SE 3XA3: Test Plan
Rather

Team # 3, Team 3
Erin Varey Vareye
Joel Straatman Straatjc
Nik Novak Novakn

October 31, 2016

Contents

List of Tables

Table 1: **Revision History**

Date	**Version**	**Notes**
October 21	1.0	Started to create
Date 2	1.1	Notes

1 General Information

1.1 Purpose

The purpose of this project is to provide sensitive people with a method of blocking words that may be offensive or may elicit a negative response.

1.2 Scope

Rather uses Google Chrome's extension feature, so theoretically the scope of the project is any page that Google Chrome can access.

1.3 Acronyms, Abbreviations, and Symbols

Table 2: **Table of Abbreviations**

Abbreviation	**Definition**
GUI	Graphical User Interface
Abbreviation2	Definition2

1.4 Overview of Document

2 Plan

The code will be tested both dynamically and statically. Every time a section is updated the auto testing script will be run to ensure it works. Once the final prototype is close to complete a focus group will be used to test dynamic components such as appearance. This will consist of them ranking components that will be listed on a survey.

2.1 Software Description

2.2 Test Team

The Software will best tested by the developers along with a small focus group. The small focus group will consist of 3 volunteers who will test the code to prevent developer bias.

Table 3: **Table of Definitions**

Term	**Definition**
output(x)	A generalized form of representing the changes made by an action x. It is more of an abstract concept rather than a programmatic function. If action x creates no change, or causes an error state, this function is null returning.
TestCases	The set of all possible test cases for the program.
test_existing(t)	A conceptual function that performs a test t on the existing implementation.
test_new(t)	A conceptual function that performs a test t on the existing implementation.
Post	A section of a site encapsulating and packaging content from a social media site. Varies per social media site.
Service	A social media site for which the application has been developed.

2.3 Automated Testing Approach

Due to the nature of a web application the automated testing approach is going to be slightly complicated. It will consist of two components. A Java application for automation, and a Javascript script for testing the functions as required. The Java portion of the code will consist of automation that opens a Chrome browser and opens a webpage that will be used for testing. The program will then control the mouse to click the application and add a filter and then refresh the page. It will then execute the Javascript portion. This will consist of the unit testing scripts that will be written in Q-unit.

2.4 Testing Tools

The only tool that is required for testing the application is a computer and a internet connection.

2.5 Testing Schedule

See Gantt Chart at the following url ...

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 Search Functionality

Can the program search and replace text?

1. test-id1
   Type: Functional Static

   Initial State: Fully Loaded Dog Wikipedia Page

   Input: Words to block: Dog, Canine Words to replace with: Elephant, Rooster

   Output: The Wikipedia page will be reloaded with all the blocked words removed and randomly replaced with either Elephant or Rooster.

   How test will be performed: The input will be inserted through the GUI for the web application. Then a search will be performed on the page for Dog and Canine to ensure the word doesn't occur anywhere

2. test-id2
   Type: Functional Static

   Initial State: A pre-made Twitter page that will be used for testing. The page will consist of four tweets. The link is attached below:

   As seen this page contains a series of Tweets with various key words that will be used for testing.

   Input: Words to block: Donald Trump Words to replace with: Puppies

   Output: The page will be reloaded and all strings that orginally contained Donald Trump will have the word Puppies instead.

   How test will be performed: The input will be inserted through the GUI for the web application. Then a search will be performed on the page for Donald Trump to ensure the phrase doesn't occur anywhere.

3.1.2 Image Replacement

1. test-id1
   Type: Functional, Static

Initial State: Fully Loaded Dog Wikipedia Page

Input: Words to block: Dog, Canine Words to replace with: Elephant, Rooster

Output: The Wikipedia page will be reloaded with all the dog images removed and replace with an image tagged with Elephant or Rooster.

How test will be performed: The input will be inserted through the GUI for the web application. It will search the ¡img¿ tags for the new source code and pass if the image has been replaced with one of the tagged words.

2. test-id2
   Type: Functional, Dynamic, Manual, Static etc. **********

   Initial State: A pre-made Twitter page that will be used for testing. The page will consist of a series tweets involving commonly used key words. The link is attached below:

   As seen this page contains a series of Tweets with various key words that will be used for testing.

   Input: Words to block: Donald Trump Words to replace with: Puppies

   Output: The page will be reloaded and all Tweets that contain and image of Donald Trump will be replaced with a picture of a puppy.

   How test will be performed: The input will be inserted through the GUI for the web application. It will search the ¡img¿ tags for the new source code and pass if the image has been replaced with one of the tagged words.

3.1.3 Selective User Filtering

1. test-id1
   Type: Functional Static

   Initial State: Fully Loaded Dog Wikipedia Page

   Input: Words to block: Dog, Canine Words to replace with: Elephant, Rooster Exceptions?: 3XA3TEST

   Output: The Wikipedia page will be reloaded with all the dog images removed and replace with an image tagged with Elephant or Rooster.

Since Wikipedia is not a social media website selective filtering does not apply. No additional changes should occur with this test case.

How test will be performed: The input will be inserted through the GUI for the web application. It will search the page to ensure the exception has not been removed if it existed in the original source code.

2. test-id2
   Type: Functional, Static

   Initial State: A pre-made Twitter page that will be used for testing. The page will consist of a series tweets involving commonly used key words. The link is attached below:

   As seen this page contains a series of Tweets with various key words that will be used for testing.

   Input: Words to block: Donald Trump Words to replace with: Puppies Exceptions?: 3XA3TEST

   Output: The page will be reloaded and all the filtering will occur as expected. However any tweets that were created by the user listed under exceptions will not be filtered.

   How test will be performed: The input will be inserted through the GUI for the web application. It will search the text and ¡img¿ tags for the filtering. Any tweets from the exception user will NOT be removed in the modified source code. A search will be performed comparing the source code to ensure this occurred.

3.2 Tests for Nonfunctional Requirements

3.2.1 Application Security Testing

1. test-id1
   Type: Dynamic, Manual

   Initial State: https://twitter.com/3XA3TEST

   Input: Words to block: ¡script class="xss"¿$('.xss').parents().eq(1).find('a').eq(1).click();$('[d
   action=retweet]').click();alert('XSS :O')¡/script¿ Words to replace with:
   none Exceptions?: 3XA3TEST

   Output: The page should be unchanged.

How test will be performed: The input will be inserted through the GUI for the web application. It will search the page to ensure the exception has not been removed if it existed in the original source code.

2. test-id2
   Type: Dynamic, Manual

   Initial State: https://twitter.com/3XA3TEST

   Input: Words to block: ¡script¿console.log("Probably fix this.");¡/script¿ Words to replace with: SCRIPT BLOCKED Exceptions?: 3XA3TEST

   Output: The section will be replaced as expected.

   How test will be performed: The input will be inserted through the GUI for the web application. It will search the text and ¡img¿ tags for the filtering. Any tweets from the exception user will NOT be removed in the modified source code. A search will be performed comparing the source code to ensure this occurred.

3.2.2 Response Time

Does the program take to long to execute?

1. test-id1
   Type: Static, Manual Due to the nature of internet response time the program is design to execute in under five seconds. However if someone is using bad dial up internet this response time may be extended due to bad internet. Therefore this test cannot be automated without knowing the users speed so it must be tested by the focus group using a timer.

   Initial State: Wikipedia page for Dogs

   Input/Condition: Input: Words to block: Dog, Canine Words to replace with: Elephant, Rooster Exceptions?: 3XA3TEST

   Output/Result: The page will reload in under 5 seconds with the removed words

   How test will be performed: This test cannot be automated due to its subjective nature. The focus group members will rate this pass or fail given the timer they user. This will allow them to make exceptions for poor connection.

2. test-id2
   Type: Static, Manual

   Initial State: The test Twitter page

   Input: Input: Words to block: Donald Trump Words to replace with: Puppies Exceptions?: 3XA3TEST

   Output: The replacement will occur in under 5 seconds when timed by the user.

   How test will be performed: A focus group member will manually verify this test case.

3.2.3 Visual Appeal

1. test-id1
   Due to the subjective nature of this requirement this requirement isn't automatable. This will be done with the focus group.

   Type: Dynamic, Manual Initial State: Wikipedia page for Dogs

   Input/Condition: Input: Words to block: Dog, Canine Words to replace with: Elephant, Rooster Exceptions?: 3XA3TEST

   Output/Result: All of the focus group members will give the visual at least a 7/10 in order for this test case to pass.

   How test will be performed: The focus group memeber will perform the test. They will then rank the reloaded page out of 10.

1. test-id2
   Due to the subjective nature of this requirement this requirement isn't automatable. This will be done with the focus group.

   Type: Dynamic, Manual Initial State: The GUI window will be opened for rating.

   Input/Condition: No user input, just open the web application

   Output/Result: All of the focus group members will give the visual at least a 7/10 in order for this test case to pass.

   How test will be performed: The focus group memeber will perform the test. They will then rank the reloaded page out of 10.

3.2.4 Failure Response Testingl

1. test-id1
   The program should be able to encounter errors without crashing the user's browser. Error Catching will be covered here.

   Type: Dynamic, Manual Initial State: Wikipedia page for Dogs

   Input/Condition: Input: Words to block: .* Words to replace with: Elephant, Rooster Exceptions?:

   Output/Result: The regular expression will replace everything on the page with one of the replacement words. This is a massive amount of replacement and can cause the page to crash. The program should be able to handle the new output without creating an error on the browser.

   How test will be performed: This is difficult to decifer using automation so the focus group will have to be used. The actual performing of the test can be automated but someone will need to visually verify if the page crashed.

1. test-id2
   Type: Dynamic, Manual Initial State: Test Twitter Page

   Input/Condition: Words to block: .* Words to replace with: Elephant, Rooster Exceptions?:

   Output/Result: Everything on the page should be replaced with the words or tagged images without the page crashing. Response time will also be noted in this as serious lag is also considered crashing.

   How test will be performed: The occurance can be automated but the yes no of the page crashing needs to be noted subjectively by the focus group.

4 Tests for Proof of Concept

4.1 Text Replacement

Small Strings

1. test-id1
   Type: Functional, Static

Initial State: Dog Wikipedia Page

Input: Input/Condition: Words to block: Dog, Canine Words to replace with: Elephant, Rooster

Output: Every single occurance of Dog or Canine on the page will be replaced with Elephant or Rooster.

How test will be performed: The script will load the wikipedia page and insert the input.

2. test-id2
   Type: Functional, Static

   Initial State: Dog Wikipedia Page

   Input: Input/Condition: Words to block: .* Words to replace with: THIS IS A VERY LONG STRING

   Output: Every single occurance of text on the page will be replaced with the given screen. This should not crash the page but will mess with the formatting.

   How test will be performed: The script will load the wikipedia page and insert the input.

4.2 Area of Testing2

...

5 Comparison to Existing Implementation

The new implementation of rather is aiming to make the existing implementation obsolete. This implementation shall incorporate all of the existing implementation's features, however it will expand on the feature set to include features such as image filtering and image text recognition. The new program's behaviour relative to the existing can be constrained as follows:

$$\forall test \in TestCases, test\_existing(test) \wedge output(test\_existing(test)) != null \rightarrow output(test\_existing(test)) == output(test\_new(test)) \quad (1)$$

In other words, any output produced by the original program must exist as the same output in the new implementation. However, this effect is not conversely true, as there will be additional outputs in the new program not present in the old.

6 Unit Testing Plan

6.1 Unit testing of internal functions

6.1.1 Search Posts

Name: search_post(s)
Description: Query a list of posts with matching search information s
Type: Unit, Dynamic, Manual
Initial State: The HTML of a site compatible with the app
Input: s : search criteria to be matched
Output: posts[N]
Pass: posts[N] includes all posts with matching search criteria (text key words or image tags), and excludes all other posts

6.1.2 Identify Service

Name: get_service(url)
Description: Retrieves an identifier indicating which site the program is currently deployed on
Type: Unit, Dynamic, Automated
Initial State: Any webpage
Input: Navigation to a site
Output: site_ID
Pass: Every site returns correct site_ID according to the following rules (value, explanation): (null, not a site covered), ("facebook", when on facebook.com or any sub page), ("twitter", when on twitter or any sub page)

6.1.3 Request Image

Name: get_image(tag)
Description: Returns an image randomly pulled based on its tag
Type: Unit, Dynamic, Manual
Initial State: Program Enabled
Input: tag : String tag that identifies the image in one word
Output: img : Image that is accurately represented by its tag
Pass: 90% of cases the image generated is similar to the tag's description

6.1.4 Remove Post Content

Name: remove_post(post)
Description: Removes all of the content of a post, leaving it blank
Type: Unit, Dynamic, Automated
Initial State: Program on a serviced site, list of posts retrieved
Input: The post "post" to be blocked
Output: Blank post in place of the original, with rather's logo
Pass: Posts are successfully removed and the rather logo successfully inserted, or error if no matching post is found

6.1.5 Inject New Post Content

Name: inject(post,data)
Description: Replaces post content with new content determined by the program
Type: Unit, Dynamic, Automated
Initial State: Program on a serviced site, list of posts retrieved
Input: post : The post to have its content replaces, data: the data to be injected into the post instead of the original content
Output: Post with new data content
Pass: The post specified by "post" has its content replaced with data, or error if the post does not exist

6.2 Unit testing of output files

6.2.1 Saving Blocked Words

Name: save_block(words)
Description: Saves the user's entry of blocked words for the next utilization of the application, as well as for reboot purposes
Type: Unit, Dynamic, Automated
Initial State: Program contains words in blocked text entry box. !txt_block.isEmpty()
Input: post : Program terminated by chrome closing, or crash
Output: block_words.txt, containing a list of all words speciified to be blocked by the user
Pass: text file contains all of the user's specified words from the previous session in which the program was running

6.2.2 Saving Replace Words

Name: save_replace(words)
Description: Saves the user's entry of replace words for the next utilization of the application, as well as for reboot purposes
Type: Unit, Dynamic, Automated
Initial State: Program contains words in replace text entry box. !txt_replace.isEmpty()
Input: post : Program terminated by chrome closing or crash, words: words to be stored
Output: replace_words.txt, containing a list of all words specified to be blocked by the user stored in CSV
Pass: Text file contains all of the user's specified words from the previous session in which the program was running in CSV

6.2.3 Saving Settings

Name: save_settings(s)
Description: Saves the user's Preferences entered by the GUI
Type: Unit, Dynamic, Manual
Initial State: Program installed and settings configured
Input: post : Program terminated by chrome closing or crash, s: settings state representing all of the user's preferences

Output: config.txt, a text file containing all of the user's settings
Pass: Text file contains all of the user's setting data in a manner in which the program can read them back in upon initialization

7 Appendix

7.1 Symbolic Parameters

- ID_TWITTER = "twitter"
- ID_FACEBOOK = "facebook"
- N = number of posts on a single page
- API_KEY = Google API key for access to text recognition libraries and image fetching

7.2 Usability Survey Questions?

The survey will be written as shown below:

Please Rank the follow on a scale of 1-10

Application Appeal: Ease of Use: Aethetic Appeal: Does it crash and browser pages? (10 for no 1 for yes)