Change request log

1. Concept Location

Step #	Description	Rationale
1	We built and ran the jEdit system	
2	We explored the various aspects of the system	To get familiar with the software, we checked out all the menu items available and other features of the system
3	We looked for StatusBar in the project.	Upon exploring the system, we found the area that is responsible for displaying the line count, caret position, etc. is called as Status bar (from the view tab in the menu bar) and decided to look for this item in the source code.
4	From the search results, we understood StatusBar was the class we needed to update to incorporate CR01	As this class was solely responsible for displaying status bar elements on GUI.
5	We inspected the StatusBar class.	To check for methods that were used to set the StatusBar values.
6	In the updateCaretStatus method, we updated the caret Position to display a random string instead of the actual position	To confirm if this is the method of our interest, we updated the value and re-ran the system
7	We marked the class – StatusBar and method – updateCaretStatus as located	After running the system again, we confirmed that the data displayed in the status bar was indeed populated from this method and this had to be modified.

Time spent (in minutes): 35.

Classes and methods inspected:

- org/gjt/sp/jedit/gui/StatusBar
    - updateCaretStatus
- org/gjt/sp/jedit/textarea/ScreenLineManager
- org/gjt/sp/jedit/textarea/JEditTextArea
- org/gjt/sp/jedit/gui/statusbar/BufferSetWidgetFactory

2. Impact Analysis

Step #	Description	Rationale
1	We traced the JEditTextArea class.	The current caret position was fetched from this class and hence we decided to inspect the same.
2	We searched for “wordcount” string in the project	While exploring the software, we found a feature that shows the number of words present in the text area. The method of calculating the number of words would be useful for our change request.
3	We found the doWordCount method in JEditTextArea class and updated it to statically display a random number of words instead of calculating the same	To find out if this method calculated what we wanted, we decided to perform this change. This method was marked as it calculated the number of words and we need not write a new method to do the same.

4	We appended "(/)" to the status bar buffer	To check if we could add new information next to the line and character count information in the status bar. We marked this for change as well.
5	We checked out the Selection and SelectionManager classes.	On the GUI, the word count feature allowed to calculate the number of words for some selected text area instead of the entire text area. We decided to check out these classes to see if they could in some way give us the information on the current word the caret is pointing to, to know the position of the word from the beginning of the text area. But since these classes did not, we did not select these classes for change.

Time spent (in minutes): 30.

Classes and methods inspected:

- org/gjt/sp/jedit/gui/StatusBar
  - updateCaretStatus
- org/gjt/sp/jedit/textarea/JEditTextArea
  - doWordCount
- org/gjt/sp/jedit/textarea/Selection
- org/gjt/sp/jedit/textarea/SelectionManager

3. Actualization

Step #	Description	Rationale
1	We created a new method to get the Word count in JEditTextArea class.	We need to calculate the total number of words in the text editor and hence, it was a good idea to have a separate method to do the same instead of writing the code repeatedly. Class – JEditTextArea (line – 383)
2	To reduce redundancy, we updated the doWordCount method to get the word count from our new method instead of calculating the same once again.	
3	Updated the updateCaretStatus method in StatusBar class.	We used 2 variables to store the total word count and current word position (count) details. We then appended this information to the status bar. Class – StatusBar Method – updateCaretStatus (Line – 359,362 426-430),
4	We built the project and checked if everything worked as intended	To test if our changes are reflected in the system.

Time spent (in minutes): 40.

Classes and methods inspected:

- org/gjt/sp/jedit/gui/StatusBar
  - updateCaretStatus
- org/gjt/sp/jedit/textarea/JEditTextArea
  - doWordCount

- getWordCount

4. Validation

Step #	Description	Rationale
1	Test case 01: Test if the system loads correctly and if the user can write, save, and open text. Inputs: None Expected output: The system should work correctly with all features working without any errors	This is the expected behavior. Test case passed.
2	Test case 02: Test if the correct total word count is displayed in the status bar. Inputs: Random string of words. Expected output: The status bar should display the correct total of the number of words.	The correct total number of words in the text editor should be displayed in the status bar. Test case passed.
3	Test case 03: Test if the correct word position is displayed in the status bar when the caret is in the middle of a word. Inputs: Random string of words. Expected output: The status bar should display the position of the word out of the total words in the text editor.	When the caret is in the middle of a word, the status bar should display the position of the current word out of the total words in the editor in the status bar. Test case passed.
4	Test case 04: Test if the correct word position is displayed in the status bar when the caret is at the beginning of a word. Inputs: Random string of words. Expected output: The status bar should display the position of the previous word out of the total words in the text editor.	When the caret is at the beginning of a word, the status bar should display the position of the previous word out of the total words in the editor in the status bar. Test case passed.
5	Test case 05: Test if the correct word position is displayed in the status bar when the caret is at the beginning of the text editor. Inputs: Random string of words. Expected output: The status bar should display 0/(total words)	When the caret is at the beginning of the text editor, the status bar should display the position of the previous word out of the total words in the editor in the status bar. Since there is no previous position, it should display 0/(total words) Test case passed.

Time spent (in minutes): 20.

5. Summary of the change request

Phase	Time (minutes)	No. of classes inspected	No. of classes changed	No. of methods inspected	No. of methods changes
Concept location	35	4	0	1	1
Impact Analysis	30	4	0	2	2
Actualization	40	2	0	3	3
Verification	20	-	-	-	-
Total	125	10	0	6	6

6. Conclusions

Since we thoroughly inspected the system, concept location and finding relevant methods and classes using keywords was an easy task. We took some time to brainstorm how to calculate the position of the current word since we thought of multiple approaches, but they needed modification of a lot of other classes and meth We then found out a simpler way to calculate this and successfully implemented the same. Since we just had to test the GUI, we decided to perform this activity manually as there was a single element to be checked and verified.