save state</li> <li>4) Start program</li> <li>5) Data reload into program</li> </ol>
<b>Expected Result:</b>	File currently loaded will be reloaded upon restart
<b>Actual Result:</b>	File reloads into peachy galaxy
<b>Pass/Fail:</b>	Pass
<b>Comments:</b>	Potential bug: Save state does not save current tab, user must switch back to tab which data was loaded into
<b>Solution:</b>	N/A

<b>Test Case ID:</b>	025
<b>Unit to test:</b>	Saving users state
<b>Requirements:</b>	- Saving users state
<b>Assumptions:</b>	Peachy galaxy will save users only when accepted
<b>Test Data:</b>	Program_Teaching_expanded.csv
<b>Steps to be executed:</b>	<ol style="list-style-type: none"> <li>6) Load file</li> <li>7) Select data</li> <li>8) Exit program, and press yes to save state</li> <li>9) Start program</li> </ol>



	10)Data reload into program 11)Load different data into program 12)Exit program and decline save 13)Start program, data from step 7 loads
<b>Expected Result:</b>	Save state will only save when prompt accepted
<b>Actual Result:</b>	Save state will only save when prompt accepted
<b>Pass/Fail:</b>	Pass
<b>Comments:</b>	Potential bug: Save state does not save current tab, user must switch back to tab which data was loaded into
<b>Solution:</b>	N/A

<b>Test Case ID:</b>	026
<b>Unit to test:</b>	Clicking Bar Button
<b>Requirements:</b>	- 3.1.6 Final System Delivery (Visualizing data)
<b>Assumptions:</b>	When a member name is selected and the Bar Chart Button is selected, a bar chart is produced
<b>Test Data:</b>	Program_Teaching_expanded.csv, Lavi, Shahr
<b>Steps to be executed:</b>	4) Load file 5) Click Lavi, Shahr 6) Click Bar Chart Button
<b>Expected Result:</b>	Data is modeled on Bar Chart
<b>Actual Result:</b>	Data is modeled on Bar Chart
<b>Pass/Fail:</b>	Pass
<b>Comments:</b>	N/A
<b>Solution:</b>	N/A

<b>Test Case ID:</b>	027
----------------------	-----

<b>Unit to test:</b>	Clicking PieChart Button
<b>Requirements:</b>	- 3.1.6 Final System Delivery (Visualizing data)
<b>Assumptions:</b>	When a member name is selected and the Pie chart Button is selected, a bar chart is produced
<b>Test Data:</b>	Program_Teaching_expanded.csv, Lavi, Shahar
<b>Steps to be executed:</b>	7) Load file 8) Click Lavi, Shahar 9) Click Pie chart Button
<b>Expected Result:</b>	Data is modeled on Pie Chart
<b>Actual Result:</b>	Data is modeled on Pie Chart
<b>Pass/Fail:</b>	Pass
<b>Comments:</b>	N/A
<b>Solution:</b>	N/A

<b>Test Case ID:</b>	028
<b>Unit to test:</b>	Column Sorting
<b>Requirements:</b>	Column Sorting
<b>Assumptions:</b>	Loaded data can be sorted by their column name.
<b>Test Data:</b>	Program_Teaching_expanded.csv
<b>Steps to be executed:</b>	1) Load file 2) Click on column name
<b>Expected Result:</b>	Data is sorted automatically
<b>Actual Result:</b>	Data is sorted automatically
<b>Pass/Fail:</b>	Pass
<b>Comments:</b>	Column is sorted lexicographically instead of incrementally
<b>Solution:</b>	N/A

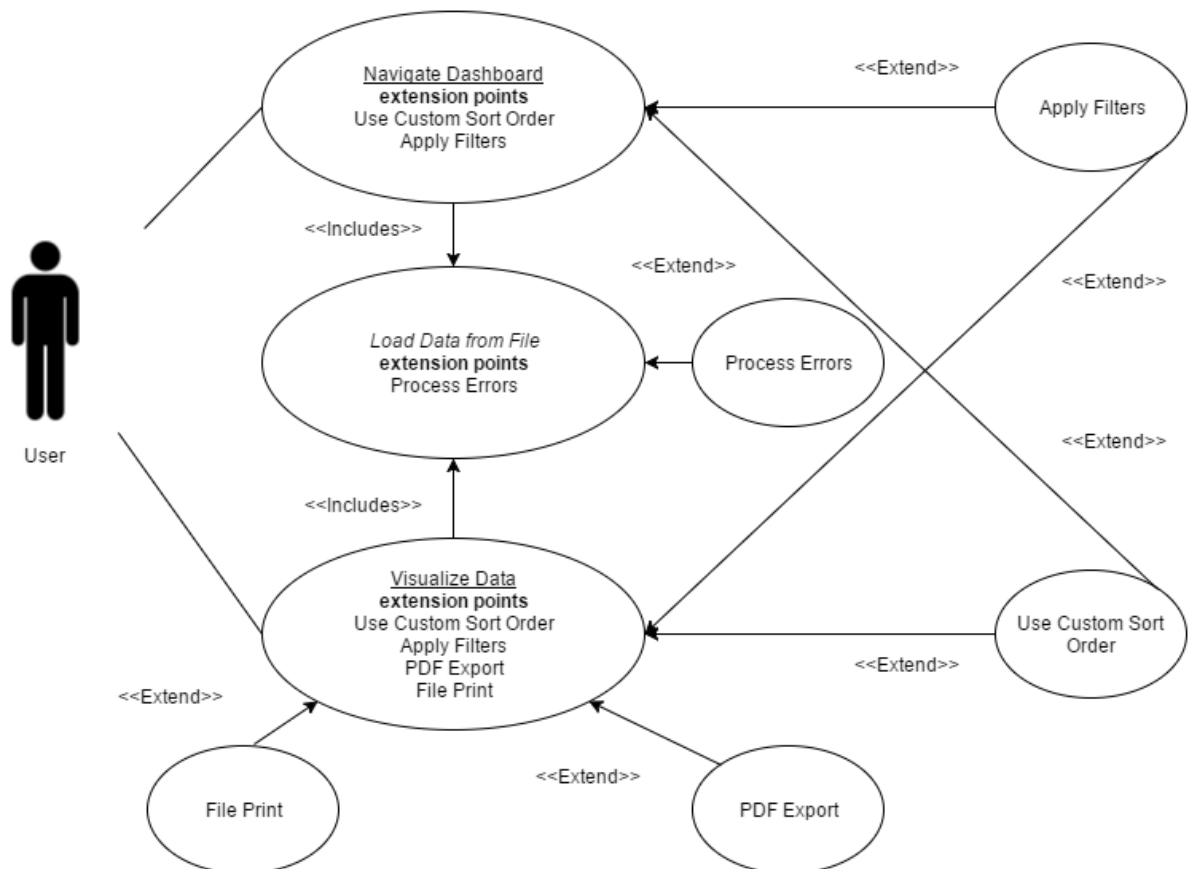
# Test Matrix

Requirements	Test Case #	Description
3.1.1 Loading Data from File	1	Passing invalid location to sorter
	2	Passing invalid location to sorter
	3	Passing correct location to sorter
	4	Passing correct location of corrupted file
	5	Passing invalid file type to reader
	6	Loading non csv file
3.1.2 Error Processing	7	Opening incorrect csv data type on tab
	8	Clicking edit button
3.1.4 Applying filters & 3.1.5 Using sort order	14	Filtering by member name alphabetically
	15	Filtering by member name alphabetically
	16	Filtering by member name numerically
	17	Filtering by member name alphabetically with case sensitivity
	18	Filtering date
	19	Filtering date
3.1.6 Visualizing Data	11	Selecting user data
	12	Selecting user data and opening in scatter chart
	13	Selecting user data and opening in line chart
	26	Selecting user data and opening in bar chart
	27	Selecting user data and opening in pie chart
3.1.7 Printing to file	10	Opening print dialog
	20	Printing to file
	21	Printing to file with no data selected
3.1.8 Exporting to PDF	9	Exporting file dialog
	22	Export file
	23	Export file with no data selected
Saving users state	24	Checking if users state was saved
	25	Checking if users can decline save state
Coloumn Sorting	26	Check is column sorting works

# Diagrams

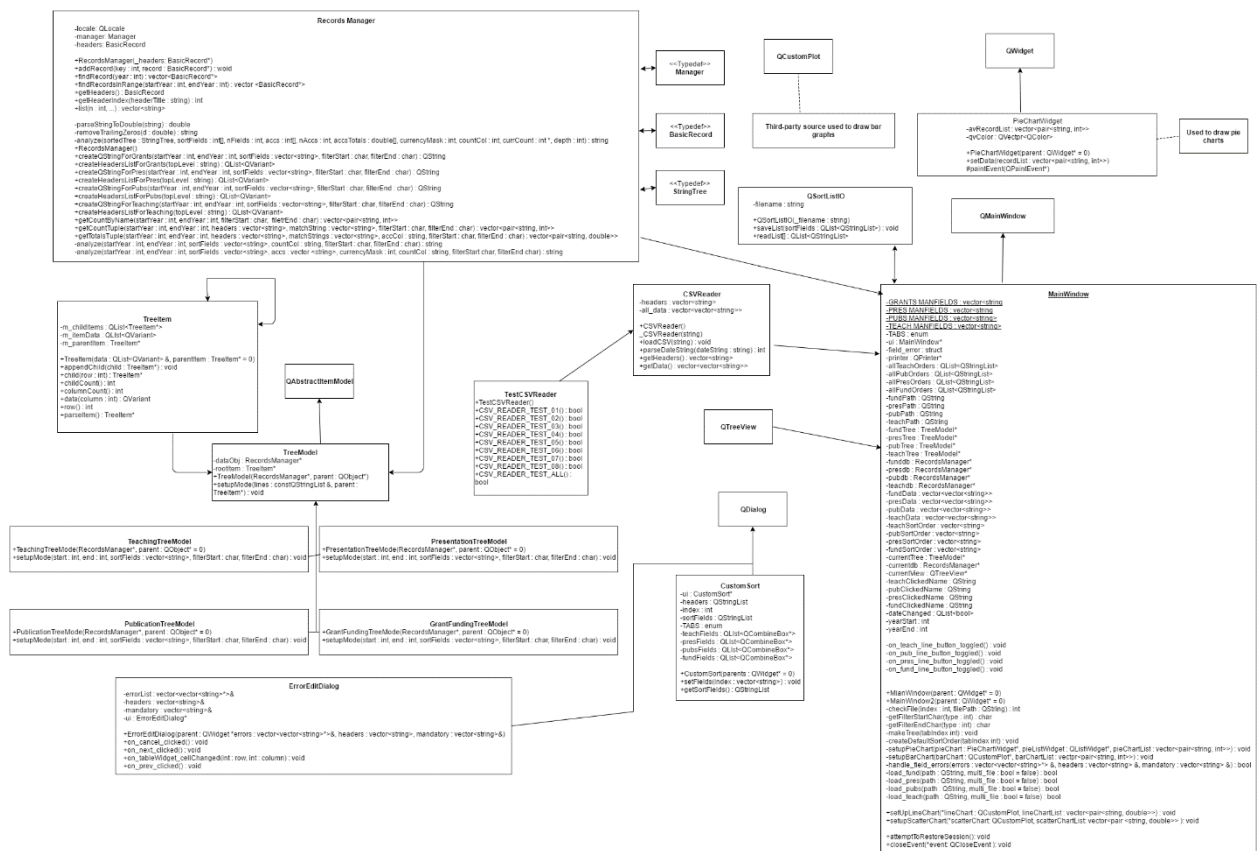
## Case Diagram Description:

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.



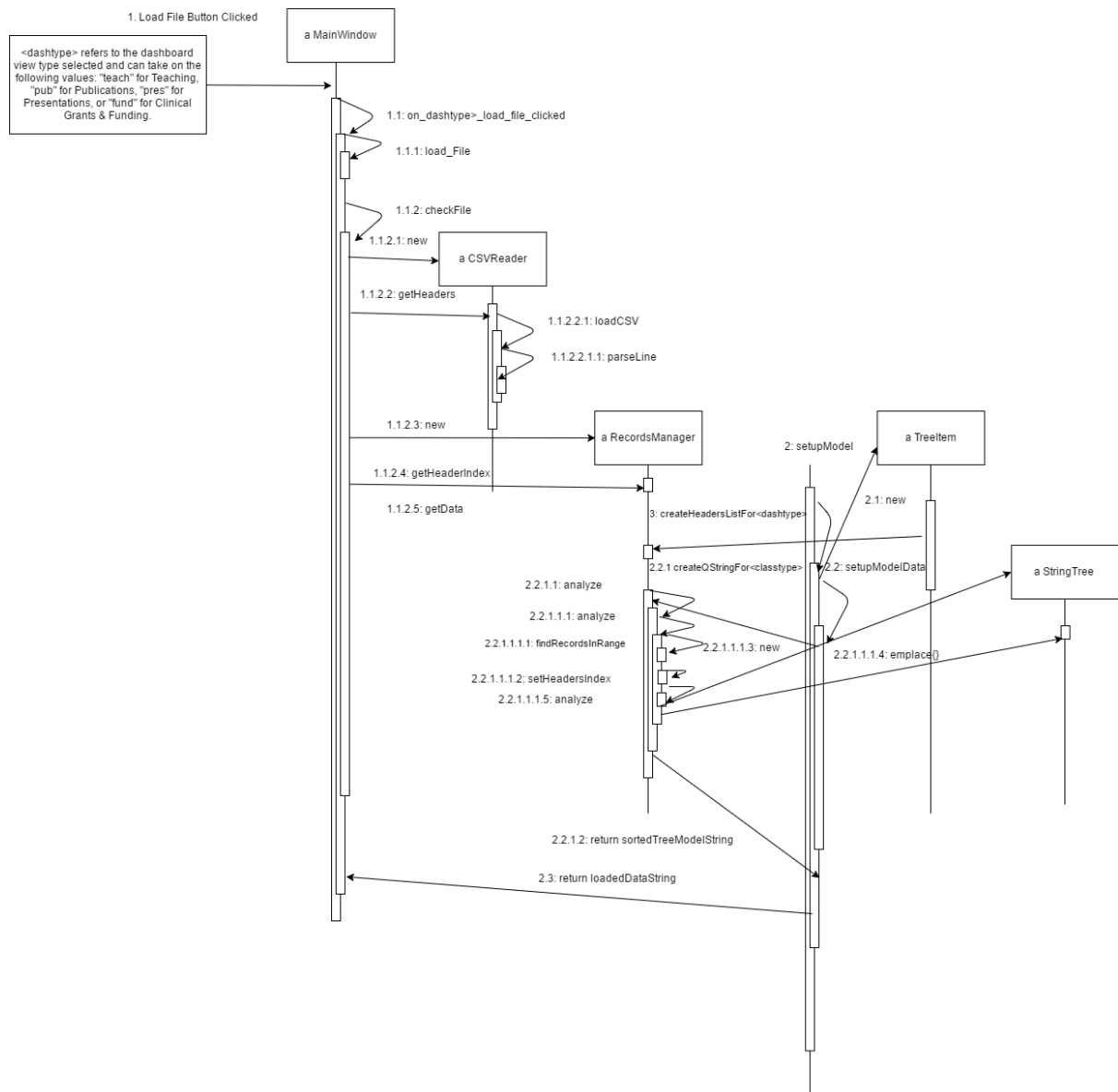
## Class Diagram Description

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”, and “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.



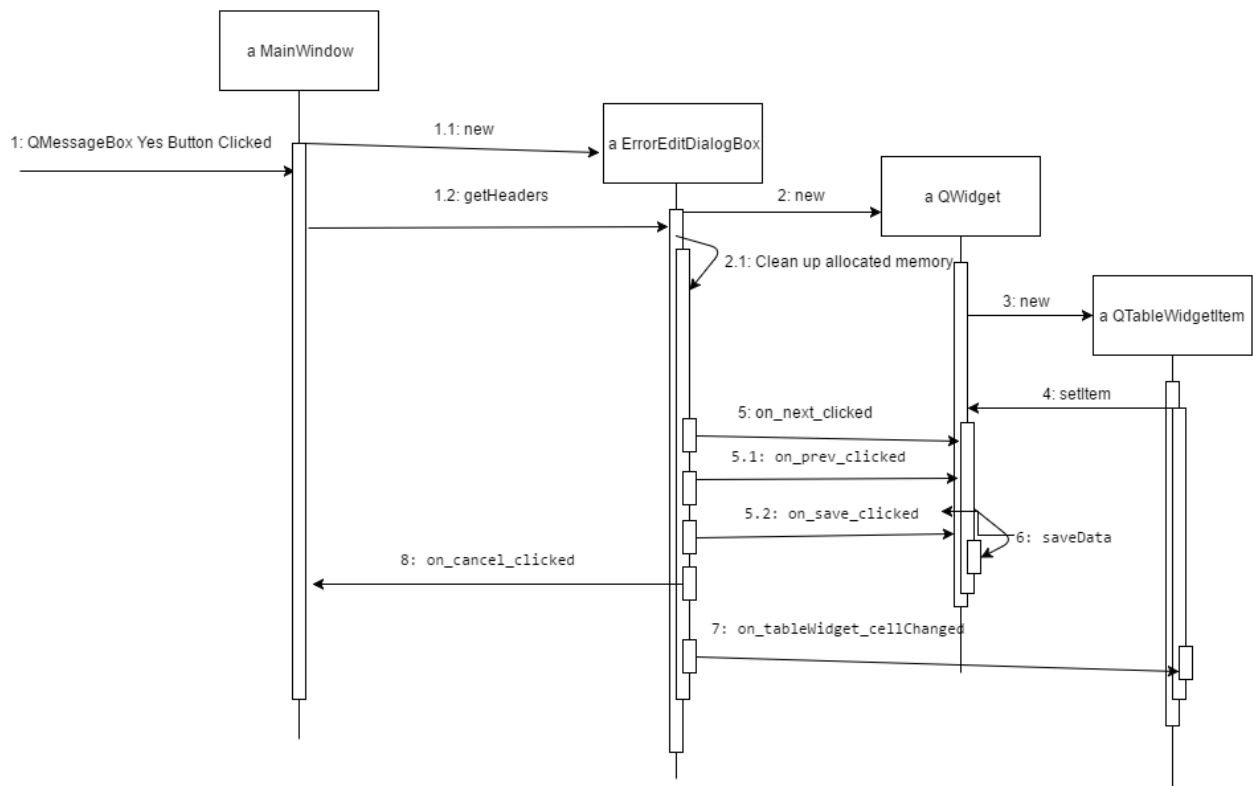
## User Clicks Load Button Sequence Diagram

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”.



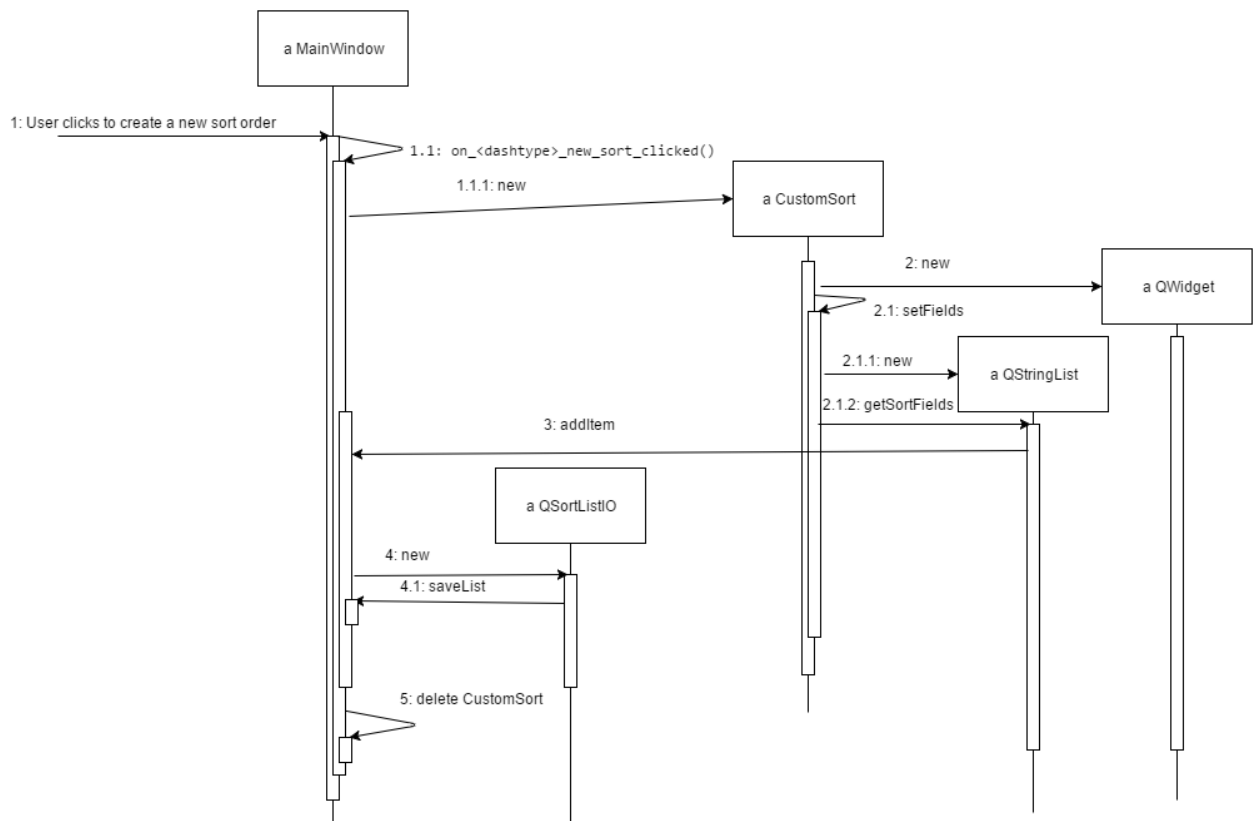
## User Fills In Data Sequence Diagram

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\_next\_clicked” 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

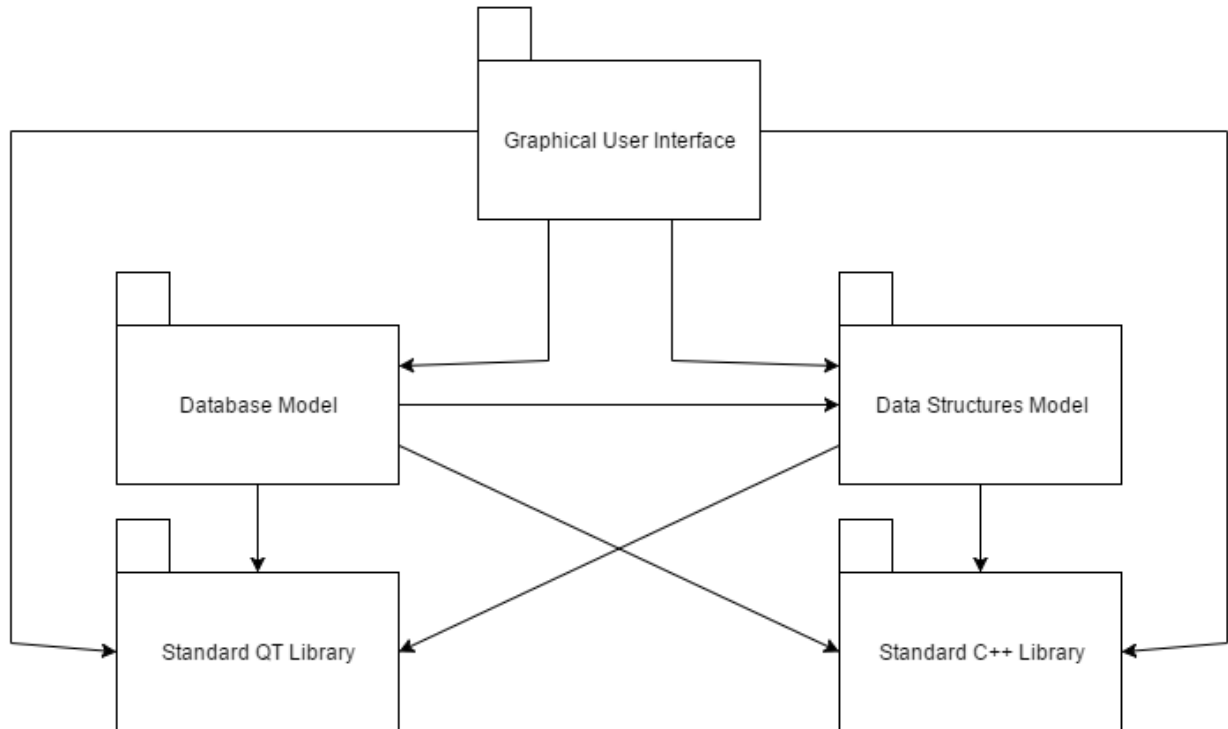
The action is initiated from the “MainWindow” when a user clicks the button to create a new sort order. This begins an event to create a new sort order, each specific to the data type for the open window. A “CustomSort” object is then created. This then creates new “QWidget” from which the user can input 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.

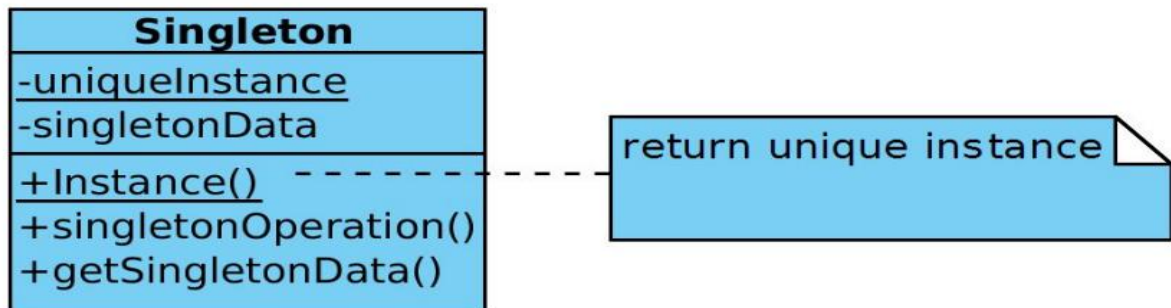


# Design Patterns:

For most of our deliverables, we followed the design patterns of Peachy Galaxy as we are adding additional functionality and not creating the program from scratch. Therefore, all our deliverables are coded as additional functions in Peachy Galaxy's classes.

## **Singleton**

Singleton design solves the following problem: application needs one, and only one, instance of an object. The class is responsible for keeping track of the single instance (prevents other instances from being created as well as controls access to the instance).



## **Benefits to Singleton:**

- Using a singleton gave us an advantage over global variables because we can be 100% sure about the number of instances. It also reduces namespace by avoiding global variables that store single instances
- Easier to refine operation and representation as the Singleton class can be subclassed
- Singleton encapsulates the instance; therefore, it has control over access and creation of instances
- Flexible number of instances (if we want to add instances)

## **Implementation in C++:**

Singleton design is used in the implementation of MainWindow (controls the graphical user interface). The additions we have made to this class are the new graph functions `setupScatterChart` and `setupLineChart` which continue to follow the singleton design pattern.

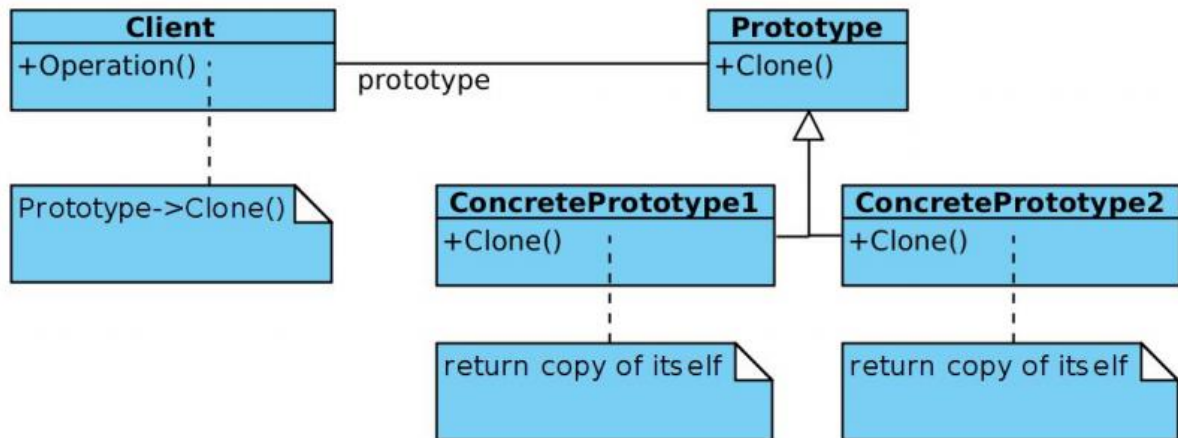
## **Sources:**

[https://www.tutorialspoint.com/design\\_pattern/singleton\\_pattern.htm](https://www.tutorialspoint.com/design_pattern/singleton_pattern.htm)

[https://sourcemaking.com/design\\_patterns/singleton](https://sourcemaking.com/design_patterns/singleton)

## Prototype

Prototype design solves the following problem: application “hard wires” the class of object to create in each “new” expression. If an application/program has many objects which have similar behavior (slight differences in behavior), we can use prototype design to clone instances of the desired class.



### Benefits to Prototype:

- Hides the concrete product classes from client (reduces the number of names clients know about)
- Allows client to work with application specific classes without modification
- Allows for adding and removing products at run-time: can introduce new product class into system by registering a prototype instance.
- Reduce number of needed classes: a client can exhibit new behavior by delegating responsibility to a prototype. By changing values for object variables, we can basically define “new” classes without doing so
- Prototyping, unlike Factory Method, doesn't require sub-classing

### Implementation in C++:

Prototype design is used in the implementation of the TreeModel class. It uses a prototype manager which means when the number of prototypes in a system isn't fixed, it keeps a registry of available prototypes.

Sources:

[https://sourcemaking.com/design\\_patterns/prototype](https://sourcemaking.com/design_patterns/prototype)

[https://www.tutorialspoint.com/design\\_pattern/prototype\\_pattern.htm](https://www.tutorialspoint.com/design_pattern/prototype_pattern.htm)

# Agent Task View

	Haitian	Yusi	Claire	Ryan	Aboudi	Matthew	Anthony
<b>Deliverable 2</b>							
Test Cases				x			
GUI Analysis		x	x				
Code/Algorithm Analysis	x						x
Documentation Analysis						x	
Test Case Fixes				x			
GUI Recommendations							
Code/Algorithm Recommendations	x						x
Documentation Recommendations						x	
<b>Deliverable 3</b>							
Timeline Creation/Upkeep	x						
Agent Task View	x						
System Design Diagrams						x	
Test Cases				x			
Test Case Fixes				x			
Mandatory Requirement 9							x
Stretch Requirement 3 (Begin Work)	x						
<b>Deliverable 4</b>							
Training Video	x						
Sequence Diagram						x	
Package Diagram						x	
Design Patterns	x						
Implementation in C++	x						
Test Cases for Stage 2 Deliverables				x		x	
Mandatory Requirement 2 (Other Graph)		x	x				
Save State			x				x
Verify Program Works on Windows 7/10	x	x	x				x
User Proof Install of Project	x	x	x				x
Fix Parsing (Publications, Presentations, and Grants)		x					
Documentation	x						
<b>Deliverable 5</b>							
Sort By Member's Division					x		
Sort by User Selected List					x		
Fix All Errors So Product is Error Free	x	x	x	x	x	x	x
Test Cases for Final				x		x	
Training Video Completed	x						
Stretch Goal 2		x			x		
Stretch Goal 3			x				x
Inspections And OO Metrics	x				x		
Lessons Learnt and Retrospective Analysis	x						
Polish Deliverable (Title Page, Table of Contents)		x					

# Timeline + Achievements

Date	Summary of Meeting	Members Absent
10/7/2016	Split deliverable 2 (one due on October 19 <sup>th</sup> ) into individual assignments	
	- Anthony & Haitian = coding structure & efficiency	
	- Ryan & Aboudi = testing the program and creating test cases	
	- Matthew = assessment of documentation	
	- Yusi & Claire = assessment of the user interface	
	As each person evaluates portion of code, think of potential improvements in the area	
10/14/2016	Brainstormed improvement ideas	Aboudi
	Addressed problems with respective areas of the program	
	Consolidated all information	
10/21/2016	Discussed test cases	Aboudi
	- GUI and CSV Test cases	
	- Plan to add more test cases focused on GUI before deliverable due date	
	Added a few recommendations to code/algorithm analysis	
	Compiled deliverables (each individual's analysis)	
	Prepared for demo on Monday	
Stage 1 Achievements	Half Test Cases	
	Analyzed Code	
	Analyzed Algorithms	
	Analyzed Documentations	
	Analyzed GUI	
	Recommendations for Code	
	Recommendations for GUI	
	Recommendations for Documentation	
10/26/2016	Recommendations for Fixing Errors	
	Demo1 Enhanced Requirements Determined	Anthony
	Demo1 Enhanced Requirements Tasks Allocated to Specific Group Members	
	Plans to Create Remainder of Test Cases	
	Plans to Create Solutions for All Test Cases	
10/28/2016	All Test Cases Done	Aboudi
	Half Test Cases Solved	
	Case Diagrams Done	
	Original Class Diagrams Done	
	Work Done on Requirements	
	Development Plan Completed	
10/31/2016	All Requirements for Demo1 Done	Aboudi
	All Test Cases Solved	
10/31/2016 (In Class)	Work on Any Incomplete Requirements for Demo1	Aboudi
	Fix Any Test Cases Not Yet Solved for Demo1	
	Allocate Responsibilities for Demo2	
	Update Timeline and Agent Task View	
Stage 2 Achievements		
	All Original Code Test Cases + Solutions Complete	
	Line Graph Added to Project	
	Navigation of Erroneous Errors Added to Project	
	Started Test Code for Deliverables	
	Project Timeline Up To Date	
	Agent Task Up To Date	
	Started Work on Saving State	
	Case Diagram Done	
	Original Class + Enhanced Class Diagrams Done	

11/3/2016 (In Class)	Work on Demo2 Requirements	Aboudi
	Work on Stretch Requirements	
	Begin Work on Package Diagram	
	Begin Work on Sequence Diagram	
11/4/2016	Meet To Talk About Progress of Demo2 Requirements	Aboudi
	Address Any Pain Points	
	Begin Creation of Test Cases for Phase 2 Requirements	
	Write Up Design Patterns	
11/7/2016	Write Up Implementation in C++	
	Talked about deliverable 2	Ryan
	What we need to fix (mostly test cases)	Aboudi
	Need to add more test cases and solve all test cases	
11/11/2016	Must create and solve test cases for deliverables	
	Talked about the CSV errors	
	Problems with array out of bound errors	
	Problems with MAC? Consider using only Windows OS	
11/18/2016	Test cases created matrix	Aboudi
	Progress on diagrams	Anthony
	Save state ideas plan created	
	- Will begin coding for this week	
11/21/2016	Will begin second graph coding this week	
	- Scatterplot	
	Begin fixing problems with the CSV files	
11/18/2016	Save state completed	
	CSV reading fixed	
	Scatterplot graph completed	
	Package diagram completed	
11/21/2016	Sequence diagram almost completed	
	- Waiting on sort code	
	Sort code will be done by Monday	
11/21/2016	<b>Still To Do</b>	
	User proof install	
	Test cases for deliverables	
	- Save state	
Stage 3 Achievements	- Navigation of errors	
	- New sort test code	
	Test cases in matrix	
11/21/2016	Test cases mostly completed	Aboudi
	Still waiting for Aboudi's sorting code	Anthony
	GUI test cases missing (Aboudi said this was done a long time ago but hasn't committed to Github)	
	Compile all documentation into a pdf	
Stage 3 Achievements	Figuring out how to create an install package	
	Verification that application works on Windows 7 completed	
	Scatterplot graph completed	
	Installer completed	
Stage 3 Achievements	Test code for deliverables mostly completed	
	Project Timeline Up To Date	
	Agent Task Up To Date	
	Save state completed	
Stage 3 Achievements	All diagrams completed	
	Work done on new sorting requirements	

# Plan for Final Deliverable

11/25/2016	Assign everybody their roles for the final deliverable	
	Figure out as a group how to complete the sorting requirement	
	Work as a group to try and find errors so the program can be error free	
	Begin work on stretch goals	
	Begin work on final test cases	
12/2/2016	Sorting requirement has been completed at this date	
	Program is now error free	
	Test cases should be almost complete if not complete by this date	
	Training video (for mandatory deliverables) should be complete	
	Inspection work begins	
11/5/2016	Lessons learnt and retrospective analysis complete	
	Inspections and OO analysis metrics complete	
	Test cases completed	
	Stretch goals complete	
	Training video including stretch goals complete	
11/7/2016	Polish up final deliverable (title page, table of contents)	
	Ensure final deliverable is completely error free	