**SimpleMerge Software Requirements Specification**

**Version 1.0**

**May 12, 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

**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.

**II. Use Cases**

UC1 Start

UC2 Load

UC3 Edit

UC4 Compare

UC5 Merge (1) – Copy to Left

UC6 Merge (2) – Copy to Right

UC7 Save

**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* Use Case**

* 1. **Preconditions**

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

**2.2 Main Flow**

1. User presses the Load 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. Then a new directory window pops up for the right panel’s file.
5. User chooses the wanted file for the right panel, then presses the Open button.
6. Panels should display the corresponding files the user has chosen.

**2.3 Subflows**

**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 *Edit* Use Case**

* 1. **Preconditions:**

1. UC1 Start must be preceded
2. UC2 Load must be preceded
   1. **Main Flow**
3. User clicks Edit button on the top of the program window.
4. The program unlocks the panels and allows user to edit the text files.
5. User makes changes to the files.
   1. **Subflows**

None.

* 1. **Alternative Flow**

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

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

* 1. **Preconditions**

1. UC1 *Start* must be preceded
2. UC2 *Load* must be preceded.
   1. **Main Flow**
3. User clicks Compare button
4. Program compares the two text files and highlights the different lines.
   1. **Subflows**
5. The standard panel for the comparison is the left file.
6. 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)
7. If there is no blank on the left, but right panel has a blank line, then the blank line is erased. (only for visualization, no changes occur in the text file)
   1. **Alternative Flows**
8. If user clicks Compare button without preempting UC2, no changes occur.
9. If both panels have the same contents, pop-up window indicating that the contents are the same tells the user such information.

**UC5 Flow of Events for the *Merge(1): Copy to Right* Use Case**

* 1. **Preconditions**

1. UC1 *Start* must be preceded.
2. UC2 Load must be preceded.
3. UC4 Compare must be preceded.
4. There must be highlighted lines (different contents) due to the result of UC4.
   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”.

**UC6 Flow of Events for the *Merge(2): Copy to Left* Use Case**

**6.1 Preconditions**

1. UC1 *Start* must be preceded.

2. UC2 *Load* must be preceded.

1. UC4 *Compare* must be preceded.
2. There must be highlighted lines (different contents) due to the result of UC4.

**6.2 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.

**6.3 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.

**6.4 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”.

**UC7 Flow of Events for the *Save* Use Case**

* 1. **Preconditions**

1. UC1 *Start* must be preceded.
2. UC2 *Load* must be preceded.
   1. **Main Flow**
3. Use clicks the Save button, and the contents of the files are saved.
4. User clicks the Save As button, and saves the contents in the wanted directory.
   1. **Subflows**
5. User clicks the Save As 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.
   1. **Alternative Flows**
6. The program saves the files contents regardless of the previous activities. (even if the user did not make any changes to the original file)
7. If user indicates an invalid saving space, program shows a pop up window telling the indicating error, and does not save.
8. If user requests Save As, with a file name same as the original file in the same directory, program overwrites the previous file. (same as Save)

**III. Misuse Cases**

**UC8 Flow of Events for the *Editing without using EDIT* Use Case**

* 1. **Preconditions**

1. UC1 *Start* is preceded.

2. UC2 *Load* 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.

**UC9 Flow of Events for *Editing Text File while Using SimpleMerge* Use Case**

* 1. **Preconditions**

1. UC1 *Start* is preceded,

2. UC2 *Load* 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. 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**

**V. Constraints**

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

**VI. Requirements Dependency Traceability Table**

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

**VII. Development and Target Platforms**

1. Windows Operating System

2. Eclipse IDE

**VIII. 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.

**IX. Document Revision History**

|  |  |
| --- | --- |
| Version | 1.0 |
| Name(s) | Yang Insu |
| Date | May 12, 2018 |
| Change Description | Original creation of the SRS. |