A4. Info Design and Portfolio Project

Project Description and Goals:

I chose option B for my final project because I wanted to continue to learn more about accessibility and information design but in a different context. I chose to investigate the usability of nine different University of Wisconsin school disability and accessibility service websites. I was inspired from the research article I used in the A3 project to do this research.

One of my goals for this project was to conduct my own web accessibility research. I have worked with college students with disabilities and have experience seeing their struggles with web usability. When we started to learn about accessibility in class, I was motivated to further my knowledge in it because it is something I am interested in.

My second goal was to use different tools than I have previously used in other projects. In the beginning of the course, I noted in one of the first discussion posts that I wanted to learn how to use different technology to further my information design skills. In projects A1 and A2, I used Canva and Excel, but I was already familiar with them. In this project I decided to use WAVE (Web Accessibility Evaluation Tool), Excel, and RStudio.

What I investigated:

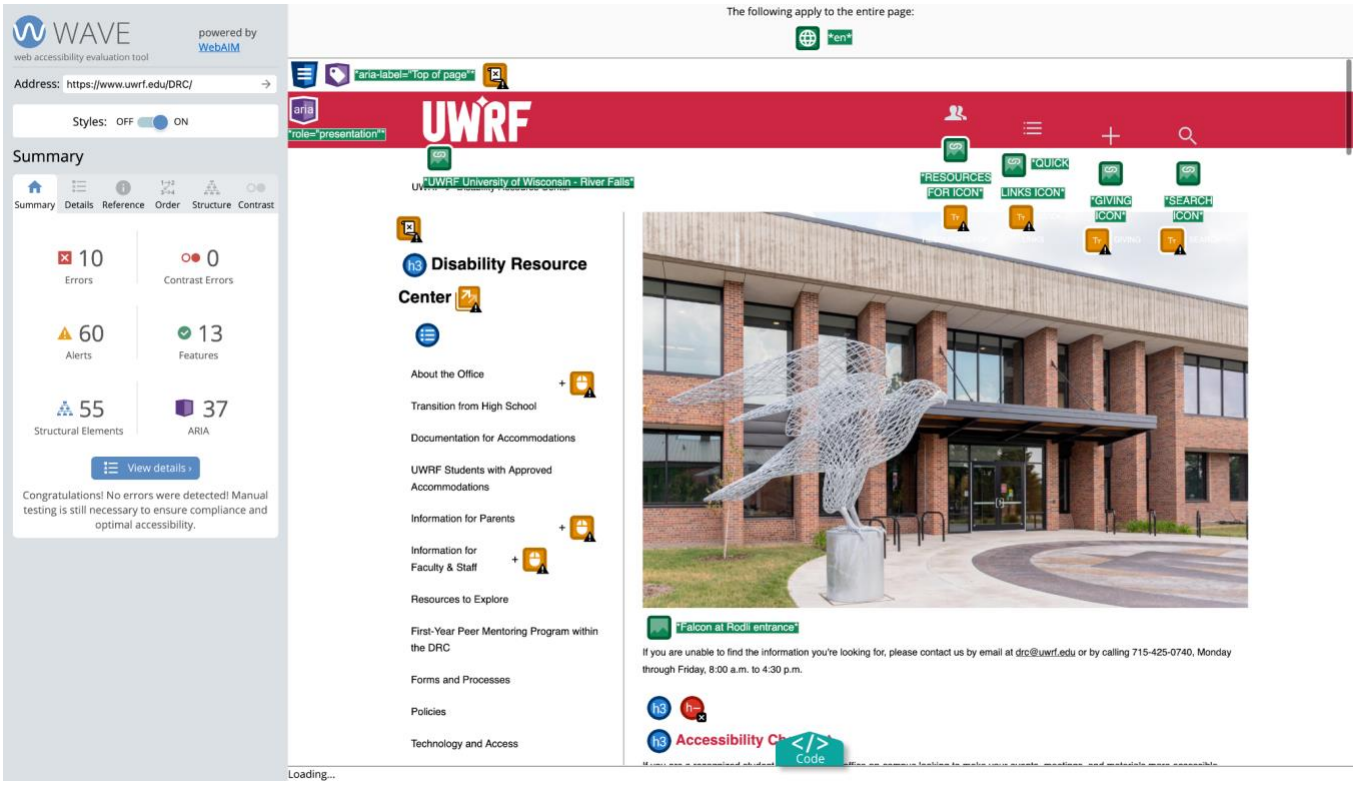
I investigated nine different University of Wisconsin system disability and accessibility service websites. The list of websites and their links are below:

- UW-La Crosse Disability Resource Center
- UW-Milwaukee Accessibility Resource Center
- UW-Eau Claire Services for Students with Disabilities
- UW-Parkside Student Accessibility Services
- UW-Oshkosh Center for Accessibility and Disability Resources
- UW-Platteville Disability Access Center
- UW-River Falls Disability Resource Center
- UW-Green Bay Student Accessibility Services
- UW-Whitewater Center for Students with Disabilities

The University of Wisconsin System states that “Our commitment to accessibility reflects our belief that knowledge should never be limited by how someone navigates the digital world”. I wanted to research how well the UW System websites follow through with this commitment. I found that three out of nine have no errors. All websites have many alerts that raise concern. To find my data, I used a tool called WAVE (Web Accessibility Evaluation Tool).

WAVE Tool:

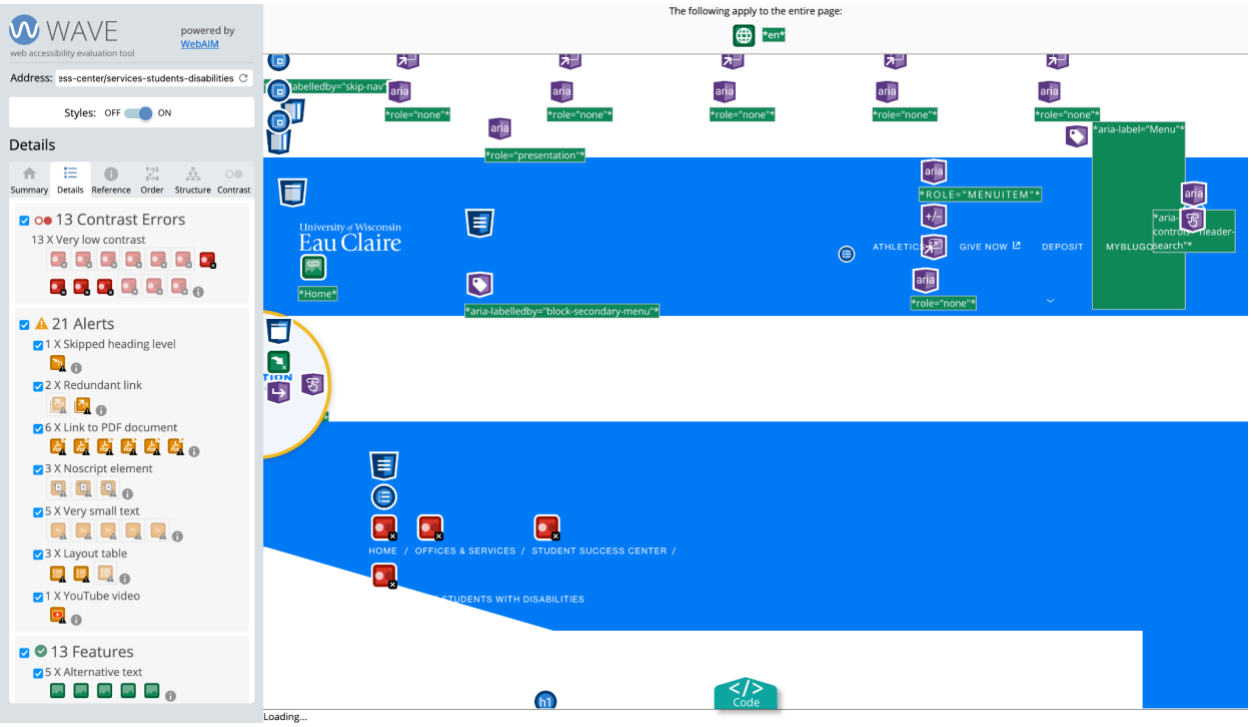
WAVE was created by the group Web Accessibility in Mind (WebAIM) at Utah State University. To use this tool, a user can simply paste a website URL into the search box.



From there, the tool will display a summary, breakdown of errors, alerts, and features, and the order and structure of content. I started on the home page of each website's disability or accessibility service program and then to each subpage usually through a navigation bar. In the example screenshot above from the UW- River Falls site, I started on their Disability Resource Center home page, and I went to each link on the left side and recorded data from each page.

How Data was Recorded:

To record the data from WAVE, I used Excel. I created a sheet for each website and then created a table that kept track of the type of error, contrast error, or alert and the number of times they appeared across each page. The Excel sheet is in the Google Drive folder linked on the first page of this document. Some alerts or errors were the same across multiple pages. One example is in the contrast errors on the UW-Eau Claire website.



This website contained 169 very low contrast errors. The white text displaying the webpage path was caught on every single page. Other websites had high noscript elements, redundant links, redundant title texts, and PDF document alerts. After I entered all of the data in, I used RStudio to create my graphs. Below are some definitions of alerts and errors that need more context before discussing the graphs. All definitions are from the WAVE website.

> **Noscript Element**: Content within <noscript> is presented if JavaScript is disabled. Because nearly all users (including users of screen readers and other assistive technologies) have JavaScript enabled, <noscript> cannot be used to provide an accessible version of inaccessible scripted content.
>
> **Redundant Title Text**: The title attribute value is used to provide *advisory* information. It typically appears when the users hover the mouse over an element. The advisory information presented should not be identical to or very similar to the element text or alternative text.
>
> **Layout Table**: Layout tables exist merely to position content visually - to create columns, insert spacing, or align content neatly for sighted users. Their content is not at all tabular in nature. Layout tables should not be used in HTML5. They can introduce reading and navigation order issues. Screen readers may interpret them as data tables (i.e., announcing column and row numbers), especially if they contain table header (<th>) cells. This introduces significant overhead on screen reader users.

Skipped Heading Level: Headings provide document structure and facilitate keyboard navigation by users of assistive technology. These users may be confused or experience difficulty navigating when heading levels are skipped.

Orphaned Form Label: An incorrectly associated label does not provide functionality or information about the form control to the user. It usually indicates a coding or other form labeling issues.

Possible Heading: Heading elements (<h1>-<h6>) provide important document structure, outlines, and navigation functionality to assistive technology users. If heading text is not a true heading, this information and functionality will not be available for that text.

Empty Heading: Some users, especially keyboard and screen reader users, often navigate by heading elements. An empty heading will present no information and may introduce confusion.

Missing Form Label: If a form control does not have a properly associated text label, the function or purpose of that form control may not be presented to screen reader users. Form labels also provide visible descriptions and larger clickable targets for form controls.

Results:

Accessibility Alerts

Four graphs were created to visualize the data recorded from using the WAVE tool. First, the total number of alerts were compared. In total, there were 1,703 alerts recorded across all websites. Each website had a different amount of pages which affected the total number of alerts and errors. For the graph, I created a horizontal bar chart. I did this because of the close data points that some universities share with each other. Also, the university names are long and will take up a lot of space on the x-axis, so I moved them to the y-axis. Data labels are present so the audience can distinguish each data point. This design is consistent across all the graphs, but the bar color is changed in each one.

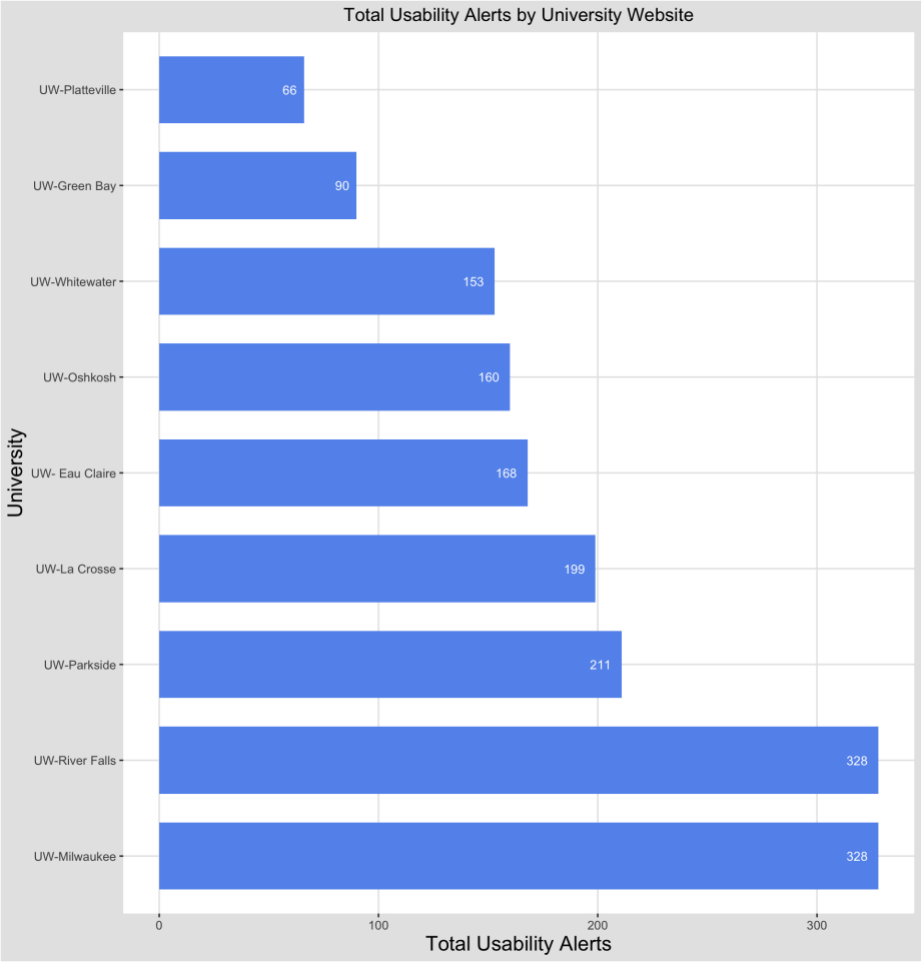


Image 1: Total Usability Alerts by University Website

UW-River Falls and UW-Milwaukee had 328 alerts. UW-River Falls had 13 unique alert types while UW-Milwaukee had 11. Those two websites were also the largest out of the nine. UW-Whitewater, UW-Green Bay, and UW-Platteville had smaller websites and had less alerts. Many of the alerts were the same across multiple pages which resulted in high totals. In Image

2 below, the top ten types of alerts were recorded.

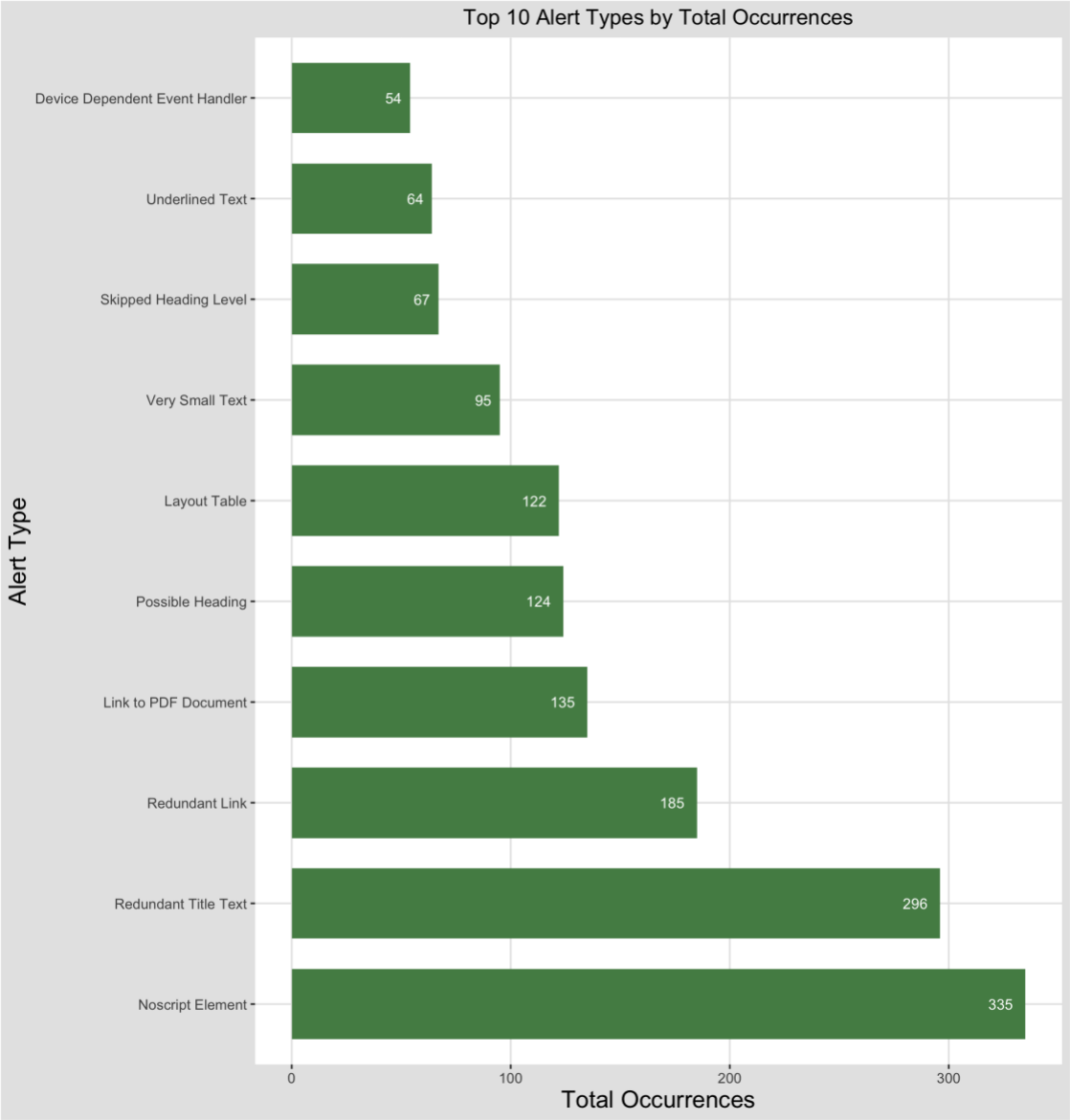
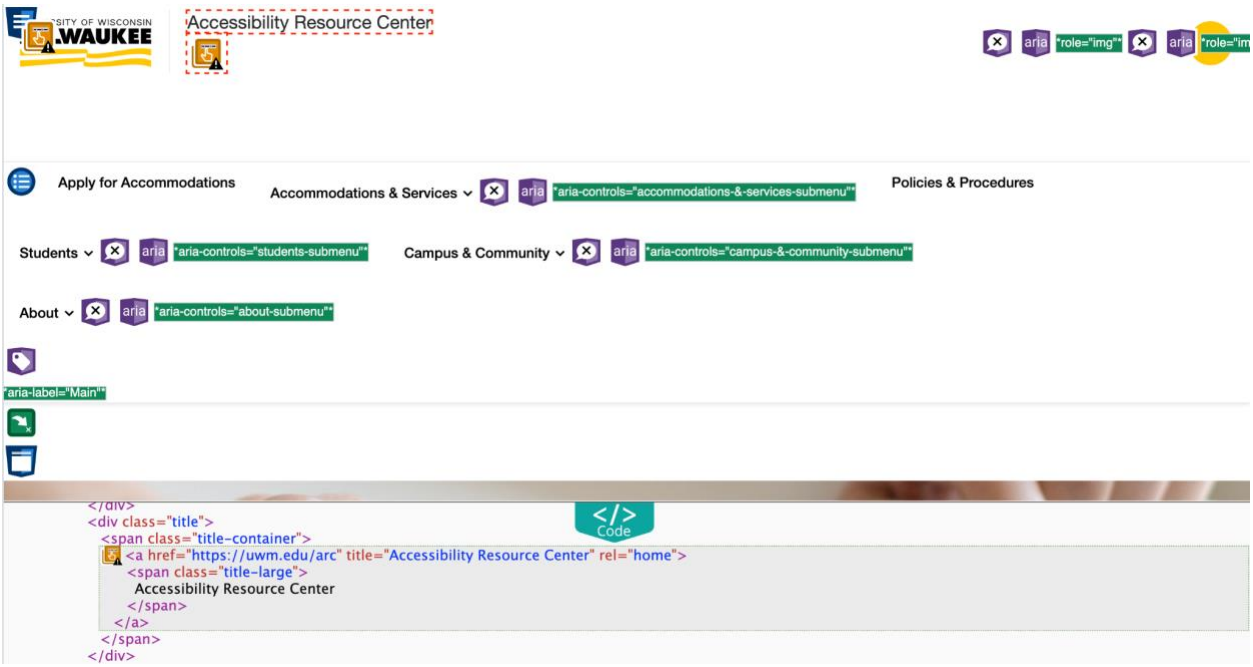


Image 2: Top 10 Alert Types by Total Occurrences

Noscript elements were the top alert. Noscript elements help users with JavaScript disabled to potentially view inaccessible content. Most assistive technology has JavaScript enabled, which means users might not view the noscript element content. All content must be accessible even if it falls under the noscript element. Next was the redundant title text. This appeared when the title of a webpage was the same as the text in the HTML code. An example of this from the UW-Milwaukee website is below.



The title “Accessibility Resource Center” is the same on the page and in the code. These alert matters because if a user is using a screen reader, they may get the title text twice. Fixing this includes deleting the title attribute in the code or modifying the text so it is not redundant. The third most common alert was redundant links. These alerts came from the link on top of the page that would return to the university’s home page or a link in the footer that appeared on most pages. These do not create any concern for normal users, but those using a screen reader will have additional navigation and repetition.

Accessibility Errors

Three websites contained zero errors. Those websites are UW-Whitewater, UW-La Crosse, and UW-Eau Claire. UW-Parkside contained 100 errors which was the most out of the nine sites. 91 of those errors were missing form labels. UW-Parkside has an accessibility feature that changes the website’s content to Spanish with one button click. An error occurs from this in a feedback form that is hidden from the main styles. That form contains empty labels that does not let the user know what the function of the form is. An example of a form label would be when a user clicks on an empty form box, a text description will appear telling them what to put in that box, such as their name, contact information, feedback, etc. Since the feedback form is not visible to users, it is not a huge problem. UW-River Falls and UW-Milwaukee had the second and third most errors. These two websites had 328 alerts which was the most out of the nine websites.

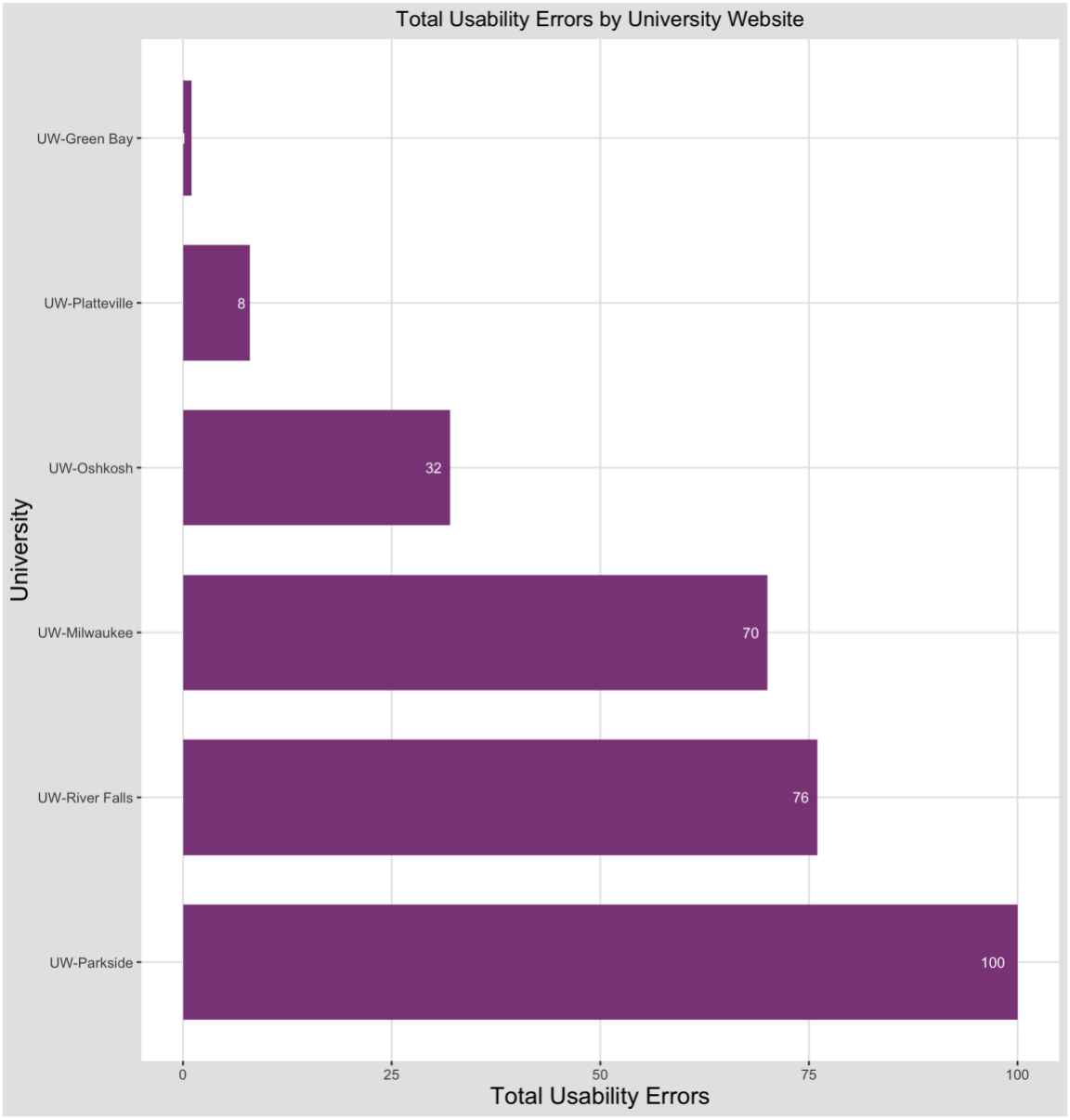


Image 3: Total Usability Errors by University Website

UW-Milwaukee's errors were also a majority of missing form labels at 45 errors. UW-River Falls had many empty heading errors at 49. Empty heading errors are important because users with screen readers navigate by head elements. This can cause confusion if the heading has no text. This is a major accessibility problem that needs to be fixed on the UW-River Falls disability services website.

Next are the top errors that occurred on the six websites. UW-Parkside and UW-Milwaukee combined to have almost all the missing form label errors. The second and third most errors are the most important. Again, empty headings can cause confusion for those with screen readers because of those users navigate using heading elements. Empty links are also crucial because they restrict navigation. Information may not be accessible because of this.

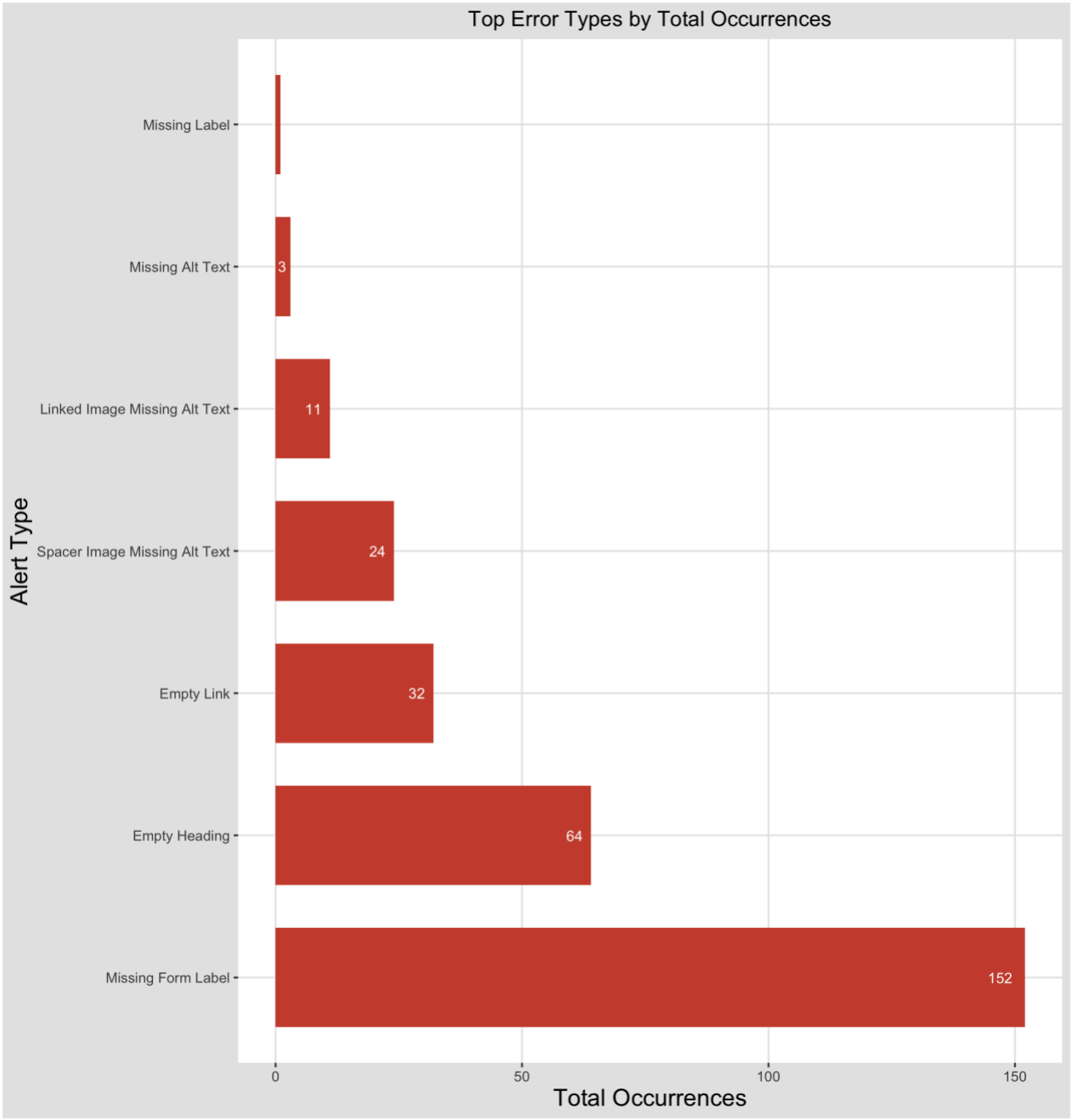


Image 4: Top 10 Error Types by Total Occurrences

Totals

Below are tables created in Excel displaying total alerts and errors and the number of websites the type of alert or error appeared in. They are in order of the type of alert or error appearing the most to least and the number of websites that the specific alert or error appeared on. I used an alternating pattern of blue with black text to make the text readable.

Total Alerts of all Websites

Alert Type	Total	Number of Websites
Noscript Element	335	8
Redundant Title Text	296	6
Redundant Link	185	9
Link to PDF Document	135	7
Possible Heading	124	4
Layout Table	122	6
Very Small Text	95	2
Skipped Heading Level	67	8
Underlined Text	64	4
Device Dependent Event Handler	54	1
Link to Word Document	36	4
Broken Same-Page Link	28	1
Orphaned Form Label	18	3
Suspicious Link Text	17	4
Redundant Alt Text	16	2

Tabindex	14	1
Missing Form Label	6	1
Long Alt Text	6	3
YouTube Video	5	5
Images with Title	4	1
Suspicious Alt Text	3	2
Link to PowerPoint Document	2	2
Missing Fieldset	1	1
Missing First Level Heading	1	1
Possible List	1	1

The most prevalent alerts were redundant links which appeared on all nine websites. This could be because of links in the header and footer of each page. Redundant links can cause confusion with users because of restricted navigation. Next prevalent alerts were noscript elements and skipped heading levels. These alerts can cause content confusion. Users using screen readers might encounter content twice because it is not organized well. The next highest alert was links to PDF documents. Websites need to make sure that the links to PDFs are not broken and will be downloaded correctly. Some computers may not have PDF readers and will keep users from opening PDFs.

Total Errors of all Websites

Error Type	Total	Number of Websites
Missing Form Label	152	3

Empty Heading	64	4
Empty Link	32	2
Spacer Image Missing Alt Text	24	1
Linked Image Missing Alt Text	11	1
Missing Alt Text	3	1
Missing Label	1	1

Seven error types were recorded in six of the websites. The most prevalent was empty headings. This error can restrict users with a screen reader because they might get confused on the organization of the website. The heading will be read as blank or empty and not let the user know where they are on a page. Next, missing form labels were on half of the sites. This again can cause users to not know what to input in a form text box. Empty links were the third prevalent error. This error can cause frustration in a user when trying to access specific information on a page. The link could bring them to a different page or document but the link will do nothing when clicked.

Conclusion of Research:

WAVE was used on nine different University of Wisconsin system disability or accessibility service websites. RStudio was used to create graphs and visualize the data from WAVE. Excel was then used to create data tables to reinforce data from the graphs. The results of the web accessibility evaluation on UW System websites found that a majority of websites have accessibility errors and alerts that can restrict accessibility for those that use assistive technologies. UW-Milwaukee and UW-River Falls had the most total alerts and errors combined. Users with disabilities that try to access services at their university may encounter navigation issues or missing content that will leave them unsatisfied.

Reflection:

Challenges and Triumphs in completing deliverables.

One challenge was doing manual evaluations. Manual evaluations would take a long time and more effort. Mistakes also would have been more likely, and my own observations could be subjective. It was interesting to still think about information design when assessing each website and looking through the results through WAVE. Another challenge was getting perfect graph designs in RStudio. In both error graphs, UW-Green Bay and Missing Label had values of 1. The data labels for those do not appear correctly because of it. A third challenge was using different types of graphs. My data only made sense in a horizontal bar graph. The x-axis titles were long and could not be displayed on the y-axis. The last challenge I had was organizing my data in Excel. I included the Excel file in the Google Drive Folder. I believe it is not organized well, and I could not think of ways to change it. It still worked for what I was trying to do but it could look a lot better. That is why I created separate tables to display information and included them in the report.

A triumph was getting meaningful data from using the WAVE tool. At first, I was getting worried about my data because there were not many errors. The amount of alerts was very useful in analyzing accessibility. Manually evaluating would be very hard and having the tool to do it made everything easier. Another triumph was using RStudio to create the graphs. I included the code I used in the Google Drive Folder. I have previous experience with RStudio in other courses, and I wanted to apply my knowledge. In the beginning of the course, I wanted to use RStudio in a project instead of Excel. I still used Excel to store my data and create tables, but I wanted a challenge when creating my graphs.

How has my understanding of information design changed?

My understanding of information design has changed throughout the course. I came into the class having data science and web design experience but never took the time to learn more about information design. I have learned that information design is more than content looking aesthetically pleasing. It is about creating and presenting content in an accessible way. There are no strict set of rules to follow, and the design needs to fit the context of the audience. The story you are portraying also needs to be clear to the audience so they can learn the content easily. I have also found a new interest in web accessibility that I want to keep exploring and integrating in my other courses.

How have you applied what you learned to other courses, work, or parts of life?

I started to apply what I learned at my job. I work in admissions at Indiana University, but I am placed in a high school in Indianapolis working as a college adviser. I create flyers in Canva that promote Indiana University programs for high school students and other flyers for events at the school. I am also going to start redesigning my personal website based on what I

learned from class and how the accessibility of it can be improved. It is awesome to see how I am applying ideas from class into the real world.

What gaps remain?

A gap that remains for me is learning new software to create content. I have always wanted to learn InDesign and other Adobe programs, but I might have to start learning on my own. But now I feel confident that what I have learned can be applied to new software I am wanting to learn.

Any other thoughts about my experience in the course.

I think this class has given me a solid foundation to keep building upon in other courses and in real world situations. As I said before, I took courses in data science and web design but never dug deep into the rationale behind design choices or accessibility.

References

https://wave.webaim.org/

https://uwm.edu/arc/

https://www.uwec.edu/offices-services/student-success-center/services-students-disabilities

https://www.uwgb.edu/student-accessibility/

https://www.uwlax.edu/disability-resource-center/

https://www.uwosh.edu/cadr/

https://www.uwp.edu/live/offices/accessibilityservices/

https://www.uwplatt.edu/department/disability-access-center

https://www.uwrf.edu/DRC/

https://www.uww.edu/csd

https://www.wisconsin.edu/digital-accessibility/

R-Studio Code

```
install.packages("ggthemes")

#Import Excel Data Just In Case
library(readxl)
Website_Data <- read_excel("Downloads/Website Data.xlsx")
View(Website_Data)


#Website Total Alerts
library(ggplot2)

universities <- c("UW-La Crosse", "UW-Milwaukee", "UW- Eau Claire", "UW-Parkside",
                  "UW-Oshkosh", "UW-Platteville", "UW-River Falls", "UW-Green Bay",
                  "UW-Whitewater")
alert_totals <- c(199, 328, 168, 211, 160, 66, 328, 90, 153)

alert_data <- data.frame(University = universities, Alerts = alert_totals)

ggplot(alert_data, aes(x = reorder(University, -Alerts), y = Alerts)) +
  geom_bar(stat = "identity", fill = "cornflowerblue", width = 0.7) +
  geom_text(aes(label = alert_totals), vjust = 0.5, hjust = 1.5, size = 3.5, color = "white") +
  labs(title = "Total Usability Alerts by University Website",
         x = "University",
         y = "Total Usability Alerts") +
  theme_igray() +
  coord_flip()+
  theme(plot.title = element_text(hjust= 0.5),
         axis.text.x = element_text(angle = 0, hjust = 0.5, vjust = 1),
         axis.text.y = element_text(hjust = 1),
         axis.title.x = element_text(vjust = 1, size = 16),
         axis.title.y = element_text(vjust = 1, size = 16))


#Website Total Errors
library(ggplot2)

universities <- c("UW-Milwaukee","UW-Parkside","UW-Oshkosh", "UW-Platteville",
                  "UW-River Falls", "UW-Green Bay")
error_totals <- c(70, 100, 32, 8, 76,1)
```

```
error_data <- data.frame(University = universities, Errors = error_totals)

ggplot(error_data, aes(x = reorder(University, -Errors), y = Errors)) +
  geom_bar(stat = "identity", fill = "orchid4", width = 0.7) +
  geom_text(aes(label = error_totals), vjust = 0.5, hjust = 1.5, size = 3.5, color = "white") +
  labs(title = "Total Usability Errors by University Website",
          x = "University",
          y = "Total Usability Errors") +
  theme_igray() +
  coord_flip()+
  theme(plot.title = element_text(hjust= 0.5),
          axis.text.x = element_text(angle = 0, hjust = 0.5, vjust = 1),
          axis.text.y = element_text(hjust = 1),
          axis.title.x = element_text(vjust = 1, size = 16),
          axis.title.y = element_text(vjust = 1, size = 16))


#Top Type of Alerts
library(ggplot2)

alert_type_data <- data.frame(
  AlertType = c("Noscript Element", "Redundant Title Text", "Redundant Link",
                "Link to PDF Document", "Possible Heading", "Layout Table",
                "Very Small Text", "Skipped Heading Level", "Underlined Text",
                "Device Dependent Event Handler"),
  TotalOccurrences = c(335, 296, 185, 135, 124, 122, 95, 67, 64, 54)
)

ggplot(alert_type_data, aes(x = reorder(AlertType, -TotalOccurrences), y = TotalOccurrences))
+
  geom_bar(stat = "identity", fill = "palegreen4", width = 0.7)+
  geom_text(aes(label = TotalOccurrences), vjust = 0.5, hjust = 1.5, size = 3.5, color = "white") +
  labs(
          title = "Top 10 Alert Types by Total Occurrences",
          x = "Alert Type",
          y = "Total Occurrences") +
  theme_igray() +
  coord_flip()+
  theme(plot.title = element_text(hjust= 0.5),
          axis.text.x = element_text(angle = 0, hjust = 0.5, vjust = 1),
          axis.text.y = element_text(hjust = 1),
          axis.title.x = element_text(vjust = 1, size = 16),
          axis.title.y = element_text(vjust = 1, size = 16))
```

```
#Top Type of Errors
library(ggplot2)

error_type_data <- data.frame(
  ErrorType = c("Missing Form Label", "Empty Heading","Empty Link",
                "Spacer Image Missing Alt Text", "Linked Image Missing Alt Text",
                "Missing Alt Text", "Missing Label"),
  TotalOccurrences = c(152, 64, 32, 24, 11, 3, 1)
)

ggplot(error_type_data, aes(x = reorder(ErrorType, -TotalOccurrences), y = TotalOccurrences))
+
  geom_bar(stat = "identity", fill = "tomato3", width = 0.7) +
  geom_text(aes(label = TotalOccurrences), vjust = 0.5, hjust = 1.5, size = 3.5, color = "white") +
  labs(
        title = "Top 10 Error Types by Total Occurrences",
        x = "Alert Type",
        y = "Total Occurrences") +
  theme_igray() +
  coord_flip()+
  theme(plot.title = element_text(hjust= 0.5),
        axis.text.x = element_text(angle = 0, hjust = 0.5, vjust = 1),
        axis.text.y = element_text(hjust = 1),
        axis.title.x = element_text(vjust = 1, size = 16),
        axis.title.y = element_text(vjust = 1, size = 16))
```