12/3/2014

ASSBUS

Shamiya Gissentanna

Shaina Daya

Bridgette Reisinger

Shuaib Mushfiq

Umesh Velagapudi

Angelica charles

Project Name:

AB Editor

Software Engineering

CSc 4350 – Fall 2014

Table of Contents

**Topic2**

**Requirements Analysis Document3-24**

Introduction3

*Overview3*

*Scope of the System3*

*Objectives and Success Criteria of the Project3*

Current System4

*Functionality4*

Future System4

*Problem Statement4*

*Requirements Traceability Matrix5*

System Model7

*Use Cases7*

*Interactive Diagrams11*

*User Interface12*

*Test Cases13*

High Level Time Line Diagrams17

*Gantt Chart17*

Appendix18

*Prototypes18*

*Sequence Interaction Diagrams19*

*Category Interaction Diagrams21*

*References23*

Glossary24

**Function Point Cost Analysis25**

**Work Shared Document (WSD)27**

**Resumes28-33**

Bridgette Reisinger28

Shaina Daya29

Umesh Velagapudi30

Shuaib Mushfiq31

Shamiya Gissentanna32

Angelica Charles33

**User Guide34-42**

Student Manual34

Administrator Manual41

**Project Legacy 43**

**Source Code44**

Topic

# Text Editor

For our project, we'd like to design an executable text editor. As the semester progresses and we get a better grasp of the complexity expected of the project and what we can produce, we could add additional features.

REquirements Analysis Document

## 1 Introduction

* 1. Overview

There are countless numbers of text editors for consumers to create, edit, and/or view text based documents. However, the countless text editors are not made to help elementary/middle school students better grasp the functions of the editor. The purpose of this project is to help elementary/middle school students with the technological skills needed to create a media document to advance in upper level education and upper level jobs. The editor helps to introduce students to the 21st century technological advancements needed to excel in their social, educational, personal, and professional lives. Finally our last motive for this project was to create an easy to use text editor that will not only be fun to use, but also easy to use.

* 1. Scope of the System

AB Editor is a text editor that will help students learn how to create, open, save, cut, copy, paste, find, and exit a text based document effeciantly. The editor will have fun and creative logos for each of the functionalities to help students understand what each function does. Also, the editor will also have short cuts to undo a mistake easily rather than tapping on the backspace key to undo it.

* 1. Objectives and Success Criteria of the Project

The success of the application depends upon meeting the following core set of objectives:

* + 1. The design of a text editor to encapsulate all executable functions for elementary/middle school students.
    2. The design of a fun based text editor that will gain the students attention.
    3. The design of an easy to use text editor that the students can grasp

1. Current System
   1. Functionality

The AB Text Editor will provide a user-friendly environment to create and modify documents at the user's leisure. The current system is designed with this identical purpose, offering a more verbose setting for simple interactions. The AB Text Editor, hereby referred to as "the editor", will target children and create a workspace that will familiarize younger generations with this form of software in a simplistic manner. Images will be used as icons to help the users relate to their usage. Different colors and shapes can be used to track progress and to make comprehension simple. The editor will provide an easy-to-use medium for daily use.

1. Future System
   1. Problem Statement

3.1.1 Our client, Bridgette Reisinger, has asked our team of software developers to create a text editor. Bridgette has asked our team to complete the project by December 03, 2014. The reason why we are creating this text editor is because our client wanted a personal text editor for children with very specific functionalities.

3.1.2 The program shall allow for creation of new text files and modification of existing text files. The program shall allow this with text files of the .txt format. The program shall encode these text files with the UTF-8 character encoding. The program shall only have one file open per instance of the program. The program shall be written in Java. The program shall be deployed as an executable .jar file.

3.1.3 The program shall allow for text entry and deletion via the keyboard. The program shall allow for a spell check function to allow users to check accuracy of spelling. The program shall allow for the deletion, copying, and cutting of multiple characters via selecting, or highlighting, multiple characters. The program shall allow for the pasting of text within the text file. The program shall allow for the printing of the text file. The program shall allow for the user to search for a specific string of characters within the file, receive information as to whether the string exists, and be directed to where the string is in the file. The program shall save the file in the computer's file system according to a name and location provided by the user. The program shall notify the user if it was unable to save the file. If an existing file was opened by the program, the program shall save the file to the existing name and location. If the user has made modifications to a file and exits the program before saving them, the program shall alert the user to this and give them an opportunity to save before exiting.

3.1.4 Continuing on, the program will have the team logo in the bottom right hand corner of the user interface. The program shall have small icons designated for each functionality. The functionalities that will include icons are save, load, cut, copy, paste, print and exit. The interface is designed for children that are around the ages of 6-10. The program shall also have a word count function at the bottom left hand side of the user interface.

* 1. Requirement Traceability Matrix

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Entry #** | **Para. #** | **Requirement Descriptions** | **Type** | **Category** | **Use Case ID** |
| 1 | 3.1.3 | The program shall allow for creation of new text files and modification of existing text files. | SW | F | 4 |
| 2 | 3.1.2 | The program shall allow this with text files of the .txt format. | SW | F | 1,2,3,4,5,6 |
| 3 | 3.1.1 | The program shall encode these text files with the UTF-8 character encoding. | SW | F | 1,2,3,4,5,6 |
| 4 | 3.1.2 | The program shall allow for text entry and deletion via the keyboard. | SW, HW | F | 1,2,3,4,5,6 |
| 5 | 3.1.3 | The program shall allow for the deletion, copying, and cutting of multiple characters via selecting, or highlighting, multiple characters. | SW | F | 5,6 |
| 6 | 3.1.3 | The program shall allow for the pasting of text within the text file. | SW | F | 3 |
| 7 | 3.1.3 | The program shall allow for the printing of the text file. | SW, HW | F | 1 |
| 8 | 3.1.3 | The program shall allow for the user to search for a specific string of characters within the file, receive information as to whether the string exists, and be directed to where the string is in the file. | SW | F | 2 |
| 9 | 3.1.3 | The program shall save the file in the computer's file system according to a name and location provided by the user. | SW | F | 1 |
| 10 | 3.1.3 | The program shall notify the user if it was unable to save the file. | SW | F | 1 |
| 11 | 3.1.3 | If an existing file was opened by the program, the program shall save the file to the existing name and location. | SW | F | 4 |
| 12 | 3.1.3 | If the user has made modifications to a file and exits the program before saving them, the program shall alert the user to this and give them an opportunity to save before exiting. | SW | F | 1 |
| 13 | 3.1.2 | The program shall only have one file open per instance of the program. | SW | F | 1 |
| 14 | 3.1.1 | The program shall be written in Java. The program shall be deployed as an executable .jar file. | HW, SWC | F | 1,2,3,4,5,6 |
| 15 | 3.1.4 | The program will have the team logo in the bottom right hand corner of the user interface. | HW, SWC | NF | 4 |
| 16 | 3.1.4 | The program shall have small icons designated for each functionality, including icons such as save, load, cut, copy, paste and exit. | HW, SWC | NF | 1,2,3,4,5,6 |
| 17 | 3.1.4 | The interface is designed for children that are around the ages of 6-10. | HW, SWC | NF | 1,2,3,4,5,6 |
| 18 | 3.1.4 | The program shall also have a word count function at the bottom left hand side of the user interface. | HW, SWC | NF | 4 |
| 19 | 3.14 | The program shall allow for a spell check function to allow users to check accuracy of spelling. | HW,SW | F | 6 |

1. System Model
   1. Use Cases
      1. Save

|  |  |
| --- | --- |
| RTM Entry: | The program shall save the file in the computer's file system according to a name and location provided by the user. |
| Use Case ID:  Use Case Name:  Use Case Participants: | 1  Save Functionality AB-Editor  Editor User |
| Entry Condition: | Student/User clicks on the Save Icon on the top toolbar |
| Flow of Events: | 10 Steps to the Save Functionality |
|  | 1. The user will type content into the editor. 2. The user will click on the save icon on the top toolbar. 3. If there is no content in the editor then it will prompt the user to see if he/she wants to save a document that is blank. After the user types content, if there is content in the editor then it will prompt the user to specify a name for the document. 4. If the desired name is already in use then the editor will ask the user to type in another name that is not a duplicate. 5. Otherwise, the editor will then ask the user to specify a location to save the file onto the system. 6. Finally, the editor will save the desired document into the specified location. 7. The editor will then allow the user to resume editing the document once the save functionality has been finished. |
| Exit Condition: | The editor saves the file in the specified location designated by the user. |
| Quality Constraints: | Each pop up prompt from the program should take no longer than 2 seconds to load. |

* + 1. Find

|  |  |
| --- | --- |
| RTM Entry: | The program shall allow for the user to search for a specific string of characters within the file, receive information as to whether the string exists, and be directed to where the string is in the file. |
| Use Case ID:  Use Case Name:  Use Case Participants: | 2  Find Functionality AB-Editor  Editor User |
| Entry Condition: | Student/User clicks on the Find Icon on the top toolbar |
| Flow of Events: | 9 Steps to the Find Functionality |
|  | 1. The user selects the AB editor icon to run the text editing program. 2. There will be a blank document immediately available after the program has been run. 3. The user will type content into the editor. 4. The user will click on the find icon on the top toolbar. 5. The editor will prompt the user with a pop up message asking what word or phrase the user would like to find in the document. 6. If the word or phrase cannot be found then the user will be prompted of this. 7. If the editor does find the word or phrase, then the program will prompt the user by notifying them where the word has been found. 8. The editor should highlight all the instances of where the word/phrase has been found. 9. Once the functionality has fulfilled its purpose then the editor will let the user to continue editing the document. |
| Exit Condition: | The editor finds the designated word or phrase and notifies the user by highlighting all instances of the word/phrase in the document. |
| Quality Constraints: | The find functionality should take no longer than 2 seconds for any document that is one page or less. |

* + 1. Paste

|  |  |
| --- | --- |
| RTM Entry: | The program shall allow for the pasting of text within the text file. |
| Use Case ID:  Use Case Name:  Use Case Participants: | 3  Paste Functionality AB-Editor  Editor User |
| Entry Condition: | Student/User clicks on the paste Icon on the top toolbar |
| Flow of Events: | 10 Steps to the Paste Functionality |
|  | 1. The user selects the AB editor icon to run the text editing program. 2. There will be a blank document immediately available after the program has been run. 3. The user will type content into the editor. 4. The user will highlight a word or phrase that the user wants to copy. 5. The user will then either use the cut or copy function by clicking on the either icon on the top toolbar. 6. If the user clicked the cut functionality then the program should delete the highlighted word/phrase and then have it saved so that it can be pasted in a different location. 7. If the user clicked the copy functionality then the program should not delete the highlighted word/phrase and then have it saved so that it can be pasted in a different location. 8. The user should find the desired location of where to paste the text and place the cursor in the appropriate position. 9. The user should then press the paste icon on top toolbar which will paste the copied/cut text into the desired location. 10. The editor will allow the user to continue modifying the text after the functionality has fulfilled its purpose. |
| Exit Condition: | The editor pastes the copied/cut word or phrase in the specified location designated by the user. |
| Quality Constraints: | The process of pasting should not take longer than two seconds unless the content is extremely lengthy. |

* + 1. Load

|  |  |
| --- | --- |
| RTM Entry: | The program shall allow for modification of existing text files. |
| Use Case ID:  Use Case Name:  Use Case Participants: | 4  Load Functionality AB-Editor  Editor User |
| Entry Condition: | Student/User clicks on the load Icon on the top toolbar |
| Flow of Events: | 7 Steps to the Save Functionality |
|  | 1. The user selects the AB editor icon to run the text editing program. 2. There will be a blank document immediately available after the program has been run. 3. The user will select the load functionality. 4. The program will show the current files available to be loaded. 5. User selects the desired file to be loaded. 6. Program loads the file for the user. 7. The editor will then allow the user to review the selected document or edit. |
| Exit Condition: | The editor opens the file specified by the user. |
| Quality Constraints: | The file selected by the user should not take more than 4 seconds to load. |

* + 1. Copy

|  |  |
| --- | --- |
| RTM Entry: | The program shall allow for the copying of multiple characters via selecting or highlighting. |
| Use Case ID:  Use Case Name:  Use Case Participants: | 5  Copy Functionality AB-Editor  Editor User |
| Entry Condition: | Student/User clicks on the copy Icon on the top toolbar |
| Flow of Events: | 10 Steps to the Copy Functionality |
|  | 1. The user selects the AB editor icon to run the text editing program. 2. There will be a blank document immediately available after the program has been run. 3. User can create a new text file 4. Otherwise, loads an existing file. 5. If user creates a new text file, the user inputs the desired characters. However, if loading an existing file, user will select the load functionality 6. User highlights the desired characters by clicking and holding the left mice button and moving the cursor across it. 7. After the characters have been highlighted, the user selects the copy functionality located on the top of the editor. 8. The program saves the copies characters for use by the user. 9. User pastes the characters in the specified location. . 10. The editor will then allow the user to continue reviewing the selected document for editing. |
| Exit Condition: | The editor copies the characters specified by the user. |
| Quality Constraints: | The characters selected by the user should not take more than 2 seconds to copy. |

* 1. Interactive Diagrams
     1. Load

The user clicks on the icon and opens up a new document. It allows the user to edit. The load functionality is the most important in order to open the application and prepare it for use.

* + 1. Copy

After editing, the user may want to select pieces of text and move them to a different location within the document. Copy allows the user to duplicate the text into a temporary clipboard where the text is saved and can be placed at the user’s discretion. Copy also allows the highlighted text to remain unaltered.

* + 1. Paste

The paste functionality takes the string of texts from the clipboard and inserts the text where the cursor is. The paste function is shown as a glue bottle on the top of the screen.

* + 1. Find

The find functionality has a magnifying glass because children are familiar with correlating the magnifying glass to searching for something. In addition, the magnifying glass is a universal icon for programs find function when the program does feature a find function. The find function prompts the user for a string and when the user clicks ok the program locates every occurrence of given string in the text field. The program alerts the user of these occurrence in a second prompt. The function then highlights every occurrence of the given string within the text field. When the user resumes typing the strings are no longer highlighted.

* + 1. Spell Check

The spell check function underlines misspelled words entered in the text editor to be corrected. We chose to include this functionality to help children become familiar with proper spelling.

* + 1. Save

We chose the save function so that children can keep up with their work. We originally had the piggy bank so children would relate to it, but then we realized we should use the floppy disk since it is the universal symbol for save.

* + 1. Print

The print functionality opens the print dialog after the user clicks the print icon. This enables the user to print the document created in the text editor. We chose this functionality because children should have the opportunity to print documents.

* 1. User Interface
     1. We have provided a prototype for the client. The prototype is just a visual representation of what the text editor will look like. The prototype also shows all the functionalities of what will be provided in the finished project.
     2. The eight functionalities are new, open, save, cut, copy, paste, find, and exit. New will open up a next document for the user. Save will save any data that has been typed into the text editor. Open will open any previously saved files. Cut will delete a certain amount of text but it also allows for the functionality of pasting the deleted text into another location. Cut is an exception function, due to being accessed through a shortcut command through the keyboard instead of an icon. Copy will copy a certain amount of text. Paste will allow the user to paste the copied text into another location. The find functionality will allow the user to find a keyword or phrase in the document. The final functionality that is required for most programs is the exit functionality. The top right icon will allow for the program to exit.
     3. Another functionality that we decided to add was the word counter at the bottom left portion of the interface. We also decided to add our group logo at the bottom right of the text editor. This is the current prototype that our group of developers have designed.
  2. Test Cases
     1. Load

|  |  |
| --- | --- |
| *Test-case Identifier* | Load |
| *Feature to be Tested* | Specified document is opened for viewing and editing |
| *Feature Pass/Fail Criteria* | This test passes if a desired file specified by the user opens in the editor window for reviewing within 4 seconds of selection. |
| *Means of Control* | An Open object is created and the openFile() method called. |
| *Data* | The file name and path are selected by the user via a JFileChooser object. The bytes are read in from the selected file and saved as a String to the Open object's fileText field. The UserInterface object then sets the stage text to match the fileText. |
| *Test Procedure* | The test is started by clicking once on the 'Open' icon found in the top toolbar. The test opens a dialog box, requesting further input. The test is complete once the dialog box disappears and the selected file is open for review. This shall not take longer than 5 seconds. |
| *Special Requirements* | The executable for AB Text Editor must be installed onto the system and is needed for the test execution. |

* + 1. Copy

|  |  |
| --- | --- |
| *Test-case Identifier* | Copy |
| *Feature to be Tested* | Specific characters are duplicated into the system clipboard |
| *Feature Pass/Fail Criteria* | This test passes if all characters highlighted by the user are duplicated into the system's clipboard within 2 seconds of copying. |
| *Means of Control* | A Clipboarder object is created and the toClipboard() method is called. |
| *Data* | The string of selected text in the stage is sent to the toClipboard() method, which in turn accesses the system clipboard. The text is set as the contents of the system clipboard. The selected text also remains on the stage. |
| *Test Procedure* | The test is started by clicking once on the 'Copy' icon found in the top toolbar. The test will run with no further interference until completion. The test should not take longer than 3 seconds. |
| *Special Requirements* | Access to the system clipboard is needed for the test execution. |

* + 1. Paste

|  |  |
| --- | --- |
| *Test-case Identifier* | Paste |
| *Feature to be Tested* | Most recent cut/copied text in the system clipboard is placed into desired location by the user |
| *Feature Pass/Fail Criteria* | This test passes if the last selection of characters placed into the system's clipboard by the user is removed from this clipboard and placed into the desired location within the open document in the editor. |
| *Means of Control* | A Clipboarder object is created and the fromClipboard() method is called. |
| *Data* | The fromClipboard() method accesses the system clipboard. If the data on the system clipboard is text, it returns the text as a String and the String is inserted at the carat's current location. If text data is not available on the system clipboard, the String returned is null. |
| *Test Procedure* | The test is started by clicking once on the 'Paste' icon located in the top toolbar. The test will run with no other interruptions until completion. The test should not take longer than 3 seconds. |
| *Special Requirements* | Both the test stubs Cut or Copy and access to the system clipboard are needed for the test execution. |

* + 1. Find

|  |  |
| --- | --- |
| *Test-case Identifier* | Find |
| *Feature to be Tested* | Specific words or phrases are found and highlighted |
| *Feature Pass/Fail Criteria* | The test passes if all instances of designated words or phrases indicated by the user are found and highlighted throughout the open document and takes no longer than 2 seconds to do so. |
| *Means of Control* | A Find object is created and its findText() method is called. |
| *Data* | Highlighted text will be copied into the Clipboard for temporary use. A duplicated version of the text is searched for via recursion throughout the entire document. Each character of the highlighted text is separated into an array, with each element being searched for individually. True results are highlighted within the document, while false results are bypassed. |
| *Test Procedure* | The test is started by clicking once on the 'Find' icon found in the top toolbar. The test opens a dialog box and waits for further input. The test is complete once the dialog box disappears and the appropriate text is highlighted. This test should take no longer than 6 seconds. |
| *Special Requirements* | Text within the editor window is needed for the test execution. |

* + 1. Spell Check

|  |  |
| --- | --- |
| *Test-case Identifier* | Spell-Check |
| *Feature to be Tested* | Editor searches for misspelled words and highlights them for correction |
| *Feature Pass/Fail Criteria* | This test passes if the editor compares each string within the open document to a list of already-defined words and indicates whether the string is misspelled to the present list of words within a dictionary. |
| *Means of Control* | A Spell Check object is created and its corresponding initializeSpellCheck() method is called |
| *Data* | The libraries from the spellCheck jortho library is imported upon initial method call. Once the main editor window has been edited with text, the spell check compares each word to the compiled dictionary of 3million words. If a word does not exist, it will be underlined with a red line. This will enable the window to allow the word to be right-clicked with the mouse. The popup menu appears via the configurePopup() method and displays suggestions, along with the option to add the word to the user dictionary. |
| *Test Procedure* | The test is started by clicking once on the 'Spell Check' icon found in the top toolbar. The test opens a dialog box, awaiting further input. The test is complete once the dialog box disappears. The test should not take longer than 7 seconds. |
| *Special Requirements* | The Java dictionary library is needed for the test execution. |

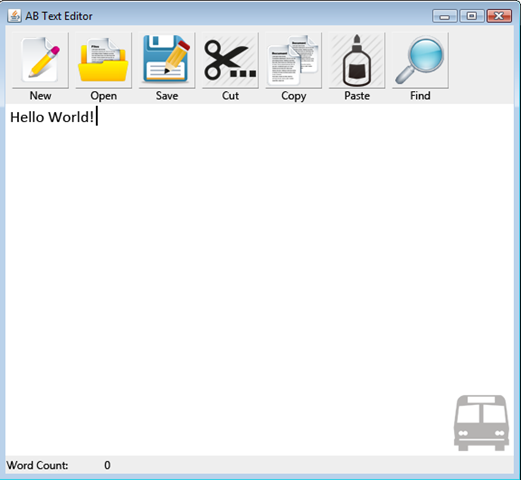
* + 1. Save

|  |  |
| --- | --- |
| *Test-case Identifier* | Save |
| *Test Location* |  |
| *Feature to be Tested* | All changes within current document are saved into memory |
| *Feature Pass/Fail Criteria* | The test passes if the open document is saved successfully under a particular name into a specified location and takes no less than 2 seconds to save. |
| *Means of Control* | A Save object is created and its saveFile method is called. |
| *Data* | The String matching the text on the stage is sent to the method. The method creates a JFileChooser object and calls its showSaveDialog method, which allows the user to pick a file location and name. When the user hits save, the ".txt" extension is appended to the file name and the recieved String is written to the file. |
| *Test Procedure* | The test is started by clicking once on the 'Save' icon found in the top toolbar. The test then opens a dialog box and awaits further input. The test is complete once the dialog box disappears and the open document has been saved. The test should not take longer than 5 seconds. |
| *Special Requirements* | Unique, non-existing document titles are needed for the test execution to succeed. |

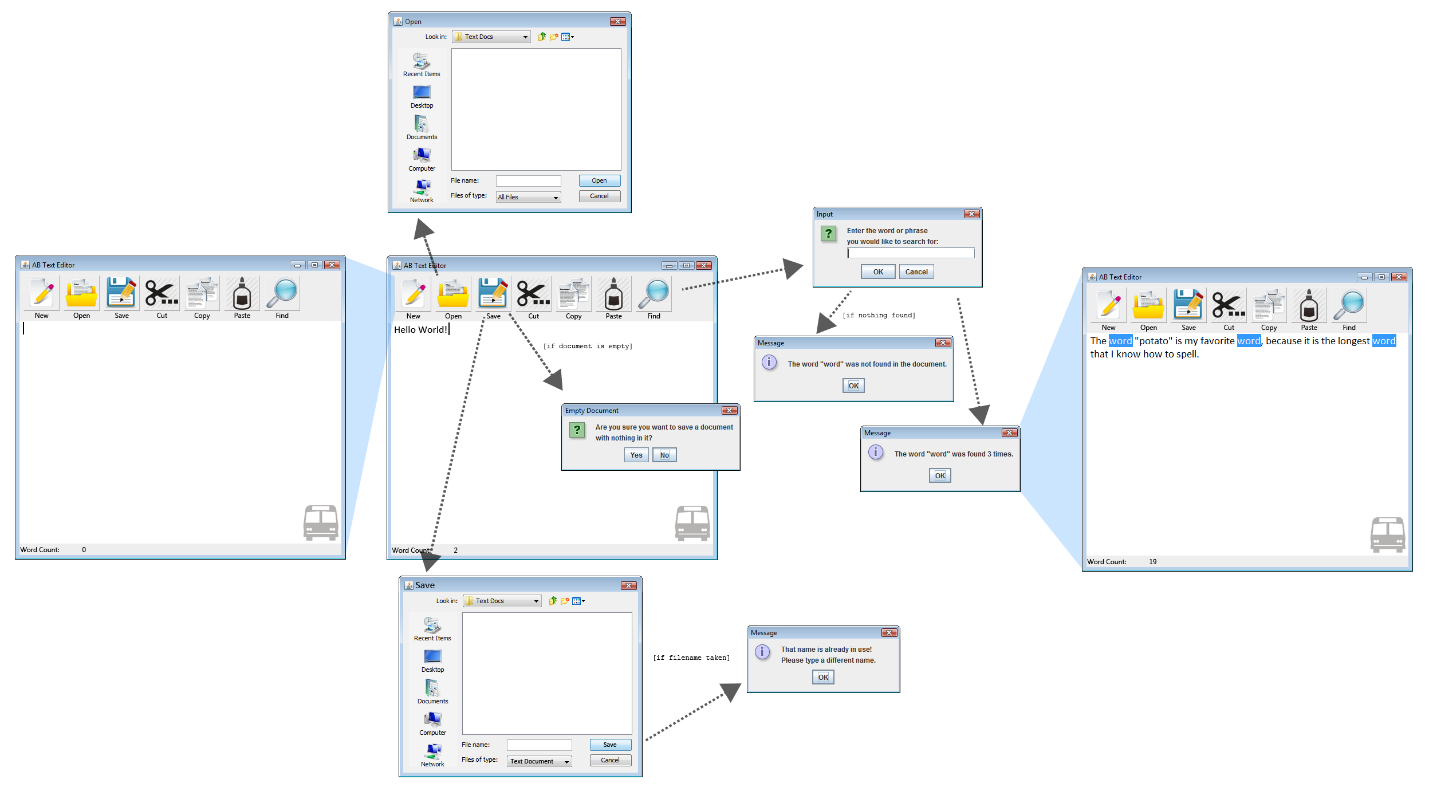
* + 1. Print

|  |  |
| --- | --- |
| *Test-case Identifier* | Print |
| *Feature to be Tested* | Current document within the editor is sent to system's printer |
| *Feature Pass/Fail Criteria* | This test passes if the editor opens a dialog box requesting the user to specify which system printer to send the open document to for printing. |
| *Means of Control* | A Print object is created and its printDocument() method is called. |
| *Data* | The printDocument() method calls upon the system's devices. A separate system prompt is called, accessing all devices and printers currently installed on that system. Once a selection has been made, the system's dialog box disappears, and the entire document is sent to the desired device to be printed as a hard copy. |
| *Test Procedure* | The test is started by clicking once on the 'Print' icon found in the upper toolbar. The test opens a dialog box for further input. The test is complete once the dialog box disappears and the document is sent to the system printer. The test should not take longer than 5 seconds. |
| *Special Requirements* | Access to system printers and devices are needed for test execution. |

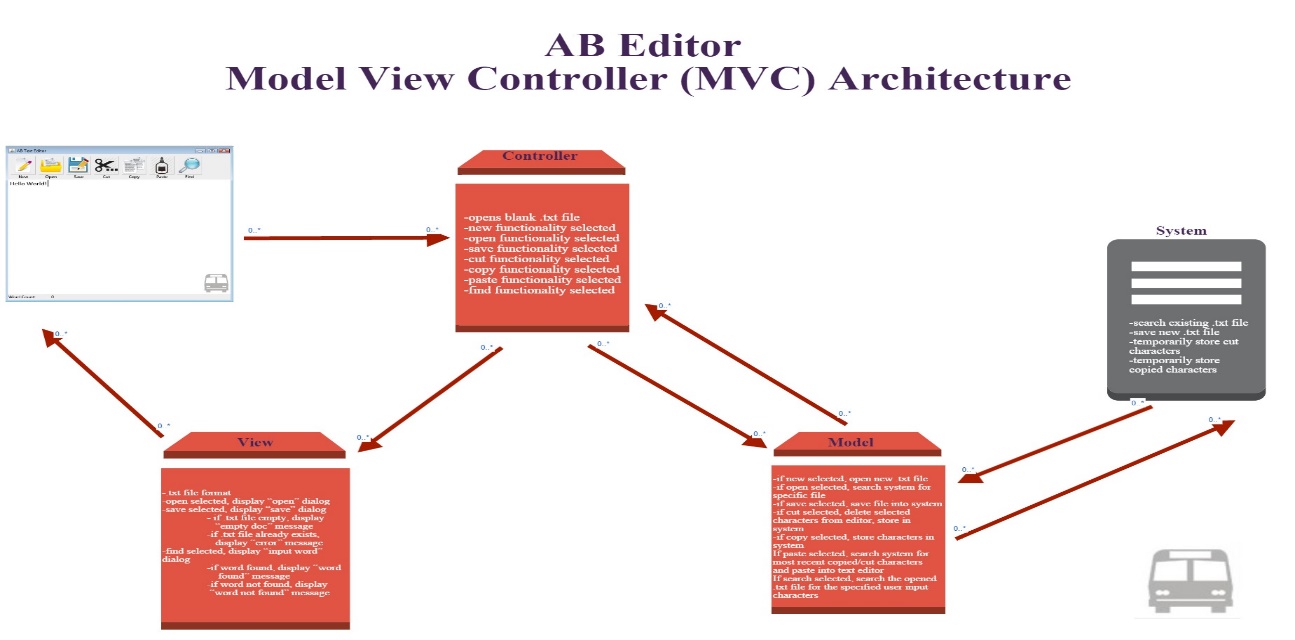
1. High Level Time Line Diagrams 
2. Appendix
   1. Prototypes
      1. Visual Design



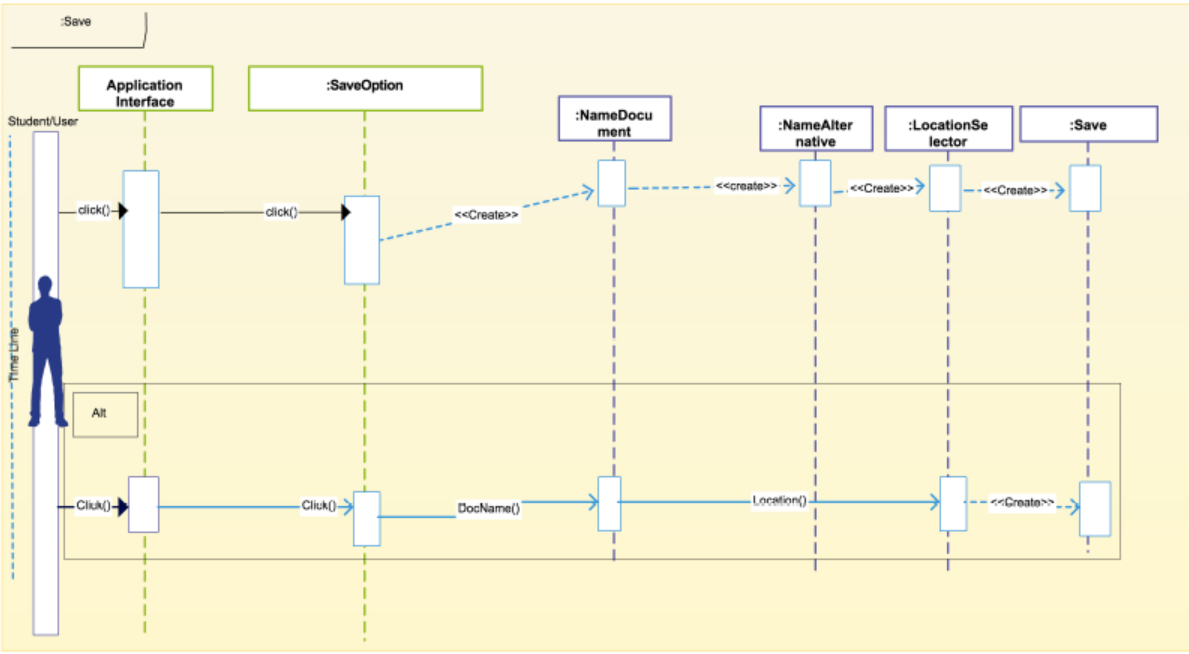
* + 1. Horizontal Prototype



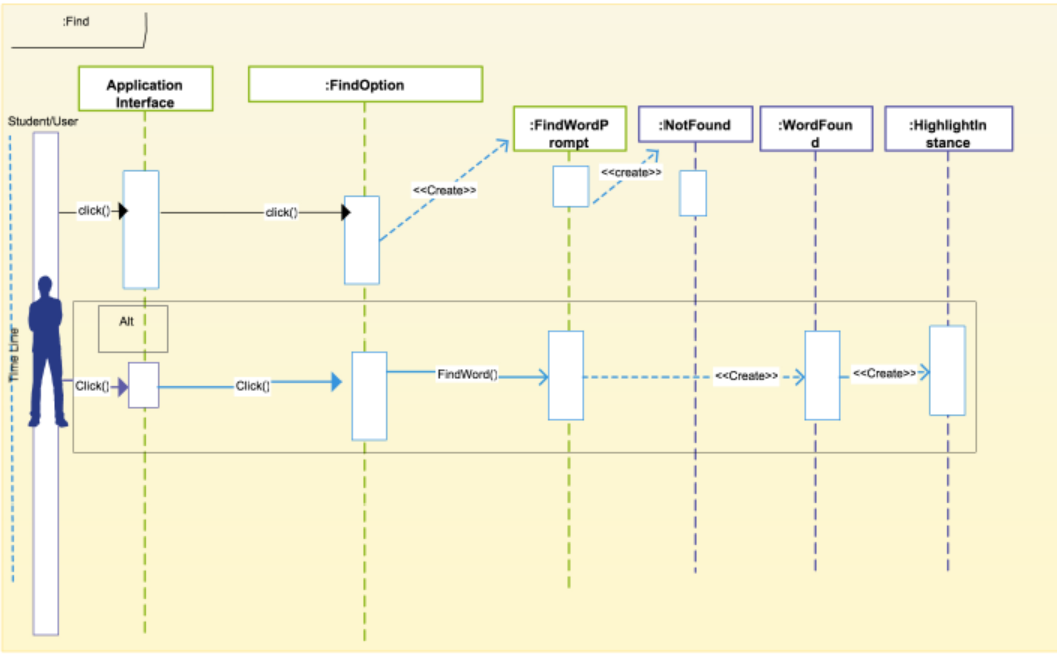
* 1. Software Architecture



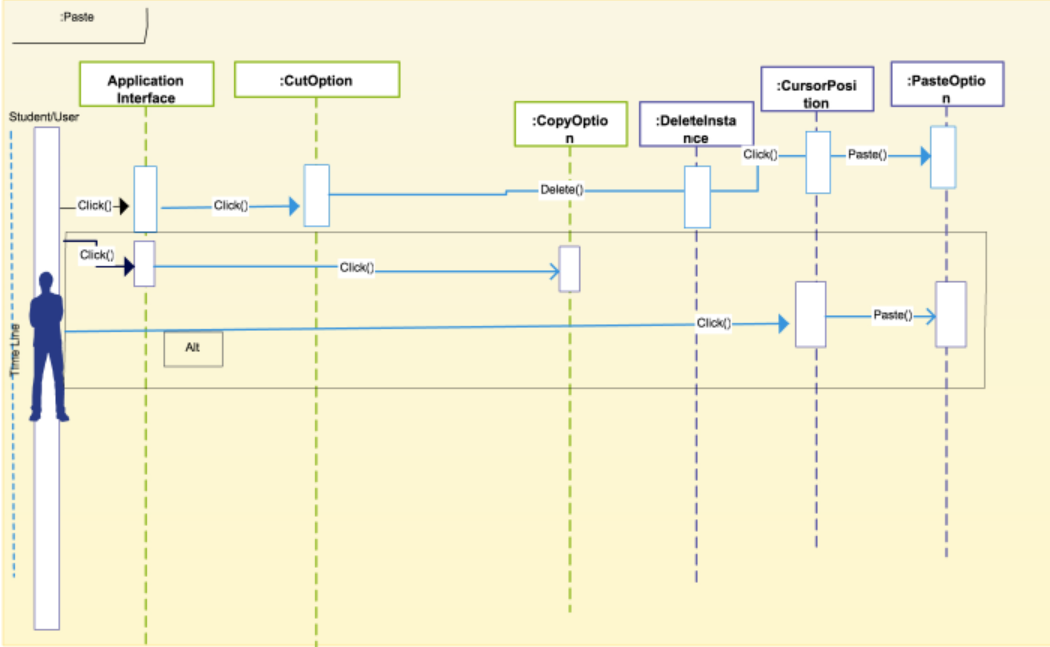
* 1. Sequence Interaction Diagrams
     1. Save



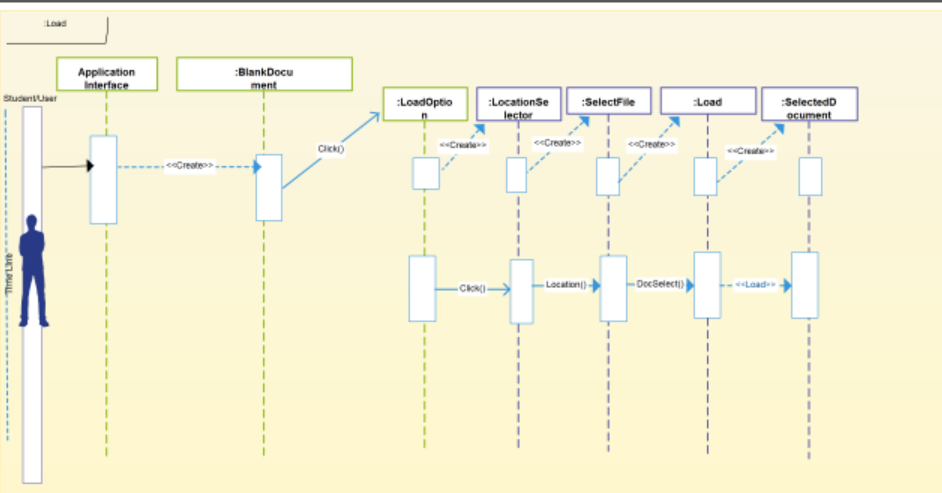
* + 1. Find



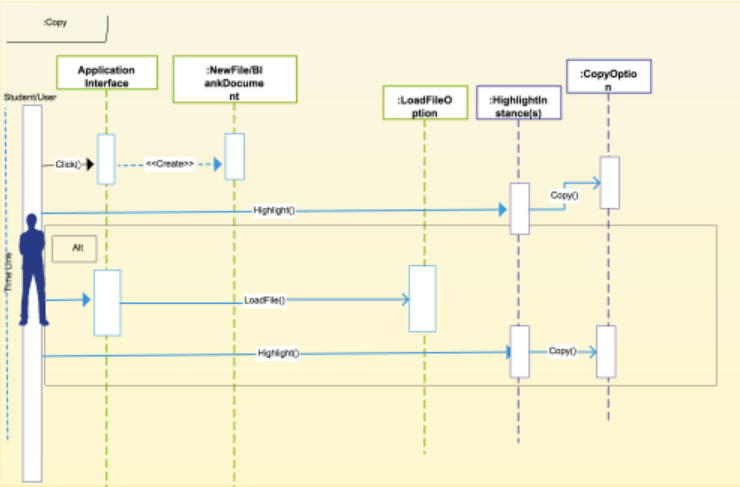
* + 1. Paste



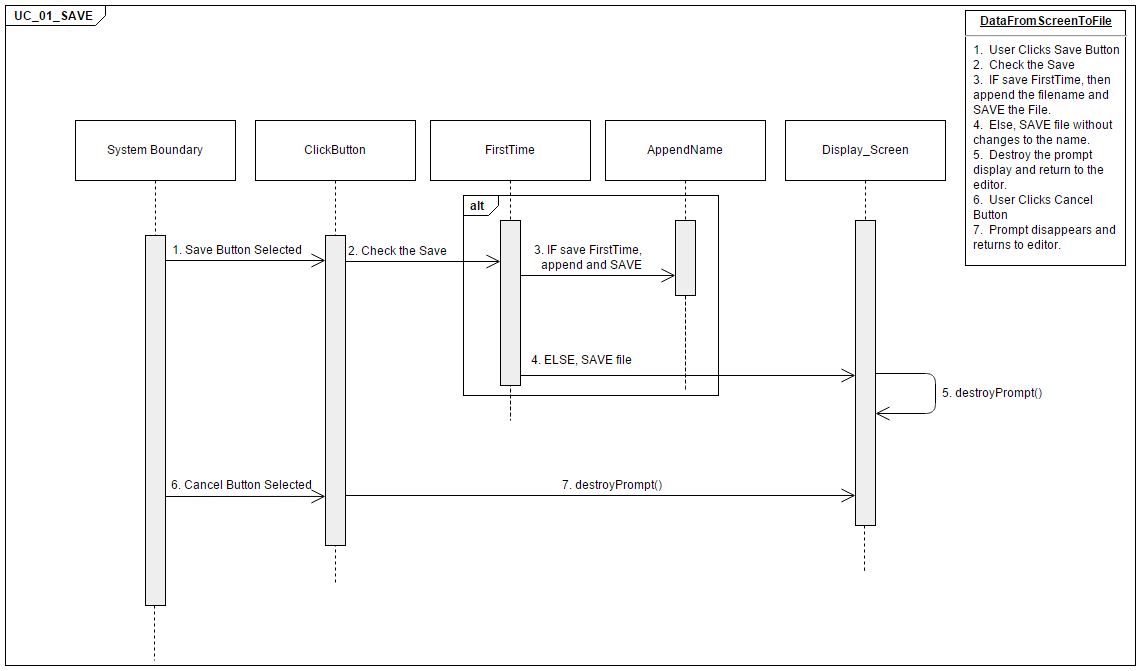
* + 1. Load



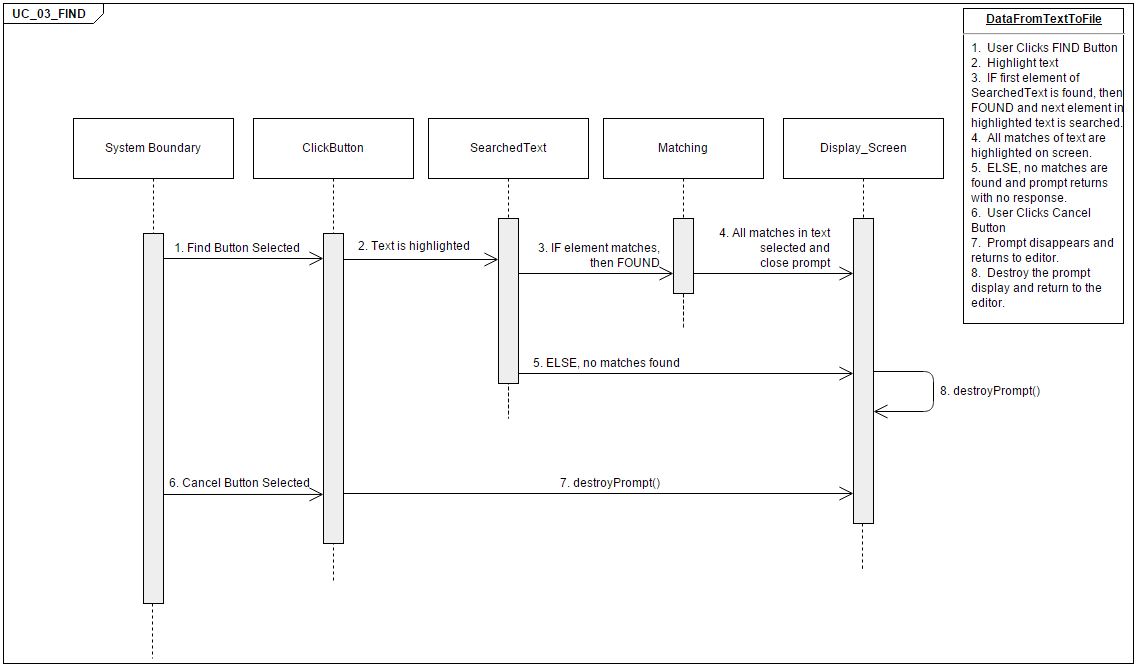
* + 1. Copy



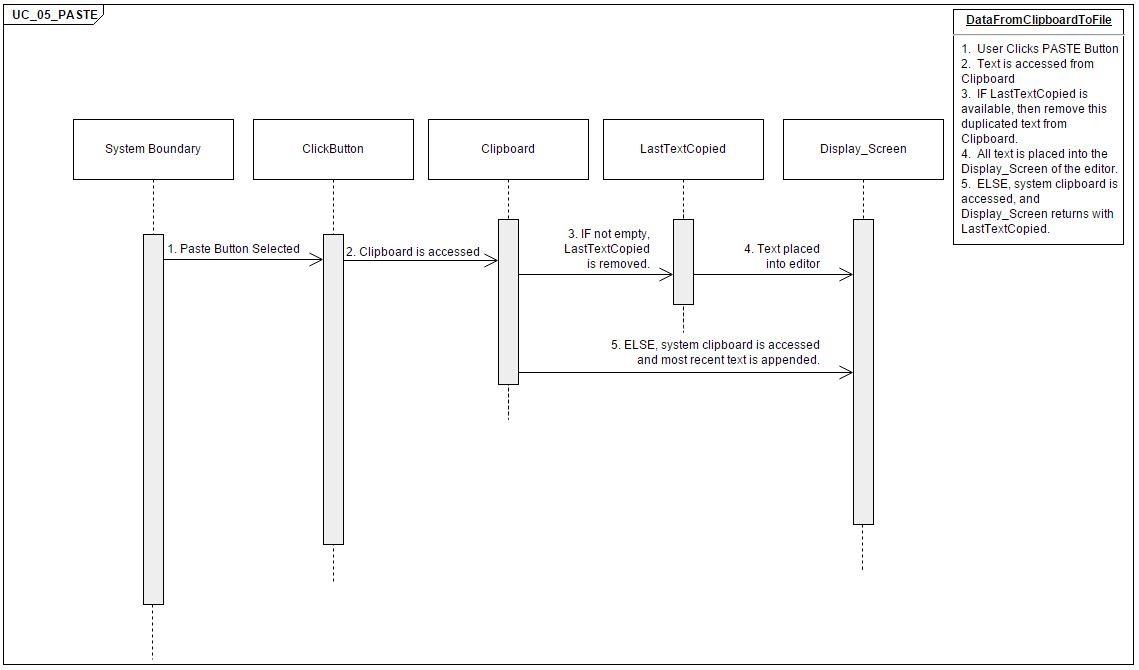
* 1. Category Interaction Diagrams
     1. Save



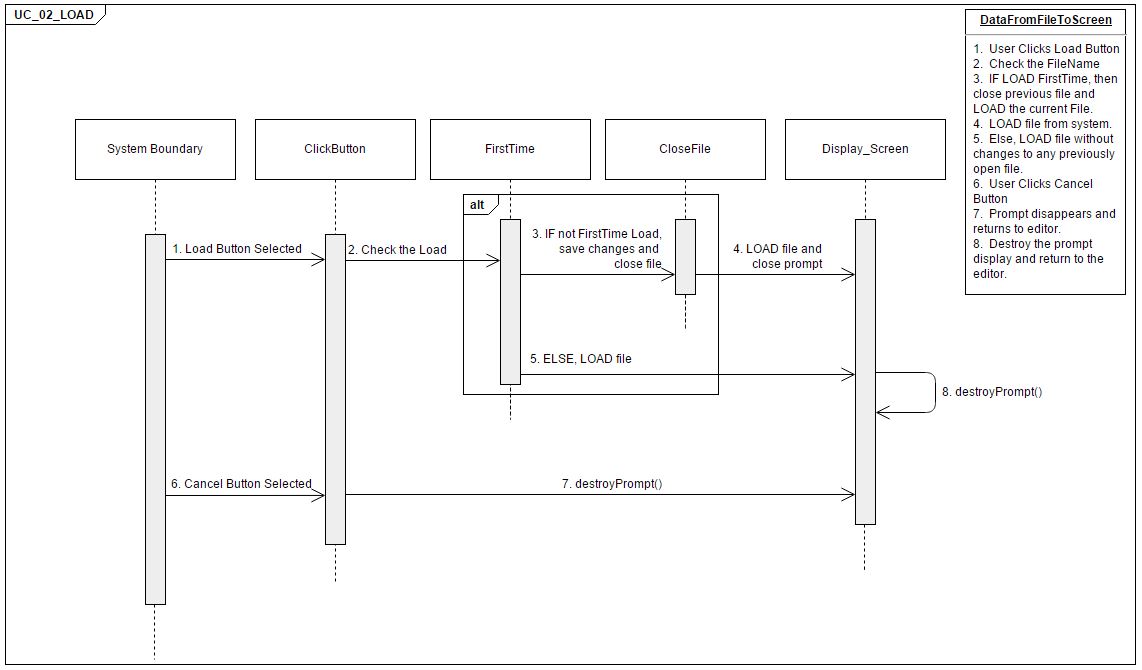
* + 1. Find



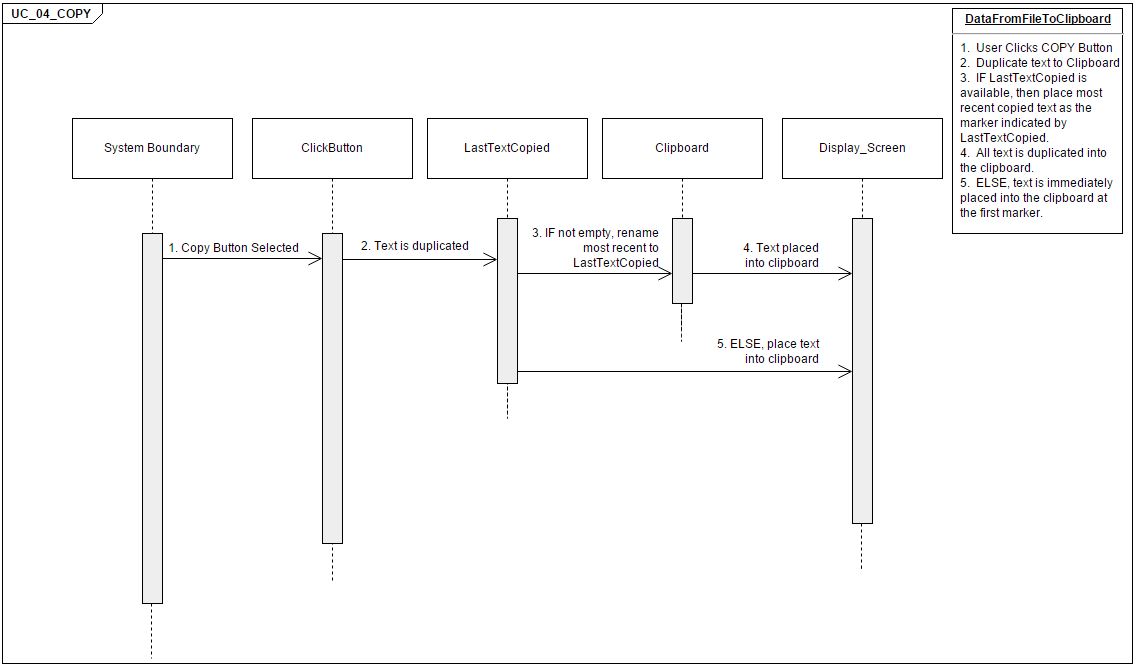
* + 1. Paste



* + 1. Load



* + 1. Copy



* 1. References

Important references will be added as the team continues to progress in the development of the project.

1. Glossary
   1. *.txt format*

A file format for files consisting of text usually containing very little formatting. Files within the .txt extension can easily be read or opened by any program that reads text and is considered universal.

* 1. *Copying*

A feature in text editor use to improve the formatting, style, and accuracy of text. The copying feature creates a duplicate of selected data from its original position.

* 1. *Cutting*

A feature in text editor. It is a command that removes the selected data from the original position.

* 1. *Executable.jar*

A file format is aggregate many Java class files and its associated data into one file to distribute application software or libraries. They are archive files build on the zip file format and have a .jar file extension. Archive files are used to collect multiple data files together and for single file for easier portability and storage or simply to compress files to use less storage space.

* 1. *File system*

A software which organizes files on the storage medium such as a disk drive. It is used to facilitate access the files and directories. It is used to separate data into individual pieces by giving them names which makes it easy to separate and identify.

* 1. *Find*

A function used in the text editor for the user to search for the inputted string

* 1. *Pasting*

When copying and pasting the selected data is placed in a clipboard, which is a software facility used for a short time data storage. The data stored in the clipboard is in inserted in the position that the paste command is used.

* 1. *Spell Check*

A function used in the text editor to verify correct spelling for the user input

* 1. *Text editor*

A type of program designed to use for editing text files. Can be used to change files.

* 1. *Text file*

A computer file structured as a sequence of lines of electronic text (meaning document is read in digital form). It exists within the computer file system text files can contain plain text but they are not limited to such.

* 1. *Text highlighting*

A feature of text editors used for programming or scripting. This feature displays text in different colors and fonts. It does not affect the meaning of the text it is a form of notation meant to reinforce text.

* 1. *Unicode International Encoding*

Standard for use with different languages and scripts. Each letter digit or symbol is a find a unique numeric value that applies across different platform and programs.

* 1. *Utf 8-character Encoding*

This is the most common character encoding. It means universal character set transformation format 8-bit. Character encoding is capable of encoding all possible characters to Unicode. The encoding is variable length and uses 8-bit code units.

Function Point Cost Analysis



WSD

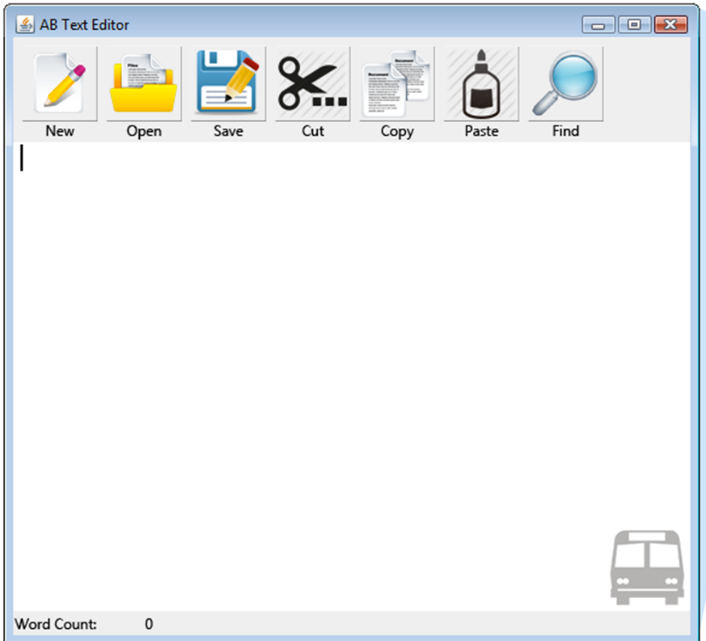
# Team Name: ASSBUS

|  |  |
| --- | --- |
| Group Member | Assigned Roll |
| SHAMIYA GISSENTANNA | Project Manager : WSD, RAD, Terminology, Gantt Chart, RTM, Project Legacy |
| SHAiNA DAYA | Developer, Test Cases, Category Interaction Diagrams |
| BRIDGETTE REISINGER | Developer |
| SHUAIB MUSHFIQ | Developer, Software Architecture |
| UMESH VELAGAPUDI | Category Interaction Diagrams; Developer |
| ANGELICA CHARLES | User Manuals, Sequence Interaction (Use Case) Diagrams |

User Guide

# Students’ Manual

1. About AB Editor
   1. AB Editor is a text editor that will help students learn how to create, open, save, cut, copy, paste, find, and exit a text based document efficiently. The editor will have fun and creative logos for each of the functionalities to help students understand what each function does.
2. Creating a New .txt File
   1. A .txt file is A file format for files consisting of text usually containing very little formatting. Files within the .txt extension can easily be read or opened by any program that reads text and is considered universal.
      1. Upon opening the AB editor a blank document will appear where text can be entered and edited.



1. Loading an existing .txt File
   1. Once the AB editor is opened, to load a file click on the open icon:



* 1. From there a prompt will open up where you can specify a location and .txt file to load. The Picture below is an example of what it should look like:

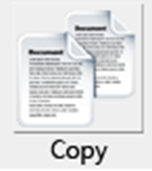


3.3 Click on the open option and the .txt file will load into the editor.

1. Cut, Copy, and Paste
   1. Highlight specific text/phrase using the cursor which would look like:



* 1. Select the copy icon:



* 1. Move the cursor to where the copied item will be placed and select the paste icon :



* 1. Text should now be pasted in desired area.
  2. If you would like to cut select the cut icon:



* 1. Move the cursor to where the cut item will be placed. The item will be deleted from its original position.
  2. Select the paste icon:

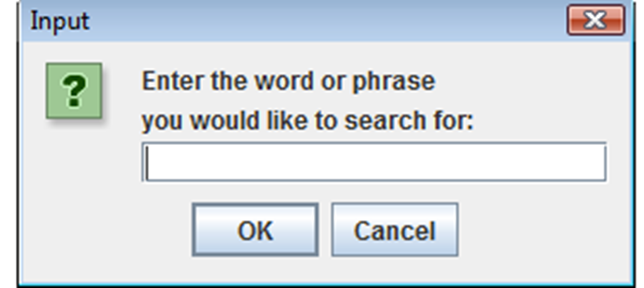


* 1. The selected text will now be pasted into the desired area.

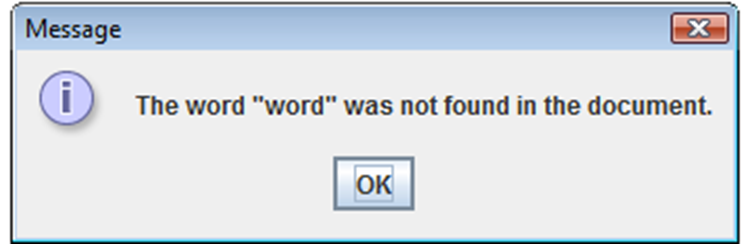
1. Find Text
   1. Select the find option in an open AB editor :



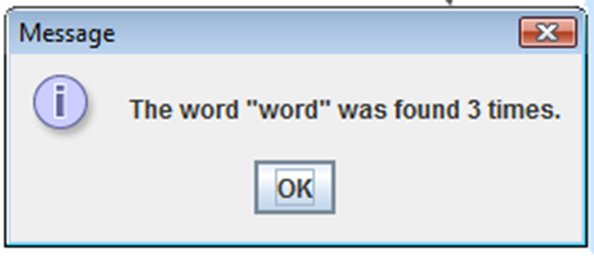
* 1. An open dialogue box will appear asking to enter the word or phrase you are searching for :



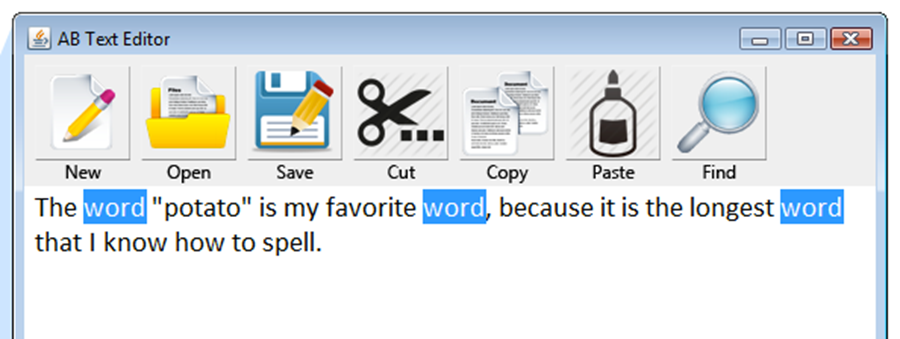
* 1. If nothing is found a message will pop up:



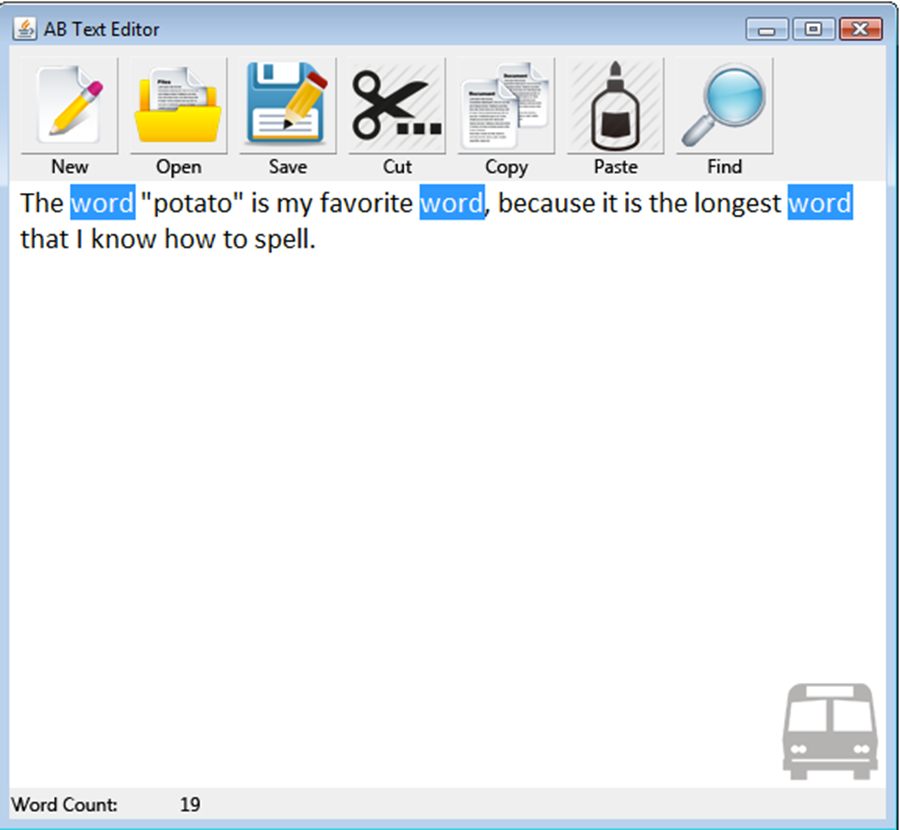
* 1. If the word was found a message will show saying that the particular word was found and how many times it was found. Example: Search for the word “word”. If “word” is found this is what it will look like:



* 1. If the text was found it will be highlighted in the text area of the AB editor:



1. Word Count
   1. With the word count feature you are able to receive a count for the amount of words within the .txt file that is opened in the editor. Word count is located at the bottom left of the editor:



1. Spell Check
   1. The spell check function will underline misspelled words.
2. Printing
   1. In order to print the .txt file select the print icon in the toolbar :

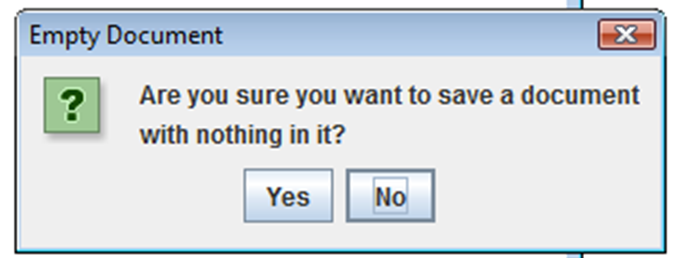


* 1. Text should now print

1. Saving an .txt File
   1. User will select the save icon:



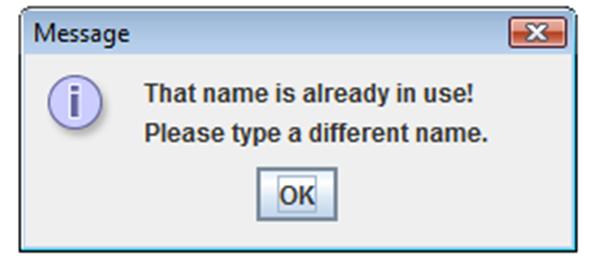
* 1. If the document is empty a message will appear:



* 1. If there is text a window will open giving the option to locate a destination and select a name:



* 1. If the name is already in use a message will appear:

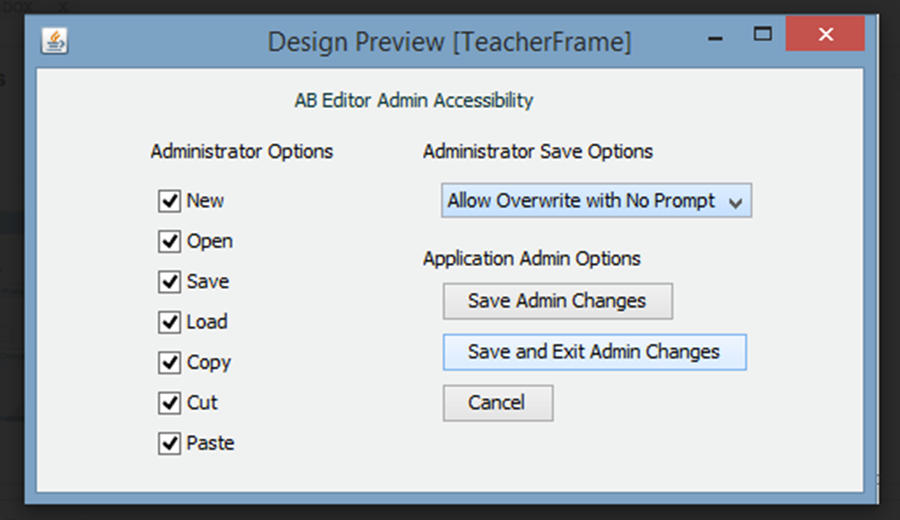


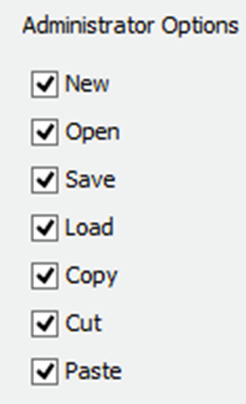
* 1. If the name does not already exist select the save button and the file is saved. You always want to save your work so that it will not get lost.

# Administrator Manual

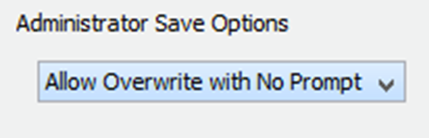
The administrator/ Instructor will be able to access and change settings in the AB editor.

Here is a preview to what it should look like:

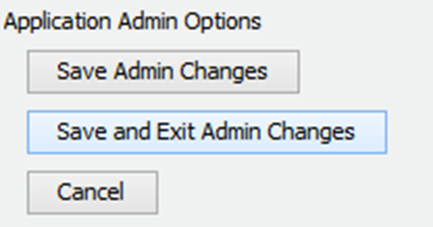


Here the administrator can check whether the functions listed above are on or off using a check.

The admin can also overwrite the save option to save without a prompt.

****

When all changes have been made the instructor can save the changes, save and exit, or cancel.

****

Project Legacy

Due to the time given we were only able to achieve the primary functions of a text editor targeted to elementary school adolescents and the instructor. If more time permitted, more detailed functions would have been added that would have strongly emulated a typical word processor but appealed to children. Functions such as page setup, document templates, and inserting objects in documents are just a few examples.

For the future endeavors of our project, we hope that elementary schools nationwide will be able to use the AB Editor as an educational tool for learning the fundamentals of word processing soft wares.