Software Requirements Specification

Madlib Editor

by
Sarah Richmond
Richard Trest
Michael Bashford
Aaron Powers
Chris Swiantkiewicz

Version 2
February 26th, 2014

Table of Contents

1. Introduction
    1.1 Purpose
    1.2 Scope
    1.3 Definitions, acronyms, and abbreviations
    1.4 References
    1.5 Overview

2. Tokens
    2.1 Token Overview
    2.2 Token Definitions
        2.2.1 Token
        2.2.2 User Defined Token
        2.2.3 Token Group
        2.2.4 Token Question
        2.2.5 Token Question Limit
    2.3 Token Interactions
        2.3.1 Tokens and Token Groups
        2.3.2 User Defined Tokens and Token Groups
        2.3.3 Token Groups and Token Questions
        2.3.4 Token Question and Token Question Limit

3. Features
    3.1 Starting a Madlib
        3.1.1 Creating a New Document
        3.1.2 Token Handling
    3.2 Importing a .txt file
    3.3 File type
    3.4 Preview Format
    3.5 Encoding
        3.5.1 User Defined Tokens
        3.5.2 Predefined Tokens
    3.6 Database

3.7 Auto Generating Tokens
3.8 Profanity Filter

4. Interface Layout
4.1 User interfaces
4.2 Menu Bar
4.2.1 File Menu
4.2.2 Edit
4.2.3 Tools Menu
4.2.4 Help Menu
4.3 Toolbar
4.4 Text Editor Panel
4.4.1 Basic Text Editing
4.4.2 Token Editing
4.4.3 Right click Options
4.4.4 Text Editor Overview
4.5 Group Panel
4.5.1 Overview of the group Panel
4.5.2 Creating groups
4.5.3 Adding tokens to groups
4.5.4 Removing tokens from groups
4.5.5 Deleting groups
4.6 Token Panel
4.6.1 Original Word List
4.6.2 Token List

5. Conclusion
5.1 Overview

1. Introduction

This section will briefly introduce all the information that will be in included in this document, the purpose of this document followed, and a list of definitions that the layman may not understand. The scope of the this software will be introduced, and then expanded upon in subsections. A list of references will be included. An overview of the software will be explained. As well as a section on suggested GUI design, this section is meant to allow the reader of this document have a clearer image in their head of what the software's GUI might look like in the first draft of the software. The last section will be a conclusion.

1.1 Purpose

The purpose of this document is to describe and specify all the requirements for the "Madlibs Editor" software, specified by the customer, Mr. Scott Strentzsch. This document will be presented to the customer for final approval or farther revisioning. It will also be used extensively as a reference for the first version of the software, therefore every feature or function of the software must be explained in explicit detail.

1.2 Scope of software

The "Madlibs Editor" is a stand alone software application that allows users to create their very own custom madlibs. While using the Madlibs Editor software the user must be able to upload any file with a .txt file extension, and be able to customize the text file to create tokens. Users must be able to define groups of words to be defined as the same token. As well as be able to save a madlibs text file. User must be able to choose a "generate list of suggested tokens" option that will search the document for certains words likely to be a suitable token automatically then report these findings to the user, and allow them to accept, edit, or decline the findings.

Users must be able to create groups of token (each token of a group is the same word, and edit or remove these groups.

1.3 Definitions, acronyms, and abbreviations

Term	Definition
User	Anyone who uses the Madlibs Editor
Token	A defined place in the madlibs document, where a word is to be given by the person playing the madlib.
Predefined Token	
User defined token	A token defined by the user. These kinds of tokens include proper nouns such as names and places.
Token Grouping	A grouping of the same word made into a token, and a special type of user defined token. A grouping of tokens is considered one token when counting towards the token question limit. Token groups are commonly made of user defined tokens.
Token Question	How a token is displayed in preview mode. A token question appears beneath a blank where a word is to be inserted in the madlib.
Madlibs	Mad Libs is a phrasal template word game where one player prompts another for a list of words to substitute for blanks in a story, before reading the – often comical or nonsensical – story aloud
Application	In this document, any reference to the "application" are referring to the Madlib Editor.
Menu bar	The area at the top of the application window. Includes such options as "Help", "File", "Edit" and etc.

Toolbar	The area at the top of the application window right below the menu bar. Includes such options as "Create Token" and "save .txt".
Text Editor Panel	The area of the application in which the madlibs text is to be manipulated.
Group Editor Panel	The area of the application where groups are to be created, removed, and customized.
Token Editor Panel	The area of the application where Tokens are to be customized.
.txt	Refers to a text file.
.madlib	Refers to a madlib type file. This is a custom file type that can be read by either the madlib editor or the madlib player.
verb	Conveys an action, occurrence, or state of being.
noun	Denotes a person, place, thing, or idea.
adjective	A describing word. Giving more information about the object specified.
adverb	An adverb is a word that changes or qualifies the meaning of a verb adjective, other adverb, clause, sentence or any other word or phrase, except that it does not include the adjectives and determiners that directly modify nouns.
token question limit	The recommended maximum number of token questions a user might include in a madlib file.

1.3 References

Source 1: Mr. Scott Strentzsch provided all software requirements for the Madlib Editor.

Source 2:

http://www.cse.chalmers.se/~feldt/courses/reqeng/examples/srs_example_2010_group2.pdf
Provided an example template for this SRS document.

1.4 Overview

The remainder of this document first explains the concept of a token, followed by a section on the key features of this software. Finally, a chapter on the user interface layout and what all of the key features of the Madlibs Editor software might look like. The following section is organized by the 5 main areas (or Panels) in the user interface.

2 Tokens

Since "tokens" are the most important object in the madlib editor software, this section will explain what is a "token" and other token related items and features early on in this document.

2.1 Token Overview

This section will go over what tokens, token groups, token questions, and the token question limit are. This section will also go over what each of these is, and how they interact with each other.

2.2 Token Definitions

2.2.1 Token

A token is a defined place in the madlib document, where a word is to be given by the person playing the madlib. Common tokens might be “verb 1” or “noun 3” to show that each instance of a noun is different.

2.2.2 User Defined Token

A user defined token is a token defined by the user. These kinds of tokens include proper nouns such as names and places.

2.2.3 Token Group

A token group is a group of the same word made into a token, and is a special type of user defined token. A group of tokens is considered one token question when counting

towards the token question limit. Token groups are commonly made of user defined tokens.

2.2.4 Token Question

A token question is how a token is displayed in preview mode. A token question appears beneath a blank where a word is to be inserted in the madlib.

2.2.5 Token Question Limit

The token question limit is the recommended maximum number of token questions a user might include in a madlib file. If the user exceeds the limit, a warning should be displayed saying so, and asking if they would like to remove some token questions.

2.3 Token Interactions

2.3.1 Tokens and Token Groups

A token group can be made of regular tokens. A possible application of this would be a group of adjectives that appear multiple times, always describing the same noun. An example of such a use of token groups would be to take the phrase “tall man” in the madlib, and replace the word “tall” in all instances of the phrase with a token.

2.3.2 User Defined Tokens and Token Groups

A token group can be made of user defined tokens. This is a common use of token groups. One possible way of using token groups and user defined tokens is the replace all instances of a name within a madlib with a token, or possibly one word with a generic

definition. Examples of these uses might be changing the name “Lucy” into the token “Name of girl in room” or the word “dog” with type of animal.

2.3.3 Token Groups and Token Questions

A token group should only be considered one token question, as all tokens in the group should be replaced with the same token.

2.3.4 Token Questions and Token Question Limit

All token questions count towards the token question limit, which is only a recommendation to the user about how many token questions should be in a madlib. This is meant to prevent the number of token question from becoming too large. A good token question limit might be in the range of twenty(20) to twenty-five(25) token questions.

3 Features

This section will go over all the madlib editor software features in detail.

3.1 Starting a Madlib from scratch

3.1.1 Creating a new document

There must be a way for the user to create a brand new text document. This new document must have all available text editing options that a word processor such as those found within Microsoft word. This new document must act the same as if it were an imported .txt or .madlib file.

3.1.2 Token handling

The user must be able to define their own tokens and group the tokens as they are writing the document. The grouping panel should be used to keep track of these tokens. The user interface should automatically update as the user is writing into the text document. For example if a token is defined by the user or if the user chooses a predefined token, these changes must be reflected immediately on the interface. The encoding section will go into more detail on how this should look and work.

3.2 Importing a .txt file

The user will be able to load a file with the .txt extension into the madlib creator. When loading the file, the madlib editor will create a copy of the .txt file for editing and leave the original file unchanged. The madlib creator will display the contents of the .txt file in the editor pane, allowing

the user to edit the text as detailed in Section 2.4 (Text Editor Panel). For example, if a user imports a file called “my_story.txt,” then saves their madlib as “my_madlib.madlib,” the “my_story.txt” will remain in the location it was imported from, exactly as it was at the time it was imported.

3.3 File type

All files edited with the madlib editor will be saved with the .madlib extension. This extension will identify the file as bearing encoded token and group data which can be read by either the madlib editor or the madlib reader. When loading a file with the .madlib extension, the application will use this encoding to identify token groups and their corresponding locations in the text, bringing the file to the same state it was in when it was last saved.

3.4 Preview format

The application will allow the user to preview the madlib in its final, playable form, not unlike the Print Preview view mode in a text editor. This preview format will show the madlib with all tokens within the text replaced by blank spaces. The user will be able to switch between the Preview and Editor formats at will.

A separate panel will allow the user to view all token groups associated with the file. When a group is selected in this panel, the application will highlight the corresponding blanks in the file.

3.5 Encoding

3.5.1 User Defined Tokens

There must be a way for the user to communicate with the program to know what declares a token. A set of characters, such as open bracket("[") followed by a closing bracket ("]") must be used around the words the user wants to define as a token. The tokens should also be changed to a different color (this is a token) or have a blank space (_____) to identify to the users that their token has been declared. These tokens will also be updated and indicated the same way within the group panel.

3.5.2 Predefined Tokens

As with user defined tokens, there must be a way to indicate that there is a predefined token. Example "My name is Joe" would be replaced with "My name is (person's name)." These tokens must also be represented in the group panel and shown accordingly.

3.6 Database

The database will be the storage for all common nouns,verbs, etc. and may later implement file storage. Basically this is meant to store all the words the program automatically finds whenever the user asks to auto generate tokens(defined later). Files can be saved and imported into the database so that users can share their files. This database must have an internet connection or maybe have the database come along with the editor itself?

3.7 Auto Generating Tokens

The user needs to be able to generate tokens automatically by reading what is already in the editor. This could be done with maybe a button, perhaps? When the user decides so, the editor

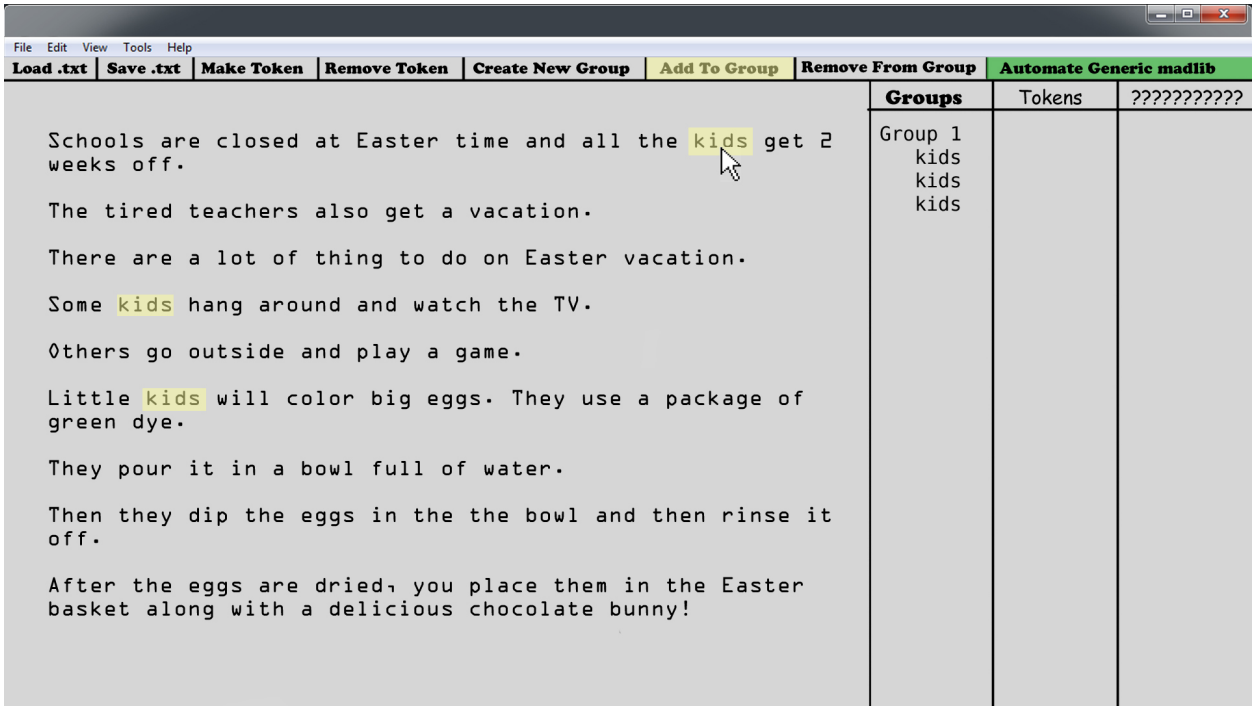
should analyze each word and check into the database if it is currently in the database. It should check which every language the user has pre defined the madlib to be in. This way the program doesn't look endlessly in a spanish dictionary when the word is something like ‘dog.’ Once the word is found in the dictionary/database the program should suggest the word be a possible token. If the user wants it to be a token, the word is replaced and the token is put into the words place. The user will have to ultimately decide whether or not a word automatically found becomes a token.

3.8 Profanity Filter

There must be a way to toggle a profanity filter in order to make this software appropriate for all age ranges. This filter must be linked to a database of profane words. The document will then be scanned for these words and replaced with a sequence of characters to hide the profanity. A sequence of random special characters will be used. This sequence must be the same length for every instance in order to hide the word used. An example would be if the profane word were to have six letters or eight you would always replace with four characters. Example : (ImaginaryProfaneWord = @#$% or MadeUpThisWord = %^&%)

4. Interface Layout

This section describes what the user interface might look like, the user interface is suggested to be divided up into 5 main panels, these are the Menu bar, toolbar, text editor panel, group panel, and the token panel. Below is an example user interface design. Below is a picture for clarification.



4.1 User interfaces

This section describes the 5 basic areas of the user interface and the features within those areas.

4.2 Menu bar

The menu bar features many items that are required in the madlib editing software. Things like

interacting with files, editing the page, and using other tools to effect the page. The menu bar and its items are meant to cover the overall tools of the program, and can be found through other means, ie copying text could be in the edit menu bar and also by hitting ctrl + c.

4.2.1 File Menu

- New - Creates a new empty madlib page. This page, like all others is completly editable.
- Open - Gets an already saved madlib either from the user's computer or maybe later on we can send them to a website's server.
- Save/Save as - Stores the madlib into either the user's computer.
- Exit - Will return the user to the desktop or any other windows the user has turned on.

4.2.2 Edit

- Undo - The last action done is cancelled and whatever it did is resorted back to before it was done.
- Redo - The action that was cancelled is done again.
- Cut - Removes a piece of text and copies it to the clipboard.
- Copy - Copies a piece of text to the clipboard.
- Paste - places the copied text to the madlib.
- Delete - removes the highlighted text.

4.2.3 Tools Menu

- Find tokens - Runs through the madlib and looks for nouns, verbs, etc and

suggests to the user to make them tokens.

- Make tokens - Turns the highlighted word into a token.
- Remove tokens - Turns the highlighted token into plain text.
- Group tokens - All tokens highlighted will be grouped into one group.
- Run madlib - Lets the user play the madlib and enter what they want as if they actually had one.
- More features tbd.

4.2.4 Help Menu

- Documentation - Lets the user read what each button does and watch tutorials.
- About - Lets the user know what version they are using, company info, and legal stuff.

The Menu Bar is meant to hold as much abilities for the user to use as possible. It has many features that can be used in other areas, and that is meant to be so that there can be multiple ways to do what needs to be done. Try to keep the menu bar as concise as possible.

4.3 – Toolbar

The toolbar will provide functionality within a madlib file. It will allow the user such actions as saving or loading a .txt file for a madlib, creating and erasing tokens, and manipulating token groups.

- Load .txt - The load .txt button will open a secondary window to allow the user to

select a .txt file from a location outside the madlib creator.

- Save .txt - The save .txt button will open a secondary window to allow the user to save the current .txt file to a location outside the madlib creator.
- Make Token - The Make Token button will mark the currently selected text as a token.
- Remove Token - The Remove Token button will remove the token designation from the currently-selected token.
- Create New Group - The Create New Group button will create a new token group with a default name. This new group will include the currently-selected token(s), if any.
- Add To Group - The Add To Group button will add the currently selected token(s) to the selected token group.
- Remove From Group - The Remove From Group button will remove the currently selected token(s) from the selected token group.
- Suggest Tokens - The Suggest Tokens button will find occurrences of commonly-used words in the current text and highlight them. The user will be asked if they would like the words to be automatically added to the group list.

The toolbar allows easy access to the program's most commonly used functions, including saving and loading .txt files and basic token manipulation. These functions should allow the user to easily and quickly manipulate tokens, token groups, and even entire .txt files.

4.4 Text Editor Panel

This area will discuss what features are required to be available to edit any of the text contained in the madlib editing software. This includes basic text editing functions, declaring and removing tokens, declaring predefined tokens, allowing the user to define tokens, and right click options that should also be available to the user to make editing easier.

4.4.1 Basic Text Editing

There must be a basic text editing functions like some of the ones in Microsoft Word. (e.g. fonts, font size, bold, italic, underline, and justification formatting.)

This should also have the ability to automatically highlight or change color of suggested words when a document is started or imported. For example if its a predefined token is grouped that group changes to blue, green, etc..

4.4.2 Token Editing

The user should have the ability to highlight any word and declare it to be a token, add it to a group of tokens, remove it as a token entirely, declare it to be a predefined token, or remove it from a token group.

There must be a way for the user to define their own tokens. These user defined tokens should have the same functionalities as predefined tokens.

4.4.3 Right Click Editing Options

- Highlight tokens- this highlights all tokens of the same type and should highlight every occurrence of that same word without highlighting any words that have a similar prefix or suffix.
- Find- a search that allows you to locate tokens of a specific type or group. This

should also allow you to find the next token of that type or group

- Group tokens- a function that will group all currently selected words/tokens.
- Remove tokens- which allows you to remove the selected word/token from the group it is currently in and has an option remove it entirely as a token.
- Format- gives all basic formatting and text editing options.
- Save as- allows you to specify what format to save.
- Auto-generate - finds and suggests predefined tokens and replaces the words
- Spell Check - checks for spelling errors

4.4.4 Text Editor Overview

The text editor panels allow you to make any necessary modifications to the madlib document. These included basic text editing functions such as those found in Microsoft Word, justification formats, and fonts style and size. Token editing should be easy to do within the text document itself and should allow you to add, remove, group tokens, as well as declare predefined and user defined tokens. Right click options should be available to help speed up the process of changing anything contained within the document such as spelling, formatting, and handling of tokens.

4.5 Group Panel

The group panel is an area of the application used for managing groups of tokens.

4.5.1 Overview of the group Panel

A "group" in the context of this document refers to a collection of tokens, of the same name. Groups can be created, modified and deleted at will by the user. The Group panel

is a part of the user interface that houses all the information about the groups a user is currently working with.

4.5.2 Creating groups

Users can create groups by selecting the "create group" button. Once selected, a new group will be shown in the group panel. Every group is created with a default name such as "group 1" although a group name can be changed by the user, by double clicking the name text.

4.5.3 Adding tokens to groups

Once a word is declared as a token, is can be added to a group. To do this the user must simply click that token in the text editor panel, the token will be highlighted once selected, and then the user must select the "add token to group" button.

4.5.4 Removing tokens from groups

Tokens can be removed from groups by selecting the desired token in the group panel, and then selecting the "remove token from group" option.

4.5.5 Deleting groups

Groups can deleted by the user by selecting the group he wishes to delete, and then selecting the "remove group" button.

4.6 Token Panel

The token panel should show all tokens created by the user in a two paned list while in edit view.

4.6.1 Original Word List

On the left side of the token panel might be the complete list of words that have been changed into tokens. An example might be the name Yorick from William Shakespeare's play *Hamlet*.

4.6.2 Token List

On the right side of the token panel might be the complete list of tokens next to the word they have replaced, including token groups listed as a single token. An example token might be “animal” as a token next to the word dog, or “verb 1” as a token next to the word jump.

5. Conclusion

5.5.1 Overview

The "Madlibs Editor" is a stand alone software application that allows users to create their very own custom madlibs. While using the Madlibs Editor software the user must be able to upload any file with a .txt file extension, and be able to customize the text file to create tokens. Users must be able to define groups of words to be defined as the same token. As well as be able to save a madlibs text file. User must be able to choose a "generate list of suggested tokens" option that will search the document for certains words likely to be a suitable token automatically then report these findings to the user, and allow them to accept, edit, or decline the findings. Users must be able to create groups of token (each token of a group is the same word, and edit or remove these groups.

The basic "Madlibs Editor" software requirements are:

- It must be a stand alone application.
- Users can:
    - Create their own custom madlibs.
    - Upload an existing text file to be used as a madlib template.
    - Save a madlib file.
    - Define words in the text to be tokens and restore tokens back to their original state.
    - Create, modify and delete groupings of tokens.
    - Automatically generate a list of "suggested" tokens.