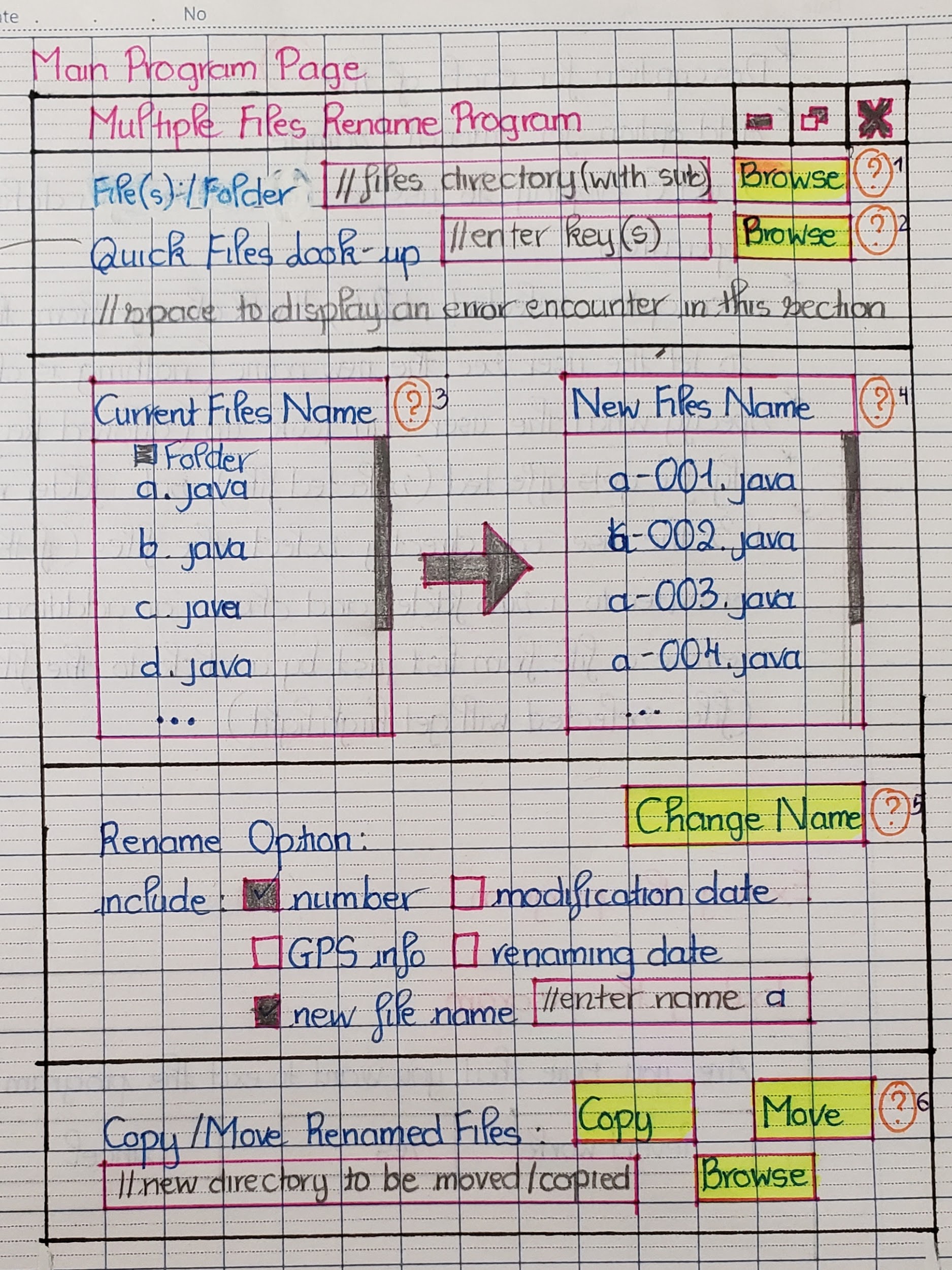
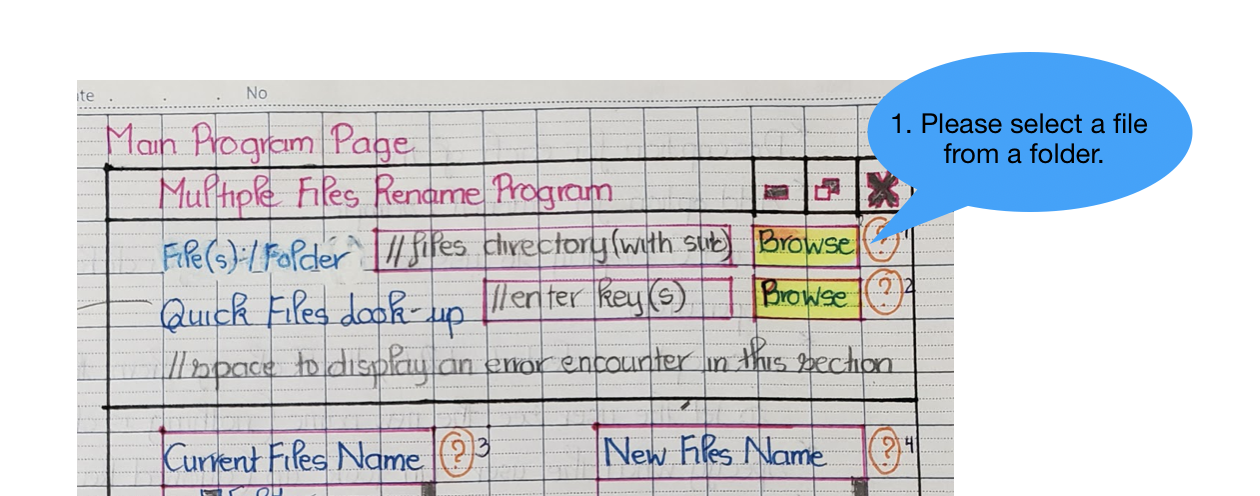
**Assignment 2**

**Group Member: Tyrell Unser, Bowei Yao, Khai Phan, Mamadou Diallo**

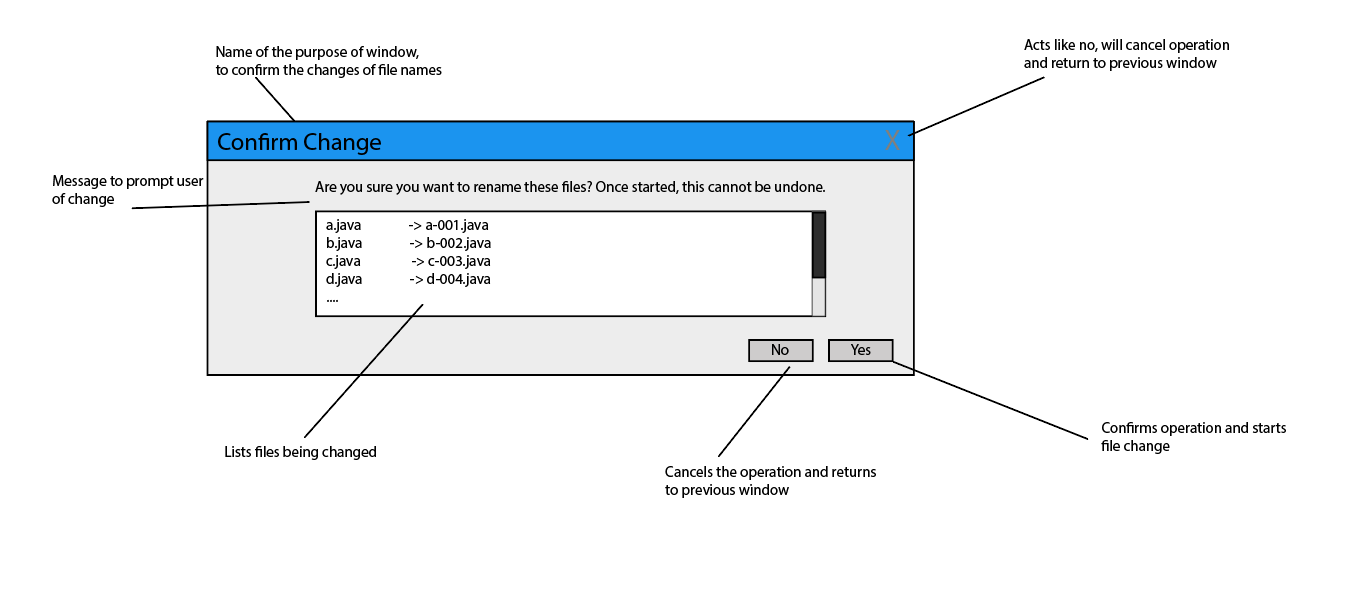


*Image 1: Main user interface*

This will be the main interface of the program. On the top left the program has the name of “**Multiple Files Rename Program**” so the user won’t get confused with other similar running programs (such as File Rename Program or Multi-file Rename Program). To begin and facilitate understanding for a user we’ve incorporated small pop ups that appear when a user hovers over an annotated question mark. 

In the first section, the user will need to specify the **folder containing the file(s) or folder₁** by clicking on the browse. After selecting a number of files or a folder, the user can sort and filter them by name. For example, in the **Quick Files Look-up₂**, if the user enters A, all the files have the letter A on it will be filtered and shown in the next section. So for this, the more detail the user can provide, the more accurate the results. The filter is capable of accepting regular expressions as a form of input.

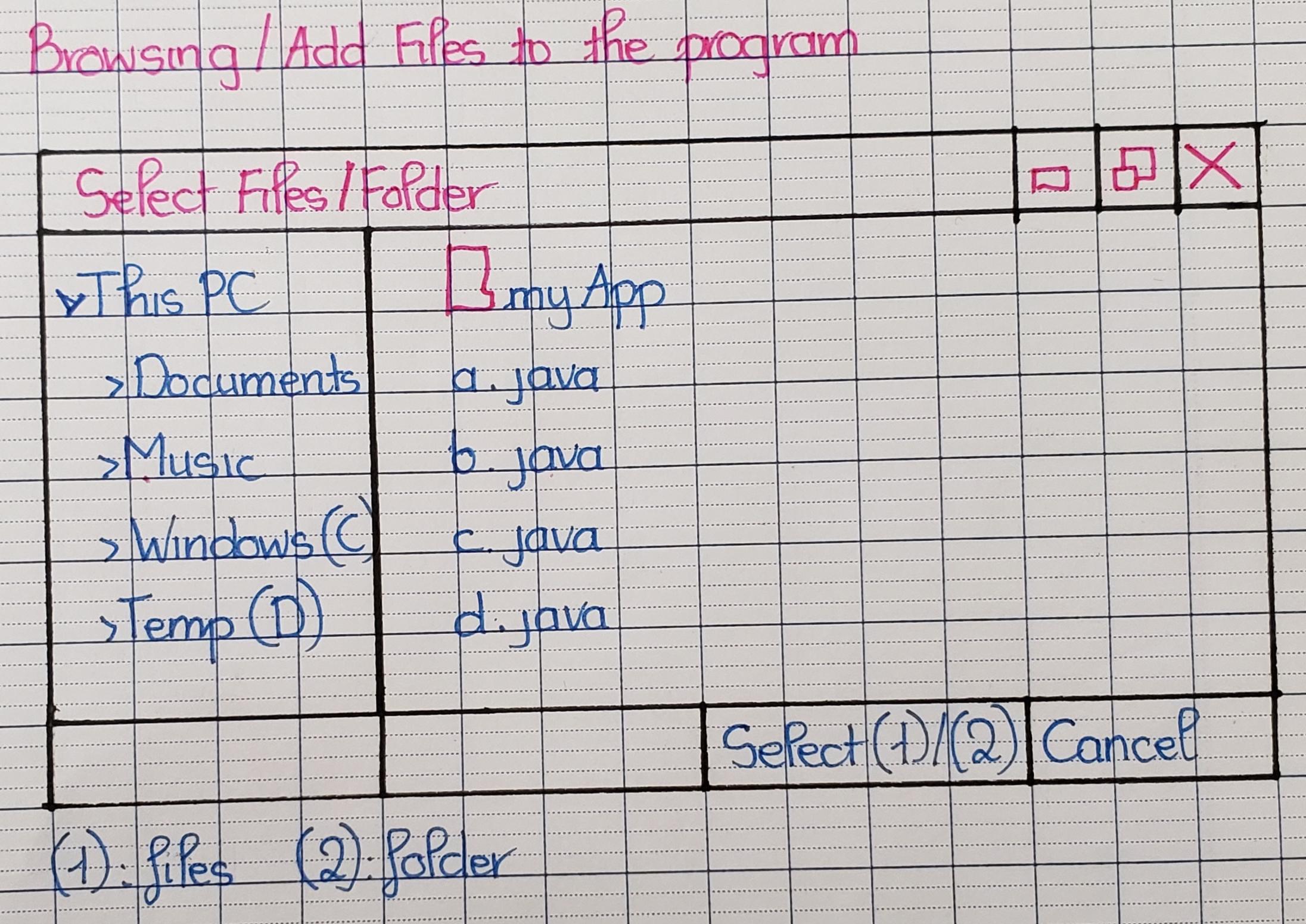
The section immediately below is for display and user feedback. On the left will be **Current File Name₃** which are the files the user has selected and/or filtered. Selected files will be highlighted (with background blue colour). This will also display sub-folder (if available) for a user to go in there and select files that could be in a separate folder. Again, those files selected in the subfolder will be highlighted blue so the user can distinguish from other files. If a user changes their mind, they can unselect the file by clicking on the file to remove the highlight. On the right side is the **New File Name₄** section, which displays what the new filenames will be after the user selected in the filename option in the next section. Keep in mind that it is just the temporary display for the rename option to let the user see what their new filename will be before they can make the operation.   
 The next section is the Rename Options which give the user plenty of options such as adding a sequence number, last modification date or GPS info to the file. A user can also define their custom file names. Keep in mind that multiple options can be stacked here, so a user can define their custom file names plus the plus a sequence number. If the file does not have the metadata for an option (such as GPS) the checkbox will be greyed out. When the user has decided on the new file name, they can click on the Change Name option and a new pop-up will be shown, ask the user to confirm their actions. A warning message will also be displayed in the confirmation, telling the user that once it is started, they cannot undo the work. These elements are meant to make it more difficult for the user to commit errors.



*Image 2: Confirmation window*

After the files have been modified, the user will have the additional option to either copy or move the newly changed files to another folder. Note that the file and folder structures will remain exactly the same as they were from the source. If a sub-folder is required to be created in advance, the program will ask the user if the program can automatically create the folder.

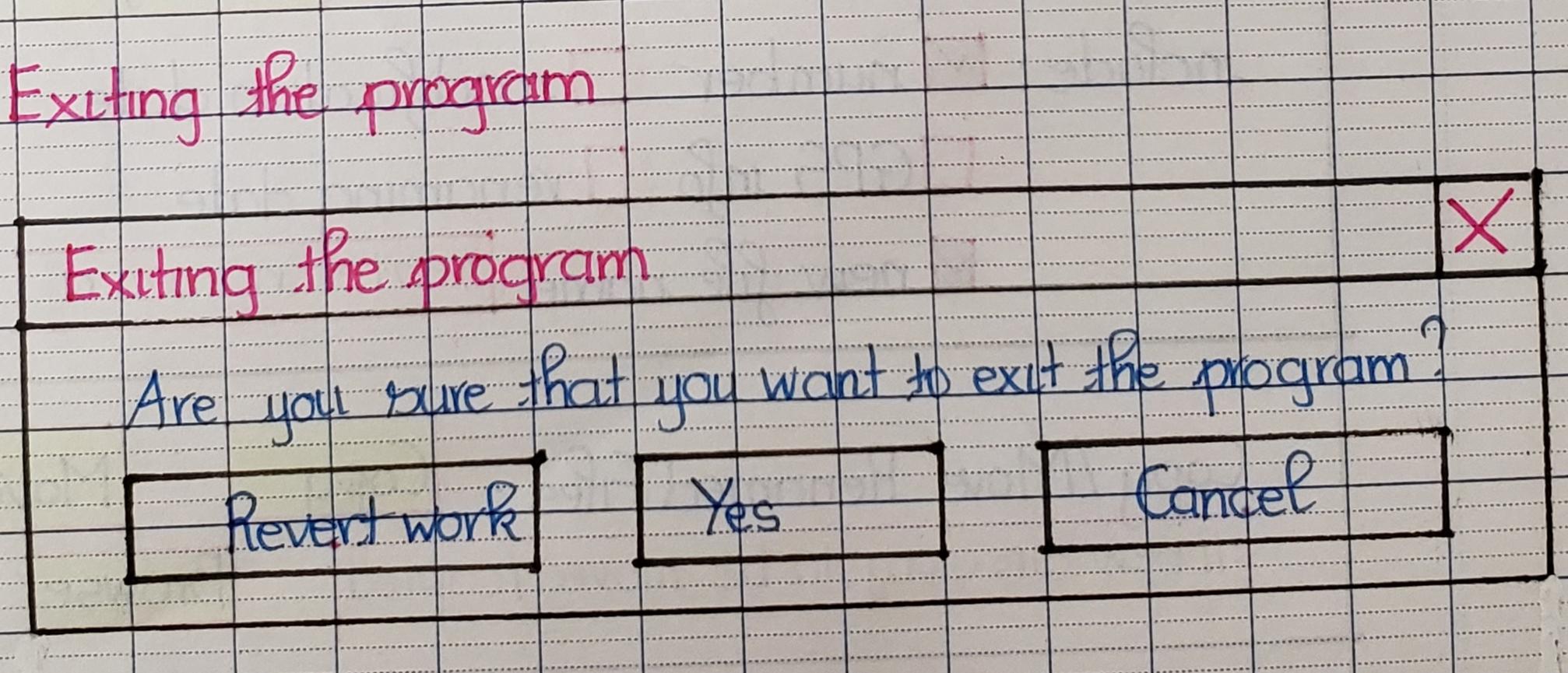
To avoid confusion as we don’t use lots of colours in our program,instead we will be mainly using font size and font style to distinguish between different parts of the program. We will consider using san-serif fonts with the bolding in section titles, such as “**Change Name₅**” in the Rename Option or “**Copy**”**₆** and “**Move**”**₆** in **Copy/Move₆** section. This would capture the user’s attention and ensure the user understands that these are important actions. Section Name such as “**Current File Name**”, “**New Files Name**”, “**Rename Option**” and “**Copy/Move Renamed Files**” will be bold and colored in red, to alert the user of the section’s purpose.



*Image 3: Browsing/Add Files to the program*

Clicking on the Browse button will open up a transient window like the one above. It lets the user select files and folders and accept the common hotkeys available to Windows users. For example, holding shift while selecting an item will select all items in range from the initially selected item to the current one. Holding control while selecting an item will either add or remove it from the existing pool of selected items.

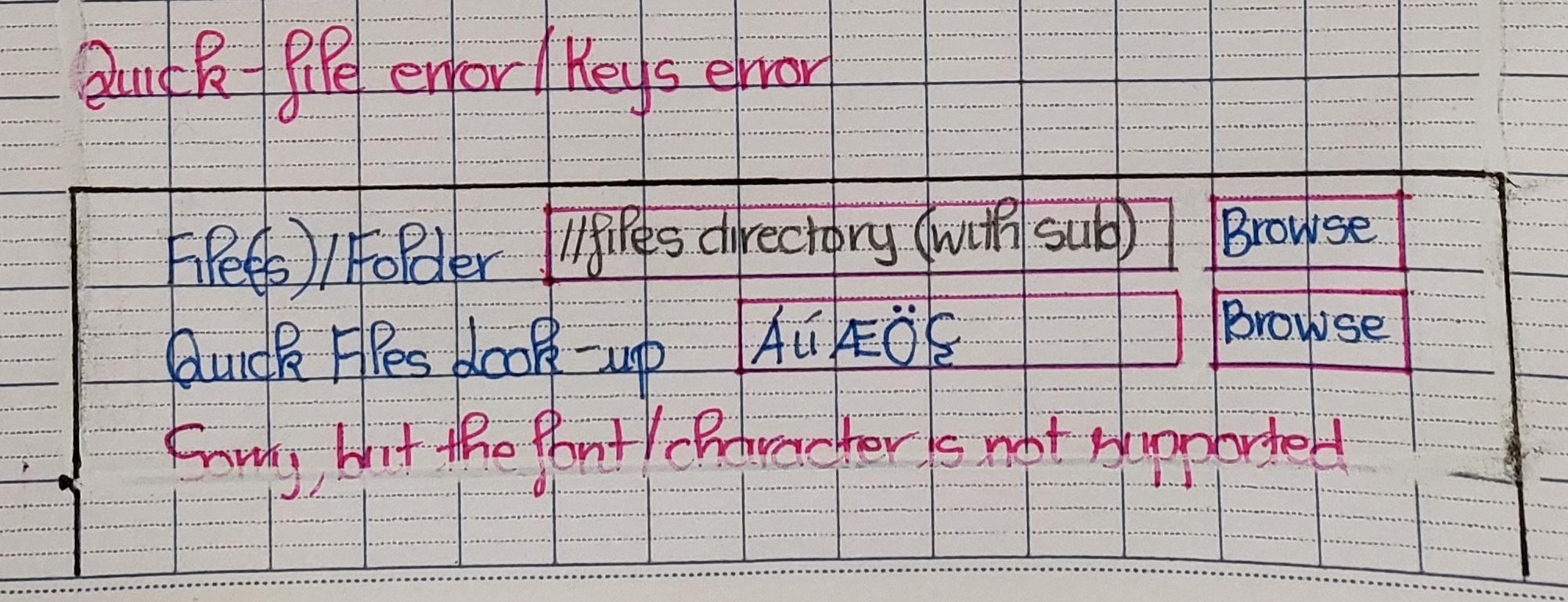
“**Select Files/Folder**” and “**Cancel**” are in bold to alert the user about the purpose of this transient window.



*Image 4: Exiting the program*

This pop-up is shown to the user when they click the X on the program. Yes is to continue the exit operation, Cancel is to remain on the current state of the program and Reverted Work is to Revert any changes applied to the file (like Rename Option)

“**Revert Work**”, “**Yes**” and “**Cancel**” will be shown in San-Serif font and in bold to alert the user about the importance of these options.



*Image 5: Error on the Quick Files Look-up*

This on-page feedback will alert the user of any errors when they try to search for a file with a string that is not valid. This error is meant to be informative and alerts the user of the existing state of the system. The error is color coded in red to express its severity. While the error is up, any additional operations below this section are disabled and greyed out. Users cannot proceed to select individual files for renaming, nor can they perform change names, copy or move functions until the program gets out of the error state.

We decided to not create a separate error window popup for the user because we wanted the feedback to be continuous and in real time. As the user types in their keyword or regular expression in the quick file look up box, the system is continuously validating their input and providing them with the most up to date feedback. “**File(s)/Folder**” and “**Quick Files Look-up**” are in bold to highlight the section.

This layout is a very simple but direct interface that provides lots of information to a user without confusing them. Features and buttons were separated in their own section with different fonts and styling so that a user would be navigated without having to need a deep understanding of the interface. The reason why we choose to use this layout is so that the user follows the exact order that they have in mind when they want to change a file’s name. In order to change a set of files to a new filename, they must first locate the folder or the file(s). The Quick Files Look-up is optional if they have too many files in the folder. Then after they select a bunch of files or a folder, a list of files will be displayed in the app so the user can control and know which files they are dealing with, so they are not going to be messed up later (this would be very hard to keep track if the user changes the name of each individual file). Then instead of messing around of how to keep track of those files properly, we give them a few examples where they can use to manage the order of the files and if they want to use a new name that never exists in one of the files, simply fill in the “New File Name” and select how they want to order (by Number or by Date) and the program will change according to their selection (for number, it will be by order of the file that displayed in the image, and it was originally ordered by alphabetical). After that, the user can simply move the file around to save as a record or to compress them and submit it.

A good thing from our design is that as long as the user doesn’t click on a button such as “Change Name”, “Copy” or “Move”, they have a seamless experience. And if by any chance they did, as a safeguard the program will ask them again at least once (2 for the “Change Name”) as the program wants to be sure that the button wasn’t selected by accident and the user wants to perform the operation.