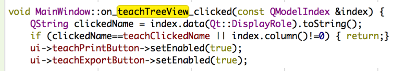
Test Case Id	Unit to Test	Requirements	Assumptions	Test data	Steps to be executed	Expected result	Actual result	Pass/Fail	Comments
1: Test Loading Data from a file									
1-1	CSVReader.cpp	Reads Grants and Clinical Funding CSV files successfully	The file header will not be empty	GrantsClinicalFunding_ sample_sample.csv	1. Read the file using CVSReader and 2. Check if the header is not empty. If the header is not empty we know that it has been read correctly	The header will not be empty	The header was not empty	**Pass**	
1-2	CSVReader.cpp	Reads Presentations CSV files succesfully	The file header will not be empty	Presentations_sample. csv	1. Read the file using CVSReader and 2. Check if the header is not empty. If the header is not empty we know that it has been read correctly	The header will not be empty	The header was not empty	**Pass**	
1-3	CSVReader.cpp	Reads Publications CSV files successfully	The file header will not be empty	Publications_sample. csv	1. Read the file using CVSReader and 2. Check if the header is not empty. If the header is not empty we know that it has been read correctly	The header will not be empty	The header was not empty	**Pass**	
1-4	CSVReader.cpp	Reads Teaching CSV files successfully	The file header will not be empty	Teaching_sample.csv	1. Read the file using CVSReader and 2. Check if the header is not empty. If the header is not empty we know that it has been read correctly	The header will not be empty	The header was not empty	**Pass**	
1-5	CSVReader.cpp	Does not read in a file that does not exist	The file path we are attempting to load does not exist	"thisfiledoesnotexist.cvs" (this file does not exist in the working directory)	1. Read the file using CVSReader and 2. Check if the headers are still empty. If the headers are still empty we know that the file was not read.	The file was not read.	The file was not read.	**Pass**	
1-6	CSVReader.cpp	Does not read in a invalid file type		invaliddata.txt	1. Read the file using CVSReader and 2. Check if the headers are still empty. If the headers are still empty we know that the file was not read.	The file was not read.	libc++abi.dylib: terminating with uncaught exception of type std::out_of_range: basic_string The program has unexpectedly finished.	**Fail**	Fixed Below
1-6 BUGFIX	CSVReader.cpp	Does not read in a invalid file type		invaliddata.txt	1. Read the file using CVSReader and 2. Check if the headers are still empty. If the headers are still empty we know that the file was not read.	The file was not read.	The file was not read.	**Pass**	The previous code was not checking the file extension. Just if the file exists. This means selecting any file other then a .csv crashed the program. This was fixed by checking the file extension before loading the file
2: Test Displaying a summary of data									
2-1	mainwindow.cpp	Grants data is displayed on screen		GrantsClinicalFunding_ sample_sample.csv	1. Go to the grants tab. 2. Press Load file and select a grants CVS. 3. Load the file and discard any errors.	The summarized data should be displayed for the user to look at	The summerized data was displayed	**Pass**	
2-2	mainwindow.cpp	Presentations data is displayed on screen		Presentations_sample. csv	1. Go to the presentations tab tab. 2. Press Load file and select a presentations CVS. 3. Load the file and discard any errors.	The summarized data should be displayed for the user to look at	The summerized data was displayed	**Pass**	
2-3	mainwindow.cpp	Publications data is displayed on screen		Publications_sample. csv	1. Go to the publications tab. 2. Press Load file and select a publications CVS. 3. Load the file and discard any errors.	The summarized data should be displayed for the user to look at	The summerized data was displayed	**Pass**	
2-4	mainwindow.cpp	Teaching Data is displayed on screen		Teaching_sample.csv	1. Go to the teaching tab. 2. Press Load file and select a teaching CVS. 3. Load the file and discard any errors.	The summarized data should be displayed for the user to look at	The summerized data was displayed	**Pass**	
3: Test visualizing/graphing of data									
3-1	mainwindow.ui / piechartwidgit.cpp	Check that the pie chart is displayed for all four subject areas	Pie charts will be displayed	PieChartWidgit	For each subject area select pie chart option, select a section, see if pie chart is displayed	The pie chart will be displayed	The pie chart was displayed	**Pass**	
3-2	qcustomplot.cpp	Check that the bar chart is displayed for all four subject areas			For each subject area select bar chart option, select a section, see if pie chart is displayed	The bar chart will be displayed	The bar chart was displayed	**Pass**	
4: Test dashboard navigation									
4-1	mainwindow.ui	Expand button expands sections and collapse button collapses sections		teachTreeView, pubTreeView, presTreeView, fundTreeView	1. Load a file. 2. Expand a section in the tree view. 3. Collapse a section on the tree view. Repeat for all 4 file types	The section will expands then collapse	The sections expanded and then collapsed	**Pass**	
4-2	mainwindow.ui	The user can use the tab bar to navigate to different tabs for each of the 4 subjects. Teaching, Publications, Presentations and Funding		categoryTab	1. Click on the Grants tab to navigate to it. 2. Click on the Presentations Tab to navigate to it. 3. Click on the presentations tab to navigate to it 4. Click on the teaching tab to navgation tab	You can navigate to all the tabs	You can navigate to all the tabs	**Pass**	
5: Test filtering and sorting data									
5-1	mainwindow.cpp	Changing a item in the filter_from box filters the data		teach_filter_from, pub_filter_from, pres_filter_from, fund_filter_from	1. Change text in the filter_from box. 2. Check if the data has been filtered	The change in the filter_from box will trigger a textChanged() event and cause the tree view to be update to reflect the filter	The data was filtered	**Pass**	
5-2	mainwindow.cpp	Chaning an item in the filter_to box filters the data		teach_filter_to, pub_filter_to, pres_filter_to, fund_filter_to	1. Change text in the filter_to box 2. Check if the data has been filtered	The change in the filter_to box will trigger a textChanged() event and cause the tree view to be update to reflect the filter	The data was filtered	**Pass**	

```
QString qfile_name = QString::fromUtf8(file_name.c_str());
if (!qfile_name.endsWith("csv")) {
    return;
}
```

Test Case Id	Unit to Test	Requirements	Assumptions	Test data	Steps to be executed	Expected result	Actual result	Pass/Fail	Comments
5-3	customsort.cpp	Allow the user to choose how to sort the data on screen		Teaching_sample.csv	1. Load a data file. 2. Create a custom sort. 3. Change the sort order to the custom sort	The data will be sorted	The data was sorted	Pass	If you attempt to create a sort with the Start Date first it will tell you you cannot sort by date first. While this is not a bug it is not very user friendly. This should be added to our "improvements" and we should prevent users from being able to sort by date first for a future deliverable.
6: Test PDF and Print Export									
6-1	mainwindow.cpp	You can export a pie chart as a pdf		Teaching_sample.csv	1. Load a data file. 2. Select a pie chart. 3. Export as a pdf	The chart will be exported as a pdf	The chart was exported	Pass	
6-2	mainwindow.cpp	You can export a bar chart as a pdf		GrantsClinicalFunding_sample.csv	1. Load a data file. 2. Select a bar chart. 3. Export as a pdf	The chart will be exported as a pdf	The char was exported	Pass	
6-3	mainwindow.cpp	You cannot export a blank pdf		Teaching_sample.csv	1. Load a data file. 2. Before selecting a chart press the export button.	The program will not let you export a empty pdf	The empty pdf was exported	Fail	If you export a PDF before selecting a data value you can export a empty PDF. Fixed Below
6-3: BUGFIX	mainwindow.cpp	You cannot export a blank pdf		Teaching_sample.csv	1. Load a data file. 2. Before selecting a chart press the export button.	The program will not let you export a empty pdf	The program will not let you export a empty pdf	Pass	Before the print buttons were being enabled as soon as data is imported to the program. This was fixed by only enabling the print buttons once they have selected data to print using "teachTreeView_clicked"
7: Testing Tree data structure									
7-1	treeitem.cpp void appendChild (TreeItem *child);	You can insert items into a tree	If the number of children is equal to the number of children appended to the TreeItem the insert is working correctly	TreeItem parent item, child1, child2, child3	1. Create all 4 TreeItems using test data. 2. Set the child items to the parent item. 3. Check the number of children. Compare the number of children with the number of children appended to the tree.	The number of children inserted should be equal to the number of children	The number of children inserted was equal to the number of children	Pass	
7-2	treeitem.cpp TreeItem *parentItem();	You can get the parent from a tree		TreeItem parent item, child1, child2, child3	1. Create all 4 Tree items user test data. 2. Set the child items to the parent items. 3. Check if the child nodes parent is equal to the parent node	The child nodes parent should be equal to the parent node	The child nodes parent should be equal to the parents node	Pass	
7-3	treeitem.cpp TreeItem *child (int row);	You can get the children from a tree		TreeItem parent item, child1, child2, child3	1. Create all 4 Tree items user test data. 2. Set the child items to the parent items. 3. Check if the children node is equal to the parents children	The parent nodes child should be equal to the child node	The parent nodes child should be equal to the child node	Pass	
8: Test Error Processing									
8-1	EditErrorDialog.cpp	If a data file contains any invalid records the program will propt the user to edit or discard them	The datafile we are testing with contains at least 1 invalid record	Publications_sample.csv	1. Load a data file that contains an invalid record	The program showns the number of invalid records andpromps the user to edit or discard the invalid record	The program showns the number of invalid records andpromps the user to edit or discard the invalid record	Pass	
8-2	EditErrorDialog.cpp	If the user presses the button to edit invalid records the user will be brought to a screen to fill in the missing data in all invalid records.	The datafile we are testing with contains at least 1 invalid record	Publications_sample.csv	1. Load a data file that contains an invalid record 2. Press the edit button in the popup	The program displays the valid records along with the invalid that was corrected	The program displays the valid records along with the invalid that was corrected	Pass	
8-3	EditErrorDialog.cpp	If the user fills in the missing data and presses save. The program will display the valid data and the invalid data that has been corrected	The datafile we are testing with contains at least 1 invalid record	Publications_sample.csv	1. Load a data file that contains an invalid record 2. Press the edit button in the popup 3. Fill in the missing data. 4. Press the "save" button	The program displays the valid records along with the invalid that was corrected	The program displays the valid records along with the invalid that was corrected	Pass	
8-4	EditErrorDialog.cpp	If the user presses the button to discard invalid records. The program will discard these records and display the remaning data	The datafile we are testing with contains at least 1 invalid record	Publications_sample.csv	1. Load a data file that contains an invalid record 2. Press the "discard" button	The program discards the invalid records and only displays the valid records	The program discards the invalid records and only displays the valid records	Pass	
9: Testing QSortListIO									
9-1	QSortListIO class	The class should be able to save a QList of QStringLists to a file and then read the QList back out of the file.	The class will have the functionality to save a QList to a file and read it back out, the QList that we read from the file should be identical to our original QList. We will use QSortListIO::saveList (QList<QStrings>) to save our test QList to a file and then use QSortListIO::readList() to read it back out.	Made a test QList<QStringList>	1. Create a QList<QStringList> 2. Create a QSortLIstIO object 3. Use QSortListIO::saveList() to save to file 4. UseQSortListIO::readList() to retrieve the list 5. Compare the list to the original to make sure it did not get corrupted in the process	The List returned by readList() will be the same as the original	The Lists were identical	Pass	



```
void MainWindow::on_teachTreeView_clicked(const QModelIndex &index) {
    QString clickedName = index.data(Qt::DisplayRole).toString();
    if (clickedName==teachClickedName || index.column()!=0) { return;}
    ui->teachPrintButton->setEnabled(true);
    ui->teachExportButton->setEnabled(true);
```