SE 3XA3: Test Report
Rather 2.0

Team # 3, Team 3
Erin Varey - vareye
Nik Novak - Novakn
Joel Straatmen - Straatjc

December 8, 2016

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
December 5th	1.0	Starting Document
Date 2	1.1	Notes

This document is a report of Test cases performed in document Test Plan. The section and test case id is referenced along with whether the test case passed or failed. If context is needed it is given with the test case. The testing of the web application was performed using Q-unit on portions of the code along with manual testing. The manual testing was performed by both the group members and a focus group consisting of ten students. These members are in a variety of majors and half do not have any technical knowledge or experience.

1 Functional Requirements Evaluation

Can the program search and replace Text?:
Test-Id-1:
This test case was automated and is described in more detail below. Test case passed.
Test-Id-2
This test case was automated and is described in more detail below. Test case passed.

Image Replacement:
Test-Id-1:
This test was performed manually by team members. (It's hard to automate whether an image replaced correctly through unit testing.) This test cased passed.
Test-Id-2:
This test was performed manually by team members. The test case passed.

Selective User Filtering:
Test-Id-1:
This test case passed.
Test-Id-2
This test case passed.

2 Nonfunctional Requirements Evaluation

Security Testing Results:
User inputting a script:
Originally this test case was performed on Twitter. We have since limitted it to Facebook and this test has been modifed which is reflected in the Test Plan document. When script tags are inputted in the list option nothing happens. Chrome extensions auto block all scripts as input to prevent malicious content. Both test cases contained script tags and passed the manual testing.
Response Time:
Test-Id-1:
This test was performed on Erin's internet as it is the most consistent connection of the group members. Test 1 passed. Results were outputted in under one second.
Test-Id-2: This test was performed on the Mac Secure network. This Wifi connection is the least consistent and slowest and was used as an edge case to see how the application responds. Test case 2 passed. Results were outputted in 3 seconds and loaded at the same rate as the original page.
Visual Appeal:
See Usability section for survey results. Average results was above 7. Test passed.
Failure Response Testing:
Test-Id-1:
The application still accepts regular expressions but it does not crash the page when used. The code limits the replacement to just the news feed/ page feed so there are no issues created in the side bar or chat bar. Test case passed.
Test-Id-2:
The page did not crash. Test case passed.

2.1 Usability

This section of the testing was performed using a survey from our focus group. The survey consisted of the following questions:
Please Rank the follow on a scale of 1-10
Application Appeal:
Ease of Use:

Aethetic Appeal:
Does it crash anybrowser pages? (10 for no 1 for yes)
Would you reccomended this application? (10 for yes 1 for no)
Additional comments?

10 people were surveyed in total. The results were over all positive with the average value of the results described below.

Name: Cameron
Application Appeal: 7
Ease of Use: 7
Aesthetic Appeal: 9
Does it crash anybrowser pages? (10 for no 1 for yes) 10
Would you reccomended this application? (10 for yes 1 for no) 10
Additional comments? I was impressed with all the images this could recognize. Wish I didn't have to refresh the page to reset it.

Name: Josh
Application Appeal: 9
Ease of Use: 8
Aesthetic Appeal: 8
Does it crash anybrowser pages? (10 for no 1 for yes) 10
Would you reccomended this application? (10 for yes 1 for no) 10
Additional comments? I would definitely use this for my Facebook feed. I wish it worked on other social media websites like Twitter though.

Name: Stuart
Application Appeal: 9
Ease of Use: 9
Aesthetic Appeal: 9
Does it crash anybrowser pages? (10 for no 1 for yes) 10
Would you reccomended this application? (10 for yes 1 for no) 10
Additional comments? Pretty Cool

Name: Nicholas
Application Appeal: 10
Ease of Use: 9
Aesthetic Appeal: 7

Does it crash anybrowser pages? (10 for no 1 for yes) 10
Would you reccomended this application? (10 for yes 1 for no) 10
Additional comments? I'll definitely start using this. It isn't very pretty though

Name: Jessica
Application Appeal: 9
Ease of Use: 7
Aesthetic Appeal: 10
Does it crash anybrowser pages? (10 for no 1 for yes) 10
Would you reccomended this application? (10 for yes 1 for no) 10
Additional comments? There was a bit of a learning curve for those who arn't technically inclined

Name: Phillip
Application Appeal: 10
Ease of Use: 10
Aesthetic Appeal: 10
Does it crash anybrowser pages? (10 for no 1 for yes) 10
Would you reccomended this application? (10 for yes 1 for no) 10
Additional comments? FANTASTIC!!

Name: Matthew
Application Appeal: 10
Ease of Use: 10
Aesthetic Appeal: 8
Does it crash anybrowser pages? (10 for no 1 for yes) 10
Would you reccomended this application? (10 for yes 1 for no) 10
Additional comments? This could be really useful for using in public or at work. I like that it remembers my kill list.

Name: Shannon
Application Appeal: 9
Ease of Use: 9
Aesthetic Appeal: 8
Does it crash anybrowser pages? (10 for no 1 for yes) 10
Would you reccomended this application? (10 for yes 1 for no) 10
Additional comments? I wish I could use this on other websites other than

Facebook

Name: Brie
Application Appeal: 10
Ease of Use: 9
Aesthetic Appeal: 9
Does it crash anybrowser pages? (10 for no 1 for yes) 10
Would you reccomended this application? (10 for yes 1 for no) 10
Additional comments? I suffer from anxiety and this is definitely something I'll use to remove things that trigger it.

Name: Emma
Application Appeal: 10
Ease of Use: 7
Aesthetic Appeal: 8
Does it crash any browser pages? (10 for no 1 for yes) 10
Would you reccomended this application? (10 for yes 1 for no) 10

Average appeal score: 9.3
Average ease of use score: 8.5
Average aesthetic appeal score: 8.6
Crash score: 10
Reccomendation score: 10
As shown above the average of each score is above 7. Each of these test cases pass.

2.2 Performance

Application performance testing was included in the functional and non-functional requirements. Performance time, malicious script handling, and error catching is included in these test. Each of these three sections had all of their test cases pass. The passing of these cases were integral to the performance of the application and were performed until they passed.

3 Comparison to Existing Implementation

The test were performed on the existing implementation. This section is not applicable.

4 Unit Testing

Unit Testing was an important aspect of code development. Separating various functions from the code base and testing them, both manually and automatically helped solve bugs and allow for a deeper understanding of method behaviour and interaction. Below is the list of Unit Tests originally developed during the Unit Testing Plan. Each additionally contains an "Expected Output" field as well as an "Actual Output".

5 Changes Due to Testing

Initially, a few modules failed under unit testing. This resulted in changes of the code to account for failing conditions.

Unit test 6.2.1 failed under the extreme case in which the user entered 999 words to be saved each of which were of length 99. This lead to the discovery that the HTML5 localStorage allows only 5Mb of storage. Due to this discovery, the code was changed to avoid this case, and notify the user that too many words could result in a loss of the extension's userdata.

Unit test 6.1.1 also failed under the assumption that all facebook posts contained a certain class ID that identified all posts when used in a query selector. It was discovered due to testing that this ID was unique only to each facebook user. This implied that the program would only work on the initial developer's facebook account. An adjustment was quickly made to broaden scope of the search to a segment of the class that posts are stored under.

6 Automated Testing

Automated testing was employed on several units to cover a wide range of inputs to each function being tested, and compute whether the outputs were correct. This resulted in a very efficient and effective test coverage, spanning from normal use cases to extreme boundary cases. Many functions were non-automatable, as much of the program works on the basis of image analysis, and correctness due to image content. The few functions that could qualify for unit testing were tested, and the following results were obtained:

The test report can be found in the documentation, under AutomatedTestReport.png.

All test scripts used to generate the report can be found under the directory testingdir. The report consists of five HTML test pages and matching test scripts. Each test script acts on its HTML test page, and attempts to perform its function. At the end of the test, the state of the HTML page is compared with the desired results for that test case. If the results agree, a "PASS" is issued, otherwise "FAIL". In the case that the program errors, a "FAIL" is issued, accompanied with an error message.

7 Trace to Requirements

Req.	**Modules**
1. User can input ”kill list”	popup.html
2. User can input ”replacement list”	popup.html
3. User can decide whether to mute or replace posts	popup.html
4. Program identifies individual posts and is only active on Facebook.	Facebook-API.js, Detect-Website.js
5. Program searches for keywords in both text and image contents.	Image.js, Content.js
6. Program can identify images within a post.	Image.js
7. Program mutes or replaces content when keywords are found.	Image.js, Content.js, Popup.js
8. Program can ”un-filter” specific people’s posts.	popup.html

Table 2: Trace Between Requirements and Modules

8 Trace to Modules

Level 1	**Level 2**
Module and requirements	
Functional	Facebook-API.js —— Allow the user to block specific items in their newsfeed Filter-Feature.js —— Allow the user to filter individual post they do not want censored Detect-Website.js —— Allow the user to block items on specific web feeds. (Will allow functionality for Twitter and Facebook) Image.js ——If a keyword is found images containing or tagged with it will be filtered page.js —— Main Controller for direction of information flow and program state popup.js —— Allow the user to filter content on websites that are not twitter and facebook content.js —— Allow the user to detect both content and tags related to a word popup.html —— Allow the user to input a ”kill list” of words they want to remove popup.html —— Allow the user to input a ”replacementl list” of words they want to remove popup.html —— Allow the user an option to mute or replace content
Non-Functional	? manifest.json —— performance requirements and file naming index file popup.html —— Allow the program to have a sleek look and feel

Table 3: The modules that satisfies each requirement is listed

9 Code Coverage Metrics

This was not deployed in our automated testing implementation.