Accessibility Evaluation Report

Research and Improvements for FADGI Open-Source Software Applications

Sep 28, 2023

Joan Hua

AVP | weareavp.com

Glossary

APG — Authoring Practices Guide

ARIA — Accessible Rich Internet Application

W3C — World Wide Web Consortium

WAI — Web Accessibility Initiative

WCAG — Web Content Accessibility Guidelines

POUR — Four principles of WCAG: Perceivable, Operable, Understandable, Robust

USWDS — The United States Web Design System

UI — User interface

CLI — Command Line Interface

GUI — Graphical User Interface

Keyboard Focus — Focus refers to which element on the screen currently receives input from the keyboard.[1]

Focus Indicator — A visual indication of the currently selected element on a page, which enables users who are using keyboard access to know their location while navigating content.[2]

Focus Order — A visual indication of the currently selected element on a page, which enables users who are using keyboard access to know their location while navigating content.[3] This is sometimes referred to as tab order.

Modal — Generally, modals are pieces of text that pop up inside of the main web page window, prompting you to an action or giving you a reminder.[4] When enabled, it does not allow actions on other parts of the page, often indicated by faded colors on the main content.

Semantic Markup or **Semantic HTML** — The use of HTML markup to ensure content is identified by its meaning as opposed to just its appearance. Semantic markup enables screen readers to more accurately and usefully interpret the content of web pages.[5]

[1] https://web.dev/learn/accessibility/focus/
[2] https://www.pearson.com/accessibility-guidelines/glossary.html
[3] https://www.pearson.com/accessibility-guidelines/glossary.html
[4] http://web-accessibility.carnegiemuseums.org/code/dialogs/
[5] https://www.pearson.com/accessibility-guidelines/glossary.html

Contents

Executive Summary

At the Library of Congress, the Federal Agencies Digital Guidelines Initiative (FADGI) seeks to enhance accessibility in open-source desktop applications used in the digital preservation community. Recognizing the limitations in a number of these tools, FADGI launched a project to provide accessibility evaluations of select tools and to make broad recommendations that can be applicable to more digital preservation tools and scenarios.

In spring and summer 2023, FADGI engaged consulting firms Tech for All and AVP to conduct evaluations on three selected open-source desktop software applications. The three applications are **embARC, Handbrake**, **and BWF MetaEdit**. These activities led to general guidelines that provide directions for the future development of open-source software applications that perform digital preservation activities. The embARC evaluation also serves as the basis for the development sprint cycles of the software, which are to incorporate the updates identified to make it more accessible as part of this project.

This report highlights themes surfaced through the accessibility evaluations and provides recommendations for improving accessibility in open-source applications that support digital preservation efforts.

Accessibility Evaluations

All three applications exhibit serious usage barriers for people who are blind or have mobility impairment. None of them conform with the standards overall. Some cases were so severe that the applications were rendered unusable.

Reference Appendix A for detailed accessibility evaluations on each application, and reference Appendix B for questions and answers providing clarifications on how to read the reports.

Recommended Guidelines

- Make use of keyboard control and navigation
- Describe and label interactive elements
- Check for color contrast, and avoid relying on color to convey meaning
- Structure content and create semantic relationships
- Consider digital preservation use cases and how they might differ from those developed for the general public

Community Recommendations

1. **Defined purpose:** Clarify principles, goals, and purpose of the application.
2. **Actual users:** Conduct usability testing with a diverse user base, including users with various disabilities.

3. **Realistic user stories:** Create realistic use cases that are not abstract, and use these as the basis for the requirements of the application.
4. **Continuous feedback:** Create a mechanism to collect direct feedback on the tools' accessibility; remediate accessibility issues when found.
5. **Standardized specifications:** Follow design systems and coding standards.
6. **Tiered prioritization:** Prioritize accessibility issues to fix based on how prevalent an issue is and how much redesign (of color scheme, layout, user journey/interactions, and so on) is involved.
7. **Wide impact:** Make improvements that have a positive impact universally on various groups of users, regardless of whether they are using a piece of assistive technology or experiencing an impairment.

1. Introduction

> *"[Disability] isn't a discrete community or field of interest. It is complex, multifaceted and pervades all kinds of cultural identity through race, socio-economic level, gender identity, and faith. If you create an inaccessible product or service, you are almost guaranteed to be disenfranchising someone, including your future self."*[6]
>
> —Heydon Pickering

Disability is not merely defined by impairments but also other kinds of participation restrictions. There are different levels of impairment. An individual may have more than one—which can cause varying levels of difficulty in completing a task—and almost all humans can experience disability at some point in their life.[7] Digital accessibility is about building interactions and interfaces that give an effective experience to users with disabilities in a given context and those who do not.

At the Library of Congress, the Federal Agencies Digital Guidelines Initiative (FADGI)[8] seeks to enhance accessibility in open-source desktop applications used in the digital preservation community. Recognizing the limitations in a number of these tools, FADGI launched a project to provide accessibility evaluations of select tools and to make broad recommendations that can be applicable to more digital preservation tools and scenarios. The resulting guidelines provide directions for the future development of open-source software applications that perform digital preservation activities.

With an interest in following through on recommendations for an active, well used application, the group chose to evaluate **embARC** with AVP's[9] help. Furthermore, to establish a basis for general recommendations and guidelines, the team evaluated two other desktop applications that are in active development, have wide adoption, and are used by the digital preservation community. These applications were **BWF MetaEdit** and **Handbrake**. Below are more details on the three software applications.[10]

- **embARC** ("metadata embedded for archival content"):
    - enables users to manage (i.e. audit, validate, and correct) embedded metadata in Digital Picture Exchange (DPX) files as individual files or an entire DPX sequence,
    - as well as SMPTE RDD 48-compliant Material Exchange Format (MXF) files, including FFV1 in MXF files.

[6] Heydon Pickering writes on the website heydonworks.com on topics that include inclusive design, design systems, JavaScript and performance, algorithmic web layout, critical theory, and more.
[7] "Disability," World Health Organization, accessed August 24, 2023.
[8] https://www.digitizationguidelines.gov
[9] weareavp.com
[10] "New FADGI Project: Researching Accessibility in Open-Source Digital Preservation Applications," *The Signal* (blog), Library of Congress Blogs, May 30, 2023.

- **BWF MetaEdit**:
    - supports embedding, validating, and exporting of metadata in Broadcast WAVE Format (BWF);
    - supports the FADGI Broadcast WAVE Metadata Embedding Guidelines;
    - initially funded by the Library of Congress and FADGI; designed and led by AVP; developed by MediaArea.
- **Handbrake**:
    - supports multi-platform video transcoding from one format or codec to another.

The evaluation, completed by the accessibility consultancy Tech for All,[11] informs the development roadmap to improve the accessibility of embARC.

AVP prepared this report based on the findings of Tech for All's evaluations. The report identifies and discusses issues and themes surfaced through the accessibility evaluations and provides recommendations for improving accessibility in open-source applications that support digital preservation efforts.

The report is intended for those in the digital preservation community seeking to improve these applications, either through testing, development, or planning.

2. Review of Literature

Users with disabilities represent a large percentage of the population—the World Health Organization estimates this to be 16 percent of the global population at a given time. This includes those temporarily experiencing a disability: "Almost everyone will temporarily or permanently experience disability at some point in their life."[12] It is clear, then, that to design and develop applications that are accessible is to make them usable to a large group of users.

2.1 Accessibility Guidelines and Resources

The **Web Content Accessibility Guidelines** (WCAG)[13] offers the widely accepted standards and guidelines for accessibility compliance and best practices in the United States. Created by the World Wide Web Consortium (W3C) Web Accessibility Initiative (WAI),[14] WCAG offers a set of success criteria that digital tools can follow and be measured against. The guidelines are created primarily for web content, addressing technology-independent applications, and many of the rules can be applied to desktop and mobile applications as well. As of this writing, the standard is WCAG 2.1, while a WCAG 2.2 has been proposed. The website offers a "How to Meet WCAG"[15]

[11] TFAConsulting.com
[12] "Disability," World Health Organization, accessed August 24, 2023.
[13] Web Content Accessibility Guidelines (WCAG) 2.
[14] W3C Web Accessibility Initiative (WAI): Making the Web Accessible.
[15] How to Meet WCAG (Quick Reference): A customizable quick reference to Web Content Accessibility Guidelines (WCAG) 2 requirements (success criteria) and techniques.

quick reference, as well as additional explanations of intent, techniques to meet the criteria, and examples of failures.

WCAG's 78 success criteria are marked with three levels of conformance: Level A, Level AA, and Level AAA (highest). They are separated into four principles, known as POUR:

1. **Perceivable**—information and user interface components must be presentable to users in ways they can perceive.
2. **Operable**—user interface components and navigation must be operable.
3. **Understandable**—information and the operation of the user interface must be understandable.
4. **Robust**—content must be robust enough that it can be interpreted reliably by a wide variety of user agents, including assistive technologies.

The four principles are the foundations of the guidelines. If any of these principles are not true, according to the initiative, "users with disabilities will not be able to use the Web."[16] In most cases, organizations aim for Level AA conformance, while organizations that specialize in accessibility may aim for Level AAA.[17]

United States federal agencies regulated by **Section 508**,[18] which amended the Rehabilitation Act of 1973, includes a recommendation that their websites adhere to Level A and Level AA of the WCAG 2.0. This recommendation was new with the 2017 "refresh" of the amendment, which updated the accessibility requirements for Information and Communication Technology (ICT).[19]

Meeting the higher level also means meeting criteria in a lower level. Below are examples of these criteria:

> ***1.3.1 Info and Relationships*** *(Level A)—Information, structure, and relationships conveyed through presentation can be programmatically determined or are available in text.*
>
> ***1.3.5 Identify Input Purpose*** *(Level AA)—The purpose of each input field collecting information about the user can be programmatically determined when:*
>
> - *The input field serves a purpose identified in the Input Purposes for User Interface Components section; and*
> - *The content is implemented using technologies with support for identifying the expected meaning for form input data.*

[16] "Understanding the Four Principles of Accessibility," W3C.
[17] Laura Kalbag, "Laws and Guidelines," in *Accessibility for Everyone* (New York: A Book Apart, 2017).
[18] "Rehabilitation Act of 1973," the U.S. Access Board, accessed August 23, 2023, https://www.access-board.gov/law/ra.html#section-508-federal-electronic-and-information-technology.
[19] These revised standards and guidelines were published in the Federal Register on January 18, 2017.

> ***1.3.6 Identify Purpose** (Level AAA)—In content implemented using markup languages, the purpose of User Interface Components, icons, and regions can be programmatically determined.*

Resources offered by professionals in digital accessibility—referred to as "a11y" for short—offer helpful explanations that may be easier to understand to accompany the official WCAG and its precise yet inevitably jargony language. Several online resources include Holistica11y[20] and the A11ly Project.[21] The accessibility consultancy TPGi offers a number of useful tips and strategy,[22] such as its "Software Accessibility Checklist: SaaS and ADA Compliance."

In addition to WCAG, there are tools that define development standards; adhering to them avoid usage barriers that arise when the digital application behaves in unexpected ways. **WAI-ARIA**[23] (Accessible Rich Internet Applications Suite) is a resource for advanced user interface controls. The solutions are given with technologies developed with HTML and JavaScript in mind. (Consult WAI-ARIA for meeting criteria such as *4.1.2 Name, Role, Value* from WCAG 2.1.) As part of this suite of web standards, Authoring Practices Guide (APG)[24] is applicable in the usage of complex elements, such as widgets.

For legal compliance, accessibility requirements are covered by access and disability discrimination laws. In the United States, **Section 508** is typically cited. This amendment, made in 1998, requires federal agencies to make their electronic and information technology (EIT) accessible to people with disabilities. With the 2017 update, the definition of covered technologies shifted to Information and Communication Technology (ICT), focusing on their function more than product type. This provides clarification and anticipates the evolution of technology (which increasingly blurs the lines of content and applications developed for the web, mobile devices, and desktop computers). The 2017 update also added more explicit reference to WCAG. It is understood that the guidelines apply to not only web content, as the name suggests, but also software and documents.[25] The result is the ICT Accessibility 508 Standards.[26] Federal websites and digital services must also abide by the **21st Century Integrated Digital Experience Act**,[27] including its website standards, last updated in 2020. The "Checklist of requirements for federal

[20] https://holistica11y.com/category/accessibility/wcag/
[21] https://www.a11yproject.com/posts/#how-to
[22] "Accessibility Strategy," TPGi, accessed August 24, 2023, https://www.tpgi.com/accessibility-strategy/.
[23] https://www.w3.org/WAI/standards-guidelines/aria/
[24] https://www.w3.org/WAI/ARIA/apg/
[25] Matt Feldman, "Section 508 Refresh (Part 1)," TPGi, updated June 29, 2023, https://www.tpgi.com/section-508-refresh-part-1/. See also "Comparison Table of WCAG 2.0 to Original 508 Standards," U.S. Access Board, which explains the incorporation of WCAG 2.0 in the refresh: "WCAG 2.0 is written in a way that is technology neutral and is therefore directly applicable to a wide range of content types and formats."
[26] "Revised 508 Standards and 255 Guidelines," Information and Communication Technology, the U.S. Access Board, August 25, 2023, https://www.access-board.gov/ict/#E101-general.
[27] 21st Century Integrated Digital Experience Act, Pub. L. No. 115-336, 132 Stat. 5025 (2018).

websites and digital services"[28] covers additional laws, policies, and regulations for federal agencies.

USWDS design principles states, "Legal requirements are a critical starting point for factoring accessibility into your decision-making, but these requirements are only the beginning." The **U.S. Web Design System (USWDS)** offers a list of important considerations to embrace accessibility and ensure that the web content and digital systems are created and developed with users in mind and inclusive of their diverse needs.[29]

2.2 Designing Inclusive and Accessible Digital Applications

Simply following the guidelines does not guarantee that the digital application or website will be usable by persons with disabilities. Laura Kalbag writes in *Accessibility for Everyone* that "**accessibility goals are also usability goals. Good accessibility is good usability**."[30] Literature on accessibility for digital products often points out the importance of thinking about the usage scenarios and underlying usability. Some also suggest the potential pitfalls of leading with legal compliance rather than value on the product and user experience.[31] Works such as *A Web for Everyone*,[32] *Beyond Accessibility Compliance*,[33] *Accessibility for Everyone*,[34] "Inclusive Design Principles,"[35] and *Universal Principles of Design*[36] discuss accessibility alongside usability, human-centered design, universal design, and inclusive design. Important steps toward usable and accessible digital applications include:

- Design for a broad range of abilities, styles, and needs
- Conduct user research and usability testing
- Include persons with various disabilities in the feedback cycle
- Evaluate for accessibility ongoingly after the application or product is launched
- Improving an application to be accessible and conform with guidelines ultimately lead to better usability outcomes, adding value

[28] "Checklist of requirements for federal websites and digital services," Digital.gov, U.S. General Services Administration, accessed August 24, 2023, https://digital.gov/resources/checklist-of-requirements-for-federal-digital-services/.
[29] "Design principles," USWDS, U.S. General Services Administration, accessed August 24, 2023, https://designsystem.digital.gov/design-principles/.
[30] See chapter 4, "Content and Design."
[31] See, for example, Chadha's chapter 4, "Gameplan."
[32] Sarah Horton and Whitney Quesenber, *A Web for Everyone: Designing Accessible User Experiences* (Sebastopol: Rosenfeld Media, 2014).
[33] Sukriti Chadha, *Beyond Accessibility Compliance: Building the Next Generation of Inclusive Products* (New York: Apress, 2022), doi:10.1007/978-1-4842-7948-9.
[34] Laura Kalbag, *Accessibility for Everyone* (New York: A Book Apart, 2017).
[35] "Inclusive Design Principles" are broad guidelines given by TPGi: "It's about designing for the needs of people with permanent, temporary, situational, or changing disabilities."
[36] William Lidwell, Kritina Holden, and Jill Butler, "Accessibility," in *Universal Principles of Design*, 3rd ed. (Minneapolis: Rockport Publishers, 2023).

2.3 Open-Source Software

Open-source software allows those in the community to use, modify, or redistribute the software. Though there may be various licenses specifying the distribution terms, open-source means that both the source code and the software are to be freely available.[37] Open-source software invites collaboration, and its value resonates with the library and digital preservation ethos.

While open-source systems are meant to lower the barriers for access in one sense, this does not mean they are inherently accessible. A recurring idea in the literature on the topic of accessibility is that the tools are developed according to standards and standard practices, which relates to a quality of open-source software that prioritizes interoperability. This is not just for compliance and conformance; rather, the outcome supports better usability and accessibility.

One way to achieve this involves following a **design system**, such as USWDS mentioned previously. A design system can benefit the user interface (UI) design team, making the work repeatable and scalable, creating a unified language and style, and allow the tool to be built upon by new generations of the design and development team.[38] As the Nielsen Norman Group explains, a design system consists of two parts: a repository and the people to manage it. The design repository typically includes a style guide, a component library, and a pattern library.

Open-source design systems offer valuable benefits of scalability and replicability. In a set of case studies conducted by the Open Source Working Group (OSWG) of the International Medical Informatics Association (IMIA), the authors suggest that "individual digital health services may not have the budgets or skill sets required to design optimal user experiences independently. In the healthcare sector, this can result in reducing access to services for some of the most vulnerable members of the community who have the highest needs for accessibility and standardisation."[39] They suggest that government-sponsored, open-source projects that implement design systems and frameworks become lessons and resources for future projects. In a piece that supports this idea, Sheri Byrne-Haber points out common problems with open-source technologies that hinder their accessibility: "underestimating cost of open source software" and "skimping on usability."[40] Conversely, the problems of inaccessible systems can be exacerbated if a design system provides building blocks that are not accessible.

Digital projects that utilize design systems and frameworks also benefit the users. A system that offers optimal usability should work as expected and not require the user to put in a great deal of

[37] See definition of "Open Source" given by the Open Source Initiative.

[38] Therese Fessenden, "Design Systems 101," Nielsen Norman Group, published April 11, 2021, https://www.nngroup.com/articles/design-systems-101/.

[39] Chris Paton, Jørn Braa, Andrew Muhire, Luis Marco-Ruiz, Shinji Kobayashi, Hamish Fraser, Luis Falcón, and Alvin Marcelo, "Open Source Digital Health Software for Resilient, Accessible and Equitable Healthcare Systems: Contribution from the IMIA Open Source Working Group," *Yearbook of Medical Informatics* 31, no. 1 (2022): 67–73, doi:10.1055/s-0042-1742508.

[40] Sheri Byrne-Haber, "Accessibility and Open Source," *Sheri Byrne-Haber's Blog*, published August 22, 2019, https://sheribyrnehaber.com/accessibility-and-open-source/.

effort to learn it. Horton and Quesenber put forth "built to standard" as one of the accessible user experience principles (which they cross-reference with WCAG 2.0)—"People feel confident using the design because it is stable, robust, and secure."[41] This is echoed by the inclusive design principles shared by TPGi, which states, "Use familiar conventions and apply them consistently."[42] Not only are these ideas applicable to accessibility criteria, but they improve usability and ultimately create better user experiences. Users' ability and assistive technology, as well as their experiences and preferences, determines their interactions with digital systems. The goal is not to design an application usable by only a specific technology or user group but by a diverse set of users. "The method of interaction should strive to maintain consistency, provide valuable and accurate information, answer in the form of feedback, prevent errors, and resolve the diversity of users and devices," write Briones-Villafuerte et al.[43] They suggest not a one-size-fits-all methodology but that it is important to evaluate the system and UI for different categories of users.

2.4 Digital Preservation: Desktop Applications

With the proliferation of browser-based tools, general users today may interact with digital content on the web more often than via a desktop application. In the profession of **digital preservation**, however, it is common to utilize desktop applications for specific tasks. Digital preservation work commonly involves migrating large sets of data, emulation, preserving the original integrity and technical metadata, manipulating files systematically, editing and reading embedded metadata, and more. These activities may be performed in batch workflows, requiring large processing power, transparent actions, and/or localized settings. They call for highly specialized tools. Many are listed in a compilation of resources called Community Owned digital Preservation Tool Registry (COPTR) Tools Grid[44] and its original predecessor made by Digital POWRR.[45] As of this writing, COPTR lists 594 tools.

The available accessibility resources and established practices address primarily web accessibility. Not as much is published that explicitly examines accessibility for open-source applications used in a desktop environment. With their focus on content distributed via the web, the digital accessibility guidelines and standards are applicable to a broad range of software as a service (SaaS) and cloud-based applications. Albeit not explicitly mentioned, these guidelines are largely relevant to desktop applications as well.

[41] Horton and Quesenber, 51–63.
[42] "Inclusive Design Principles" are broad guidelines given by TPGi: "It's about designing for the needs of people with permanent, temporary, situational, or changing disabilities."
[43] Gabriela Briones-Villafuerte, Alberto Naula-Bone, Mónica Vaca-Cardenas, and Leticia Vaca-Cardenas, "User Interfaces Promoting Appropriate HCI: Systematic Literature Review," *RISTI : Revista Ibérica de Sistemas e Tecnologias de Informação*, no. E47 (2022): 61–76.
[44] "Community Owned digital Preservation Tool Registry (COPTR)," DigiPres Commons, accessed August 25, 2023, https://coptr.digipres.org/index.php/Main_Page.
[45] Preserving digital Objects With Restricted Resources (Digital POWRR) produced an original tool list for the digital preservation community in 2013.

3. Methodology

This project has a number of components. First, it involves surveying the existing open-source desktop applications used to support digital preservation work, making selections for several that are suitable for in-depth accessibility testing. These evaluations were then conducted on three applications, some of them involving multiple use cases and versions for both macOS and Windows operating systems.

AVP's objectives during this phase of the project include:

- Provide broadly applicable recommendations to FADGI and the digital preservation community on accessibility best practices for open-source software
- Make accessibility improvements on embARC so that it serves as an example of implementing accessibility best practices

3.1 Selection for Accessibility Evaluation

AVP worked together with Kate Murray, Digital Projects Coordinator in Digital Collections Management and Services at the Library of Congress, to select applications that serve as examples of open-source, GUI-based desktop applications that support digital preservation efforts. They look for these characteristics and components in applications suitable for accessibility evaluation.[46]

- Open-source
- Free
- Graphical user interface (GUI)
- Active development
- Wide usage (based on indicators like download statistics)

As a result, three applications were chosen. Each has additional versions and use cases that were individually tested. These are in effect seven total application versions:

1. embARC for Windows — DPX use case
2. embARC for Windows — MXF use case
3. embARC for macOS — DPX use case
4. embARC for macOS — MXF use case
5. embARC Command Line (CLI)
6. Handbrake for Windows
7. BWF MetaEdit for Windows
8. BWF MetaEdit for macOS

[46] "New FADGI Project: Researching Accessibility in Open-Source Digital Preservation Applications," *The Signal* (blog), Library of Congress Blogs, May 30, 2023.

3.2 Accessibility Evaluation

Tech for All, Inc. (TFA) analyzed and tested the three applications chosen by FADGI and AVP. They conducted the evaluations in accordance with the WCAG 2.1 (Level AA) and produced a narrative report summary for each of the three tools. In addition, they provided detailed reports in spreadsheet form to list out specific, technical issues in the use cases and cross-referenced them with WCAG guidelines. See the summary and detailed reports in Appendix A.

The TFA team consisted of low vision, blind, hearing, and mobility accessibility experts. They evaluated embARC applications in May 2023 and Handbrake and BWF MetaEdit in August 2023.

TFA evaluated for four general groups experiencing various types of disabilities:

1. People with physical or mobility impairment
2. People who are deaf and hard of hearing
3. People who have low vision
4. People who are legally blind

No specific steps were taken to evaluate the tools for other types of disabilities such as cognitive impairment, learning disabilities, or low literacy.

The following steps were taken to evaluate for these groups:[47]

	Tools Used	Checks
Mobility Impairment	Keyboard	Navigate applications and controls using solely the keyboard, without a mouse
Auditory Impairment	Captioning	Confirm existence of proper captioning
Low Vision	1) Screen magnification, palette, and contrast controls 2) Color Contrast Analyzer version 2.2	Assess color contrast and ability to increase visibility
Blindness	Screen readers *(NVDA and JAWS for Windows; VoiceOver for macOS)*	Navigate applications and controls using solely the screen reader

[47] WAI offers further explanations on the "Diverse Abilities and Barriers" in *How People with Disabilities Use the Web* (https://www.w3.org/WAI/people-use-web/). All of these disabilities are usually experienced in a range—for instance, mild to moderate hearing loss or vision in one or both eyes. It is important to keep in mind that these are not discrete populations. Someone might have a temporary disability, a situational disability, or an age-related disability that develops overtime. Yet someone else might have more than one disability to varying degrees, and not all of them may cause an individual to experience accessibility barriers with a digital tool.

		(completed by an evaluator who is legally blind)

3.3 Synthesis of Findings

This report is a synthesis of these technical evaluations. First, it presents an overview of the accessibility issues flagged in TFA's technical evaluations, comparing the three applications and mapping them to WCAG 2.1 guidelines. Second, it compiles generalizable takeaways that can be applicable to the digital preservation community, beyond the teams that may be maintaining these three specific applications. This is done by inspecting the recurring violations of guidelines across the three applications. Joan Hua at AVP compared the specific technical issues and what they aim to achieve in the applications, which are often geared toward tasks common in digital preservation. Investigating how WCAG guidelines apply to these functionalities, she then identified patterns that suggest problem spots or areas of improvement that are likely fundamental to other similar applications. In parallel, she analyzed a number of sources, such as USWDS documentation and Universal Design principles,[48] to derive process-oriented community recommendations that intend to improve the approach to building and maintaining quality, usable digital tools that do not treat accessibility as an afterthought.

3.4 Limitations

The three selected applications do not offer audio information. As a result, users with auditory impairments can use them without accessibility barriers, and no violation of success criteria affecting this group was flagged. Yet the lack of audio information in these applications also likely means they do not point to attention spots or possible improvements for tools presenting media-rich content. This may be a particular challenge for tools that rely on users to decipher audio information or interact with the application aurally, not those concerning the processing of textual data about the media-rich content. To supplement the evaluations, consult the Accessible Digital Media Guidelines[49] developed by the National Center for Accessible Media at GBH.

4. Evaluation Results

According to TFA's evaluation, **all three applications have serious usage barriers** for people who are blind or have mobility impairment (determined by relying on keyboard control). **None of them**

[48] https://universaldesign.ie/what-is-universal-design/the-7-principles/

[49] "Accessible Digital Media Guidelines: Provide access to multimedia presentations for users with sensory disabilities," NCAM, GBH, accessed August 24, 2023, https://www.wgbh.org/foundation/services/ncam/accessible-digital-media-guidelines-guideline-h-multimedia.

conform with the standards overall. Some cases were so severe that the applications were rendered unusable.

Below is an overview of the conclusions:

- **embARC** — "a number of **severe** accessibility and usability barriers for users who are blind and those with mobility impairment, rendering many essential functions completely inaccessible for these groups"
- **Handbrake** — "a number of accessibility and usability barriers for users who are blind and those with mobility impairment, rendering many functions difficult to use for these groups"
- **BWF MetaEdit** — "a number of **severe** accessibility and usability barriers for users who are blind, those with mobility impairment, and users with low vision, rendering many essential functions completely inaccessible or difficult to use for these groups"

Below is a table to illustrate these conclusions:

	Groups Affected	Accessible?
embARC	● Blindness ● Mobility impairment	Completely inaccessible
Handbrake	● Blindness ● Mobility impairment	Difficult to use
BWF MetaEdit	● Blindness ● Mobility impairment ● Low vision	Completely inaccessible or difficult to use

The issues map to WCAG 2.1 success criteria. Below is a table showing the distribution of these issues.

WCAG 2.1 Success Criteria	embARC	Handbrake	BWF MetaEdit
Principle 1 – Perceivable			
1.1.1 Non-text Content	✘		✘
1.3.1 Info and Relationships	✘	✘	
1.3.2 Meaningful Sequence	✘		
1.4.3 Contrast (Minimum)	✘	✘	✘
1.4.11 Non-text Contrast	✘		

1.4.13 Content on Hover or Focus	✘	✘	
Principle 2 – Operable			
2.1.1 Keyboard	✘	✘	✘
2.1.2 No Keyboard Trap			✘
2.4.2 Page Titled	✘	✘	
2.4.3 Focus Order	✘	✘	✘
2.4.4 Link Purpose (In Context)		✘	
2.4.7 Focus Visible	✘	✘	✘
2.5.3 Label in Name		✘	
Principle 3 – Understandable			
3.3.2 Labels or Instructions	✘	✘	✘
Principle 4 – Robust			
4.1.2 Name, Role, Value	✘	✘	✘
4.1.3 Status Messages		✘	

4.1 Comparison and themes

The table above shows the WCAG 2.1 success criteria with which the issues identified in each application do not conform. Every cell marked with a black ✘ symbol shows that the tool has one or more issues that do not meet the criteria listed in the row. This table maps out an at-a-glance view of the standard criteria that embARC, Handbrake, and BWF MetaEdit most frequently fail to meet. Some observations can be derived:

4.1.1 embARC struggles to achieve Principle 1

Most of embARC's issues violate the first principle—Perceivable—of WCAG 2.1. Perceivable is distinguished from another similar sounding principle, Understandable, in that it has to do with whether or not the user can *perceive* through different senses the existence of an element or component in the digital application. When color contrast is not sufficient, for instance, and particularly when the color contrast for keyboard focus (the marker on screen showing the focus, typically apparent while using a keyboard and its `TAB` key) is insufficient, the user cannot *perceive* what step it is on. The frequent issues may have to do with the color scheme currently designed for embARC. More importantly, many windows or functions are not announced by screen readers, as some controls—such as checkboxes, buttons, editable fields—are not labeled.

4.1.2 More issues relate to Principle 2

Together, the three applications most frequently fail to meet criteria designed to achieve the second principle—Operable—of WCAG 2.1.

There are a number of issues related to the 2.4 series—Navigable—suggesting issues related to navigation. Criteria in this series require the application to "provide ways to help users navigate, find content, and determine where they are" by creating focus order.

4.1.3 Fewer issues with Principles 3 and 4, but two criteria call for extra attention

There are generally fewer criteria in Principles 3 (Understandable) and 4 (Robust) of WCAG. In general, the applications also show fewer issues in the two areas. They should not be overlooked, however. A notable number of issues with all applications evaluated violate criteria 3.3.2 Labels or Instructions.

Examples of where in embARC these violations show up:

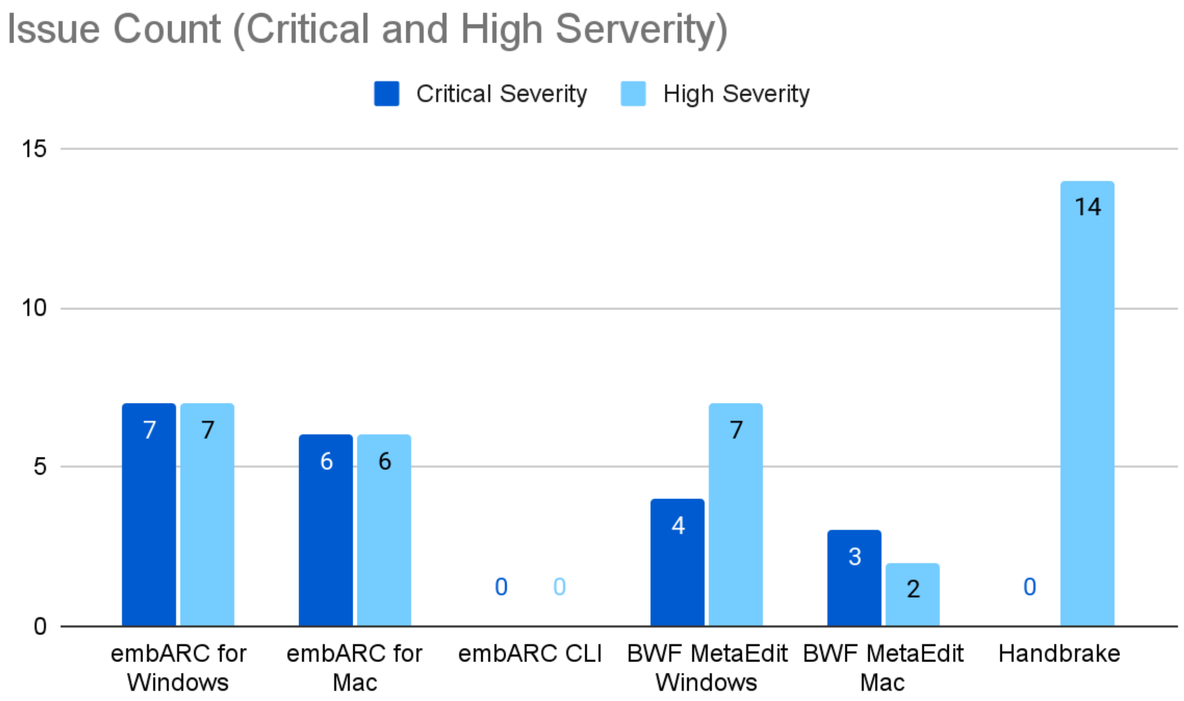
- Sidebar controls are not labeled in the code, even though there is visible text displayed (not explicitly associated with the controls). As a result, screen readers announce these controls as merely "checkbox" or "edit." The user using a screen reader cannot know *what* is being checked or edited.
- "Pop-out" buttons ↗ are not labeled in the code and therefore not announced by screen readers.
- Image pop-up is not announced by screen readers.

4.1.4 embARC shows the highest count of critical issues

Comparing across the three tools, it is apparent that both Windows and macOS applications of embARC have the greatest number of *critical* issues. On the other hand, Handbrake has no issue rated as critical. As a result, the application is generally considered difficult to use, while embARC and BWF MetaEdit are completely inaccessible. Reference chart below, titled "Issue Count (Critical and High Severity)."

- **Critical Severity:** Causes portions of the site to be extremely difficult to impossible to use.
- **High Severity:** Causes portions of the site to be difficult to use.

The Windows version of embARC performed slightly worse than the macOS version, due to the difference in the information each environment's screen readers were able to pick up. The command line interface application of embARC has no issues. But, as the TFA report points out, the CLI version—while clearly navigable by screen readers and keyboard controls—may be harder for general users to use and require a higher level of technical familiarity with the application.

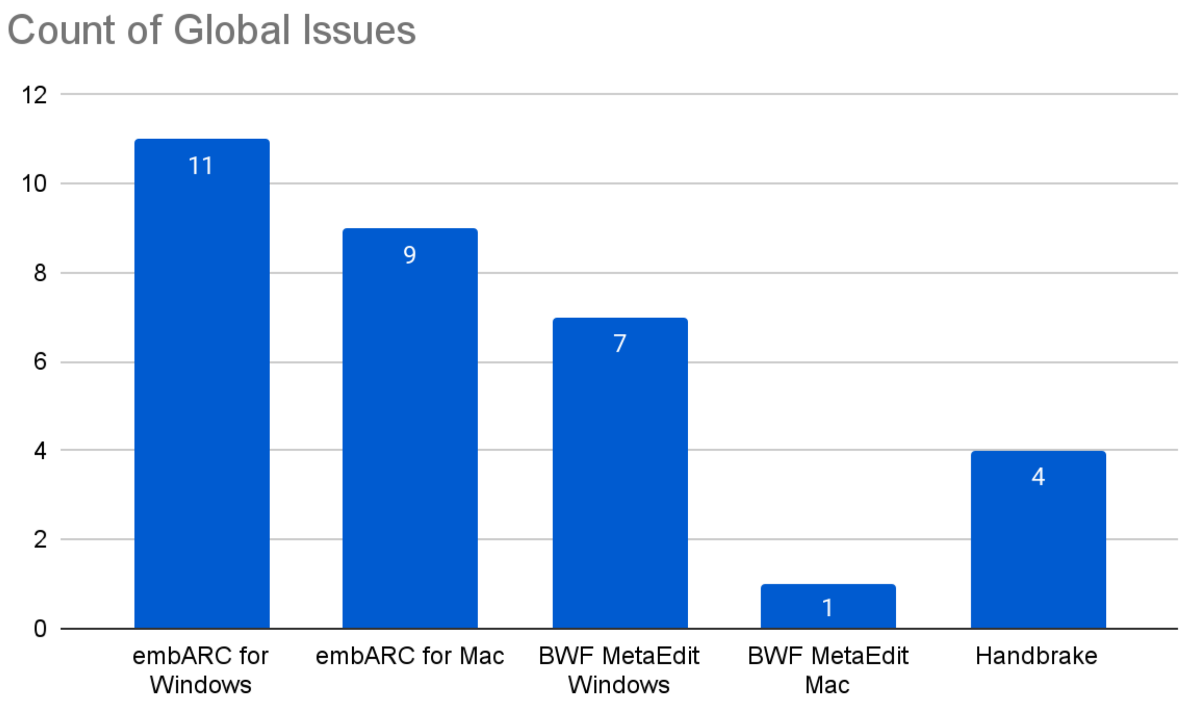


4.1.5 Both Windows and macOS applications of embARC have the greatest number of *global* issues

Global issues are issues that come with certain components, such as all data tables or all pop-up dialogues, in an application. The definition supplied by TFA is: "These are the issues that either occur on all screens of the application (common components like the menu bar) or occur with all components of this type used throughout the application. For example, if all grids have the same issue, we would call it out as 'Global.'"

Global issues indicate barriers that are prevalent in the application, regardless of the specific user journey. They also indicate problem spots to prioritize—both because of the high visibility and the positive impact of fixing the issue properly.

Reference chart below, titled "Count of Global Issues."



4.2 Problem spots in each application

The table below focuses on only WCAG 2.1 success criteria with which two or more issues in each application do not conform, omitting the criteria where the applications did not show significant issues. For embARC and BWF MetaEdit, the issue counts represent the WIndows version, as it is the most commonly downloaded and used.

Attention Spots (2 or more issues)	**embARC**	**Handbrake**	**BWF MetaEdit**
1.1.1 Non-text Content	3 issues	0 issue	1 issue
1.4.3 Contrast (Minimum)	2 issues	1 issue	4 issues
2.1.1 Keyboard	4 issues	3 issues	6 issues
2.4.3 Focus Order	5 issues	3 issues	3 issues
2.4.7 Focus Visible	5 issues	1 issue	1 issue
2.5.3 Label in Name	0 issue	3 issues	0 issue
3.3.2 Labels or Instructions	6 issues	9 issues	2 issues

4.1.2 Name, Role, Value	5 issues	1 issue	3 issues
4.1.3 Status Messages	0 issue	2 issues	0 issue

	embARC	**Handbrake**	**BWF MetaEdit**
Problem Spots in Application	● Application window ● Data table ● Sidebar	● Summary of the file ● Open files from the local environment	● Application window ● File screen ● Tech screen

4.2.1 embARC

A number of issues with embARC are **encountered at the splash screen, blocking the rest of the user experience**.

The user experience of embARC begins on the splash screen (the very first screen presented after the application is launched), where users begin either the DPX or MXF use case by selecting the file. If a user relies on the screen reader, they would not be able to perceive this step as it is not announced. If a user relies on a keyboard, they too would be blocked at this step, as the selection of files can only be completed using a mouse.

As shown in the table above, the accessibility issues throughout the application occur largely in keyboard control, keyboard focus (the marker on screen showing the focus, typically apparent while using a keyboard and its `TAB` key), and labels of elements in the code.

When it comes to **data tables**, they are navigable with neither a screen reader nor a keyboard. It is not possible to traverse between columns to get header information, and no announcements are made when the user selects or un-selects a row from the table. **Buttons launched from the sidebar**, such as "pop-out" ↗ and download ⬇ buttons are not exposed to screen readers.

A number of embARC issues map to criteria 4.1.2. This tends to affect spots in embARC such as:

- dialog window when the files are being processed;
- text elements in the sidebar, which are currently not exposed to screen readers and not focusable, or selectable by keyboards; and
- buttons in the sidebar that currently do not receive keyboard focus.

4.2.2 Handbrake

Handbrake was tested following three use cases:

1. Standard use case: Generate an access copy from a single video

2. Standard use case: Generate access copies from a batch of videos
3. Creating and applying presets

In this application, many barriers were encountered in the first use case of working on a single video, but the issues are spread across all use cases. They are the most prevalent in the application pages that show a **summary of the file** and for **opening files from the local environment** (machine or local desktop).

The common issues have to do with the elements not receiving keyboard focus. For example, on this summary screen, the preview controls do not receive focus and can only be operated with a mouse. Another issue is with labeling or exposing elements to the screen reader. For example, when switching between files in the batch processing use case, it is nearly impossible for the user operating on a screen reader to select a specific file from the Title drop-down, as only the file number and length are announced. All non-interactive content in Handbrake also is not announced by screen readers and does not receive keyboard focus.

4.2.3 BWF MetaEdit

BWF MetaEdit was tested following three use cases:

1. Standard use case: Open one or more WAV files for validation
2. Changing the validation preferences
3. Editing and saving embedded metadata

More issues were related to color contrast and to keyboard navigation. A standard use case requires the user to interact with elements on the file screen (where an overview of the file's basic technical metadata is presented), yet it is not possible to activate any of the controls on this screen, including edit, save, and remove ✏ 💾 ✕ buttons and any of the pop-out ⧉ buttons, using a keyboard.

Another critical issue is in the third use case of editing metadata. While labeling interactive elements is the minimum, in this use case, it is necessary to understand all information presented in this grid view; the column headers and all static information are currently not accessible by screen readers.

See the summary and detailed reports in Appendix A.

5. Recommended Guidelines

5.1 Make use of keyboard control and navigation

Applications must create logical tab order, or focus order, so that the keyboard user can properly move around the applications and complete tasks. This also involves making sure the keyboard focus (the marker on screen showing the focus, typically apparent while using a keyboard and its `TAB` key) is clearly visible using appropriate style and color contrast. The order in which elements receive input and whether that is programmatically determined also affect the experience operating with a keyboard.

The ability to use keyboard control is important to various user groups, including those using assistive technologies, those struggling with fine motor disabilities, and even those with a temporarily sprained wrist. Keyboard control also helps those without disabilities, as they "may choose to use a keyboard to navigate a page due to personal preference, efficiency, or broken hardware."[50]

All three applications exhibited issues concentrated around keyboard navigation and focus order. Some issues were encountered repeatedly in the following criteria of WCAG 2.1:

> ***2.1.1 Keyboard*** *(Level A)—All functionality of the content is operable through a keyboard interface without requiring specific timings for individual keystrokes, except where the underlying function requires input that depends on the path of the user's movement and not just the endpoints.*
>
> ***2.4.3 Focus Order*** *(Level A)—If a Web page can be navigated sequentially and the navigation sequences affect meaning or operation, focusable components receive focus in an order that preserves meaning and operability.*
>
> ***2.4.7 Focus Visible*** *(Level AA)—Any keyboard operable user interface has a mode of operation where the keyboard focus indicator is visible.*

This suggests that these three success criteria represent a problem spot for other open-source desktop applications developed for digital preservation work.

5.2 Describe and label interactive elements

To make an application understandable and allow users to easily operate it, the tool should provide input assistance, making it clear how controls are supposed to be used. This may be overlooked when a type of input, such as a button, is represented on the graphical user interface using a symbol or image that provides only a visual cue for its function, yet not an explicitly named one that can be accessible to screen readers. It may simply make the application difficult to use for any type of user. For instance, if a button showing a folder icon is not labeled, a user may guess it means one can press on it to launch a window to select a file from the local machine when in fact it can mean a command to scan a folder. When the same button has text visually near it but not associated with it as a label, a user can only infer the label. Aside from this button example, a best practice is to identify and describe interactive elements, such as pop-up

[50] https://web.dev/learn/accessibility/focus/

windows, check-boxes, and tables and their column headers. Doing so achieves conformance with guidelines, especially 3.3.2 Labels or Instructions.

> ***3.3.2 Labels or Instructions*** *(Level A)—Labels or instructions are provided when content requires user input.*

Note that alternative methods can also help achieve the goal of providing accessible names and descriptions. See recommendation 5.4 below.

5.3 Check for color contrast, and avoid relying on color to convey meaning

Color contrast affects readability. Users with low vision, color blindness, or other sight issues may struggle to process content with low color contrast. The issue may show up in keyboard focus or when an element changes dynamically based on user input. When these dynamic changes are difficult to perceive, they impede the functionality of the application.

The color contrast success criteria of WCAG 2.1 give tiered requirements based on the size of the text and the context in which it appears.

> ***1.4.3 Contrast (Minimum)*** *(Level AA)—The visual presentation of text and images of text has a contrast ratio of at least 4.5:1, except for the following:*
>
> - *Large Text: Large-scale text and images of large-scale text have a contrast ratio of at least 3:1;*
> - *Incidental: Text or images of text that are part of an inactive user interface component, that are pure decoration, that are not visible to anyone, or that are part of a picture that contains significant other visual content, have no contrast requirement.*
> - *Logotypes: Text that is part of a logo or brand name has no contrast requirement.*

There are online color contrast checkers that help achieve this. Web AIM's contrast checker[51] Color Oracle,[52] and Paul Tol's color schemes and templates[53] are just some examples. In addition to meeting color contrast requirements, consider the readability and usability of an interface. Using color alone, and even visual cues alone, to convey meaning can often cause confusion. Instead, combining colors with textures, textual labels, or symbols can provide a better user experience.

[51] https://webaim.org/resources/contrastchecker/
[52] https://www.colororacle.org/
[53] https://personal.sron.nl/~pault/

5.4 Structure content and create semantic relationships

It is a recommended practice to programmatically code the roles of interactive elements, and make explicit relationships between elements. This allows assistive technology to access the interface.

The evaluations on embARC, Handbrake, and BWF MetaEdit reveal issues that result in static text being inaccessible by screen readers or announced inaccurately. Similar to the section 5.2 recommendation above, this happens when the layout is coded to make sense only visually, but the elements' relationships are not established semantically for assistive technology to understand.

Some of these issues map to criteria 4.1.2 of WCAG 2.1, which states:

> ***4.1.2 Name, Role, Value*** *(Level A)—For all user interface components (including but not limited to: form elements, links and components generated by scripts), the name and role can be programmatically determined; states, properties, and values that can be set by the user can be programmatically set; and notification of changes to these items is available to user agents, including assistive technologies.*

WCAG provides explanations of common failures.[54] One of them occurs when interface controls are made using generic HTML elements like `<div>` and `<span>`. Instead of generic elements, use elements that have defined roles, such as links and buttons. There are additional resources that help implement this technique. See, for instance, further explanation on semantic HTML on Carnegie Museums of Pittsburgh's web accessibility guidelines[55] and Mozilla's MDN Web Docs on the subject.[56] Resources such as IBM Accessibility Requirements[57] reference WCAG 2.1 and Section 508 for both web and non-web software alike.

For web applications, this criterion is relatively easy to meet, as standard HTML controls already enable this. The fact that the evaluated tools failed to pass this criterion likely suggests that it needs special attention in the case of desktop applications and complex controls.

For desktop applications, the principle is the same, but the techniques for achieving this may depend on the programming language used. If the UI is delivered through JavaScript, HTML, and CSS, the WCAG criteria are applicable. Today's desktop applications may be developed using web technologies and embedding a framework such as Chromium, presenting a GUI with web application components and web browsing functionalities. Uses of an application framework that supports programmatically accessible UI help ease the burden on individual application developers, although using frameworks does not automatically make an application accessible. If

[54] https://www.w3.org/WAI/WCAG21/quickref/#name-role-value
[55] http://web-accessibility.carnegiemuseums.org/foundations/semantic/
[56] https://developer.mozilla.org/en-US/docs/Glossary/Semantics#semantics_in_html
[57] https://www.ibm.com/able/requirements/requirements/

the UI is built for a command-line script or application, then accessibility is achieved in the backend portion of the code.[58]

5.5 Consider digital preservation use cases

5.5.1 Pop-out windows

A number of the issues highlighted by the three applications evaluated demonstrate challenges with pop-out windows. These are perhaps more common in desktop interfaces than browser-based interfaces. They are also heavily used in BWF MetaEdit and present in embARC. Unfortunately, these pop-out windows are often not labeled properly and cannot be accessed by a screen reader. They also affect keyboard controls. In one critical example, controls in the pop-out screens from the Tech screen of BWF MetaEdit are completely inaccessible. TFA describes the user experience of editing metadata in their detailed report (Appendix A):

> "Selecting any of the editable fields with the Spacebar opens additional windows. These windows contain unlabeled controls, rendering them completely inaccessible. Windows containing tables and textarea fields trap keyboard in those controls, making the rest of the window controls and text unreachable."

When using such pop-out windows, it is important to consider **the buttons that initiate the pop-out**, **the labeling of windows and dialog boxes**, and how one can **dismiss such windows using solely a keyboard**. In some instances, it would be appropriate to reconsider this design and deliberately **limit the use of secondary or pop-out windows**.

5.5.2 Tables

As many tools used to accomplish digital preservation tasks are metadata-driven, their interfaces often utilize tables and present data in grid form. Meeting the WCAG guidelines is the minimum, yet tables can still remain inaccessible even when they pass the criteria. For some of these digital preservation tools, having inaccessible tables can cause serious usability barriers or make a tool unusable to those relying on assistive technologies.

For digital preservation applications with heavy uses of tables, especially those that show editable or interactive controls in grid form, **ensure that the tables are easy to use and navigate with a keyboard and using assistive technologies**. Attention spots may include:

- The ability to traverse columns and rows using TAB on the keyboard.
- The ability to understand the column and row headers, as well as other essential yet noninteractive elements, of the table using screen readers.

[58] There are currently no standards addressing accessible UI for command line applications, but consider this paper: Harini Sampath, Alice Merrick, and Andrew Macvean, "Accessibility of Command Line Interfaces" *CHI '21: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (May 2021), doi:10.1145/3411764.3445544.

- If using parts of the table to launch additional edits or pop-up windows, consistently setting the keyboard focus so that it returns to a logical part of the table after complex interactions. (For example, if the intended workflow is to launch an edit window to edit a cell, then, after closing that window, the keyboard focus ought to return to where it was before launching the window, rather than back to the top of the page.)

Tables may need to be designed to a higher standard than what is required, and that could make justification difficult. As Sarah Horton writes:

> "I often encounter things in the evaluation process that I believe impact the experience of people with disabilities, but that do not have an associated WCAG 2.0 success criteria I can reference. For example, a complex table where it's difficult to track across rows, particularly using screen magnification … Sometimes I will note this type of issue when reporting, but I know without the backing of a standard, it's difficult to make a strong case for fixing it."[59]

This speaks also to the importance of conducting proper testing and, ideally, user research early in the process and for accessibility evaluations. The design decisions to use tables and pop-up windows should support user experience and accessibility.

5.5.3 Access local environment

Desktop applications used to accomplish digital preservation tasks often involve a component of accessing the local environment (e.g., modifying a file saved on the local machine). This requires the interface to pass controls between the application and the local machine, which may also trigger pop-up windows from the application or the local machine.

In the case where the control may be handed over to the local machine, it is best to limit custom interfaces. Make judicious design decisions to prioritize usability in these complex interactions. If custom interfaces are necessary within the application—for previewing the file or editing the embedded metadata, for example—ensure the components receive keyboard focus and are accessible by screen readers. Where secondary windows and dialog boxes are necessary, ensure they follow standards as much as possible, are labeled and announced by screen readers, and can be launched and dismissed using a keyboard.

5.5.4 Digital preservation user

It is likely that typical users of the open-source desktop applications designed for digital preservation are using them for professional work. This report does not attempt to create personas or user stories for these tools, but the target users—unlike perhaps that of a government website or general mobile application—are those with technical or specialized needs. The tasks the software helps to accomplish may be more complex. Some of these needs

[59] Sarah Horton, "The Value of Involving People with Disabilities in User Research," User Testing, March 4, 2014, https://www.usertesting.com/blog/the-value-of-involving-people-with-disabilities-in-user-research.

may take extra steps to ensure they are addressed with accessible applications. Meeting web accessibility guidelines alone may not be sufficient.

When a functionality is poor in usability, it is possible the digital preservation user may spend extra time troubleshooting or maneuvering their way around a barrier compared to a casual user browsing a website. But when a functionality is inaccessible, it stands in the way of people completing professional work, and there are likely no alternative tools for the same tasks.

6. Improvements on embARC

One of the goals of this project is to offer input to the embARC development team to update embARC for improved accessibility. The accessibility evaluations on embARC (Appendix A) inform the community guidelines and recommendations in this report. This final report, along with the evaluations, serves as the basis for the creation of development stories for embARC's development sprint cycles. The embARC development team will incorporate them as next steps to resolve the issues reported. The work to make accessibility improvements will take place in parallel with ongoing developments of embARC and will not inhibit existing functionality.

7. Community Recommendations

The community recommendations in this section are process-oriented and conceptual. They provide broader, more general guidance for readers who are not directly working on an application but may be making decisions about processes—from usability testing to resource allocation.

This report recommends considering both the **big picture** for creating accessible applications, focusing on usability, as well as desired outcomes **within the application**. These general recommendations are derived from the technical recommended guidelines outlined above. They spotlight the areas that warrant special attention when it comes to digital preservation tools, as evidenced by the accessibility evaluations of embARC, Handbrake, and BWF MetaEdit—three representative open-source desktop applications that support digital preservation work—and demonstrated throughout this report. Based on the themes that emerged in the analysis, this report offers the takeaways below.

7.1 Key takeaways

1. **Defined purpose:** Clarify principles, goals, and purpose of the application.
2. **Actual users:** Conduct usability testing with a diverse user base, including users with various disabilities.
3. **Realistic user stories:** Create realistic use cases that are not abstract, and use these as the basis for the requirements of the application.

4. **Continuous feedback:** Create a mechanism to collect direct feedback on the tools' accessibility; remediate accessibility issues when found.
5. **Standardized specifications:** Follow design systems and coding standards.
6. **Tiered prioritization:** Prioritize accessibility issues to fix based on how prevalent an issue is and how much redesign (of color scheme, layout, user journey/interactions, and so on) is involved.
7. **Wide impact:** Make improvements that have a positive impact universally on various groups of users, regardless of whether they are using a piece of assistive technology or experiencing an impairment.

7.2 Outcomes within an application

1. Users have comparable experiences regardless of their usage of assistive technology.
2. Users can perceive functionalities and information. They can see that it is present.
3. Users can interpret or understand how to use the site: feedback is given within the system when users click around, and instructions are in plain language.
4. Users can navigate the interface using a keyboard, and the arrangement of components and keyboard focus are in logical order.
5. The system has mechanisms in place to prevent user errors and anticipate or forgive them.
6. Users can interpret or understand the meaning of the information presented.
7. Users can complete the tasks or complete the journey of a use case.
8. Use pop-ups, modals, and disappearing objects judiciously, and avoid them when they are not necessary.
9. Avoid surprises, and build elements according to standards.

7.3 Final considerations

There are opportunities to uphold a commitment to accessible application development throughout the lifecycle of an application, from conceptualization to maintenance. Improvements can be made in the development methodology, release and maintenance schedule, mechanisms for tracking changes and bugs for both better usability and accessibility.

Design methodologies may be applied to the process of solving an accessibility problem. Rather than focusing only on the guidelines and specifications, and only on how to fix the violations, it is more valuable to consider how the products will be used. When thinking beyond the piece of technology but about the social and environmental contexts, the solution is oriented toward the people who use the technology. A further step from usability may yield innovative ideas, leading to broader solutions that address not just incremental fixes but lasting benefits to members of the digital preservation community.

Appendix A: Detailed Accessibility Evaluations

embARC

1. Detailed reports of accessibility evaluation (Windows and macOS)
2. Summary report

Handbrake

3. Detailed reports of accessibility evaluation
4. Summary report

BWF MetaEdit

5. Detailed report of accessibility evaluation of BWF MetaEdit for Windows
6. Detailed report of accessibility evaluation of BWF MetaEdit for macOS
7. Summary report of BWF MetaEdit for Windows
8. Summary report of BWF MetaEdit for macOS

Appendix B: Q&A with Tech for All

Below is a list of questions and answers that provide clarification on reading the detailed and summary reports in Appendix A.

Question: In the detailed report for embARC, only one issue is named for MXF use case for Windows and Mac, and Windows has another listed as "General" use case. Can you clarify whether this indicates all the issues listed for DPX use case are present exclusively in DPX use case only? That may not be entirely accurate if, for example, the first issue listed for Windows application is present on the splash screen (likely before any file format is chosen). It would be good to get TFA's clarification on how to read the use case labels.

> **TFA Answer:** For the embARC apps, most issues present in the DPX use case were also applicable to the MXF use case, specifically the ones denoted as "Global."

Question: The report summaries for embARC and Handbrake mention "high-impact" issues for persons who are blind. Are "high-impact" issues deemed more or less severe than "significant" issues in the context of these summaries?

> **TFA Answer:** We use "high-impact" to denote "issues that make portions of the application inaccessible."

Question: In the detailed reports of issues, TFA has a column labeled "Global," and an issue is marked as global or not global (implied as blank). Can TFA define what makes an issue considered "global"?

> **TFA Answer:** These are the issues that either occur on all screens of the application (common components like the menu bar) or occur with all components of this type used throughout the application. For example, if all grids have the same issue, we would call it out as "Global".

Question: What conclusions or findings did TFA draw on the software architecture and design and development approach for each of the three applications? The reports mentioned these in the methodology, but the results point out only issues experienced on the user's end.

> **TFA Answer:** Since we did not evaluate the source code or development environment for these projects, our conclusions could only be influenced by observing user interface interactions with the assistive technology, which influence remediation recommendations for each specific observed issue.