TabbyCat

**Intro:**

If you’ve ever had dozens of tabs open in one window, you’ve likely realized how inefficient the user interface for tabs are. All of the tabs are crammed into a thin horizontal bar at the top of the screen, making navigation difficult and decreasing productivity. Despite having made browsers run much faster, one of its most commonly used yet limiting functionalities have yet to be fixed. What I aim to create is a chrome extension that provides a menu interface that makes managing and navigating to tabs more convenient. Through the exploration of different color keys and menu displays, my goal is to create a much better interface for the web. It’s important to me that through the efficient use of visual channels I keep the application intuitive and ergonomic.

**Problem Statement:**

How can I create a better interface for the web browser using effective visual channels?

**Survey of other solution ideas:**

Upon further research I found that many tab managers already existed on the chrome web store, each with their own unique methods of reducing tab clutter. For example, the extension Toby offers a menu that can group tabs into clusters for better organization. Another extension named Tabili adds functionality to save groups of tabs for reloading later. When reviewing these alternate solutions I found that their merits often came from added functionality. As a result these applications came with more complicated menus from the variety of options. After using a few of these extensions in my own free time, I found that these extra functionalities often were unnecessary. While they helped perform tasks for niche circumstances, I often didn’t need the extra buttons designating functionalities such as unique menu layouts or customizations.

**Metrics to measure success in the field:**

To ensure the application runs smoothly it is important that the application is responsive to both user input directly on the menu and also to tabs/windows being manipulated on the user interface. Different scenarios where tabs/windows are deleted, restored, or moved need to be accounted for to ensure that the extension is keeping good track of the tabs. To make sure the application is running reliably, its important to test app functionalities separately to ensure each method outputs what is expected of them, bug free.

Two competing qualities of a tab managing application are intuitiveness and complexity. While simpler applications may be more intuitive to users and provide a convenient user experience, it might lack the ability to handle more niche/demanding situations. In contrast an application with multiple functionalities may have many choices for the user, making complexity rise and decreasing intuitiveness.

**My design and its tradeoffs:**

The ideology of the application “TabbyCat” is to prioritize an intuitive user interface over more complex functionalities. Because having too many functionalities can be unnecessary and distracting when there are other tasks that need to be done, I plan on minimizing the number of buttons. The tab display is designed to be large and easy to access so the user can easily navigate his way through many tabs. The display of tabs will be color coded, so the tabs visited most often are of higher luminance than rarely used ones. This allows the user to at a glance easily distinguish between many tabs using color. The application will by nature lose in functionality to the toolkits that have many different methods, but in doing so promises to stay much more intuitive and friendly to the common user.

**My results and application of metrics:**

A major issue with my software development process was underestimating the amount of work that needed to be done. With no experience in JavaScript and CSS, it took a lot of time to learn and be comfortable coding what I needed for the project. This included also picking up React, a JavaScript library useful for building user interfaces. As a result, it took significantly longer to code the application and many unforeseen bugs/ struggles were discovered. I did throughout the coding test the methods through the console to try to maintain reliability. I also covered the different scenarios where tabs could be rearranged/recovered to make sure the application kept a consistent representation of the current tabs. The interface was kept large and easy for the user to utilize, with no other unnecessary buttons to create clutter. This resulted in a more intuitive process while sacrificing functionality. In the end, full functionality of the app was not completed by the deadline and will require additional time to implement due to the many bugs/struggles. This means that full color coding of the tabs are not in place. To implement, I need to resolve how D3, another JavaScript library can be implemented to make a better data visualization.

**Lessons Learned, Future Work:**

One lesson I learned is that software engineers really are optimistic that they can solve any problem. Working with unfamiliar languages invited many difficulties that slowed me down significantly more than I had expected. As a result the application still has functionalities that can be added to add to an intuitive tab menu for the user. Future work thus involves adding the visual channel of color to reinforce effective tab navigation.

Tab Managers researched:

Toby: <https://chrome.google.com/webstore/detail/toby-for-chrome/hddnkoipeenegfoeaoibdmnaalmgkpip?utm_source=chrome-app-launcher-info-dialog>

Tabili: <https://chrome.google.com/webstore/detail/tabli/igeehkedfibbnhbfponhjjplpkeomghi>

**SCHEDULE OF DELIVERABLES**

**(PLANNED BUT SENT BACK BY BUGS/LEARNING CURVE):**

Feb 27th

-Prototype that extracts necessary tab data for menu construction

March 19th

-Prototype with implementation of color coding

April 16th

-Extension with documentation / optimization / resolved bugs

REQUIREMENTS CONSIDERED:

- The application functions on Chrome without running into any errors

- When the Application is opened, all current tabs in all windows are initialized in the menu.

- Incognito mode does not influence or initialize any tabs on the menu.

- Moving tabs in a different order also reflects the same changes on the tab menu.

- Opening and closing a new tab reflect those changes in the menu.

- Dropping a tab to a new/old window doesn’t remove it from the tab menu and is reflected by its tab order.

- Clicking on a tab on the menu sends the user to that tab

- The tabs on the menu are represented with large easy to read rectangles with both the appropriate icon and name of the tab.

- A scroll feature ensures that regardless of the number of tabs, they will all fit on the menu.

**Test Results:**

The requirements of the application were tested by finding the outputs of different functionalities through the console. Through separate repeated testing of the requirements I found that the code did have some positive results:

– Tabs are correctly initialized upon application startup

- The manipulation of tabs are successfully reflected by the menu (dropping, moving, closing, etc)

-The menu can be utilized by the user to send the current tab to the desired one.

Although end color mapping functionality was not completed, further work for full implementation is planned to ensure the app functions properly.