TabFairy Final Report

A. Alkevicius, T. Almog, D. Hemphill & P. Patel
University of Massachusetts Lowell — Software Engineering II
Prof. James Daly

April 2021

The Problem

Our objective for TabFairy was to extend the Mozilla Firefox web browser with features that would allow users to keep tabs organized while surfing the internet. For many users, organizational challenges occur when they try to open too many tabs. The tab bar quickly becomes cluttered, and since there is only so much space in the tab bar, the tab titles become smaller and unreadable. Finding a needed tab becomes a challenging process. It becomes nearly impossible to achieve a productive workflow, especially if working on several projects that require working with many tabs.

One method to keep the tabs organized is to designate separate Firefox windows for different projects. The user might keep several windows running for the duration of the projects. While this solution is better than using just a single Firefox window, it adds the additional organizational challenge of switching between windows. It becomes even more difficult to transition between windows if multiple other programs are running simultaneously.

Current Practice

There are few approaches currently available in solving tab clutter in Firefox. One solution is for the user to implement their organizational methods, such as dedicating separate browser windows for different projects as described earlier. Another solution is to install a Firefox extension that provides methods to organize the tabs. There are a few dozen of them available on the Firefox Browser Add-ons site[1]. The most popular extensions are Panorama Tab Groups[2] and Tree Style Tab(TST).[3]

Panorama Tab Groups takes a unique approach by grouping tabs into visual containers, taking up an entire page to view. This allows the user to easily drag tabs into labelled boxes and clean up any clutter. It can also be used similar to the pop up windows that Chrome extensions employ. Panorama Tab Groups mixes a considerably large amount of features with an aesthetic beyond what the native browser can provide for organization. Despite this, the extension is lacking in unavoidable ways. In order to open a group, the user must click on a tab within said

[1] "Add-ons for Firefox (en-US) - Add-ons for ...." https://addons.mozilla.org/en-US/firefox/. Accessed 13 Apr. 2021.
[2] "Panorama Tab Groups – Get this Extension for Firefox (en-US)." 18 May. 2020, https://addons.mozilla.org/en-US/firefox/addon/panorama-tab-groups/. Accessed 1 Mar. 2021.
[3] "Tree Style Tab – Get this Extension for Firefox (en-US)." https://addons.mozilla.org/en-US/firefox/addon/tree-style-tab/. Accessed 1 Mar. 2021.

group while simply clicking on the group itself offers a popup to delete the group. This can become a problem if the user anticipates the question to be 'save this group' or 'open this group' which would have been more intuitive and logical, and clicks yes without reading. Allowing groups to be dragged and placed on top of another group is also an issue, as whole groups can be lost in the organization screen.

Tree Style Tab is another popular extension available in Firefox to organize tabs. The "tree tab" opens as a sidebar and allows you to create a hierarchy of tabs and nest them in a parent/child relationship. Multi-level nesting is also possible with this extension. On the surface, this is a more than adequate way to nest groups within the idea of a larger project and keep tabs saved in almost bookmark fashion. However, this extension comes with considerable drawbacks in use for the common user. For example, when creating a nested tab into a group, the options provided are not intuitive. "Create Child Tab," "Create Sibling Tab," and "Create Next Sibling Tab" are some of these options. While they are technically correct, it would be hard for an average user to understand what they mean. The options are placed under menus that seem unintuitive and awkwardly placed for an extension that boasts an easy way to organize tabs. The learning curve for using this product is considerable and intimidating.

Our Approach

This section covers the intended design of the TabFairy tool; we discuss the functionality we were able to implement over the course of the semester in the Results section. TabFairy is meant to merge the strengths of other extensions while avoiding their weaknesses. We planned to combine a graphical user interface (GUI) similar to TST with the Panorama View's tab grouping functionality. The goal was to develop a simple yet effective tab organizing tool.

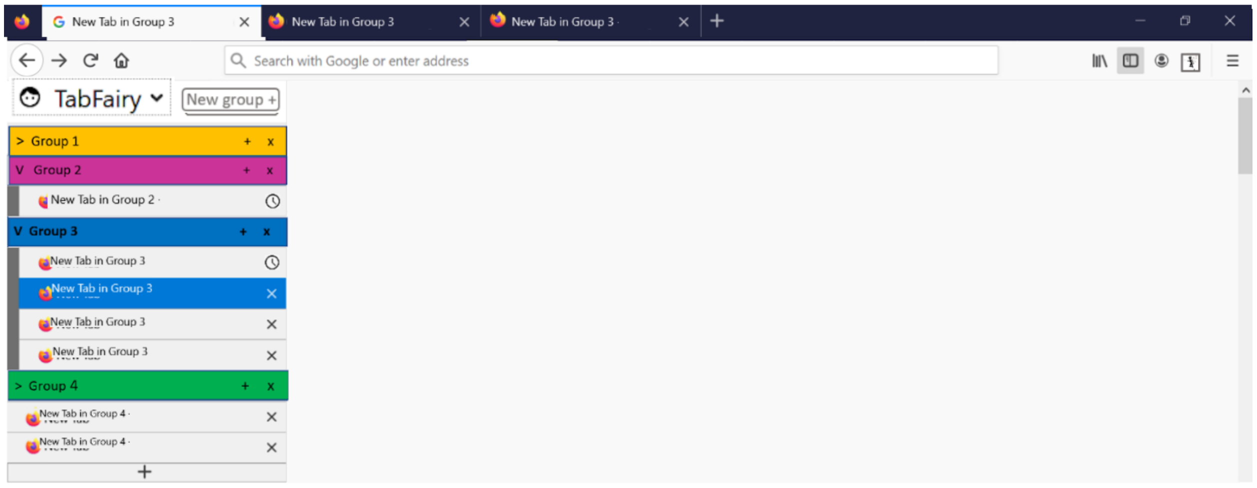

Figure 1. TabFairy GUI model.

Launching TabFairy reveals a minimizable sidebar that serves as the tool's GUI (Figure 1). In this sidebar, the user could create, delete, rename, collapse, expand groups. This sidebar would also enable the user to create, delete, rearrange, group, and regroup tabs. The groups would be color-coded and easily differentiable. One layer of nesting would be allowed to keep the organization simple. TabFairy would minimize the clutter in the tab bar by displaying only the tabs of the selected group. If an ungrouped tab is selected, then all other ungrouped tabs would be visible in the tab bar. All the groups and ungrouped tabs would remain visible in the sidebar. As mentioned above, the user would have an option to expand/collapse groups.

Based on our goals, TabFairy has achieved about half of our intended features. Users can create new tabs through the actual window and delete tabs through both the window and the sidebar. These tabs get tracked in TabFairy's UI and are labeled. Groups can be created and deleted, and are renamable from the UI. Unfortunately, we were not able to combine the two features during this semester.

Interface Design

The sidebar mock-up design is presented in Figure 2.

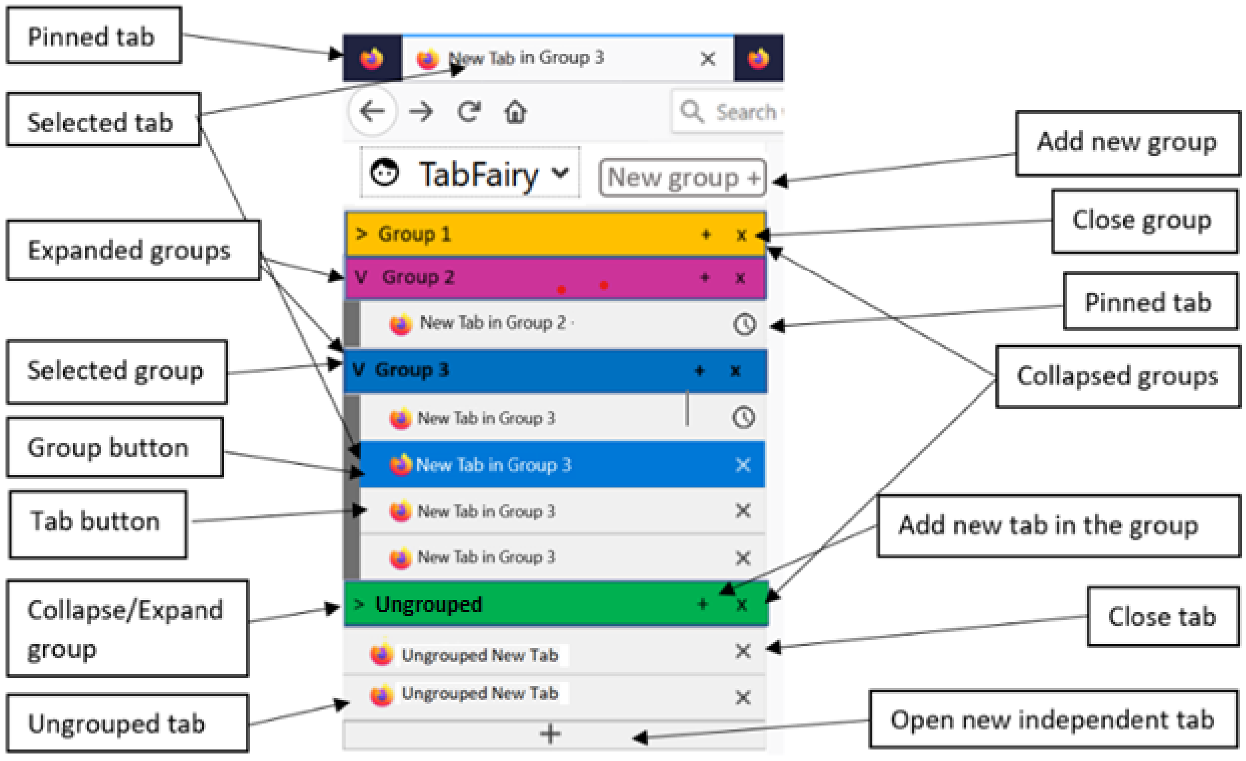

Figure 2. TabFairy sidebar.

Most of the default browser functions operate unchanged; however, there are some modifications in the Firefox GUI. The Use Case table (Table 1) and diagram (Figure 3) below detail intended user interactions with the TabFairy extension.

	Use Case	**Action**
1	Start initial TabFairy session	Click the TabFairy icon in the toolbar. Working tabs are populated in the sidebar as ungrouped tabs. Unless closed, TabFairy runs on startup subsequently.
2	Restart previous TabFairy session	Click the TabFairy icon in the toolbar. Tabs launched between sessions are populated in the sidebar as ungrouped tabs.
3	Restore TabFairy session	Relaunch Firefox and use Firefox commands to restore the previous Firefox session(s) then restart the previous TabFairy session (see 2).
4	Close TabFairy	Click the TabFairy icon in the toolbar; group information is saved with session info. Firefox resumes default tab organization. Working tabs are populated in the tab bar.
5	Create group	Click the *New group* + button in the sidebar. Automatically selects the group.
6	Create grouped tab	● Click the + icon in the group button. ● Select group (see 11) → Click the + icon in the tab bar.
7	Create ungrouped tab	● Click the + icon at the bottom of the sidebar. ● Click the + icon in the tab bar (Select ungrouped tab if applicable (see 12))
8	Close single tab	Unpin tab if applicable (see 13) ● Click the x icon on the tab or tab button. ● Right-click tab or tab button → click *Close Tab.*
9	Close multiple tabs	Unpin tabs if applicable (see 15). ● Right-click the tab: ○ Click *Close Multiple Tabs* → ■ Click *Close Tabs to the Right.* ■ Click *Close Other Tabs.*

		● Right-click the tab button: ○ Click *Close Multiple Tabs →* ■ Click *Close Tabs to the Top.* ■ Click *Close Tabs to the Bottom.* ■ Click *Close Other Tabs.* ● Close group (see 10).
10	Close group	Unpin tabs if applicable (see 15). Click the x icon on the group button → click *Close group* in the dialog.
11	Select group	● Click the group button or tab button. ● Close the only ungrouped tab (see 8). ● Create a new group (see 5).
12	Select ungrouped tab	Click on the tab or tab button.
13	Select grouped tab	Expand group if applicable (see 10 or 20). Click on the tab or tab button.
14	Pin tab	Right-click tab or tab button → Click *Pin tab.*
15	Unpin tab	Right-click tab or tab button → Click *Unpin tab.*
16	Transfer ungrouped tab to the group	Select tab (see 12) → Click and drag tab button to the desired group.
17	Transfer tab to another group	Select tab (see 13) → Click and drag tab button to the desired group
18	Transfer tab to another browser window	Select tab (see 12 and 13): 1. Drag tab button outside the sidebar. 2. Drag tab outside tab bar.
19	Collapse group	Click the *Collapse Group* arrow on the group button.
20	Expand group	Click the *Expand Group* arrow on the group button.
21	Rename group	Right-click group button → Type new name → Press Enter
22	Move sidebar to the right	Click Arrow in the TabFairy button → Click *Move Sidebar to Right*
23	Hide sidebar	Click Arrow in the TabFairy button → Click *Hide*

		Sidebar
24	Unhide sidebar	Click the TabFairy icon in the toolbar.

Table 1. TabFairy use cases.

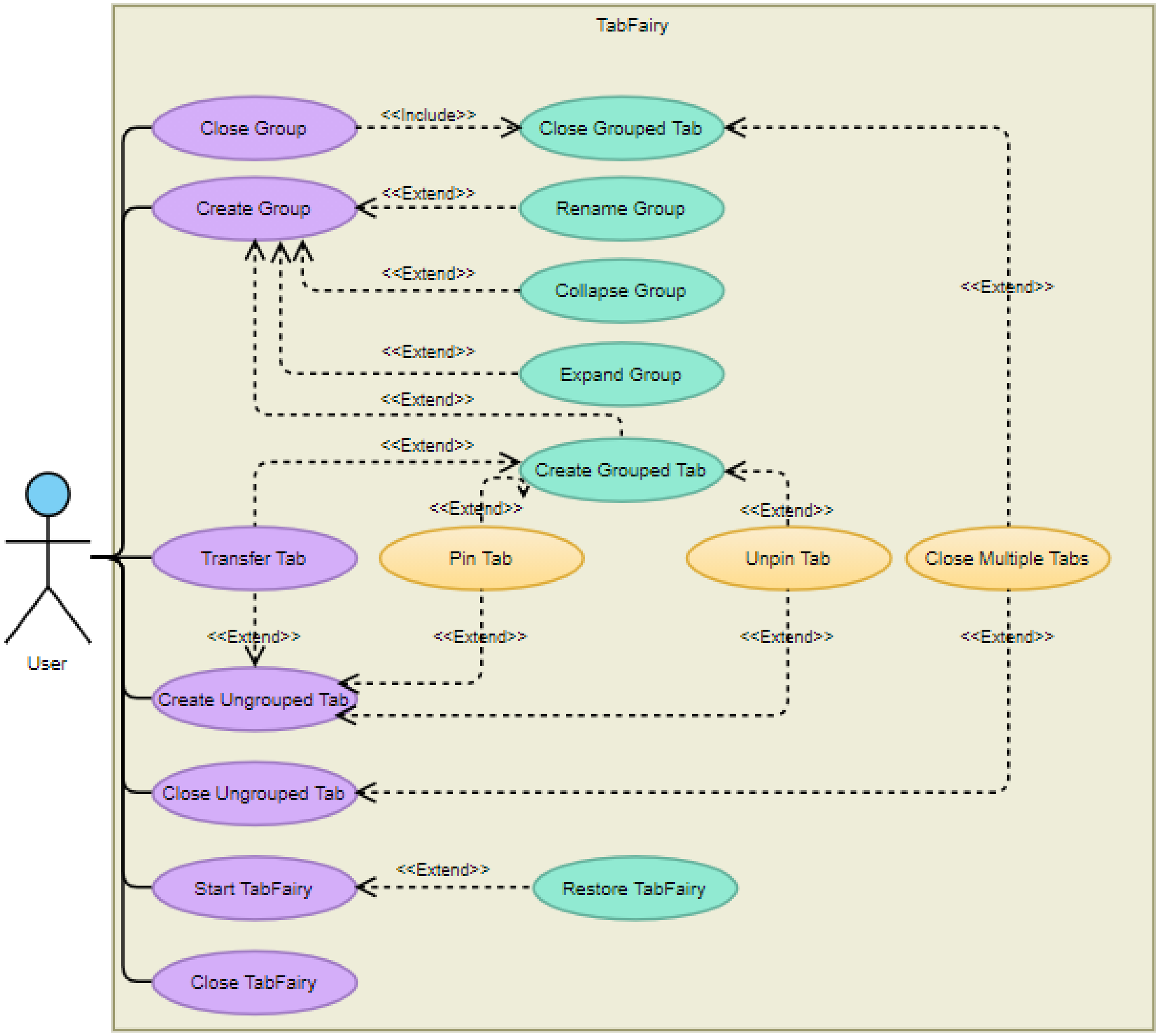

Figure 3. TabFairy use case diagram

Target Audience

TabFairy's target audience is primarily researchers, programmers, and students. Realistically, however, any tab collector can benefit from using this product. The connecting feature of these people is their need to switch between concurrent projects. For example, If a student has 4 classes and each class requires an average of 3 tabs, they would have 12 tabs to manage, each with its own category. TabFairy would help to organize those tabs into one easily accessible place.

Architecture

We developed TabFairy using Firefox's WebExtension APIs[4]. These included tools for creating and modifying sidebars,[5] limited manipulation of the tab bar[6], storage and retrieval of session information[7], and window event listeners[8]. These functions are contained in the Firefox component in the UML diagram (Figure 4).

We drew inspiration from competing Firefox extensions TST and Panorama Tab Groups, which each implement functionality that needed to be present in TabFairy-- TST has a similar sidebar layout and behavior, and Panorama allows the creation of groups as well as manipulating tab visibility in the tab bar-- but did not end up borrowing from their code, despite their weak copyleft licensing.

For this project, we designed a model-view-controller pattern. What this means is that instead of making the changes ourselves we rely on the Firefox client to do these changes for us. When a user clicks a button, TabFairy would send a message to Firefox, which will do the operation then send a confirmation, which will cause the TabFairy UI to show the change. While Firefox will be handling a majority of the operations, TabFairy builds on its API to add functionality.

The sidebar is the main component of our project. It serves as the main controller for all tabs and tab groups. It contains the layout for the entire UI and interfaces directly with the Firefox APIs. It contains all independent tabs and facilitates the transfer of tabs from groups or from independent tabs. The majority of the functions listed in the sidebar portion of the diagram below return nothing, but make changes to the UI and/or to the browser.

The tab groups act as containers; these contain groups of tabs and show or hide them based on which group is active. Each group is able to add or remove tabs to itself or move tabs between groups and independent tabs. Each individual tab is represented by tabID, allowing us to manipulate them with more ease. We did not end up being able to implement the tab groups during this semester.

[4] "WebExtensions - MozillaWiki." https://wiki.mozilla.org/WebExtensions. Accessed 1 Mar. 2021.
[5] "sidebarAction - Mozilla | MDN." 19 Feb. 2021, https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/sidebarAction. Accessed 1 Mar. 2021.
[6] "tabs - Mozilla | MDN." 9 Feb. 2021, https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/tabs. Accessed 1 Mar. 2021.
[7] "sessions - Mozilla | MDN." 1 Feb. 2021, https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/sessions. Accessed 1 Mar. 2021.
[8] "windows - Mozilla | MDN." 18 Dec. 2020, https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/windows. Accessed 1 Mar. 2021.

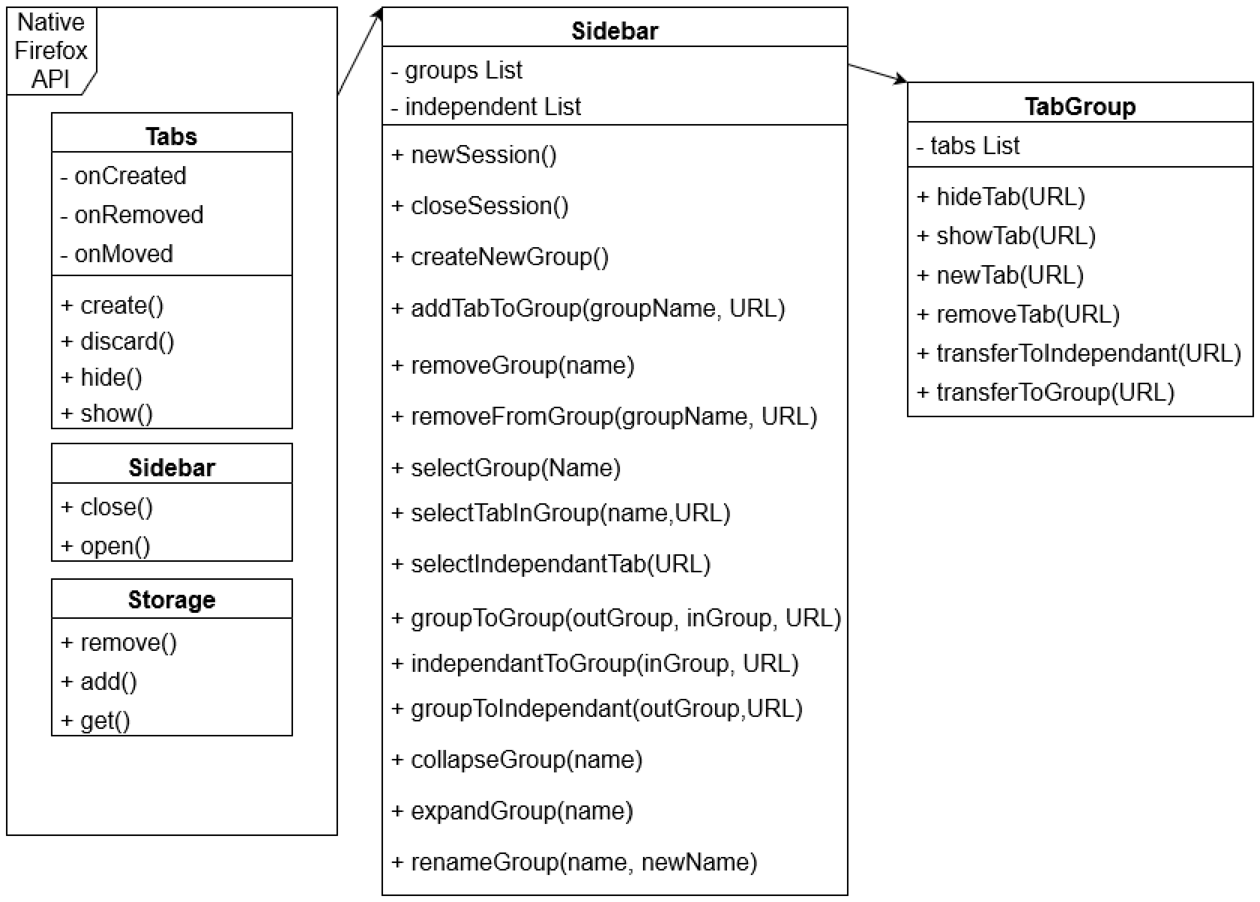

Figure 4. TabFairy UML diagram

Testing

We used Travis CI for continuous integration of building and testing. The project is built in a Node.js runtime environment and has fully functional automated test driving through Selenium and geckodriver. It used Jest as it's test framework to run unit tests. We wrote a few unit tests to check basic browser functionality such as navigation and tab creation but were not able to create automated tests using our extension APIs because no message passing was established between the browser and the extension.

Assessment of Risks and Challenges

The TabFairy tool is a free-to-use extension for the Firefox browser; this means that it competes with several other tab grouping extensions. Our product as designed has several advantages over our competition; however, it proved a difficult task to implement all of its features. During development, we met challenges using Firefox's WebExtension APIs. This resulted in the risk of not being able to develop all the functions that were proposed as improvements over the currently available extensions. Our risks, therefore, ended up not being limited to failing to implement non-critical features such as hiding tabs from the tab bar but also being unable to implement core functions. While features like creating a nice aesthetic are important, they were not at risk of disrupting TabFairy's functionality. In contrast, we were not able to manipulate the tabs the way we originally envisioned, and the implication of the API documentation resulted in an obstacle in the continuation of TabFairy's development. Ultimately, our team was inexperienced in extension development and the learning curve proved to be too steep in the allotted time.

Results

During development, we discovered that the scope of our project is much larger than we initially thought. In particular, the interface between Firefox and our extension turned out to be complex due to the browser's security mechanisms. Because of this issue, we were not able to modify the horizontal toolbar to create a higher bar for organizing the groups. Instead of calling the extension APIs directly, an infrastructure of message passing between background and content scripts provides the only means of communication between the extension and the native browser, which in turn requires familiarity with the Document Object Model (DOM) logical tree structure and corresponding APIs. These discoveries pushed the timescale of our project past the deadline.

Despite the setback, we have partially implemented some of the GUI modules of our intended interface design. The current sidebar implementation (Figure 5) allows the user to create as many new groups as needed. Each new group is assigned a randomly generated background color and appears at the bottom of the existing groups. The user can rename the group by clicking the default title. There is no limit on how many times the user can rename the group. The user can remove the group by clicking the *Remove* button. The sidebar can be relocated to the right side of the screen if needed. The sidebar can be closed by clicking the *x* icon. The ungrouped working tab titles are visible in the sidebar and these tabs can be removed by clicking the *Remove* button. Once the user closes the tab in the tab bar, the terminated tab is automatically removed from the sidebar. Similarly, if the user removes the tab from the sidebar, the tab will be automatically terminated in the tab bar. Although the user can create and delete the groups, the feature of transferring or grouping the tabs remains unimplemented at this time.

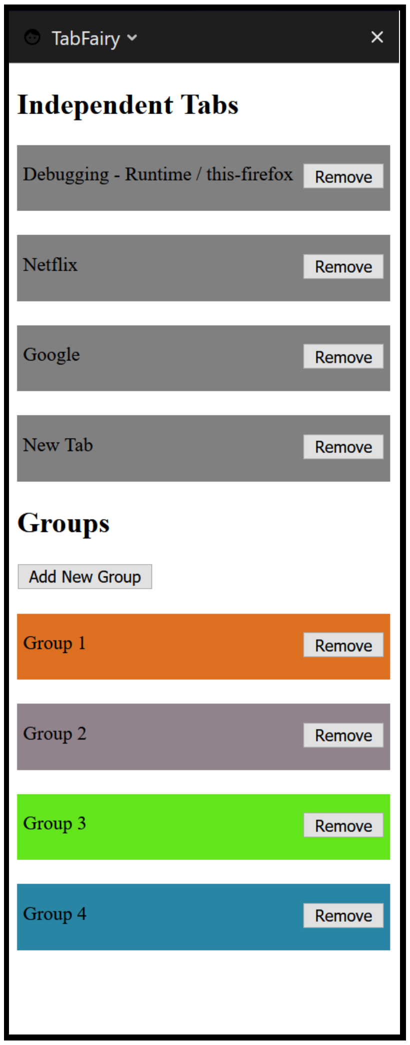

Figure 5. The current implementation of the TabFairy interface.

We were able to set up continuous integration for our project and succeeded in automating building and testing by linking Travis CI to our GitHub repository. Travis CI does this by loading a compressed version of the extension into Firefox through a script. It then runs a suite of unit tests using the Jest testing framework, driven by Selenium and WebDriver APIs, to test if the functionality was what we expected. If the package did not successfully build or failed one or more of the tests, we would be notified in our repository directly due to Travis CI integration. We also wrote some unit tests for basic browser functionality; our final test script includes a test that opens to the Google homepage and another test that opens a new tab and navigates to the website of our repository. The script cleans up after itself by creating a new browser instance for each test and erasing it upon completion. As noted in the Testing section, we weren't able to write tests for our extension because we did not establish the required message passing infrastructure between it and the native browser.

Future Work

We believe that the TabFairy extension has the potential to benefit Firefox users. Our next steps to developing TabFairy would be to take what we already have--the ability to track and delete existing tabs, as well as add and remove groups, and combine the two features. We plan to add drag and drop functionality to the sidebar so that users can easily move and organize tabs. Then we would be free to consider features like manipulating the windows to show only the tab group selected. Although the current state of TabFairy is unpolished and unfinished, I would consider this a good foundation for the overall objectives and goals that TabFairy aimed to accomplish. We estimate that another month of additional, uninterrupted work would be enough to reach our goals for TabFairy.

Appendices

A. Running TabFairy as a Temporary Installation in Firefox

To see TabFairy's development progress follow these steps:

1. Clone repository at https://github.com/tzuralmog/TabFairy.
2. In Firefox:
    a. In the address bar launch about:debugging page.
    b. Click "This Firefox" at the top left.
    c. Click "Load Temporary Add-on" at the top right.
    d. Select manifest.json file located in the webextensions folder of a cloned repository on your local machine.

TabFairy sidebar should now be visible at the left of the browser panel.
Our build history is available at https://travis-ci.com/github/tzuralmog/TabFairy/builds.

B. Team Member Responsibilities

Aurimas	Developer	Sidebar, Pinned Tab Functionality, Presentations
Tzur	Developer	Tab Bar, Presentations
Dan	QA Engineer	Repository, Testing, Presentations
Pooja	Developer	Sidebar, Pinned Tab Functionality, Presentations

C. Glossary

Extension	The program that can be installed into Mozilla Firefox in order to change the browser's functionality.
Graphical User Interface	Abbreviated GUI. A visual way of interacting with a computer using items such as windows, icons, and menus, used by most modern operating systems.
Headless Testing	Testing without rendering UI, can be done server-side very quickly
JSON Manifest	Provides basic metadata about the extension to Firefox.
Metadata	A set of data that describes and gives information about other data.
Mozilla Firefox	Firefox is a free and open-source web browser developed by the Mozilla Foundation and its subsidiary, the Mozilla Corporation.
Toolbar	A graphical control element on which on-screen buttons, icons, menus, or other input or output elements are placed.
TST	Tree Style Tabs; an existing extension available on the Add-ons for Firefox marketplace.