SE 3XA3: Test Plan
MacSidenotes

Team 4
Josh Mitchell mitchjp3
Matthew Shortt shorttmk

December 7, 2016

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	Version	Notes
Oct 30th, 2016	0.5	Sections 2, 4, 5, 6 Added
Dec 7th	1.0	Revision 1 Modifications

This document specifies the tools and techniques that will be used to test the adherence of MacSidenotes to its Functional and Non-Functional Requirements.

1 General Information

1.1 Purpose

The purpose of this document is to create a plan that testers can easily follow in order to catch and resolve as many program errors as possible. This plan includes a wide variety of test cases that cover both functional and non-functional requirements together with automated and manual testing strategies.

1.2 Scope

The Chrome extension that is created for this project is one that is very simple to use and very useful for many people. Since the project is very compact, the scope for our test plan will cover the front end and the back end of our implementation. Our front end being the GUI the user interacts with, and the back end being what the code does behind the scenes, such as using Chrome's localStorage to save users Notes. The scope of the test plan will inevitably expand proportionally to the growth of the project, which may include an option of saving the Note to the users Google Drive.

1.3 Acronyms, Abbreviations, and Symbols

Table 2: **Table of Abbreviations**

Abbreviation	**Definition**
localStorage	Google Chrome's method for storing data locally.
JS	JavaScript
URL	Uniform Resource Locator
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
POC	Proof Of Concept

Table 3: **Table of Definitions**

Term	Definition
Note	Text created by the user. Usually intended to be saved and viewed at a later time.
Sidebar	The popup window that appears when a user clicks on the MacSidenotes icon. In the sidebar: users write, save and delete notes, as well as view the master list of notes.
Master List	The list containing all previously saved notes. It can be viewed by clicking the List button.

1.4 Overview of Document

This document is a summation of all the tests that will be performed on our projext to ensure that we detect as many errors as possible. This includes tests which are automated as well as manual.

2 Plan

2.1 Software Description

MacSidenotes is a Chrome extension that allows users to create notes for a webpage and save them alongside that page's URL. They can then view all previous notes in a list and navigate to a URL to continue their previous work.

2.2 Test Team

The following project members will be responsible for writing and executing tests on MacSidenotes:

- Josh Mitchell
- Matthew Shortt

2.3 Automated Testing Approach

Given the UI-driven nature of this project, much of the automated testing will be focused on user interaction, which can be simulated with our Testing Tool ~~QUnit~~Jasmine.

2.3.1 Whitebox Testing

Values held in the program will be examined before and after ~~a simulated click or keyboard event~~ internal functions are called, to ensure proper updating of system variables. These tests are Whitebox or Structural tests, as they require knowledge of variable names and the "under-the-hood" storage architecture used by the program.

2.3.2 Blackbox Testing

The very basic functionality of MacSidenotes involves the appearance and disappearance of UI elements. Testing this functionality can be done without knowing the internal structure of the program, just the knowledge that the "Show List" button should show the user a list.

2.4 Testing Tools

~~QUnit~~ Jasmine will be the dominant tool for testing the JavaScript functionality of the extension. It facilitates automated unit testing, assertions, and synchronous and asynchronous callbacks ~~as well as testing user actions through simulated mouse clicks and keyboard input~~.

2.5 Testing Schedule

See Gantt Chart at the following links:

- .gan format
- .pdf format

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 Sidebar - Typing Window

Title for Test

1. Sidebar Open

   Type: Functional, Dynamic, Manual

   Initial State: Default Chrome browser window.

   Input: Extension icon click.

   Output: Sidebar will appear at the top right of the page.

   How test will be performed: Test will be performed by clicking the icon for the MacSidenotes extension at the top right hand corner of the Google Chrome browser.

2. Sidebar Close-1

   Type: Functional, Dynamic, Manual

   Initial State: Sidebar appears at the top right of the page.

   Input: Extension icon click.

   Output: Sidebar disappears leaving solely the browser page.

   How test will be performed: Test will be performed by clicking the icon for the MacSidenotes extension at the top right hand corner of the Google Chrome browser.

3. Sidebar Close-2

   Type: Functional, Dynamic, Manual

   Initial State: Sidebar appears at the top right of the page.

   Input: Click event on browser page.

   Output: Sidebar disappears leaving solely the browser page.

How test will be performed: Test will be performed by clicking anywhere on the browser page excluding the extension icon or within the text window.

3.1.2 Note Taking

Typing

1. Input 1

   Type: Functional, Dynamic, Manual

   Initial State: Blank text area.

   Input: User keyboard input.

   Output: User keyboard input.

   How test will be performed: Test will be performed by typing any keyboard input into the text area and verify that the output matches exactly what was typed.

3.1.3 Note Saving

Save Note

1. Save Note - 1

   Type: Functional, Dynamic, Manual

   Initial State: Text window with keyboard input.

   Input: Click event on 'Save Note' button.

   Output: Confirmation Message 'Note Saved!'

   How test will be performed: Test will be performed by clicking the 'Save Note' button and verifying that when the note is closed and re-opened it is the same as when the save button was last clicked.

3.1.4 Master List

List Management

1. List View - Open

   Type: Functional, Dynamic, Manual

   Initial State: Sidebar with solely text area and buttons.

   Input: Click event on 'List' button.

   Output: Master List shown below the contents of the Sidebar.

   How test will be performed: Test will be performed by clicking the 'List' button located on the extension window and verifying that the Master List appears on the Sidebar.

2. List View - Closed

   Type: Functional, Dynamic, Manual

   Initial State: Sidebar with text area, buttons and Master List.

   Input: Click event on 'List' button.

   Output: Master List disappears from the Sidebar.

   How test will be performed: Test will be performed by clicking the 'List' button located on the extension window and verifying that the Master List disappears on the Sidebar.

3.1.5 Note Deletion

Delete Note

1. Delete Note

   Type: Functional, Dynamic, Manual

   Initial State: Text Area with a note saved to it.

   Input: Click event on 'Delete Note' button.

   Output: Deletion confirmation message.

How test will be performed: Test will be performed by clicking the 'Delete Note' button and assuring that a confirmation message appears saying 'Are you sure you want to delete the note?', along with two choices, 'YES' or 'NO'.

2. Delete Confirm - YES

   Type: Functional, Dynamic, Manual

   Initial State: 'Delete Note' has been clicked prompting confirmation.

   Input: Click event on 'YES' button.

   Output: Confirmation of deletion message.

   How test will be performed: Test will be performed by clicking the 'YES' button and assuring that a confirmation message appears saying 'Note Deleted' as well as affirming the note has been removed from Master List.

3. Delete Confirm - NO

   Type: Functional, Dynamic, Manual

   Initial State: 'Delete Note' has been clicked prompting confirmation.

   Input: Click event on 'NO' button.

   Output: Confirmation message that note was not deleted.

   How test will be performed: Test will be performed by clicking the 'NO' button and assuring that a confirmation message appears saying 'Note Was Not Deleted' as well as affirming the note remains in Master List.

3.2 Tests for Nonfunctional Requirements

3.2.1 Look and Feel

Look

1. Sidebar Size

   Type: Manual, Dynamic

Initial State: Default Chrome browser window.

Input: Extension icon click.

Output: Sidebar will appear at the top right of the page.

How test will be performed: Test will be performed by clicking the icon for the MacSidenotes extension at the top right hand corner of the Google Chrome browser and ensuring that the Sidebar does not take up more than SIDEBAR_WIDTH

3.2.2 Usability and Humanity Requirements

Usability & Humanity

1. Communication - English

Type: Manual, Static, Dynamic

Initial State: Varied

Input: N/A

Output: N/A

How test will be performed: The test will be performed by activating every message to the user and ensuring that they are easily understandable and in English.

3.2.3 Performance

Speed

1. Extension Response

Type: Manual, Dynamic

Initial State: Varied

Input: User input.

Output: Extension output.

How test will be performed: This test will be performed by ensuring that any interaction the user makes with the extension ~~takes less than RESPONSE_TIME~~ is not too slow as to disrupt their usual flow.

4 Tests for Proof of Concept

4.1 Sidebar

Sidebar Opening

1. Sidebar Opening

   Type: Functional, Dynamic, Manual

   Initial State: Default Chrome browser

   Input: Click event to the extension icon.

   Output: Sidebar pops-up.

   How test will be performed: This will be tested by clicking the extension icon and ensuring that the sidebar opens.

4.2 Save Feature

Local Save

1. Saving Note

   Type: Functional, Dynamic, Manual

   Initial State: Opened Sidebar.

   Input: Click event on the 'Save' button.

   Output: Confirmation message appears ensuring that note has been save.

   How test will be performed: This will be tested by clicking the 'Save' button and ensuring that in the JS of the project the note and the URL associated with that note is accessible in the localStorage of that user.

5 Comparison to Existing Implementation

The project that MacSidenotes is derived from (sidenotes) is currently unable to be tested. It relies on Dropbox's Datastore API which was retired 2 years ago, so it has been defunct since then.

That said, the basic superficial functionality of MacSidenotes can still be contrasted against sidenotes, as the original repository contains a system-level description of its abilities and a gif displaying a user's interaction with the extension.

6 Automated Unit Testing Plan

6.1 Unit testing of internal functions

Each of the below tests requires its own driver. None require a stub. Some previously planned automated unit test cases had to be removed, as the testing framework does not simulate the calls to the Chrome extension API. More unit tests that Jasmine can execute were added to compensate.

6.1.1 deleteNote

~~deleteNote removes the note associated with the URL the user is currently viewing from the master list of notes.~~
~~**To test:** Simulate a click of the Delete Note button with QUnit, then search through Chrome's local storage to ensure no notes exist with a URL that matches the URL the "user" was viewing when deleting that note. If there is no match, the test is successful.~~

6.1.2 deleteEmpty

~~deleteEmpty removes all empty notes from the master list.~~
~~**To test:** Call deleteEmpty, search through local storage to ensure no empty notes exist. If no empty notes are found, the test is successful~~

6.1.3 clickCounter

clickCounter increments numClicks, which counts the number of times the List button has been clicked.
To test: Check the value of numClicks, ~~simulate a click of the List button,~~ callClickCounter() and check to see if numClicks has incremented by 1. If so, the test is successful.

6.1.4 showList

showList displays the master list of notes to the user. It also removes it from the screen if it is already present.
To test: ~~Simulate a click of the List button~~Call showList() and clickCounter(), check the visibility of the List element. ~~Simulate another click~~Call the functions again and check the visibility again. If ~~they do not match~~ after the first click it is visible and after the second click it is not, the test is successful.

6.1.5 updateMasterList

~~updateMasterList appends the note the user has saved to the master list so it can be viewed when the List button is clicked.~~
~~**To test:** Simulate writing a note and clicking the Save Note button. Simulate a click of the List button and check to see if an element of that list exists with the same content and associated URL of the typed note. If both the content and the URL match, the test is successful.~~

6.1.6 updateNote

~~If the user has previously saved a note for the URL they are currently viewing, updateNote will display it when the extension icon is clicked.~~
~~**To test:** Simulate the writing and saving of a note. Close MacSidenotes and click on the icon again. Check the value of the sidebar's text area against the typed note. If the content of the text area matches the typed note, the test is successful.~~

6.1.7 saveNote

~~saveNote saves the current note in local storage along with it's associated URL.~~
~~**To test:** Simulate the writing and saving of a note. Check local storage to ensure that the master list includes a note that contains the same content and associated URL as the typed note. If both the content and the URL match, the test is successful.~~

6.1.8 getURL

~~getURL returns the URL of the webpage the user is currently viewing.~~
~~**To test:** Call getURL and check it's value against Chrome's official code for grabbing URLs found here~~

6.1.9 emptyMasterList

emptyMasterList empties the master list of notes.
To test: Add a number of dummy rows to the Master List. Confirm that the number of rows added is as intended. Then call emptyMasterList() and check to see if the List has no rows in it. If it has no rows, the test is successful.

6.1.10 showNotice

showNotice reveals an icon to the user, notifying them when a Note has been saved or deleted.
To test: Check if the saveNotice and deleteNotice icons are not displayed. call showNotice(), passing both saveNotice and deleteNotice. Check if they are both now displayed. If they were only displayed after the function call, the test is successful.

6.2 Unit testing of output files

MacSidenotes does not create output files, as everything is contained within the Chrome browser's local storage.

References

7 Appendix

This is where you can place additional information.

7.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

Table 4: **Table of Symbolic Parameters**

Symbolic Constants	**Value**
RESPONSE_TIME	The maximum amount of time the system has to respond to a user interaction. This number is 2 seconds.
SIDEBAR_WIDTH	The maximum width that the Sidebar should be upon first opened. The maximum sidebar width is 30% of a users full browser size.

7.2 Usability Survey Questions?

During October of 2016 Matthew asked his two roommates about the usability of the product. At this point the product was at the proof of concept stage.

Some of the questions asked were:

What do you like about the product?
What do you think could be improved?
What features would you like to see added to the product?
Would you use the product?

Both roommates liked the product as-is due to the facility in which it can be used in tandem with it's practicality. Some suggestion put forward included:

Bullet Points

Font Colours/Sizes/Effects
Highlights
The ability to add pictures
Notification if page has note on it already
Titles for notes

Overall the meeting was very successful and produced many great ideas that could find their way onto the final product. Both roommates said they would definitely use the product.