Software Requirements Specification

for

TabUltra

Version 3.0 approved

Prepared by Vincent Bistricky, Adam Abdulhalim, Kyle Butler, Dylan Brown, Christopher Begley

UNF School of Computing, CEN4010 Software Engineering

03/08/23

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1

2. Overall Description 2

2.1 Product Functions 2

2.2 User Classes and Characteristics 2

2.3 Operating Environment 2

2.4 Design and Implementation Constraints 3

2.5 User Documentation 3

2.6 Assumptions and Dependencies 3

3. External Interface Requirements 4

3.1 User Interfaces 4

3.2 Hardware Interfaces 4

3.3 Software Interfaces 4

3.4 Communications Interfaces 5

4. System Features 5

4.1 System Feature 1 4

4.2 System Feature 2 (and so on) 4

5. Other Nonfunctional Requirements 4

5.1 Performance Requirements 4

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

6. Other Requirements 5

Appendix A: Glossary 5

Appendix B: Analysis Models 5

Appendix C: To Be Determined List 6

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Purpose

This requirements document is for the project TabUltra for all currently developed versions (currently TabUltra V3). The purpose of this document is to develop a Google Chrome extension for managing, grouping, and saving tabs. It will detail and outline functional and non-functional requirements, how the user interfaces with the product, and any constraints on the product itself.

## Document Conventions

Functional/non-functional requirements with more detailed descriptions can be assumed to be of higher priority in development and implementation. Requirements will also not be listed in order of creation/importance and may vary in length and importance. Some references used may not be accessible to general audiences due to requiring course codes/monetary purchases.

## Intended Audience and Reading Suggestions

This document is intended for team developers to review current necessary requirements, as well as college professors for grading and review. All product specifications and necessary requirements will be listed in further sections, including but not limited to API integration, UI design, and system architecture.

## Product Scope

This project aims to create a Google Chrome extension that allows the user to select and save any currently open tabs into a group, which will then be stored locally on the user’s device and can be then re-opened at any time. We aim to develop the extension in a way that is unintrusive as a way to differentiate from similar products. Giving the user choices in which tabs to save/open is also a major goal in product development.

## References

* Chrome storage API: <https://developer.chrome.com/docs/extensions/reference/storage/>
* JavaScript/HTML zyBooks online lessons (taken from UNF access codes)

# Overall Description

## Product Functions

TabUltra consists of these major functional requirements:

* Program must register when browser tabs are opened/closed. This extends to tabs across multiple windows.
* Program must display list of currently open tabs to user.
* User must be able to save all currently open tabs into a group.
* Program must display list of all saved groups, with each tab in said group listed.
* User must be able to select a specific group to open.
* User must be able to delete a group that is no longer being used.
* User must input a name in order to create a group.
* Program must be able to switch between views of groups and tabs.
* User must be re-directed to landing page when selecting the “cog” icon.

## User Classes and Characteristics

The user class that will likely use this product the most will be students. TabUltra is designed for users who consistently have large amounts of tabs open, so students needing resources for projects, studying, or research will frequently be creating new groups and deleting groups no longer necessary. Most functions of the product will be used by this user class as nearly all functions are geared towards making the primary functionality of saving/grouping tabs easy to use and understand. Professors/workplace professionals may find similar use for this product, as topic/job research have similar requirements to a student researching for a product. The least important user class is likely an elderly demographic. While easy to use, it is unlikely that an elderly person will need more than a few tabs at any given time for any given reason, so there would be little purpose in a product like TabUltra.

## Operating Environment

TabUltra is currently only operable on Google Chrome as it makes heavy use of Google Chrome’s “chrome.storage” API. TabUltra retrieves all of its data from API calls to document and list the user’s currently opened tabs, as well as storing the group data locally, so any groups saved on TabUltra are not transferrable across devices. This also means that no other third-party extension should have any effect on TabUltra’s functionality barring extensions that change the physical properties of Google Chrome browser tabs. TabUltra is also currently only available for non-mobile devices such as PC and laptop. TabUltra is available for use on both Windows and macOS devices.

## Design and Implementation Constraints

The largest implementation constraint to TabUltra was the data storage limits to the “storage.sync” API calls that were initially being used for group storage that allowed us to store groups across multiple devices if the user has their Google Chrome accounts synced. However, the data limit for this API call was only approximately 100 KB, which could only store two or three tabs total (across all groups) before new tabs could not be saved. This was simply not a justifiable compromise for our product, so syncing across devices had to necessarily be scrapped (resulting in TabUltra V2).

We are also confined to limited tools for retrieving and manipulating data received from API calls due to developing the product with a JavaScript backend. This severely limited our ability to manipulate saved user data in ways that would make using the product easier and more efficient (resulting in TabUltra V3). Lastly, while unlikely this data cap would be realistically reached, “storage.local” does also have a data cap of 5 MB, but this can be circumvented by requesting the “unlimitedStorage” permission if it is deemed necessary.

## User Documentation

Currently the only user documentation that is provided along with the product is a landing page redirect on the main extension window. The landing page is a website (currently only a series of html files) that include a “Frequently Asked Questions” section, as well as a support email to reach out with any questions the user may or may not have.

## Assumptions and Dependencies

The largest assumption is that users will not reach the 5 MB data cap constrained by “storage.local.” If deemed necessary, the appropriate adjustments will be made. In order to implement other secondary requirements that were not possible due to the limited functionality we found with JavaScript, potentially re-writing the code for the extension in a more complex language such as Angular may provide the functionality needed to implement these scrapped functionalities. Landing page functionality would also be dependent on a web-hosting server that may or may not be hosting the landing page at any given time (depending on if the host is paid or has any sort of visitor limit).

# External Interface Requirements

## User Interfaces

TabUltra is split into two different “views.” Each view will feature the TabUltra logo at the top, along with a “cog” that will re-direct the user to a landing page for more information. Each view also features its own “sub-window” that includes a scrollbar for viewing each view’s specific information. The first of these views are used to display all of the currently open tabs to the user. Each tab will be listed in the order in which they appear in your list of Chrome tabs and have an associated number keeping track of them. Each tab also features a “close tab” button that will close the tab when clicked and a checkbox that the user can select when saving specific tabs. The currently active tab will have a black box surrounding it and black text. In this view, you can input the name of the group you wish to save tabs to, along with two buttons: one for saving specifically “checked” tabs, and one for saving all currently open tabs to a group. Buttons and drop-down menus do also currently exist for saving tabs to a pre-existing group, but this feature has yet to be implemented.

The groups view lists all groups currently saved and displays all tabs within that group below, once again listed with a number and checkbox. The user can rename any group they wish within this view. At the bottom of each group, a button will be available for the user to open all tabs within the group, as well as a button to delete the group if it is no longer being used.

## Hardware Interfaces

TabUltra does not communicate directly with any physical computer hardware.

## Software Interfaces

The Google Chrome extension created utilizes several software components and application programming interfaces (APIs) to function properly. The extension relies on the Chrome storage API to save and retrieve the user's saved tab groups. Specifically, the extension uses the chrome.storage.local.get and set methods to save and retrieve the data. Additionally, the extension uses the chrome.tabs.query and chrome.tabs.create methods to retrieve the currently opened tabs and open saved groups of tabs. These APIs are provided by the Google Chrome browser and are specific to the version of the browser that the extension is developed for.

The extension communicates with the Google Chrome browser to perform its functionality and does not rely on any external databases, operating systems, or libraries. The data items being passed into the system include the user's saved tab groups, which are then stored using the Chrome storage API. The data items going out include the newly opened tabs when the user chooses to open a saved group of tabs. The purpose of each message is to enable the extension to perform its core functionality of saving and reopening groups of tabs.

No specific implementation constraints are required for data sharing between software components, as the extension utilizes the Chrome storage API for data storage and retrieval. However, the extension requires the user to have a compatible version of Google Chrome installed on their operating system to function correctly. The API protocols used by the extension are documented by Google and are publicly available for reference by developers.

## Communications Interfaces

The only communication required for TabUltra is the “cog” icon re-directing to a publicly available website hosting the landing page for the product. There is currently no active webpage, but it is likely that HTTP would be used in some form to host the webpage. All emails sent from users to the personal product support email would not be encrypted in any form.

# System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

## System Feature 1

<Don’t really say “System Feature 1.” State the feature name in just a few words.>

4.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

4.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

4.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

REQ-2:

## System Feature 2 (and so on)

# Other Nonfunctional Requirements

## Performance Requirements

TabUltra must be able to register when a tab is opened/closed either in the extension itself or in Chrome within 0.2 seconds. It is imperative that the user knows exactly what tabs can/cannot be saved. Groups should also be created without the need to refresh the extension in the same timeframe. Due to an increase in memory usage once a very large number of tabs are opened, this timeframe can increase to a maximum of 1 second during peak memory usage. Groups that are deleted by the user should follow this same timeframe for the same reasons as listed above.

## Safety Requirements

There are no requirements that concern any possible loss, damage, or harm, as no code or attribute of the product has the capability to inflect any serious damage to a user or the user’s hardware.

## Security Requirements

User groups being stored locally using the “storage.local” API call may be a security/data risk, but at this time, no measures have been put in place by the developers to safeguard against this risk. It is assumed that Google Chrome itself has a built-in safeguard to protect user information from any sort of attack.

## Software Quality Attributes

TabUltra has the benefit of both ease of use and ease of learning due to the simplistic nature of its primary functionalities, as well as self-explanatory UI elements and error prompts. Once the product is in a serviceable state, it is planned to have it be put onto the Google Chrome store to be publicly downloaded by anyone around the world. Testing and bug-fixing is currently underway to optimize the reliability and maintainability of the product as much as possible. In the future, if there is a way to mitigate or disregard the small data limit of the “storage.sync” API call, cross-platform syncing would be an extremely beneficial feature, but at this time we are unsure if this would be possible using only chrome storage (a developer owned database may be necessary to accomplish this goal).

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>