**Software Requirements Specification**

**SimpleMerge Project**

**Team 19**

**Written by Yang Insu**

**Version 1.1**

**SimpleMerge Software Requirements Specification**

**Version 1.1**

**May 17, 2018**

**Project Team:**

**Team 19:**

Kim Hyeontae

Son Chang Woo

Kang Min Soo

Lee Jun ho

Park Jin Hyuk

Yang Insu

**Document Author(s):**

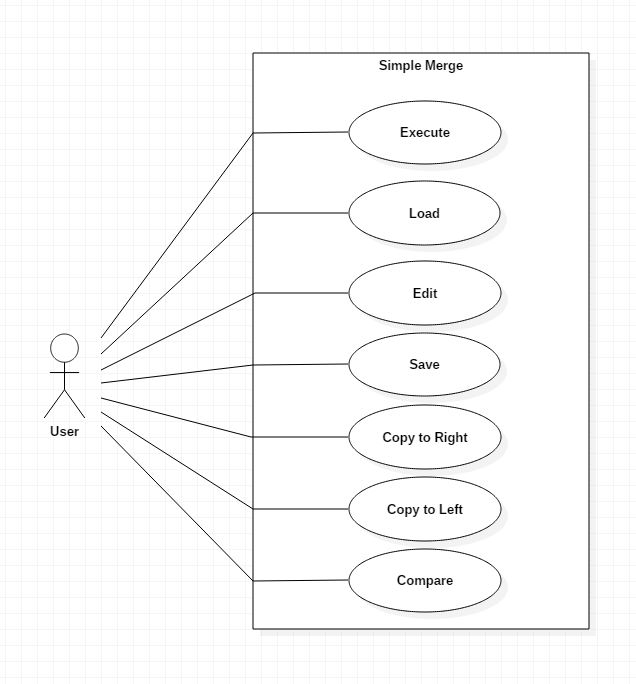
Yang Insu

**Table of Contents**

1. **Introduction**
2. **Use Cases**
3. UC1 Flow of Events for the Start Use Case
4. UC2 Flow of Events for the Load Left Use Case
5. UC3 Flow of Events for the Load Right Use Case
6. UC4 Flow of Events for the Edit Use Case
7. UC5 Flow of Events for the Compare Use Case
8. UC6 Flow of Events for the Merge(1) Use Case
9. UC7 Flow of Events for the Merge(2) Use Case
10. UC8 Flow of Events for the Save Left Use Case
11. UC9 Flow of Events for the Save Right Use Case
12. **Misuse Cases**
13. UC10 Flow of Events for the Editing without using Edit Use Case
14. UC11 Flow of Events for the Editing Text File while using SimpleMerge Use Case
15. **System Sequential Diagrams**
16. Load Sequential Diagram
17. Edit Sequential Diagram
18. Compare Merge Sequential Diagram
19. Save Sequential Diagram
20. **Nonfunctional Requirements**
21. **Constraints**
22. **Requirements Dependency Traceability Table**
23. **Development and Target Platforms**
24. **Project Glossary**
25. **Document Revision History**

**I. Introduction**

The objective of this project is to make a ***SimpleMerge*** program written in Java language. The ***SimpleMerge*** displays two text files to the user (on the left, and right) and allows him/her to utilize several functionalities: Load, Edit, Save Merge, and Compare. The following document is written to describe the requirements of the to-be implemented program.



Use case diagram for the SimpleMerge Program

**II. Use Cases**

UC1 Start

UC2 Load Left

UC3 Load Right

UC4 Edit

UC5 Compare

UC6 Merge (1) – Copy to Left

UC7 Merge (2) – Copy to Right

UC8 Save Left

UC9Save Right

**UC1 Flow of Events for *Start* Use Case**

* 1. **Preconditions:**

None.

* 1. **Main Flow:**

User boots the ***SimpleMerge*** Program.

* 1. **Subflows:**

None.

* 1. **Alternative Flows:**

None.

**UC2 Flow of Events for the *Load Left* Use Case**

* 1. **Preconditions**

1. UC1 Start must be preceded.
2. User has text files to load in the set directory.

**2.2 Main Flow**

User chooses text file that he/she wants to display on the left panel, and ***SimpleMerge*** loads the corresponding file.

**2.3 Subflows**

1. User presses the *Load Left* button at the top of the program window.
2. A new directory window pops up.
3. User chooses the wanted file for the left panel, then presses the *Open* button.
4. Left panel should display the corresponding files the user has chosen.

**2.4 Alternative Flows**

1. If user loads invalid file format, error message pops up.
2. If user loads a blank file (no text content), program accepts as a blank file.

**UC3 Flow of Events for the *Load Right* Use Case**

**3.1 Preconditions**

1. UC1 Start must be preceded.
2. User has text files to load in the set directory.

**3.2 Main Flow**

User chooses text file that he/she wants to display on the right panel, and ***SimpleMerge*** loads the corresponding file.

**3.3 Subflows**

1. User presses the *Load Right* button at the top of the program window.
2. A new directory window pops up.
3. User chooses the wanted file for the right panel, then presses the *Open* button.
4. Right panel should display the corresponding files the user has chosen.

**3.4 Alternative Flows**

1. If user loads invalid file format, error message pops up.
2. If user loads a blank file (no text content), program accepts as a blank file.

**UC4 Flow of Events for the *Edit* Use Case**

**4.1 Preconditions:**

1. UC1 Start must be preceded
2. UC2 Load Left or UC3 Load Right must be preceded

**4.2 Main Flow**

1. User clicks Edit button on the top of the program window.
2. The program unlocks the panels and allows user to edit the text files.
3. User makes changes to the files.

**4.3 Subflows**

None.

**4.4 Alternative Flow**

1. Nothing happens if user tries to edit text without clicking the *Edit* button.
2. Edit function is disabled when clicked again.

**UC5 Flow of Events for the *Compare* Use Case**

**5.1 Preconditions**

1. UC1 *Start* must be preceded
2. UC2 *Load Left* and UC3 *Load Right* must be preceded.

**5.2 Main Flow**

1. User clicks Compare button
2. Program compares the two text files and highlights the different lines.
   1. **Subflows**
3. If there is a blank line in the left panel, and no blank line in the right, compare results in producing a blank line one the right panel. (only for visualization, no changes occur in the text file)
   1. **Alternative Flows**
4. If user clicks Compare button without preempting UC2 and UC3, no changes occur.
5. If both panels have the same contents, pop-up window indicating that the contents are the same tells the user such information.

**UC6 Flow of Events for the *Merge(1): Copy to Right* Use Case**

* 1. **Preconditions**

1. UC1 *Start* must be preceded.
2. UC2 *Load Left* and UC3 *Load Right* must be preceded.
3. UC5 Compare must be preceded.
4. There must be highlighted lines (different contents) due to the result of UC5.
   1. **Main Flow**

1. User clicks on the line that he/she wants to copy. (put the cursor)

2. User clicks the *“Copy to Right***”** button on the top of the program window.

3. Selected line is copied to the right panel.

* 1. **Subflows**

1. After selected line has been dealt with *“Copy to Right”*, the cursor is moved to the next line that is different by clicking *“Copy to Right”*again.

* 1. **Alternative Flows**

1. If contents of the left panel is blank,*“Copy to Right”* results in right panel’s contents erased to blank.
2. If there are no different lines left, a message is popped to the user indicating “The files have same contents”.

**UC7 Flow of Events for the *Merge(2): Copy to Left* Use Case**

**7.1 Preconditions**

1. UC1 *Start* must be preceded.

2. UC2 *Load Left* UC3 *Load Right* must be preceded.

1. UC5 *Compare* must be preceded.
2. There must be highlighted lines (different contents) due to the result of UC5.

**7.2 Main Flow**

1. User clicks on the line that he/she wants to copy. (put the cursor)

2. User clicks the *“Copy to Left”*button on the top of the program window.

3. Selected line is copied to the left panel.

**7.3 Subflows**

1. After selected line has been dealt with *“Copy to Left”*, the cursor is moved to the next line that is different by clicking *“Copy to Left”*again.

**7.4 Alternative Flows**

1. If contents of the left panel is blank*, “Copy to Left”* results in right panel’s contents erased to blank.
2. If there are no different lines left, a message is popped to the user indicating “The files have same contents”.

**UC8 Flow of Events for the *Save Left* Use Case**

**8.1 Preconditions**

1. UC1 *Start* must be preceded.
2. UC2 *Load Left* must be preceded.

**8.2 Main Flow**

1. Use clicks the Save Left button, and the contents of the files are saved.
2. User clicks the Save As Left button, and saves the contents in the wanted directory.

**8.3 Subflows**

1. User clicks the Save As Left button, a directory window pops up, asking the user where to save the file. After the user indicates the directory of the new to-be saved file, clicks the Save button (on the pop up window), program saves the following file.
2. User clicks the Save Left button, and the contents of the panels are saved in the previous file.

**8.4 Alternative Flows**

1. The program saves the files contents regardless of the previous activities. (even if the user did not make any changes to the original file)
2. If user indicates an invalid saving space, program shows a pop up window telling the indicating error, and does not save.
3. If user requests Save As Left , with a file name same as the original file in the same directory, program overwrites the previous file. (same as Save Left)
4. If user did not preempt UC2 Load Left, and requests for Save/Save as Left, error message is popped indicating that there are no contents to save.

**UC9 Flow of Events for the *Save Right* Use Case**

**9.1 Preconditions**

1. UC1 *Start* must be preceded.

2. UC2 *Load Right* must be preceded.

**9.2 Main Flow**

1. User clicks the Save Right button, and the contents of the files are saved.

2. User clicks the Save As Right button, and saves the contents in the wanted directory.

**9.3 Subflows**

1. User clicks the Save As Right button, a directory window pops up, asking the user where to save the file. After the user indicates the directory of the new to-be saved file, clicks the Save button (on the pop up window), program saves the following file.

2. User clicks the Save Right button, and the contents of the panels are saved in the previous file.

**9.4 Alternative Flows**

1. The program saves the files contents regardless of the previous activities. (even if the user did not make any changes to the original file)

2. If user indicates an invalid saving space, program shows a pop up window telling the indicating error, and does not save.

1. If user requests Save As Right, with a file name same as the original file in the same directory, program overwrites the previous file. (same as Save Left)
2. If user did not preempt UC2 Load Left, and requests for Save/Save as Right, error message is popped indicating that there are no contents to save.

**III. Misuse Cases**

**UC10 Flow of Events for the *Editing without using EDIT* Use Case**

* 1. **Preconditions**

1. UC1 *Start* is preceded.

2. UC2 *Load Left* or UC3 *Load Right* is preceded.

* 1. **Main Flow**

User tries to edit text file without clicking the Edit Button.

* 1. **Sub-flows:**

None.

* 1. **Alternative Flows:**

Nothing happens. The program does not accept any keyboard inputs since the Edit button acts as a toggle key for the editing feature.

**UC11 Flow of Events for *Editing Text File while Using SimpleMerge* Use Case**

* 1. **Preconditions**

1. UC1 *Start* is preceded,

2. UC2 *Load Left* or UC3 *Load Right* is preceded

3. User separately opens the text file by other text program (i.e. Notepad for Windows)

**9.2 Main Flow**

User edits the text file using other text programs, therefore the text loaded by the ***SimpleMerge*** is no longer valid.

**9.3 Subflows:**

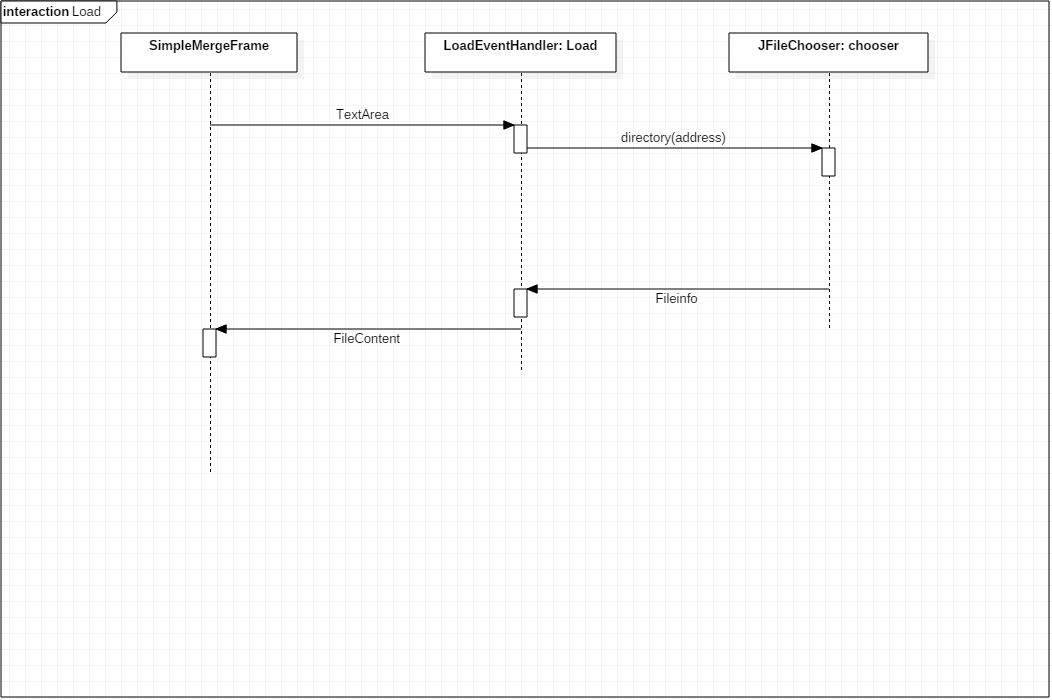
None.

**9.4 Alternative Flows:**

***SimpleMerge*** ignores any changes to the original text file.

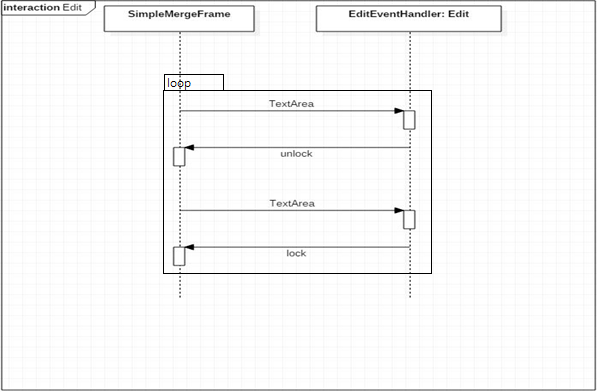
**IV. System Sequential Diagrams**

1. Load SD



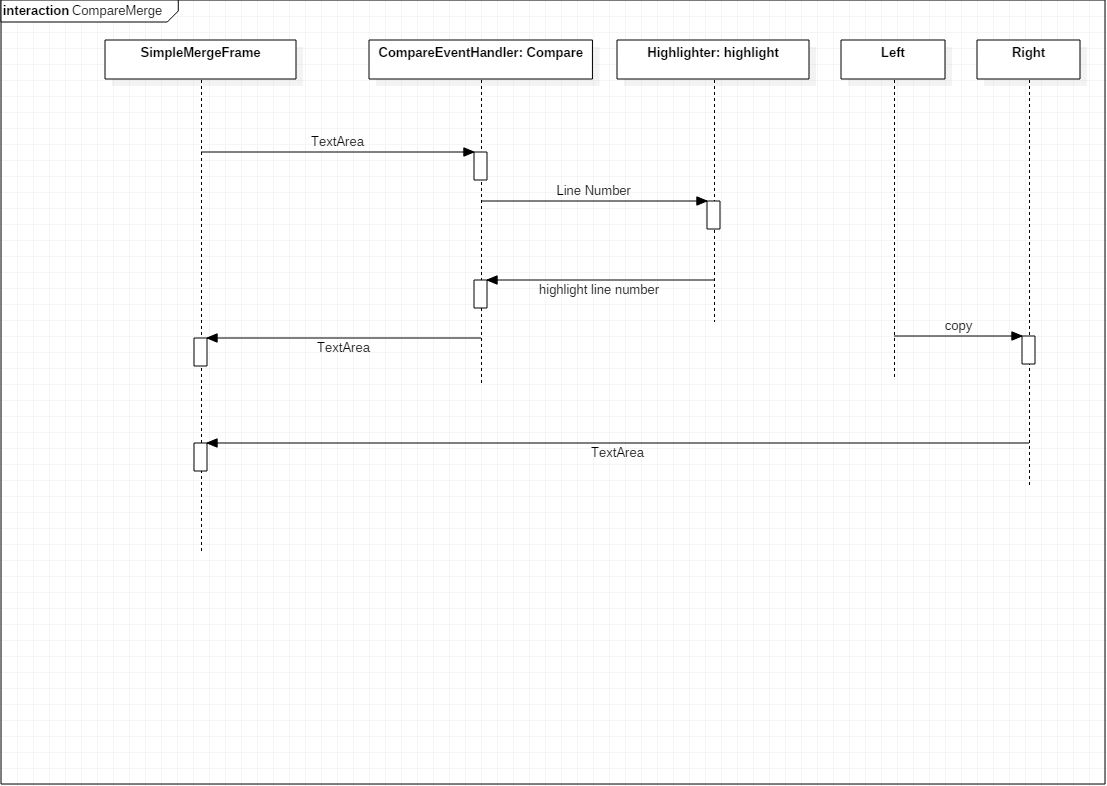
Description: SimpleMergeFrame is a instance created by SimpleMergeFrame Class. In the SimpleMergeFrame class, there are functions that create the SimpleMerge program window. Among the elements, there is TextArea instance, which indicate the panel’s area where the text file is to be loaded. (logic for the use cases Load Left, Load Right are the same)

2. Edit SD



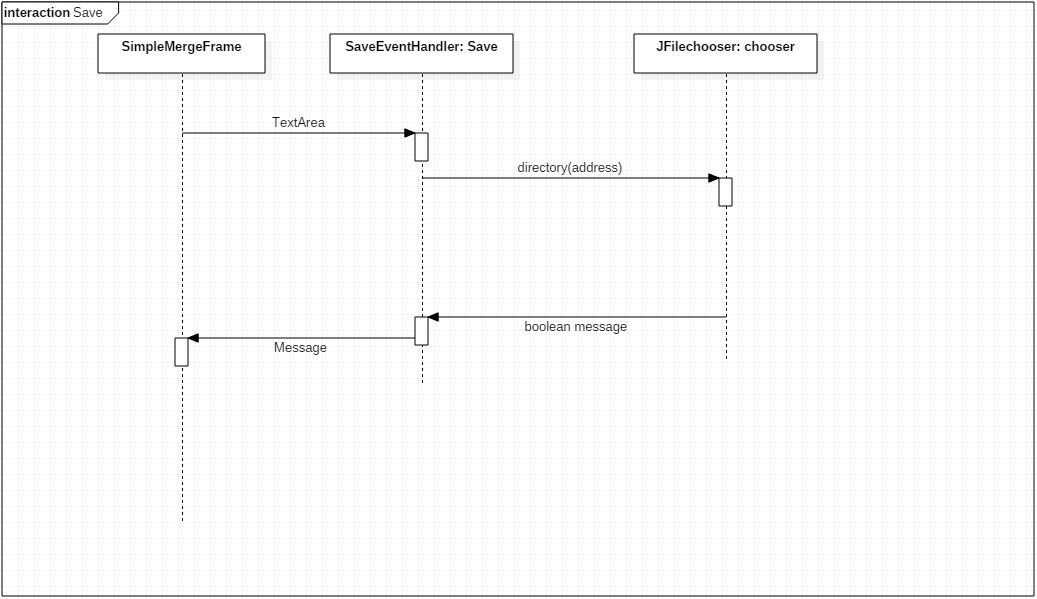
Description: The default settings of the TextAreas are locked. Therefore, in order to edit the text files, user must click Edit firsthand. If the Edit button is triggered, then the Event handler for edit unlocks the TextArea. To lock the Text Area again, it is logical that unlock has been preceded.

3. Compare Merge SD



Description:

The following diagram indicates the Compare and Copy to Right Use Case. For the Copy to Left, all other contents are same except for the direction of the copy attribute.



Description:

The scope of the program disallows the user to see if the file has been saved correctly or not, therefore Save event handler returns message to tell the user if the Save command has been executed successfully or not.

(logic for the use Cases for Save Left/Right, Save as Left/Right are same in this interaction)

**V. Nonfunctional Requirements**

**NR1. Performance**

***SimpleMerge*** does not execute any functions until user makes certain requests. All functions shall be completed according to the requested function, and quickly.

**NR2. Usability**

1. Graphical User Interface: The SimpleMerge program provides the user with a GUI to conduct all tasks of the program. The GUI must be intuitive, and easy to use.
2. Language: All of the contents of the program is written and described in English.
3. Access: SimpleMerge program is accessed by the JVM.

**VI. Constraints**

All code implementations shall be done with the Java language. All project progress is saved and shared via GitHub.

**VII. Requirements Dependency Traceability Table**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | UC1 | UC2 | UC3 | UC4 | UC5 | UC6 | UC7 | UC8 | UC9 | UC10 | UC11 |
| UC1 |  |  |  |  |  |  |  |  |  |  |  |
| UC2 | X |  |  |  |  |  |  |  |  |  |  |
| UC3 | X |  |  |  |  |  |  |  |  |  |  |
| UC4 | X | X | X |  |  |  |  |  |  |  |  |
| UC5 | X | X | X |  |  |  |  |  |  |  |  |
| UC6 | X | X | X |  | X |  |  |  |  |  |  |
| UC7 | X | X | X |  | X |  |  |  |  |  |  |
| UC8 | X | X |  |  |  |  |  |  |  |  |  |
| UC9 | X |  | X |  |  |  |  |  |  |  |  |
| UC10 | X | X | X |  |  |  |  |  |  |  |  |
| UC11 | X | X | X |  |  |  |  |  |  |  |  |

**VIII. Development and Target Platforms**

1. Any operating systems with JVM

2. Eclipse IDE

3. JUnit for Testing Tools.

**IX. Project Glossary**

**panel**: a field in the program window where text files are to be loaded. There are two panels in the program, left and right.

**directory**: a window specifying the placements of files in the local disk, allows user to choose which file to open in the program. Directory window is popped up during UC2 Load, and UC7 Save.

**X. Document Revision History**

|  |  |  |
| --- | --- | --- |
| Version | 1.0 | 1.1 |
| Name(s) | Yang Insu | Yang Insu |
| Date | May 12, 2018 | May 17, 2018 |
| Change Description | SRS created. | System Sequence Diagram, Use cases added. |