**MadLib Editor**

**Specifications Requirement**

Edwin Gomez, Drew Blanchette, Kris Lara,

Jonathan Villarreal, and Alfred Gutierrez

**1.0 MadLib Creation**

1.1.1 This section will cover the basic functions of how a MadLib should be created. The MadLib should be made up of a short story with keywords that will be replaced by blanks.

**1.1 MadLib Components**

1.1.1 Title: Contains a title of what the MadLib is going to be called.

1.1.2 Author: Contains the author of the MadLib.

1.1.3 Story: A short story with key words replaced by blanks.

1.1.4 Keywords: Such as noun, verbs, places, adjectives.

1.1.5 Tokens: A symbol that the editor recognizes, to replace keywords.

1.1.6 Profanity Filter: Setting that allows or does not allow the use of profanity, for full description see Section 1.2.

**1.2 Profanity Filter**

1.2.1: A sub-setting which modifies text to remove words deemed offensive.

**1.3 Summary**

1.3.1 A MadLib is a tokenized short story, the tokens represent keywords that are replaced to provide a humorous story.

**2.0 Editing Functions**

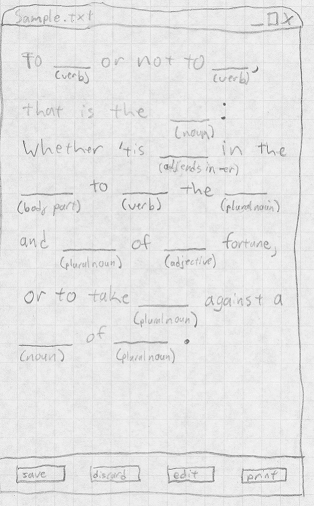
In this section I will go over the editing functionality. This occurs after (and during to some extent) a MadLib file has been created or imported. This functionality will allow the user to designate words to be used as substitution tokens (see section 5), as well as spell check and format what has been written using basic word processing functionality. There will be an optional profanity filter during editing to allow for younger users to be able to use our product as well as a print preview page which show the MadLib file as it would be represented on paper if it were a standard MadLib. I will be going over these in detail throughout this section.

**2.1 General Editing Information**

2.1.1: First of all when will the editing phase come into play? Mostly it will be after the main MadLib story has already been written or imported. The user will be able to import his or her own pre-existing MadLib or story and edit it to his or her liking, or write one from scratch after which the editing comes into play. The main point of the editing is not to fine tune and polish your written work (although that will be available) but to turn a story itself into an actual MadLib. The user will be able to mark special words to be used as tokens for substitution, as well as select all other instances of those words. They will also be able to preview their work as it would look on paper as a regular MadLib, and be able to filter out profanity automatically from a pre-compiled list of restricted words. The default behavior for this functionality will be on however the user will be able to change this in the settings. A lot of more immature adults I think would like this to be off but as a way of being able to market this product to a broader audience it will be on by default.

2.1.2 General Word Processing Ability: The MadLib editor will contain much word processing functionality as comparable to Microsoft Word or Open Office or any other current generation text editor; though it will not go into as much minutia as a lot of current word processors do as it is unnecessary to do so. However simple formatting such as a spell checker, font size, selectable fonts, bold, italic, underline and copy and paste functionality wouldn’t be beyond our scope to add in.

2.1.3 Marking Special Words as Substitution Tokens: The nature of a MadLib is such as to let the user put in his or her own words into a story in places where they make sense grammatically but may be completely out of context when relating to the original story. As such any digital MadLib would need to make it easy for the computer to recognize these special words, and separate them from the rest of the text. This is where the concept of tokens comes in and while editing the user would be able to select certain words and make them into tokens, perhaps with normal meanings such as a noun or an adjective and perhaps with their own custom meaning to be defined as they wish. The editor should also be able to recognize normal parts of speech such as nouns or adjectives and automatically convert a word, when selected, into such a token. Also the program will have an option to where, once selected, all other duplicates of the same word will be selected and the user will be able to tokenize each instance of the word with a single click or button press.

2.1.4 The Print Preview Page: This page will show the user what his or her MadLib would look like if it were to be a normal analogue paper MadLib. The main story will be regularly sized and formatted font for the text that is non-Changeable, and the changeable font (AKA tokens) will be substituted for blanks with a short description in smaller text under the blanks contained within parentheses (see attached sketch). The smaller descriptive text will be fit to the parentheses but in a set range of sizes with a fixed maximum and minimum font size as to still be readable. The print preview page will be its own window that pops up separately from the main screen and will have buttons on the bottom for convenient save, discard, edit, and print functionality. This page pops up whenever the “Preview” button on the main page is pushed, and will disappear whenever the “edit” button is pressed as the user returns back to the main page to further edit their MadLib.

2.1.5 Profanity Filter: This functionality has but one simple purpose, to filter out profane words automatically from a pre-defined list of known obscenities. Its default behavior will be activated but it should be easily de-activated in case the user would like to leave them in. It should be a simple check box inside the settings and serve its purpose well.

**2.2 Summary**

2.2.1: In Summary the editing phase of the MadLib Editor is a big one, it will have regular word processing functionality, be able to easily mark words as substitution tokens, have a print preview page, and a profanity filter. The word processing functionality need not be too extensive however be functional for whatever the needs of the user are, at least comparable to most current word processors. The ability to mark words as tokens to be used for substitution later is also necessary, also the user must be able to select all instances of the same word and change them together as one. There should also be a button on the main page that leads to a print preview page in which all special words are substituted for blanks with a brief description under them as it would be on a regular paper MadLib. A profanity filter that automatically filters out the obscenities should also be added into the settings so that this product may be marketed to younger audiences as well. These functions shall be added into the editing portion of the MadLib Editor to these specifications to enhance the users experience while using our next great product the MadLib Editor.

**3.0 Support**

The support functions of the MadLib Editor are designed to help the user in creating and editing a MadLib. The two main functions are the find functionality and the help functionality. The find functionality will search through the MadLib to find all instances of the word. Once found, the user will be able to tokenize the word found or replace the word. The help functionality will provide information for the user to better understand some of the main functionalities of the MadLib editor. The help functionality will also give some examples to help better understand how some of the functionalities work. It will also contain a search text field that allows the user to search through the help functionality to quickly find what they are looking for. It will also be link to a website that will contain more information and examples if needed.

**3.1 Find Functionality**

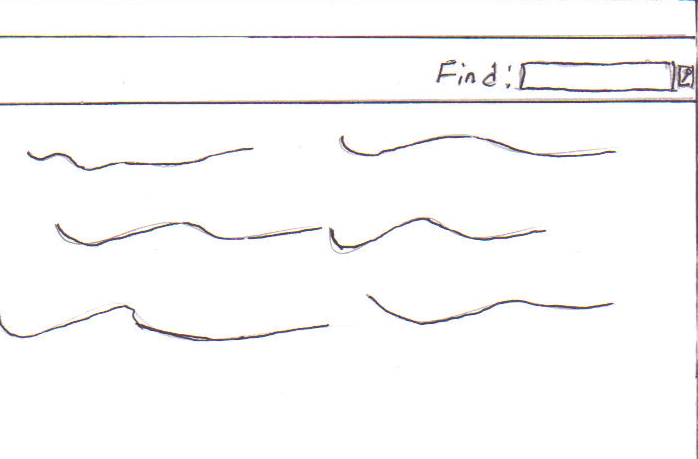
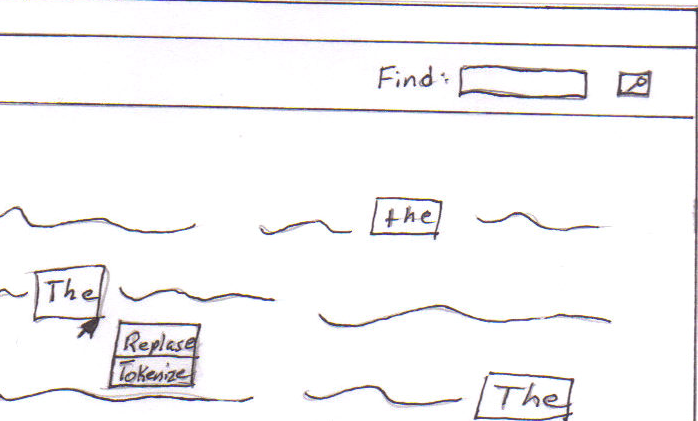
3.1.1: The find functionality would be located on the top left of the toolbar so that it will be easy to access text field. 

Image on where the text field is located

3.1.2: The find functionality would find all instances of the word in the MadLib ,select the word so that the user can delete any word they choose, deselect the word if the user had made an error in choosing a certain word, and allow the user to right-click on any instances of the word. When the user right-clicks the word, a small window will pop up on the side with the options of “Replace” and “Tokenize”.



What it will look like when the user right-clicks on a word.

3.1.3: The “Replace” option would open up a new window where the user will be able to type in the new word that they would replace the selected word with. They will have the option to either replace one or all of the words.

3.1.4: If the user chooses to select to replace one of the words, the word that they right-clicked on will be changed to the new word.

3.1.5: If the user chooses to select the replace all words, all of the instances of the word that are found in the MadLib will be replaced.

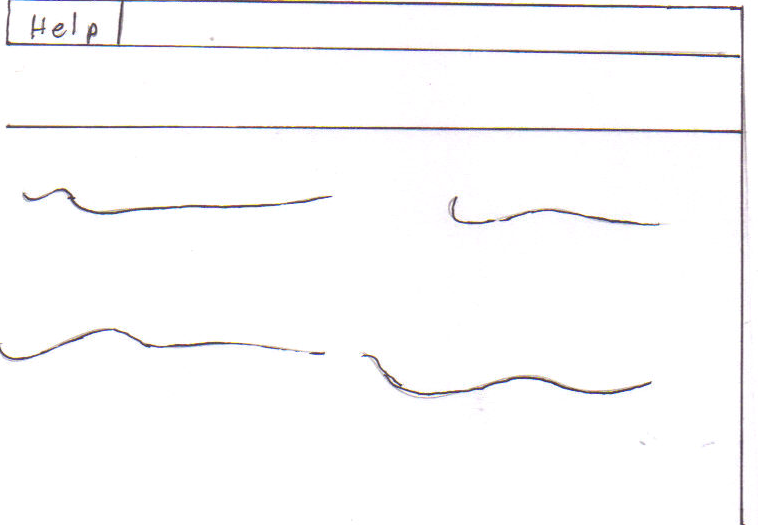
3.1.6: The “Tokenize” option would let the user be able to automatically tokenize the selected word. The word will then be moved to the token library.

3.1.7: Once the word is in the token library, the word will act like a predefined token, allowing the user to substitute the word with another word.

3.1.8: The user will also be able to customize the definition that is attached to the word that is selected.

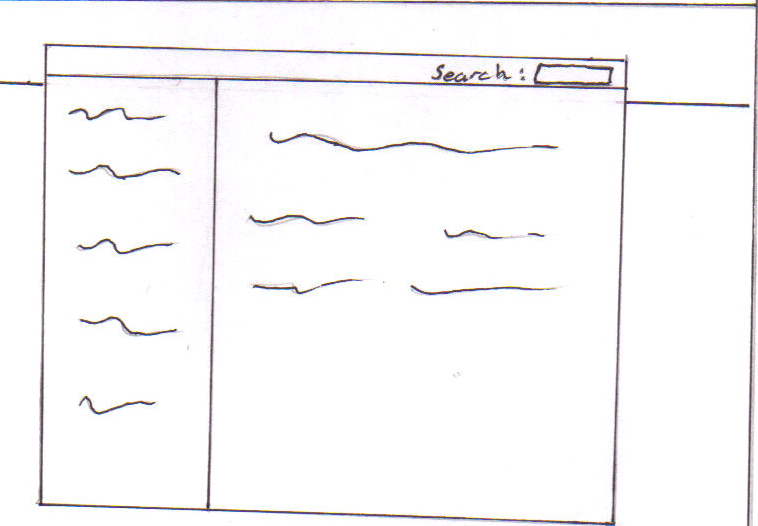
**3.2 Help functionality**

3.2.1: The Help functionality provide basic information for the user about the different functions of the MadLib Editor and will be located in the top of the toolbar so that the user will be able to access it whenever they want.



Where the Help button will be located.

3.2.2: The Help functionality will open up in a new window and have information and instructions on all the main functionality.



What the help function looks like when clicked on.

3.2.4: The function will also have a small search text field so the user will be able to quickly search through all of the instructions that are included.

3.2.5: The help function will also be connected to the main website for additional help if needed.

**3.3 Summary**

3.3.1: The support functions of the MadLib would help the user be able to create, maneuver, and understand the MadLib editor. The find function will allow the user to quickly find, replace, and tokenize a word that they find. The help function will give the user the information and examples to help them understand and use some of the key functions of the MadLib.

**4.0 Handling Tokens**

The editor needs to allow two types of tokens: Pre-Defined Tokens and User-Defined Tokens. Pre-Defined Tokens are held internal to the Editor and cannot be changed. User-Defined Tokens are editable by name and description.

**4.1 Definition**

4.1.1 Pre-Defined Tokens: A storage device within the MadLib editor for words assigned a particular part of speech (i.e. noun, verb, adjective). These tokens represent normal words within sentences that can be used in any passage. Pre-Defined tokens cannot be edited in any way; they can only be placed in the spot the User-Editor requires. Pre-Defined tokens will be capitalized in red text, as such: NOUN.

4.1.2 User-Defined Tokens: A storage device for words/phrases to fit a specific requirement in a sentence (i.e. “a verb ending in –ing”, “a part of the body”). These tokens can be named and defined by the User-Editor. User-Defined tokens will be capitalized in green text with quotes, as such: “BODY”.

**4.2 Token Design Sheet**

4.2.1 - A separate sheet within the MadLib Editor called the Token Design sheet will contain all Pre-Defined Tokens (with descriptions) that can easily be added to the passage, the ability for the User-Editor to name and define his/her User-Defined Tokens, and a count for each different Pre-Defined Token and User-Defined Token that is currently in the passage being edited. The “Token Design” sheet will be broken into two sections, the top containing each Pre-Defined and User-Defined Token with definitions, and the bottom containing the count for each token. In the count section, the User-Editor can double-click a specific type of token to open a window displaying each occurrence of the token. The line # and word # will associate the token, for example: a NOUN token is being used on line 4 and is word 7; it will be displayed as NOUN (L4, W7). If the User-Editor should right-click on an occurrence, the option to DELETE, REASSIGN (change token type), and GO TO should appear.

**4.3 Insert and Edit**

4.3.1 Design Tab: The User-Editor will have access to different methods of inserting and editing. One method is to access the Token Manipulator group located in the DESIGN tab of the Cross Bar Tools. This group will contain editing tools for token manipulation. These tools will allow the User-Editor to insert any type of token at the current location (if the User-Editor is attempting to insert a new User-Defined Token, he/she must name and define the token at that moment); edit a pre-existing User-Defined Token; reassign the highlighted token(s) to a different token type; perform standard actions such as delete, find all, and open the Token Design sheet.

4.3.2 Token Properties: Another method to insert and edit tokens is to right-click in the passage and access the Token submenu. This submenu will have all of the same options as the Token Manipulator group.

**4.4 Summary**

4.4.1 Reflection: The MadLib editor will require two basic ways of handling the two different types of tokes (Pre-Defined and User-Defined). There will be a separate sheet specifically designated for token editing and token recording.

**5.0 Save and Load**

The following application will support a user friendly input and output functionality. The MadLib editor will allow the user to save a MadLib locally on their computer and also on the MadLib website for worldwide usage. This will allow the user to keep a number of MadLib files for future use. The MadLib editor will allow the user to load previously made MadLib from their computer or storage device and the MadLib website into the MadLib editor for editing. Usability is a key and an important aspect to the MadLib editor.

**5.1 Save**

5.1.1: The MadLib editor save functionality is easy for the user to use. This feature allows the user to save their files into different file types so they can have an easy way of accessing it on computers that don’t have this application. Some file types are listed but not limited to text document, Microsoft word document, MadLib file and WordPad document.

**5.2 Load**

5.2.1: The MadLib editors load functionality is another easy to use feature. The load feature allows the user to load a MadLib file, or one of the supportive file types that was stated in section 5.1 of this document. This feature will allow the user access to the MadLib to edit it as they please.

**5.3 Download**

5.3.1: The MadLib editor download functionality will give the user the options to download a MadLib from a wide collection from our MadLib website. In the options menu there is a tab that will take you to the website directly were the user can download a MadLib. This will allow the user to see others people MadLib from around the world. The user is required to have access to the internet in order to download a MadLib from our website.

**5.4 Upload**

5.4.1: The MadLib editor upload feature will give the user the option to upload a MadLib to our MadLib website were other users from around the world can see and edit as they please. In the options menu there is a tab that will take you to the window were the user will confirm there the decision to upload there MadLib into out collection.

**5.5 Summary**

5.5.1: The following application will support a user friendly input and output functionality. Reusability is a key important aspect in mind for the user. The saving, loading, downloading, and uploading MadLib feature will allow the user to save and share their MadLib with other people around the world.