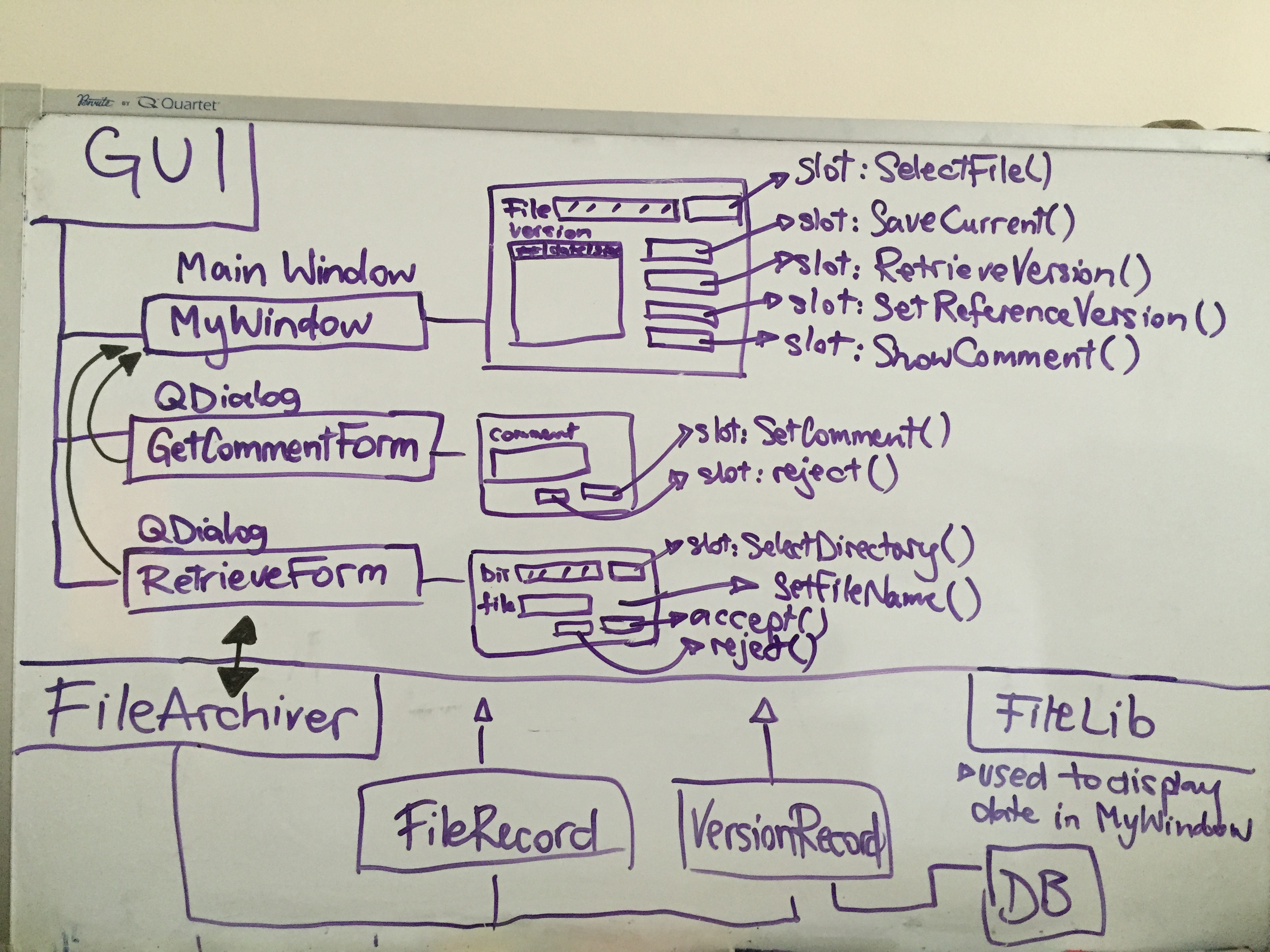
**GUI**

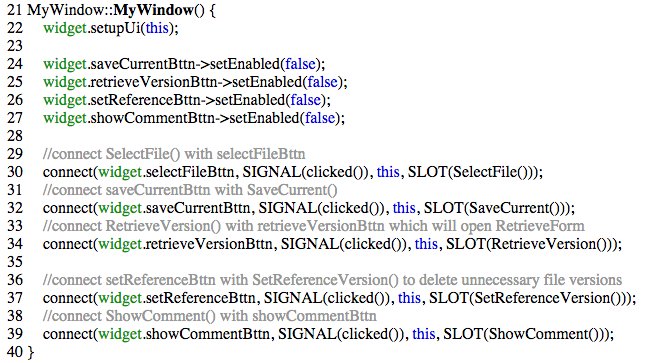
In the process of getting the GUI started, the initial task was to create the main window of the program and its dialogs using Qt Designer, which generated all necessary files for the GUI elements in the NetBeans project.

Next thing on the agenda was to set up all needed slot functions and connect them to the elements of the GUI.

After a discussion with other implementers to understand how different components of the program will interact with each other, diagram was drawn on the whiteboard to provide and overview which elements will contain each functionality, and how it will interact with rest of the program.



The following image is the example of how the diagram shown was used to start off the MyWindow class.



From the above whiteboard diagram it can be seen that the GUI will be communicating mostly with FileArchiver class, but also using FileRecord and VersionRecord class through FileArchiver.

This was a helpful way to start implementing the functionality of the slot functions. First slot function needed to be defined was SelectFile(), as all other functionality of the program is based on the file that the user has selected to initiate the file or retrieve version of the chosen file from the persistent storage.

As the backend code was initially tested separately from the GUI functionality, the implementation with the GUI was pretty straight forward, as the test code provided insight of the usage of different class objects and how they interact.

The most time was spent on the decision which model will be used to display retrieved file versions in the table view. In order to find an easier and more concise solution than writing our own QAbstractTableModel class, we have researched the abilities of QtCore classes and decided to use QAbstractItemModel class. QAbstractItemModel supplied all the functionality required for our program.

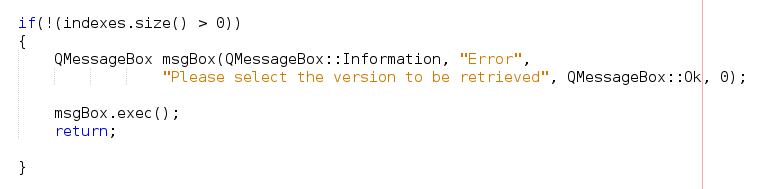
The following screenshot shows the usage of table view and QAbstractItemModel to retrieve information for a selected version of the file by wrapping the RetrieveVersionDataForFile function in MyWindow class.



After all functionality was implemented, we focused on making sure that the control flow and error-handling logic of the program solves each issue where users can make erroneous interactions through the GUI.

Each possible user interaction case was considered and we made sure to inform users how to achieve what was intended by supplying information in form of different kinds of message dialogs.

The screenshot below shows how this was implemented in the case where the user tries to retrieve a version of a file, but has not selected the version to be retrieved from the display in the table view.



Finally, after GUI was completely functional and there were no bugs, we made sure everyone was happy with the final product.