Change request log

1. Concept Location

Step #	Description	Rationale
1	Downloaded JEdit project folder and jsr305-3.0.2 jar	Setup the project in local system
2	Created a Github repo with name JEditTeam17, and pushed the JEdit project folder to the remote.	To have a central version control system for easy collaboration and contributions
3	Follow the instructions provided and start the JEdit application	Navigate through the application and explore the features
4	We inspected the project directory structure and examined the flow of how the application is working	Understood the project setup and code base
5	We navigated through org.gjt.sp.jedit folder to check for different modules of the application	We were confident enough that this folder would contain the code that manipulates the UI options
6	We inspected the class SelectionLengthWidgetFactory situated under org.gjt.sp.jedit.gui.statusbar folder	Observed that the class has a factory design pattern implementation and understood the class usage
7	We started to inspect the location where the above class SelectionLengthWidgetFactory is being called.	We were unable to see any usages of the factory class anywhere in the code base
8	We inspected ActionBar class	We confirmed this class had nothing to do with the status bar.
9	We marked the StatusBar class as located	We confirmed this class contains the methods that manipulates the status bar

Time spent (in minutes): 30

Classes and methods inspected:

- `org/gjt/sp/jedit/gui/statusbar/SelectionLengthWidgetFactory.java`
  - `void update()`
- `org/gjt/sp/jedit/gui/ActionBar.java`
  - `void invoke()`
- `org/gjt/sp/jedit/gui/StatusBar.java`
  - `Widget getWidget(String name)`
  - `Void updateCaretStatus()`
  - `Void updateBufferStatus()`
- `org/gjt/sp/jedit/jEdit.java`
  - `void getBooleanProperty(String name)`

2. Impact Analysis

Step #	Description	Rationale
1	We inspected the getBooleanProperty method from the jEdit.java class	To track the usage of Boolean property elsewhere
2	We inspected the class StatusWidgetFactory classes	We examined if the updateCaretStatus() from StatusBar.java is being used anywhere in the factory classes

3	We inspected the class JEditTextArea.java class	We checked if the newly added code affects the text area, and realized that it does not require any changes.

Time spent (in minutes): 45

Classes Inspected:

- org/gjt/sp/jedit/gui/statusbar/StatusWidgetFactory.java
  - void update() of it's implementations
- org/gjt/sp/jedit/textarea/JEditTextArea.java
  - void home()
  - void smartHome()
  - void userInput(char ch)
- org/gjt/sp/jedit/jEdit.java
  - void getBooleanProperty(String name)

3. Prefactoring (optional)

Step #	Description	Rationale
1	We committed our changes with git.	Just in case we need to revert our changes.

Time spent (in minutes): 5

4. Actualization

Step #	Description	Rationale
1	We made changes to the StatusBar.java class	We realized that updateCaretStatus() handles the status bar content, so modified it show the word offset count.

Time spent (in minutes): 25

Classes inspected & modified:

- org/gjt/sp/jedit/gui/StatusBar.java
  - Void updateCaretStatus()

Classes inspected:

- org/gjt/sp/jedit/gui/statusbar/StatusWidgetFactory.java
  - void update() of it's implementations
- org/gjt/sp/jedit/textarea/JEditTextArea.java
  - void home()
  - void smartHome()
  - void userInput(char ch)
- org/gjt/sp/jedit/jEdit.java
  - void getBooleanProperty(String name)
- org/gjt/sp/jedit/gui/ActionBar.java
  - void invoke()

5. Postfactoring (optional)

Step #	Description	Rationale
1	We committed and pushed our changes with git.	Just in case we need to revert our changes.

Time spent (in minutes): 5

6. Validation

Step #	Description	Rationale
1	We performed functional testing	We performed functional testing
2	To make sure everything works as expected functionally	To make sure everything works as expected functionally
3	We tested use case of the requirements	To make sure word offsets are as expected

Time spent (in minutes): 20

7. 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	30	4	1	6	1
Impact Analysis	45	5	1	6	1
Prefactoring	5				
Actualization	25	2	1	4	1
Postfactoring	5				
Verification	20				
Total	120	5	1	6	1

8. Conclusions

Finding concepts and comprehending the code base required a lot of work because there are many classes and methods. However, finding the classes was made easy because folder structure naming was on point and easy to understand. The primary modification that we found challenging was the impact study, which took a long time to complete due to the size of the codebase.