Test Case ID	Functional Requirement	Test Case Objective	Test Case Description	Expected Result
1	Creating a New File/Opening a File	Ensure that new blank files are created and that text files can be opened (with CEditorObjects if there are sublines).	Create a new file from the menu and tool bar. Open a text file with some basic pseduo code, that includes sublines. 'InsertionSort.txt' from the Design section of this project will be used.	Opens a new file that is either blank if it was created new. If opened 'insertionSort.txt', it should be filled with the text of 'InsertionSort.txt' with the sublines used.
2	Save Files	Ensure that opened files can be saved in a text file, with line numbers indicated on each line.	Create a basic file (with sublines) and save it, making sure that it saves with the appropriate line spacing, line numbers and text.	Saves a .txt file of the document that was saved. It has the line numbers on each line with the appropriate spacing.
3	Print file and export as PDF	Ensure that the app can connect to a printer and print the correctly formatted file. Ensure that the file can be exported as a PDF, in the same format as the printed format.	Print the 'InsertionSort.txt' file to my home printer and two different school printers, to ensure that the printing is consistent even with different printers. While printing each time, export to PDF at the same time and check if the PDF and print match.	Prints the pages on each printer. Each print being the same. The exported PDF is identical to the prints.
4	Edit OS print settings	Ensure that the app can edit the OS print settings when printing, such as print colour, double sided, page size etc…	Print the 'InsertionSort.txt' file to a printer twice, each time with different colour settings, to ensure that the print settings work and are consistent.	Prints the pages, using the correct colour settings. Each time printing with the same settings on different printers should have identical prints.
5	Exitting the application	Ensure that the app can close without any errors like memory leaks or assertion failures.	Close the application several times and test if it works. Test close from menu function, top right 'X' button on window and via the task bar.	Application closes, is removed from the task bar and from the OS task manager.
6	Copy text to the clipboard	Ensure that the user will be able to copy highlighted text to the clipboard from the menu, the toolbar and commands (CTRL + C).	Using 'InsertionSort.txt', copy one of the editors and attempt to paste it into a text editor.	Highlighted text will be copied and will be pasted into Microsoft word, exactly as it was highlighted. There should be correct line spacing as well. It should also be visible on the OS clipboard.

7	Paste text from the clipboard into the application	Ensure that the user can paste next from the clipboard into the current active editor, including a carridge character to add new lines.	Create a small word file in a text editor (including some keywords) and copy it, attempt to paste it into the application.	Application adds the text into the current editor, from the active line in the current caret position. It creates a new line when it recieves a carridge character (\n).
8	Opening customise window dialog box	Ensure that the customise dialog box can open without any faults or errors.	Open the window in several different running instances of the application and check for errors. No need to test the actual functionality of the dialog box as this is being handled by the OS and the MFC framework.	Customise dialog box opens without any faults or errors.
9	Enabling/disabling the tool bar	Ensure that the tool bar can be enabled and disabled.	Use this button several times in several different instances of the application and check for failures.	Tool bar disappears and the tabs are moved up at the top, the space inside the view is made bigger.
10	Enabling/disabling the status bar	Ensure that the status bar can be enabled and disabled.	Use this button several times in several different instances of the application and check for failures.	Status bar disappears and the tabs are moved down at the bottom, the space inside the view is made bigger.
11	Changing the windows theme	Ensure the user can change the windows theme to a selection of in-built themes.	Use this function several times in several different instances of the application, with different themes, and check for failures.	The window theme is changed appropriatley based on the selected theme. Colours may change, and some graphics may change as well.
12	Close any tabs	Ensure that any tabs can be closed from the main window, or from a dialog box that specifically manages active tabs.	Use this button and function several times in several different instances of the application and check for failures.	Tab closes and current tab changes to the most recently used one, or, if there are no other tabs, all views disappear and the background appears to be a grey screen.
13	Open a new window from the app	Ensure that new instances of the app can be made from the menu bar.	Use this function several times in several different instances of the application and check for failures.	A new window is created, it is shown on the task bar
14	Use the SmartColour dialog box	Ensure that the SmartColour dialog can be opened. Ensure that the user can add and remove keywords to the selection, change colours, change fonts and new indentation sizes. Ensure all changes happen as soon as the dialog is closed.	Add/remove several words to/from each list box and check that they are being implemented by the application. Change several colours and check that they are also being implemented. Test each font to make sure that all are bale to be used. Make sure all changed take place as soon as the dialog box closes	SmartColour dialog can be opened without any faults or errors. Keywords can be added and removed. Colours can be changed. Fonts and indentation size can be changed. When the dialog is closed, the changes are automatically updated.

15	Open the website from the app	Ensure that the website can be opened from the menu bar in the default browser.	Use this function several times in several different instances of the application and check for failures. Also test it with different default browsers (Microsoft Edge and Google Chrome).	Website opens in the default browser.
16	Open/close the about dialog box	Ensure that the about dialog box opens and closes without error.	Open/close the about dialog several times in several different instances of the application and check for failures.	About dialog box is shown/closed.
17	Add/remove text from lines	Ensure that the user can add/remove text to/from the editor.	Write a basic file and make sure there are no problems when adding or removing text.	Text is deleted and removed from the line. Text is added and rendered on to the line. Text is editted at caret position.
18	Add/Remove lines and subline blocks	Ensure that the user can add/remove line and sublines to/from the editor. Ensure that when lines are added/removed, that sublines are incremented apporpriatley.	Write a basic file and make sure there are no problems when adding or removing lines and sublines. Check that when adding sublines, if it already exists, the application selects the one that already exists. Check that when removing lines (via highlight or backspace) that if its subline exists, don't remove it.	Lines and or a subline is/are deleted and removed from editor/file. A line or subline is added and rendered on to the line. A new line is created in the line underneath the old line, it is empty or contains the text that was to the right of the caret. This line is selected. A new subline is created in the correct place (ie, is in the right order of lines). This new line is selected for editting.

19	Run GUI objects and interface	Ensure that all GUI objects render efficiently and apropriatley.	Check that each GUI object is being rendered, do this by running the application and loking at each object. (Subline button arrow, highlighter, selected line rectangle, sidebar, text and line numbers, caret, cursor).	Subline button arrow is rendered when hovering and selecting it. The highlighter is in the appropriate position as it should be expected to be. It is also transparent and the colour that is selected in the SmartColour. The sidebar rectangle reaches from the top to bottom of each editor, each editor has its own sidebar. Text and numbers coloured depending on the SmartColour colours. Line numbers render as the one colour. Caret renders where the editor, line and caretPos is selected. Cursor image depends where in the screen the cursor has moved to. It should change depending on where it is.