Term Project Report

### Project: Reverse Madlibs

### Date: October 31, 2011

# Formal Description

## Overview

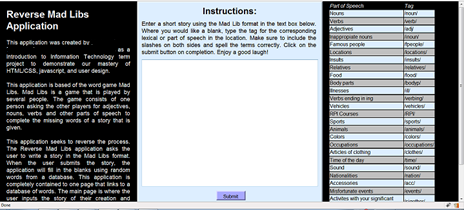
For the term project, we propose to create a reverse Madlibs application. Mad Libs is a word game that is played by several people. The game consists of one person asking another for adjectives, nouns, verbs and other parts of speeches to complete the blanks of a story that is already given. Our application is the complete opposite of this. The user of the application will write a story of their own and indicate the blanks as well as the part of speech that should go there. When the user submits the story, the application will fill in the blanks using random words from a database. The application consists of one page that links to a database. The main page is where the user inputs the story of their creation and receives the output. The database consists of multiple lists separated by type.

## Who it serves:

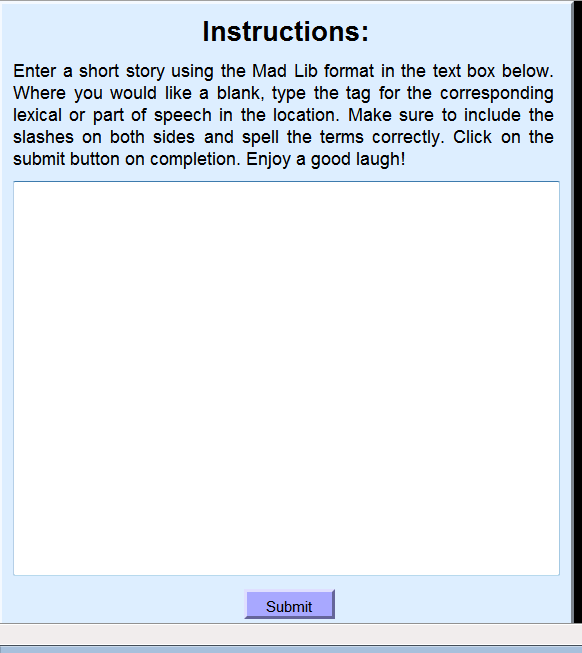
This application can serve a variety of different groups, but it is mostly for people who spend their time online. It is for people that like to play online games and complete puzzles. The main audience is people who are bored and looking for an interesting way to fulfill their time.

# Information Architecture

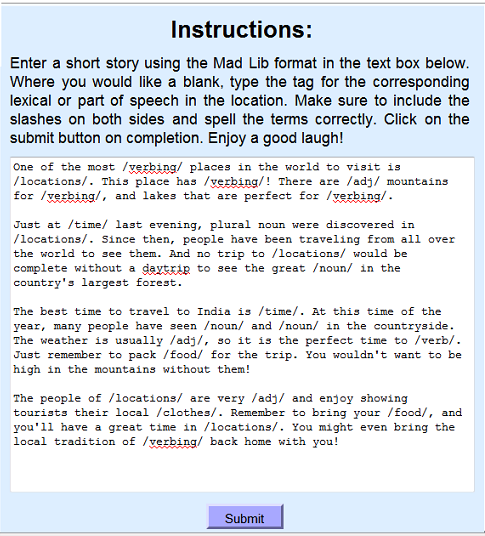
## Storyboard

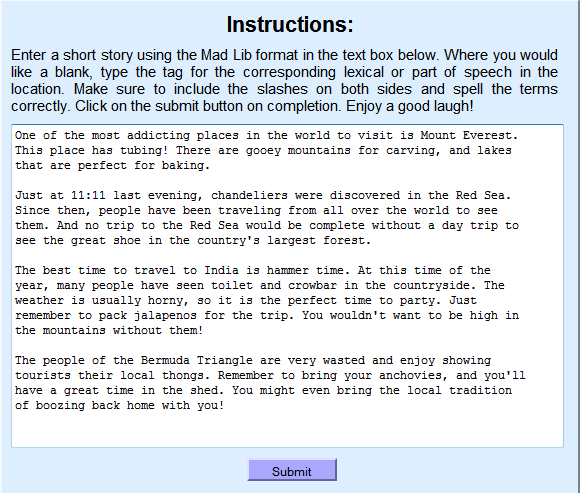


1. When the user accesses our application, this is the page they will see when they retrieve it in the browser.



The user will select the text box (which will empty itself) where they will enter a story of their liking.

The user will enter their story, inserting the correct label that fits the category they would like to use, such as nouns, verbs, adjectives, etc.

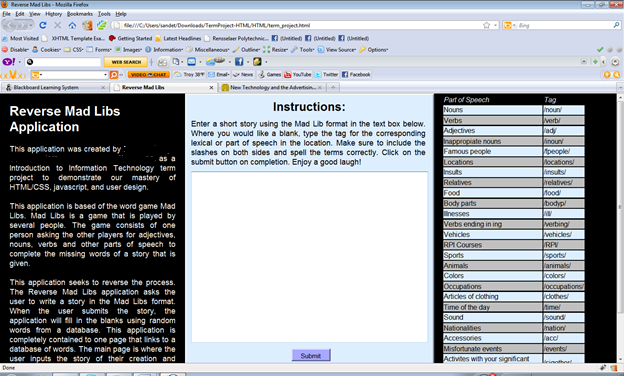


Lastly, the user will hit the submit button and the application will insert the random word from a database into the blanks that were inserted by the user.

## Interface

Instructions visible to the user, which provide detail about how the application should be used.

Explains the purpose of the application and what it is all about



Takes the users input and runs it through the application pulling words from a database to finalize the users input.

Information provided for the user to know exactly how to word the tags they inserted into their stories

A given location provided for the user to insert their input

## Flow of data

# Use cases

## Brief Description

This use case describes how a user interacts with the Reverse Madlibs System to create a story and use the application to entertain them. After the user has inputted a story, the user clicks on the submit button. The system will respond with a confirmation message asking if the user is ready and wants to submit. If all the input guidelines are met then the systems returns a reverse mad lib page.

**User:** A member of the public (MP)  
  
**Use case:** The MP is creating a story to feed into the application.

**Preconditions:** The MP has to have a working internet connection and must open the RMS page.

**Triggers:** The user indicates that they want to submit the story they have keyed in by clicking on the submit button.

## Scenario A

1. Page loads
2. User reads the instructions above the input box.
3. User inputs a short story, inputting tags in place of “blanks”.  The format for the tags is provided in the right hand column.
4. User clicks on the submit button.
5. The submit button calls a JavaScript function.
6. JavaScript searches through the input for valid tags.
7. When found the tags are replaced with words from a JSON database.
8. The page outputs the result to the user in the text box replacing the user’s input.
9. Use case ends successfully

## Exception

3)  User has incorrectly inputted the tags. Application will alert of open tags and unrecognized tags.  User must fix tags and resubmit.

9) Use case ends unsuccessfully.

## Alternative

3) User creates a short story in the format specified using the parts of speech by typing in the tag in the specified locations desired.

Post-conditions  
**1) Successful Completion:** The user can view the output of the application  
**2) Failure Condition:** The user has to restructure input to meet the specified requirements of the application.

## Special Requirements

1) The Application shall work only if valid tags are used.

Ten Usability Heuristics

### Student 1

1. Visibility of system status-1
2. Match between system and the real world-5
3. User control and freedom-5
4. Consistency and standards-4
5. Error prevention-4
6. Recognition rather than recall-3
7. Flexibility and efficiency of use-3
8. Aesthetic and minimalist design-5
9. Help users recognize, diagnose, and recover from errors-4
10. Help and documentation-4

### Student 2

1. Visibility of system status- 1
2. Match between system and the real world- 5
3. User control and freedom-1
4. Consistency and standards-4
5. Error prevention-5
6. Recognition rather than recall-3
7. Flexibility and efficiency of use-1
8. Aesthetic and minimalist design-3
9. Help users recognize, diagnose, and recover from errors-4
10. Help and documentation-1

### Student 3

1. Visibility of system status- 1
2. Match between system and the real world- 5
3. User control and freedom-4
4. Consistency and standards-4
5. Error prevention-5
6. Recognition rather than recall-5
7. Flexibility and efficiency of use-1
8. Aesthetic and minimalist design-5
9. Help users recognize, diagnose, and recover from errors-4
10. Help and documentation-3

### Average

1. Visibility of system status- 1
2. Match between system and the real world- 5
3. User control and freedom-3
4. Consistency and standards-5
5. Error prevention-5
6. Recognition rather than recall-4
7. Flexibility and efficiency of use-2
8. Aesthetic and minimalist design-4
9. Help users recognize, diagnose, and recover from errors-4
10. Help and documentation-3

## Analysis

### Visibility of system status

The system does not keep users informed about what is going on throughout. Although, the status of the program does not change. On submit, the user is made aware if there were any incomplete tags.   
Match between system and the real world  
The system uses simple language, with words, phrases and concepts familiar to the user.  
User control and freedom  
The application supports undo and redo natively through the browser. Although the user must input the tags as opposed to having the application insert them.  
Consistency and standards  
Platform conventions are followed and consistency is maintained. All tags begin and end the same way.  
Error prevention  
Errors are not prevented but are identified on submit and the user is notified where the error is.  
Recognition rather than recall  
The instructions and tags are always visible to the user. This allows the user to quickly reference either.  
Flexibility and efficiency of use  
The tags can only be inputted by the user typing them. If the application was more flexible, it would have buttons to add them as well.  
Aesthetic and minimalist design  
CARP standards are complied with.  All objects are distinguishable from the background. Similar themes are grouped together and well aligned in an organized fashion. The design has only one button simplifying the process of submission.  
Help users recognize, diagnose, and recover from errors  
User is notified through an error message when tags are incorrect. Once the tags are corrected, Application runs.  
Help and documentation  
Instructions and documentation are included on the screen. No real time feedback and help is provided though.