**MadLib Editor Requirements**

**Version 2.0**

Edwin Gomez, Drew Blanchette, Kris Lara,

Jonathan Villarreal, and Alfred Gutierrez

**Table of Contents**

1. Introduction
   1. MadLib Editor Overview
   2. MadLib Definitions
2. Editing Functions
   1. General Editing Information
      1. General Word Processing Ability
      2. Substitution Tools
      3. Design View Mode
      4. Profanity Filter
      5. Find Function
   2. Summary
      1. Functionality Discussed
3. Handling Tokens
   1. Token Design Pane
      1. Token Definitions
      2. Token Properties
      3. Special Functionalities
   2. Insert and Edit
      1. Design Tab
         1. Design Tab Tools
      2. Design Submenu
   3. Summary
4. Support
   1. Find Functionality
      1. Print Preview
      2. Token Description
      3. Pop-Up Window
      4. Help Tab
      5. Layout Tutorial
      6. Quick Search
      7. Contact Information
   2. Summary
5. Save and Load
   1. Save
   2. Load
   3. Download
   4. Upload
      1. Direct Upload
      2. Windows Explorer Upload
      3. Internet Requirement
   5. Summary
6. Conclusion

**1.0 Introduction**

This software design document describes the specifications requirements of the MadLib editor and its functionalities. The MadLib editor allows for the editing and creation of MadLibs. A MadLib is a phrasal template word game, played by two or more players; it is made up of a short story with a list of keywords that are substituted by blanks. The MadLib editing functionality will allow designated words to be used as substitution tokens as well as spell checking and word processing format, along with an optional profanity filter, allowing for a broader audience of users

**1.1 MadLib Editor overview**

The document overview the requirements needed to edit an already existing MadLib. Since the MadLib is already a short story basic word processing operations are needed to allow the story to be edited. Word processing consists of editing, printing, formatting written material, searching, font application, spell checking, automatic text correction, saving and loading text documents. As a MadLib, blanks are substituted by a list of keywords either pre-defined or user defined. The document is organized to describe the MadLib editor’s requirements, started by editing a MadLib, print preview page, support functionalities, Handling tokens, Token design sheet, Save, Load, Download, and Upload.

**1.2 MadLib Definitions**

Dynamic Text

*All caps bold text used to show tokens in the text editing mode.*

Static Text

*Regular text used without being tokenized as part of the main story.*

Descriptive Text

*Verbiage in the design view mode to show the question attached to the token it belongs to.*

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.*

*List of Pre-Defined Tokens:*

* *NOUN*
* *NOUNS*
* *ADJECTIVE*
* *VERB*
* *PASTVERB*
* *ADVERB*

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”.*

MadLib File

*A MadLib file is file type that is readable to a person and doesn’t have a special setup like an xml file. It reads the character on the file one letter at a time until it reaches the end of the file.*

XML File

*A XML file or Extensible Markup Language is a markup language that displays a set of rules for encoding the document in a format that a person and computer can read.*

**2.0 Editing Functions**

This section goes over the text and token editing functionality. MadLib editing occurs after (and during to some extent) a MadLib file has been created or imported. This functionality must 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 Design View mode which show the MadLib file as it would be represented on paper if it were a standard MadLib. The following section will cover these in detail.

**2.1 General Editing Information**

Editing a MadLib comes into play mostly after the main MadLib story has already been written or imported.

The user must be able to import his or her own pre-existing MadLib or story and edit it to his or her liking.

The user must be able to write a MadLib 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 must be able to mark specific words to be used as substitution tokens, as well as select all other instances of those words by themselves.

They must 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, to be compiled list of restricted words. The default behavior for this functionality will be on however the user must be able to change this in the settings.

2.1.1 General Word Processing Ability

The MadLib editor will contain basic word processing functionality, comparable to Microsoft Word or Open Office or any other current generation text editor, though it will not support advanced editor functions which are unnecessary or unrelated to the basic purpose of creating a MadLib. Simple edit functions 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.

2.1.2 Substitution Tokens

The whole point of a MadLib is to let the user put their own words into a story, based on what part of speech is needed, without giving the user access to the context those words will be in after the story is reassembled. 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 tokens come in. While editing, the user must be able to select specific words they would like to make into tokens, of two types: normal parts of speech such as nouns or adjectives; and, user defined tokens with custom meanings to be specified by the user.

The editor should also be able to recognize normal parts of speech such as nouns or adjectives using a parts of speech tagger such as the Brill Tagger, or whatever we can get that’s functional and works well enough to tag most of the words needed while leaving out unneeded parts of speech such as prepositions or conjunctions. Also it must, at the user’s discretion, convert a word or group of words, when selected, into such a token.

The program must have an option to where, once a word is selected, other instances of the same word will be selected, and the user will be able to tokenize the group or subset thereof with a single click or button press.

2.2.3 The Design View Mode

This mode will show the user what his or her MadLib would look like if it were printed digital paper MadLib.

The main story must use a twelve point font for the static text, and the dynamic text or tokens, will be substituted for blanks with a short description in smaller text under the blanks contained within parentheses.

The smaller descriptive text will be fit to the parentheses but in a set range of sizes with a maximum of ten point, and a minimum of six point font size as to still be readable. Descriptive text will also have a character limit on it so that the user can’t put too much text in that it wouldn’t fit under the blanks.

The design view pane must be a separate viewing mode for the edited portion, which the user can switch back and forth from with the click of a button.

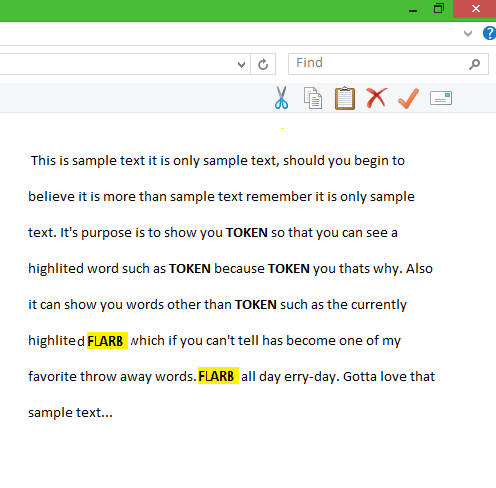
Furthermore, the user must also be able to still edit the MadLib within this pane as they would be able to normally. The token screen will also be present at this time so the user can see a list of their current tokens and be able to edit them either in groups or individually, and be able to drag and drop words directly to the token design screen to tokenize words. (See section 3)

2.2.4 Profanity Filter

This function has one simple purpose, to filter out profane words automatically from a pre-defined list of known obscenities, such as F\*\*\*, S\*\*\*\*, B\*\*\*\*, A\*\* and so on. Its default behavior will be enabled 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 setting. Also words must be easily addible to the list at the user’s discretion.

2.2.5 The Find Function

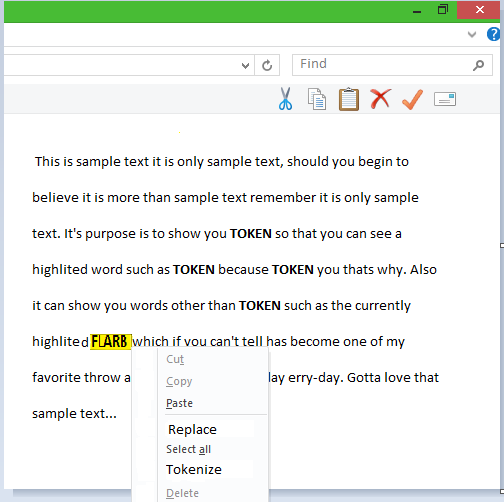
The find functionality must be located on the top left of the toolbar so that it will be easy to access text field.



The find functionality must find all instances of the inputted word in the MadLibs; allow the user to can deselect any words they choose, and 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”. Replace simply changes one word out for another inputted word as per normal word processing functionality. Tokenize sends a copy of the word to the token design screen and marks that word as a token on the main screen.

The “Replace” option must 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 must have the option to either replace one or all of the words.

If the user chooses to replace one of the words, the word that they right-clicked on must be changed to the new word. They can either be changed to match existing tokens or as simple editing functionality.



If the user chooses the replace all words, all of the instances of the word that are found in the MadLib must be replaced as a token if necessary or as static text if not.

The “Tokenize” option would let the user be able to automatically tokenize the selected word. The word will then be moved to the token screen as a user-defined token. Also the word will be changed to dynamic text, the blanks in design view mode or just as a regular token in text mode.

Once the word is in the token screen, the word will become a user-defined token, allowing the user to define the appropriate question to be asked for substitution.

**2.3 Summary**

The editing phase of the MadLib Editor is complex, it must have regular word processing functionality, be able to easily mark words as substitution tokens, have a design view mode, a find function, and a profanity filter.

The word processing functionality need not be too extensive but, be functional within the discussed scope.

The ability to mark words as tokens to be used for substitution later is the primary editing function. The user must be able to select all instances of the same word, and group them together as one. There should also be a design view mode in which all tokens are substituted for blanks with a brief description under them as it would be on a regular paper MadLib, also a profanity filter that automatically filters out the obscenities as per previous discussion.

**Functionality Discussed In This Section**

* General word processing functionality
  + Spell check
  + Insert/Delete/Copy/Paste etc...
* Marking specific words as substitution tokens
  + Ability to select or deselect all instances of a certain word
* Print preview pane
  + Allows the user see what his custom MadLib would look like on paper
  + Allows the user switch between two different view modes printed and normal
* Profanity filter
  + Filters out obscenities from a pre compiled list
  + Default on
* Find Functionality
  + Located in the top left
  + Finds and highlights all instances of a certain word
  + Able to replace or tokenize words
  + Also able to replace or tokenize all instances of words

**3.0 Handling Tokens**

The editor must 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. Many editing tools will be necessary to handle tokens correctly.

Tokens must be editable within the editor pane as well as within the Token Design pane (see section 3.1). Tools to edit within the editor pane must be accessible within a tab on the Crossbar Tools as well as from within the text by accessing a submenu through right-clicking a token.

Special Functionalities between the Token Design Pane and Editor Pane will help accommodate the User-Editor’s time and efficiency. Such functionalities as Drag-and-Drop Insertion, Highlight All, Reset

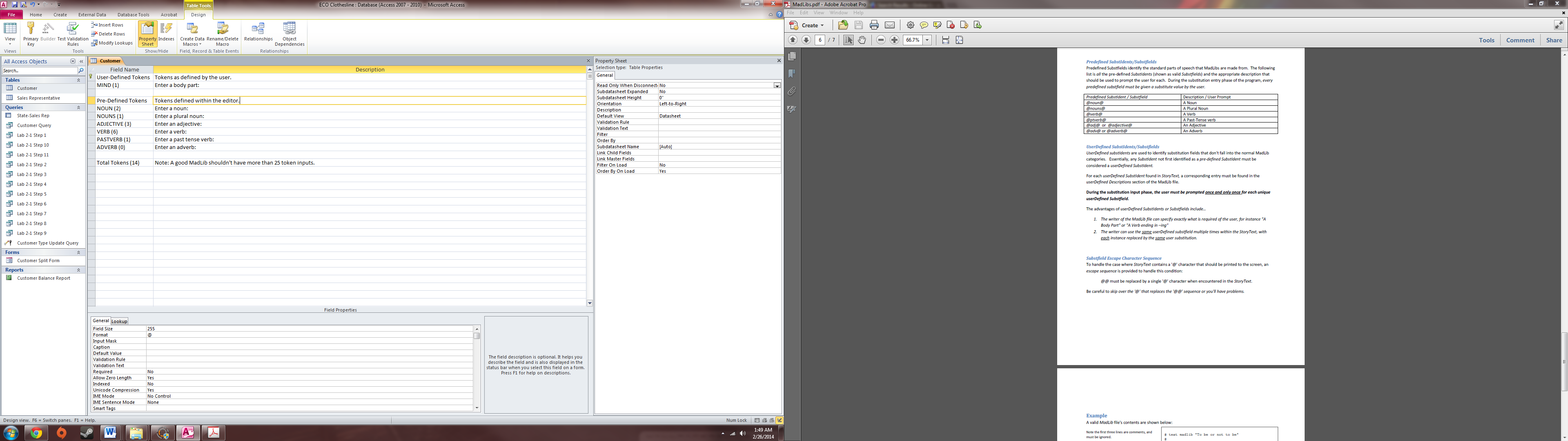
**3.1 Token Design Pane**

A separate pane within the MadLib Editor called the Token Design pane must contain all Pre-Defined Tokens and User-Defined Tokens that are within the passage. The pane must allow the user to create a User-Defined Token and see all existing Pre-Defined Tokens.

The “Token Design” pane must be divided into two sections; the top containing each Pre-Defined and User-Defined Token with definitions and the bottom containing extra properties for each token.

3.1.1 Token Design Definitions

In the definitions section, a count of each different Pre-Defined Token and User-Defined Token already located in the file must be available. All questions for each token must be displayed. Using general editing features, this section must allow the user to easily insert new User-Defined Tokens.



3.1.2 Token Design Properties

In the properties section, the User-Editor must be able to double-click a specific type of token to highlight each occurrence of the token within the editor pane. The option to DELETE, REASSIGN (change token type), and GO TO should appear in a pop-up submenu.

If the user only single-clicks a token, a list of each token occurrence within the file should appear. There must be a line # and word # that is associated with each token, for example: a NOUN token is being used on line 4 and is word 7; it will be displayed as NOUN (L4, W7). The list would continue on until all occurrences of that token type are listed.

3.1.3 Special Functionalities

There must be unique functionality between the editor pane and token design pane. Functionalities such as: Drag-and-Drop Assignment, Multi-Token Assignment, and Preview-Original.

**3.2 Insert and Edit**

The User-Editor will have access to different methods of inserting and editing tokens. One method is to access the Token DESIGN tab of the Cross Bar Tools. The other method is the Token DESIGN submenu.

3.2.1 Design Tab

The DESIGN tab must contain editing tools for token manipulation. These tools must allow the User-Editor to insert any type of token at the current location.

*3.2.1.1 Design Tab Tools*

***NEW USER-DEFINED****: Inserts a new User-Defined Token at text-pointer location. The user then is prompted to define the token with a Name and Question.*

***GO TO “Token Design”****: Redirects the user to the Token Design pane to edit a pre-existing User-Defined Token.*

***REASSIGN****: Allow the change of a highlighted token(s) to a different token type.*

***GENERAL****: Standard actions such as delete, find all, and text formatting.*

3.2.2 Design Submenu

Another method to insert and edit tokens is to right-click a set of highlighted token(s) in the passage and access the Token submenu. This submenu will have all of the same options as the DESIGN tab and will allow the user ease of access.

**3.4 Summary**

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.

**4.0 Support**

The support functions of the MadLib Editor are designed to help the user in creating and editing a MadLib. The support functions must also give the user information in the functions of the MadLib editor to help him or her answer any questions or problems that they might have. The two main functions are the print preview functionality and the help functionality. The print preview functionality will help the user see what their MadLib would look like as a printed digital paper MadLib. The print preview will have formatted font for the title, tokens, and regular text in the MadLib. There will also be a small descriptive text under the tokens that will have their own format so that the text can still be readable. This function will also have its own pop up window when open and will contain a print, save, and load functionality on the bottom. The window will close once the “Close view” button is pressed, which will take the user to the main editor page.

The help functionality will provide information for the user to better understand some of the main functionalities of the MadLib editor. It must give information about the MadLib editor so that the user will be able to better understand and navigate through the different panes and be able to find answers to any problem that they encounter while using the program. It will include contact information if the user cannot find any information regarding their problem so that they will be able to talk to a person who has a better understanding of the program for help.

**4.1 Find Functionality**

4.1.1 The Print Preview Page

The page will show the user what his or her MadLib would look like if it were a printed digital paper MadLib.

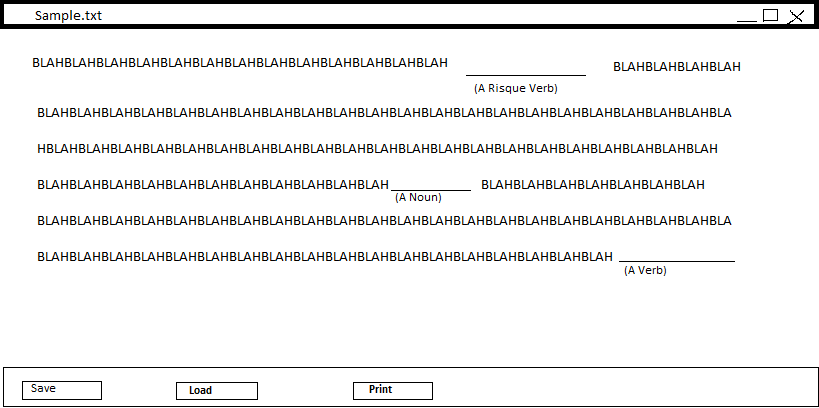
The main story will use an exact size and formatted font for the text that cannot be changed and a different fort for tokens, which will be substituted for blanks with a short description in smaller text under the blanks contained within parentheses that give a brief description as to what new word will be placed there (see attached sketch). The font will be Times New Roman style for the whole MadLib.

4.1.2 Token Description

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. This description will tell the user what word to place in the blank space.

4.1.3 Pop up Window

The print preview page will be a pop-up window and will have buttons on the bottom for convenient print functionality. This page pops up whenever the “Preview” button on the main page is pushed and will disappear whenever the “Close View” button is pressed. There will also be three buttons on the bottom of the preview window: print, save, and load. The print button will automatically print the MadLib out. The save and load button will save the file in whatever format you want and load any MadLib file that you would have. The user returns back to the main editor page to further edit their MadLibs when the preview window is closed.



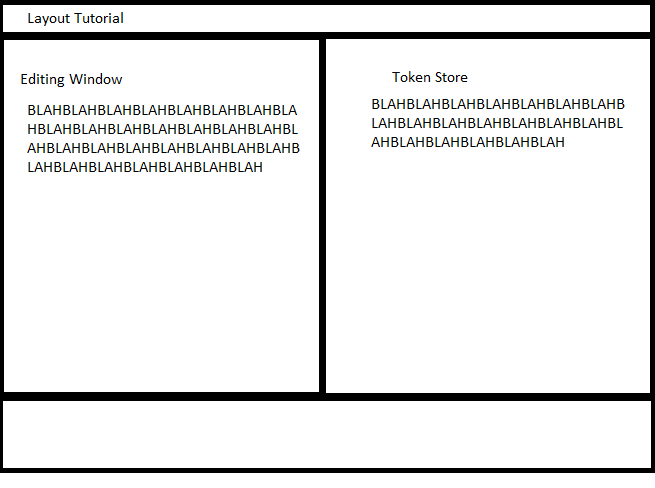
A mock up version of what the Print Preview will look like.

4.1.4 Help Tab

The Help Tab would 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. Once click, it will open up another window that will contain a layout tutorial, a quick search with a text field, and contact information for the user to use.

4.1.5 Layout Tutorial

There must also be a tutorial section that will show the basic layout of the MadLib editor and will give a small description on where each part is and how to use each of the sections. All functionalities of each panel will also be displayed here as well. This will allow the user to see where each area is visually as well as read about them.



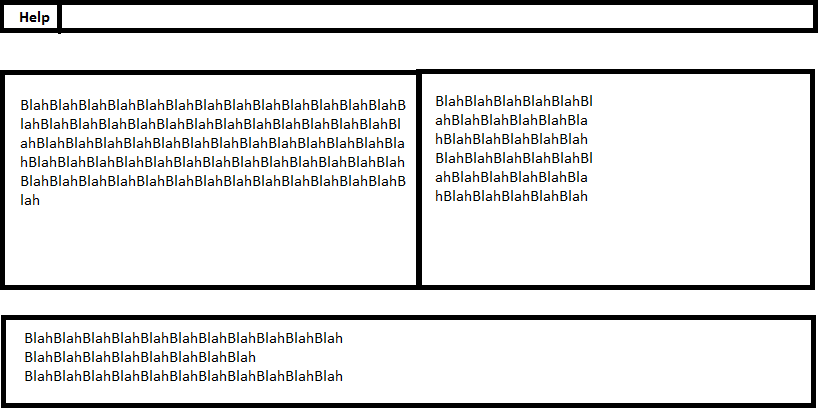
What the layout tutorial would look like.

4.1.6 Quick Search

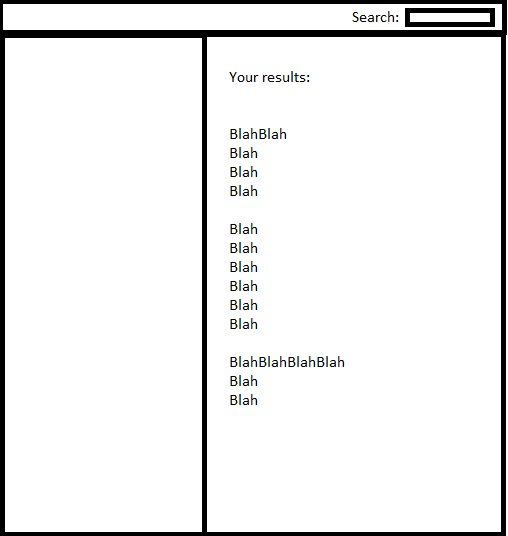
The Quick Search option will also have a text field that must allow the user to be able to type in any word or phrase in a text field and search for any instance of the word in all of the files that are contained in the function. Once all instances have appeared, the user will be able to select one of the instances that appeared and will be taken to that file’s location.

4.1.7 Contact Information

The feature must also have contact information so that the user will be able to get a real person to help them with their problem. There must be a contact number and email address for the user to be able to try and reach someone to explain their issue to. The contact information will also give the user a hyperlink to the main website to get any other information that they will need.



Where the Help button will be located.



What the help function looks like when clicked on.

**4.2 Summary**

The support functions of the MadLib would help the user be able to create, maneuver, and understand the MadLib editor. The print preview function of MadLib must show the user what his or her MadLib will look like as a printed digital paper MadLib. The function will also have formatted font for the title, main story, and descriptions for the tokens so that everything will still be readable. The print preview will also open up in a pop up window that will have a print, save, and load button on the bottom of the page and will disappear when the “Close view” button is pressed.

The help function of MadLibs will provide information to the user about the different functionalities of the MadLibs Editor. This function is key into better helping the user understand how to navigate and use the MadLib Editor in an efficient way. The help tab will give a list of shortcuts that the user will be able to use to select different functions with ease. There must also be a layout tutorial that gives a detailed explanation of each different panel as well as how to use it and where it is located. There will also be a quick select option that will provide a text field that will allow the user to quickly search through all of the help files to find an answer to the problem they are having. There will also be contact information for the user so that they may call or email someone to get help on a certain problem that they could not figure out themselves.

**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 file editor must allow the user to load previously made MadLib from their computer or storage device, and the MadLib website into the MadLib editor for editing.

**5.1 Save**

The MadLib editor save functionality is another easy to use feature. 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.

The MadLib Editor File types are presented below. Please refer to the section 5, 1 for the definition of the file types

* XML
* MadLib

The MadLib editor will save the file in the setup that it was made in.

**5.2 Load**

The MadLib editors load functionality is another easy to use feature. The load feature allows the user to load a MadLib file, and a XML file type that was stated in section 5.1 and 5.2 of this document. This feature will allow the user access to the MadLib to edit it as they please.

The MadLib editor must load the file according to how the user saved it.

**5.3 Download**

The MadLib editor download functionality will give the user the options to download a MadLib from a wide collection from our MadLib website.

The download option is different from the uploading options because it takes you to a page in the website that is for downloading MadLibs only. 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.

An internet connection is required for the download function of the MadLib editor. If internet connection is not detected when the user is trying to download a MadLib file a blank internet screen is shown instead of the MadLib website,

**5.4 Upload**

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. The MadLib editor has two way of uploading a MadLib file.

5.4.1 Direct Upload

A MadLib file can be uploaded directly from the current MadLib in the editor. In the options menu there is a tab that will take you to the window were the user will confirm their decision to upload the current MadLib into our collection

5.4.2 Windows Explorer Upload

You can also be uploaded a file by selecting which MadLib you wish to upload from the windows explorer. In which you will move though the file system on your computer to the designated location of the MadLib file you wish to upload.

5.5.3 Internet Requirement

An internet connection is required for the download and upload functions of the MadLib editor. If an internet connections in not detected the upload options will require the user to establish a connection.

**5.5 Summary**

The following application will support a user friendly input and output functionality. The saving, loading, downloading, and uploading MadLib features will allow the user to save they MadLib files in there computer and also share their MadLib with other people around the world. The MadLib editor will allow the user to load a MadLib file from a storage device or from the MadLib web site. Some of the features of the MadLib editor will require outsource help in order for the features to work.

**6.0 Conclusion**

The MadLib will be comprise of four functionalities: editing, handling tokens, support, and save and load. Editing functionality will allow for the user to designate a word to be a certain token as well as spell check and word processing format. Handling the tokens will divide the tokens into two categories, Pre-Defined and User-Defined token, which are edited in the edited window. Support will provide basic information of the MadLib Editor so that the user will be able to create and edit a MadLib as efficient as possible. Saving and loading will give the user a friendly input and output functionality by having the user be able to save a MadLib for later use and load a previously saved MadLib for editing.