Final Design Document

Senior Design Fall 2024-Spring 2025 Team G05

The Charles Brockden Brown Archive

Gael A., Rodrigo A.S., Bernardo Miguel J. G., Kiara L., Dawn M.

Sponsored by:

Dr. A. Giroux & Dr. B. Miller

Table of Contents

Executive Summary

Charles Brockden Brown is one of the earliest ambitious and accomplished writers from America, being born in 1771 and dying in 1810. Throughout his literary career, his writing spanned across many genres and different materials to write in. He was revered as an author due to the fact that he wrote about and took part in debates surrounding the culture and politics of the new founded nation of the United States of America. One specific topic he was well known for was talking about women's rights. Since his works were so prestigious, having an archive that is able to preserve these works is important to a lot of people.

Our job is to take the current website of the Charles Brockden Brown Archive, and make it become a more modern version of itself in accordance with the current accessibility standards. This will allow the website to be better for everyone to access the archive, along with making more people able to fully use it. The mobile version of the website will also get a revamp, as in its current state it is nearly unusable. Another part of the project is to update the archive to Red Hat 9, from Red Hat 7. This will be to update various outdated packages, have tighter securities, and get more support for future work on the website. Along the lines of updating the backend to be more modern, we will also be migrating all of the files into the Microsoft Azure Cloud, along with setting up instructions in order to transfer everything to STARS - UCF's own digital repository.

Introduction

Introduction - Gael

The Project - Gael

Our project is a renewal of the website for the Charles Brockden Brown Archive. This Archive is a website where users can search for and browse through the writings of American author Charles Brockden Brown. The site is currently hosted on an infrastructure based on Red Hat Enterprise Linux 7. Its frontend is mostly coded in PHP, and its search engine accesses the documents stored on the database, most of which are formatted as XML files.

The importance of Brockden Brown's contributions to the world of literature would be enough to justify the importance of this project—his stories would "exploit horror and terror" in a manner that would later be reused by the great Edgar Allan Poe (*Britannica*). The preservation of literature is a commendable mission, and those who maintain the Archive have done an excellent job of this. However, the website itself is another reason why our project is significant.

The architecture upon which the website rests has aged rather poorly over the last 10 years. The site is built on an older version of Red Hat and has many bugs that we found in our research. Our new version of the frontend will fix many of these and make the experience of browsing the Archive much more user-friendly. One of our most important goals is to make the website more accessible; this will ensure that more and more people will be able to enjoy the writings of Brockden Brown, as well as many references that give context and meaning to the documents presented.

The owners of the website asked us to preserve the website's current looks and aesthetics. Because of this, most of our work will be focused on improving what's already there, rather than introducing new features. While we will strive to keep the site looking similar to the way it does now, we will inevitably need to change some parts of the layout to make it more accessible.

The technical details of how we will implement our project will be discussed later in this document. The project is compact in scope. It happens that the upgrade we're performing is more about stability than new features: we're really making a simpler, less broken version of the current site. We're painstakingly recreating the aesthetics of the current version and combining them with a fresh undercoat of React, PHP, and RHEL 9.

Societal Impacts - Gael

Impact on Users with Disabilities

The first is the impact that our accessibility measures will have on people with disabilities, especially those who use screen readers to navigate the Web. Right now, the site isn't very easy to use visually. The Search function requires a few clicks through the menu, and the way the Search Page is set up is suboptimal. Add to that, the Document view page is not very polished in the way it displays images of the document and then transcribed text. It makes it very difficult to navigate the page, and makes the documents less readable for users who are blind and low-vision.

By implementing the WCAG AA accessibility standards using the methods described in this document, the website will become a more welcoming place for all users. Those who currently find the website difficult

to navigate will have a much easier time searching through the Archive. They will also find the Document view much easier and more convenient, with adjustments that will help them read through the documents better and faster.

Impact on All Students and Educators

The other important impact of the project is on education. For educators and students around the world, it is essential to have access to high-quality preservations of important literature. Brockden Brown's writings had an enormous impact on the landscape of American writing, and it is important for those teaching the history of American literature to understand this by reading through his literature for themselves. Without the Archive, it would be much more difficult to find complete, easily accessible copies of Brockden Brown's works.

Students will also benefit from this project by having easier access to an important resource for their learning. The Archive's large collection is a great resource for those seeking to learn more about Brockden Brown himself or about the America and the world he lived in.

Considerations on Legality, Ethics, and Privacy - Gael

We have thankfully not run into many issues regarding our project under this topic. However, it is still worth discussing each aspect of it.

Legality

Any project aiming to preserve media is bound to run into copyright challenges. However, it seems that all of these were settled long before we took on our project. The Archive seems to be safe when it comes to storing and displaying Brockden Brown's literature, and the site itself is protected under copyright.

Our version of the frontend will similarly be protected under copyright laws, so we are not worried about legal issues. The technologies we will be using are all being used under their appropriate licenses, most of them being free-and-open-source (FOSS). Additionally, we of course will not be making any money off this project, so there's no concern about us using any of the technologies mentioned in this document "for profit".

Ethics

A project like this has few ethical shortcomings on the surface. However, one that we noted was the lack of documentation on the original version of the project. While we have been given access to the frontend PHP files and the document XML files, we have no access to any explanatory documents on how the site and its underlying code were built. Because of this, one of the important parts of our project is to properly document our work so that future developers on the Archive will have an easier time understanding the changes we have made.

The documentation process will definitely include instructions related to the process of migrating from Red Hat Enterprise Linux 7 to Red Hat Enterprise Linux 9. This move, thankfully, should be fairly simple. Additionally, there should also be instructions included detailing the migrations of data to exist-db and to STARS.

Privacy

As discussed later, the website also has very few issues when it comes to user privacy. We only found two major issues. The first was involved in the Bookbag function, which is also described later. This function had some serious flaws when it came to a user entering personal information. However, as will be discussed in this document, the Bookbag feature will be dropped entirely. The only other privacy-related issue the Sponsors brought up was the need for the website to become GDPR-compliant. This won't be a problem, however, since the current version of the website doesn't have invasive cookies in the first place!

Motivations

Bernardo

My main motivation to select this project was the fact that I was very interested by the technology stack, as well as the overall idea of the project. I am a very big Linux enthusiast and I have always been interested in doing a project that will help me apply the skills I use in my job to school somehow. On top of that, I am currently studying for my RHCSA (Red Hat Certified System Administrator) and I believe that through this project I will

be able to develop parts of my Linux knowledge that I haven't been able to develop through work or personal projects, which will be extremely helpful when I take the certification test.

I have also been a very big enthusiast of literature and history, but I haven't had as much as contact with the American literature as I would like, mostly due to the language barrier that I encountered in my first couple of years in the United States. I believe that through this project I will naturally get closer to the American literature and historical world and I'm very interested by that possibility. I also believe that by achieving the desired functionality and design of this project I can bring other people closer to the literary world of Charles Brockden Brown as the archives will become more attractive to people.

Another thing that really interests me aside from the backend's redesign portion of the project (RHEL 7 to RHEL 9) is the fact that we will work directly with PHP, which I really enjoy although I haven't worked much with it. During the Processes for Object-Oriented Software Development class that I took on last year's summer, we had a project that utilized PHP and I ended up enjoying it a lot, so I am very excited to further develop my skills on that. On top of that, my work's software is mostly made in PHP and JavaScript, which is essentially the project's tech stack, so I believe this will provide me skills that will not only help me in this project, but also help me grow in my career, as it will be easier for me to understand the base code of the software I work with.

Gael

Initial Motivations

My main motivation for choosing this project is that it uses a tech stack I'm somewhat familiar with—namely, Apache Tomcat and the Ubuntu operating system. Thanks to my coursework as a Computer Science student, I've been able to experiment in my projects with these two technologies. Another major motivation of mine is to learn these and the other technologies involved at a deeper level. I'm excited to learn about the worlds of Red Hat Linux and PHP to achieve the project's requirements.

Finally, the most important motivation for me to participate fully in this project is to earn real-world work experience as a programmer. So far, most of the projects in my Computer Science coursework involve simple programs that are only run inside an IDE or from a command-line. Having a final product that's an actual, public website will work very well for job interviews—the same can't be said about all the command-line applications we had worked on in my previous programming classes. This project is a great opportunity for me to expand my horizons and contribute to a real web-site and, at the same time, boost the literary preservation community.

After considering the project's requirements and specifications, many ideas have come to mind for how I could contribute. For one, the requirement for a GDPR cookie warning seems very straightforward, and I think I'd like to implement it using language borrowed directly from the European Union's guidelines on internet privacy.

Another idea I've had is to lead the creation of the documentation for installation on Red Hat Enterprise Linux 9. I'm rather comfortable with word

processor software, and I've got a taste for writing instructions—I'm sure it'll be a breeze to write that guide.

Additionally, at first I thought it would be fantastic if I were able to work on the search function for the Library of Congress headings. This is clearly a refined and well-defined system, so it sounded like a lot of fun figuring out how a user might search using such index headings.

Later Considerations

One last thought I've had is to work on the web-site's Section 508 compliance–this means to make the archived texts accessible to those who use devices such as screen readers. This is especially useful when the archived text is an "image" PDF, meaning that, without alt-text, there's no way for someone using a screen reader to actually read the document. My idea is to implement this by also taking inspiration from international guidelines on accessibility rather than limiting myself to the letter of the United States law. I feel that'll be hugely beneficial for readers around the world.

It also happens that this is my first time working with most of our technologies—exist-db, for example, is completely new to me. The same goes for PHP and the XML format that our documents are saved in.

This project has really motivated me to re-analyze my time management skills as well. My teammates and I decided that I would be Project Manager, and that made me realize that now I'm not only managing my own time, but I'm also responsible for supporting my team and ensuring they have the help they need in managing their own time, as well. While Bernardo has been in charge of managing our Jira backlog, it's been up to

me to make sure everyone is on track to actually complete our goals listed on the backlog and to support everyone through the Sprints successfully.

I consider Senior Design to be the culmination of my years of work in college and university: all these years I've been learning more and more skills in both programming and project management. My Game Design and POOSD courses, for example, were great examples for me on how to work as a team and meet programming deadlines. I'm excited to see how this project will turn out and the work we'll be presenting next semester.

Dawn

In ranking the 25 projects for this class, this took the number one spot. I felt drawn to this project for a number of reasons. Despite not being entirely clear on the specifics of the project, I had been magnetized by the thought of being able to work on, and make more accessible, something similar to a library where information is being hosted for the benefit of others really resonated with me.

Not only that, but the systems at use are of interest to me, some things I've worked somewhat with before, but I'd like to increase my proficiency in those areas, and I think this project is a really great opportunity for that. I love Linux, and I'm generally interested in web development. I've worked on my own personal website for fun, so beyond HTML/CSS I've been marginally exposed to some Javascript and PHP. What I've done there was unplanned and not in-depth, but this project presents a much more structured path.

I had some experience with Processes for Object Oriented Software Development (POOSD) with Dr. Leinecker on web development stacks like LAMP and MERT for our projects. In that time, I worked on both the frontend and backend, as well as web and mobile development, which may be similar to this project. However in that class's case, the two projects we worked on were in a much shorter time frame. Furthermore, while I did enjoy my time in that class, the coordination within my group in particular on the second project was not ideal. That's something that must be avoided here.

Beyond fixing up the backend and upgrading existing systems, the archive's website could use an upgrade in layout and readability. It is clear to anyone visiting the website that it has not been updated in a long time, in fact it says so on the page. The website was also not created with a mobile experience in mind. I would like to address these issues. And I feel there is a wide potential on what I can work on for this project, perhaps whatever works best for the team.

Kiara

One of my motivations for choosing this project was the fact that it was dealing with just a website, and not being its own piece of software. There are two reasons why that appealed to me specifically. The first is that throughout all of my classes here at college, I've never had to work on a website. I've always wanted to learn and work on websites, but haven't had the motivation to do it on my own, so having to do it will ensure I learn how. I also have experience working with Ubuntu, which is a nice head start. The other reason is that this project seems that it will be completely manageable and easy to stay focused on it. Some of the other projects seemed like it might've been a bit much, especially if next to no one else had any

experience, while this project seems to be 100% doable. It also has a clear high level idea on what needs to be done, so that straying off on a possibility it might help won't cross my mind.

As someone who frequently uses my tablet in order to just look at random websites, I really want to help contribute to making the site more mobile friendly. I also think I can help with the overall look of the website. Especially with a topic the average person might not give a second glance at, having the website look aesthetically pleasing will help draw and keep people in. I am one of those people where if given the opportunity to research a topic, a historical figure, centering around literature, would probably be near the bottom, or even at the bottom. However, like I said, I still like to just browse random wiki articles and random websites, meaning I might end up on one like that. So, I want to bring this website to a place where others like me would end up on it, and stick around. This would entail helping with frontend elements - the dropdown menus, anything to be done with the pictures (like the zoom in features, and they are placed awkwardly depending on the platform so making that streamlined), and also helping out with the search functionality.

Another reason why I decided to work on this project is not specifically a reason that comes from what I want to do, but rather the passion behind the project. Just from the initial presentation given from Dr. Giroux, it was clear that the people who were running the project in the background cared deeply about Charles Brockden Brown, his works, and his contributions to literature. The website clearly shows its age, which just makes all of this more impressive. This isn't just a guy that has recently gained notoriety for some reason and they are jumping on a bandwagon, this is truly work that those who want it deeply care about. To me, knowing that this passion was

running through the website gives me more of a push to work on it and make it the best it can be - it’s not just another thing that has to be done, it’s something that actual, real people will look at. These people want to have a website that they can actually use to its full potential, and allow it to be a page worthy of not only upholding Charles Brockden Brown’s legacy, but also one that they are happy with other people stumbling upon to learn more about him.

Lastly, when viewing the projects, a couple - including this one - stood out to me as someone whose brother has previously been in the education field. He was a teacher at a public school. This means that having widespread, public access is something that is a big part of my beliefs. Taking this project on will allow me to contribute to a website that will have a more broad, widespread influence on the masses.

Rodrigo

My main motivation for choosing this project is that it reminded me of what I have done for CCDC (National Collegiate Cyber Defense Competition) practice since the Fall semester of 2023. I joined UCF’s Cybersecurity competition team (C3) last Fall semester, and our main competition is CCDC. The main idea of the competition revolves around system administration, system hardening, and fulfilling business needs. To practice defending an enterprise network, which is a major part of the competition, the team and I would practice configuring various services to then harden and maintain them. I have configured and maintained web servers on Windows countless times, usually using XAMPP, and loading WordPress, MediaWiki, or a BookStore project the team found on GitHub. Apart from configuring those

services from scratch multiple times during practice, I have also experienced maintaining them in the competitions themselves. Web Servers are a common part of a business environment and are often given to us to manage. I have managed an HR specific web application (OrangeHRM), a custom made Streaming Website, a management website among others. I personally enjoy setting up the various services and making sure they work, having to spend countless hours on troubleshooting. With how common Web Servers are in practice and competition, I have also taken a special liking to working with them. Working on a web server is something I am comfortable with and wish to learn more about.

I am also looking forward to getting more hands-on Linux practice. My function in UCF's CCDC competition team is limited to Windows administration. I spend countless hours practicing managing different services on Windows, but I lack the same practice and expertise on the Linux operating system. Working with Linux is not my specialty. Although that is the case, I have done classes and configured services on Linux countless times before, so I do know my way around it, however, not to the extent that I would want to. I look forward to setting up the services related to this project on a Linux distribution I am not the most familiar with (RHEL). I look forward to spinning up various RHEL virtual machines to do testing relating to this project. It is also possible that we will be utilizing containers. This will give me the opportunity to practice and interact with this technology that I am not the most familiar with.

Other than my previous experience with managing web servers, another potential area where I can add value relates to automation. For this Fall semester of 2024, I helped host a CyberSecurity competition, akin to the ones I competed on, for the benefit of UCF students and Hack@UCF

members. In this process, I was part of the Black Team. This team is responsible for arranging the infrastructure of the competition, as well as designing and configuring the various virtual machines and environment that will be used by the competitors. In this process, I spent a lot of time using Ansible to automate virtual machine deployment. Ansible is a software suite that empowers infrastructure as code. Leveraging that technology, I wrote code that would be responsible for configuring a Web Server on Windows among other services. With this baseline of experience, I could attempt to automate this project's Web-Server provisioning with Ansible as a stretch goal. Although the bulk of my experience lies in Windows and using Ansible to automate deployment in Windows systems, I believe I can use that experience to facilitate learning and the implementation when attempting to emulate it in a Linux distribution. I believe this will take a lot of work, as I predict our project will be more complex than the machine I previously designed, but I am confident that with enough time, I would be able to make it happen. I would have to discuss with our project sponsors on whether this is something they are interested in. I presume we will have various stretch goals and prioritizing will be a necessity. I know our sponsors are interested in the creation of documentation to explain the deployment of our project, so it might be helpful to generate code that can be easily run to deploy it.

I am very excited to learn new programming languages and technologies. Despite my experience in managing PHP based applications in various competitions, I am not very versed in PHP, or JavaScript. I am confident that learning more about those two technologies will be very helpful for my future career. I have also never used React or interacted with exist-DB. Those are two technologies that I will have to learn from scratch. I look forward to learning different programming languages and different technologies as I work on this project. I also look forward to working with a

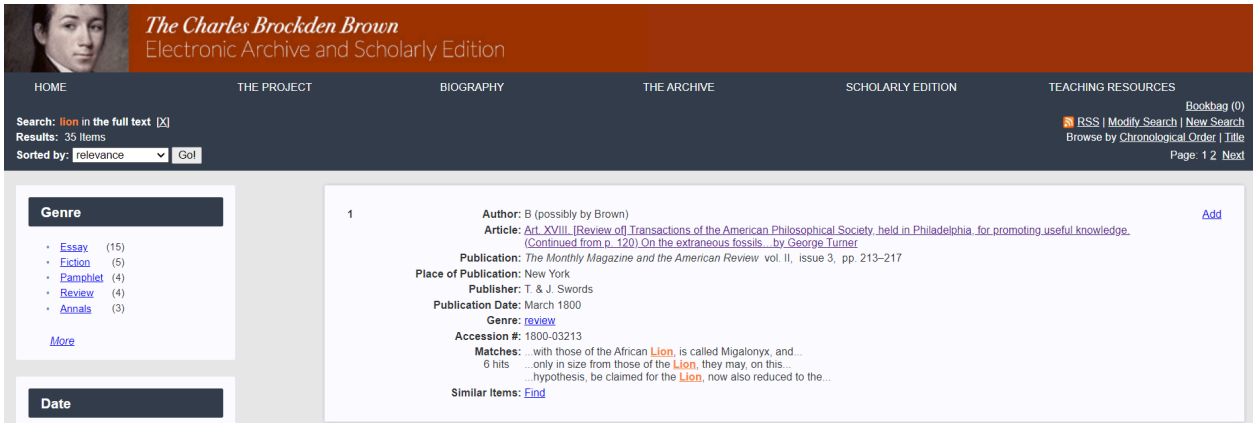
team. I look forward to learning from my teammates who have different sets of skills and expertise. I am also excited to practice and improve my teamwork and communication skills which will similarly be crucial in the future.

Project Characterization & Design Documentation

Bug Fixes - Gael

The Bookbag - Gael

On the Archive's website, there is a function called the Bookbag. It is meant to work in the following way: when a user performs a search that yields results, they are shown a page with all the available literature containing words that match their search term(s). They are able to "Add" any search results to their Bookbag. Once they have added the literature they'd like to save to their Bookbag, they can click on the Bookbag button. This will show them all the items they've stored. The user can then follow a menu that allows them to enter their email and receive a message with permalinks to all the literature they saved in their Bookbag.



The cookies should persist throughout a user's session. Since the Archive does not provide user accounts, each user should be tracked by their IP address. If the same IP address visits the website again after having

saved items to their bookbag and then having left, their Bookbag should not contain anything from the previous session. A new session is started each time the user closes their web browser.

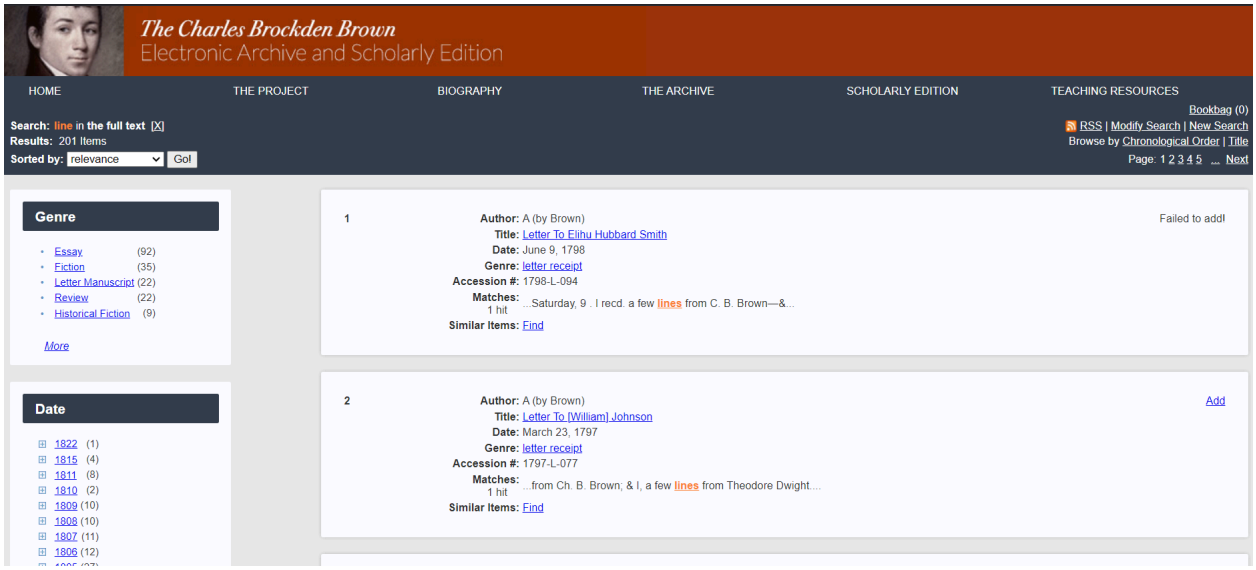
However, if the user has cookies disabled in their browser settings, they will see a short message on each entry letting them know that the Bookbag does not work without cookies. A user may need to visit their web browser's settings and ensure that the domain involved here (brockdenbrown.cah.ucf.edu) is properly able to place cookies. Otherwise, since the feature depends on cookies, disabling cookies will render the Bookbag unusable.

Thus, the intended usage flow of the Bookbag feature should be as follows:

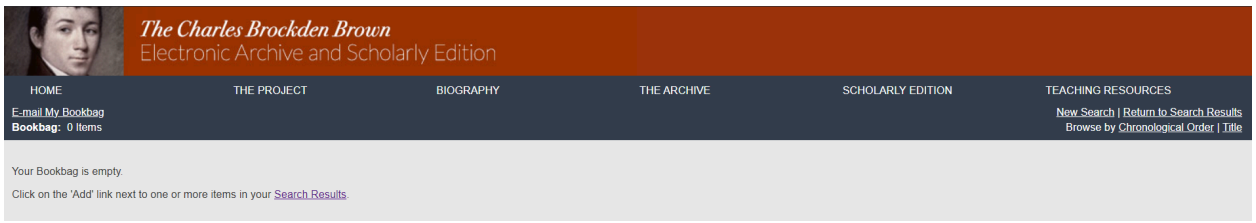
1. The user performs a search. They are taken to the Search Results page.
2. Each search result has a hyperlink on it. When the user clicks on it, one of two events should be triggered:
    a. If the user has cookies enabled in their browser, hyperlink should read "Add". When clicked, the Add link will become plain text reading "Added!". The Bookbag, which resides in the header at the top of the page, should now display a number +1 higher than the one before, indicating that another article has been added.
    b. If the user has cookies disabled, the link should read "Requires cookie*". When clicked, the website uses JavaScript to send an alert directly through the user's web browser reading "To use the bag, you must enable cookies in your web browser."

3. Should the user choose to visit their Bookbag, they should see a new page (https://brockdenbrown.cah.ucf.edu/xtf3/search?smode=showBag) that behaves differently depending on how many items the user has stored.
    a. If the user has not stored any articles in their Bookbag, the page shows the following text: "Your Bookbag is empty. Click on the 'Add' link next to one or more items in your Search Results."
    b. If the user has one or more items in their Bookbag, the page will display different text. (However, as detailed below, it is not currently possible to know what that page should look like.)
4. At the top of this page, there is a link reading "E-Mail My Bookbag". This button opens a new page in a new window (https://brockdenbrown.cah.ucf.edu/xtf3/search?smode=getAddress) which prompts the user to enter their email address and click Submit. If the user has not disabled their browser's protection/security settings, it will throw a warning that the field being filled is not secure.
5. The user should then receive an email with permanent links to the items they chose.

There are several problems with this feature. The first is that attempting to Add items to the Bookbag simply does not work. While enabling cookies will change that message to a hyperlink reading "Add", clicking it only yields a message reading "Failed to add!". There is no viable way to use the Bookbag, which means that there's no way for users to "permanently" save any permanent links from the Archive.



The "Search Results" link in the Bookbag page does not actually redirect the user back to their results; it only refreshes the current page.



Finally, the E-Mail submission window is broken as well. The text box where the user types their email address appears to be insecure plaintext.

When clicking submit, the average web browser will actually throw a warning to the user, letting them know that the form is not secure. (We know it uses a GET request because it redirects to https://brockdenbrown.cah.ucf.edu/xtf3/search?email=ga193888%40ucf.edu&smode=emailFolder where I've used my own email as an example). If the user chooses to proceed anyway, they are led to an error screen:

At first, we planned to fix these bugs; it was initially one of our requirements. During our testing, we found that users are given cookies from the domain "brockdenbrown.cah.ucf.edu". In our project we would likely have needed to edit the code handling these cookies so that the Bookbag would finally be fully available. It seemed as though we might have also needed to look into how the PHP code files are handling the email submissions.

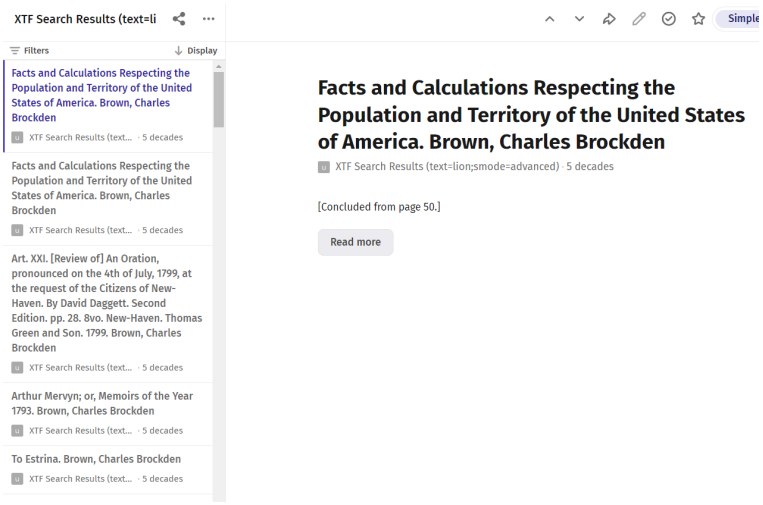
After discussing the situation with our sponsors, however, we decided together that the feature is not worth bringing to the new frontend. Per Dr. Giroux, the bugs surrounding the Bookbag have existed for over ten years, but few users have raised any complaints. This seems to indicate that almost no one is attempting to use the Bookbag. Therefore, our Sponsors and our team together decided that it is best to drop the feature entirely.

RSS Feeds - Gael

The RSS Feed feature of the website is also only partly functional. When the user performs a search and is taken to their Search Results, there is a link reading “RSS” near the “Bookbag” link.

Upon clicking this button, the user is taken to an RSS feed which, obviously, a web browser can’t render correctly. An app known as an “RSS reader” must be used to view the feed correctly.

When plugged into a reader app, the feed displays as follows:



The RSS feed takes the Search Results from where the user clicked the RSS link and displays them to the user. However, this feed only displays the title of the article and perhaps a few words from the beginning. There's no way to read full items or articles from within an RSS reader. In some reader apps, such as the one displayed in the image above, a Read More button is provided which, if clicked, will redirect the user to the desired article on the Archive's website:

Since this feature does not work universally with all RSS readers and does not produce any helpful results, the Sponsors have agreed that this feature will be dropped as well.

Library of Congress Headings - Gael

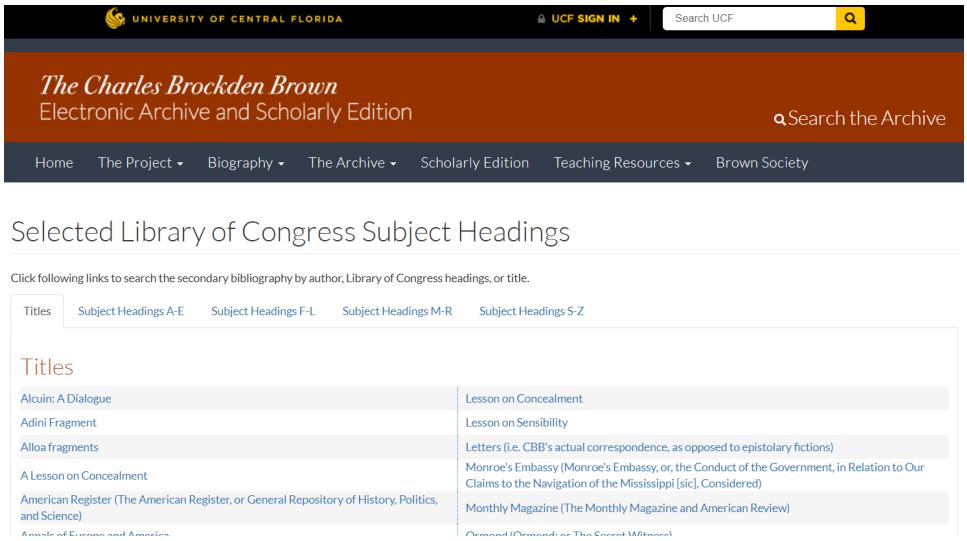
The Subject Headings provided by the Library of Congress serve a special function in the realm of literature preservation and categorization. With them, each preserved article can be categorized under one or more "keywords", usually nouns or adjectives that describe broadly what the document is or the subject matter of the document.

According to the Library of Congress, this system goes back to 1898—less than a century after the death of Charles Brockden Brown. Each heading is identified by a word—for example, School Sports–Fiction is a heading. It is also identified by the alphanumeric *sh2010112128*. Each heading has a unique alphanumeric identifier, making them easier to sort and comb through (for a computer, at least).

In the Charles Brockden Brown Archive, Library of Congress Subject Headings are used to search through the site's Secondary Bibliography. First, we should be clear: the Secondary Bibliography is a comprehensive collection of references to scholarly articles (mostly dissertations) that focus on the writings of Charles Brockden Brown. The section of the site that searches using Library of Congress headings can be found as follows:

1. The user goes to the Archive's webpage (https://brockdenbrown.cah.ucf.edu/archive.php).
2. They scroll down to the box that reads Library of Congress Subject Headings. The user can either click that text or the button beside it that reads "Selected Library of Congress Subject Headings to enhance your search for scholarship on Brown". Both then redirect to the page where this search will be performed (https://brockdenbrown.ucf.edu/loc.php).

3. The user is now presented with various subsections containing many links to keywords–the first "tab", named "Titles", allows the user to search for scholarly articles that contain the title of a Brown piece. The other tabs contain actual LOC headings; they are alphabetized keywords such as "abolitionism", "melodrama", and many others.
4. Should the user click on one of those links, they are redirected to https://brockdenbrown.ucf.edu/archive/secondarysearch.php?keyword=x where x represents the subject heading the user clicked on. The user can then select a date range and search through the Secondary Bibliography.



The problem with this search is that the information for the Secondary Bibliography is hard-coded and stored in two different places: one is in the file LOC.XLS and the other is stored in the XML files. This is clearly not maintainable or streamlined. The current system requires a developer to enter CHDR, look for the appropriate file, edit it by hand, and then reload the website. This is not only an inconvenience for the maintainers of the site, but it is also a solution prone to error and possible bugs.

Our solution for this problem was to implement the Secondary Bibliography in exist-db as well. This allowed us to use exist-db as a central location for the references that can be dynamically edited—much better than having to search for the .xls file in the website's internal code and editing it by hand.

Although this was initially considered a stretch goal of our project, and we had started working on it at some point, our sponsor decided that there was no need for it to be implemented as the PI would rather have those manually edited, so we ended up not bringing this feature to the production environment.

Backend: Summary

Our goal was to migrate all the existing XML files for the website to a system called exist-db. Our Sponsors showed us how to access the database and where to find the XML files for the project. The project will be built on and deployed in the 9th edition of Red Hat Enterprise Linux (RHEL9 for short). The Sponsors explained to us that no documentation of the code for the site exists, which means that we were faced with the task of analyzing the code with their assistance to truly understand what needs to be upgraded. This will also be essential when it is time to build documentation on how to make the official migration from RHEL7 to RHEL9.

Linux - Bernardo

Why Red Hat 9 over Red Hat 7?

There are multiple reasons why our sponsor decided to change the website's backend from Red Hat 7 to Red Hat 9, however, the ones that will be highlighted in this document will be the following: the presence of outdated packages, security concerns, performance constrains, containerization limitations, and lack of support.

When using Red Hat 7, the default version of PHP will be 5.4, which has had its end of life in September 15th of 2015 (PHP End of Life), and therefore, it is severely outdated. Due to that, it lacks support for several modern PHP features that have been introduced in later versions such as return type declarations, anonymous classes, and null coalescing operators. On top of that, several modern frameworks such as Laravel and Symfony require PHP 7.4 and above (Requirements for Running Symfony). However, that does not mean that it is not possible to get a more up-to-date version in Red Hat 7, but in order to install a newer version, the use of external repositories such as Remi would be required, which can bring more complexity to the overall project, as its packages and dependencies can conflict with other third-party ones, as well as, with the default Red Hat 7 ones. On top of that, some PHP extensions are simply not available within Red Hat 7's repositories such as Xdebug and Redis, and web server technologies such as Apache and Nginx only contain their outdated versions within those repositories.

Within the security risks spectrum, we have several sub-categories that can be explored, as an outdated operating system, especially in a Linux-based environment, can generate vulnerabilities from several different points. Since Red Hat 7's end of life, users that would like to receive security updates are required to pay for Extended Lifecycle Support (ELS) (Red Hat Enterprise Linux Extended Life Cycle Support (ELS) Add-on) which still does

not guarantee that most of the vulnerabilities will be addressed. Along with that, we also have the fact that Red Hat 7's most up-to-date kernel is 3.10.x, which does not have several security features that can be found on more modern kernels such as seccomp filtering and enhanced address space layout randomization (ASLR). Being on an older kernel version also makes your system more susceptible to privilege escalation attacks, as it tends to be easier to gain control of the file system of outdated kernels. An outdated operating system can also cause security vulnerabilities due to the issue previously mentioned with the outdated packages, for example, Red Hat 7 including OpenSSL 1.0.2, which does not support TLS 1.3 ([openssl-users] openssl 1.0.2 and TLS 1.3), makes a web server running on it more vulnerable to DDoS attacks.

Most of the performance constrains come from the factors previously mentioned, as more up-to-date packages tend to also contain performance improvements, however, it is important to highlight some that will be essential for this project. The main performance improvement that will be seen when the backend is moved over to Red Hat 9 will be due to the machine's kernel version, as newer kernels contain several I/O handling improvements, which is essential to improve the performance of PHP applications that interact with the machine's file system. On top of that, the system's ability of handling multiple incoming requests is also related to those I/O handling improvements, which will allow the system to utilize multithreading more efficiently.

When it comes to the containerization limitations present in Red Hat 7, there are several factors that need to be taken into account, starting with the fact that containerization will be heavily utilized in this project, and therefore, it being the most optimized as possible is essential for the

backend to run efficiently. The containerization software that we planned to utilize for this project was Podman, which is a very light and secure open-source containerization tool. Although Podman is highly indicated when working with Red Hat servers, native support for it was only introduced in Red Hat 8, which means that external repositories are needed to install Podman on a Red Hat 7 server. On top of that, Podman requires some libraries such as systemd and libc to be more up-to-date than their native versions in Red Hat 7. The older kernel versions supported by Red Hat 7 also lack features that are essential for Podman's full potential to be achieve such as rootless containerization and may cause it to display worst I/O performance than expected when accessing files and interacting with the file system in general. One of Podman's main characteristic is also its high compatibility with Kubernets, which tends to suffer when it is being run on Red Hat 7 in multiple aspects, one of them being the network capabilities. As the version of Container Network Interfaces (CNI) present in Red Hat 7 is considerably outdated, it might be harder than expected to perform more complex network setups, such as setting up overlay networks.

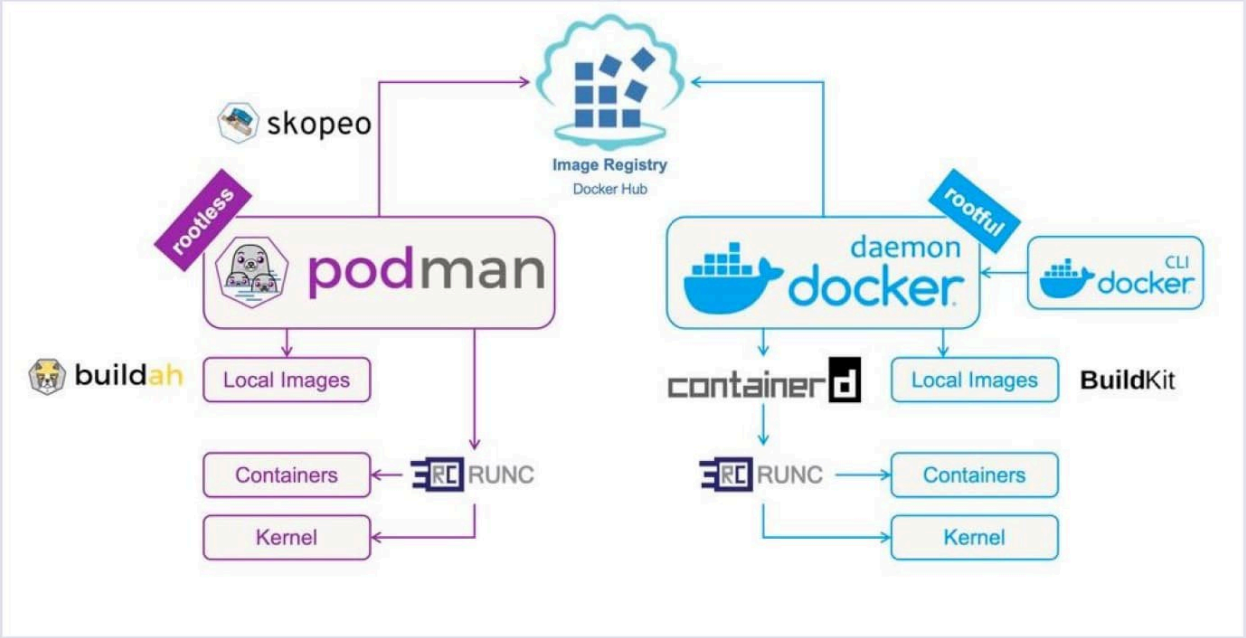
Aside from that, IBM, which currently owns Red Hat since 2018, has announced that Red Hat 7 has gone out of support as of June 30th of 2024 (IBM is announcing Red Hat Enterprise Linux 7 is going End of Support on 30 June 2024). This means that as of that day Red Hat 7 will no longer receive security patches, hotfixes for bugs, neither regular maintenance updates, unless the user in question purchases the Extended Lifecycle Support (ELS), which will allow them to receive critical security updates and support for selected packages. An operating system going end of support also means that a lot of applications will stop updating their software for it, which in the long run will cause Red Hat 7 to only contain heavily outdated libraries of most applications and to be excluded from several third-party vendor

support. Although not obvious, in the long run, this will also generate a bigger maintenance cost, as the company will not only need to pay for the Extended Lifecycle Support (ELS), but also will require to do a lot of manual workaround to get certain packages and applications to work with that server.

Why Podman over Docker?

Although our sponsor ended up deciding for not implementing containerization within this project, we would like to point out some of the pros and cons that lead us to pick Podman over Docker in the early stages of the project. The main reason why we initially selected Podman was the fact that our time was considerably limited for everything we were doing, and changing the containerization software would implicate in performing a lot more changes and testing than us and the sponsor anticipated when the project was proposed. Even though that was the main reason, we did weight in other factors into this decision, since if Docker was by far the best option, we would've put in the effort to get it to work within the allocated time we have to complete the project, if we were indeed implementing this feature.

The pros that will be highlighted in this document are Podman's better security, lightweightness and modular design, integration with Systemd, and support specifically for Red Hat 9. On the other hand, the cons that will be highlighted are Podman's composer feature being less developed than Docker's, ecosystem limitations, and necessity to utilize other tools to achieve setups that Docker has within it.



Schematic demonstrating how Podman and Docker work

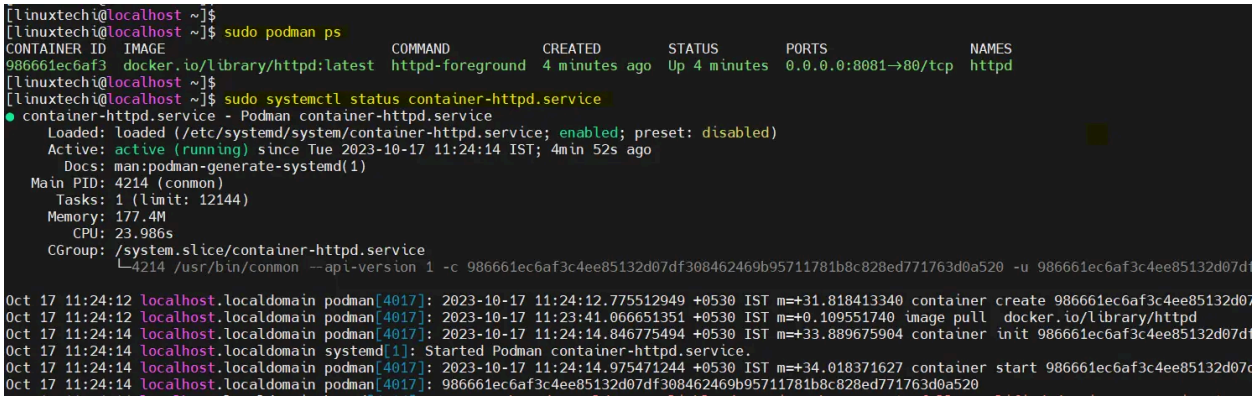
Source: Battle of Containerization Titans: Podman vs. Docker

When it comes to Podman's security, there are a few points that we need to highlight, the first of them being how it handles root users within a container. While Docker runs containers with root-privileged daemon, Podman allows rootless containers (Rootless containers with Podman: The basics) to exist, which significantly limits the amount of damage that can be caused by a compromised container, as it will not have root access to the host itself and neither to the other containers. Podman is also natively integrated with multiple kernel security modules, SELinux security policies (Chapter 9. Creating SELinux policies for containers), and syscall filtering features, such as seccomp, while Docker requires additional work to be integrated to those.

On top of having better security standards, Podman's modular design makes it more lightweight than Docker, which improves the server's performance significantly. As Podman is daemon-less, it allows containers to

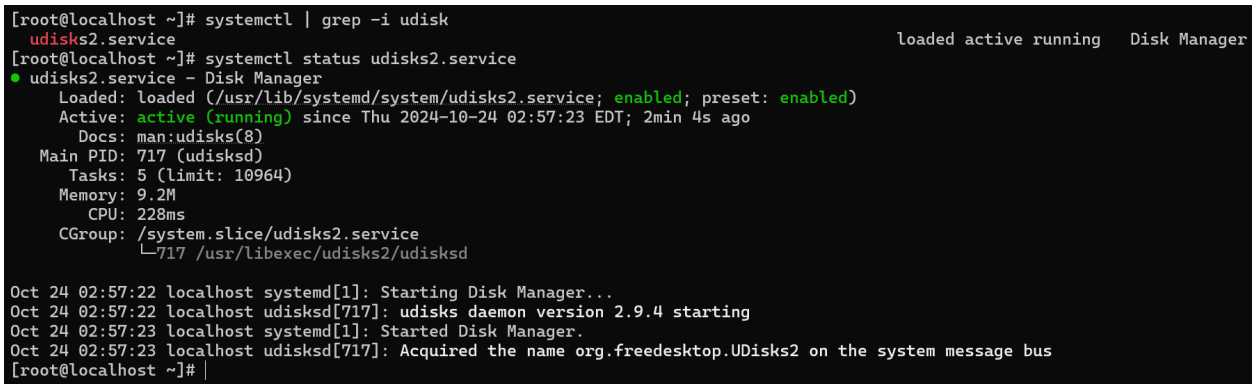
run as independent child processes, which allows resource management to be more efficient, resulting on minimal resource usage when compared to Docker. It's modular design also provides tools that allow flexibility and more efficiency for specific tasks, such as Buildah (What is Buildah?), which can be used for image building. Due to those factors, Podman's performance also tends to be better, as its lack of a daemon process allows the containers to be initialized faster.

Podman's integration with Systemd (Chapter 4. Running Containers as systemd Services with Podman) helps a lot in terms of practicality, as it allows us to manage the containers as system services via systemctl and copy certain files over the system folder for automatic startup (/etc/systemd/system). This facilitates working with processes that need to be automatically started post-reboot or through specific jobs, as it is way easier to manage the start, restart, and stop of those processes through systemctl (Configure a container to start automatically as a systemd service). This also significantly increases the reliability of Podman, since systemd being a default Linux package will allow the containers to behave and run like any traditional Linux service. Since most of our group did not have previous Linux experience, this also makes our learning curve of the tool way quicker, as we do not need to learn both proprietary commands and technologies from Podman, but instead, we can use a default Linux tool to manage it. Below we can see an example of how containers are managed through systemctl with Podman and its comparison with a regular service's status.



Podman container running as a service and being managed through systemctl

Source: [How to Run Docker/Podman Container as Systemd Service]



Example of a regular Linux service being managed through systemctl

Source: Personal Red Hat 9 Virtual Machine

Along with all the pros previously mentioned, the biggest one of them is certainly the fact that Podman is officially supported by Red Hat as of Red Hat 8 making it fully optimized for the RHEL ecosystem and causing Red Hat to provide long-term support for it. Due to this compatibility, it is way easier to install and manage Podman's packages, as we can do so by utilizing native Red Hat tools such as dnf, which, as previously mentioned, helps us a lot considering that most of the team members do not have previous Linux experience. On top of being able to use dnf for package installation and

management, Podman is also compatible with other features within the Red Hat operating system such as Red Hat Insights (Red Hat Insights), which allows proactive troubleshooting and security management of containers through its monitoring system. Although Docker is more commonly used than Podman, the native integration between Podman and the Red Hat system provides a compatibility level that cannot be disregarded, and therefore, this was the main point that lead us to maintain our containerization tool as it is. However, as previously mentioned, it is important to highlight some of the cons of using Podman over Docker, as we did debated the possibility of utilizing it at some point.

It is notable that a lot of Podman's down sides are related to it less developed ecosystem, as it is mostly utilized in Red Hat based environments, while Docker is compatible with all three most common operating systems (Mac, Windows, and Linux). It is natural that a software that is compatible with more operating systems and multiple Linux distros will attract more users and consequently, be more developed. Based on that, the first point that we would like to highlight regarding Docker's pros over Podman is the fac that its Compose feature is considerably more developed than Podman's (Podman Compose or Docker Compose: Which should you use in Podman?). The compose feature consists on a tool to manage multi container applications utilizing YAML files, and its main idea is to make complex setups easier to be created and managed in the long-term. However, due to Podman's composer not being as developed as Docker's, some of those more complex setups might require a lot of troubleshooting and workaround to be achieved, if they are even possible to be created through the tools' capabilities. Docker's composer feature also has a lot more support from the community, as it has more users due to the reasons previously described,

which provides more troubleshooting and management resources for those utilizing it.

Following this same idea, we can also understand the next topic, which consists on the fact that Podman is way more limited than Docker when it comes to third party tools and add-ons. Due to Docker being so popular and multi-platform compatible, several CI/CD tools have Docker as the default containerization tool, which causes Podman users to have to perform additional configurations or utilize custom scripts to integrate it with those tools. On top of that, several monitoring tools such as Grafana and DataDog have plugins that integrate directly with Docker's API, while we do not observe the same availability of those for Podman.

Considerations about other Linux distributions

Just like at some point we weighted the possibility of moving the backend to Red Hat 8 instead of Red Hat 9, we also considered if it would be feasible to run the backend in a different Linux distro, such as Ubuntu or Fedora. The main reason why we went away from this idea very quickly was due to the fact that this would also implicate us changing our containerization tool from Podman to Docker, as a lot of the pros we had weighted to maintain Podman were highly related to our backend running on Red Hat. It wouldn't not make sense for us to change the operating system the backend is running on and not changing our containerization tool, since we already had several pros in moving to Docker, and most of the pros we had related to Podman were not going to exist anymore as soon as we decided to utilize a different distro.

Based on that, we decided that the time constraints we had for the project would make this process more of a time crunch than we would like it to be, as we can still achieve the desired result maintaining everything in Red Hat and Podman, especially because this was the sponsors initial request, although they always made themselves very open to our improvement ideas.

When it comes to the reasons why we considered Ubuntu and Fedora specifically, there are a few of them that we would like to highlight in this document, which are their available development tools, community support, and overall licensing cost. Although those were the main reasons why we considered those distros at some point, that does not mean that other Linux distros were not suitable to replace Red Hat on this project, neither that they do not fall into one of the reasons listed above.

In regards to the available development tools, both Ubuntu and Fedora naturally will have more available, as Red Hat's focus is on stability, and consequently, it requires a lot more manual work than the other two to interact with modern software stacks. Fedora for example comes pre-configured with several developer tools and has support for multiple modern features and libraries such as Composer. Ubuntu, although not coming pre-configured with as much as Fedora, does have extensive documentation and community engagement, which causes it to be very easy to install and manage web stacks on it, since there's a lot of resources available to troubleshoot it as needed. This also brings us to our next topic, which is the community support.

While Red Hat does have some community, its main focus is on enterprise level usage, and therefore, it has considerably less interaction and

support from its community, as lot of them are big tech companies. On the other hand, Ubuntu and Fedora have several forums and documentations that can be used to resolve issues quicker and assist in the project's development. Red Hat's focus on enterprise usage also causes its licensing model to be less attractive to small businesses and single individuals that are looking in developing their softwares using it, while Ubuntu and Fedora are both open-source and free, which is essentially one of the best things for a developer to work with.

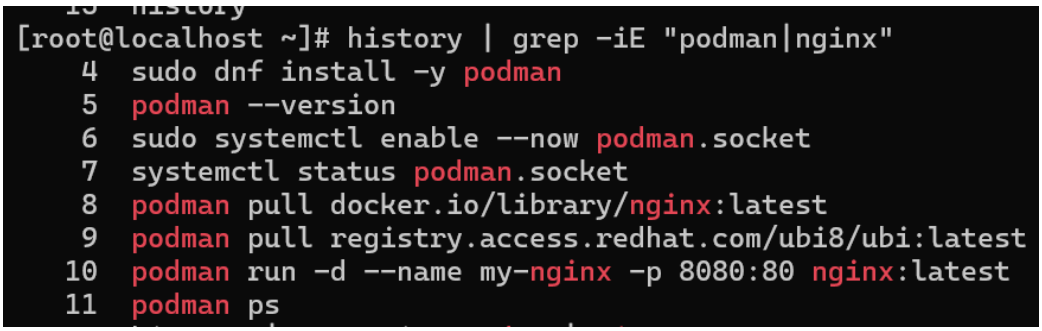
One very important factor that made us stick with Red Hat was due to its increased security when compared to Ubuntu and Fedora. Due to Red Hat enterprise focused, its security also needs to be more robust than other distros, as it needs to be compliant with several industry security standards such as FIPS 140-2. Red Hat also enforces SELinux way more strictly than Fedora, while Ubuntu utilizes a different security-enhancement tool called AppArmor, which isn't as strict as SELinux. On top of that, Red Hat has their own security response team which allows them to release vulnerability patches quicker. It is also important to note that Red Hat guarantees up to 10 years of security updates through the previously mentioned Extended Lifecycle Support (ELS), while Ubuntu only offers 5 years under their LTS licenses.

Red Hat Virtual Machine Set Up

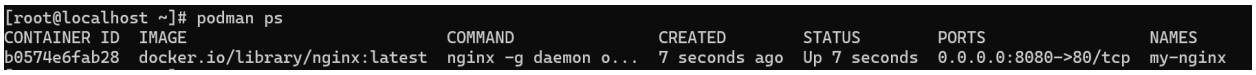
In order for the entire team to get familiar with Red Hat's environment and Podman container management, each one of us created a virtual machine running Red Hat 9.4 (Plow). To get the ISO, we created an account under Red Hat (Red Hat Account) and that allowed us to get a copy of the ISO, as well as activate the operating system's license during the

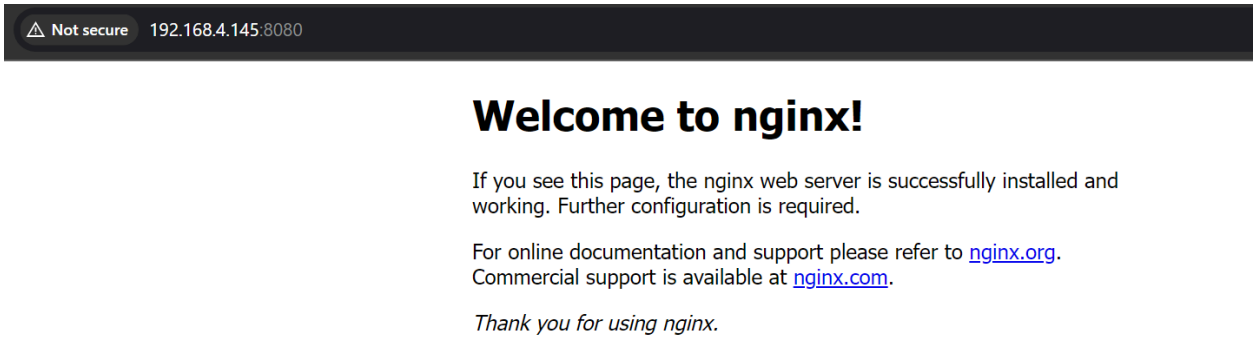
installation. In terms of getting the actual virtual machine spun up, most of us utilized Virtual Box, however, two of our team members had access to VMware and decided to utilize that instead.

After getting the virtual machine spun up and working as intended, we installed Podman and initiated a container to confirm it was running successfully, as per the screenshots displayed below.



Commands that were run to set up a basic Podman container on one of the Red Hat 9 virtual machines





Proof that the container was created successfully

In order to complete those steps, we followed the installation and basic set up tutorial under Podman's documentation (Podman Installation Instructions), which also helped some of our members to get familiar with basic Red Hat commands such as dnf and systemctl.

We also worked on moving the cloned Git repository files from our local virtual machine to the container utilizing Podman's commands such as podman-cp (Podman's cp documentation used) and podman-exec (Podman's exec documentation used). Although somewhat simple, we all found Podman's documentation very easy to follow and apply while performing testing with the containers, which highlights Podman's simplicity that was previously mentioned.

What changes from PHP 8.3.8 to PHP 8.4.1?

Although the current production server is running a very outdated operating system, which has even reached its end of life already, its PHP version isn't as old as we anticipated when we first started this project. This brings some pros and cons, like everything else.

Pros:

- As there are not a lot of changes between the two PHP versions, due to them being considerably close to each other, we do not need to make heavy adjustments to the current codebase

- Due to the PHP version not being extremely outdated, we have a good base structure to work and a codebase that is easy to get information from and about

- It is very easy to find documentation on error messages or bugs that we encounter while coding

Cons:

- When we first started the project, we expected that more improvements could be made to the codebase, however, since the PHP version is not as outdated as we expected, there aren't that many improvements we could do

Even though those are pretty close versions of PHP, there are some key changes that have happened between the releases that separate those two, such as the following:

- The introduction of Property Hooks, which allow us to define custom behavior when retrieving or modifying an object's properties. This feature reduces the need for explicit getter and setter methods.

- The introduction of Asymmetric Property Visibility, which us to set different visibility levels for reading and writing object properties. For example, an object called Counter that contains an attribute count and a method called increment, could be set up in a way that counter can only be updated via the increment method, although we could access the value of counter and print it. This enhances encapsulation, which is essential to avoid issues later on with the code.

- Improvements in the DOM API, which include support for HTML 5 parsing and serialization. This significantly increases the efficiency when manipulating HTML and XML documents. This will be one of the

most important features for our team to explore when implementing PHP 8.4.1, as we will be handling a considerable amount of XML files through the backend.

- The introduction of Lazy Objects, which avoids an object from being created until it is called/needed. This can improve performance by avoiding the overhead of creating objects that may not be utilized during execution.

- The implementation of driver-specific subclasses within data objects, which optimizes the code's interaction with databases by providing access to driver-specific methods and properties. This would have been very useful within our project if we were utilizing one of the databases that support this implementation, such as MySQL, however, this is not the case for exist-db.

- The introduction of BCMath, which provides an object-oriented interface to perform mathematical computations.

From those features, we expect to possibly utilize three of them within our project: Property Hooks, Lazy Objects, and the improvements to the DOM API. We expect the use of property hooks and lazy objects to significantly improve the backend's efficiency, and the DOM API improvements to allow us to better manipulate and serialize the XML and HTML files that will be retrieved from exist-db. However, as our main focus is to have the backend working properly within Red Hat 9 and improving the UX design of the website, adding those features to the backend code will not be part of our requirements, although it is something we would like to implement for the overall efficiency of our backend infrastructure.

PHP-Backend - Gael

GET vs. POST - Gael

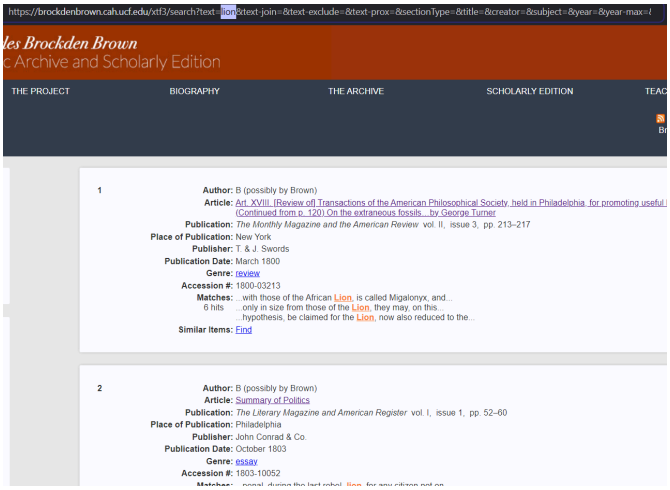
It was definitely a good idea to switch all our methods from GET to POST. GET is problematic because it sends requests through the web browser's address bar. This means that any user with enough knowledge of the URLs of the Archive could easily cheat the system and inject prompts into areas like Search.

For example, using the Search function normally works as follows:

1. The user goes to the Archive's webpage (https://brockdenbrown.cah.ucf.edu/archive.php).
2. They scroll down to the box that reads Primary Bibliography and enter a search term in *keyword, genre, title*, etc.
3. The page then redirects to the search results.

Notice, however, that the user's search term will show up in the address bar. For example, searching for the keyword "lion" yields this URL: https://brockdenbrown.cah.ucf.edu/xtf3/search?text=lion&text-join=&text-exclude=&text-prox=&sectionType=&title=&creator=&subject=&year=&year-max=&type=&smode=advanced

This makes it incredibly easy to modify the search terms without using the webpage's forms again. Just replacing the term "lion" in the URL with any other search term will allow the user to enter whatever term they want.



This is a security problem—it allows the user to interact with the site in unintended ways. To fix it, the site can instead POST methods to protect the privacy and security of the site's searches. This ensures that the only way a user can perform searches—and any other functions—is by going through the menus that the developer intended for them to use.

The flow of the site's search engine would be a bit like this:

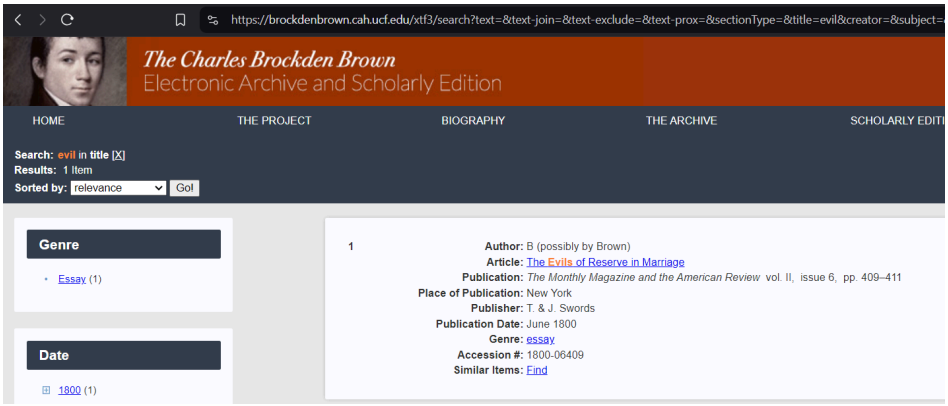
1. The user goes to the Archive's webpage (https://brockdenbrown.cah.ucf.edu/archive.php).
2. They scroll down to the box that reads Primary Bibliography and enter a search term in *keyword, genre, title*, etc.
3. The page then redirects to the search results.

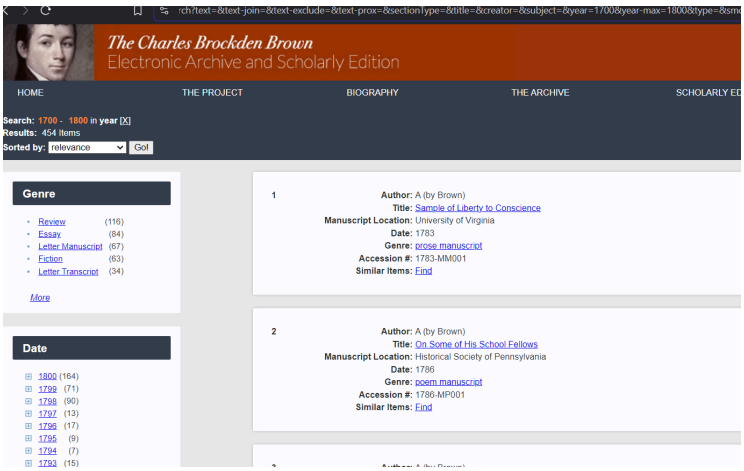
The difference this time around is that the page redirected to has a much shorter URL—something similar to: https://brockdenbrown.cah.ucf.edu/xtf3/search. Because a POST request method was used, the details of the search are no longer leaked to the address bar. This protects the site from unwanted queries that might cause bugs or glitches.

This applies to every part of the Search experience on the Archive. Searching using other filters, such as genre, title, or date range will yield long URLs such as these:

- https://brockdenbrown.cah.ucf.edu/xtf3/search?text=&text-join=&text-exclude=&text-prox=&sectionType=&title=&creator=&subject=fiction&year=&year-max=&type=&smode=advanced
    - This URL represents a search where the field *subject* has been given the keyword "fiction". It will display any articles that have been categorized as fiction.
    - 

- https://brockdenbrown.cah.ucf.edu/xtf3/search?text=&text-join=&text-exclude=&text-prox=&sectionType=&title=evil&creator=&subject=&year=&year-max=&type=&smode=advanced
    - This search looks through only the titles of each item in the Archive and looks to match with the user's keyword—"evil", in this case. The Search Results will display any item whose title contains that term.

- 

- https://brockdenbrown.cah.ucf.edu/xtf3/search?text=&text-join=&text-exclude=&text-prox=&sectionType=&title=&creator=&subject=&year=1700&year-max=1800&type=&smode=advanced
    - Finally, this one uses a date range. Note that, because two fields were technically filled, the GET request populates two arguments in the URL: year and year-max.
    - 

Clearly, these are suboptimal and make the user's experience cumbersome. Using a POST request may help change this.

PHP-Backend - Rodrigo

PHP and Exist-DB - Rodrigo

The Brockden Brown website is currently using PHP for the back end. We continued the usage of PHP and worked on developing new functionalities as needed. For instance, eXist-DB uses REST APIs through HTTP for access to the database, so we needed to code that functionality. We had to develop a function to make HTTP requests through php, interfacing with eXist-DB's API endpoint. Exist-DB listens for REST requests, by default, on port 8080 with the subdirectory of /exist/rest, the subdirectories following that indicate the path to a database collection. In the request to http://localhost:8080/exist/rest/db/brockden/document1.xml, the server checks if the document1.xml file exists in the db/brockden collection. It then responds with the contents of the file if it exists, or returns with HTTP 404 if it is not found. We had to leverage that when building out the function to properly access files on eXist-DB.

PHP and Security - Rodrigo

There are several ways we utilized to implement security into our development process. One of the php functions we leveraged is the htmlspecialchars function. This function is used when receiving input from the user. It is important not to inherently trust any input received externally, as it can be potentially tampered with. We used htmlspecialchars to convert all characters received from the browser to HTML entities. This helped avoiding code injection attempts, which is when a user submits information in an input field to have the server process the information and potentially execute the instructions given. This will in turn allow for further exploitation,

data exfiltration, and much more, as it ensures that special php characters are converted to HTML objects and treated as such, rather than executed as PHP code.

Another function we leveraged is filter_input. This function is used to validate variables received from insecure sources (user input). It can validate that the correct HTTP request was used with INPUT_GET or INPUT_POST. We can then specify which filter to use, such as FILTER_SANITIZE_STRING which will strip tags and HTML encode double and single quotes. This function will further help in preventing code injection attempts.

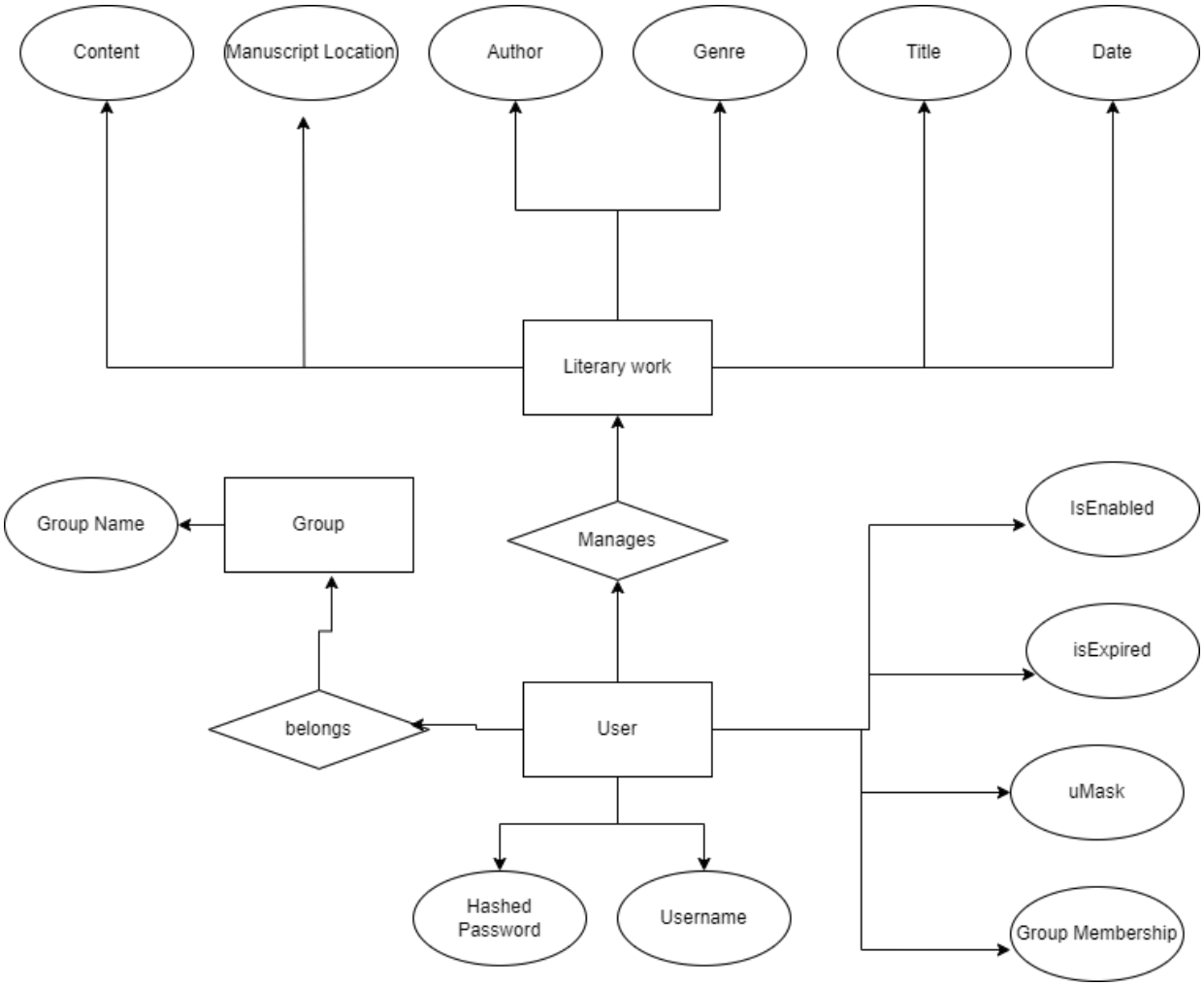
Backend Storage - Rodrigo

Entity Relationship Diagram - Rodrigo

We created an entity relationship diagram (ERD) to help us understand how data is stored and how different entities interact with it. Our Database solution is Exist-DB, which focuses on storing XML files. Our main entity of our project consists of the Literary Works that are stored. These literary works have various attributes, there are additional TEI attributes that were not added to the diagram, such as TEI edition, encoding name (the name of the person responsible for the TEI encoding), and many more. The main attributes of each literary work include the content of the literary work along with various metadata information, such as Author, Title, Manuscript Location, Date at which the work was written and Genre (which could be Essay, Letter Manuscript, Fiction and so on). The Literary work entity is the entity our back-end server will be interacting with. We also included two other entities into the diagram. The user entity represents the user

information that is stored in the Database. This user information is used for authentication in the Database management system. Each user has several attributes, such as its username, the hashed password, group membership, user mask, and two boolean variables, isEnabled and isExpired which indicate whether the account is usable or not. A user is then a member of a group which only has the attribute of group name.

The relationship between the three entities is as follows. The user which authenticates into the management console is a member of a create group and manages the literary work, with actions such as addition, modification, and deletion.



Entity Relationship Diagram of our project

TEI - Rodrigo

All of the XML files that will be stored in the database are encoded in TEI, so it is important to understand how it works, so we can better leverage it. TEI stands for Text Encoding Initiative and involves a convention, a standard for XML files to ensure interoperability between different projects. It involves the usage of TEI tags to describe structural divisions and characteristics of a given document. Those tags are also referred to as modules which are used to declare particular XML elements and their attributes. Those elements can then be combined to form a schema to a particular set of requirements (the TEI Infrastructure - the TEI Guidelines, n.d.).

XSLT - Bernardo

One of the main goals of our project was to ensure that the document viewer that opens up whenever you select a document was working as intended. Since all the documents are in XML formatting, we needed to utilize XSLT transformations for the system to interpret the TEI markups within that XML file as HTML/CSS. Essentially what this does is to grab a text like the one displayed below and matches it with a pre defined HTML/CSS rules

"Is she well?—is she happy?—The letter—have you any letter?—

What is the matter with you? Why are you so impatient? I <hi rendition="#UL">have</hi> a<lb/>

letter for<del rendition="#overstrike">m </del>you. My Sister is in health.

For example, with the text above the following transformations would be applied:

1. The would be converted to display like a regular p tag in HTML - which displays the text as a regular paragraph
2. The <hi rendition "#UL"> would be converted to a highlighted text that might be also rendered as underlined if we applied CSS
3. The <lb/> would be rendered as a line breaker
4. The <del rendition="#overstrike"> may appear struck-through if styled properly

This entire process of accounting for all the transformations was very time consuming as we have almost a thousand files in the archive and we had to essentially go through all of them to confirm we had added all the possible markups to the style sheet, while also making sure we go through every single file in the staging server to see if there's anything not being rendered as intended.

One thing that did help us a lot was the fact that our sponsor was able to schedule a meeting between us and one of the original developers of the website, who is indeed an expert in XML handling and knows a lot about XSLT transformations. He provided us with a lot of insight on how the stylesheets work as well as gave us tips on how to make this process less time consuming. He also pointed out how helpful it would be for us to get Oxygen licenses, which is an application made for XML handling and editing. The sponsor ended up deciding, based on the former developer's suggestion

that it was worthy to provide us with those licenses to make the development process quicker, and we were granted two licenses, one for Bernardo, and one for Dawn, as they were the main people involved in this portion of the project.

One of the challenges we encountered was displaying the thumbnails for the original documents, as we not only had to display them nicely, but they also had a specific functionality where if you click them, it will open a new tab on your browser with the document in regular size. Although this was a bit harder to accomplish compared to the rest of the transformations, which were mostly busy work, we ended up getting it without the need to reach out to the former developer, which was our main goal.

XTF and Exist-DB - Rodrigo

One of our team's requirements for this project and part of our Minimum Viable Product (MVP), is to migrate the back-end from using XTF, to Exist-DB. XTF stands for eXtensible Text Framework; it is built on Java and XSLT, and provides flexible indexing, displaying, and querying solutions, supporting searching across collections of heterogeneous data. The workflow of this software begins with a web-based user search query directed to the server. This engages a servlet (crossQuery) which checks the query against an index of available documents, producing a list of matching documents for display. A different servlet is then invoked (dynaXML) which then retrieves and formats the actual document to then have it be displayed in the web browser as a result of the initial web query (*XTF » Fundamental Concepts*, n.d.).

Analyzing the Archive website, we find that the main page of the Brockden Brown archive resides on archive.php (which is located in the base directory). Upon accessing the webpage, the user is presented with various options such as searching for primary bibliography and secondary bibliography. When a search for a primary bibliography is made, a file located on /archive/process_search.php is called. The process_search.php file then takes the fields provided on the web request, assigns them to variables, and then redirects the user to brockdenbrown.cah.ucf.edu/xtf3/search?... with the search arguments present after the question marker. Since the request isn't coming from the server itself, the request is visible from the user's browser with the response coming directly from the XTF server to the user's browser.

Exist-DB will be our database solution chosen by our sponsors for our project. Exist-DB is an open-source NoSQL database, built on XML technology. It is effectively used to store, index, and retrieve XML files. It communicates through the use of REST APIs which is what we will use to retrieve information (Meier et al., n.d.).

We explored the reason for the change from XTF to Exist-DB. We inquired our sponsors to better understand if they had any pain points and gripes while utilizing and maintaining XTF that we could address when we implement our solution to utilize Exist-DB. We learned that the reason for the switch was twofold. On one hand, their existing technical support contract for XTF was about to expire. More precisely, this technical support is set to expire in May of 2025 which would force them to look for a different solution. On the other hand, there is an existing project using exist-DB at UCF. The Johnson's Dictionary project utilizes Exit-DB to solve their database storage and indexing needs. This meant that our sponsors interacted

frequently with this solution and have an existing understanding and familiarity with this software. The decision to move Brockden Brown's Archive to Exist-DB was heavily influenced by the fact they had an existing degree of proficiency in the software and would be able to better maintain it.

Indexes on Exist-DB - Rodrigo

To have a working Exist-DB instance, we had to configure indexing. This will allow requests to be checked against an index before retrieving from the database. There are 7 different indexing options for Exist-DB. They are Structural Indexes, xml:id Index, New Range Indexes, Full Text Indexes, NGram Indexes, Legacy Range Indexes, and Spatial Indexes. (*EXIST-DB Documentation*, n.d.) We were advised by our sponsors that Structural Indexes are the most straightforward and that another project has been known to use Full Text Indexes in their project, by integrating Lucene into the XQuery engine. We researched this matter to decide which indexing solution will work best for our project.

Indexes aren't created through the use of commands, but rather the creation of files. The conf.xml file can be created in the /db/system/config directory. Indexes can be configured in a collection by collection basis. So different collections in an Exist-DB installation can use different indexing methodologies. This is done by creating subfolders and the configuration file inside the subfolder such as /db/system/config/db/brockdenbrown/collection.xconf. If that file doesn't exist for a specific collection, the system wide default configuration will take precedence. This distinction is important to keep in mind, especially if the production Exist-DB instance is shared among different projects. At that point, we'd have to make sure to only configure indexing on our specific

project's configuration subdirectory, rather than the default configuration. Another aspect to keep in mind is the need to declare namespaces. For this project, since our XML files are encoded using TEI, in our index configuration files, we'd need to declare the TEI namescape. Another important aspect of index configuration is that it is automatically updated if changes are made to the database. This means that once we configure indexing to a collection, future collection modification events will be indexed based on our created configuration. The exception is when a new indexing definition is added to an existing collection. The new index will only apply to new data, unless a reindexing is done by running the XQuery "xmldb:reindex('/db/collectionName')". There are also other methods to trigger a reindexing operation. We will need to keep this in mind as we work with various indexing implementations. When changing configuration of our indexing implementation, or changing the indexing methodology, we will need to re-make the index to make sure it works as intended (*EXIST-DB Documentation*, n.d.).

We began our research with structural indexes. Structural indexes are created and maintained automatically by exist-DB. It tracks tags, attributes and structures for all XML documents in the collection with which it is configured. It works by mapping every element and attribute to a list of pairs, such as <documentId, nodeId>. This list of pairs is then used to resolve XQueries that would eventually be received from our web-server back-end. This index is stored in a file named structure.dbx. (*EXIST-DB Documentation*, n.d.) Due to the fact that this indexing approach is automatically created, and no further configuration is required, it can easily be considered the easiest indexing solution. For this reason, this will be our baseline comparison which we will initially use. We plan to use this indexing methodology which would require little work, changing to a different method

if we encounter issues when implementing and testing our project implementation.

The next indexing method we researched was Full Text index, as we were informed there was a project that was currently using it. For the Charles Brockden Brown project, we'll need to do various complex queries. These searches can be based on various criterias such as keyword, genre, title, date range and so on. This brings two areas that we will need to pay attention to. The first one will be run time and efficiency. For the sake of the user experience, we don't want each query to last dozens of seconds of even minutes, so we need to be especially careful to make sure our querying process is efficient. The other aspect that we needed to pay attention to is capability, as we had to make sure that we are able to replicate the previous querying capabilities, which were linked to a different database system, on Exist-DB. We had to implement some additional logic into our indexing method in order to make this work, as there were some portions of it that were not being translated properly. Full Text indexing gives us additional capabilities and options as needed. Full Text index is based on Apache Lucene; it's a module that can be added to the indexing configuration in the collection.xconf file. Once added, subsequent database modification events will notify the added plug-in, meaning that we wouldn't need to manually re-index to keep it up-to-date, unless we wish to re-index the previously existing data. With Lucene, we can then create indexes/fields based on elements in an XML document. This can be done with the statement <text qname="nameOfXmlElement">. These new fields can then be queried by using ft:query, the name of the field, and the value that is being searched. With this indexing methodology, we can also create analyzer classes. Analyzers can be used to configure how text is tokenized and handled, which can be used in the case we need to further modify how Exist-DB indexes to

achieve previous functionality, such as keyword searching (*EXIST-DB Documentation*, n.d.-b).

We also did research on other indexing methodologies based on Range indexing. Legacy Range index works on solving the issue that without range index, comparison operators like greater than and less than default to "brute-force" querying method. Brute-force querying refers to looking through all nodes, casting the value to the query value type, then performing the comparison. The Legacy Range index allows queries to precast data of the data type configured in the indexing configuration. Subsequent queries can then easily refer back to the index and leverage the index to be more efficient. This implementation has been made obsolete by the Range index (*EXIST-DB Documentation*, n.d.-c). The new Range index set out to solve several aspects. For one, the new Range index allows the rewriting of expressions to help with concurrency and efficiency by limiting the amount of values that are queried. Another aspect that the Range index set out to fix was the reliance on collections, which meant that as collections grew, updating and removing documents became more and more time consuming. The new Range Index addressed this by organizing itself based on each document/node, rather than collections. Another aspect that was improved was through the use of Lucene rather than B+- tree. This increased the efficiency of the indexing (*EXIST-DB Documentation*, n.d.-d). Between the Legacy Range index and the New Range Index, we found the New Range indexing to have clear advantages. Compared to the Full Text Indexing, we found it to be simpler to configure but less powerful on the amount of flexibility it provides.

We also did research on NGram and Spatial indexing. NGram Indexing works best for exact substring searches. That's the case because it specifies

nodes and attributes to be split into tokens based on the value of N which can be configured and is 3 by default. This works better for the specified use case in comparison to other indexing methodologies due to the reliance by Lucene on whitespace to separate tokens. In cases where searches cannot be easily tokenized based on whitespace, NGram would be best. We can consider using this Indexing methodology if we encounter issues in our KeyWord search implementations due to limitations in the Lucene technology. We also looked into Spatial Indexes. We learned that it's still a work in progress which would lead us to steer away from this implementation. It works based on spatial geometries derived from the Geography Markup Language (GML). It stores Well-known Binary (WKB) into a JDBC database (Java Database Connectivity). Due to the experimental nature of this indexing method, combined with the high complexity of this implementation, we have decided not to implement this solution in our project (*EXIST-DB Documentation*, n.d.).

Exist-DB Stretch Goal

One of our stretch goals that was implemented was a way to set up a system for files to be uploaded and updated to the database, as previously the sponsors had to manually copy files to the server using a file transfer utility. After performing the copy, they also had to modify some pages as needed. All this procedure is outlined on a 3-page instruction document, so one of our goals was to make this process easier. We were able to accomplish that.

Initially, we considered creating a login system for this matter, which was later discarded. There are various considerations when thinking about

creating a login system. For instance, at that point, we would also analyzed the need of an additional database that would be used to store usernames and passwords. This would've added to the complexity of the project by introducing an additional system that our project would have to communicate with. In addition to that, we would've also needed to learn an additional technology stack, such as MySQL in case of using a SQL database to implement safe and efficient username and password checking, as well as user creating. We would've also needed to be careful to implement the system in a safe manner. Some factors that we would've needed to keep in mind include not storing passwords in plain-text, having all communication be encrypted, enforcing password complexity requirements, supporting multi-factor authentication among others. The use of cookies to keep track of session information was also considered, however that would've led to additional steps to be taken to fulfill accessibility requirements.

Taking in consideration the amount of features we would need to implement to allow our sponsors to conveniently manage files in the databases, we explored feasible alternatives in case we run out of time implementing our Minimum Viable Product (MVP).Since we did not have time to implement the system that was originally considered for the database files management, we decided to create some documentation explaining the process with which files can be added, removed and modified, explaining the steps our sponsors would need to take to accomplish that, which was the process outlined above. As mentioned, rather than manually interfacing with the Linux operating system of the back-end server, and manually modifying the files in the file system (such as copying files over with a transfer protocol, deleting files with the usage of the "rm" utility, or modifying files with a file editing tool such as "nano" or "vi"), our sponsors will interface with our Exist-DB installation. We believe performing these tasks on Exist-DB

will be easier. For one, Exist-DB has its own authentication mechanism. Our sponsors can authenticate to Exist-DB's web interface, and perform administrative tasks from it. For instance, by choosing the "file" dropdown and going to "manage", our sponsors would have the option to easily navigate the files stored in Exist-DB as well as upload, delete and organize them as they see fit. In the current implementation, our sponsors shared that they need to manually modify various files in the back-end server for a file modification to be properly reflected into the website. In our implementation of the website, we strived to streamline this process as much as possible.

Another stretch goal we had on our project, more specifically to our back-end implementation, relates to an activity that our sponsors support. Our sponsors are responsible for managing our website in the times to come now that our project is done. This website is focused on the work of Charles Brockden Brown which we came to learn is a major focus for Dr. Kamrath. Once a semester, the professor chooses one of Brockden Brown's literature for their class to transcribe into the TEI format so they can practice and learn its intricacies. Since this is a graded activity, the professor generally asks our sponsors to temporarily remove the chosen literature work from the Charles Brockden Brown archive to prevent students from accessing it during the activity and being exposed to the "correct answer". As explained above, the process of managing files of the current implementation of the website was quite extensive. Our sponsors, at the time, requested us to consider implementing a feature to make it easier to hide a file for a period before easily restoring it. This feature was implemented and documented through the following process:

- In order to hide a file, you have to create a folder outside the collection, for instance, if the collection is on db/apps/brockdenbrown/data, create a folder in db/apps/backup, or db/apps/hide, and then move the file you want to upload to the database using exist-db's web UI. In order to do that, the user needs to go to file -> manage, navigate to the file in questions, choose the cut option, then paste to the backup/hide collection. After that, they have to go to the collection.xconf which configures the indexing, press save, and re-index the files. Lastly, confirm that the file is no longer accessible.

- If the file is moved to a subdirectory of the collection, such as db/apps/brockdenbrown/data/backup, then the website would still be able to read it

- If the file is simply renamed, the website would still read it

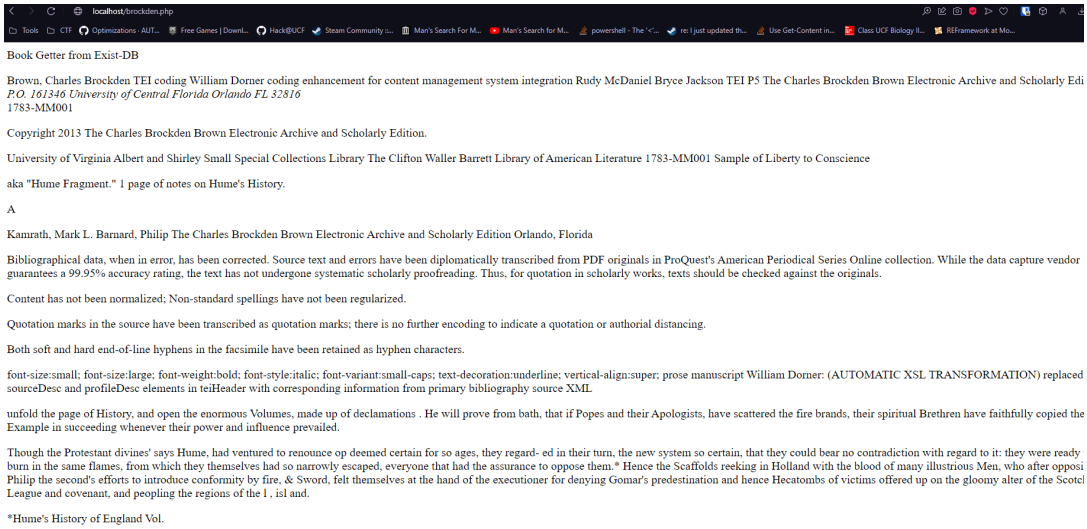
Exist-DB Prototype - Rodrigo

We worked on creating a proof of concept of a PHP back-end communicating with Exist-DB through API calls. In the process of creating this prototype, we learned some particular properties of TEI files and XQueries (the query language for Exist-DB). The architecture of this prototype consists of a Virtual Machine running on VirtualBox running RHEL which is a linux distribution which will match the operating system that will host our project in the production environment. In this virtual machine we have an Exist-DB installation in a container running on Podman as this is what our sponsors suggested we do, as running Exist-DB in a container

would make it easier to maintain according to our sponsors. We also installed an Apache instance to work as the back-end web server in the RHEL virtual machine, however, we also installed Apache on our Windows machine through a XAMPP application for ease of testing and to test communicating between two different machines through API calls.

Upon standing up the infrastructure that we would use for a proof of concept, we worked on understanding Exist-DB's query language XQuery (Contributors to Wikimedia projects, 2007). We performed various queries for practice. We began by utilizing sample xml files, and then retrieving documents and data inside those documents based on different tags. We then initially attempted to replicate these with the TEI encoded files from the Charles Brockden Brown Archive, however, we initially encountered errors. Upon further research, testing, and discussion with our sponsors, we learned that when working with TEI documents with XQuery, we need to indicate that the TEI namespace is being used. We use the following statement to accomplish that: `declare namespace tei="http://www.tei-c.org/na/1.0";' . Other than that, we also learned that the documents themselves had a dependency on another xml file named "cbb_encodingDesc.xml" which was initially zipped in the CHDR server and missing in our infrastructure. This caused all of our queries to not work despite declaring the proper namespace. This was finally found through troubleshooting together with our sponsors.

Upon resolving our obstacles with XQuery, we proceeded with creating a simple php page, which simply has a variable declaration with the EXist-DB API endpoint, and retrieved the contents of a specified file on the Exist-DB database as can be seen below.



Picture of back-end prototype where a document was retrieved from Exist-DB.

Data Migration - Rodrigo

Initially, having documentation for the full installation of the project on RHEL 9 was part of our MVP, however, after further discussions with the sponsor we decided that was not needed as we would have the website set up in production prior to the end of the project. We did make a README file within our GitHub repository that has essentially all the information regarding the website and all the features that were implemented on both back and front end, as well as instructions on how to run a clone of the website locally, in case another group takes over our project to implement additional features or perform maintenance of the website within a few years from now. One of the things that we had to research a lot about during that initial phase, where we still believed we had to write that documentation, was file transfer methods, so we could transfer any files to the target server as needed.

We performed research on various transfer utilities and protocols so we know the various methodologies we could've indicated in our documentation to assist in the transfer of data from one of the production servers to the target server hosted in Azure Cloud. We explored various protocols and tools such as SMB, FTP, SSH (SCP), SFTP, rsync and Azure File sync.

SMB, FTP, and SFTP are all file transfer protocols that involve a client and server communication pattern. SMB stands for Server message block and is a widely used protocol in Windows systems. In order to interface with a server and access and transfer files, first, a folder would need to be configured as sharing and Read and Write permissions would need to be assigned to the user account the client will be using. Once that is done, and TCP port 445 is open on the firewall, a Windows client, for instance, would be able to interface with the shared folder by “Mapping a Network Drive”, or “Using a Network Location.” This would then have the folder appear in the File Explorer folder, and allow files to be added to it by dragging and dropping. Once the Network Drive is mapped, the process of transferring files is very simple. This simplicity is a pro to this approach, many people are used to using the Windows file system, and allowing them to utilize that would make things simpler (Sheldon & Scarpati, 2021).

Having in mind that this project is run and will be run on a Linux distribution, however, does beg the question whether this simplicity of SMB on Windows is ported over to linux. Samba is an open source software that brings SMB file and print services to Linux. Configuring Samba entails downloading the packages for that software, and modifying the configuration files, indicating which folders we would want to have shared. This is fairly straightforward, but does require some knowledge of linux management. In

order to utilize SMB shares from linux, the smbclient utility can be used. At that point, the network share can be mounted into the operating system and files can be transferred (*What Is Samba?*, n.d.). Although possible to be used in the Linux operating system, the simplicity is lost making this not a worthwhile solution. This approach also have further downsides other than the time necessary to configure a Samba server in the Linux server, and then having the CHDR server or some other machine interface with it using smbclient. The other downsides include the risk of hosting an additional server in a production environment. Temporarily creating a server to then tear it down after usage provides an acceptable balance of risk management, but this might not be a desirable outcome.

We also observed similar findings when exploring the permutations of the FTP protocol with FTP, and SFTP. FTP stands for File Transfer Protocol, and uses TCP in an active connection or passive connection to interface with a server and perform the download or upload of files. FTP mainly works through ports 20,21 as well as some high ports that are used for connection, depending on whether active FTP or passive FTP is used (GeeksforGeeks, 2024a). SFTP stands for Secure File Transfer Protocol and was designed to replace FTP due to its improvements to security, as it transfers files by utilizing encryption to protect data. SFTP leverages SSH encryption algorithms for data protection, supporting usage of algorithms such as 3DES, Blowfish, AES, and more. There are other features of SFTP that make it desirable, such as the ability to pause and resume transfers to make sure intermittent network access doesn’t interrupt transfers. SFTP uses port 22 as it leverages the file transfer protocol (Spencer & Spencer, 2024).

Comparing FTP with SFTP makes it clear that SFTP would be the desired protocol to be used between the two. On one hand, SFTP uses

encryption which would help protect the data that is being transferred. On the other hand, the footprint on the server using SFTP is much smaller. SFTP leverages the SSH protocol, and uses port 22 (only one port), while FTP uses various depending on whether it is using an active or passive session. This difference would mean the increase in attack surface of the server using FTP would be greater than using SFTP. Furthermore, port 22 is often allowed to remote Linux servers for management reasons, so there is a high chance that no modifications to the host and network firewalls would be needed. This is contrasted to having to open ports 20,21, and other high ports for FTP, and opening port 445 for SMB. These actions would increase the attack surface, while using a protocol that is already likely allowed (SSH), makes no change to the attack surface and is a more desirable outcome.

We also performed further research on the scp command on linux and whether it would be a valid approach. SCP stands for Secure Copy Protocol in Linux and is a command-line tool that is used to securely copy files and directories between two locations, leveraging encryption over an SSH connection. In order to use this command, the user would need a valid set of credentials to the target's SSH server. The path to the files being downloaded or uploaded would also need to be provided. To upload files, the location to the files in the current system is provided, followed by the username to authenticate to the remote system, and then the remote system name or IP with the path to the location the file will reside. To download files, the order is swapped, with the information about the remote system coming first on the command, and the location the downloaded file will reside on the current system following after (McKay, 2023). These commands allow for the simple transfer of files between 2 machines and is a valid solution to the file transfer issue. This utility is also what we personally

used to transfer files from the CHDR server to our local machines and can be said to work.

Having used it before, we believed using the scp command would've been the most optimal choice, however, we came to find that SFTP offers many benefits. SFTP and scp are both command-line utilities that leverage TCP port 22 and employ SSH for security in the process of transferring files. They differ, however, in that SFTP can resume file transfers interrupted by a lost connection while scp does not have that capability. SFTP also has the ability of using directory management command, which would allow a user to list the directory contents, or even delete files on the remote device, while scp does not have that capability. This capability would make it easier to transfer files. With scp, you need to know the path to the file you have, and to the path you want to the file to reside, utilizing SFTP and its directory management commands would make this task easier (Cooper & Cooper, 2023).

Using SFTP is also very straightforward. In order to leverage SFTP to transfer files, the "sftp" command can be used followed by the username to authenticate to on the remote system, followed by the name, or IP address of the remote system. Utilizing a non-default SSH port is also supported in case the SSH server is configured to use a non-standard SSH port. Upon creating a connection with the command above, then the user can utilize a question mark "?" to view a list of available commands. Using that command, they can see how to navigate the directory with "cd", list directory contents with "ls", download files with the "get" command, and upload files with the "put" command. SFTP supports simple file manipulation, such as changing the owner of the file with "chown" and changing the file permissions with "chmod". Finally, the connection can be closed with "bye".

The ease of usage combined with the flexibility and various utilities makes SFTP one of our most recommended utilities (*How to Use SFTP to Securely Transfer Files With a Remote Server | DigitalOcean*, n.d.).

We also discovered the command rsync through our research. Rsync is a versatile command-line utility, used for synchronizing files and directories between two locations over a remote shell, or a remote rsync daemon. It provides the utility of providing incremental file transfer, transferring only differences between the source and the destination. This can help when wanting to check that all files are uploaded for our project. Rsync can be installed on RHEL (the target server operating system) by using the command "yum install rsync". Although rsync does not have a default method to ensure data security, it can be paired with SSH to establish a secure connection. This can be done with the rsync connection followed by ssh, the path to the local files, followed by the username and IP address (or DNS name) of the remote host (*How to Use RSYNC to Sync Local and Remote Directories | DigitalOcean*, n.d.). This will then transfer files that were noticed to be changed or found missing on the remote server. Although this tool would work, the need to manually pair this utility with SSH adds complexity to the process. Since this tool has the added benefit of checking if the file already exists and prevents the upload of duplicate data, we plan on adding this as a possible utility to transfer files to our documentation in addition to using SFTP and the scp command.

STARS - Rodrigo

STARS is a platform that UCF's students and staff can use to share their work with the rest of the world. It stands for Showcase of Text, Archives, Research & Scholarship. It currently stores 164,583 papers from

958 distinct disciplines with download requests originating from all over the world, totaling up to 11,246,837 downloads over the existence of STARS and 2,129,106 downloads for the past year. With that being said, it is clear STARS is an important platform for the sharing of knowledge and distribution of historical papers and various research (*STARS - Showcase of Text, Archives, Research & Scholarship at UCF*, n.d.).

Keeping in mind its importance, initially, our MVP also included assisting with the migration of data from the Brockden Brown Archive to STARS, however, our help was ended up not being needed. Although we weren't going to be physically copying files from one server to another, we would've been responsible for arranging and formatting the files to an acceptable state to be submitted. We would've needed to generate a metadata manifest for all currently created XML files, which total up to around 1,000, as well as created a PDF file of each document image set. We would've also needed to create a PDF file of transformed XML for each image set.

Considering the large number of files and data we were going to work with, we explored alternatives on how to automate the work. Since we didn't fully understand the full depth of the task in front of us, we were unable to determine if there are ready-made tools to assist us in the process, or if this is something we would've needed to develop.

Our sponsors explained that they were in contact with the Librarian to get more specific information, so we received some additional clarification at that point that we needed to provide metadata information in a spreadsheet format. This meant to us us that we we needed to gather Metadata and format it into a Spreadsheet format. Some PowerShell or Python scripting

might've been useful to help with this task depending on how the metadata was generated. For PDF's that contain images, they needed to contain alt text. Further research was needed on how to accomplish this task and what level of alt text was needed.

Version Control and Collaboration - Rodrigo

We gained access to UCF's CHDR server, where we had access to the files that comprise the front-end, back-end as well as the XML files, pictures and pdfs used in Brockden Brown's Archive. We worked on copying those files over to our local machines through the use of winscp and scp. The total size of the whole project came to 16.5 GB, with 23,636 files. All of these files, and size greatly exceeds GitHub's size limit (*Repository Size Limits for GitHub.com*, n.d.).

GitHub recommends Git repositories remain less than 1 GB, with an upper limit of 5 GB being strongly recommended. Our project's 16.5 GB is outside of the recommended boundary set forth by GitHub and would cause issues. We also had issues with the per file size limit. GitHub does not allow the upload of files larger than 25MiB through the browser, they will give a warning if files larger than 50 MiB are uploaded, and outright block files larger than 100MiB. Sifting through our data, we found several files with sizes exceeding 25 MiB, and plenty others that surpassed 50 MiB, with a few that reached the 105 MiB mark (*Repository Size Limits for GitHub.com*, n.d).

We attempted to upload all files to GitHub, but due to the issues mentioned, we were unable to. We agreed as a team that version control in our project would be an important part of our workflow. Part of our team contract involved having GitHub in our workflow. We agreed as a team that

we would not merge code to our repository without first making a pull request for any changes, in order to help prevent conflicts. Having established the importance of having this tool for this project, we worked on finding solutions to our predicament.

We looked into the possibility of using GitLab as an alternative. We had previously seen GitLab being used by companies as a development platform, so we did research on that capability and whether it would resolve our situation and allow us to upload all of our data and version control it. GitLab works similar to GitHub, it also provides web-based Git repositories that include free open and private repositories. It also similarly has issue-following capabilities among others that would be helpful to our project. GitLab and GitHub differ in that GitLab is open-source and focuses more on the CI/CD pipeline and provides an all-in-one solution to the entire software development lifecycle, while GitHub focuses more on being versatile and robust with its tools and third party integration, with a larger developer community (GeeksforGeeks, 2024). Unfortunately GitLab's Software as a Service (Saas) has 10GiB of free storage for the Git repository (*Storage | GitLab*, n.d.). If this limit is exceeded, the repository is changed to a read-only project and no new changes to the project are allowed until more storage is purchased. This will not work as a solution for us. We need to be able to modify our code and we do not have the funds necessary to buy more storage and invest in GitLab's service.

Focusing on the fact that GitLab is open-source, we did research on whether that can be leveraged to solve our issue. We discovered that in a self-hosted instance of GitLab, the limits per project can be manually configured (Heddings, 2020). This would allow us to create a project with the full 16.5GB files and 23,636 files. This solution would resolve our

situation, however, self-hosting GitLab could become a tricky situation. If we self-hosted on our home network, with our personal computers, we would need to work on configuring a VPN endpoint on our home network, then share with our team the VPN configuration, at that point, the GitLab server would only be reachable through the use of a VPN. Using a VPN in this setup would become necessary for security reasons.

Exposing our personal computers to the network would open our privacy, personal information, and safety to great risks. In case of misconfigurations of the server, a vulnerability on GitLab's application, or a vulnerability in some other aspect of the underlying operating system of the server, a Threat Actor in the internet could exploit said vulnerability and potentially gain access to our computer. This would potentially expose our personal credentials, personal documents, files and much more in case of a breach scenario. Due to that, a VPN would be necessary to limit our exposure while allowing a degree of collaboration. The glaring downside of this solution includes the extensive time commitment and inconvenience of utilizing it. The time required to research, learn and configure a VPN server and a GitLab instance on a personal computer is extensive. We would also need to work on finding the appropriate storage and compute resources to host all the project files in our personal computer as well as managing the server and troubleshooting for potential failures.

Self-hosting a GitLab instance would also introduce a single source of failure. Errors with the server would lead to a halt in development progress until it is successfully troubleshot as our code is residing in one server. This also begs the risk of data loss. With limited resources in a personal computer scenario, there is a limited capability to perform backups and data redundancy of our project. In the case of a hard drive failure, we could

potentially lose all of our project. If this happened after a sizable amount of work was done, we might not have enough time to re-do the project to completion.

Lastly, the inconvenience of utilizing a self-hosted GitLab solution is also a great factor. Having to connect to a VPN to connect to our project introduces a great amount of friction to the act of development. This added time for development work would make collaboration less straightforward and more complicated. We would need to manage having to use Cisco's anyconnect to access the CHDR server, and potentially yet another VPN client to access a GitLab instance. With an already complex project, adding complexity to it is undesirable and would make the solution less likely to be frequently utilized by our development team.

Due to the time requirement to learn and set up a GitLab self-hosted instance, alongside a VPN for remote connectivity, combined with the added risks due to the single point of failure and added complexity and friction to the development process, self-hosting GitLab is not an ideal solution. Although the software does what we need it to do, the glaring downsides led us to reject this solution for our solution. Rather than focusing on learning a technology tangential to our project, we decided it would be more worthwhile to focus on learning the technologies used in our project such as PHP, HTML, JavaScript and so on.

Ultimately, after ruling out GitLab, we were left with the choice to forego version control and do our development in the CHDR server and our local test machines. We refused to completely forego version control and opted for a middle ground. Although our entire project cannot be version controlled, we can still version control parts of it. We decided to upload the

critical files, such as the php backend, the HTML files, as well as other files that are less than 25MiB to conform to GitHub's individual file upload limit. We also limited the total repository size, with our whole repository totaling 789 MB. This will ensure that we can live up to our team agreement of making a pull request prior to merging new code and gives us a platform to work on the files that will have a greater impact on our project.

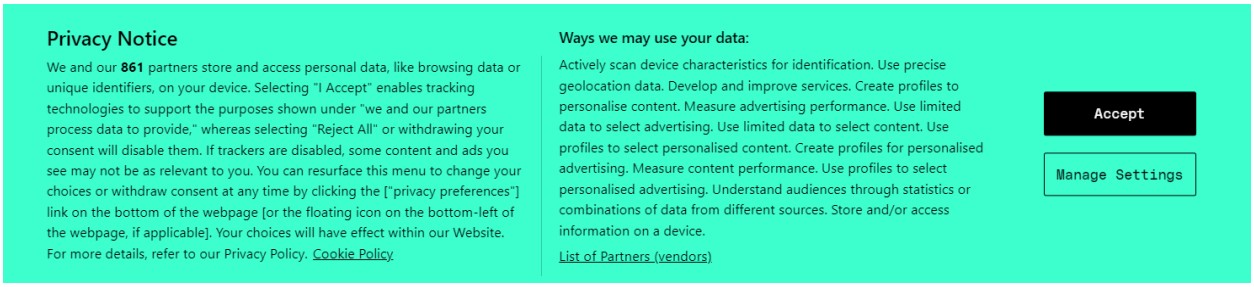
The downsides to uploading only part of our project to GitHub include making it more challenging to set it up, adding more steps to having our project, the Brockden Brown's Archive, functional in a new server. Since the updated code will reside in GitHub, and the large files will reside on the CHDR server as well as the current production environment, on RHEL 7, we would need two separate workflows to link those files.

GDPR Compliance - Gael

How does GDPR compliance work? - Gael

One of our main goals was to implement compliance with the General Data Protection Regulation (GDPR) law. This is a law enacted by the European Parliament, effective mainly in the European Union, but it applies to most websites due to the global aspect of the Web. According to their official website: "The General Data Protection Regulation (GDPR) is the toughest privacy and security law in the world. Though it was drafted and passed by the European Union (EU), it imposes obligations onto organizations anywhere, so long as they target or collect data related to people in the EU. The regulation was put into effect on May 25, 2018."

The GDPR's most visible effect on websites has been the addition of "Cookie Settings" to so many websites. Now, when a user visits a website, they'll be presented with something that looks like this:



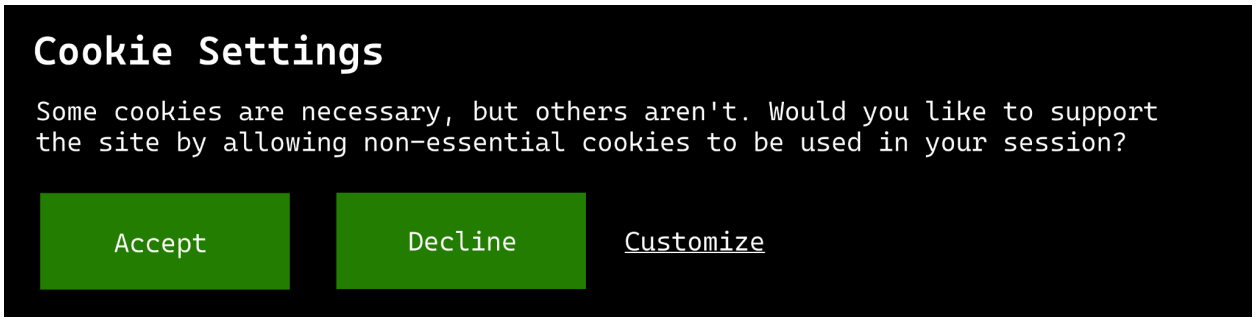
(source: The Verge)

On this website, the user is informed that they will be seeing ads on the site. It contains a link to the site's Cookie Policy, which specifies exactly how trackers are used to collect visitors' personal information and how that information may be used to serve more ads. Essentially, on a site like this, although the user is able to visit and enjoy the site's content for "free", in reality, they pay with their personal information.

How did the Archive become GDPR-compliant? - Gael

In a website like the Charles Brockden Brown Archive, though, most of this doesn't apply. Let's analyze how the GDPR was applied to our project.

Firstly, complying with GDPR means that any first-time user should be presented with a menu asking whether they authorize the web site to place non-essential cookies on their browser.



A cookie is a program that tracks the user's session to easily know that the same person who searched for a term is the one who navigated to the page for this search result.

An essential cookie is something that keeps track of basic information about a user's session. An example would be a cookie that tracks which pages the current user has visited. If the user were to click the Back button on their browser, the essential cookie in this case would direct them to the last page they visited on the website. Without essential cookies, the user's Search experience would be very cumbersome.

An example use case would be as follows:

1. The user visits the website and goes to the Search page.
2. The user enters a prompt and clicks Search.
3. The user is taken to their Search Results.
4. The user clicks on the first search result and is taken to the corresponding document.
5. The user clicks the Back button.

Without essential cookies, the user would be taken back to a blank Search page, without their results. On the other hand, with the use of cookies, the user will be taken back to the Search Results they had obtained,

and will now be able to continue searching through the results for the same Search they already performed.

What about non-essential cookies? There are many other cookies that track much more than just a user's session. Often, these will pick up on what parts of the site the user interacts with—which buttons they click, which menus they visit, etc.—and send them back to some server that will process this information and accumulate usage metrics about the website. However, this is a process that is most common on websites that place ads on their pages for profit. We do not plan to implement ads on our site, so there's really no reason for us to implement any cookies that involve this type of tracking.

The same goes for third-party cookies: these often are just feeding usage metrics (often tied to user-specific information, an invasion of privacy!) to third-party ad engines that are predominantly operated by Google. Once again, however, we will not be implementing these.

The only type of non-essential cookies we may implement are those that save a user's session after they've cleared their browser cache. This would require fingerprinting of some sort, perhaps based on the user's IP address.

Frontend

We were building the new frontend for the website in React-JS. This would allow us to create a modern, easy-to-maintain interface. Around the time of our third Sprint, we came to the decision to use React to build the frontend. Upon informing our Sponsors, they were delighted, and only requested that we code the frontend in a way that resembled the original.

That way, long-time users of the site wouldn’t be confused when they were faced with the new version.

Updated Current State: The Search Form Page - Dawn

In the Archive at the time, when a user went to search the archive, it brought them to a search form page. There were four tabs on this page: “Keyword,” “Advanced,” “Freeform,” and “Browse.”

With a keyword search, it was fairly self-explanatory, consisting of a search box with two buttons next to it for “Search” and “Clear.” Below the search box was a list of example searches, which increased the complexity and fine-grained the search despite not being under the advanced tab. The examples showed different ways a user could search: one keyword that searched all text and metadata in the Archive, multiple keywords separated by spaces to find documents containing both, keywords surrounded by quotation marks for exact phrases, a star symbol after a keyword to find partial matches, and a quotation mark after a keyword to find matches with one character appended.

Under the advanced tab, the layout was similar to the keyword tab, with search and clear buttons and examples below, but text and metadata were split into two sections. Fine-grain controls like quotations could still be used, but additional options were available. The "Entire Text" section had two radio buttons to switch between searching with "all of" the provided words (AND) or "any of" them (OR). Below that was an exclude field that functioned with AND logic. Proximity search was available via a dropdown, with values like none, 1-5, 10, and 20, although this feature appeared

nonfunctional during testing. Users could also limit searches to headings or citations instead of the full text.

The "Metadata" section included fields to specify a document's title, author, genre (which was a plain text field at the time), and file type. Document types included "All," "EAD," "HTML," "Word," "NLM," "PDF," "TEI," and "Text," though not all types were actively in use. Genre search was noted as difficult without a dropdown, and some types like "Word" documents were not actually present in the Archive. The form also allowed users to specify a date range, but it did not validate both fields being filled before submission. Lastly, searching for too many hits resulted in an unformatted error page instead of graceful handling.

The third tab was "Freeform." Functionally, freeform search was like an expanded keyword search with capabilities similar to regex, supporting NOT, OR, AND, field-specific search (title: and creator:), and parentheses.

"Browse" was the final tab, consisting of two links: viewing the Archive chronologically or alphabetically by title. Although simple and intuitive for new users, it was hard to find.

During development, the frontend behavior was enhanced. When a user clicked "Search" or chose browsing terms, React navigated from keywordsearch.js, advancedsearch.js, freeform.js, or browse.js to searchresults.js. This page took the search parameters passed from the form pages and queried the backend to retrieve results. The query ran independently every time the Search Results page reloaded, making it possible to share direct links to search results without requiring users to redo the search.

On the Search Results page itself, the query could be extended through the genre and date filter boxes without needing to perform a brand-new search.

Updated Current State: The Results Page - Dawn

When the user performed a search or chose to browse chronologically or by title, they were brought to the search results page. Searching or browsing chronologically shared the same layout; browsing by title used a different layout.

At the top was a header with information and functions. On the left side, it displayed the search query (“Search”), the number of results (“Results”), and a sort dropdown (“Sorted by:”) with a "Go!" button. It showed the type of search performed (e.g., "in keywords," "in full text," "in title," etc.), with an X next to each tag allowing users to remove that search condition. If only one remained, it directed users back to the Search Form page. Sorting options included relevance (default), title, accession number, author, publication date, and reverse date. Accession numbers followed a ‘year-identifier’ format and could include an 'L' to signify a letter.

On the right side were links and pagination: “Bookbag,” “RSS,” “Modify Search,” “New Search,” “Chronological Order,” and “Title.” The Bookbag link attempted to take users to their saved documents, though this functionality was deprecated later. The RSS link attempted to load an RSS feed but yielded no usable result.

The pagination showed pages in groups of five, using "..." to allow jumping backward or forward by five pages at a time.

Below the header was a sidebar on the left side. The sidebar displayed "Genre" and "Date" filters. Genre showed up to five genres initially with a "More" button to expand the list and a "Less" button to collapse it, although the placement of these toggles shifted inconsistently. Date allowed users to filter by individual years from the search results.

To the right of the sidebar were the search results themselves. Each result was numbered and displayed a collection of fields. Results varied, but common fields included Author, Title, Manuscript Location, Date, Genre, Accession Number, Matches, and Similar Items. Published articles also included fields like Article, Publication, Place of Publication, Publisher, and Publication Date. Some results contained a "Hybridity Note," clarifying the authorship or historical context of a piece.

Fields like Title and Matches were clickable, directing the user to view the document. Genres were also clickable to launch a new search. Clicking "Find" under Similar Items fetched up to five similar documents, although this view could not be closed without refreshing the page. Each result included an "Add" link for the Bookbag feature, which displayed "Failed to add!" due to deprecation.

Further updates improved the behavior on this page: when a user clicked to view a document, React navigated from searchresults.js to viewdocument.js, passing search parameters along. In viewdocument.js, three backend queries were made: one for the table of contents in the left

pane, one for the rendered document itself on the right, and one to retrieve the raw TEI XML file for download.

Viewdocument.js also used the original search parameters to highlight matched keywords inside the viewed document, enhancing search continuity.

Original Plan for Desktop Frontend - Dawn

Despite the fact that our goal was to recreate the frontend of the Archive in a way that looked similar to the current one, there were some glaring issues with it overall such that the layout of elements had to be changed. The functionality of the website was difficult to wrap one's head around at a first glance, and that was a sign of design that needed work. It took our team a non-insubstantial amount of time to fully grasp how the Archive was laid out and functioned. That was why it was important that the interfaces for accessing the Archive were easy to understand, so people were not turned away by seeing it as complicated. But of course, it was also necessary to keep the delicate balance of resembling the original, such that it didn't become too unfamiliar for returning folks.

Originally, the plan for designing the website was with a modern outlook for something that used a visual style and elements seen commonly today, something like material design being in mind. However, this did not get very far, as plans were changed. This explained what followed. Below was a design created in Figma based on the existing Search Results page. The actual visual style of it was very rough and was not what was in mind for a final version; rather, it was being used first to visualize the proposed layout instead of what was currently on the site. The idea was, instead of the user going to a Search Form page in order to search the Archive, and then

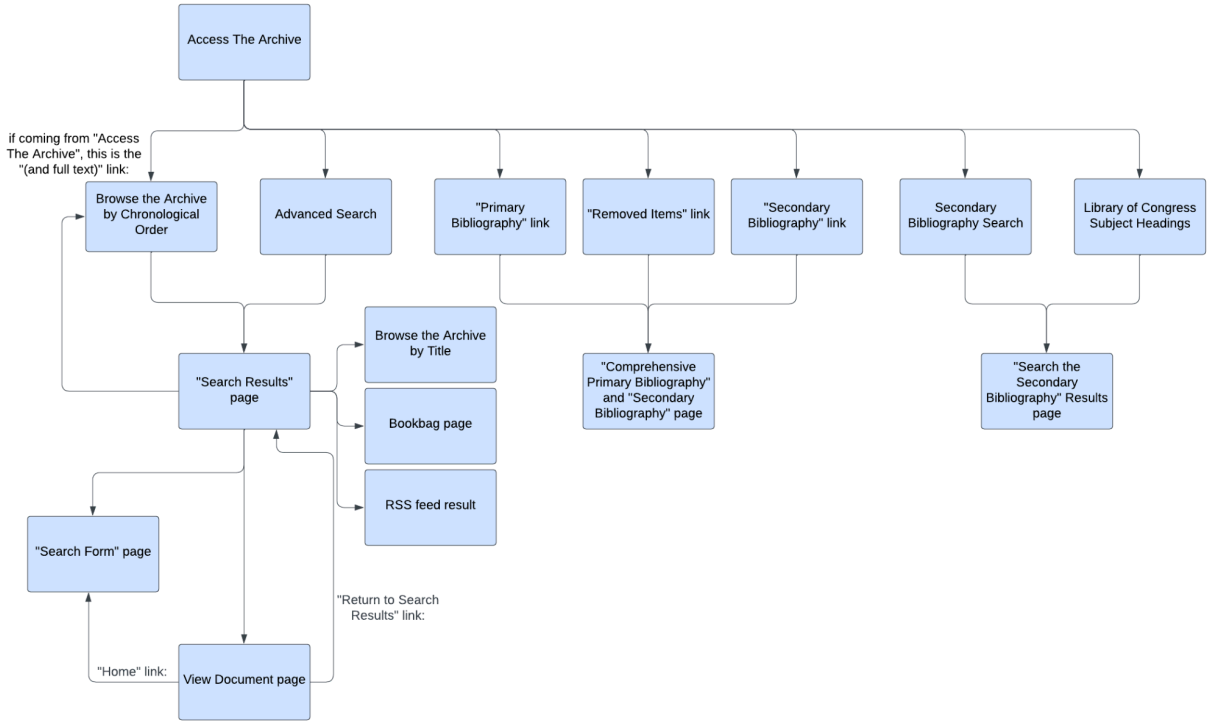
wanting to change their search, and having to constantly bounce back and forth between the two pages, to have it all on one page.

In this proposed layout, the keyword search was instead centered at the top of the page, as opposed to having its own tab on the Search Form page. Below, there were twice as many documents that could be displayed at one time with this layout, which made it easier to see more at once. Also, instead of the button to add a document to the bookbag being text at the top right of a document's result box, there were checkboxes next to the name of the document. These checkboxes allowed users to add as many as they selected into their bookbag by clicking the button above for "Add Selected to Bookbag," and then they could go to view the bookbag with the "View Bookbag" button next to it. On the left was the sidebar, which did exist currently; however, this had some added functionality. Advanced search and freeform search were now part of the sidebar, above genre and date, all of which could be minimized or maximized as necessary.

This was not shown in the Figma layout, but as opposed to being static, the sidebar was intended to follow the page as you scrolled down, as well as be hidden or resizable to allow more room for the content of the results if desired. Each of these fields should have been scrollable.

Some of these ideas were good for accessibility, and ended up being implemented. Because the bookbag function itself was axed from the project, as well as the RSS feed not pictured in this Figma design, those were not considered for the final design.

Proposed layout for the Search Results page made in Figma



UML flow diagram of the entire frontend of the Archive, excluding the header at the top with links to other parts of the website

Accessibility Guidelines on the Web - Dawn

The Web Content Accessibility Guidelines (WCAG) 2.1 guidelines were mentioned by our sponsors in meetings. These had five main pillars by which the guidelines went: perceivable, operable, understandable, robust, and conformance. UCF-affiliated projects such as this one were becoming required to follow this policy of WCAG AA, so it was good to do for more than one reason. Not only would it have increased the usability and accessibility of the Archive, but it would have complied with UCF's policies.

Conformance was less of a technical guideline itself than the others, in that the other four were the goals to reach with accessibility, while conformance focused on how to meet the actual formal requirements for WCAG compliance. The language used in the WCAG 2.1 guidelines was

mainly normative, meaning it had weight on what the guidelines were looking for. This was in contrast to some of the non-normative (also referred to as informative) content in the WCAG, which was essentially supporting material to help understanding of the requirements, like "diagrams, examples, and notes"; the introduction in the WCAG, for example, was non-normative. Conformance was addressed first, because within each of the guidelines specified for the four other sections of the WCAG, conformance level was mentioned.

There were three levels of conformance, each becoming more difficult for a web developer or team to satisfy as the accessibility of their website increased, such that a higher level might have been illogical to attempt to implement depending on the type of content, use case, and structure a website had. Level A conformance was the minimum conformance level necessary to meet the WCAG standards. All of the level A "Success Criteria" had to be met either directly on the web page in question, or with a "conforming alternate version"—meaning a separate page that mirrored the information and did conform to the standards could also be satisfactory if it was desired to have a non-conforming page. To meet level AA conformance, both level A and AA success criteria were required, and to meet level AAA conformance, level A, AA, and AAA success criteria were required. Both of these were also acceptable under the "conforming alternate version." One of the notes below this section on conformance levels in the WCAG mentioned that level AAA conformance was not recommended generally for every site, because there were cases where the content on the website simply could not be manipulated in a way that met all of the criteria.

Perceivable meant that all information presented had to be able to be shown to users in a way such that they could understand it. Alt-text for

media (anything not text) was a big part of this, to accommodate people with vision disabilities. On the website, this media needed to be able to change into different forms that people needed. When there was something that required user input, it had to have a name. A name was defined by the WCAG as text that software helping the user could see as something inside the content of the website. This was different from a label, which was a text alternative to some content to help the user understand what they were in front of. Labels were something every user could see, while names could be disabled from being shown if the user wasn't using assistive software. Often, a name and a label had the same content, but not always.

If the media was something that only showed up for a specified amount of time, then it had to have alt-text or something able to identify and describe it. One part of this timed media was if a video was a pre-recording and was only audio or only video, there had to be an alternative that correctly displayed information equal to what was being shown, such as text with the same timing as the video. Captions on pre-recorded videos as well as live videos were required, as well as an audio description of what the content of a video was generally. If there was media for an exam, something like a diagram that wouldn't be helpful if displayed as text, then at least a description of what it was generally had to be shown. If there was a "specific sensory experience"—such as WCAG's example, "a performance of a flute solo"—then alt text needed to describe what it was. If there was a CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) on the page that required media, then that had to be described with alt text, but there also needed to be an alternative CAPTCHA with a mode to allow people with varying disabilities to complete it. If the media was only for decoration and served no purpose, then accessibility software could safely exclude it.

Underneath perceivable, pages or content had to be adaptable, meaning they could be displayed in multiple ways without sacrificing the structure of a site or causing loss of information that would otherwise be available. The site had to be able to convey structure or relationship between elements in ways that allowed accessibility software to still understand them, like a list of items that still worked if someone read it using a screen reader, not just by how it looked visually. If the order of content on a page mattered, it had to be clear and usable by people and accessibility software. The site must not have relied solely on visual characteristics or sensory cues, as this could have made elements less accessible. The content on the page needed to work in both portrait (vertical) and landscape (horizontal) modes unless it had to be viewed a certain way. Some examples given in the WCAG where a specific orientation was necessary included things like “a bank check” and “a piano application.” Whenever there was an input field that collected user information, the site had to tell accessibility software what the field was for, so people could understand the purpose and interact with it correctly.

Also under perceivable was distinguishable. This meant making it easier for users to see and hear content, including having good contrast between text (the foreground) and background. When it came to the use of color, it was necessary not to rely on color alone to show important information. For example, color blindness might have made such content inaccessible. If audio started automatically and played for more than three seconds, there had to be a way to pause, stop, or control its volume. Text had to meet minimum contrast levels between itself and the background: regular text needed at least a 4.5:1 contrast ratio, while large text (like big

headings) needed at least a 3:1 contrast ratio. Some exceptions applied, like text on an inactive button or text in a logo or brand name.

Without losing readability or functionality, users needed to be able to resize text up to 200%, except for captions or images of text. Sites had to use actual text instead of images of text whenever possible, so users could adjust font size, color, and other settings as needed. Some exceptions included logos, and if the image of text was customizable by users, that also worked. Users needed to be able to view content on a page without needing to scroll in two directions (both up/down and left/right) when zoomed in — vertical scrolling was fine, but horizontal scrolling should have been avoided for narrow screen widths (down to 320 CSS pixels, about the width of a mobile screen at 400% zoom).

There were exceptions—for example, content requiring a two-dimensional layout, such as "games, presentations, data tables"—which were allowed to scroll both directions. Similarly to the text requirements, non-text content had to have enough contrast to stand out from its background, with at least a 3:1 contrast ratio. Text spacing (when a website used markup languages with text style properties like font, color, size, padding) needed to follow WCAG's suggested settings: line spacing at least 1.5 times the font size, paragraph spacing at least twice the font size, letter spacing at least 0.12 times the font size, and word spacing at least 0.16 times the font size. Extra content that appeared when a user hovered or focused on something had to be easy to interact with and dismiss. Specifically, users needed to be able to close it without moving the mouse or changing focus away from the window, and they had to be able to move the mouse over the new content without it disappearing.

For a website to be operable, all pieces of the user interface and navigation between them needed to be easy for users to operate. It had to be keyboard accessible—not just some parts of the website but the entire website needed to work via the keyboard—to make it accessible for people who could not use a mouse. Keyboard usage also could not expect specific timing, like key sequences within set time limits. The WCAG noted that this did not discourage providing support for other input methods like a mouse; it was simply unwise to allow only one input method. If a keyboard user could reach a part of the page, they also had to be able to leave it easily via keyboard. If navigation keys differed from standard ones like Tab or Arrow keys, users needed to be informed.

In order to be understandable, the user interface and information had to be clear and easy to understand. The default language of the website had to be set in the code so accessibility software could detect it and adjust as needed. If a section of a page used a different language, that had to be specified too, except for cases like "names, technical terms," and words that had become part of common usage.

A robust design meant that content needed to be built to work well with commonly used software like browsers and assistive technologies. This was what React was intended to assist with for the Archive.

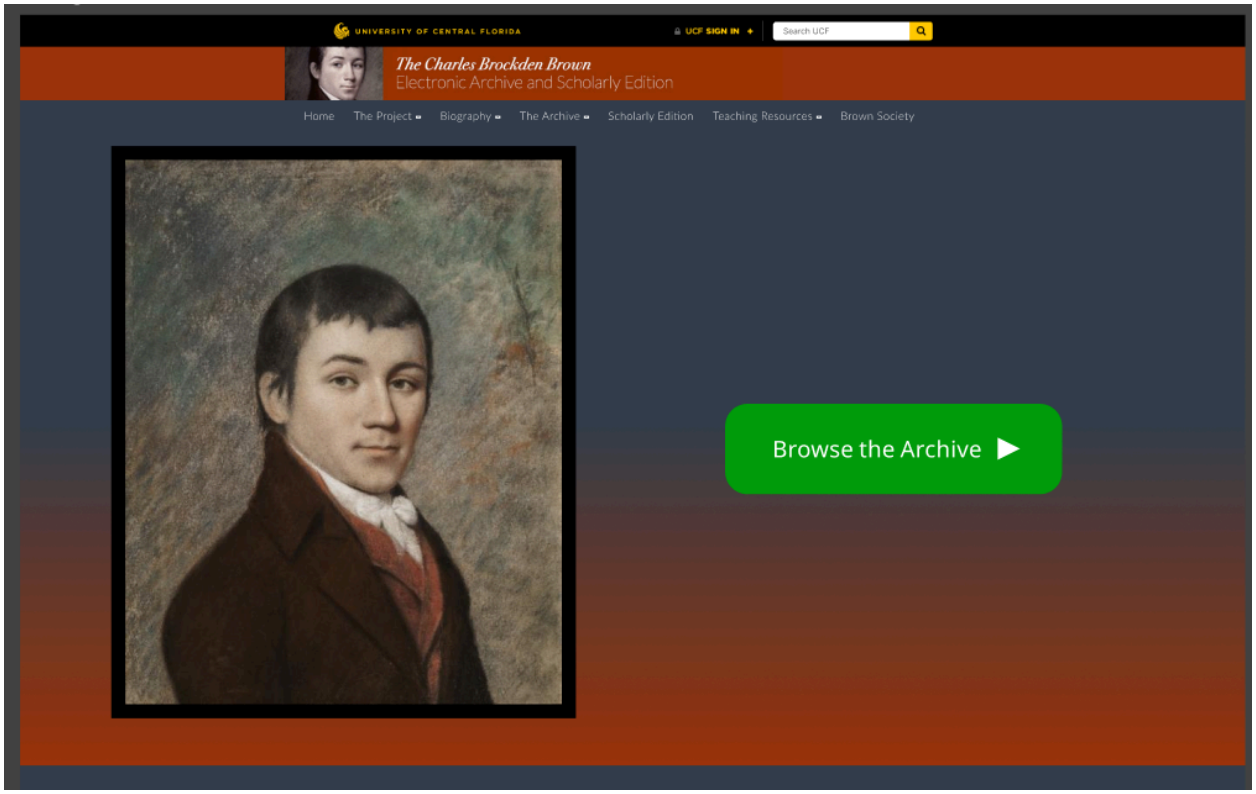
The Plan for Desktop Frontend - Dawn

A meeting with the sponsors decidedly changed our view on how to proceed. This was because of who primarily used the website, and how changing up elements might have been to their detriment. It was pointed out that the users browsing the archive were not too unfamiliar with the current design of the website and others like it due to how long they had

been around this space and what they were used to, so changing the design to be sleek and modern and completely upturn the layout of the website was not the goal. The main point of this project was to upgrade the archaic backend and frontend systems to make maintaining the archive long into the future feasible. So on the frontend side, the big focus was on accessibility.

At the time, the website was entirely PHP, as in the frontend was merely a collection of PHP files. Moving from plain PHP for everything to React on the frontend, while using PHP for its intended purpose as a backend, allowed our team to modernize the internal structure of the website.

The philosophy that the Archive should be more accessible in terms of legibility, even with assistive technologies to the side for a moment, led to a full attempt at a redesign of the desktop frontend in Figma. With everything in mind about the nature of the project and what the PI was looking for, it was thought that a simple reorganization of elements with a few tweaks to the user interface to make it easier to interact with would be the way to go. Unfortunately, even this was too far, the PI felt it was too different. Regardless, let us explore what this design was offering.



Top of the proposed main Archive page Figma design

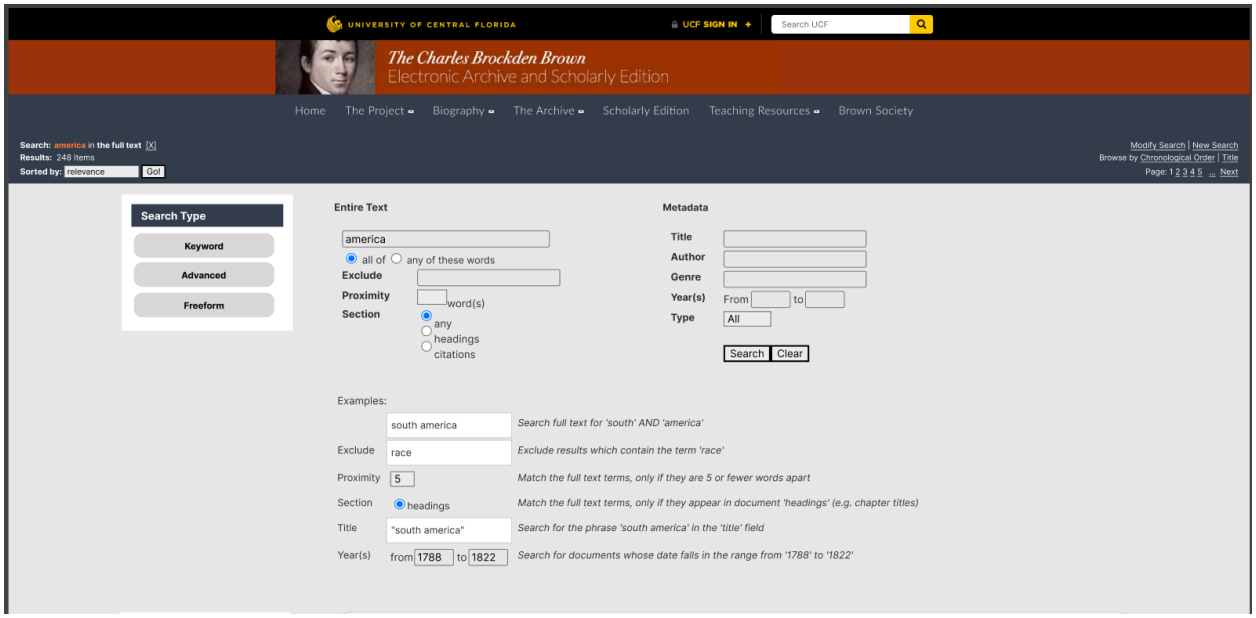
The idea with this design in mind is to make it as easy as possible to access the Archive. Originally, the link "(and full text)" is functionally the button to browse the archive, which can be, and frankly was, confusing for our team to comprehend. All of the little quirks about how the website works and each avenue you can take is not something that is readily apparent, it's something one has to learn, which is the sign of a design that needs work. So instead, having a picture of Charles Brockden Brown, with a big button front and center to browse the Archive, is a good way to guide new users to actually viewing the contents of the Archive, without leading them astray at first glance.

Bottom of the proposed main Archive page Figma design

The previous design may have been intimidating for those not acquainted with it. And so, below the Browse the Archive button, the sections on the current site were reorganized and condensed. At the time, there were multiple instances of repeated and redundant information about the Archive being displayed on the tabbed view of "Access the Archive" and "About the Archive." This way, it was all also in one place, the tabbed structure removed in favor of a simple scrolling page, with three sections — the first being the Browse the Archive button, and the others being

"Information about the Archive" and "Bibliographies." The section showing information about the Archive was just that: the information displayed in a manner that did not repeat itself when possible, keeping in mind the goal of retaining all of the necessary information written there so that this would only be an upgrade in terms of design, without causing regression in functionality or in what was available.

Below the information section was the section on bibliographies, which, unlike the information about the Archive, did have functionality besides only text and links inside that text. On the left side of this section was a description of the primary bibliography and secondary bibliography (which included the Library of Congress Subject Headings) that were available on the Archive, as well as a link to Daniel Edwards Kennedy's biography on Charles Brockden Brown and related resources. On the right side of this section were buttons to send the user to each place that the information on the left described — "Primary Bibliography," "Secondary Bibliography," and "Library of Congress Subject Headings" — and they were made to be easy to find as a user, just like the Browse the Archive button. Below those buttons were the input fields necessary to search through the secondary bibliography, with the same layout as the original, included here to keep all functionality in tact.



Top of the proposed Search Results page Figma design

This was the redesigned Search Results page which, like the first design, once again attempted to combine the Search Form page onto one place here. However, this time, instead of a full redesign, it was almost entirely the same as the original, with the goal of not making too much of a change. The first change had the top header stay consistent with the main Archive page, as all pages were intended to keep that the same. Beyond that, there were a couple of removals from the header for the Search Results page — the RSS Feed icon and button, as well as the "Bookbag" link. These features were decided by the Sponsors/PI to be cut from the project, due to those features not working for a tremendous amount of time already, with no requests or complaints about them being nonfunctional. So, that was why they were not shown here in the Figma design.

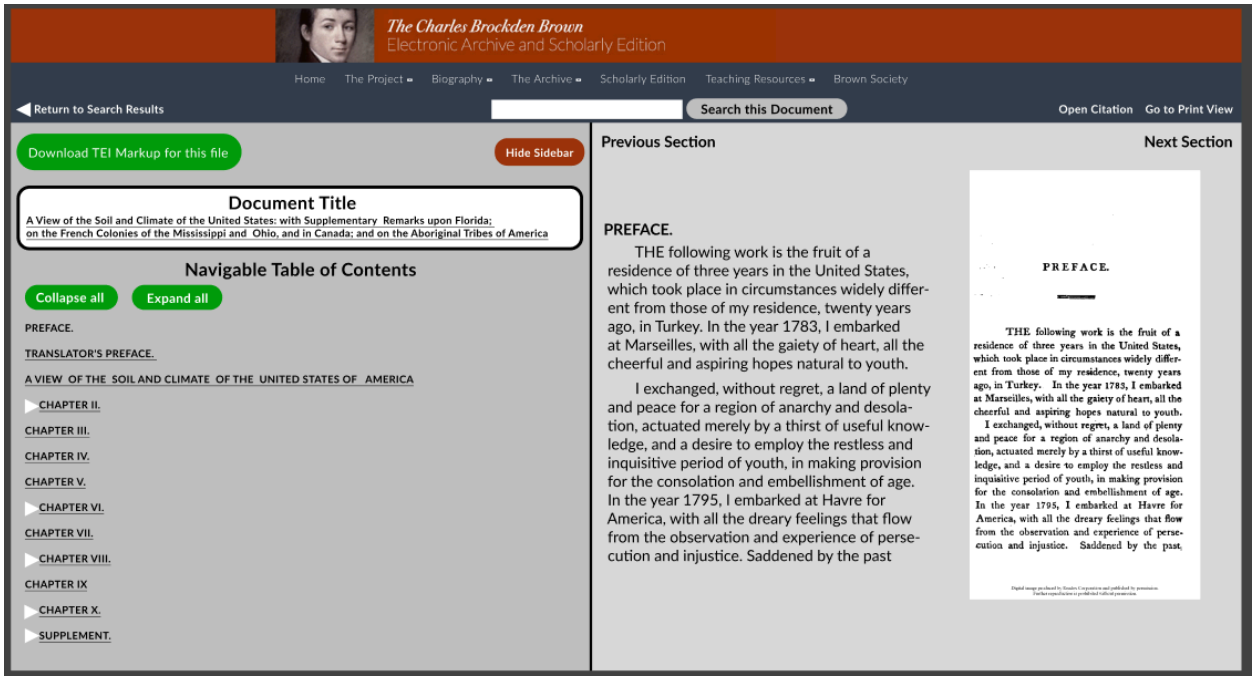
Below that, though, were the Search Form search fields and examples exactly the same, essentially copy and pasted. Although, it was chosen to recreate in the Figma the look of the Search Form when a search failed, as instead of failing on the Search Form itself, it ended up on the Search

Results page with the input fields garnering a different (arguably better) look than on the Search Form page normally. To the left of the input field(s) were the options that were on the Search Form page but in a collection of buttons instead of the tabbed layout, for “Keyword,” “Advanced,” and “Freeform” searches. This was made up of buttons instead of tabs because it followed a more understandable design, especially considering what was below this. And as the tab to browse the Archive that was normally on the Search Form page only consisted of two links to browse the Archive in either chronological order or by title — both also accessible from the Search Results page at the top — this was not added to the list of options in the Search Type box, because that was not necessarily a search type.

Bottom of the proposed Search Results page Figma design

Below the Search Type forms are the actual search results, which stay very largely the same, as to maintain the same design. One change that was done is to make each of the boxes that contain the results to have rounded corners, as the “Search Type”, “Genre”, and “Date” boxes on the left side of

the screen are all rectangular, it was a very small change but one that brings the results more to a forefront by showing a slight contrast in the design.



Proposed View Document page Figma design

This redesigned View Document page has the biggest changes made compared to the other pages previously mentioned. Below the consistent header at the top, are some options. Originally, this has “Home | Return to Search Results” on the left side. Presumably, this is supposed to be two links, one for returning to the main page of the Archive, and another to go back one page to the Search Results that led to viewing of this document. However, clicking the home link actually sends the user to the Search Form page, and return to search results does not have a clickable link, which means none of this functions as what seems to be intended. In this implementation, the link to go ‘home’ was not included, as that is a part of the consistent header, instead providing a functional link back to the Search Results page, which has an icon that is also consistent with the previous

links that lead to different parts of the Archive from the front page. To the right of that link is the search box for searching inside that document, which goes essentially unchanged. To the right of that are two links to view the citation for this particular document, as well as a print view of the document to print it if necessary. Clicking on citation opens a new window in the current version of the site with the citation, which may need to be changed in some way for the new version for accessibility concerns. Print view opens itself in the current browser tab, and does not make a new window unlike with the citation. These functions go unchanged, but there was also a "Choose Language" option there, which does not work in the current implementation, opening a new window with an error, which is why it was not included in the Figma design.

On the main part of the screen is the information about the current document on the left, and the document itself along with its transcription on the right. Both have different shades of grey as the background, although the pure white or a consistent color may be desired instead. The feature to click and drag the center of the screen left and right is retained, however now the left side of the screen also has the ability to function as a sidebar that the user can hide or show with the press of the button on the top right of that section. On the top left, is the button to download the TEI markup file for the current document. Originally this was at the bottom instead of the top, but it made sense to put it up there to be something front and center on each file, regardless of how long the information below it goes on for. And for what is below, instead of being without any sort of indication, the document title is now under a heading, and inside a box with rounded corners so that it's clear that is important information. Below that is the navigable table of contents, which has the "Collapse all" and "Expand all" buttons at the top as opposed to the bottom in the original for usability, as again this is easier to

access instead of requiring the user to scroll all the way down first. The table of contents is then scrollable, and sections that have expandable parts are indicated with an icon, although to be more accessibility friendly, this may need a better option with alternative text. On the right side where the actual document is, there is the ability to move to the previous or next section using the text at the top. This could use icons, however a different implementation of that is ideal, because the current one hides the “Previous” and “Next” text, which is an accessibility issue. So these will make it apparent when they are functional to be clicked. Originally, the document below is all shown in a vertical fashion, with the document image centered on the page, and followed by the transcription left justified below it. This can lead to some difficulty in viewing both in tandem, having to scroll up and down to look at each of the two pieces. That’s why in this design instead, the transcription is on the left side, with the corresponding image on the right, which allows a much smoother viewing experience.

React has support for working with accessibility in websites, and commonly is able to implement this with standard HTML. This is necessary so that assistive technology can parse the webpage and function properly. The things to keep in mind with this are: WAI-ARIA, semantic HTML, accessible forms, focus control, and more. WAI-ARIA is “Web Accessibility Initiative - Accessible Rich Internet Applications”, and React supports aria HTML attributes in JSX. Semantic HTML means it is important to prioritize semantic elements over generic ones, like using instead of <button>, as the button element is inherently semantic, has built-in keyboard support, and works better with screen readers. For accessible forms, when there is an input field, like with the Archive the Search Form input fields, each of those elements in the code need to have descriptive labels that are available to screen readers. In the case of an error, that must also be visible to screen

readers in some way. Focus control means that the website must be able to fully function with only the keyboard. One part of this is providing a way for users to skip headers and repeated things like navigation bars at the top of the screen in order to get to the main content of the page. Another piece to that is to use "landmark" elements like <header>, <nav>, <main>, <aside>, and others, to show the assistive technology how the page is structured and how it can navigate through these different regions.

By beginning the codebase with these good and necessary practices in mind for supporting assistive technologies at the start, the development can go smoothly, and lay a good foundation which doesn't need to be completely overhauled because this approach wasn't conceived. Once the main groundwork is laid for this support, features that allow options for users using assistive technology is ideal. As an example, Webcourses has an "Immersive Reader" mode, which implements a number of accessibility enhancements to allow the user to modify the display of text on the page to make it easier for them to read. This may be a gold standard, as it includes a lot of options. There are text preferences, which have changeable text size, spacing, fonts, and high contrast color themes. Reading preferences has a line focus feature which is essentially a built-in screen reader to the website, being able to focus on one, three, or five lines of text at a time, and also has the ability to translate text to different languages. What goes above and beyond though is the grammar options, which can show dots between syllables in words, as well as highlight words based on the part of speech it is. Something like this, granted with less options, is something in mind for the Archive.

From the section the WCAG were discussed, there are a multitude of requirements that were discussed which do not apply to the Charles

Brockden Brown Archive, due to what is being archived. The following addresses the guidelines which do not apply to the Archive by level, A Level first and AA level after. Guideline 1.2.1 is about pre-recorded audio-only and video-only content. The requirement is to have an alternative to those content types. However, the Archive does not have any audio or video content. There are no audio or video files present in the Archive because of the nature of the Archive being a historical and accurate account of documents from two centuries ago. Guideline 1.2.2 and 1.2.3 and 1.4.2 also fall into this category of not being applicable due to the content shown. 1.2.2 is for videos that have audio, to have captions on those videos, and 1.2.3 is also for videos with audio, but require either a transcript in text or some sort of description of the audio. Guideline 1.4.2 requires for any source of audio on the page to not be played automatically. Besides the fact that there is no audio content in the Archive, there are also no advertisements, especially ones that would cause this accessibility issue. Guideline 2.2.1 requires the website to have a way to make changes to time limited events that occur, like either turn off these time limits completely, or extend the time. However, the Archive does not have any such time limited events. Guideline 2.2.2 requires the website to have pause, stop, and hide functionality for content that moves and automatically updates, but there isn't any such content on the Archive, as it is quite low tech. Guideline 2.5.1 also does not apply to the Archive, as there are no gestures with the mouse. Guideline 2.5.4 is for elements on the website that react to motion, which the Archive does not have, and similarly with guideline 2.4.3, the Archive does not make use of page break locators.

In the Level AA guidelines now, guidelines 1.2.4 and 1.2.5 refer to adding captions and audio descriptions to live and pre-recorded videos respectively. These do not apply to the Archive. Guidelines 3.3.4 and 3.3.8

refer to processes that are important to keep in mind for when a site makes use of a user's personal data like with legal commitments and authentication, but these don't apply to the Archive as there is no authentication used or data of a user necessary to be collected or managed in some way.

In the previous state of the Charles Brockden Brown website, there are some guidelines in the WCAG which were already accounted for, and others which we needed to find solutions in the new React version of the site, but either way do apply to the Archive's contents. The following addresses the collection of guidelines that do apply to the Archive, following the list of guidelines by level, as in Level A first, followed by Level AA.

Guideline 1.1.1 was for non-text content, asserting that the website must have had alt text for something like a button that was only an icon, which explained what the icon did, such as searching. At the time, on the Archive, this was something that was already accounted for in almost every case, with few exceptions. This was because for buttons like "Search," "Clear," "Next," "Go!" and so on, these were all self-explanatory — as in, there was no ambiguity on what clicking that link or button would do, because there was no icon; it was the text itself already. The two cases where this was not true were on two different pages. On the Search Results page, in the header at the end of each search term, there was an underlined "[X]" which allowed the user to remove that search term from the search criteria they specified. This X served the same function that an icon would, which meant it did not self-identify what its function was, and needed either a modification to a word like "Remove," or alternate text in an accessibility mode. The second occurrence of an issue for this guideline was on the View Document page, where on the right half of the page there were two arrow

icons, one on the left side of that half and one on the right side. If there was no previous section to go to in the document, or no next section to go to, the left or right arrow icon respectively became greyed out, instead of colored. However, when this icon was greyed out, the “Previous” or “Next” text was not visible. This meant clicking the icon did nothing, and as there was no text while it was greyed out to convey this to users, this created an accessibility issue, which could be resolved by giving the greyed-out icon alt text.

Guideline 1.3.1 was about how the code behind the website needed to keep in mind how assistive technology looked for ways to interpret the content, structure, and relationships between elements on the website. For content and structure, this meant things like text and images and other elements should have had metadata like alt text for images, correct heading tags for structure inside the HTML, labels explicitly linked to their corresponding input form fields, and tables with the correct header cell attributes, among other things. The relationship between all of these allowed screen readers to identify these things as valid and use that information to better function. Guideline 1.3.2 referred to the assertion that the website must have shown content in a meaningful order to the user. This was to say, it was shown in a way that even without any CSS, the content had a proper order. The Archive did follow this already to some extent, with keyboard controls going down the top navigation bar first, then the left side of the page followed by the right, but the structure might have needed tweaking.

Guidelines 1.3.3 and 1.4.1, which were to not rely on things like sound, positioning, or color to convey something, were simple to follow for the Archive.

Guidelines 2.1.1 and 2.1.2 were about keyboard functionality, where everything on the website had to be accessible by the keyboard — both navigation throughout the page as well as ensuring that input was not limited to a specific timeframe. Nothing on the Archive was a timed event that required user input within a specific amount of time. The website itself was all pure PHP at the time; JavaScript was not involved. All of the pages on the Archive were keyboard accessible with the Tab key, with each element in the order one would expect for the keyboard to select. This was essentially complete, except for a few interactions that could have proved useful and should have been added to increase accessibility.

On the Search Results page, as opposed to the Tab key going through all of the header, then the genre and date boxes, and then finally to the list of results, it would have been ideal to have had a "Skip to Content" button on this page, allowing the user to press Enter from the top of the page to skip to the search results. This would have cut down the time needed to press Tab for each search query made if the website was being accessed entirely through keyboard input. This button for skipping to content would not have been visible to users generally, so as not to break the flow of the webpage when not necessary, but it should have been readily apparent to keyboard users.

The other interaction that might have needed an accessibility-focused option was on the View Document page, where it was possible to move the divider left and right to allow more space for one side or the other of the page. This interaction was not possible with the keyboard, as it required the user to hover over the divider with the mouse and then click and hold. Either having a specific entry for this inside an accessibility menu to move the divider using the keyboard, or enabling the keyboard to select a space in the

divider and then move it left and right with the arrow keys, were two considerations that might have worked.

Guideline 2.1.4 was easy to follow, as there were no single-key character shortcuts in the Archive, and the course of action was to not implement one. Guideline 2.3.1 required that there was no content on the website that flashed for any reason more than three times per second. This was not difficult to achieve with the Archive, as it already conformed to this policy. There were no instances of anything updating that frequently which would have caused flashing on the screen.

Guideline 2.4.1 required a function to skip repeated content on the site — content that could have made it more difficult to traverse if having to wade through it every time a different page was accessed. The previously mentioned “Skip to Content” button when discussing guidelines 2.1.1 and 2.1.2 was a solution for this. It also applied to the main Archive page and the View Document page, really any page, as the header at the top was going to be a constant between every page. At the time, the header changed slightly between pages and sometimes had bugs where certain elements were not visible, but this was going to be addressed by making it the same across all pages. This was good not just for consistency but also for accessibility, as things would not change unexpectedly.

Guideline 2.4.2 was to have clear titles on pages, which the Archive followed. Guidelines 2.4.3, 2.4.4, and 2.4.7, related to focus on elements and the purpose of links, were addressed or close to being addressed. Guideline 2.5.2, for having things happen only when a click was released and not when held down, was true of the current site.

Guidelines 3.1.1, 3.2.1, 3.2.2, and 3.2.6 were all either trivial for the new version of the Archive.

Guideline 4.1.1 simply required the code behind the website to have up-to-date HTML practices so that assistive technology could function, which the new version was intended to do. Guideline 4.1.2 was more about this code being able to function with even custom components, by having them correctly labeled, which was going to be addressed.

In the AA Level guidelines, guideline 1.4.5 asserted not to use images of text when possible, and while there were images of text in the Archive, they were of the documents themselves, which all had transcriptions attached to them. Therefore, as there were no images of text unnecessarily used instead of text itself, the Archive already conformed to this guideline.

When a user viewed a document in the Archive, both the original images being pulled from and the transcriptions of every image were available. This was fundamental to the Archive being what it was, but it also meant there was no functional alt text to display for these images, as the transcriptions served that purpose. Of course, the images still needed to have alt text, but instead of displaying the transcription twice, the alt text would let the user know information about which image they were looking at with something like a section identifier based on the table of contents on the left side of the screen.

To close this off, the remaining AA guidelines were repeats of previous guidelines with higher requirements but were going to be addressed by the things mentioned prior.

Images - Kiara

On the Charles Brockden Brown Archive, the different images were the heart and soul of the website. Inside the archive, every search result that came up had an associated scan of a document, which was stored as an XML file. These ranged from personal essays that Brown wrote, possible writing reviews he wrote, to small introductions he did for other essays. It was imperative for these documents to be secure and accessible in order to preserve his legacy. So, we planned to change how they currently worked to uphold his legacy.

Since all of the documents were uploaded to the website as images, having them be easily readable and accessible was of high importance. The images were small on the website in comparison to the surrounding text. To view the image at an appropriate size, clicking on any image on the website opened the image in a new tab. This was a problem for multiple reasons. First, since the sizes of the images could be large, it might have taken some time—especially depending on the hardware being used—for the image to fully load. Along with this, it also forced the user to bounce between two different tabs to follow the flow of the document they were previously looking at. Depending on the device used, this could have been even more of an inconvenience than it seemed—on an Android phone, this opened a tab not in its group, which required at least three taps to get to.

In order to bring the image function up to the standards of the rest of the website, we planned to remove the current function of opening the image in a new tab. In its place, we proposed implementing a zoom function for all images that would work no matter what device was being used. This would have fixed the two main issues stated above. Since the image would already be loaded when the tab it was on was loaded, no extra time would

be needed to load the image. It would also allow easy access to the tab the user was currently on without diverting any attention away. This implementation would have also allowed for a standard size of reading without having to change any part of the original image. This was to be done by having a set zoom-in size that would be standard across the entire website.

In line with increasing accessibility features on the website, we had started planning to add alternative text onto all of the pictures. Alternative text, more commonly known as alt text, was an attribute added to images that contained text to go along with them. This was key to accessibility because it could be very difficult for some people to read what was on an image, so having text in a place where either the user could read it or have a screen reader read it to them was necessary. Since the main text of the images was on the pages themselves, we wouldn't have had to include that. Instead, we would have included any extra text that was not on the page. Along with this, we would also have added any additional context that might have been valuable for the particular image.

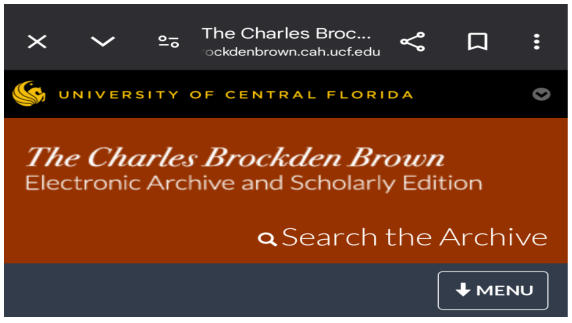
In a meeting we had with our sponsors, the conversation centered around accessibility—more specifically, the guidelines. Having alt text was important for accessibility, but since this was a website being updated, it was not an urgent matter to attend to. Furthermore, as a collective, the sponsors and the team agreed that adding alt text to every image was outside our scope. This meant that going forward, we would not be working on this.

A Look at the Mobile Viewing Problems Midway - Kiara

At the time, the website had enough functionality that one could use it on a mobile device if desired. The sponsors had expressed to us that they

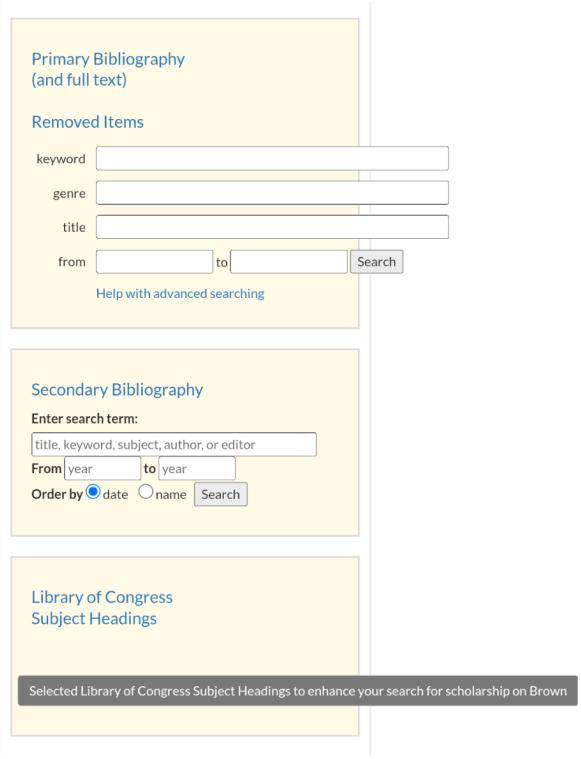
knew the mobile experience left a lot to be desired in respect to what it looked like compared to using a computer. So, this was something our sponsors had highly requested not just to continue doing, but to do so aesthetically and efficiently.

When entering the website, there was a very squished version of the one you would have seen while viewing it on a computer. The first header at the top got split in two, which read awkwardly, as the first half did not quite fill the top row, while the bottom "Search the Archive" was right-aligned. This led to blank space that was not appealing to look at. The menu converted into a dropdown menu on mobile. This particular implementation was hard on the eyes for two reasons. The first reason was that the color was seamless throughout each section. Compared to the computer, where the dropdown menu switched to a white box, it made it hard to differentiate the different sections. The other reason was that, along with the same color, there was no attempt to section them out at all. Under the "The Project" section, there was a heading and a line that looked out of place, but there was nothing further for any other heading. Lastly, there was a box surrounding text on this page. Mobile users did not see the full box — it was cut off on the right side of the page.



The awkwardly spaced menu, as mentioned above.

Once we arrived at the archive home page, the first real issues for mobile users arose. As on the arrival page, there was another box that got cut off for mobile users. However, on this page one could actually move the tab to the right to see it fully. This was because, as one continued down the page, the actual search function also extended past the page. This was true for both the "Primary Bibliography (and full text)" section and the "Library of Congress Subject Headings" section. For the "Primary Bibliography (and full text)" section, every single text box extended off the page. This included the actual search button. At the "Library of Congress Subject Headings" section, the one button they had that went to the LoC search page had all of the text on it, which had no line breaks, so it went very, very far off page.

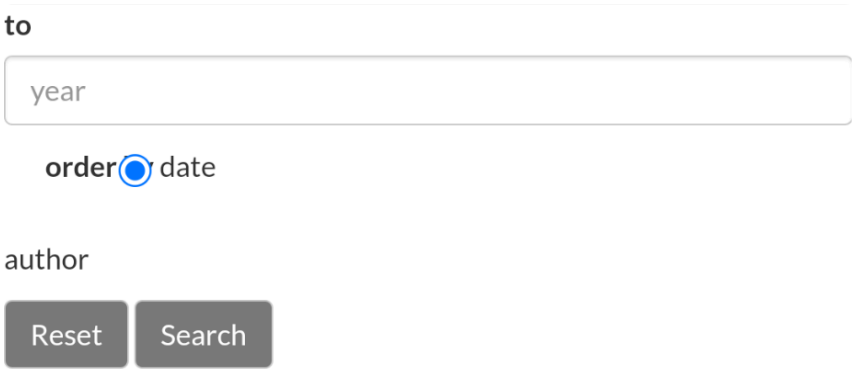


This screenshot shows how the different text elements extends outside of the normal page restraints.

On the Library of Congress Heading page, there was not much to talk about. At the top of the page where there were subheadings, the box used

to show which one you were on cut off on the left this time. Then, under the "Title" table, the indentation on the left was slightly less than the ones on the right.

The actual search using the Library of Congress headings was where we encountered real problems for mobile users. Once again, while using the search function, the same button from the previous page appeared, causing it to go way off the page. The first major problem for mobile users, though, was with how one would like the search to be ordered. On the computer version, there were two choices — date and author — which could be selected via a button menu. However, on mobile, this menu got completely messed up. There was only one button that showed up, which covered the 'date' text, with the 'author' text being a couple lines down. With the actual search, it was possible to get a match that very slightly went off page (and with very slightly meaning a couple pixels of a colon); however, the aforementioned button meant the page was messed up from the beginning. Also, the picture on the heading at the top of the page did not load in.



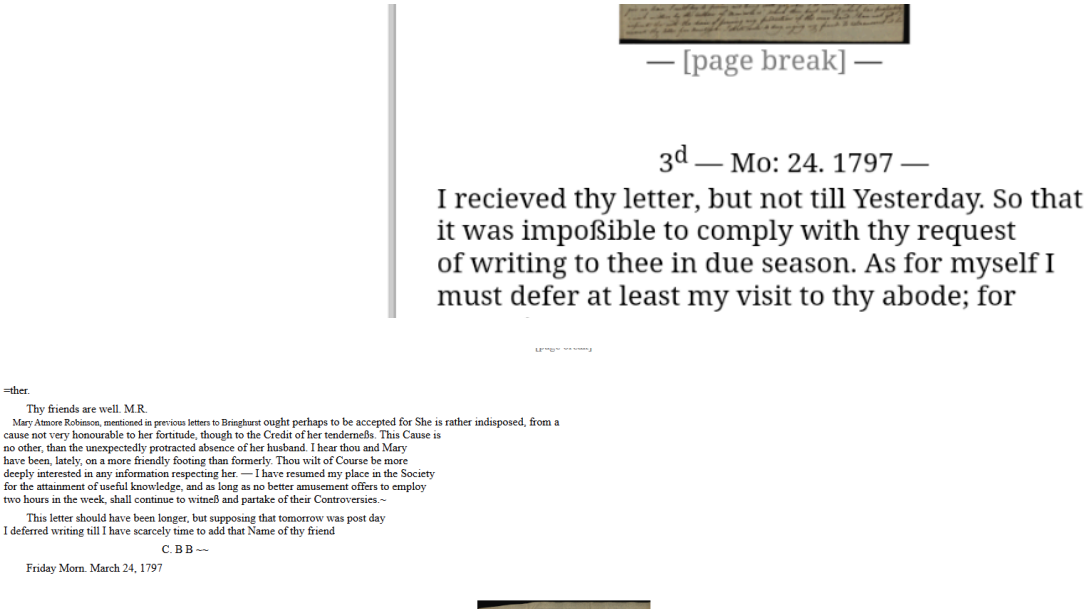
Screenshot of the messed up button menu which does not function on mobile devices.

On the basic search page, the problems only lay with the aesthetics. For starters, the header at the top extended slightly off the page. In the

advanced search page, which could be accessed via either creating a search that yielded no results or going to modify search at the top, the lines didn't match up cleanly. At the top, the headings had a slight left indentation to them. For the example at the bottom, the text boxes provided were slightly bigger than the rest of the text, which caused each consecutive line of text to get more and more misaligned. The page also looked awkward due to the footer not extending to the bottom of the page.

The search results page had the least issues. Like any of the other pages, it had the header which extended slightly off the page. Under the default view, that was the only problem. However, this was not the case if one browsed by title instead of chronologically. On certain letters (with seemingly no connections as to why), the font size increased by a sizable margin, but only for the results, not the subheadings (author, article, etc. — it was also inconsistent in the fact that sometimes one attribute would still be the same size). This caused misalignment for the actual search.

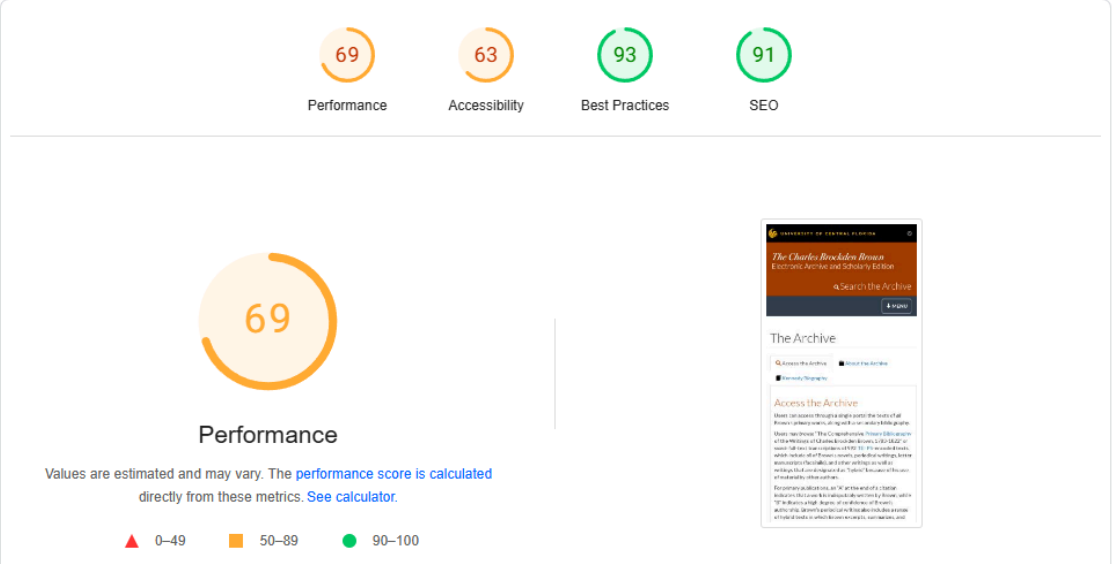
When on a page from the search results, once again the layout was where the problems resided. First off, the new header got squished, so the text box extended slightly below it. For the text on these pages, they already had set cutoffs for the text, which were independent of resolution size. So, it looked different on any screen. However, it was possible to have a narrow enough screen to have it line break further than it was programmed. This caused lines with one or two words to appear, making it practically unreadable.



Showing the difference in how the same text is displayed - top is mobile, bottom is computer

Further Insight With Google's Lighthouse - Kiara

Coming into this project, we had an idea of what accessibility was and what it would have meant to make the website more accessible. However, just having an idea of what to do was not enough for a project of this scale. So, while conducting research on how to make a mobile-friendly website, we found Google's own mobile friendly test - Lighthouse (https://developer.chrome.com/docs/lighthouse/overview/). This service ran a diagnostic on any page that one wished, and it gave scores based on different audits it ran. For the purposes of this section, we were primarily focusing on the 'Accessibility' score and audits, although the 'Performance' and 'Best Practices' scores and audits could also have shone a light on things to improve.

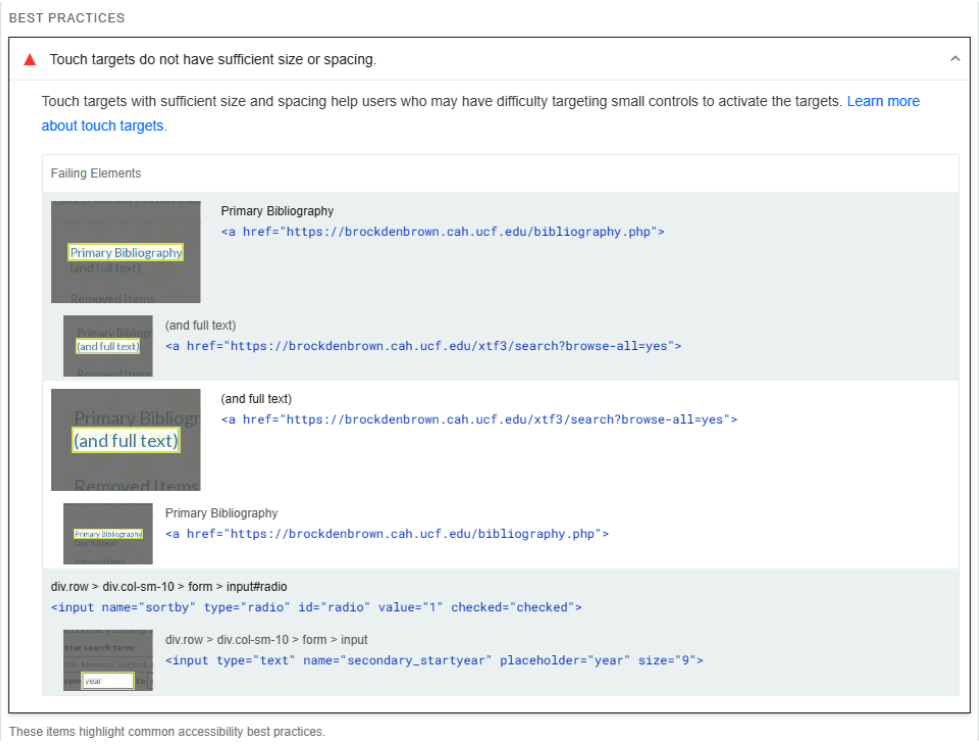


Pictured above is a report given by Lighthouse, specifically of URL: https://brockdenbrown.cah.ucf.edu/index.php.

With the starting page, Lighthouse agreed with us that the mismatched headers were a knock on accessibility. Having properly formatted headers helped with ease of navigation, along with being properly read by any sort of screen reader. This service did not touch on the dropdown menu, but still found problems with contrasting colors, specifically with links, elsewhere. It was imperative that users knew that there was a link to be clicked if desired, and for it to be known so one did not accidentally click on any links they would not have wanted to. It also brought up the fact that images opening up with a new link was problematic, once again because there was no discernable way for a new user to know this.

On the archive home page, once again indiscernible links were an issue. And with this, on this page as a whole, there was a problem with contrasting colors, which made it hard for just about anyone to read. This report brought up an aspect that we had not fully taken into account — how screen readers interacted with the page. It brought up that since the assorted form elements (which in this case would be the text boxes) did not

have any labels attached to them, it could have caused screen readers to not be able to properly read them out. The different things on the screen that one could touch were also not spaced out properly, which could have caused any user to have difficulties with being able to properly tap something, whether it be due to a small phone screen or any motor issues.

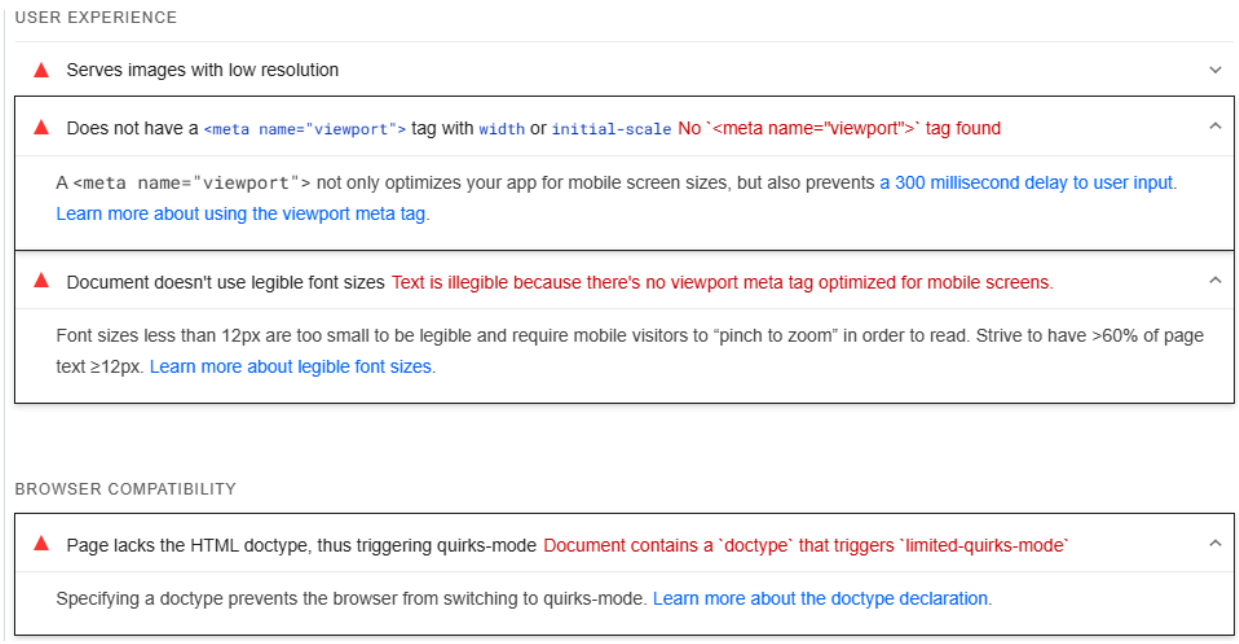


An example of a failed accessibility audit, specifically of URL: https://brockdenbrown.cah.ucf.edu/archive.php.

For Lighthouse, the Library of Congress headings page did very well in the accessibility department, only missing a couple of audits. A major one it did miss was an important one, however — it too had troubles with having a nice contrasting color foreground and background. This also applied to the actual search and search results that came from the Library of Congress headers as well.

When it came to the search page, this was when we saw that looking at the 'Best Practices' score might have been helpful when it came up as a

43. Like the archive home page, it also found that the items which one touched did not have enough spacing in between them. As well as the fact that once again, none of the form elements (which now included buttons and dropdown menus) had any labels, which was not good for screen readers. Looking at what the report said about the best practices, in regard to how it looked on a mobile view, it brought up the font size. There was not a viewport meta tag. What this meant was that instead of having a font size used specifically for mobile, it took what was used for the computer version of the website, then downscaled it to match the phone's resolution (Viewport meta tag). Having to make a user manually zoom in just to be able to read anything was a hassle — taking up much more time than needed by having to go back and forth for no reason.

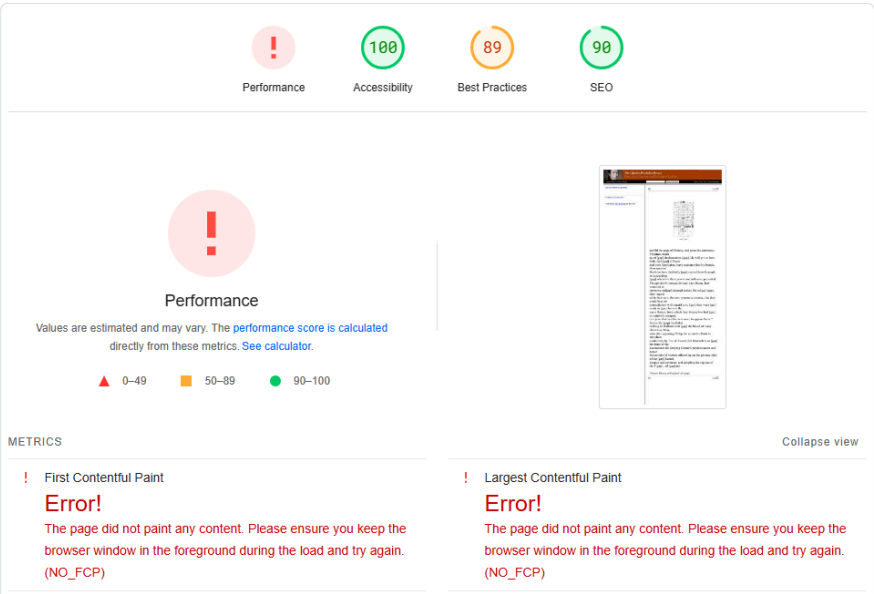


Multiple examples of best practice audits failing, specifically of URL: https://brockdenbrown.cah.ucf.edu/xtf3/search?text=a&text-join&text-exclude&text-prox&sectionType&title&creator&subject&year&year-max&type&smode=advanced.

The actual search results page, just like the search page, did fairly well in the accessibility department but failed in the same ways in the best

practices. It similarly did not have labels associated with the label elements, along with anything that required touching not being distinguished enough. And like earlier pages, the foreground and background colors were not deemed contrasting enough. One element we did not touch on with the last page but continued to be a problem with this one was that these pages lacked the HTML doctype — allowing it to go into 'quirks mode' (HTML doctype info). This was because this website had been around for so long that this was acceptable to use, but with the adoption of standard web practices, this became a challenge when loading (More info on the different modes). Since we were modernizing the backend of the website, this was also going to be changing. This would help keep the behaviors consistent and the ones desired by all users.

While trying to get a report of the webpage for any of the documents one clicked on via the results page, none worked properly. It either came up with the document completely failing to load, or like in the picture provided below, the service was not able to properly load it.



Pictured above is an error in a Lighthouse report.

One element that we had not mentioned, yet was present on nearly every page we ran a report on, was the fact that there was no ‘lang’ element on them. What this meant was that a screen reader would assume that the language of the page was whatever the screen reader default was, paying no attention to what words were on the page. This meant that if, for example, someone who had Spanish as their default language’s screen reader could have had trouble reading this English text. In the ‘Performance’ section, the shifting problems that we had caught previously popped up. So, not only did this cause the page to look disorganized and messy to read, it was also (majorly) causing the page to load slower as more network requests came in to do these multiple adjustments.

Looking at the Lighthouse reports gave us insight into features needed to reinforce upping accessibility. Previously, we were looking at it solely with our own eyes — what we saw that looked ridiculously out of place or hard to manage. With this integration, we were able to take a deeper look at why these things were happening, and at previously unthought-of things to look into as well (primarily with how exactly the screen reader interacted with the different page elements).

Mobile Viewing Planned Solutions - Kiara

Once we identified the different problems that existed with the mobile version of the website, we started thinking about some of our potential solutions for them. One thing to note, as stated before in the desktop plan, was that any redesign after that point was focused primarily on accessibility, with visuals being mainly just for consistency. However, that did not change any plans we had with the mobile issues we had taken note of, as when we referred to aesthetics previously, it was in line with having a mobile website

that works properly and as close to the computer version as possible, not changing it to look a certain way.

To start out with, we changed the layout to be consistent. There were multiple pages where the header used to extend itself past the page's limits, or break, causing the user to be able to move outside of the page's constraints, which was fixed by having the header attached to the surrounding elements and being fixed to the page's resolutions. Next, we also fixed the different box elements by making sure there's a check to see if it will go outside of the confines of the page - and either make it go to a separate line or have the info inside it go down to allow room for it. The dropdown header will also get an update to make it clear where one option to touch begins and where it ends. The document page was reworked to have the picture stay in a consistent spot no matter the resolution, with the text being fixed like this as well. Lastly, we looked into changing the colors of certain aspects of pages that make it hard to read, and find a more fit color that works with the page itself, and one that works with the eyes.

As said before, the initial idea for the mobile view was to come up with a way to make both the picture and the text for the documents be on the same page. This is to ensure that it is easily readable for any user on any device to still read, while still being true to the current website. Being true to the current website is crucial to those who run the website - they like how it currently is, so there cannot be any big changes to the aesthetic part. However, after a meeting with the sponsors, we both came upon a different solution that will make it even easier for those with lower resolution screens to read the text, and still have the pictures to accompany them.

For this, we decided on having some sort of switch on the page - this switch would toggle between wherever you are on the text to the image that accompanies it. This cleanly fixes the problem at hand. Viewing both text and images on mobile devices can make it so the material presented is not in the exact order that it was intended or in the way it was intended. As stated before, this is *very* important for those who own the website. This solution allows for the text to still be synced up with the image, as the functionality will be to toggle over to the document which matches up the current text the user is reading. This also allows for the text to be read more easily - accessibility being the key component to the project at hand - as it will now not have to share any space with the images, so the text won't have to fight for space.

With the search pages, we not only fixed how it works, but also made sure that the functionality was there as well. Making sure the button menu works for searching was a high priority, as the mobile website needs to do the same things the computer can. We changed the way it displays so it's all on one line, and that it fits the resolution instead of allowing it to overlap itself and go off line. The glitch that caused text to become enlarged on certain pages was also looked into, and once the root cause was found, it was quickly fixed.

The metadata on the pages was also looked at and fixed. We added the HTML type to the pages it was not already on, as that in itself may be behind some of the issues that are had while loading it, not only for mobile devices but for computers as well. Making sure that all text has the viewport meta tag that does not already is also integral to making sure that the text stays readable on mobile. Having working metadata is also extremely important in making sure that screen readers are able to properly read

what's on the page. Since the archive relies on its search function, making sure this is accessible is incredibly important. All text boxes and menus on these pages got their appropriate labels attached so that screen readers are able to properly pick up on them. This was also helpful on the backend side of things to do, as we, and anyone looking at it in the future, are now able to see exactly what they do and what they are for.

Image showing just how important meta viewport is for viewing purposes. Credit: https://cdn-benkb.nitrocdn.com/ZtlNGPuVDgGYZTtnpPrYcgJWctLyDuUr/assets/images/optimized/rev-972eb33/www.bruceclay.com/blog/wp-content/uploads/2018/01/2018-01-viewport-comparison.jpg

Dealing directly with a phone and not just a mobile resolution is important to fixing it as well. We have found a useful guide that will help us with exactly how the spacing should go, along with using intuition about where a person's fingers sit on their phone. So, we had to ensure that there was plenty of space for anybody to be able to tap on exactly what they wanted. This extends to both clicking on hyperlinks in text, along with any selection menus.

With all of the fixes we implemented, the mobile side of the Charles Brockden Brown website is now something that can actually be used. In its previous state, it was near unusable, as text was very inconsistent which can be hard to follow, along with features being shut out completely. Once implementation was completed, anyone was be able to search for their favorite CBB piece and view it anywhere, on the go.

Initial Website User Use-Case Diagram

Seen in the figure below will be the use-case diagram we were working with initially for a website user. It shows the key features that are needed in the website that we will be working on - browsing the archive directly, or doing various searches. From here, the user goes to either a bibliography page or the search results page. Then, they can either add a document to their bookbag, go to a feed compatible with RSS, or view any selected document.

This is useful to have to look at in order to have a better understanding of what a website user currently looks like.

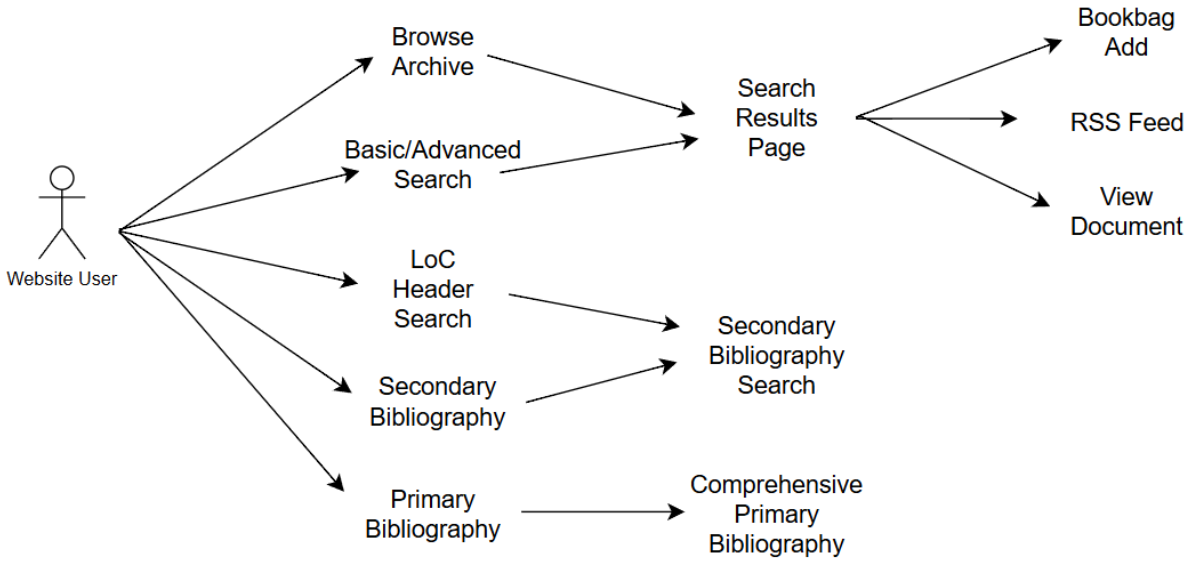


Figure ##. Shows the initial use case diagram for a website user.

Current Website User Use-Case Diagram

Seen in the figure below will be the use-case diagram we are currently working with for a website user. Since the initial case use diagram, there had been multiple meetings with the sponsors, in which conversation about the several bug fixes we were asked to fix was had. So, we had to change the case use diagram to reflect this. This meant that the bookbag and RSS feature from the search results page was removed. We also talked about how we will only be fixing the LoC Header Search as a Stretch Goal, but how it works would be the same for this use case diagram regardless.

This will be useful going forward to keep our focus on what needs to be done for a user point of view - as one can forget what it looks like for the user after working on the behind the scenes for so long.

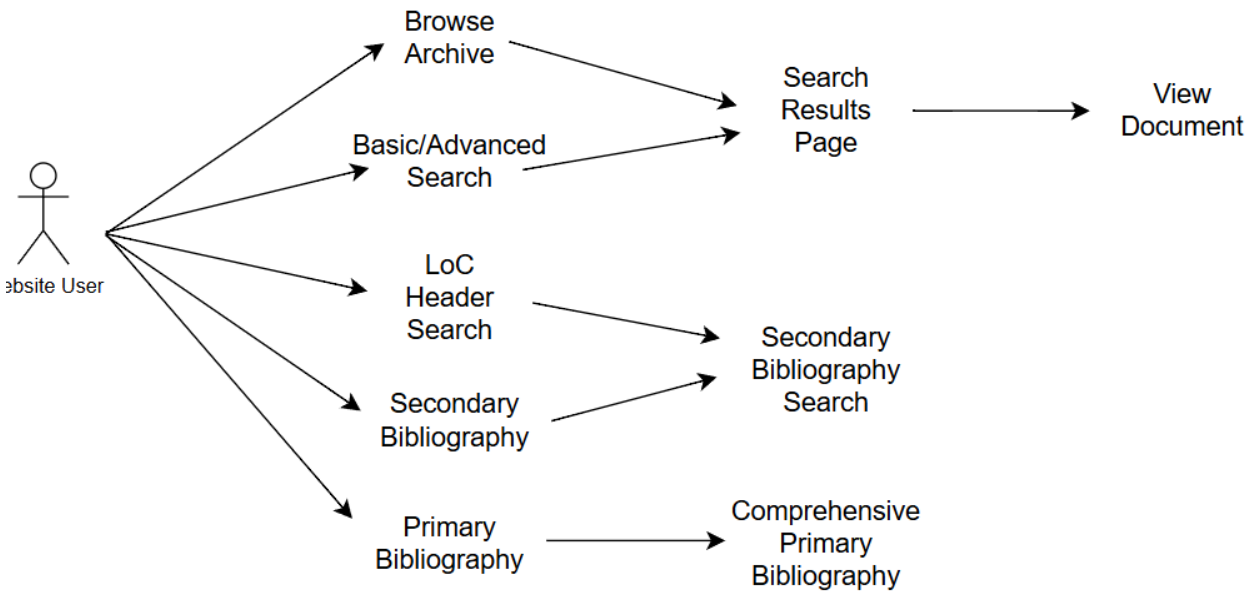


Figure ##. Shows the current use case diagram for a website user.

Initial Website Developer Use-Case Diagram

Seen in the figure below is the use-case diagram of what the website developer looks like on the live version of the website. It is currently a long process in order to be able to edit any page, having to copy any file over to the current database, and then even with that, they still have to manually edit the page.

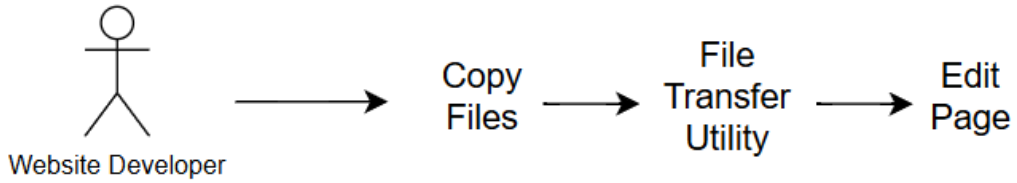


Figure ##. Shows the initial use case diagram for a website developer.

Current Website User Use-Case Diagram

Seen in the figure below will be the use-case diagram we are currently working on for a website developer. Instead of the manual uploads that currently happen, we are hoping to work on a system where the developer would upload the files to the database, which would then reflect on the pages in the archive.

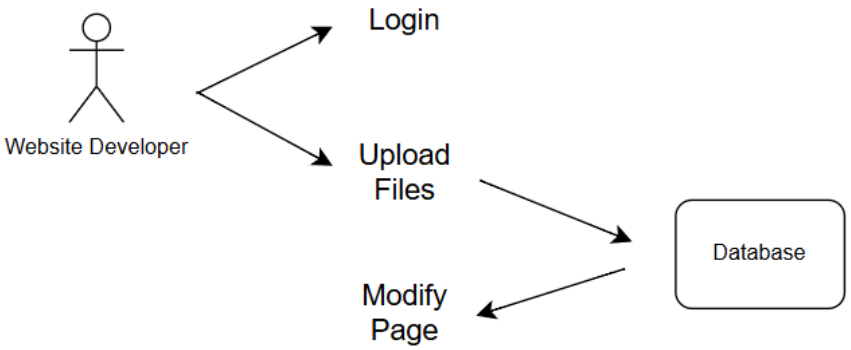


Figure ##. Shows the current use case diagram for a website developer.

State Diagram

Seen in the figure below is a state diagram depicting the assorted states of the website a user can be in. It shows how the user gets through the website into its different states. The user will always start in the archive, and can choose between either of the two bibliographies, or to the meat of the website - searching the archive. From here, the user will bounce between the search page, the search results, and the document's page.

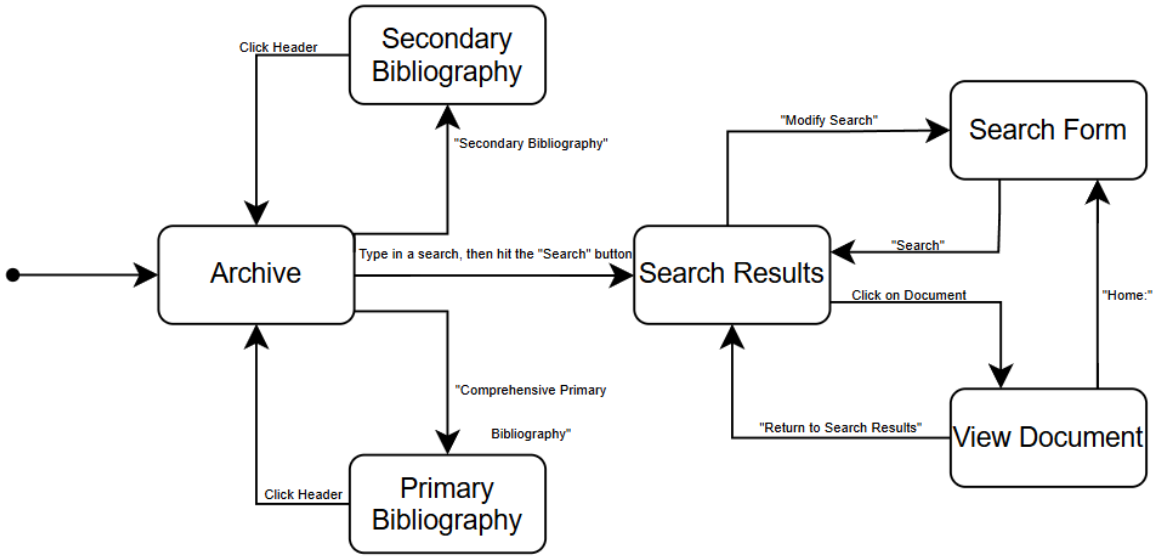


Figure ##. Shows the state diagram for the website.

Activity Diagram

Seen in the figure below is an activity diagram which depicts the choices a user will make on the website. It starts with the user going to the Archive page, then the trickling down choices they get. They can start with the bibliographies, and then either go to the search page or end their visit there. Then, the user will make their search, and look for documents matching it. They will either find a document based on this search, or revise said search and try again.

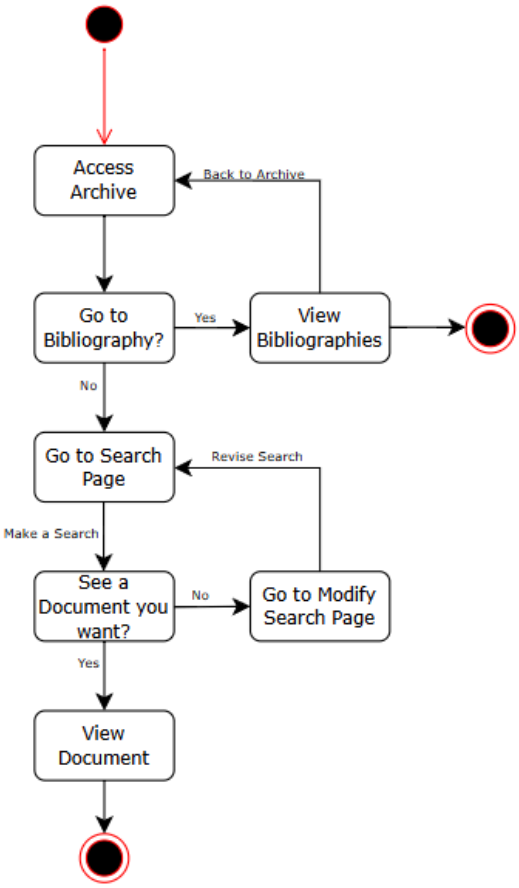


Figure ##. Shows an activity diagram for a website user.

Sequence Diagram

Seen in the figure below is a sequence diagram for the website. This shows a brief timeline of what a website user should expect while visiting the website. It shows the website user interacting with the frontend of the website, and the frontend interacts with the backend. It also shows the approximate timeline of what the user should expect.

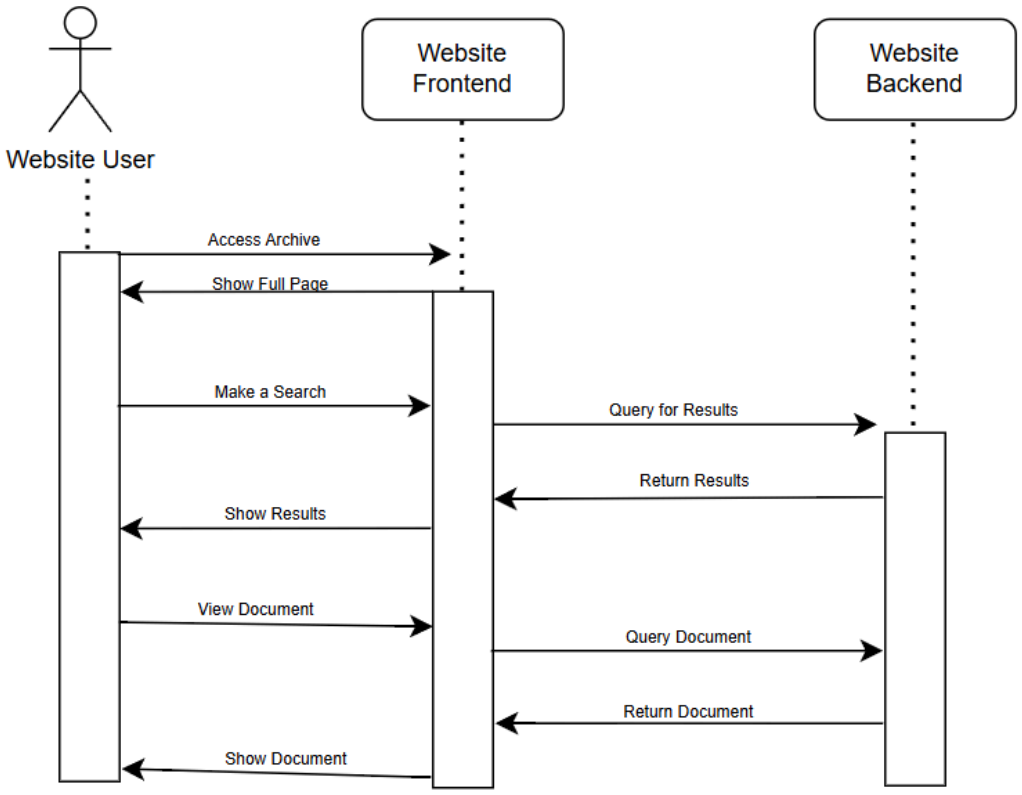


Figure ##. Shows the sequence diagram for the website.

Conclusions

At the beginning of this project, there were numerous things we were set out to fix. These included making the front end more modern, updating the mobile version of the website, fixing key bugs such as the RSS feeds, bookbag feature and Library of Congress, moving over to Red Hat 9, and transferring the files over to the UCF servers via STARS. While having various meetings with the sponsors, these requirements shifted. To start with, all of the aforementioned bug fixes were deemed not important for the future of the website, and were removed from our work load. We also had a

new perspective for changing the front end of the website. As we started, we were giving ideas on a new layout, but none were working as they ventured too far off of the original website. So, the designs that led us to the final one were focused solely on making sure that the website keeps up to code with accessibility laws.

Administration

Budget and financing

Instead of providing us with financial resources and a budget to complete our project, we were instead provided with resources and guidance. Although we do not have a budget or financial help for developing purposes, we were provided with a development environment (the CHDR server) as well as a hint on how to acquire a free copy of the licensing rights to the Red Hat operating system through a development account. We believe we do not need additional financial assistance and will be able to complete the project with the resources provided. The hosting of the project once complete will be arranged and managed by UCF in a cloud provider which will likely be Microsoft Azure. UCF will incur costs in association with the resources utilized in the cloud, as well as licensing costs (for the Red Hat Operating System license for instance) and maintenance costs.

However, towards the end of the project, we ended up encountering a scenario where we needed Oxygen licenses to better work with the XSLT transformation used in the document viewer. At that point, our sponsors were able to speak with their financial department to approve two 1-year

licenses for us. This was the only unexpected cost that came up during the project duration.

Project Administration - Bernardo

Initial Jira Learning

Initially, most of us were not familiar with Jira at all, so it took us some time to get used to checking on it frequently, marking our tasks as completed, and start thinking about the next sprints. Throughout most of Senior Design I we struggled to keep track of our progress through Jira, as we would often forget to mark our tasks as completed there after talking to each other on Discord regarding the task being completed.

Something else that also played a huge factor in our sprint planning, aside from the lack of documentation on our work from the previous sprints, we the fact that our sponsor changed some of the requirements for our project along the way, so a lot of the tasks we had scheduled for a specific sprint, were no longer needed or required way less time than we anticipated it would. The good side of that is the fact that we ended up having more time to learn things than we expected, as a considerable amount of our initial requirements were dropped, but this also caused us to need to re-arrange tasks after the learning portion of it had already been completed. A good example of this is our book bag feature, as one of our team members studied the feature and the current bugs and we ended up deciding along with the sponsor to not proceed with fixing it, as it was not something utilized by the users at all.

On top of that, we also tried to separate some time to understand and create the user stories properly, as that was one of the most important parts during the Jira setup.

Sprints

In the beginning of the project we decided that at least during Senior Design I, our sprints would be two weeks long each, mostly because a lot of our tasks consisted in learning the technologies that we would be using in the projects, which tends to take more than a week. So to avoid us having a lot of sprints with the same information, for example learning PHP, we decided to have them bi-weekly. Our first two sprints, however, were one week long, as at that point, we were still deciding how we wanted them to be set up.

During Sprint One, which lasted from September 17th until September 24th, we focused mainly on getting familiar with the current codebase and website so we knew exactly what we needed to focus on during our learning. We also wanted to ensure that we had access to the CHDR server, which is our current test environment where we have a copy of all the files running in the current production server.

On top of that, we used that week to get some of our team-related things set up such as our DailyBot where we write about our daily progress through a Discord bot, making sure all the members had access to Jira and to our Discord server with the sponsors, setting up our GitHub repository for the project’s version control, and create the Roadmap that was used during a previous assignment.

Mon, Sep 30 2024, 1:36pm	Sprint completed	SCRUM-1 Get familiar with the website SCRUM-3 Get familiar with Jira and DailyBot SCRUM-4 Set up DailyBot for Team StandUp check-ins SCRUM-5 Invite All Team members to Jira member SCRUM-6 Set up GitHub SCRUM-8 Understand project requirements SCRUM-9 Start creating roadmap SCRUM-38 As a developer, I would like to get familiar with the current version of the website SCRUM-39 As a developer, I would like to be familiar with Jira and the DailyBot to improve my workflow SCRUM-40 As a teammate, I would like to have the DailyBot set up for the standup check-ins with my team SCRUM-41 As a developer, I would like to have a GitHub repository for project collaboration SCRUM-42 As a teammate, I would like to ensure all team members have access to Jira SCRUM-43 As a developer, I would like to ensure that I understand the project requirements SCRUM-44 As a developer, I would like to have a base concept of the project's roadmap

Sprint 1 Issues and User Stories

Source: Our Team's Jira Board

Our second Sprint happened between September 24th and September 30th and it was our last one week long sprint. During this Sprint we focused on following up with the points from Sprint One, more specifically in ensuring all the files we needed to start the project were accessible in the CHDR server and that our access were all fixed, as during that week, some of our team members lost their access due to the passwords being reset.

We also worked on ensuring each member had a Red Hat 9 virtual machine set up either via Virtual Box, Proxmox, or Hyper-V. In order to get the Red Hat ISO, we had to create an account within Red Hat's official website as a student or employee of a company, which then allowed us to not only download an ISO for Red Hat, but also activate our operating system during the installation process. Along with that, we also continued working on ensuring the user stories made sense and that each task had a user stories that matched its description.

Mon, Sep 30 2024, 1:40pm	Sprint completed	SCRUM-12 Get access to CHDR SCRUM-14 Get access to XML files that need to be migrated SCRUM-18 Get RHEL VMs set up (requires lots of RAM) SCRUM-28 Write user stories for each SCRUM item SCRUM-30 As a developer, I would like to have access to CHDR for testing purposes SCRUM-32 As a developer, I would like to have access to a Virtual Machine that has RHEL 9 that will allow me to test on an environment close to production SCRUM-33 As a developer, I would like to have access to the web server files on the CHDR system SCRUM-36 As a developer, I would like to have a virtual machine setup with RHEL 9.4 for testing purposes SCRUM-37 As a developer, I would like to have access to the XML files that need to be migrated to exist-db

Sprint 2 Issues and User Stories

Source: Our Team's Jira Board

Our third Sprint, which was from September 30th until October 14th, was a bit messier than the others as we were on a mid-point for the beginning of our project. We were transitioning between setting up everything and starting to learn the content, so that caused the sprint to be all over the place when it comes to the tasks assigned. At the same time that some of our group members were already learning PHP to be able to interact with the database, we were also working on adding all the XML files that were going to be needed to test the PHP's code ability to pull information to the database to our GitHub and to exist-db. This caused the early stages of the learning to be a bit limited compared to what we would like, as we weren't able to actually test some of the queries we were creating initially.

We were also under the impression that we would be able to get access to the current production server, which will not be the case. We will pretty much use the CHDR server as like it was the current production environment, and because of that, we also needed to ensure during those two weeks that everybody was good using it. Another important section for that week that is not directly mentioned in the Sprint was the fact that some of our members had to learn some basics Linux, as the CHDR server is

Ubuntu-based. Some of our group members have never had experience with Linux aside from the very basics, such as ls and cd, so we wanted to make everybody was aware of what they were touching and doing on the CHDR server before we actually started using it.

Mon, Oct 14 2024, 3:24pm	Sprint completed	SCRUM-15 Start learning PHP SCRUM-31 As a developer, I want to be able to code and interact with the back-end that is written in PHP SCRUM-45 As a developer, I would like to get access to the current production environment running in RHEL 7 SCRUM-13 Learn how to use CHDR SCRUM-35 As a developer, I would like to ensure that everybody on my team knows how to use the CHDR system SCRUM-60 Get familiar with the project's code (there's no documentation) SCRUM-61 Add code from CHDR to GitHub SCRUM-62 As a developer, I would like to be able to collaborate with my team through the use of source control on GitHub

Sprint 3 Issues and User Stories

Source: Our Team's Jira Board

During our fourth Sprint, which lasted from October 14th until October 28th, we were still in the process of learning a lot of the technologies involved in the project, such as PHP, exist-db, JavaScript, React. We also had a very specific topic that the sponsors wanted us to work on which were the LOC Headings, so one of our team members took some time to review what those are and how to use them, as they were going to be a significant part of the project. Most of our PHP learning was done through online courses available on Youtube, and the same applies for JavaScript and React, however, for exist-db, we mostly relied on its documentation, since there isn't as much resources for it as for other databases such as MongoDB and MySQL.

Throughout that Sprint we also started creating our Figma prototype for the new website, as we needed to have that approved at least a week or two after Winter break considering our plan was to have a skeleton of the

website before we went out for the break. Although our Figma prototype wasn't complete within that Sprint, we had enough progress to discuss several points among us and ask the sponsor some clarifying questions on their ideas for the new layout.

On top of that, we also created a structure outline for our Design Document during that Sprint, as we had the first part of it due around that time. This really helped us organize our ideas and make the writing more efficient, since everybody had an idea on what they were going to write about and how the document was going to be structured. It also made it a lot easier for us to see how we were going to split the text sections for the second part of the Design Document ahead of time, which saved us a lot of time throughout the last few weeks of November.

Tue, Oct 29 2024, 12:16pm	Sprint completed	SCRUM-16 Refine JS and React skills SCRUM-24 Refine JS and React skills SCRUM-20 Start learning exist-db SCRUM-19 Learn how to use LOC Headings SCRUM-46 As a developer, I would like to be able to understand and write React JS code SCRUM-48 As a developer, I would like to be able to interact and work with exist-db SCRUM-47 As a developer, I would like to know and how to work with LOC Headings SCRUM-66 Continue learning PHP SCRUM-67 Structure Preliminary Design Document to prevent redundant writing SCRUM-68 Start creating Figma prototype SCRUM-75 Start learning exist-db SCRUM-76 As a teammate, I would like to have a sketch of the Design Document to avoid redundant writing SCRUM-77 As a developer, I would like to know enough PHP to call APIs and manage their responses SCRUM-78 As a developer, I would like to have a visual Figma prototype of what the outcome of the project should look like

Sprint 4 Issues and User Stories

Source: Our Team's Jira Board

For our fifth Sprint, which was from October 29th until November 10th, we had a lot of different things that we wanted to focus on. The first thing we wanted to work on was understanding some of the compliances that we needed to ensure we were in accordance when building the new backend

and frontend of the website. We not only had to review security compliances, but also accessibility compliances, more specifically the items listed under the Section 508 of the Rehabilitation Act of 1973.

Aside from reviewing and planning how we were going to adhere to the required compliances, we also worked on finalizing our learning of exist-db and understanding how TEI files work. Learning what TEI files are and how they work was an essential part of our project, since one of the main points of our work will be interacting and handling XML files.

During this Sprint, we also wanted to focus on having our main pages of the prototype completed and reviewed by the sponsor, as they were going to be the base needed for us to complete the rest of the website's design within the Figma prototype. Unfortunately, we weren't able to get the design completely reviewed and approved within that Sprint, since they mentioned some changes they would like to have on the website during one of the meetings that we did not include in our initial prototype for the main pages, so we ended up addressing those changes before showing the design to them. However, we still marked this task as completed since we did have the main pages fully designed, and the changes we had to do were not substantial.

On top of that, we wanted to be sure we were working on the Design Document ahead of time, as we did not want to leave anything to the last minute, so we had as one of the tasks a weekly contribution we would like all team members to do towards the Design Document so we didn't have to rush everything on the week it was due. The downside of doing that was the fact that a lot of things changed within that time frame, and we ended up

writing about certain topics that were no longer relevant or were related to features we decided not to implement.

Tue, Nov 12 2024, 11:42am	Sprint completed	SCRUM-21 Learn how the TEI (transform PDF into text) files work SCRUM-49 As a developer, I would like to understand how TEI works SCRUM-22 Write 3 pages of the Design Document every week (6 total for the Sprint) SCRUM-50 As a teammate, I would like to be able to contribute at least 3 pages a week to the project's Design Document SCRUM-71 Understanding GDPR compliances SCRUM-72 Understanding 508 compliance SCRUM-73 Having the main pages prototype completed SCRUM-74 Finish learning exist-db SCRUM-79 As a developer, I would like to have a visual Figma prototype of what the outcome of the project should look like SCRUM-80 As a developer, I would like to be able to interact and work with exist-db SCRUM-81 As a developer, I would like to understand what GDPR compliances are required for the project SCRUM-82 As a developer, I would like to understand what 508 compliances are required for the project

Sprint 5 Issues and User Stories

Source: Our Team's Jira Board

When it comes to our most recently completed Sprint, which was our sixth one and that lasted from November 10th until November 24th, we continued the previously mentioned task about contributing with a certain amount of pages to the Design Document weekly, and along with that, we wanted our documento to be completed on the day the sprint was completed, which was a day before its due date, so on the 25th we had enough time to adjust the layout of the document as well as citations. Due to that, one of the tasks included in that sprint was related to completing the Design Document within its due date.

Along with that, we also had, during this sprint, the goal of having the website's design complete on Figma so we could get the sponsor's approval to start the frontend's coding portion of the project, as well as, completing our studies on exist-db.

Our main idea was that after this sprint we would have a solid understanding of the design we need to reproduce and how we were going to be able to reproduce it, and be able to perform queries to exist-db that would return the desired XML documents, which we have been able to achieve successfully.

Mon, Nov 25 2024, 3:20pm	Sprint completed	SCRUM-69 Complete the Design Document SCRUM-70 Finalize front end's design SCRUM-84 As a developer, I would like to have a visual Figma prototype of what the outcome of the project should look like SCRUM-85 As a teammate, I would like to have the Design Document completed within its due date SCRUM-74 Finish learning exist-db SCRUM-22 Write 3 pages of the Design Document every week (6 total for the Sprint) SCRUM-79 As a developer, I would like to have a visual Figma prototype of what the outcome of the project should look like SCRUM-80 As a developer, I would like to be able to interact and work with exist-db SCRUM-73 Having the main pages prototype completed SCRUM-50 As a teammate, I would like to be able to contribute at least 3 pages a week to the project's Design Document

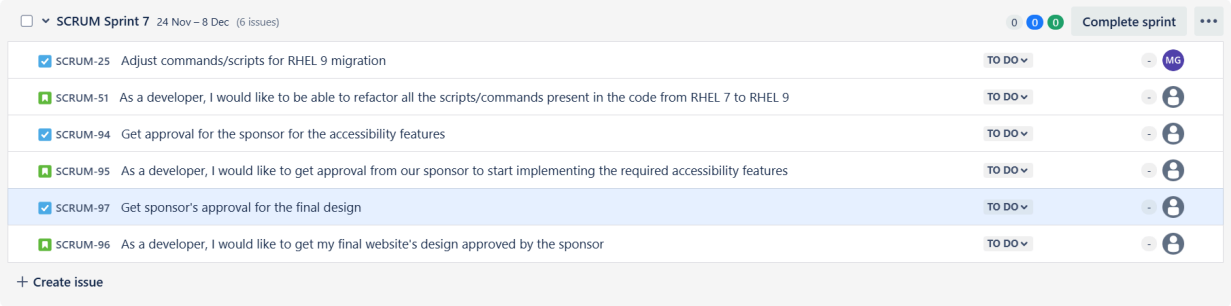
Sprint 6 Issues and User Stories

Source: Our Team's Jira Board

For our current Sprint, which is our seventh one and will last from November 24th until December 8th, we will have a less intense sprint due to the Thanksgiving break, as well as finals week for those team members that are taking other classes along with Senior Design I.

We will focus on completing the refactoring of all the scripts currently present in the code to match the changes in syntax between Red Hat 7 and Red Hat 9, as well as getting the sponsor's approval for both the latest version of the website's prototype in Figma and the accessibility features. Ideally, we would have those approved within the first week of the Sprint, so we could use its second week to code a skeleton of the website prior to the break. Since we already have a considerable amount of progress in the

backend, we would also like to leave for the break with some progress in the frontend aspect of the project.



Sprint 7 Issues and User Stories

Source: [Our Team's Jira Board]

Overall our experience with JIRA was very positive and despite the adaptation period in the beginning of the project, all the team members have been keeping up with their tasks and marking them as completed on the board. This tool has allowed us to organize the project and our contributions easily, as well as helps us maintain information regarding what still needs to be completed, which helps us plan ahead more efficiently.

Acknowledgements

Acknowledgements

- Dr. Amy Giroux (Sponsor) - we thank Dr. Giroux for her thorough explanations on the history of this website that she maintains and for her guidance on how we can improve upon the current frontend and what is expected from us as a team. Dr. Giroux also showed us how the current STARS database is set up and talked to us about the

Archive’s current architecture.

- Dr. Brook Miller (Sponsor) - we thank Dr. Miller for his explanations on how we can work with exist-db and for helping us get access to the files for the project. Dr. Miller allowed us access to the CHDR server and gave us advice on how to improve our approach to the frontend. He also communicated directly with the director of his department to ensure our project is up to the department’s standards. Additionally, we encountered an obstacle in our practice and research into Exist-DB. All of our queries were failing to produce the expected response. Dr. Miller took the time to investigate this behavior where he found that the issue was because of a zipped file that was being referenced by the xml files in the database. We appreciate and thank Dr. Miller for taking the time and for helping us with that situation.

- Mike L. (Assistant to Sponsors) - Mike helped us during the first few weeks of the project to figure out what the system architecture of our frontend would look like. He gave us some tips on how we could work out the stretch goal related to Podman containerization.

- Center for Humanities and Digital Research - we thank CHDR for allowing us to access the files they have for the Archive’s website and to create our own folders on their server to experiment with our own PHP files as well.

- Julia D. (TA) - we thank Julia for meeting with us throughout the semester and checking on our project’s progress as well as for answering our questions.

- Dr. Matthew Gerber (Coordinator) - we thank Dr. Gerber for keeping us on track in this project and for ensuring we have all the necessary knowledge we need to successfully run this project as a group.

Bibliography

TEI - Rodrigo

The TEI infrastructure - the TEI guidelines. (n.d.). https://www.tei-c.org/release/doc/tei-p5-doc/en/html/ST.html

XSLT - Rodrigo

W3Schools.com. (n.d.). https://www.w3schools.com/xml/xml_xslt.asp#:~:text=XSLT%20

XTF and Exist-DB - Rodrigo

XTF » Fundamental Concepts. (n.d.). https://xtf.cdlib.org/getting-started-tutorials/fundamental-concepts/index.html

Meier, W., Turner, J., Krebs, T., Windauer, L., & Retter, A. (n.d.). *Getting started*. https://exist-db.org/exist/apps/homepage/index.html

Indexes on Exist-DB - Rodrigo

EXIST-DB Documentation. (n.d.). https://exist-db.org/exist/apps/doc/indexing.xml#:~:text=eXist-db%2

0has%20no%20%22create,Admin%20interface%20or%20Java%20Client

EXIST-DB Documentation. (n.d.-b). https://exist-db.org/exist/apps/doc/lucene

EXIST-DB Documentation. (n.d.-c). https://exist-db.org/exist/apps/doc/oldrangeindex

EXIST-DB Documentation. (n.d.-d). https://exist-db.org/exist/apps/doc/newrangeindex

Exist-DB Prototype - Rodrigo

Contributors to Wikimedia projects. (2007, August 10). *XQuery*. Wikibooks, Open Books for an Open World. https://en.wikibooks.org/wiki/XQuery

Data Migration - Rodrigo

Sheldon, R., & Scarpati, J. (2021, August 27). *Server Message Block protocol (SMB protocol)*. Search Networking. https://www.techtarget.com/searchnetworking/definition/Server-Message-Block-Protocol

What is Samba? (n.d.). https://www.samba.org/samba/what_is_samba.html

GeeksforGeeks. (2024a, April 28). *File Transfer Protocol (FTP)*. GeeksforGeeks. https://www.geeksforgeeks.org/file-transfer-protocol-ftp/

Spencer, P., & Spencer, P. (2024, November 13). *SFTP encryption Essentials for Secure data Transfers*. Kiteworks | Your Private Content Network. https://www.kiteworks.com/secure-file-transfer/sftp-encryption/

McKay, D. (2023, October 4). *How to Use the scp Command on Linux*. How-To Geek. https://www.howtogeek.com/804179/scp-command-linux/

Cooper, S., & Cooper, S. (2023, July 12). *SFTP vs SCP Guide*. Comparitech. https://www.comparitech.com/net-admin/sftp-vs-scp/

How To Use SFTP to Securely Transfer Files with a Remote Server | DigitalOcean. (n.d.). https://www.digitalocean.com/community/tutorials/how-to-use-sftp-to-securely-transfer-files-with-a-remote-server

How to use RSYNC to sync local and remote directories | DigitalOcean. (n.d.). https://www.digitalocean.com/community/tutorials/how-to-use-rsync-to-sync-local-and-remote-directories

STARS - Rodrigo

STARS - showcase of text, archives, research & scholarship at UCF. (n.d.). https://stars.library.ucf.edu/

Version Control and Collaboration - Rodrigo

Repository size limits for GitHub.com. (n.d.). Stack Overflow. https://stackoverflow.com/questions/38768454/repository-size-limits-for-github-com

Storage | GitLab. (n.d.). https://docs.gitlab.com/ee/user/storage_usage_quotas.html

Heddings, A. (2020, March 27). How to set up a personal GitLab Server. *How-To Geek*. https://www.howtogeek.com/devops/how-to-set-up-a-personal-gitlab-server/

GeeksforGeeks. (2024, July 23). *Difference between GitLab and GitHub*. GeeksforGeeks. https://www.geeksforgeeks.org/difference-between-gitlab-and-github/

Original Plan for Desktop Frontend - Dawn

Figma. “Figma: The Collaborative Interface Design Tool.” *Figma*, 2024, www.figma.com/.

Accessibility Guidelines on the Web - Dawn

W3C. “Web Content Accessibility Guidelines (WCAG) 2.1.” *W3.org*, 2023, www.w3.org/TR/WCAG21/.

UCF. *Digital Accessibility*. 20 June 2022. https://policies.ucf.edu/documents/2-006.pdf.

Meta Platforms. “Accessibility – React.” *Legacy.reactjs.org*, legacy.reactjs.org/docs/accessibility.html.

Luke J McGrath. “WCAG 2.0 Checklist - a Free and Simple Guide to WCAG 2.0.” *Wuhcag*, 2013, www.wuhcag.com/wcag-checklist/.

Gael (intro) - Britannica, T. Editors of Encyclopaedia (2024, February 19). Charles Brockden Brown. Encyclopedia Britannica. https://www.britannica.com/biography/Charles-Brockden-Brown

Gael (GDPR) - Wolford, B. (2024, August 29). What is GDPR, the EU’s new data protection law? GDPR.eu. https://gdpr.eu/what-is-gdpr/

Kiara (Executive Summary) - The Charles Brockden Brown Electronic Archive and Scholarly Archive. (2024). Ucf.edu. https://brockdenbrown.cah.ucf.edu/index.php