Library Resource Promotion
via Browser Extension

A Technical Report submitted to the Department of Computer Science

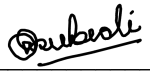
Presented to the Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science, School of Engineering

Ashish Upadhyaya
Spring 2020

Technical Project Team Members
Ashish Upadhyaya
Benjamin Ormond
Benjamin Spector
Nitesh Parajuli
Ryan Kelly
Tho Nguyen
Yukesh Sitoula

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Signature  Date 04/29/2020
Ashish Upadhyaya

Approved  Date 4/23/2020
Dr. Ahmed Ibrahim, Department of Computer Science

Table of Contents

Abstract

By developing a Google Chrome extension for the University of Virginia (UVA) library system, our team has worked to encourage users to easily and affordably access resources via the UVA Library, instead of purchasing them from common e-commerce and academic research platforms. Academic researchers and casual users of sites like Amazon, Barnes and Noble, and Google Scholar may find themselves paying for access to materials that they could just as easily access for free via the UVA Library, due to the convenience that these commercial sites provide. To enable users of these paid sites to more easily access free library resources, our team developed a Google Chrome extension which notifies users if desired content is available for free at the UVA Library or accessible in an online format. In addition, the extension provides functionality to access additional UVA Library resources, different search modes for a variety of content, and a recent search history. The extension provides an accessible system for casual and academic researchers to acquire free content and for library staff to publish resources for use by the general public. This new Google Chrome extension promoting UVA Library content is significant because it contributes to the free access of information online. This is important because the free access of knowledge through the internet improves education among the general public and lessens restrictions to information based upon wealth.

List of Figures

1. Introduction

The amount of knowledge present on planet earth is enormous, and it has never been more readily available than now - if you can afford it. From the advent of the internet through to today, more and more people are able to easily access more and more information and resources. One of the most prominent technologies in the world, the internet is actively used by 59 percent of the global population - a whopping 4.54 billion people (Clement, 2019). This unprecedented scope and scale has allowed the internet to revolutionize educational sectors for the better. It has opened the doorway to a wealth of information, knowledge, and resources, increasing opportunities for students to learn educational materials in and beyond the classroom ("Internet Access and Education", 2017).

While the educational benefits provided by the internet are immense, a large portion of the resources and information on the web is only readily available at a monetary cost to the user. As it happens, library systems, such as the UVA Library, often possess and freely offer these otherwise costly resources to patrons. In an effort to increase the usage of these often overlooked free resources, this technical project creates a Google Chrome browser extension for the UVA Library which informs students and staff about resources that can be obtained free-of-cost, serving to alleviate the potential financial toll of resource acquisition via online e-commerce sites.

1.1 Problem Statement

The UVA Library System provides the students and faculty of the University of Virginia access to millions of books, and other media materials through ten different library locations. Along with their physical presence, the UVA library has much of their content digitized and available online through the Virgo catalog search service.

One of the most significant problems facing library systems today is a decline in resource usage by the general population. A UK household survey conducted yearly has found that adult usage of public library services is at only 32.9%, and has been declining steadily over the past decades ("Taking Part Survey", 2020). Further, more than 55% of those who haven't taken advantage of library services and resources claim that they "don't need these services" ("Taking Part Survey", 2020).

Leadership of the UVA Library are currently working to solve this issue on a local level by providing access to resources through the Virgo search service, the UVA Library website, and through actions to increase accessibility like interlibrary loan services and a partnership with the Jefferson-Madison Regional Library system (JMRL). However, this approach is less effective than it could be for a number of reasons. The Virgo search service is effective in finding materials but may be less accessible than other potential solutions. When searching for resources, students are inclined to browse online research and e-commerce sites to obtain them, frequently failing to consider using the library at all.

Our team was assembled to create a product that would simultaneously promote available library resources and provide an easy-to-access shortcut to the preexisting Virgo search service. Taking the form of a Google Chrome browser extension, this product automatically searches the

UVA Library's Virgo catalog while users browse the internet, prominently displaying results that match their search terms. It also allows users to easily input their own manual search requests and view results at a moment's notice - all without navigating away from their current webpage. All of this functionality, appearing automatically and non-intrusively, works to alleviate issues of library usage - promoting resources to the masses.

1.2 Contributions

After two semesters of working together with representatives from the Library, we have successfully developed and delivered a fully functional Chrome extension. Our product works primarily to retrieve and display resources available in the UVA Library's Virgo catalog that match the user's search terms. When looking to acquire books, articles, or related media items, internet users often frequent popular e-commerce websites such as Amazon or Barnes and Noble. With the UVA Library Chrome Extension installed, users are shown similar items offered by the UVA Library while searching on their favorite platforms - Amazon, Barnes and Noble, and Google Scholar. Our completed extension works to increase library resource usage by eliminating the step where users directly visit the Library's online catalog to search for specific items. In addition, with recommended items non-intrusively displayed almost every time a user browses the sites mentioned above, the Library can implicitly promote the quantity and quality of its resources to a broader audience at very low cost. On the other hand, the UVA community also benefits by being educated about free resources that it has access to but may not previously have been aware of.

2. Related Work

Currently, the Library neither owns nor supports other web extensions besides the one our team developed. In initial client discussions, a roughly decade-old deprecated extension was mentioned as having been owned by the Library, serving the same purpose as the one we were to develop. No source code or information apart from memories remain of this previous extension, however. Presently, for a user to browse the Library's resources online, the UVA Library's Virgo web catalog is the only interface. However, there needs to be a more convenient and ubiquitous way to get data from the catalog database. Therefore, at the beginning of our project, our clients wanted a browser extension to extend ways potential users can be notified of available resources from the Library. The UVA Library Chrome Extension was developed to fulfill this requirement. Our work was inspired by the Amazon Chrome Extension, which returns Amazon listings that are similar to the user's current browsing and displays them to the user. Amazon's extension was custom written to serve solely their e-commerce website and has no generic template/API that can be overridden. Our extension replicates Amazon Chrome Extension's idea by fetching the user's search to Virgo API then retrieving and showing search results in an effective, non-intrusive UI.

3. System Design

As previously mentioned, the primary goal of our system is to promote the usage of UVA Library resources. This goal is twofold in nature, with system requirements to both automatically search the UVA Library's catalog while users browse the internet and to allow users to conveniently conduct their own searches of the Library's catalog. Our system's goals also led us to not differentiate between types of users, as everyone should be allowed to see recommendations for all library items. As such, all users should be able to directly and indirectly conduct Library catalog searches, view item details, place reservations, and alter extension settings to their tastes.

As our product was requested to take the form of a Google Chrome browser extension, our choices in programming languages were restricted to those allowable within the existing framework of these extensions. Javascript, coupled with HTML and CSS were chosen, as they are standard for these extensions. Our code is licensed under CC0, due to its inclusion in the UVA Library's code base which already has this license.

3.1 System Requirements

Gathering requirements is one of the most fundamental aspects of software engineering. It helps to build the project timeline and design the project structure. We learn and understand more about the customer's problem after gathering the system requirements. It provides an outline for what goals can be obtained and it can provide the steps needed to take in reaching the goal.

As such, we came up with a list of requirements for the extension after several client meetings.

All of the systems requirements are as follows:

Functional Requirements:

- As an academic researcher or casual reader, I'd like my browser to display related library-owned materials when I search commercial sites for books so that I can save money and time when doing research.
- As a casual reader, I would like to view a brief listing of the most relevant library resources and their availability in an additional popup, so that I know how I can quickly obtain the items.
- As a casual reader, I should be able to view UVA library resources based upon my internet search criteria on the Library Plugin Bar after submitting a search request, so that I can quickly see related library-owned materials.
- As an academic researcher or casual reader, I should be able to have access to the tool's functionality on Amazon, Barnes & Noble, and Google Scholar, so that I can save time when doing research.
- As an academic researcher or casual reader, I should be able to click on individual search results present within the Library Plugin and be redirected to their specific resource pages, so that I can quickly view more information about the items.
- As a casual reader or an academic researcher, I want to see the item's title, availability and link to it in the catalog in the Plugin Bar so that I can quickly know the availability of the item in the library.

Non-Functional Requirements:

- As an academic researcher, I should be able to have the Library Plugin Bar return the recommended items with near real time speed so that I can save time while researching.
- As a user with a disability, I should be able to access the Library Plugin Bar with ease according to the standards set by W3C's current accessibility standard WCAG 2.1, so that I have no issues utilising the plugin.
- As a casual reader or an academic researcher, I should be able to access the Library Plugin Bar without an overly intrusive user-interface so that my experience is as smooth and expedient as possible.

Future Enhancements:

- As a Casual Reader, I should be able to see what other university services have to offer through the Library Plugin Bar and potentially those of other participating schools, so that I could have greater access to free public media.
- As an Academic Researcher, I should be able to see the recommendation of books based on my history of searches through the Library Plugin Bar so that I may have easier access to related articles and books to improve my research.
- As a Casual Reader or Academic Researcher, I should be able to tell that the Library Plugin Bar is legitimate and associated with the University of Virginia's Library System, so that I know that the application won't misuse any potential data I provide it and will have accurate information related to library offerings.
- As an Academic Researcher, I should be able to have search results customized for me, based upon previous plugin clicks and usage history, so that I can have a more efficient and customized search experience.

3.2 Wireframes

To achieve the goal of non-intrusively suggesting items while delivering as much related content as possible, it is critical that our design layout was developed and revised iteratively and exhaustively. Wireframes are helpful in this case because they can be done in a shorter amount of time, require less effort and less technical backgrounds, and are easy to be redone if needed. Before beginning coding our system, we as a team worked on the design wireframes and showed them to our customers, retracted feedback from them, and revised them until the customers were satisfied. By using wireframes instead of an actual product, we saved a lot of time and possible frustrations of having to scratch off our codes and redo them, and were more open to suggestions or changes from customers or team members.

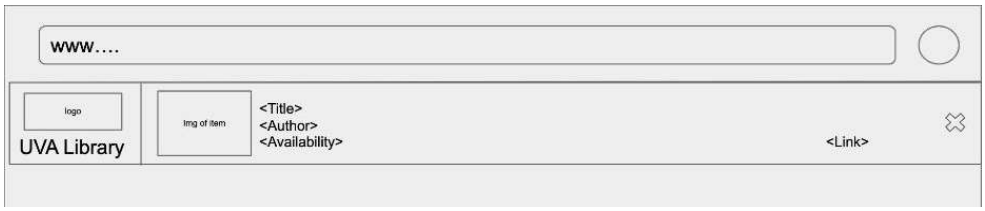
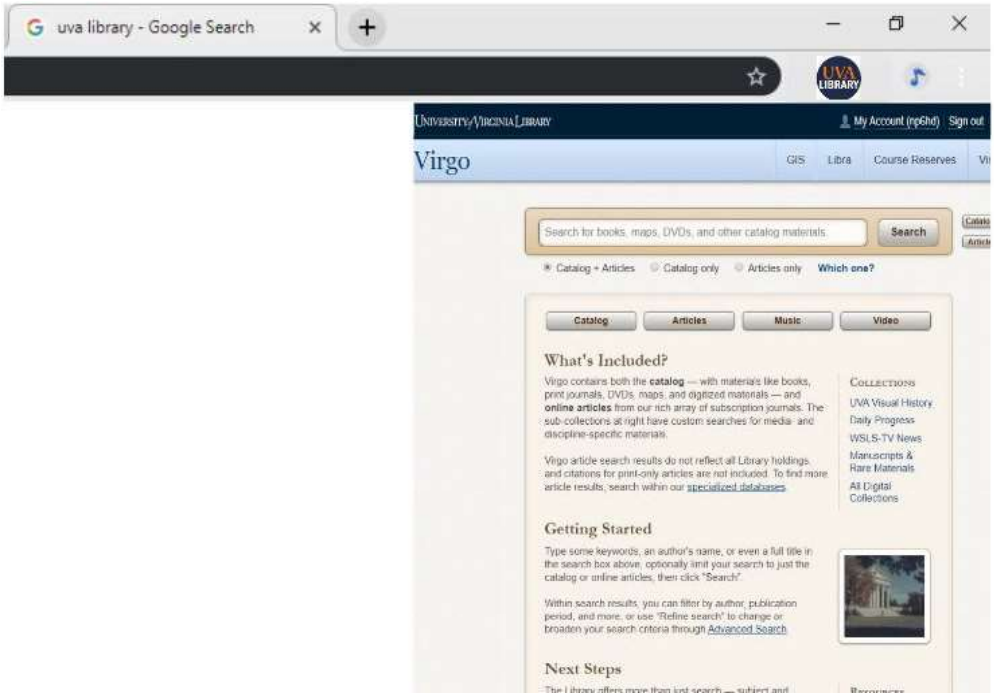


Figure 1. Initial Frame Design

Figure 2. Initial Popup Design



Possible design (when library extension is clicked in a browser) with the addition of the information in first slide.
The design will be smaller for the actual extension.

Figure 3. Alternate Popup Design

3.3 Sample Code

Obtaining Search Keyword:

```
function parseBarnesUrl(addr, url) {
  if (url.includes("/s/")) {
    keyword = document.querySelector("#searchBarBN").value;
    storeToSearchHistory({keyword: keyword, site: "Barnes & Noble", url:
url});
    return keyword;
  }
}
```

This code highlights how search keywords are obtained while users browse on specific websites. In this case, the Barnes and Noble webpage is being searched, and the keyword obtained using a querySelector manually set for the appropriate field. This keyword is saved to the extension's history, and then returned, being used as the search term in the Library's API.

Backend Search:

```
// First a POST request is sent to the Library's API
// Then the result is parsed such that the overall list
// has each item of the searched resource.
// the list is then sent to the front end to show the result
await fetch(
            "https://search-ws.internal.lib.virginia.edu/api/search",
            searchOptions
        )
        .then(response => {
            let formattedResponse = JSON.parse(response);
            fullList = []
            within "pool_results" of response {
                 within "group_list" {
                    newItem = []
                        within "record_list" {
                            within "fields" {
                                // addEachField of an item
                                newItem.push(field);
```

```
                        }
                    }
                    fullList.push(newItem);
                }
            }
            return fullList;
        }
```

This is the main part of the backend where the searching for resources within the UVA library happens. It does a POST request to the UVA library's API and gets a response back with the results based on the search. The result is then parsed, where an array of those resources are stored in the overall list. This list is then sent to the front-end, where the frame and then popup will use the different fields to showcase in the browser.

Front-End:

```
// Obtain relevant fields from local storage
chrome.storage.local.get(['id', 'title', 'author', 'cover_image',
                         'availability', 'library', 'callNumber'],
                         function(data) {
    // Creating link to Virgo Catalog using pulled 'id' field
    itemHref = "http://proxy01.its.virginia.edu/login?url=
               https://search.lib.virginia.edu/catalog/".concat(data.id);
    // Saving full and truncated item title
    short_title.innerHTML = data.title.substring(0, 25) + "...";
    full_title.innerHTML = data.title;
    coverImage.alt = data.title;
    // Assigning Virgo Catalog link to titles on-click
    short_title.href = itemHref;
    full_title.href = itemHref;
    // Saving item author information, truncating if needed
    if (data.author[0].length > 100)
        author.innerHTML = data.author[0].substring(0, 100) + "...";
    else {
        author.innerHTML = data.author[0];
    }
```

```
    // Saving item cover image information, replacing if needed
    coverImage.src = data.cover_image;
    if (data.title.length > 200)
        coverImage.src = '../images/cover_unavailable.png';
    // Saving item availability information, replacing if needed
    if (data['availability'] == undefined || data['availability'] == '')
        data['availability'] = "Available";
    else if (data['availability'] == "Online")
        status.innerHTML = "Available online";
    else if (data['availability'] == "Request") {
        leoRequest.href =
            "http://proxy01.its.virginia.edu/login?url=
            https://search.lib.virginia.edu/account_requests/"
            .concat(data.id).concat("/ill_leo");
        leoRequest.innerHTML = data['availability']
    } else
        status.innerHTML = data['availability'];
    if (data.library != "") status.innerHTML = status.innerHTML + " at ";
    lib.innerHTML = data.library;
    lib.href = (mapHref + data.library.toString().replace(" ", "+") +
"+Library+UVA");
    // Saving item call number information, if any
    if (data.callNumber != 'undefined' && data.callNumber != "")
        callnum.innerHTML = ". Call number: " + data.callNumber;
})
```

This is the main part of the front-end code, where fields are parsed after being obtained from the backend. It begins by requesting all of an item's fields from Chrome local storage. With all of the relevant information at hand, a variety of different variables are crafted, from truncated display names and availability status to Virgo catalog and Google maps links. Though somewhat long, this section of code is fairly straightforward to follow, largely consisting of crafting and assigning values to variables, which are then taken in and used by frontend HTML code.

3.4 Sample Tests

Software testing is an essential part of any growing system to ensure reliability, sustainable development, and ultimately maintain product quality to improve the consumer experience while allowing developers to monitor the state of their software.

Testing was particularly essential in making sure that backend searches were valid, with multiple different resources selected and searched. The QUnit framework was used to check against the backend's search result with the actual search results obtained from the UVA Library's Virgo catalog. Since the backend search is obtaining data from the Library's database similar to how the UVA Library's Virgo catalog obtains its data, the result should match.

Frame Testing Example:

```
//testing if href of item on showed on frame match href from search result
QUnit.test('test href', function (assert) {
    let actualHref = document.getElementById('full-title').href;
    let expectedHref= full_title.href;
    assert.equal(actualHref, expectedHref);
});
```

View More Testing Example:

```
//testing if href of item on showed on frame match href from search result
QUnit.test('test href', function (assert) {
    let actualHref = document.getElementById('title1').href;
    let expectedHref= title1.href;
    assert.equal(actualHref, expectedHref);
});
```

Settings Testing Example:

```
QUnit.test('Testing whether the global selection variable is changed to
video pool', function (assert) {
    changeSelection("video");
    assert.equal(selectionGlobal, "video");
});
```

Purchase Request Testing Example:

```
//testing that checkRequired returns true only when all required fields
are filled
QUnit.test('test form required contents not empty', function (assert) {
    if (checkRequired) {
        for (element in formElements) {
            if (element.required)
                assertNotEqual(element.value, "", element.id + " cannot be
empty.")
        }
    }
});
```

3.5 Code Coverage

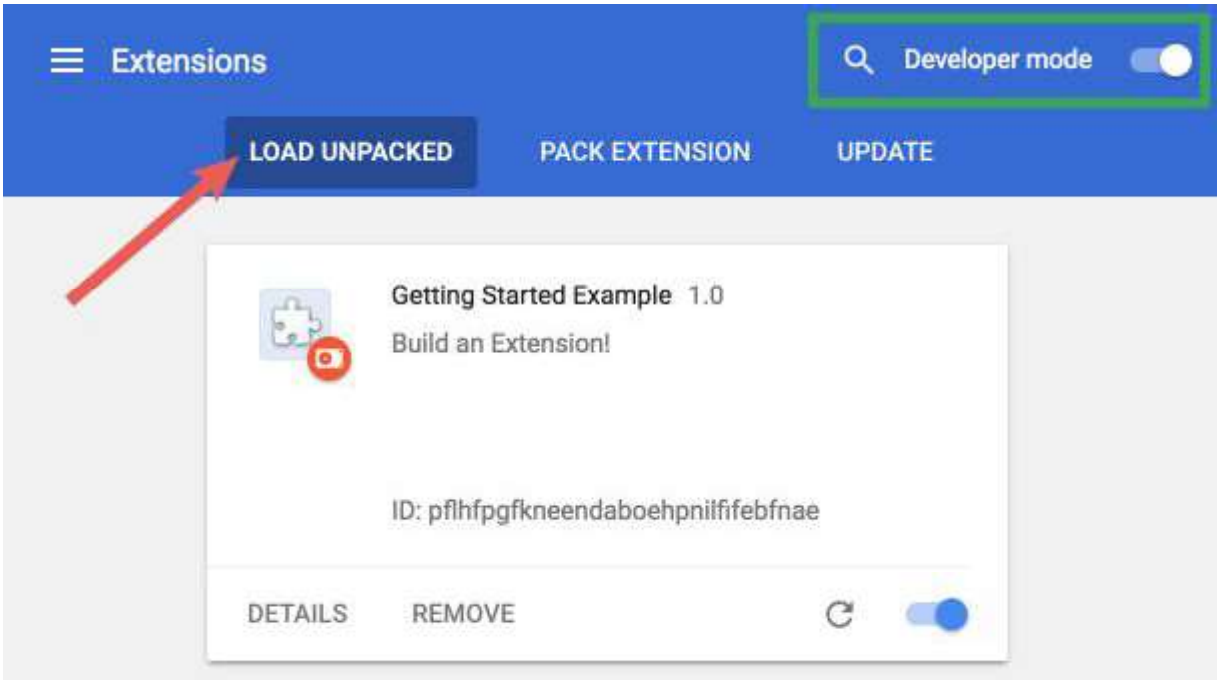
In the case of our project, code coverage wasn't pursued due to the nature of the project itself. After the creation of our automated testing tools and CI/CD pipeline, the addition and integration of a code coverage tool would have proven to be too large of a change for the size of the project given the deadline for the final deliverable. However, given additional time or the continuation of the project, our team would recommend the usage of a code coverage tool like

Istanbul.js or with another automated testing framework that came with code coverage tools. The use of Phantom.js as a headless version of chrome where we could run our tests would have made the addition of a code coverage tool increasingly difficult beyond the scope of the project. The ever changing Virgo 4 API, that our team did not develop but relied upon, also meant that test cases were prone to failure when they shouldn't so attempts at complete code coverage and testing for the project were often more of a hindrance than of use.

3.6 Installation Instructions

How to Run Locally, Directly from the Source Code:

1. Clone the Github Repo https://github.com/uva-cp-1920/UVa_Library to your Machine or Download and extract the zip file of the Repo to your desktop.
2. Open the Extension Management page by navigating to chrome://extensions in Google Chrome.
    a. The Extension Management page can also be opened by clicking on the Chrome '• • •' menu, hovering over More Tools then selecting Extensions.
3. Enable Developer Mode by clicking the toggle switch next to Developer mode.
4. Click the LOAD UNPACKED button and select the extension's "src" directory.



How to Publish the Extension to the Chrome Web Store:

1. Follow the tutorial at https://developer.chrome.com/webstore/publish

How to Obtain the Extension from the Chrome Web Store:

1. Navigate to https://chrome.google.com/webstore/category/extensions
2. Search for the extension's title and/or keywords in the "Search the Store" bar at the left of the screen.
3. Find the corresponding extension on the screen (clicking "More extensions" if necessary), and then click the "Add to Chrome" button associated with it.
4. Accept the extension's permissions in the popup, by clicking the white "Add extension" button.

4. Results

By creating a system through which free UVA library content is accessible on sites like Amazon.com and Google Scholar, the problem has been successfully addressed. In less than 10 seconds, users are able to search for content on one of the other sites and then have potential matches shown to them either through a bar at the top of the screen or through an icon from their list of Google Chrome browser extensions. Before the addition of a Google Chrome browser extension, users may have never even considered that some of the content found on Amazon would be available from the library. The UVA Library extension meets all of the requirements for the system gathered from the client and even includes some of the extra future enhancement requirements like library services and a working user search history. The UVA Library extension also complies with the W3C accessible use standards, and acts with near real time speed when searching and accessing items by link.

All of the stakeholders in the current system will benefit from the additions made by the UVA Library Google Chrome Extension. The users of the UVA Library extension can easily search for all types of media using different mode selections and be rapidly returned a list of potential results viewable in both the top bar and the tool bar. This will allow casual users and dedicated academic researchers alike to find new content. In addition to traditional media forms offered by the UVA library, library services similar to the item searched for will be returned with a link to the service. This allows librarians to advertise access to services other than just books and allows users to find out about new library services. The multi-search ability to search for different types of content also enables all of the stakeholders to benefit from types of content that

align to their needs. More casual users would likely benefit more from the digitized movie assets from the library whereas academic researchers would likely benefit from the rare books and archival search modes.

5. Conclusions

By creating an application which allows casual and academic researchers accessible and free access to library resources, our team took a significant step towards the freedom of information online. The non-intrusive nature of our software solution that subtly points users toward free library resources allows the browser extension to improve the everyday web browsing experience for any member of the UVA community. By displaying the availability of library resources alongside regular online product browsing, the extension increases the overall visibility of the library. More broadly, users will become more cognizant of the breadth of library resources available to them, leading to increased utilization of valuable services that improve academic research and performance at the university.

6. Future Work

The timing of our project left several areas open to further exploration and development. The most troublesome of these areas was the API, specifically because it would change drastically as we developed our product, forcing us to make major changes in our backend. This instability was due to the API actually being in a production phase, with the library still making changes to it. As it still has yet to be finalized, more changes in the API may require corresponding backend modifications in our extension. The other potential area of future improvement is in the sending of data from the content file to the popup and frame. In order to deliver a functional application within the timeframe allotted, we opted for a manual use of Google Chrome's local storage. A more elegant solution would require more research into a data transfer framework, which has a difficult learning curve. Fortunately, as work on the API has begun to wrap up, neither of these present a major issue at the present moment. Our stakeholders are also aware of our API and local storage dependencies, and should be able to ensure that no issues arise.

7. References

Clement, J. (2020, February 3). Worldwide digital population as of January 2020. Retrieved from https://www.statista.com/statistics/617136/digital-population-worldwide/

Internet Access and Education: Key considerations for policy makes. (2018, November 20). In *Internet Society.* Retrieved from https://www.internetsociety.org/resources/doc/2017/internet-access-and-education/

Pyle, E. (2020, March). Taking part survey: England adult report. Retrieved from https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/873111/Taking_Part_Survey_Adult_Report_2018_19_-_March_2020.pdf