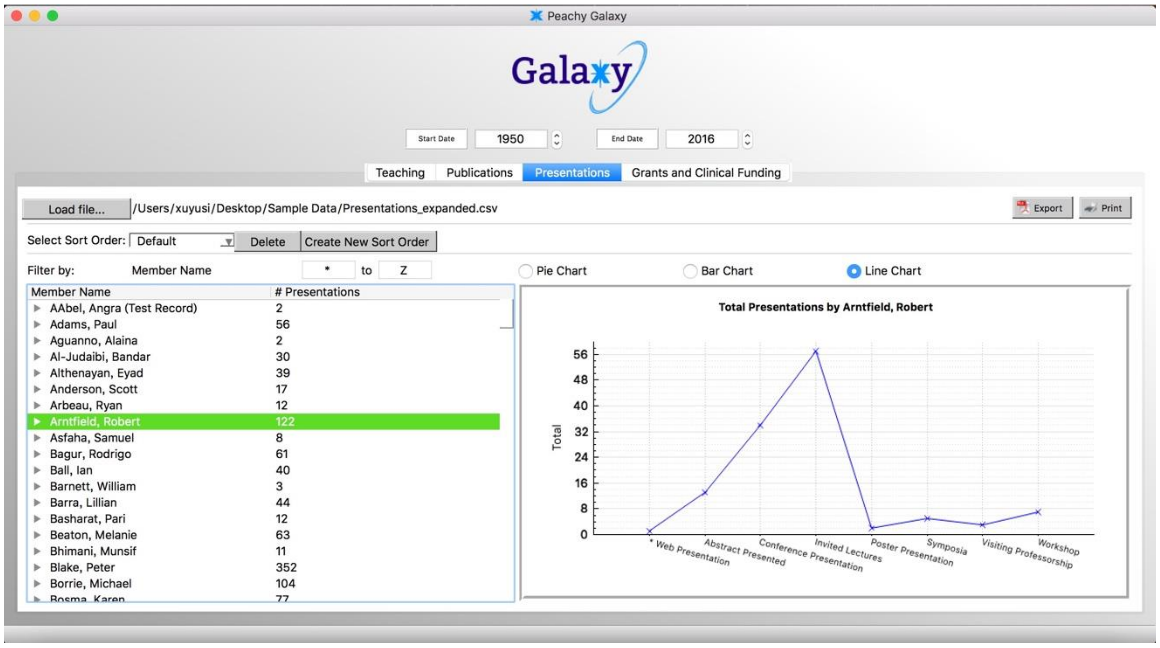
CS3307
2016 First Semester
Peachy Galaxy Phase 2
Team Lynx
Haitian Yu, Claire Robichon, Yusi Xu, Ryan Vansteenkiste, Aboudi Absi, Matthew Dudycz, Anthony Giugno
Professor Nazim Madhavji

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.



Start Date 1950 End Date 2016

/Users/xuyusi/Desktop/Sample Data/Presentations_expanded.csv

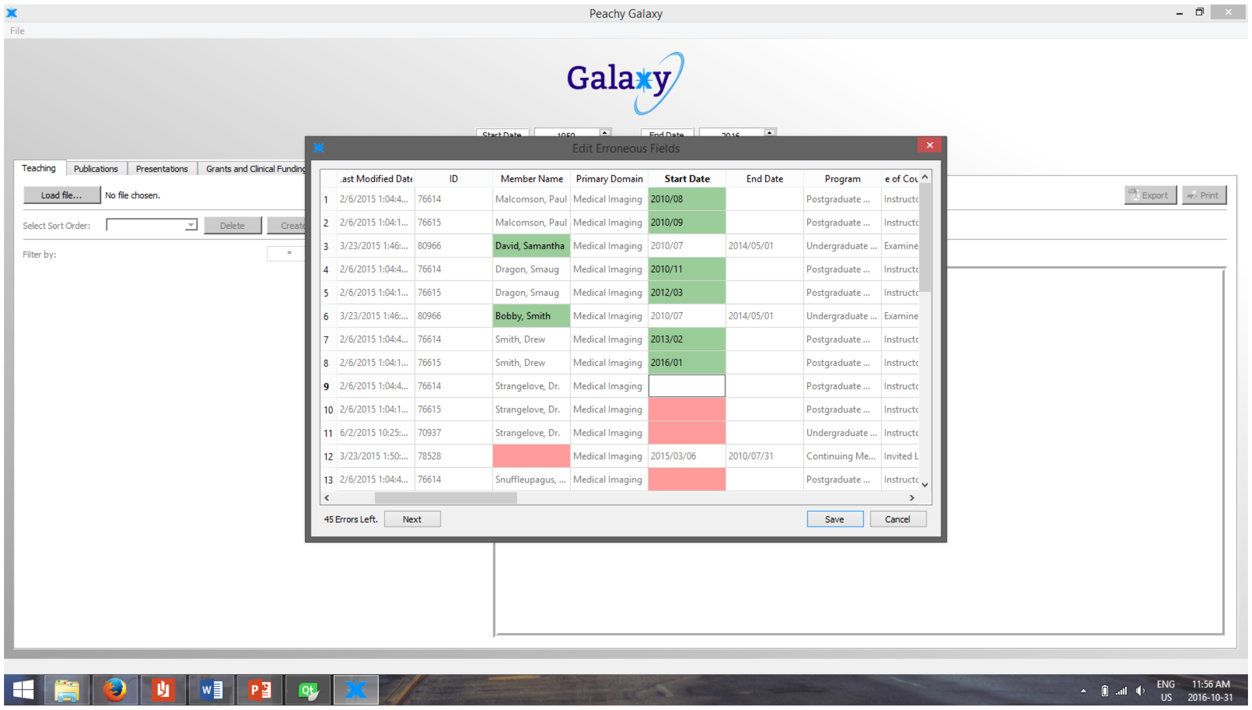
Select Sort Order: Default

Filter by: Member Name * to Z

Pie Chart [✓ Line Chart] Bar Chart

Member Name	# Presentations
AAbel, Angra (Test Record)	2
Adams, Paul	56
Aguanno, Alaina	2
Al-Judaibi, Bandar	30
Althenayan, Eyad	39
Anderson, Scott	17
Arbeau, Ryan	12
Arntfield, Robert	122
Asfaha, Samuel	8
Bagur, Rodrigo	61
Ball, Ian	40
Barnett, William	3
Barra, Lillian	44
Basharat, Pari	12
Beaton, Melanie	63
Bhimani, Munsif	11
Blake, Peter	352
Borrie, Michael	104
Bosma, Karen	77

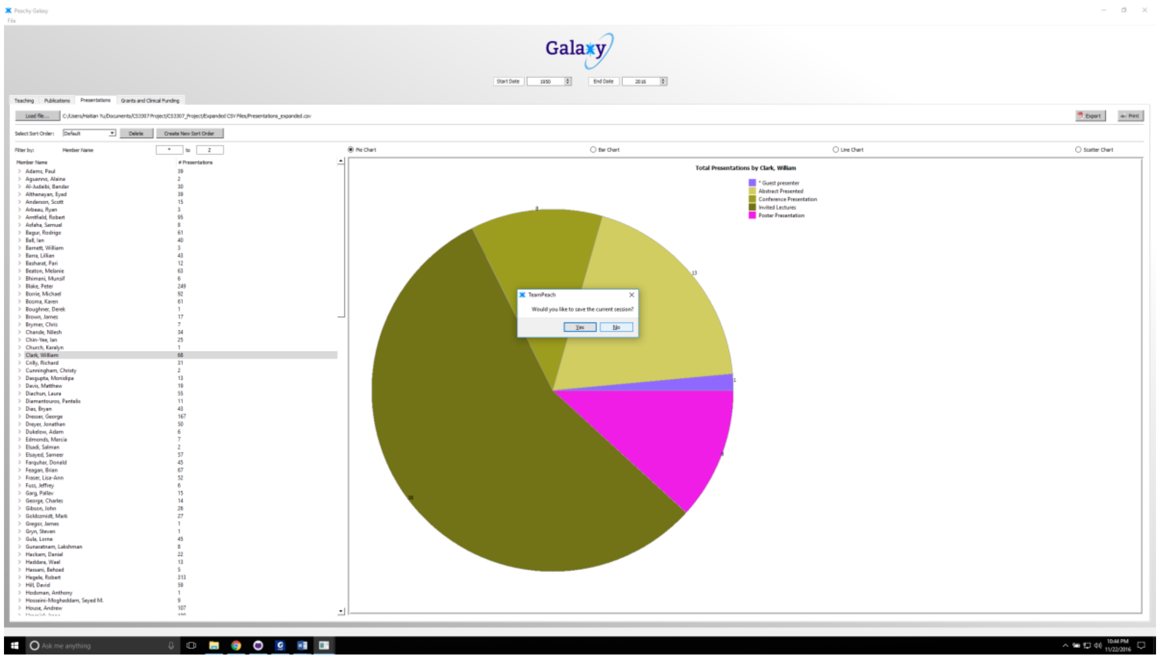
ID: MandatoryRequirement9
Description: Navigation of Erroneous Entries (Mandatory Requirement)
When the CSV file is loaded for the first time, the error dialog box should report total number of errors and "Find Next/Prev" button which jumps on to the next/prev error. As the errors are fixed, the error 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.



	.ast Modified Date	ID	Member Name	Primary Domain	Start Date	End Date	Program	e of Cou
1	2/6/2015 1:04:4...	76614	Malcomson, Paul	Medical Imaging	2010/08		Postgraduate ...	Instructo
2	2/6/2015 1:04:1...	76615	Malcomson, Paul	Medical Imaging	2010/09		Postgraduate ...	Instructo
3	3/23/2015 1:46:...	80966	David, Samantha	Medical Imaging	2010/07	2014/05/01	Undergraduate ...	Examine
4	2/6/2015 1:04:4...	76614	Dragon, Smaug	Medical Imaging	2010/11		Postgraduate ...	Instructo
5	2/6/2015 1:04:1...	76615	Dragon, Smaug	Medical Imaging	2012/03		Postgraduate ...	Instructo
6	3/23/2015 1:46:...	80966	Bobby, Smith	Medical Imaging	2010/07	2014/05/01	Undergraduate ...	Examine
7	2/6/2015 1:04:4...	76614	Smith, Drew	Medical Imaging	2013/02		Postgraduate ...	Instructo
8	2/6/2015 1:04:1...	76615	Smith, Drew	Medical Imaging	2016/01		Postgraduate ...	Instructo
9	2/6/2015 1:04:4...	76614	Strangelove, Dr.	Medical Imaging			Postgraduate ...	Instructo
10	2/6/2015 1:04:1...	76615	Strangelove, Dr.	Medical Imaging	🟥		Postgraduate ...	Instructo
11	6/2/2015 10:25:...	70937	Strangelove, Dr.	Medical Imaging	🟥		Undergraduate ...	Instructo
12	3/23/2015 1:50:...	78528	🟥	Medical Imaging	2015/03/06	2010/07/31	Continuing Me...	Invited L
13	2/6/2015 1:04:4...	76614	Snuffleupagus, ...	Medical Imaging	🟥		Postgraduate ...	Instructo

45 Errors Left. Next Save Cancel

ID: MandatoryRequirement3
Description: Ability to save session state (Mandatory Requirement)
When the user closes and reopens the application, all the data that was loaded into the program will be saved and re-opened again. The user does not have to re-upload the data.
Origin: Code created by Haitian Yu, Anthony Giugno, Claire Robichon
Supporting Example:



C:/Users/Haitian Yu/Documents/CS3307 Project/CS3307_Project/Expanded CSV Files/Presentations_expanded.csv

Filter by: Member Name * to 2

Member Name	# Presentations
Adams, Paul	39
Aguanno, Alaina	2
Al-Judaibi, Bandar	30
Althenayan, Eyad	39
Anderson, Scott	15
Arbeau, Ryan	3
Arntfield, Robert	95
Asfaha, Samuel	8
Bagur, Rodrigo	61
Ball, Ian	40
Barnett, William	3
Barra, Lillian	43
Basharat, Pari	12
Beaton, Melanie	63
Bhimani, Munsif	6
Blake, Peter	249
Borrie, Michael	92
Bosma, Karen	61
Boughner, Derek	1
Brown, James	17
Brymer, Chris	7
Chande, Nilesh	34
Chin-Yee, Ian	25
Church, Karalyn	1
Clark, William	68
Crilly, Richard	31
Cunningham, Christy	2
Dasgupta, Monidipa	13
Davis, Matthew	19
Diachun, Laura	55
Diamantouros, Pantelis	11
Dias, Bryan	43
Dresser, George	167
Dreyer, Jonathan	50
Dukelow, Adam	6
Edmonds, Marcia	7
Elsadi, Salman	2
Elsayed, Sameer	57
Farquhar, Donald	45
Feagan, Brian	67
Fraser, Lisa-Ann	52
Fuss, Jeffrey	6
Garg, Pallav	15
George, Charles	14
Gibson, John	26
Goldszmidt, Mark	27
Gregor, James	1
Gryn, Steven	1
Gula, Lorne	45
Gunaratnam, Lakshman	8
Hackam, Daniel	22
Haddara, Wael	13
Hassani, Behzad	5
Hegele, Robert	313
Hill, David	59
Hodsman, Anthony	1
Hosseini-Moghaddam, Seyed M.	9
House, Andrew	107

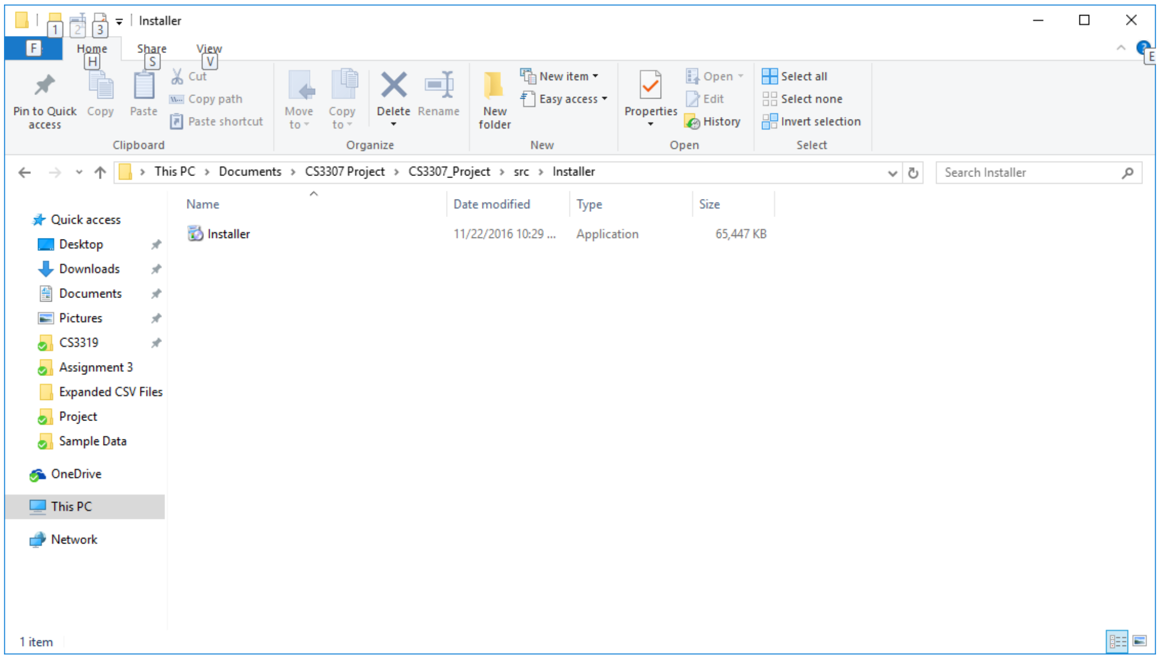
Would you like to save the current session?

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**

ID: MandatoryRequirement5
Description: Adding the ability to sort by a user selected list
Before, there was no option to sort by options other than member name, type, status date, role, or title. This will add the option to sort by anything the user wants.
Origin: Code created by Aboudi
Supporting Example: INCOMEPLETE

ID: MandatoryRequirement6
Description: Fix all errors so finished product is error free
Origin: Coded by all members of the team
Supporting Example: **INCOMPLETE**

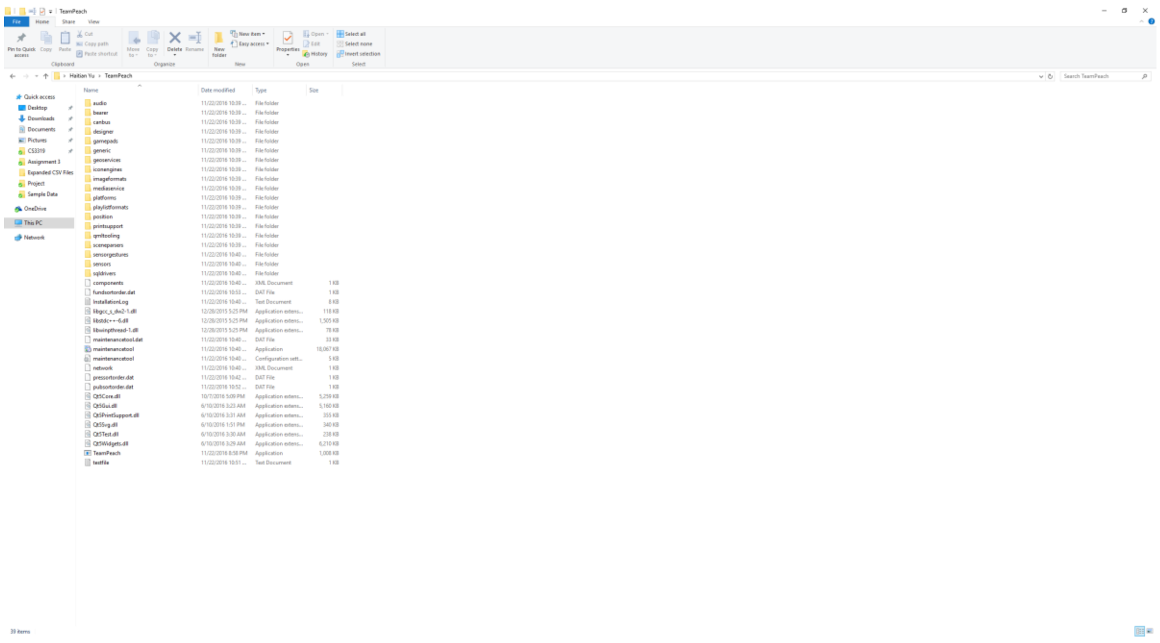
ID: MandatoryRequirement7
Description: Create a userproof install file that includes the executable file, supporting documentation, and all supporting files
Origin: Created by Anthony Giugno, Haitian Yu, and Claire Robichon
Supporting Example:



This PC › Documents › CS3307 Project › CS3307_Project › src › Installer

Name	Date modified	Type	Size
Installer	11/22/2016 10:29 ...	Application	65,447 KB

1 item



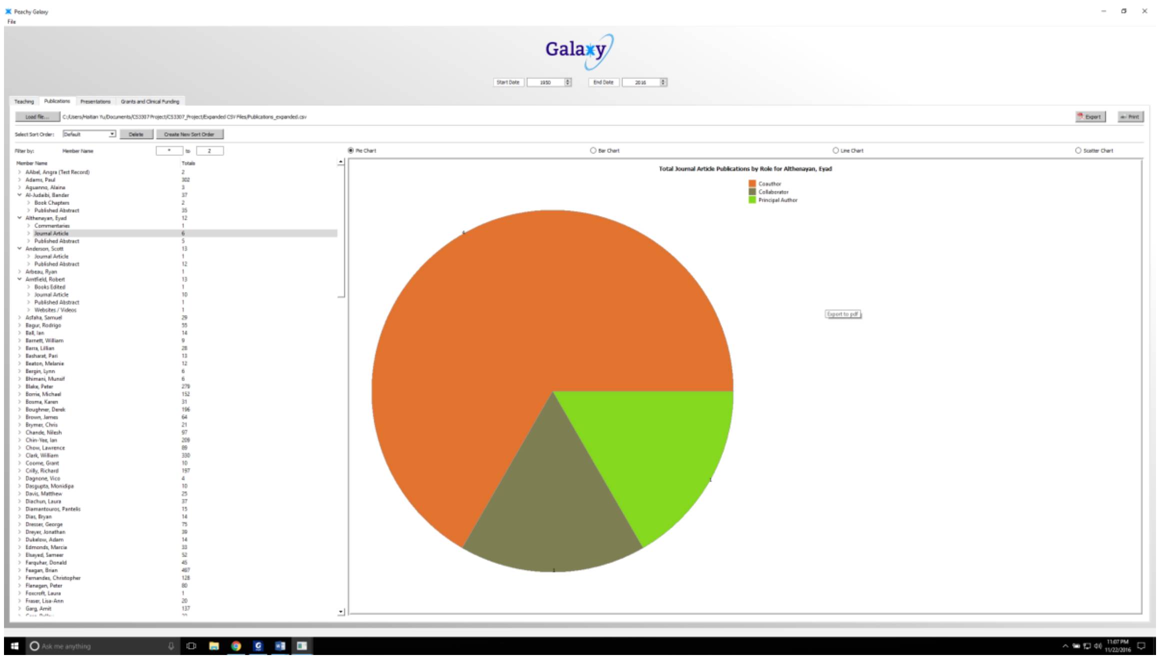
Haitian Yu › TeamPeach

Name	Date modified	Type	Size
audio	11/22/2016 10:39 ...	File folder	
bearer	11/22/2016 10:39 ...	File folder	
canbus	11/22/2016 10:39 ...	File folder	
designer	11/22/2016 10:39 ...	File folder	
gamepads	11/22/2016 10:39 ...	File folder	
generic	11/22/2016 10:39 ...	File folder	
geoservices	11/22/2016 10:39 ...	File folder	
iconengines	11/22/2016 10:39 ...	File folder	
imageformats	11/22/2016 10:39 ...	File folder	
mediaservice	11/22/2016 10:39 ...	File folder	
platforms	11/22/2016 10:39 ...	File folder	
playlistformats	11/22/2016 10:39 ...	File folder	
position	11/22/2016 10:39 ...	File folder	
printsupport	11/22/2016 10:39 ...	File folder	
qmltooling	11/22/2016 10:39 ...	File folder	
sceneparsers	11/22/2016 10:39 ...	File folder	
sensorgestures	11/22/2016 10:40 ...	File folder	
sensors	11/22/2016 10:40 ...	File folder	
sqldrivers	11/22/2016 10:40 ...	File folder	
components	11/22/2016 10:40 ...	XML Document	1 KB
fundsortorder.dat	11/22/2016 10:53 ...	DAT File	1 KB
InstallationLog	11/22/2016 10:40 ...	Text Document	8 KB
libgcc_s_dw2-1.dll	12/28/2015 5:25 PM	Application extens...	118 KB
libstdc++-6.dll	12/28/2015 5:25 PM	Application extens...	1,505 KB
libwinpthread-1.dll	12/28/2015 5:25 PM	Application extens...	78 KB
maintenancetool.dat	11/22/2016 10:40 ...	DAT File	33 KB
maintenancetool	11/22/2016 10:40 ...	Application	18,067 KB
maintenancetool	11/22/2016 10:40 ...	Configuration sett...	5 KB
network	11/22/2016 10:40 ...	XML Document	1 KB
pressortorder.dat	11/22/2016 10:42 ...	DAT File	1 KB
pubsortorder.dat	11/22/2016 10:52 ...	DAT File	1 KB
Qt5Core.dll	10/7/2016 5:09 PM	Application extens...	5,259 KB
Qt5Gui.dll	6/10/2016 3:23 AM	Application extens...	5,160 KB
Qt5PrintSupport.dll	6/10/2016 3:31 AM	Application extens...	355 KB
Qt5Svg.dll	6/10/2016 1:51 PM	Application extens...	340 KB
Qt5Test.dll	6/10/2016 3:30 AM	Application extens...	238 KB
Qt5Widgets.dll	6/10/2016 3:29 AM	Application extens...	6,210 KB
TeamPeach	11/22/2016 8:58 PM	Application	1,008 KB
testfile	11/22/2016 10:51 ...	Text Document	1 KB

39 items

ID: MandatoryRequirement8
Description: verify the code deck for operation on Windows 7 or Windows 10
Origin: Created by Anthony Giugno, Haitian Yu, and Claire Robichon
Supporting Example:

The following print screen shows the application working on a Windows 10 operating system. It also works on the windows 7 operating system as well.



Start Date 1950 End Date 2016

C:/Users/Haitian Yu/Documents/CS3307 Project/CS3307_Project/Expanded CSV Files/Publications_expanded.csv

Select Sort Order: Default

Filter by: Member Name * to 2

(●) Pie Chart

Total Journal Article Publications by Role for Althenayan, Eyad

🟧 Coauthor
🟫 Collaborator
🟩 Principal Author

Member Name	Totals
AAbel, Angra (Test Record)	2
Adams, Paul	302
Aguanno, Alaina	3
Al-Judaibi, Bandar	37
Book Chapters	2
Published Abstract	35
Althenayan, Eyad	12
Commentaries	1
Journal Article	6
Published Abstract	5
Anderson, Scott	13
Journal Article	1
Published Abstract	12
Arbeau, Ryan	1
Arntfield, Robert	13
Books Edited	1
Journal Article	10
Published Abstract	1
Websites / Videos	1
Asfaha, Samuel	29
Bagur, Rodrigo	55
Ball, Ian	14
Barnett, William	9
Barra, Lillian	28
Basharat, Pari	13
Beaton, Melanie	12
Bergin, Lynn	6
Bhimani, Munsif	6
Blake, Peter	279
Borrie, Michael	152
Bosma, Karen	31
Boughner, Derek	196
Brown, James	64
Brymer, Chris	21
Chande, Nilesh	97
Chin-Yee, Ian	209
Chow, Lawrence	89
Clark, William	330
Coome, Grant	10
Crilly, Richard	197
Dagnone, Vico	4
Dasgupta, Monidipa	10
Davis, Matthew	25
Diachun, Laura	37
Diamantouros, Pantelis	15
Dias, Bryan	14
Dresser, George	75
Dreyer, Jonathan	39
Dukelow, Adam	14
Edmonds, Marcia	33
Elsayed, Sameer	52
Farquhar, Donald	45
Feagan, Brian	497
Fernandes, Christopher	128
Flanagan, Peter	80
Foxcroft, Laura	1
Fraser, Lisa-Ann	20
Garg, Amit	137

11:07 PM 11/22/2016

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)
- New sorting methods
- Navigation of errors

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 Case ID:	001
Unit to test:	QSORT_LIST_01
Requirements:	3.1.1 Final System Delivery (Loading data from file)
Assumptions:	QSortListIO will catch and throw exception when given an invalid location.
Test Data:	"NO FILE"
Steps to be	Pass string to sorter, then call method .readList() and store

executed:	output as a QList
Expected Result:	QSortListIO success
Actual Result:	Debugger error
Pass/Fail:	Fail
Comments:	QSortListIO should contain a catch for invalid strings passed to the class.
Solution:	Design a try/catch statments for invalid files locations.

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

Test Case ID:	003
Unit to test:	QSORT_LIST_03

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

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

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

Test Case ID:	007
Unit to test:	Loading incorrect tab types
Requirements:	3.1.1 Final System Delivery (Loading data from file)
Assumptions:	Error thrown on wrong CSV tab type
Test Data:	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
Steps to be	1) Select a tab

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

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

Test Case ID:	009
Unit to test:	Clicking Export Button
Requirements:	- 3.1.7 Final System Delivery (Exporting to PDF)
Assumptions:	When clicking the Export Button Peachy galaxy should prompt me with the “Export File” dialog.
Test Data:	Program_Teaching_expanded.csv,
Steps to be executed:	1) Load file 2) Click export button
Expected Result:	“Export File” dialog box will open
Actual Result:	“Export File” dialog box will open
Pass/Fail:	Pass
Comments:	N/A
Solution:	N/A

Test Case ID:	010
Unit to test:	Clicking Print Button
Requirements:	3.1.6 Final System Delivery (Printing to file)
Assumptions:	When clicking the Print Button Peachy galaxy should prompt me with the “Print” dialog.
Test Data:	Program_Teaching_expanded.csv,
Steps to be executed:	1) Load file 2) Click print button
Expected Result:	“Print” dialog box will open
Actual Result:	“Print” dialog box will open
Pass/Fail:	Pass

Comments:	N/A
Solution:	N/A

Test Case ID:	011
Unit to test:	Clicking User Data
Requirements:	- 3.1.5 Final System Delivery (Visualizing data)
Assumptions:	When clicking a Member Name, their Data is visualized on the currently selected graph.
Test Data:	Program_Teaching_expanded.csv, Lavi, Shahar
Steps to be executed:	1) Load file 2) Click Lavi, Shahar
Expected Result:	Data is modeled on Pie Chart
Actual Result:	Data is modeled on Pie Chart
Pass/Fail:	Pass
Comments:	N/A
Solution:	N/A

Test Case ID:	012
Unit to test:	Clicking Bar Chart Button
Requirements:	- 3.1.5 Final System Delivery (Visualizing data)
Assumptions:	When a member name is selected and the Bar Chart Button is selected, a bar chart is produced
Test Data:	Program_Teaching_expanded.csv, Lavi, Shahar
Steps to be executed:	1) Load file 2) Click Lavi, Shahar 3) Click Bar Chart Button

Expected Result:	Data is modeled on Bar Chart
Actual Result:	Data is modeled on Bar Chart
Pass/Fail:	Pass
Comments:	N/A
Solution:	N/A

Test Case ID:	013
Unit to test:	Clicking Line Chart Button
Requirements:	- 3.1.5 Final System Delivery (Visualizing data)
Assumptions:	When a member name is selected and the Line Chart Button is selected, a line chart is produced
Test Data:	Program_Teaching_expanded.csv, Lavi, Shahar
Steps to be executed:	1) Load file 2) Click Lavi, Shahar 3) Click Line Chart Button
Expected Result:	Data is modeled on Line Chart
Actual Result:	Data is modeled on Line Chart
Pass/Fail:	Pass
Comments:	N/A
Solution:	N/A

Test Case ID:	014
Unit to test:	Entering g to h for Filter by: Member Name
Requirements:	- 3.1.5 Final System Delivery (Applying filters)
Assumptions:	When characters are entered into the fields for Filtering by: Member Name, all names between the characters are displayed.

Test Data:	Program_Teaching_expanded.csv,
Steps to be executed:	1) Load file 2) Enter g into first box and h into next box
Expected Result:	All entries between g and h are displayed
Actual Result:	All entries between g and h are displayed
Pass/Fail:	Pass
Comments:	N/A
Solution:	N/A
Test Case ID:	015
Unit to test:	Entering g to f for Filter by: Member Name
Requirements:	- 3.1.5 Final System Delivery (Applying filters)
Assumptions:	When characters are entered into the fields for Filtering by: Member Name, all names between the characters are displayed.
Test Data:	Program_Teaching_expanded.csv,
Steps to be executed:	1) Load file 2) Enter g into first box and h into next box
Expected Result:	No entries are displayed
Actual Result:	No entries are displayed
Pass/Fail:	Pass
Comments:	N/A
Solution:	N/A

Test Case ID:	016
Unit to test:	Entering 1 to 9 for Filter by: Member Name
Requirements:	- 3.1.5 Final System Delivery (Applying filters)
Assumptions:	When characters are entered into the fields for Filtering by: Member Name, all names between the characters are displayed.
Test Data:	Program_Teaching_expanded.csv,
Steps to be executed:	1) Load file 2) Enter g into first box and h into next box
Expected Result:	All entries are displayed
Actual Result:	All entries are displayed
Pass/Fail:	Pass
Comments:	N/A
Solution:	N/A

Test Case ID:	017
Unit to test:	Entering a to z for Filter by: Member Name
Requirements:	- 3.1.5 Final System Delivery (Applying filters)
Assumptions:	When characters are entered into the fields for Filtering by: Member Name, all names between the characters are displayed.
Test Data:	Program_Teaching_expanded.csv,
Steps to be executed:	1) Load file 2) Enter g into first box and h into next box
Expected Result:	All entries are displayed
Actual Result:	Entries starting with lower case are displayed last
Pass/Fail:	Fail

Comments:	As the box parameters are not case sensitize, the program should display the names in alphabetical order regardless of case to avoid confusion.
Solution:	Convert names to lower case before filtering.

Test Case ID:	018
Unit to test:	Entering 1999 to 2015 for filtering by date
Requirements:	- 3.1.5 Final System Delivery (Applying filters)
Assumptions:	When characters are entered into the fields for filtering by date, all names between the dates are displayed.
Test Data:	Program_Teaching_expanded.csv,
Steps to be executed:	1) Load file 2) Enter g into first box and h into next box
Expected Result:	All entries between the dates are displayed
Actual Result:	All entries between the dates are displayed
Pass/Fail:	Pass
Comments:	N/A
Solution:	N/A

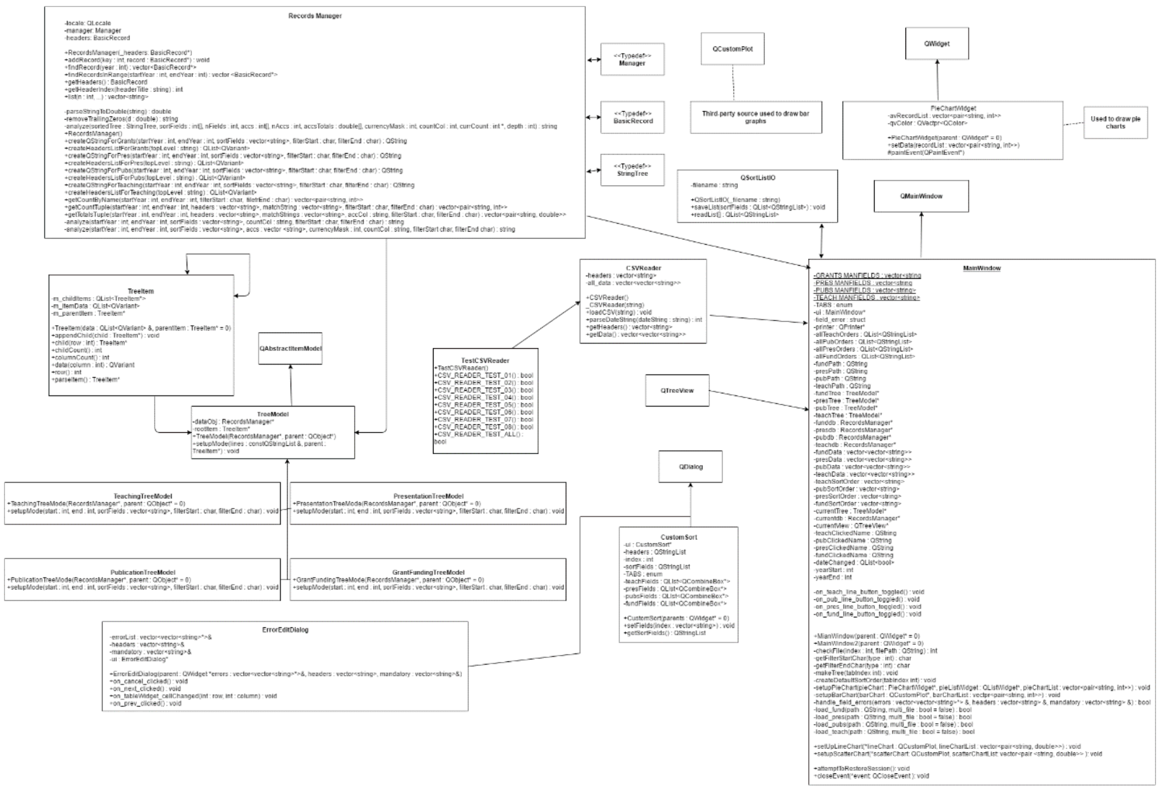
Test Case ID:	019
Unit to test:	Entering 2015 to 2014 for filtering by date
Requirements:	- 3.1.5 Final System Delivery (Applying filters)
Assumptions:	When characters are entered into the fields for filtering by date, all names between the dates are displayed.
Test Data:	Program_Teaching_expanded.csv,
Steps to be	3) Load file

executed:	1) Enter g into first box and h into next box
Expected Result:	All entries between the dates are displayed, as the last date cannpt be entered
Actual Result:	All entries between the dates are displayed, as the last date cannpt be entered
Pass/Fail:	Pass
Comments:	N/A
Solution:	N/A

Diagrams

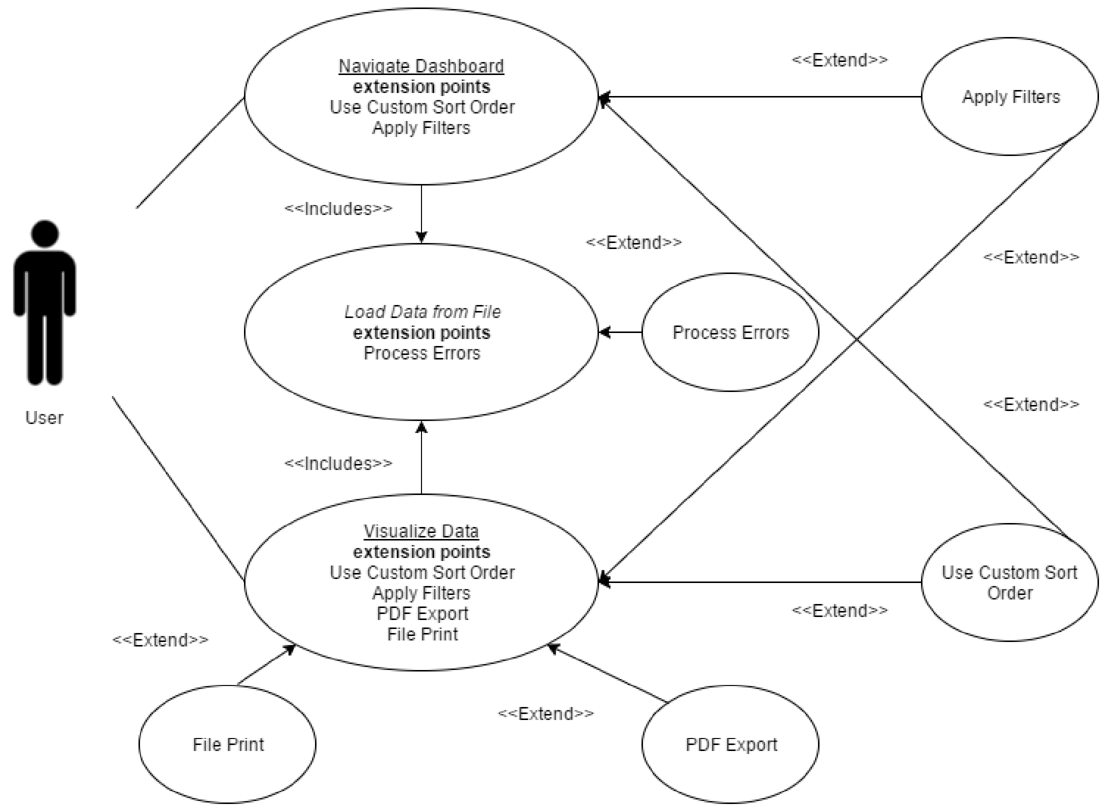
Class Diagram Description:

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.



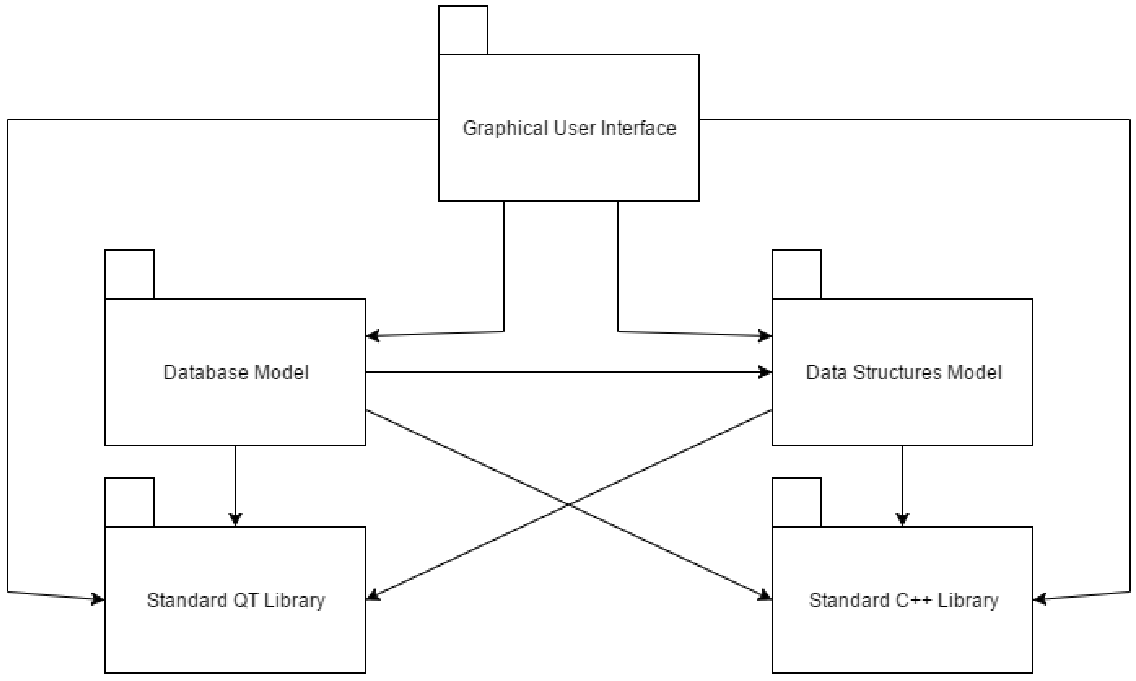
Case Diagram Description:

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.



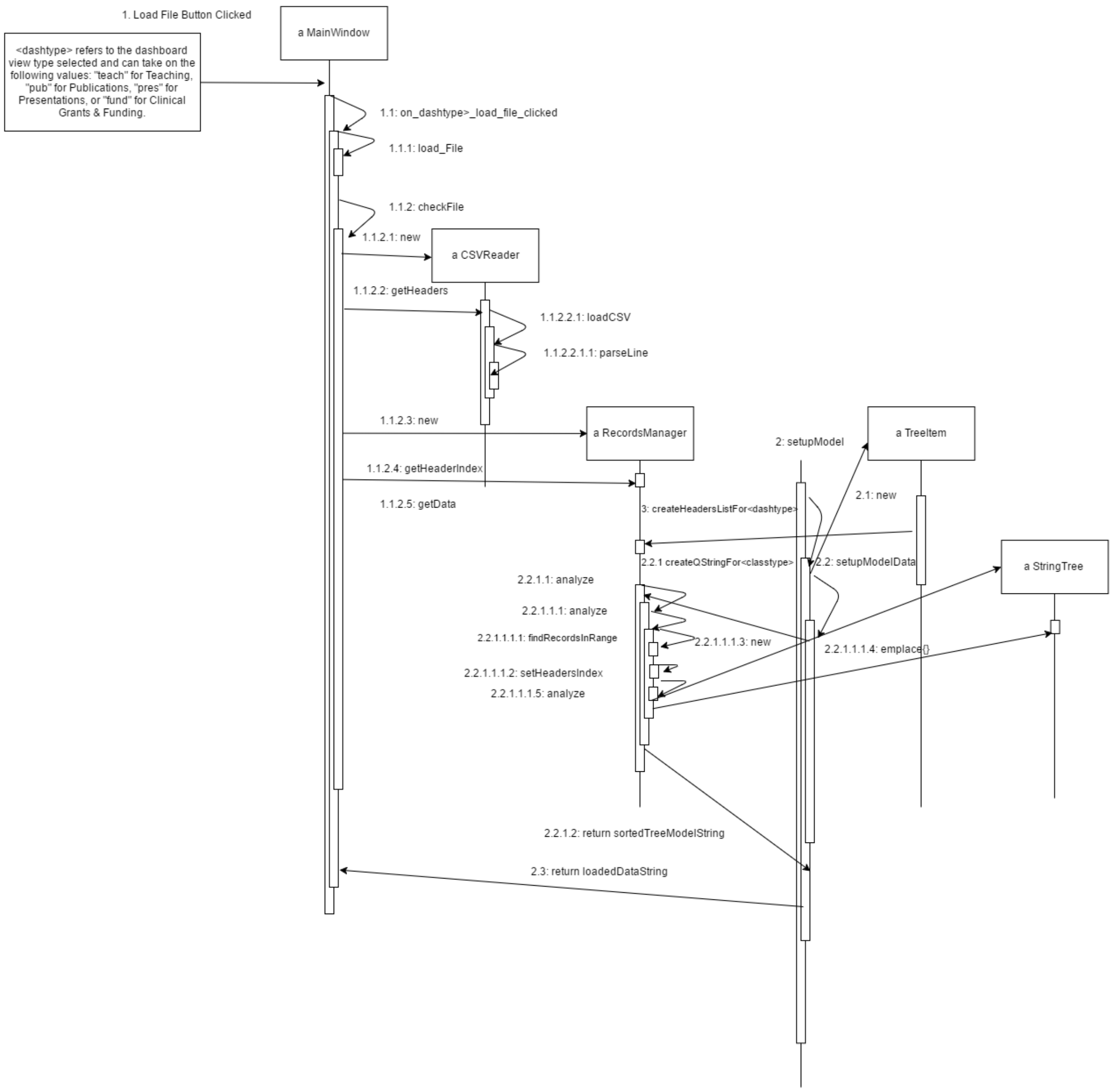
Package Diagram:

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.



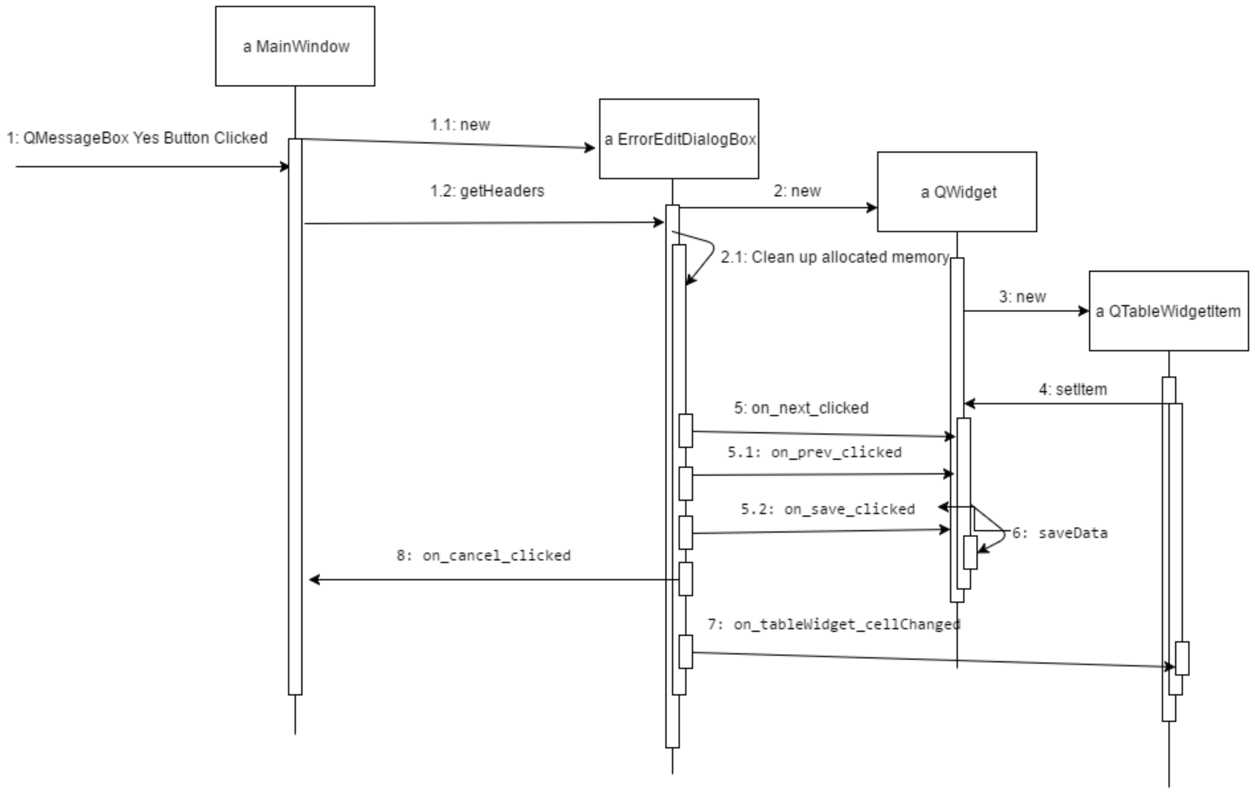
User Clicks Load Button Sequence Diagram:

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”.


1. Load File Button Clicked
a MainWindow
<dashtype> refers to the dashboard view type selected and can take on the following values: "teach" for Teaching, "pub" for Publications, "pres" for Presentations, or "fund" for Clinical Grants & Funding.
1.1: on_dashtype>_load_file_clicked
1.1.1: load_File
1.1.2: checkFile
1.1.2.1: new
a CSVReader
1.1.2.2: getHeaders
1.1.2.2.1: loadCSV
1.1.2.2.1.1: parseLine
1.1.2.3: new
a RecordsManager
1.1.2.4: getHeaderIndex
1.1.2.5: getData
2: setupModel
a TreeItem
2.1: new
3: createHeadersListFor<dashtype>
2.2.1 createQStringFor<classtype>
2.2: setupModelData
a StringTree
2.2.1.1: analyze
2.2.1.1.1: analyze
2.2.1.1.1.1: findRecordsInRange
2.2.1.1.1.3: new
2.2.1.1.1.4: emplace()
2.2.1.1.1.2: setHeadersIndex
2.2.1.1.1.5: analyze
2.2.1.2: return sortedTreeModelString
2.3: return loadedDataString

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.

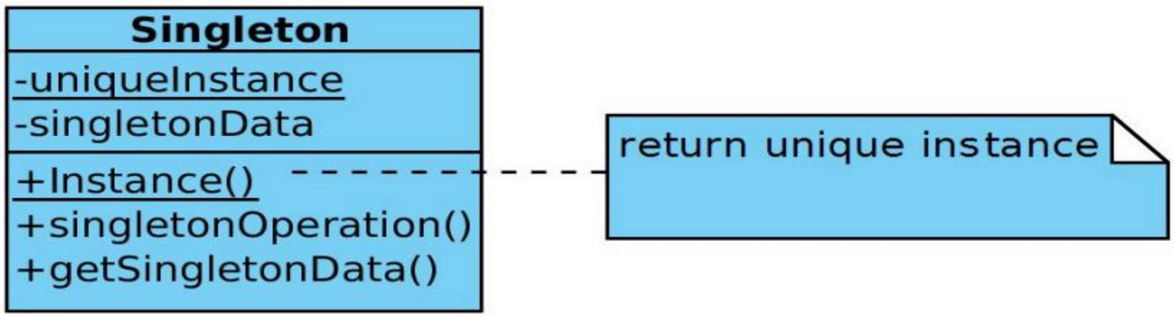


Design Patterns:

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++:

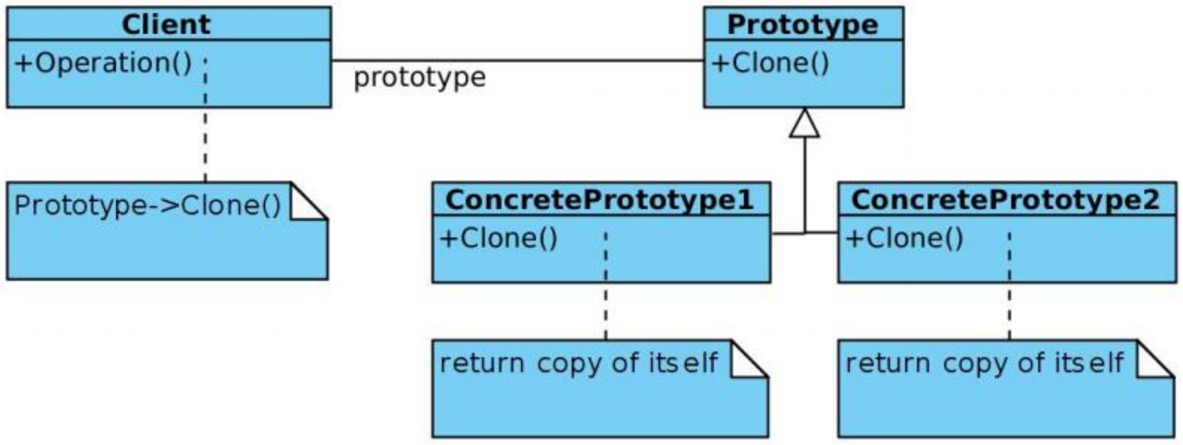
Singleton design is used in the implementation of MainWindow (controls the graphical user interface). The additions we have made to this class are the new graph functions setupScatterChart and setupLineChart which continue to follow the singleton design pattern.

Sources:

https://www.tutorialspoint.com/design_pattern/singleton_pattern.htm
https://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://www.tutorialspoint.com/design_pattern/prototype_pattern.htm

Agent Task View

	Haitian	Yusi	Claire	Ryan	Aboudi	Matthew	Anthony
Deliverable 2							
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	
Deliverable 3							
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						
Deliverable 4							
Training Video	x						
Sequence Diagram						x	
Package Diagram						x	
Design Patterns	x						
Implementation in C++	x						
Test Cases for Stage 2 Deliverables				x		x	
Mandory 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						
Deliverable 5							
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 19th) 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	
	Recommendations for Fixing Errors	
10/26/2016	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	
	Write Up Implementation in C++	
11/7/2016	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	
	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	
11/11/2016	Test cases created matrix	Aboudi
	Progress on diagrams	Anthony
	Save state ideas plan created	
	- Will begin coding for this week	
	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	
	Sequence diagram almost completed	
	- Waiting on sort code	
	Sort code will be done by Monday	
	Still To Do	
	User proof install	
	Test cases for deliverables	
	- Save state	
	- 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	
	Figuring out how to create an install package	
Stage 3 Achievements		
	Verification that application works on Windows 7 completed	
	Scatterplot graph completed	
	Installer completed	
	Test code for deliverables mostly completed	
	Project Timeline Up To Date	
	Agent Task Up To Date	
	Save state completed	
	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	