Cornell Recommended Web Accessibility Testing Process

Last revised: 28 July 2023

Overview

There are four main steps in the web accessibility testing process:

1. Review setup
2. Recording automated test results
3. Manually testing the site/product
4. Creating the review documentation

The following sections go through these steps in detail. The most critical and intensive step is the manual testing; that step identifies problems that *cannot* be identified with automated testing and Siteimprove scans. **The majority of possible accessibility failures can only be identified by manual testing.** Even if you're finding very little in the Siteimprove report and in other automated checks, you may still find large numbers of issues during the manual testing phase.

If you would like to practice identifying accessibility problems, the W3C's Before and After Demonstration site is a great resource. Begin by assessing the Before pages and recording what you find, then turn on the annotations to see if you found all the problems!

Setup

Scope of Review

The WCAG Checklist has a "Scope" sheet; this is the place to record a list of URLs/screens that should be reviewed during the manual review part of the process. Recording the review scope is an important part of documentation that helps subsequent reviewers to confirm what pages were checked manually.

To create the URL scope list, do a quick check of the site to identify examples of all site page templates and pages with unique widgets/components. Add those page URLs to the scope sheet. If, during the course of the review, you identify additional page templates or pages with unique widgets/components, add that page/screen to the Scope sheet so that anyone who looks at the review in the future knows that page was part of the review.

Review Storage

Identify the place where you will store the review documentation and create any needed folders. We recommend creating a specific folder for screenshots, which can make it much easier to identify where you found an issue when you go back to re-check it after remediation. Note that review documentation should be stored somewhere that's easy to find and share with stakeholders.

Recording Tools

WCAG Checklist

Use the WCAG Checklist to record issues that you find. start by making a copy: go to "File," click "Make a copy," and save the copy into your Google Drive. Be sure to set the permissions so that any important stakeholders have the access they need. Make sure to fill out the top row (Site Name, URL, Assessment Date, etc.), including the information about OS and browser. Detailed instructions on how to use the checklist can be viewed on the README Info sheet, including how to generate the summary document, fail priority list, and content document.

> **Note:** We recommend using a file name that will make clear what site it is, when it was reviewed, and what version of WCAG it was reviewed against. For example: "Year Product/Site WCAG 2.x AA Checklist" for the main pattern, yielding a file name such as "2023 Weill Cornell WCAG 2.1 AA Checklist."

It is likely that you will identify multiple distinct failures of individual guidelines. When that occurs, create a new row underneath the existing row for that guideline and copy the contents of the guideline's Columns A-D into the new row. Use that new row to record the new failure of that guideline.

Tips for effective issue recording

- Give the exact URL for at least one page with the issue in the "Target Element/Location" cell
- If the issue is present on more than one page, note that in the "Target Element/Location" cell
- Include a screenshot of the target element as it appears on the rendered page; upload the screenshot to Box and insert the Box link for the image in the "Image Example Link" cell
- In the "Description of issue" box, consider including information about how this barrier could affect users with specific access needs (i.e., "This image has no text alternative,

which means that screen reader users will not have access to the information conveyed by the image").

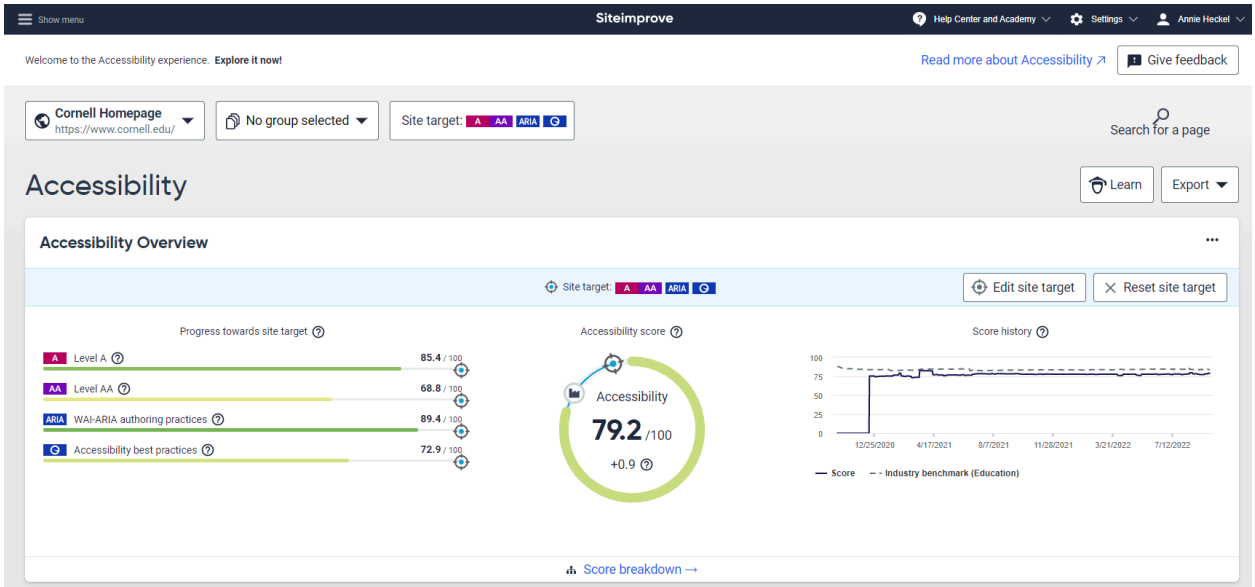
- Use the "Test Method" cell to give information about how you identified the issue.
- The "Expected Behavior" cell does not have to give a code snippet; it simply needs to describe what the element should do, were it to be made accessible–note that you can build this off of the Description text for the guideline (Column D).

Testing Guidelines

The Cornell Web Accessibility Testing Guidelines document gives detailed information on how to test for each guideline listed in the WCAG Checklist, including information about automated tools you can use and manual tests to perform.

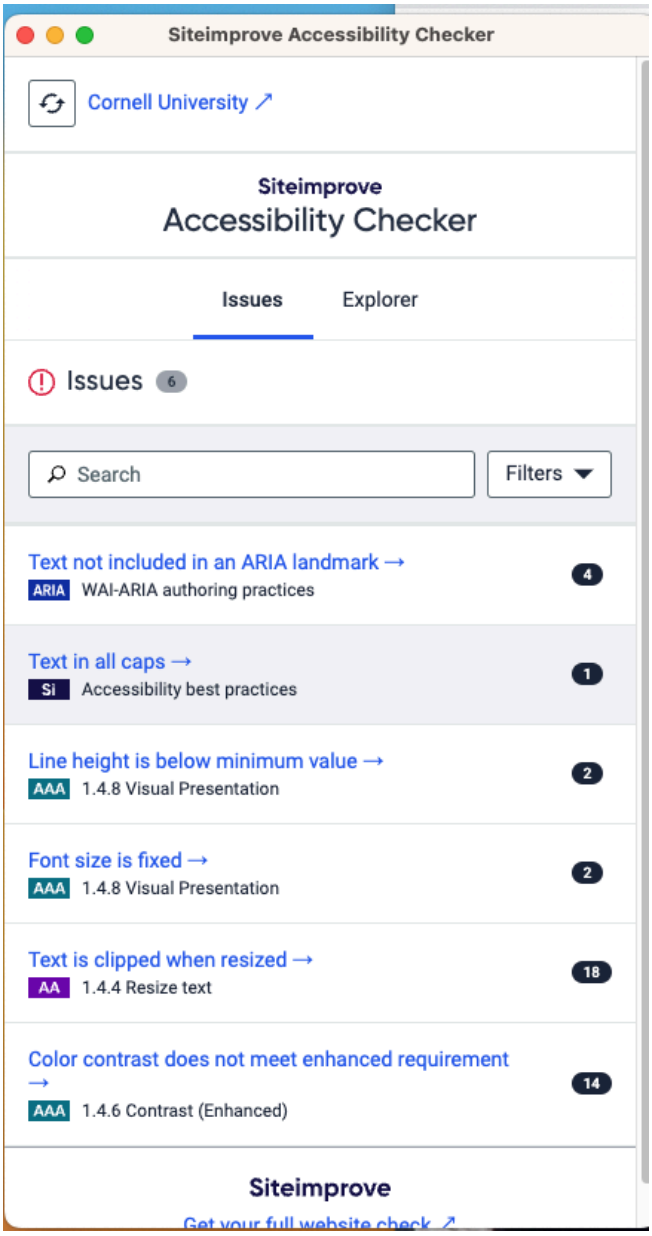
Automated Tests

Siteimprove



If you're reviewing a **website that is not behind a login**, the first stop should be the Siteimprove Accessibility Overview for that site. Begin by reviewing the Issues that Siteimprove has positively identified (click "Issues" in the left-hand navigation or scroll down to "Fix these issues to improve your score" and click the "See all issues" link). Review each issue identified by Siteimprove, ensure that it is a problem on the live page (Siteimprove does sometimes have false positives), and record all confirmed issues in the WCAG checklist (use the "Why is this an issue?" expandable section of the Page Report to determine the applicable WCAG success criterion).

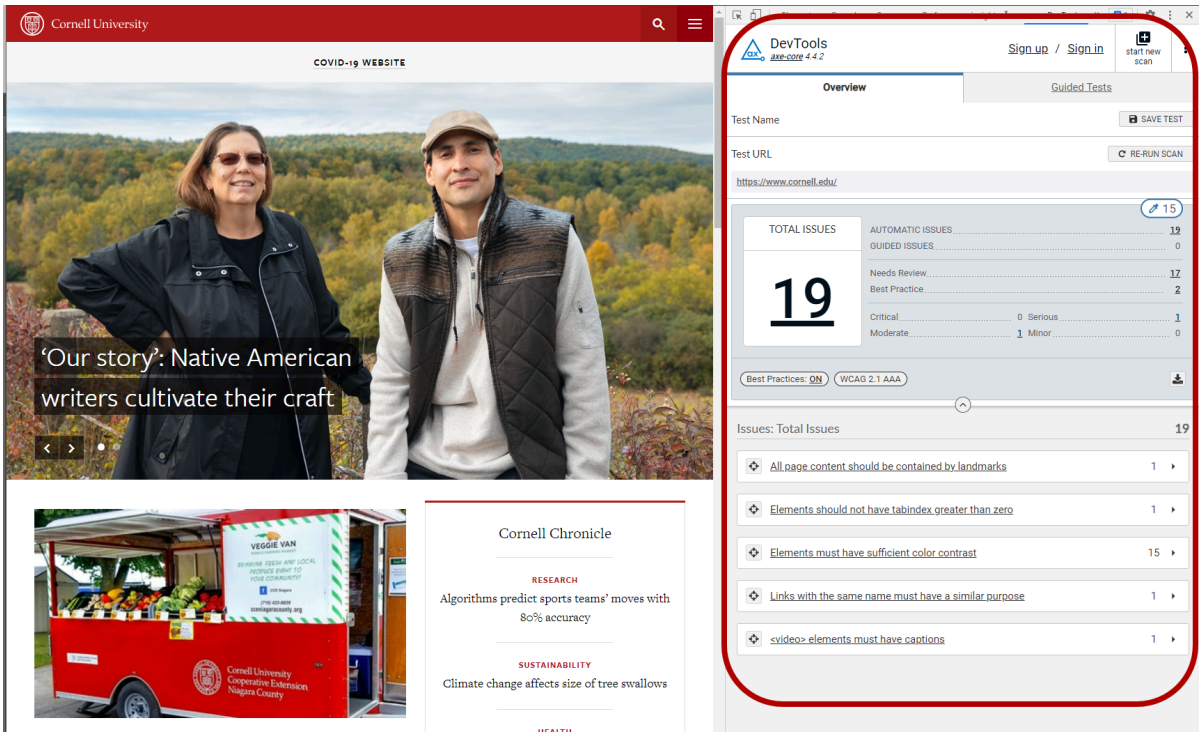
When you have finished confirming and recording the items that Siteimprove identified as "Issues," move on to the "Potential Issues" list. This is a list of items that Siteimprove cannot check directly, but which it can flag for human review. Review these items and record any that are definite WCAG failures.



If you are reviewing a **web app or a website that *is* behind a login**, you will probably not be able to use the Siteimprove Dashboard to review issues, but you can go through the URL scope list one page at a time and use the Siteimprove browser extension to perform the same tests as would otherwise be done by the main Siteimprove Dashboard scan. Confirm and check the identified issues and potential issues as outlined above. When you navigate to the page and click SI extension, a popup window will appear with a list of any issues on the page.

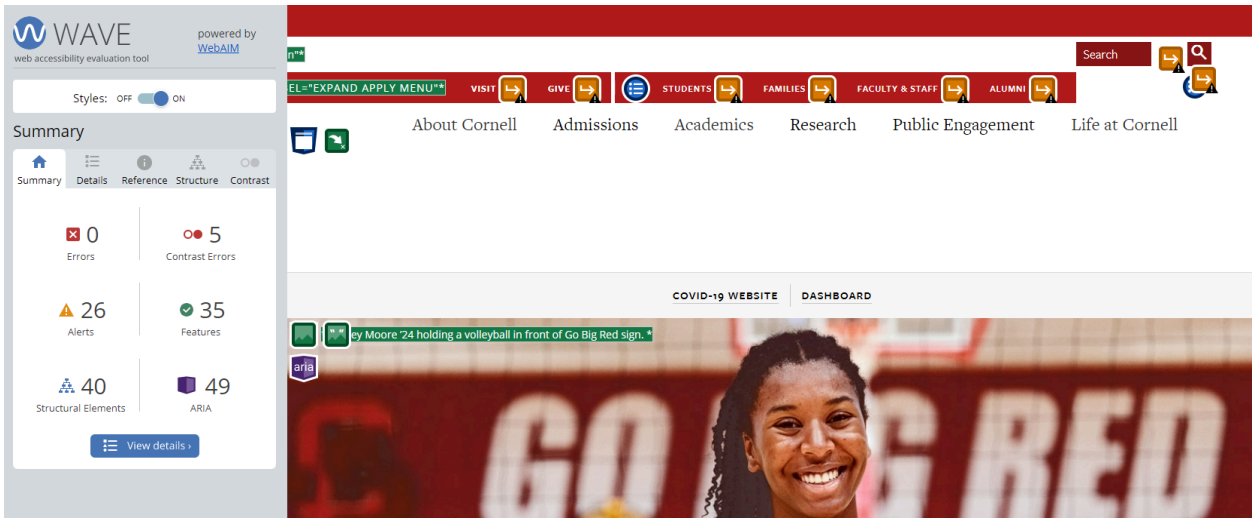
Remember to include a link to the page report for confirmed issues and potential issues (there's a Share link at the top of the Page Report frame) in the "Target Element/Location" cell.

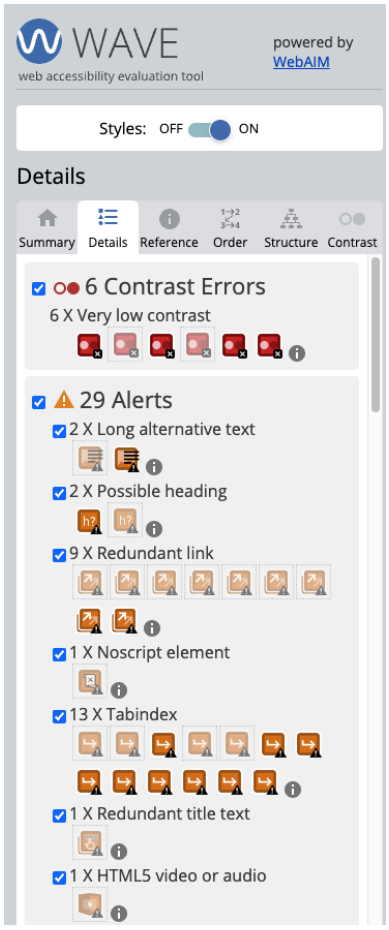
Axe (optional)



Deque's Axe DevTools can be used as a backup to the Siteimprove Dashboard, and as a substitute for the Siteimprove browser extension. Axe is also a browser extension, and as such can access any page that is rendered by your browser, even if it is behind a login. If you register for an Axe account, you will be able to use the "Share Issue" button to share a direct link to specific issues identified by Axe. Axe may have trouble accessing content inside an iframe.

WAVE (optional)

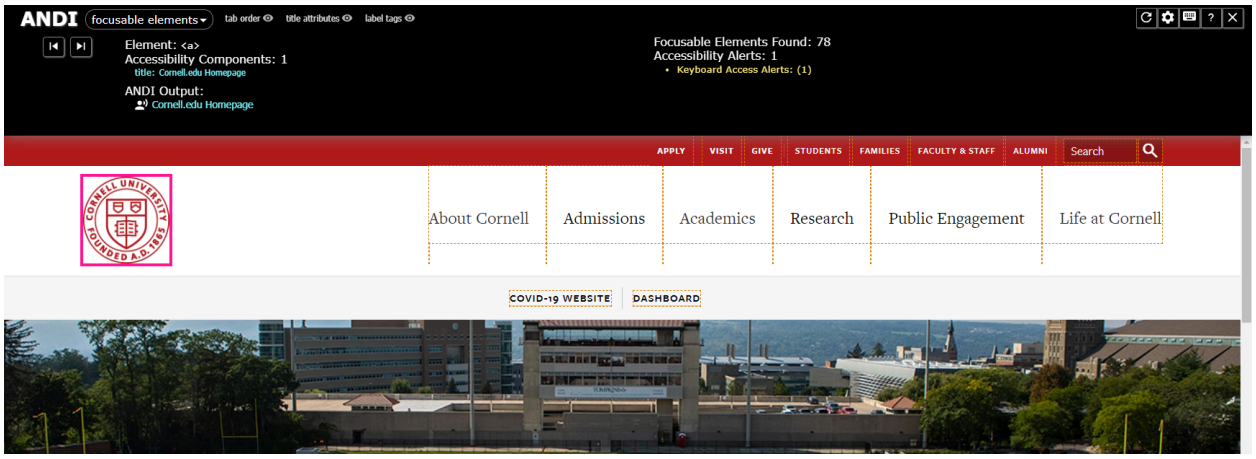




[WebAIM's WAVE](#) is a good tool for reviewers who are less comfortable with directly combing through HTML. WAVE adds visual annotations to the webpage, which can make it quicker and easier to determine if, for instance, all text on a page that looks visually like a heading actually has heading markup (see the "Structure" tab on the WAVE interface), or whether the keyboard focus order for the page's user interface components follows a predictable pattern (see the "Order" tab on the WAVE interface). WAVE does have some fully automated tests, but is particularly helpful for speeding up review tasks that require human assessment, and can be a good tool for reviewers who are not comfortable using screen readers for testing.

Errors in Wave will appear on the details tab under 'errors' and 'alerts.' You do not need to pay attention to anything including 'features' and below.

ANDI (optional)



ANDI (Accessible Name and Description Inspector) is another tool that shows accessibility visually on the rendered webpage. Like WAVE, it can be a good tool for reviewers who are less comfortable with HTML markup. It also has functionality that can make it easier to review the content of iframes; because iframes are essentially webpages embedded in a webpage, browser extension scanning tools have a difficult time checking the content of iframes. ANDI has a function that allows you to open iframes in a new window and check the iframe contents that way, which can be a big help. Like WAVE, ANDI can also be used to mark page organization (headings, lists, etc.) visually on the rendered page, show the tab order, etc., and can be a good tool for reviewers who are not comfortable using screen readers for testing. ANDI does have a bit of a learning curve; head to the SSA website for the ANDI Tutorial if you need some help sorting out how to use ANDI.

Manual Tests

Manual tests will take up the bulk of the time you spend testing. Develop a process with which you're comfortable; some of us go page by page through the Scope list, doing each of the tests below on a page before moving on to the next page, while others prefer to do the keyboard testing for all the pages at once, then go back and check color contrasts on all pages, then check screen reader access, etc. Try out different approaches and determine what works best for you.

Keyboard

Important: Keyboard testing on the Mac iOS requires modifying the default Mac system settings. See the paragraph below on keyboard testing on a Mac for more details.

Keyboard testing covers a range of WCAG success criteria, but the biggest questions to keep in mind with keyboard testing are:

- Can I see, at all times, what user interface component has keyboard focus?
- Can I access all page content, including on-hover content, and user interface components just by using the keyboard?
- Can I perform all page/app functions just by using the keyboard? (Note: if a particular UI component seems to *not* be keyboard accessible, check for keyboard-friendly alternatives to the inaccessible component.)
- Does keyboard focus ever get "trapped" in such a way that I cannot determine how to move focus off/out of an element just by using the keyboard?

To perform keyboard testing on a Windows laptop or PC, simply use the Tab key to try and jump through the different user interface components, and the spacebar or Enter key to activate all user interface components (the Enter key should trigger links, and both the spacebar and Enter key should trigger buttons) and the Esc key to close components that have no other obvious method to close them. Page elements that have on-hover content (content that appears when you hover your mouse cursor over a specific element) should also make that content available to keyboard users. See the Testing Guidelines document for more details.

Keyboard testing on a Mac is a little more complicated. To perform keyboard testing on a Mac, you will need to modify system settings depending on the browser you wish to utilize. For **Chrome**, you should be able to use the full range of keyboard navigation controls out of the box without any modifications to System Settings. For **Firefox**, you need to navigate to macOS System Settings, then Keyboard, and tick the option "Keyboard Navigation." For **Safari**, open Safari and open the Safari item in your menu bar, click Settings, then the Advanced tab, then

tick the “Press Tab to highlight each item on a webpage” checkbox in the Accessibility section. For more details and troubleshooting, see this MacOS User Guide.

After configuring your system settings on your Mac, you can now follow the same instructions for Windows PCs. Both platforms should handle keyboard navigation the same.

Color Contrasts

Many color contrasts can be checked by the automated tools, but if you find elements that have text over a gradient or an image, or ones where either the foreground or background color has not been defined in the style sheet, you will need to check those manually. TPGi’s Color Contrast Analyser is an excellent tool for performing manual color checks, as it allows both direct input of color codes (HEX/HEXa, RGB/RGBa, HSL/HSLa, HSV/HSVa) and the use of an eyedropper tool to sample the colors as rendered by the browser. **Tip:** always include the target contrast ratio in the “Expected Behavior” cell for a color contrast issue.

Zoom & Reflow

Automated tools can identify restrictions on zooming and resizing, but ensuring that content isn’t overlapping when text and other content is zoomed should be verified manually. There are a number of tools you can use for this:

- Browser zooming
- OS default text size settings
- The browser inspector (for simulating mobile interfaces)

Check to ensure that all content and functionality are still available when the text size has been increased and when the viewport has been adjusted to the equivalent of 320 CSS px high and 256 CSS px wide. In particular, ensure that hover content is still available and that you are able to dismiss hover content without moving the pointer/keyboard focus. Also check to see if focus order remains logical when the viewport is resized to a point that a layout change occurs.

Screen reader testing

Screen reader testing is optional as all issues that can be identified by screen readers can also be identified using tools like WAVE or ANDI or by direct inspection of the HTML markup for the page. Learning the basics of screen reader use is recommended, however, and can help you to be more conscious of the problems that users of assistive tech may encounter as they’re attempting to use the web. Screen reader-specific checks include assessing whether you can:

- Access all content and information when the screen reader is running, using only the screen reader keyboard commands

- Perform all page/app functions with the screen reader running (per keyboard function checks, look for alternative paths if you find a component that doesn't work with the screen reader)
- Hear error and status messages without having to navigate to the field with an error
- Hear status updates without having to navigate to the message

Deque has excellent guides to screen reader keyboard controls and swipe gestures on the Deque University Screen Reader Keyboard Shortcuts and Gestures page. It is strongly recommended that you have that page open when using a screen reader; if you're testing on a mobile device, make sure you have that page visible before you turn on VoiceOver or Talkback!

Multimedia supports

Checking multimedia supports can be time-consuming, so you should probably restrict the review to no more than 5 videos unless the client has specifically asked for more videos to be reviewed. Review the videos to ensure that:

- Captions are present and accurate
- All meaningful/important visual information is also conveyed via audio (and if it is not, that an audio described version of the media is available)
- A transcript is present and accurate (optional but strongly recommended if an audio described version is present)
- Links to alternative versions of the media content are presented immediately adjacent to the main media content

PDFs and other files

If PDFs and other files are present, download/open 2-5 and run an accessibility check. You can do this in Word for Word Docs or in Adobe Acrobat if you have a license for Acrobat Pro. If you do not have a license for Acrobat Pro or if Acrobat is not giving clear information, the PDF Accessibility Checker (PAC) is free to download and is generally more reliable than Adobe. You can also browse through the files with a screen reader to see if you can access all content.

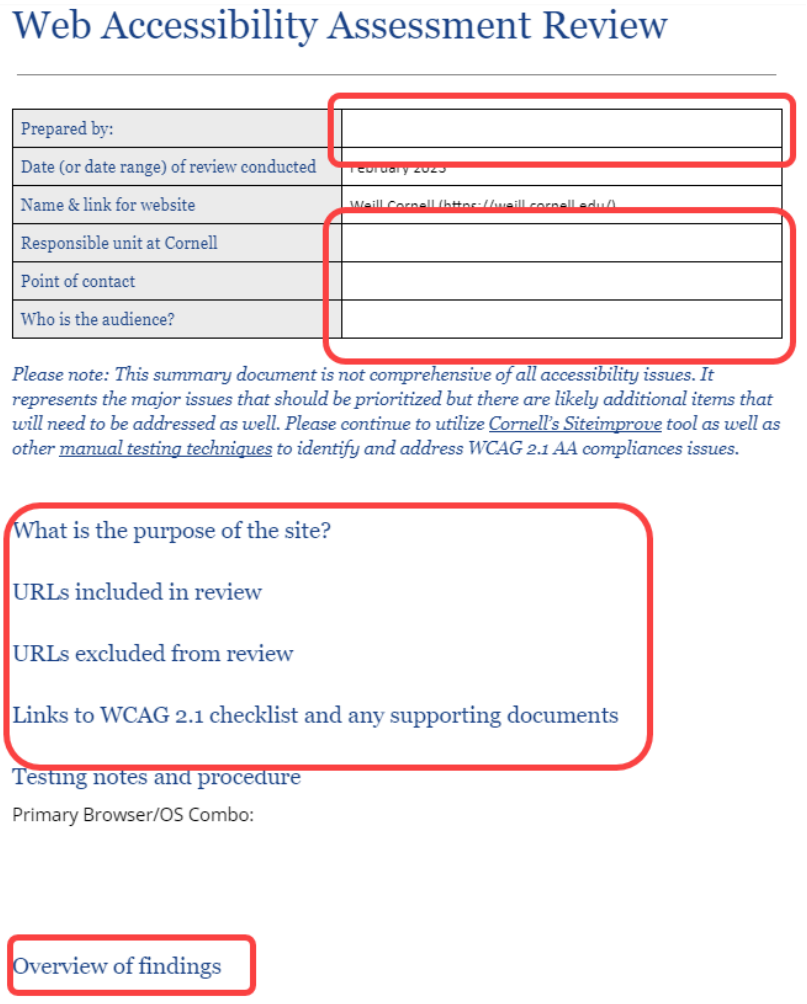
Finishing up

Once the review is finished and you have edited issues for clarity and gotten any feedback you needed, it's time to create the documentation. Two items are standard: the Summary Document (as a Google doc or Word Doc) and the Checklist itself (as a Google Sheet and/or Excel file). We may also send a Content Issues document if the client has requested one.

Create Summary document

To create the Summary document, use the “Summary Doc” button up in the top left corner of the Checklist. You may need to give the script permission to run, in which case you will need to grant permission, then use the button again. A Google Doc of the Summary will be created in your Google Drive. Open that doc and fill in the additional information items:

- Any blank cells in the table at the beginning of the document
- The Purpose, URLs Included, URLS Excluded, and Supporting Documents items
- The Overview of Findings section



Web Accessibility Assessment Review

Prepared by:	
Date (or date range) of review conducted	February 2023
Name & link for website	Weill Cornell (https://weill.cornell.edu/)
Responsible unit at Cornell	
Point of contact	
Who is the audience?	

Please note: This summary document is not comprehensive of all accessibility issues. It represents the major issues that should be prioritized but there are likely additional items that will need to be addressed as well. Please continue to utilize Cornell's Siteimprove tool as well as other manual testing techniques to identify and address WCAG 2.1 AA compliances issues.

What is the purpose of the site?

URLs included in review

URLs excluded from review

Links to WCAG 2.1 checklist and any supporting documents

Testing notes and procedure

Primary Browser/OS Combo:

Overview of findings

For the Overview of Findings section, write up a brief 5-10 sentence summary of what you found during your review, including an overall assessment of the site/app–was it basically accessible? Were there any really major barriers, such as a lack of keyboard support? Was there any aspect of accessibility that was really strong? If this is a review of a site that has been reviewed previously, have there been any significant changes in the level of accessibility since that last review? You don’t need to get too detailed here, but give enough information

that a reader might be able to pinpoint the most critical items that should be prioritized during remediation efforts.

Create Fail Priority Sheet

To create the Fail Priority Sheet, head back to the Google Sheet version of the Checklist and click the “Order Failures” button. This will create a new sheet where the failed issues are situated at the top of the sheet. Make sure that the main checklist sheet is then returned to guideline order (Column A, “GL”).

Create Content Doc (optional)

If the internal client for whom we’re doing the review would like a list of issues that could be corrected by content creators/managers, use the Content Doc button to create a list of all the failures that were marked as Content Entry items in Column M (“CE Issue”).

Share & store

Once all of the documents have been created you can share them with any stakeholders or just ensure that they’re stored in the appropriate folders. If you reviewed a website that has the Core Site designation, please also share the documentation with the Electronic Information Technology Accessibility Manager (EITA manager) so that the review results can be recorded in their documents.