© Journal of Accessibility and Design for All (JACCES), Volume 15, Issue 2, 2025, ISSN: 2013-7087
DOI: https://doi.org/10.17411/jacces.v15i2.590

Low-code solution development and accessibility

Andrews s., Digital and Technology Specialist, Northumbria University, United Kingdom, ORCID-0009-0009-0285-6455, steven.andrews@northumbria.ac.uk

Received: 2024-11-14 | Accepted: 2025-09-13 | Publication: 2025-11-11

Abstract: In recent years, digital communicators and web developers have recognised the importance of producing web applications and content that is accessible to disabled users. Digital accessibility is measured by how closely an app or page adheres to Web Content Accessibility Guidelines (WCAG). Numerous browser-based accessibility checkers exist that can help identify accessibility issues and WCAG violations. However, their effectiveness with solutions produced by low-to-no code (L2NC) platforms is mainly unknown. This paper is a research project that investigates L2NC solution development and accessibility, focusing on a custom low-code application built within Power Apps. It investigates the challenges disabled users face in accessing digital content and evaluates the effectiveness of accessibility checkers within low-code solutions. To test the low-code application, two accessibility tools (WAVE and Lighthouse) and the product's internal accessibility checker were used to test two versions of a specific solution: a staff directory. One version of the staff directory was created to be accessible, whilst the other incorporated several of the most common WCAG violations identified in the literature. User testing was also carried out on both staff directory applications to gauge user perceptions of them. Descriptive statistics were produced to explain the paper's findings. Both WAVE and Google's Lighthouse were found to be ineffective at detecting WCAG violations within low-code solutions but the internal accessibility checker, whilst not able to identify all common WCAG violations within the staff directories, was the most successful utility. However, the inbuilt accessibility checker's rules do not align well with WCAG violations and have issues detecting colour contrasts. The findings indicate that significant accessibility barriers exist within low-code platforms. The 'ease of use' capabilities and other advantages offered by L2NC utilities are achieved by abstracting out complex paradigms, such as accessible design.

Keywords: accessibility, low-code, no-code, Wave, WCAG, Lighthouse

1. Introduction

Department Y is public body that works in the field of trade and industry. As a public body, Department Y is compelled by law to deliver accessible digital products by complying with Web Content Accessibility Guidelines (WCAG) as set out in The Public Sector Bodies (Websites and Mobile Applications) (No.2) Accessibility Regulations 2018 (Alim, 2021; Ginley, 2020).

Unfortunately, Department Y has not yet achieved full WCAG compliance across all its digital commodities. The public Department Ywebsite is supported by a large, experienced team of content designers and developers, who collectively guarantee that WCAG compliance is achieved. Internal utilities, such as the corporate intranet, have a different ownership model in which a team of solution designers (known as the collaboration team) function as both product owners and platform developers. The collaboration team accept that the intranet is not WCAG compliant when compared to the external website, which can make the product hard to use for impaired users.

Department Y intranet content is primarily static material written by volunteer editors that possess little training in accessible content design. Initially, the collaboration team perceived accessibility as a measure of how consumable digital content is for disabled users via internet browsers (Taylor & Burnett, 2019). Additionally, the Department Y intranet includes low-to-no code (L2NC) solutions designed to address specific business needs that were not constructed with accessibility in mind (Martinez & Pfister, 2023). Because Department Y employs several experienced content designers, the collaboration team opted to research accessibility within L2NC solution development specifically.

1.1. Software engineering and L2NC solutions

Software engineering applies engineering design processes to design, develop, test, maintain, and evaluate computer software. Historically, software engineering has been the domain of mainframes, servers, and desktop computing (Morse, 1986), but more recently encompasses web and mobile applications (Frezza, 2016). Conventional software engineering skills include coding, code optimisation, models, frameworks, knowledge about database systems and awareness of requirements elicitation, such as functional and non-functional requirements (including accessibility).

L2NC solution development disrupts this paradigm as they are produced in low-code platforms that lower the skills ceiling typically required for solution development by abstracting out complex programming concepts (Gomes & Brito, 2022), and this is done via browser-based, graphical editors (Martínez-Lasaca, Díez, Guerra, & de Lara, 2023). L2NC developers are often called citizen developers, and while there is no consistent term academic term, these individuals are typically business users that are familiar with processes or tasks (Olariu, Gogan, & Rennung, 2016). Thus, the benefits offered (to organisations) by L2NC platforms include reductions in application development time, costs and complexity, and more significant business involvement (Alsaadi et al., 2021).

Unfortunately, accessibility is one of the concepts that is commonly abstracted out programming for the benefit of citizen development. In their L2NC benefits and limitations framework, Martinez and Pfister (2023) make no explicit mention of accessibility. Indeed, several categories, namely B1 (agility), B3 (reduced complexity), B4 (reduced maintenance) and L1 (lack of customisation) pointedly exclude all consideration for inclusive solutions. The consequence of disregarding accessibility is twofold. First, L2NC solutions are more likely to exclude disabled users. Second, due to L2NC vendor specificity, no universal or standardised approach currently exists to redress inaccessible L2NC solutions.

1.2. Importance of accessibility in software engineering

The British Computing Society's 2023 annual report states that disabled people comprised 21% of the UK working-age population in 2021 but only 15% of the national workforce (Runciman, 2023), which the disability charity Scope calculates as 14.6 million people. This group represents £274 billion in annual spending power in the UK, and it remains untapped as only 2% of the internet's top 1 million pages offer full accessibility (Perera, 2022).

The impact of inaccessible web applications in banking (Wentz, Pham, and Tressler, 2017), education (Kane, Shulman, Shockley, & Ladner, 2007), and travel (Teixeira, Eusébio, and Teixeira,

2021) is well documented in the literature. There are, however, other software products that successfully solve accessible questions such as blood-glucose monitors (Uslan, Burton, and Clements, 2008), wearable tech (Moon, Baker, and Goughnour, 2019), and gaming consoles (Brown & Anderson, 2020). The advancement in these technologies demonstrate the positive impact that inclusive software has for disabled people in living their lives.

As society moves toward a 'digital-first' perspective, it is ethically appropriate to encourage holistic, inclusive design principles. Inclusive design facilitates the creation of services for all disabled and non-disabled individuals, ensuring that no individual is unfairly barred from accessing a service (Jordan, 2000). Therefore, it is entirely reasonable to expect accessible software development tools and utilities that permit disabled individuals to establish careers within software engineering.

1.3. Research gap, aims, objectives

Accessibility engineers have audited the external Department Y website several times as part of its compliance journey. Methodologically, audits are manual tests that were performed by accessibility engineers (who are disabled) using assistive technologies. Assistive technology describes products that "maintain, increase or improve functional capabilities of individuals with disabilities" (Lahm, 2003, p. 141). Some automated tests, performed via browser-based tools, were also executed. These testing techniques align with Government guidance for accessibility testing (Government Digital Service, 2023).

The application of accessibility audits to L2NC solutions is the juncture at which knowledge gaps in the practice of accessibility testing start to emerge. The accessibility tests and tools endorsed by the UK Government assume that developers possess full access to web-application source code. The recommended tools also only assess rendered/compiled code. As such, it is unclear how useful current accessibility tools are when applied to L2NC platforms or solutions built in them by citizen developers. L2NC platforms offer different granularity of code editability, meaning that fixes suggested by automated accessibility checkers might be difficult or impossible to implement. As such, L2NC solution accessibility was the overall research topic for the collaboration team's efforts to understanding how they could improve the intranet.

Department Y is heavily invested in a proprietary low-code platform. The collaboration team performed this research academically but still addresses Department Y current accessibility issues through the lens of a low-code solution, a staff directory. Table 1 details the overall research aims and objectives. As no sensitive information was requested of the participants, there are no pertinent ethical implications of this work. A motivating factor for this research is therefore that while significant research on accessibility exists, little within it acknowledges L2NC ecosystems, so the literature review explores accessibility in-depth.

Table 1: Relationship *between research topic, aims and objectives*

Research topic	To investigate accessibility configuration in L2NC platforms
Research aim	Investigate accessibility configurations for L2NC solutions
Research objective	01 – Investigate digital accessibility challenges faced by disabled users
Research objective	02 – Conduct testing and user studies to investigate the effectiveness of accessibility checkers within a proprietary low-code platform L2NC solutions

2. Literature Review

2.1. Conceptualising accessibility

Since this research pivots around Microsoft technology, it is appropriate to start with a literature review relating to the accessibility of Microsoft Design bundles within its inclusive design ethos. Inclusive design is a methodology that draws upon the full spectrum of human diversity, and accessibility is a professional discipline aiming to "make an experience open to all" (Microsoft Design, 2024). Rather than adhering to a set vocabulary, Microsoft uses personas encompassing motor, visual, aural and verbal impairments.

Singh, Sibi and Bashir (2023) leverage the social model of disability over a medical definition. This model states that attitudinal, physical and informal barriers cause disability and that product development processes should mitigate them where possible. Nelson and Cook (2022) also subscribe to the social model, observing that disability is "measured and defined in multiple ways."

Neal et al. (2023) perceives web accessibility as a long-standing, all-encompassing design paradigm for producing digital technologies. Like Microsoft's personas, they further define cognitive impairment as an emerging concept in which individuals can face problems digitally if they struggle with complex attention, memory, learning, or language. Nihalani and Robinson (2021) expand on this via their cognitive load theory, postulating that learners have finite cognitive and attentional resources to expend. Lastly, Marcelino et al. (2015) raises the impact of ageing on physical and cognitive capabilities and group impairments analogously to Microsoft.

It should also be noted that a distinction is made in the literature between cognitive impairment theory and cognitive disability. A perceived failing of WCAG guidelines is that there is insufficient coverage for the family of cognitive disabilities, such as autism, dyslexia, and aphasia (Gartland et al., 2022). The success criteria that are included, such as session timeouts and improved error handling fall under AAA standards (Ramkumar, 2025). The W3C, in acknowledgement of WCAG's low level of neuroinclusive guidance has created the cognitive and learning disabilities accessibility task force (COGA), a body that offers guidance for making both content and user agents more accessible to those with cognitive and learning disabilities.

2.2. Education and awareness

Palmer and Palmer (2018) argue that communications practitioners and educators have a role to play in breaking down accessibility barriers (per the social model). They agree with Oswald (2013) in that current digital production techniques are rooted in ableism (discrimination and prejudice against disability). It is only through challenging established practices that cater for "normative" users that a broader culture of awareness and inclusion emerge. Palmer and Palmer feel that existing practices begin with how developers and marketers are taught.

The emphasis on developers traces back to Youngblood's (2012) look at how novice web developers are exposed to the epistemology of accessibility. Youngblood advocates a syllabus that includes WCAG standards, emulating the experiences of disabled people, and dedicated accessibility tools (accessibility checkers and screen-readers). One tool of relevance is WAVE, which shows developers where their code is sub-optimal. Reuschel, McDonnall and Burton (2023) provide examples of the impact of low-quality code noting that 40% of thirty job sites contained code intended for screen readers, which worsened the experience.

Moving beyond web developers, Nelson and Cook (2022) use interviews with advertising students to query the link between digital communicators (as a wider profession) and disability awareness. Their findings reinforce Palmer and Palmer's discourse, as most interviewees were unfamiliar with accessibility concepts and felt that the advertising industry could be more inclusive. Conversely, Schmutz, Sonderegger and Sauer (2016) extol the benefits of fully accessible sites for those who are non-disabled, which is evidenced by shorter task times. Neal et al. (2023) demonstrates a similar outcome, proven through lower cognitive loads when perusing complex privacy statements.

2.3. Guidelines, legal and ethics

Naturally, the appropriate means to challenge established practices is knowledge – and specifically knowledge about accessibility guidelines, how disabled rights are enshrined in law and the ethics that underpin these rights.

Table 2: Summary of WCAG 2.2 structure

WCAG 2.2 Principle	WCAG 2.2 Guideline	Guideline Success Criteria
Perceivable	1.1: Text alternatives	1.1 has 1 criterion
Perceivable	1.2: Time-based media	1.2 has 9 criteria
Perceivable	1.3: Adaptable	1.3 has 6 criteria
Perceivable	1.4: Distinguishable	1.4 has 13 criteria
Operable	2.1: Keyboard accessible	2.1 has 4 criteria
Operable	2.2: Enough time	2.2 has 6 criteria
Operable	2.3: Seizures and physical reactions	2.3 has 3 criteria
Operable	2.4: Navigable	2.4 has 13 criteria
Operable	2.5: Input modalities	2.5 has 8 criteria
Understandable	3.1: Readable	3.1 has 6 criteria
Understandable	3.2: Predictable	3.2 has 6 criteria
Understandable	3.3: Input assistance	3.3 has 9 criteria
Robust	4.1: Compatible	4.1 has 3 criteria

The WCAG framework is maintained by the World Wide Web Consortium (W3C) and consists of four principles (Table 2). The four principles encompass thirteen guidelines, and these guidelines are further broken down into eighty-seven checkpoints known as success criteria. WCAG success criteria offer three levels, A (lowest), AA (medium) and AAA (highest), which are accumulative, meaning that all levels reflect different accessibility aspects. The W3C mandates that A and AA success criteria must be met for a guideline to be considered compliant as some guidelines do not have an AAA equivalent. It should also be noted that A and AA compliance is enshrined in several acts of law (Table 3). Both Alim (2021) and Marcelino et al. (2015) document the specific WCAG versions used in their work. As the current WCAG version is 2.2, the collaboration team decided that it was appropriate to use it in their investigation into L2NC accessibility.

Delving into accessibility legislation, Palmer, and Palmer (2018) explore "places of public consideration" in their review of US (United States) case law. Two legal cases involving Southwest Airlines and Netflix determined that websites are not places of public consideration and, thus, exempt from ADA legislation. Conversely, other cases involving Target and Netflix determined the opposite. Reuschel, McDonnall and Burton (2023) comment on how inaccessible job sites are likely in breach of section 508. Youngblood and Lysaght (2015) introduce the 21st-century communications video accessibility act, which explicitly challenges organisations to improve the accessibility of online videos via captions.

Table 3: UK/US legislation concerning accessibility

Act of law/legislation	Region	Notes
Public Sector Bodies Accessibility Regulations 2018	UK	Compels public bodies sites and applications to be AA compliant
Equality Act 2010	UK	Defines disability as a mental or physical impairment that has a long-term negative impact on daily activities. Also, outlaws discrimination against impairments
Americans with Disabilities Act 1990	USA	Commonly referred to as ADA, this act outlaws discrimination against the enjoyment of goods and services within a "place of public consideration."
Section 508 of the Rehabilitation Act Amendments (1978) 2018 amendments	USA	Commonly referred to as Section 508, this law requires US agencies to be AA compliant
21st Century Communications Video Accessibility Act	USA	Requires online videos to be captioned by broadcasters

Griffith, Wentz, and Lazar (2022) opine that in practice not all WCAG checkpoints are equal, in that some have a more significant impact on the end-user experience than others. They also detect tensions between securing AA compliance vs creating usable sites. Furthermore, Khasawneh et al. (2023) identify a conflict between WCAG standards and section 508 legislation, which they resolve by composing a set of compliance heuristics. This heuristics tool is an evaluative model and accompanying criteria that disposes of jargon.

Lastly, the United Nations has determined, as far as ethics are concerned, that access to the internet is a fundamental human right (Palmer & Palmer, 2018). If inaccessible, the growing digitisation of financial services and utilities negatively impacts the disabled community's ability to contribute fairly to their work.

2.4. Tools, testing methods, common WCAG violations

Inspecting research methods used by researchers in the literature reveals two distinct investigative routes for WCAG compliance: manual testing and automated testing via online accessibility checkers. Manual testing involves specialised accessibility engineers and assistive software. Reuschel, McDonnall and Burton (2023) deploy accessibility engineers to assess job application sites whereas Alim (2021), Vo, Hewitt and He (2023), Singh, Sibi and Bashir (2023), Bounajim, Henderson and Hewitt (2022), Youngblood and Lysaght (2015), Youngblood (2012) and Sonnenberg (2020) resort to online accessibility checkers. All online accessibility checkers encountered are viewable (Appendix A Table 12).

Attitudes to online accessibility checkers vary, however. Singh, Sibi, and Bashir (2023) use a single tool, justifying its selection by wanting to remain consistent with previous research carried out in the tourism industry. Alim (2021) uses three, articulating that multiple tools provide broader and more in-depth coverage of webpages whilst being significantly cheaper than specialist manual testing. Sonnenberg (2020) uses one tool but combines it with additional CSS and HTML code validators. Ismailova and Inal (2022) review six online checkers against forty-one government sites and conclude that using several tools in tandem yields better results than just one.

All online accessibility checkers identified are either (i) browser plugins or (ii) online code checkers. However, a third toolset has emerged in recent years; Google's Lighthouse. Lighthouse offers a suite of analytical packages that audit a website's overall performance, gauging performance, accessibility, SEO, and adherence to best practices (McGill, Bamgboye, Liu, & Kalutharage, 2023)(McGill, Bamgboye, Liu, & Kalutharage, 2023). Lighthouse is invokable from within developer tools in Chrome and Edge but can also be run as a Node.JS module by web-developers. Lighthouse calculates a page's accessibility score as a weighted average of all accessibility audits using third party heuristics developed by Deque (Google, 2019).(Google, 2019). Unlike the other performance audits available in Lighthouse, an assessed page will fail an accessibility audit completely, if a single violation of that specific assessment is detected. Deque's heuristics validate against WCAG versions 2.0, 2.1 and 2.2 although they do not specifically pair off to each specific standard. Despite the relative youth of Lighthouse, it has appeared in recent research that investigates the performance of university websites (Ogbuju, Ayodeji, & Azeez, 2022) .

On WCAG violations encountered, some success criteria checkpoints are broken more frequently than others. However, not all papers name them explicitly. Griffith, Wentz, and Lazar (2022), Alim (2021), and Vo, Hewitt, and He (2023) document their most common violations by WCAG checkpoint reference. Singh, Sibi, and Bashir (2023) and Youngblood and Lysaght (2015) also provide WCAG references that pivot towards different checkpoints due to the visual nature of their investigations. Schmutz, Sonderegger and Sauer (2016) manipulated ten specific WCAG standards that have been documented for completeness. A full breakdown of the 23 WCAG violations identified is presented in Table 4.

What Table 4 does not detail are which WCAG violations are specific to dynamic content and advanced user interfaces that have been developed in JavaScript, HTML5 and similar technologies. The W3C has authored a separate set of web standards known as WAI-ARIA (Web Accessibility Initiative - Accessible Rich Internet Applications) that are primarily aimed at web developers not low-code application authors. Several studies in the literature explore the applicability of WAI-ARIA guidelines (Abu Doush, Alkhateeb, Maghayreh, & Al-Betar, 2013; Thiessen & Russell, 2009) in the production of accessible applications but not within a low-code environment.

Table 4: Common WCAG violations

WCAG criteria violated	WCAG version(s)	WCAG Principle & Checkpoint
1.4.11: Non-text contrast	2.0, 2.1	Perceivable – Distinguishable
2.4.1: Bypass blocks	2.0	Operable – Navigable
4.1: Parsing	2.0	Robust – Compatible
1.4.1: Use of Colour	2.0	Perceivable – Distinguishable
2.1: Keyboard navigable	2.0	Operable – Keyboard accessible
1.4.5: Images of text	2.0	Perceivable – Distinguishable
2.4.6: Headings and labels	2.0, 2.1	Operable – Navigable
2.4.4: Link Purpose (in-context)	2.0, 2.1	Operable – Navigable
1.4.3: Contrast (Minimum)	2.0, 2.1	Perceivable – Distinguishable
1.1.1: Non-text content	2.0	Perceivable – Text Alternatives
1.3.1: Info and relationships	2.0	Perceivable – Adaptable
1.4.3: Contrast minimum	2.0	Perceivable – Distinguishable
1.4.4: Resize text	2.0	Perceivable – Distinguishable
1.4.8: Visual presentation	2.0	Perceivable – Distinguishable
2.4.3: Focus order	2.0	Operable – Navigable
2.4.7: Focus Visible	2.0	Operable – Navigable

WCAG criteria violated	WCAG version(s)	WCAG Principle & Checkpoint
2.4.10: Section headings	2.0	Operable – Navigable
3.2.3: Consistent navigation	2.0	Understandable – Predictable
3.2.4: Consistent identification	2.0	Understandable – Predictable
3.3.1: Error identification	2.0	Understandable – Input assistance
3.3.3: Error suggestions	2.0	Understandable – Input assistance
2.5.3: Labels in name	2.1	Operable – Input Modalities
4.1.2: Name, role label	2.1	Robust – Compatible

2.5. Gaps & relevance

Several journal articles discuss the productivity gains offered by Power Apps but fail to mention accessibility (Boonrit et al., 2024; Diksha & Sandhu, 2021; Fifolt, Baker, Menefee, Kidd, & McCormick, 2024; Khenfer, 2023; Sharma & Gupta, 2021) . Likewise, L2NC benefits and limitations models fail to discuss accessibility concerns (Martinez & Pfister, 2023). As such, this research project conducted in this paper aims to fill the gap by investigating whether the tools and techniques discussed convey any utility to L2NC solution development.

Lastly, the proprietary low-code platform offers its own in-IDE app-checker that makes rudimentary accessibility checks, but it is extremely limited as it only caters for ten specific accessibility issues (Microsoft, 2022) . None of the low-code platform's accessibility categories align directly with WCAG 2.2 checkpoints.

3. Methodology and implementation

3.1. Research methodology

Saunders research onion was consulted to ensure appropriate conventions were followed as this research is the first academic endeavour attempted by the collaboration team (Saunders et al., 2019).

A pragmatist *philosophy* was considered appropriate for the work as it encourages using the best tools available to explore the relationship between accessibility and L2NC platforms. Positivism was considered an unhelpful approach in this context as its binary depiction of knowledge (as true or false) matches poorly against the fluidic concept of accessibility (per the social model). Conversely, interpretivism leans too much into social and cultural factors to be applicable. This study is rooted in the relationship between technology and accessibility and does not focus purely on how disability is defined. For the *research approach* abductive reasoning is preferred as the study only seeks plausible answers to research objectives (Shani, Coghlan, & Alexander, 2019) rather than "cause and effect" (deductive) explanations or data-adaptable theorems (inductive).

This study utilises two *research strategies*, action, and exploratory research. Action research places participants as the beneficiaries of their study (Whitehead, Taket, & Smith, 2003) whereas exploratory research adds definition to academic areas that lack it (Haile, 2023)(Haile, 2023). If citizen developers can be trained in building them and if they are accessible, L2NC solutions are beneficial to all users and not just those who are disabled. Likewise, this study may catalyse further investigation into the phenomena of low-code accessibility. The topic's lack of academic framework rules out case-study and archival approaches as no material exists to draw upon. The *time horizon* is cross-sectional because the abductive nature of this study removes chronology as a factor.

Research choices and *data collection/analysis* are the final components of the research methodology to discuss. This study utilises a mixed-method concurrent nested strategy that collects quantitative performance data alongside qualitative data concentrating on the "look and feel" of L2NC solutions. Qualitative research is often confirmatory of statistical data (Olds, Moskal, & Miller, 2005), hence the choice of concurrency over a sequential design. The study population is randomised amongst the staff at Department Y but does not consist of 100% disabled staff members. The organisation maintains a staff network, called "Disability and You" that does not have enough members within its ranks to provide a sizable population. Lastly, collected data is analysed quantitatively to produce descriptive statistics and qualitative data is thematically analysed (Braun & Clarke, 2006) . These analytical methods are selected due to the small population size. A summary of the study's research design is contained in Table 5.

Table 5: Summary of research design

Saunders Characteristic	Design
Research philosophy	Pragmatism
Research approach	Abductive
Research strategy	Action Research/exploratory research
Choices	Mixed-method concurrent nested
Time horizon	Cross-Sectional
Data collection/analysis	Randomised Dep. Y staff, descriptive & thematic

3.2. Staff directory L2NC application

The utility that the collaboration team settled on to explore L2NC accessibility was a staff directory search application; a common tool within enterprise intranets (Pedley, 2003; Warunek, Gruver, Bartko, & Blair, 2024). Two versions of the staff directory were constructed: one accessible and the other inaccessible. The inaccessible staff directory included the nine most common WCAG violations identified during the literature review. Both versions of the staff directory were then assessed against two online accessibility checkers (identified in the literature review) to gauge their effectiveness with the proprietary L2NC solutions. The two tools selected were WAVE and Lighthouse.

Table 6: Final list of staff directory requirements

Staff directory requirement	MosCoW
Ensure staff personal data is not exposed	Must
Show a list of staff and their contact details	Must
Staff can records staff directory record to view further information	Must
Allow Department Ystaff to filter staff by department	Must
Ensure application is WCAG 2.2 compliant	Must
Provide an edit facility for staff to update their records	Should
Provide a report of all staff and their department/location	Should

As the staff directory application fulfilled an actual need for the Department Y, the collaboration team planned its development under agile project management principles. Staff directory requirements were elicited via user stories to understand which features were deemed valuable by staff. Department Y staff were identified through internal communication channels (Viva Engage, internal newsletter and Intranet support inbox) and interviewed for their perspectives. The requirements were then sorted via MosCoW according to what the collaboration team felt would be achieved quickly and collated into a requirements database as per Figure 2 (Cadle, 2014). Finally, Table 6 lists the features deemed essential for an MVP, which included search and retrieval functions as well as WCAG 2.2 compliance. More advanced components, such as self-edits and departmental reports were deemed optional.

3.3. Data collection & survey instruments

Automated testing on both staff directories was performed using WAVE (W3C, 2024), the most frequently utilised online accessibility checker identified during the literature review (Ismailova & Inal, 2022). Some researchers advocate using two checkers for increased WCAG violation coverage (Alim, 2021; Vo, Hewitt, & He, 2023).This study opted against doing so because of the exploratory nature of the research. Furthermore, the low-code platform accessibility checker incorporates some accessibility checks. Thus, using a singular online checker is deemed sufficient as it allows a more direct comparison as to the efficiency of both utilities.

As well as assessing the two staff directory applications against online accessibility checkers, usability tests with Department Y staff were conducted using survey instruments containing both qualitative and quantitative questions. Lastly as the sample population consists entirely of Department Y staff members that have volunteered their time, they are considered as a self-selecting group that offer nothing inferential for the wider organisation. All participants provided informed consent prior to taking part in the study, and no sensitive or personally identifiable information was collected

3.4. Staff directories build and test process

The staff directory application is architected through a micro-service approach, using specific services for precise functions as per Table 7. Power Apps, as the staff directory user interface, is the only component to be assessed for accessibility. Power Bi charts and reports are not tested for accessibility as they are outside the scope of this research.

Table 7: Staff Directory technology components

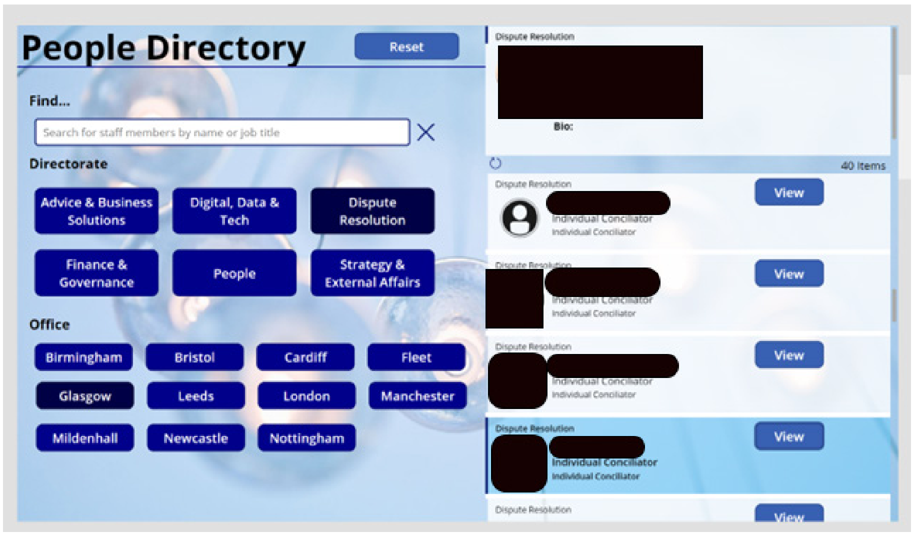
M365 Service	Usage
SharePoint Online Lists	Data storage for staff application data
Power Automate	Scheduled workflow to synchronise Active Directory changes with SharePoint
Power Apps	L2NC tool that staff directory application interface is built in
Power BI	Data visualisation tool for reports pertaining to staff office and/or directorates

To begin, a copy of Active Directory (corporate directory service) was taken and converted into a SharePoint list, which simplified the removal of personal information not fit for public consumption. Secondly a Power Automate workflow was created to keep the SharePoint list coordinated with updates performed in active directory. Both the accessible and inaccessible versions of the staff directory application utilise the SharePoint list, so a singular data source was sufficient. Lastly a Power BI dashboard was created that ingested the SharePoint list data and rendered a dynamic table. Although the Power BI dashboard is viewable by staff, it was not gauged for accessibility as data visualisations are not produced by intranet content editors.

Within the low-code, the staff directory is a single screen canvas application. A canvas application is a blank screen (canvas), in which developers add content from a series of drag and drop components including labels, buttons, images and more (Rajaram et al., 2022a). The alternative model is model-driven, which generates a multi-screen application from a data-source. The collaboration team settled on a canvas application as it affords greater customisation and design options for simple, mono-screen entities. All solutions within the Department Y intranet are canvas applications.

The accessible staff directory (Figure 1) was created first with initial configuration involving data source initialisation, and the addition of background visuals and two galleries in which user data is rendered. The top gallery was further set-up to render a single user profile when selected in the lower one (the users gallery). Further components (buttons, free text search, icons) were added and configured to filter and update the users gallery. Directorate and office filters were programmed against button OnSelect events. Lastly, the search input field was configured as a dynamic filter against the user's gallery to return results that match a name or job title search. For usability purposes, filter buttons and user gallery entries were configured to visually demonstrate when they had been interacted with. Upon completion, this build reported zero accessibility issues within the low-code platform IDE and is referred to as Version 1.

Figure 1: Accessible staff directory (version 1)

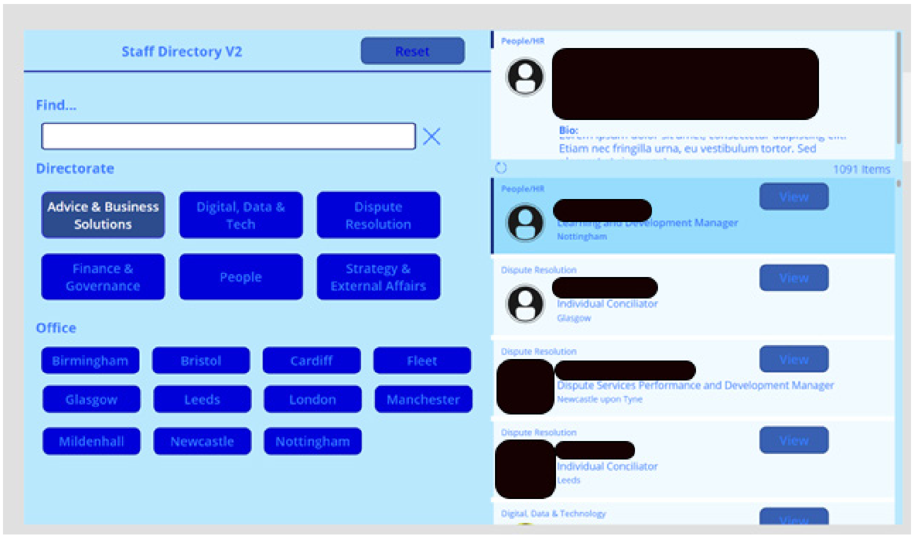


The build for the inaccessible staff directory began by cloning the accessible version, which provided a functional application build to work on. Specific changes were made to incorporate the accessibility issues identified during by the literature review (Table 4).

Adobe's colour contrast analyser was used to create a low contrast colour scheme that violates WCAG guidelines. The background colour was determined by a sampling a screenshot of the accessible version and is #b9e8fd with text set to #3D7BFF for a contrast ratio of 2.93:1. A second colour scheme for buttons was generated in which the background colour is #0000D9 and text colour of #367CFF and a contrast of 2.74:1. These colour schemes were applied visually to the application's background, button, labels, icons and text.

Most text was resized to font-size 16 and applied consistently to all labels. Text labels were also stripped of any properties that gave them specific meanings for screen readers. Non-visual changes included the removal of tooltips, accessibility labels from user galleries and icons, and OnFocus settings that indicate which user interface (UI) element is currently selected. The completed inaccessible staff directory is depicted in Figure 2 and compiled in the low-code platform IDE reporting 28 accessibility errors and is referred to as Version 2.

Figure 2: Inaccessible staff directory (version 2)



For usability testing, both versions of the staff directory were embedded in unique SharePoint Online pages for easier distribution. Aside from the M365 navigation, no other content was included within the SharePoint Online pages. These page links were then inserted into user testing surveys and distributed to testers. The surveys contained both qualitative and quantitative queries that that asked testers to perform three searches, namely locating colleagues by name, job title, directorate, or region. The population of testers was a self-selecting group that volunteered responses to a request for testers through organisational communications channels.

For automated testing, both versions of the staff directory were assessed outside of SharePoint Online pages. Each canvas application provides a direct URL that will load only the compiled app into browser, without any additional M365 features. WAVE was invoked by inserting each of the staff directory canvas app links into the online accessibility checker by W3C (2024a). Following WAVE testing, browsers were restarted and the staff directories reloaded. Both staff directories were assessed using Lighthouse, which was initialised from the browser's development tools.

Both user testing (Department Y staff) and automated testing (collaboration team) were performed on organisational laptops, which were Microsoft Surface Pros loaded with Windows 11 22H2 operating system and Chrome build 114.

4. Results & Analysis

The aim of the study is to satisfy the two research objectives as outlined in Table 1. Objective one aspires to investigate and understand the challenges faced by disabled users in their digital lives. This knowledge contextualises what is learnt via objective two, which is an investigation of the effectivity of accessibility tools and checkers when applied to the proprietary low-code platform. The full results of automated testing on both staff directories are presented in Table 8. Secondly, user testing produced a variety of results, which are presented in Appendix B Tables 13 - 24. These results are discussed in detail in this chapter.

4.1. Online accessibility checkers are ineffective with proprietary low-code platform solutions

All three tools (WAVE accessibility checker, Lighthouse, and the in-IDE low-code platform app checker) are inefficient at accurately detecting WCAG 2.2 violations. Staff directory version 2 contained eight of the nine common WCAG violations identified in the literature as detailed in Table 8. An accessibility violation for 1.4.4 is not examinable as label/button text scales appropriately when tested against OS settings, browser font-size settings and webpage settings.

Table 8: WCAG 2.2 violations implemented in staff directory version 2

WCAG criteria violated	Implemented in staff directory version 2	Detected in low-code platform IDE	Detected in Wave	Detected in Lighthouse
1.1.1: Non-text contrast	Setting Accessible Label property on both galleries to null	Yes	No	No
1.3.1: Info & relationships	Removed hint text and default text from search box	No	No	No
1.4.3: Contrast minimum	Set app background colour to #b9e8fd and text to #3D7BFF for a contrast of 2.93:1. Also set button background colour to #0000D9 and text to #367CFF for a contrast of 2.74:1	No	No	No
1.4.4: Resize text	N/A – Text resizes appropriately when staff directory app is zoom is used	N/A	N/a	N/a
1.4.8: Visual presentation	Removed default text from search engine	No	No	No
2.4.3: Focus order	Added random integers to the Tabindex property of user galleries and form controls	Yes	No	No
2.4.6: Headings & labels	The Text.Role properties on labels are set to TextRole.Default. This control can be set to H1-H4 for screen-readers	No	No	No
2.4.7: Focus visible	Setting the FocusedBorderThickness property to 0	Yes	No	No
2.4.10: Section headings	Set all section headers to font size 16 in the app. Gallery fonts were also set to 16. Text role properties not used.	No	No	No

The low-code platform app checker reports on accessibility violations in three categories; errors (E), warnings (W) and tips (T). Upon execution, version 2's app checker reported 24 accessibility errors and four accessibility tips. The 24 errors detected consist of 22 instances of "Focus isn't showing" and two "Missing accessibility labels." All four accessibility tips are 'Check the error of the screen items' notifications. Demonstrably, the low-code platform's app checker only detects accessibility violations that are resolved using specific parameters on form controls. This is best demonstrated via the IDE's inability to detect the inappropriate colour contrasts configured within staff directory version 2. The app checker for staff directory version 1 reported no accessibility issues.

Both versions of the staff directory were then tested via the WAVE facility, which checks the compliance of on-screen content against WCAG 2.2. Red icons indicate violations that need resolution whereas green icons hint at areas where improvements can be made. WAVE's reports group findings via errors (E), contrast errors (CE), alerts (A), features (F), structural features (S) and accessible rich intranet applications (ARIA). The WAVE report for staff directory version 1 is Figure 3 and that of staff directory version is Figure 4. Both applications returned identical reports, with one alert, two features, four structural elements and 37 ARIA highlighted. Tellingly, none of these report items apply to the actual staff directory but the ever-present M365 navigation features.

Lastly, in fresh browser sessions both staff directories were assessed with Lighthouse with the following settings: -

- **Mode**: Navigation
- **Device**: Desktop
- **Categories**: Accessibility

Lighthouse returns results in three categories: additional items to check manually (M), passed audits (P) and audits deemed not applicable (N/A). Lighthouse returned a 100% score for both staff directories (Figure 5) highlighting 10 manual checks, 22 passed audits and 35 items deemed not applicable. The full breakdown of Lighthouse's report is presented in Appendix B Table 13.

Figure 3: WAVE report on accessible staff directory version 1

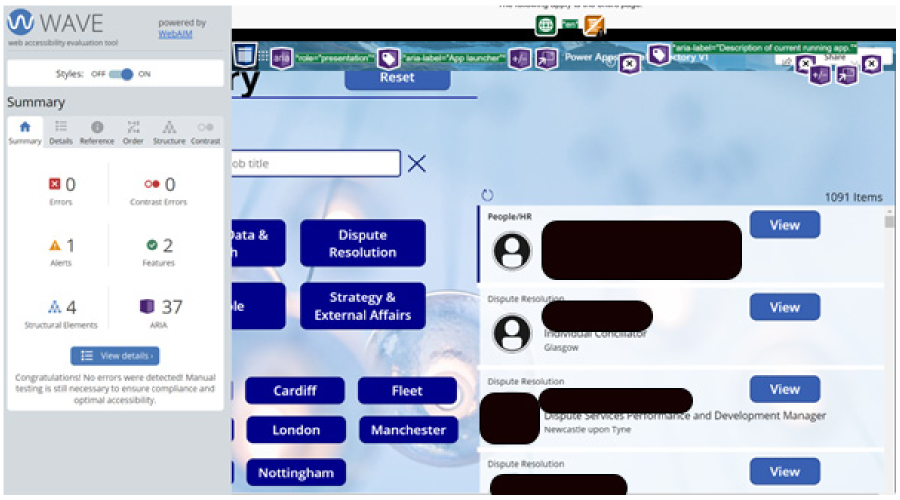


Figure 4: WAVE report on inaccessible staff directory version 2

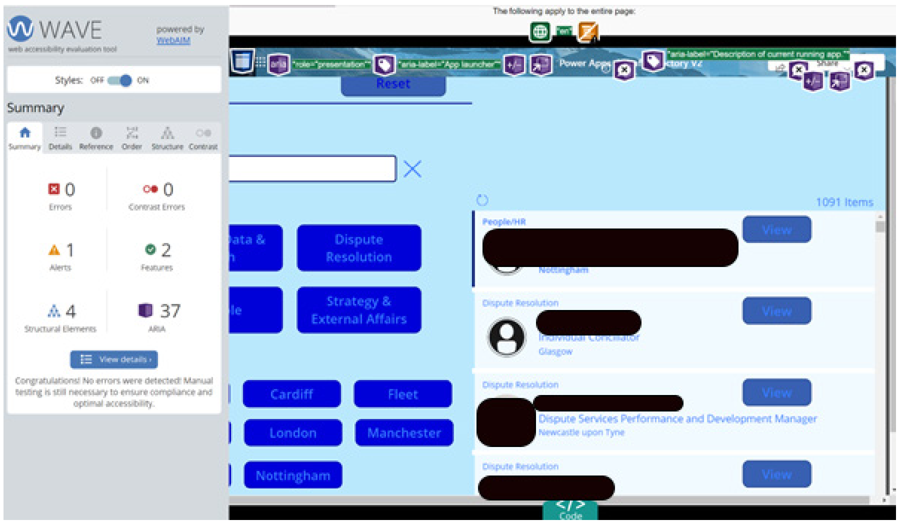
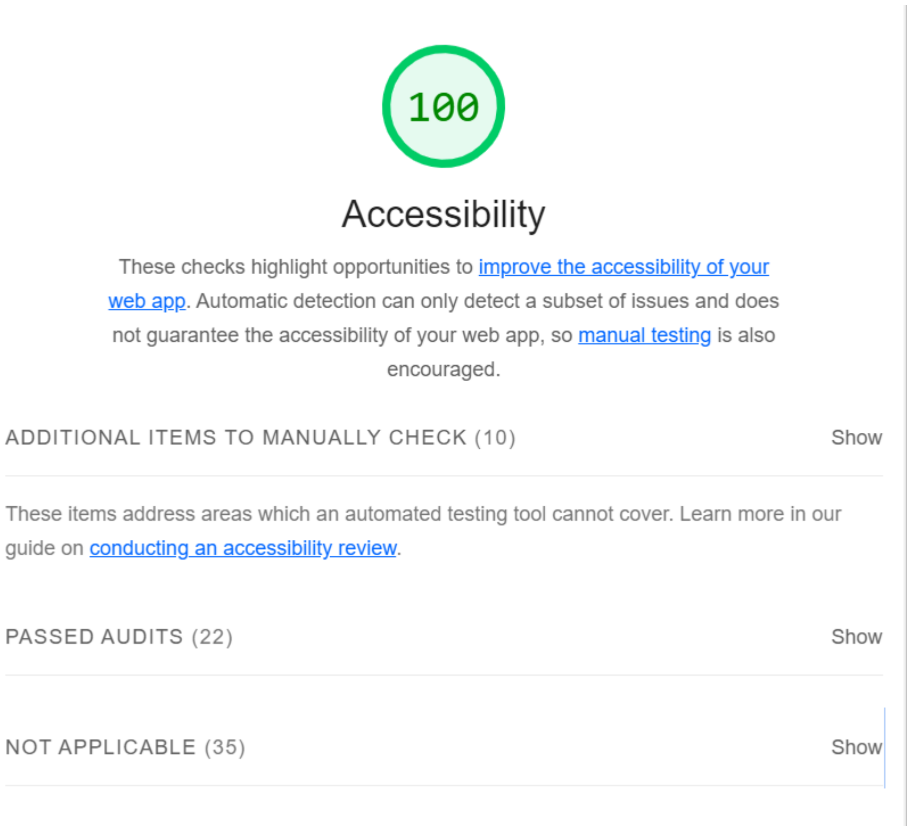

WAVE
web accessibility evaluation tool
powered by WebAIM
Styles: OFF ON
Summary
Summary Details Reference Order Structure Contrast
0 Errors
0 Contrast Errors
1 Alerts
2 Features
4 Structural Elements
37 ARIA
View details
Congratulations! No errors were detected! Manual testing is still necessary to ensure compliance and optimal accessibility.
The following apply to the entire page:
Reset
Dispute Resolution
Strategy & External Affairs
Cardiff
Fleet
London
Manchester
Nottingham
1091 Items
People/HR
[redacted]
Nottingham
View
Dispute Resolution
[redacted]
Individual Conciliator
Glasgow
Dispute Services Performance and Development Manager
Newcastle upon Tyne
Code

Figure 5: Lighthouse accessibility audit results for both staff directories



100

Accessibility

These checks highlight opportunities to improve the accessibility of your web app. Automatic detection can only detect a subset of issues and does not guarantee the accessibility of your web app, so manual testing is also encouraged.

ADDITIONAL ITEMS TO MANUALLY CHECK (10) Show

These items address areas which an automated testing tool cannot cover. Learn more in our guide on conducting an accessibility review.

PASSED AUDITS (22) Show

NOT APPLICABLE (35) Show

A full summary of accessibility violations detected by all automated checkers is presented in Tables 9 through 11.

Table 9: Accessibility violations detected by Wave

Staff Directory	Errors	Contrast Errors	Alerts	Features	Structural Elements	ARIA
Version 1	0	0	1	2	4	37
Version 2	0	0	1	2	4	37

Table 10: Accessibility violations detected by the IDE

Staff Directory	Errors	Warning	Tips
Version 1	0	0	0
Version 2	24	0	4

Table 11: Accessibility violations detected by Lighthouse

Staff Directory	Manual	Passed Audits	Not Applicable
Version 1	10	22	35
Version 2	4	22	35

4.2. User testing on staff directories

A total of thirty-four users were sourced from within Department Y and divided into two groups of seventeen thus ensuring that each staff directory was tested equally. Figure 8 reveals that the search facility in staff directory version two was marginally harder to use successfully. However, this trend does not continue with tasks two and three. Testers for both tasks skew more to the lower scale of the difficulty ratings in version 2 of the staff directory.

Figure 6: Testing results for Q1

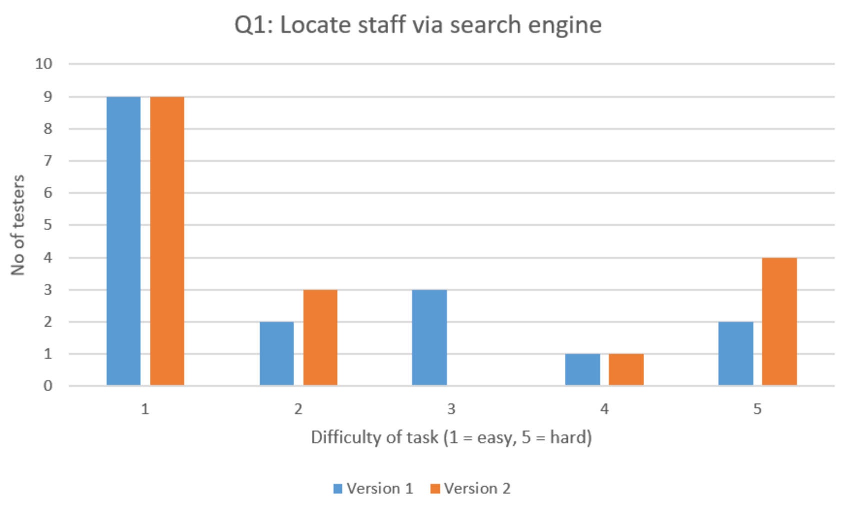


Figure 7: Testing results for Q2

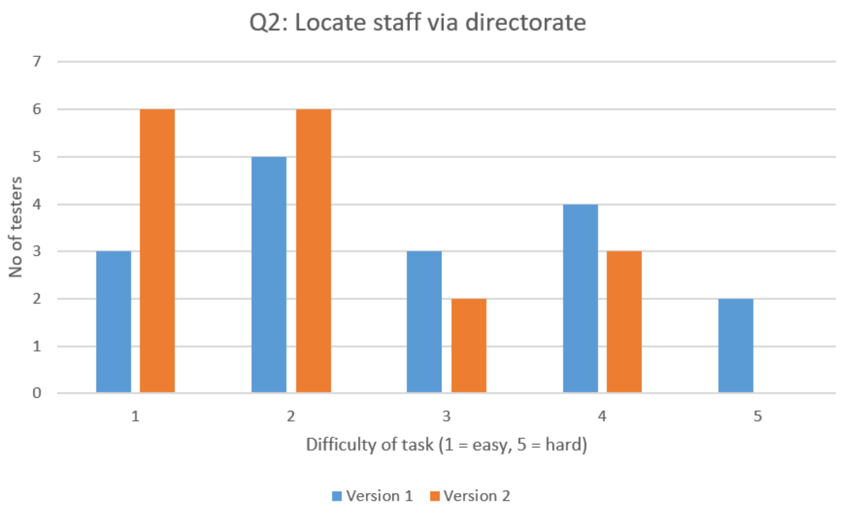
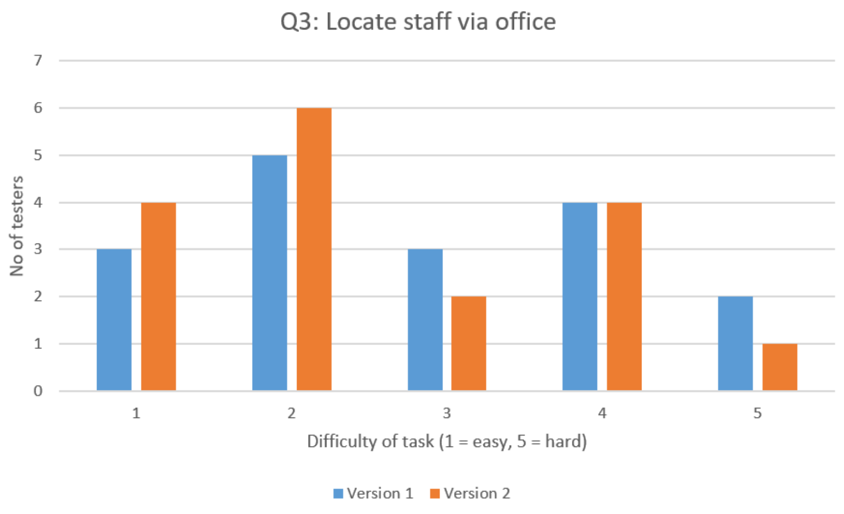


Figure 8: Testing results for Q3



5. Critical Evaluation of Results

5.1. Research objective 1: Investigate digital accessibility challenges

The line of enquiry for this research objective is underpinned by a principle that Griffith, Wentz and Lazar (2022) establish in their efforts to quantify the cost of accessibility barriers for the blind. They draw attention to what they perceive as the tension between meeting WCAG AA compliance versus genuine efforts to make websites accessible and inclusive. The same authors also state that not all WCAG checkpoints are equal in the levels of discomfort that their violations cause disabled users.

This perspective is validated by trends observed in the literature review in which 23 specific WCAG categories are identified (Table 4) as being more prevalent than the remaining 64 (Table 2). This effectively means that less than one third (26.4%) of WCAG 2.2 checkpoints are more disruptive to the digital experience of disabled users than the remaining 73.6%. Lastly, of these 23 checkpoint violations, most of them fall into WCAG perceivable or operable principles, meaning that visual and aural impediments are more common than neurocognitive or mobility issues.

Interestingly, one standard has been identified as having a simple resolution; standard 1.3.1, which is answered by simply using the HTML alt tag for the benefit of screen readers (N. E. Youngblood & Lysaght, 2015). Whilst, in a vacuum, this fix is simple, it represents the sort of intrinsic knowledge that is gained “on the job” by digital communicators, including developers and L2NC solution builders

Intrinsic knowledge then, speaks to the awareness and education of digital communicators and what they should be taught, inclusive of legal obligations and accessibility regulations. It is a legal requirement for public bodies to make their digital platforms and sites accessible (Nisbet, 2020). As intranet platforms are considered part of the modern digital communications realm (Joglekar, Purdy, Brock, Tandon, & Dong, 2022) it is reasonable to assume that Department Y L2NC solution developers should learn about legislation and accessibility alongside how to accommodate them, which leads into research objective two.

5.2. Research objective 2: Effectiveness of accessibility configurations with the proprietary low-code platform

In his exploration of video game accessibility, Anderson conceptualises the ground floor approach to digital accessibility, a metaphor grounded in architectural practices for accessible buildings (2024). Just as a building needs minimal features to accommodate wheelchair users, so digital products need to be fully inclusive. The literature review raised similar thoughts as one specific paper investigated how non-disabled individuals also gain from inclusive design principles when applied to websites (Schmutz, Sonderegger, & Sauer, 2016). Currently, a “ground floor” approach is hard to achieve with L2NC applications constructed in the proprietary low-code platform, due in part to the inefficiency of existing tools.

As presented in section four, both WAVE and Lighthouse are unable to identify any accessibility issues within the staff directory version 2. The most severe item reported in WAVE is an alert over the lack of headings within the M365 application launcher, a shared facility across all M365 applications that has nothing to with the staff directory solutions. Lighthouse returns a score of 100% (Figure 5) including a successful audit for background and foreground colour ratio, which is patently untrue for staff directory version 2. Lighthouse, at least, does encourage manual accessibility checks and provides specific examples what to check.

Furthermore, the accessibility rules built into the low-code platform's IDE only fire in specific scenarios – when explicit properties of the canvas app's drag and drop controls are set inappropriately. This then leaves Department Y L2NC solution developers with weak and inconsistent tooling to confidently build inclusive applications for the intranet. This fact is offset by the results of the usability testing, which highlighted minimal differences in task difficulty, especially with questions two and three, the button-orientated tasks. It would be foolish however to generalise anything from these findings, as the Department Y community has been clamouring for a staff directory for some time, meaning that the testers mustered more enthusiasm than they would have for an unfamiliar L2NC solution that was not in demand.

5.3. Applicability of findings

The value generation of this study's output for Department Y is something of a double-edged sword. The catalyst for the project was an acknowledgement that internal digital products, including the intranet and its embedded L2NC solutions are inaccessible. The collaboration team that monitors and develops the collaboration platforms have no experience or training with resolving accessibility issues. This juncture is where the study outcomes are positive, as the collaboration team now possesses both a very defined problem and an understanding of the limitations of existing automated tools.

This research clarifies both accessibility guidelines and the legal obligations inherent in attempting to resolve them. This knowledge ensures that the current intranet violations can be understood and addressed sufficiently. Conversely, however, are the difficulties in cleanly developing accessible solutions within the low-code platform, which may stunt the growth and evolution of the intranet platform. The existing L2NC solutions within the Department Y intranet will need refactoring to ensure that a proper "ground floor" approach is achieved.

6. Further Work

6.1. Research limitations

As with any piece of academic enquiry, this study's conclusions are tempered by limitations in design and methodology. To discuss where this study could be further developed, the limitations of the research methodology must be explored. Once these constraints have been stated and understood, future possibilities can be appropriately contextualised.

To begin this research, user testing was performed on a simple, uncomplicated low-code canvas utility. Although the product's App Checker has clear limitations regarding detecting inaccessible configurations, its existing rules could have been examined further, especially as Microsoft's literature provides some detail on how to cater for visual and audio materials (Microsoft, 2022)(Microsoft, 2022). This could be best explored through the development of more ambitious L2NC application such as a learning management system (Rajaram et al., 2022b), expense submission systems (Monteiro, Abrantes, & Ratinho, 2024) or activity trackers (Fifolt et al., 2024) . A more ambitious L2NC that contained more violations from the WCAG 2.2 "understandable" and "robust" principles could also yield a more comprehensive understanding of the limits of the low-code platform IDE.

Secondly, none of the thirty-four members in the testing group/population were disabled themselves. Returning to Anderson's work in the gaming industry, he constantly acknowledges that "nobody knows the practical implications of disability as well as people with disabilities" (Anderson, 2023). His view, aside from being ethically appropriate, is portable to all aspects of digital accessibility, including L2NC solution development. The potential involvement of disabled users could be extended to including professional accessibility engineers. Involving either community could radically alter the data generated in L2NC accessibility testing and thus paint an entirely different picture of L2NC solution accessibility.

Next, the ratings used in the user testing survey were also subjective and may not be enough to gauge complex concepts like aesthetics or inclusivity in L2NC applications. Using a modified methodology that combines subjective performance metrics with objective measures could provide more insight. For example, Neal's look at cognitive impairment demonstrates the advantage of using objective measurements in an investigation via their use of reading scales, such as the common European framework of reference to assess readability and linguistic complexity. This framework provides a common metric by which linguistic complexity can be universally framed (Neal et al., 2023). A second study from the literature review uses an objective measurement scale devised by NASA for investigating low colour palettes for individuals with low visibility (Hewitt & He, 2022).

Lastly, none of the studies within the literature discussed the most recent WCAG guidelines, which is guideline 2.2. Although some articles are circulating that mention them (Dias, Carvalho, Rocha, & Barroso, 2022; Engeset, Pfuhl, Orten, Hendrikx, & Hetland, 2022; Filipe, Pires, & Gouveia, 2023), they only refer to the draft guidelines. Eleven new accessibility criterions have been added to WCAG 2.2 (from 2.1) that have yet to examined academically, which means that they could potentially disrupt the perceived order of most disruptive accessibility violations.

6.2. Broad view for further work (industry)

Every year commercial research firms, such as Gartner or Forrester, produce reports that discuss and rank software products of various form and function. Gartner, for instance, publishes a magic quadrant, a Boston matrix diagram that groups products as leaders, challengers, niche players or visionaries. The Gartner report for 2023 discussing L2NC applications ranked this proprietary low-code platform alongside sixteen other products.

An additional sixteen low-code development platforms exist that may have the same accessibility restrictions as this proprietary low-code platform. It would make sense for an industry-wide framework or standard to evolve so that the needs of disabled users do not play a secondary role to proprietary approaches to solving accessibility issues. WCAG 2.2 is already vendor and platform agnostic, thus meaning that no product would gain or lose more than their rivals. Once a standard is agreed upon, the benefits of such a standardisation are plentiful but exhibited best by the productivity gains made in the UML community (Chaudron, Heijstek, & Nugroho, 2012; Chonoles, 2017).

6.3. Narrow view for further work (UK civil service)

Many of the limitations (of L2NC solutions) that have already been discussed open further ways in which Department Y could benefit from extending this research. A different profile of the proprietary low-code platform solution would give the collaboration team more opportunity to explore and investigate L2NC accessibility, establish a "ground floor" approach and successfully comply with legal requirements.

Secondly, the development of a training syllabus for L2NC solution developers and volunteer editors as well as the possible provision of free accessibility tools would help. Youngblood's exploration of a training syllabus for web developers is over a decade old but still holds some validity today (S. A. Youngblood, 2012). A training syllabus that accepts, acknowledges, and guides both intranet editors and L2NC solution developers on the disconnect between online accessibility checkers and low-code can only be a positive thing. A syllabus would need to address the confusion produced by trying to combine (and meet) the ideologies of WCAG compliance and anti-discriminatory legislation. One study has already produced a heuristic, evaluative model for software developers (Khasawneh, Gallagher, Jacobson, & Riley, 2023), that whilst too complex for L2NC solution developers, could be simplified for them.

Lastly, the potential of cross-government collaboration is very possible. Multi-department co-operation is common within the UK government for research, statistics, policy and so on, but not necessarily for information technology. One group that does exist is the Government Intranet Network (GIN), a collection of civil servants (of various trades) that meet monthly to exchange information on various intranets. Accessibility is a common topic of discussion that is yet to yield any productive solutions or modes of thinking. This research could catalyse further work within government circles.

7. Acknowledgements and Conflict of Interest

The author acknowledges the guidance and support of Dr Nikki Phair for her insights and input into the article. The author is also grateful to Ian Andrews and Ellie Willis for reviewing the final draft and to the two peer reviewers for their helpful feedback. The author declares no conflict of interests with regards to the research, authorship and/or publication of this article. The author received no funding or grants for the composition of this research.

8. Bibliography

Abu Doush, I., Alkhateeb, F., Maghayreh, E. Al, & Al-Betar, M. A. (2013). The design of RIA accessibility evaluation tool. Advances in Engineering Software, 57, 1–7. https://doi.org/10.1016/j.advengsoft.2012.11.004.

Alim, S. (2021). Web Accessibility of the Top Research-Intensive Universities in the UK. SAGE Open, 11(4), 21582440211056616. https://doi.org/10.1177/21582440211056614.

Alsaadi, H. A., Radain, D. T., Alzahrani, M. M., Alshammari, W. F., Alahmafi, D., & Fakeih, B. (2021). Factors that affect the utilization of low-code development platforms: survey study. Romanian Journal of Information Technology & Automatic Control/Revista Română de Informatică Și Automatică, 31(3). https://doi.org/10.33436/v31i3y2021.

Anderson, S. L. (2023). Video Game Accessibility Defined Through Advocacy: How the Websites AbleGamers.org and CanIPlayThat.com Use the Word Accessibility. Games and Culture, 15554120231170156. https://doi.org/10.1177/15554120231170156.

Anderson, S. L. (2024). The Ground Floor Approach to Video Game Accessibility: Identifying Design Features Prioritized by Accessibility Reviews. Games and Culture, 15554120231222580. https://doi.org/10.1177/15554120231222580.

Boonrit, N., Klaidokchan, N., Niyomdecha, A., Noppamas, J., Suknuntha, K., Prasertsan, P., … Ruanglertboon, W. (2024). Development and Evaluation of a Prototype Mobile

Application for Intravenous Drug Dose Calculation in Overweight and Obese Thai Children: Precision Dosing in Practice. Hospital Pharmacy, 00185787241229141. https://doi.org/10.1177/00185787241229141.

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. Qualitative Research in Psychology, 3(2), 77–101. https://doi.org/10.1191/1478088706qp063oa.

Brown, M., & Anderson, S. L. (2020). Designing for Disability: Evaluating the State of Accessibility Design in Video Games. Games and Culture, 16(6), 702–718. https://doi.org/10.1177/1555412020971500.

Cadle, J. (2014). Developing information systems: practical guidance for IT professionals. BCS, The Chartered Institute for IT.

Chaudron, M., Heijstek, W., & Nugroho, A. (2012). How effective is UML modeling ? Software & Systems Modeling, 11(4), 571–580. https://doi.org/10.1007/s10270-012-0278-4.

Chonoles, M. J. (2017). OCUP 2 Certification Guide: Preparing for the OMG Certified UML 2.5 Professional 2 Foundation Exam. San Francisco: Morgan Kaufmann.

Dias, J., Carvalho, D., Rocha, T., & Barroso, J. (2022). Automated Evaluation Tools for Web and Mobile Accessibility: proposal of a new adaptive interface tool. Procedia Computer Science, 204, 297–304. https://doi.org/10.1016/j.procs.2022.08.036.

Diksha, & Sandhu, J. K. (2021). Robotic Process Automation for Prioritizing the Refund in Online Travel Agency. 2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), 1006–1011. https://doi.org/10.1109/ICACITE51222.2021.9404718.

Engeset, R. V., Pfuhl, G., Orten, C., Hendrikx, J., & Hetland, A. (2022). Colours and maps for communicating natural hazards to users with and without colour vision deficiency. International Journal of Disaster Risk Reduction, 76, 103034. https://doi.org/10.1016/j.ijdrr.2022.103034.

Fifolt, M., Baker, N., Menefee, R. W., Kidd, E., & McCormick, L. C. (2024). Addressing infection prevention and control in Alabama through the long-term care strike team. Journal of Infection Prevention, 17571774241239782. https://doi.org/10.1177/17571774241239782.

Filipe, F., Pires, I. M., & Gouveia, A. J. (2023). Why Web Accessibility Is Important for Your Institution. Procedia Computer Science, 219, 20–27. https://doi.org/10.1016/j.procs.2023.01.259.

Frezza, S. T. (2016). Issues in student valuing of software engineering best practices. 2016 IEEE Frontiers in Education Conference (FIE), 1–4. https://doi.org/10.1109/FIE.2016.7757556.

Gartland, S., Flynn, P., Carneiro, M. A., Holloway, G., Fialho, J. de S., Cullen, J., … Cullen, C. (2022). The State of Web Accessibility for People with Cognitive Disabilities: A Rapid Evidence Assessment. Behavioral Sciences, 12(2). https://doi.org/10.3390/bs12020026.

Ginley, B. (2020). Working remotely if you are visually impaired. British Journal of Visual Impairment, 0264619620925702. https://doi.org/10.1177/0264619620925702.

Gomes, P. M., & Brito, M. A. (2022). Low-Code Development Platforms: A Descriptive Study. 2022 17th Iberian Conference on Information Systems and Technologies (CISTI), 1–4. https://doi.org/10.23919/CISTI54924.2022.9820354.

Google. (2019, September 9). Lighthouse accessibility scoring | Chrome for Developers. Retrieved 21 October 2024, from https://developer.chrome.com/docs/lighthouse/accessibility/scoring.

Government Digital Service. (2023). Testing for accessibility - Service Manual - GOV.UK. Retrieved 30 May 2024, from GOV.UK website: https://www.gov.uk/service-manual/helping-people-to-use-your-service/testing-for-accessibility.

Griffith, M., Wentz, B., & Lazar, J. (2022). Quantifying the Cost of Web Accessibility Barriers for Blind Users. Interacting with Computers, 34(6), 137–149. https://doi.org/10.1093/iwc/iwad004.

Haile, Z. T. (2023). Power Analysis and Exploratory Research. Journal of Human Lactation, 39(4), 579–583. https://doi.org/10.1177/08903344231195625.

Hewitt, D. H., & He, Y. (2022). Cognitive Load and Website Usability: Effects of Contrast and Task Difficulty. Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 66(1), 1809–1813. https://doi.org/10.1177/1071181322661051.

Ismailova, R., & Inal, Y. (2022). Comparison of Online Accessibility Evaluation Tools: An Analysis of Tool Effectiveness. IEEE Access, 10, 58233–58239. https://doi.org/10.1109/ACCESS.2022.3179375.

Joglekar, Y., Purdy, D., Brock, S., Tandon, A., & Dong, A. (2022). Developing Digital Communication Competency in the Business Classroom. Business and Professional Communication Quarterly, 85(2), 141–168. https://doi.org/10.1177/23294906221089887.

Jordan, P. W. (2000). Inclusive Design: An Holistic Approach. Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 44(38), 917–920. https://doi.org/10.1177/154193120004403865.

Kane, S., Shulman, J., Shockley, T., & Ladner, R. E. (2007). A web accessibility report card for top international university web sites. In ACM International Conference Proceeding Series (Vol. 225). https://doi.org/10.1145/1243441.1243472.

Khasawneh, A., Gallagher, P. B., Jacobson, E. I., & Riley, L. (2023). Enhancing Ada Compliance for Websites and Online Applications: A Heuristic Evaluation Approach. Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 67(1), 1457–1463. https://doi.org/10.1177/21695067231192722.

Khenfer, J. (2023). A hands-on guide to conducting field experiments using mobile applications. International Journal of Market Research, 65(4), 380–401. https://doi.org/10.1177/14707853231165635.

Lahm, E. A. (2003). Assistive Technology Specialists: Bringing Knowledge of Assistive Technology to School Districts. Remedial and Special Education, 24(3), 141–153. https://doi.org/10.1177/07419325030240030301.

Martinez, E., & Pfister, L. (2023). Benefits and limitations of using low-code development to support digitalization in the construction industry. Automation in Construction, 152, 104909. https://doi.org/10.1016/j.autcon.2023.104909.

Martínez-Lasaca, F., Díez, P., Guerra, E., & de Lara, J. (2023). Dandelion: A scalable, cloud-based graphical language workbench for industrial low-code development. Journal of Computer Languages, 76, 101217. https://doi.org/10.1016/j.cola.2023.101217.

McGill, T., Bamgboye, O., Liu, X., & Kalutharage, C. S. (2023). Towards Improving Accessibility of Web Auditing with Google Lighthouse. 2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC), 1594–1599. https://doi.org/10.1109/COMPSAC57700.2023.00246.

Microsoft. (2022). Review a canvas app for accessibility in Power Apps - Power Apps | Microsoft Learn. Retrieved 20 June 2024, from Microsoft Learn website: https://learn.microsoft.com/en-us/power-apps/maker/canvas-apps/accessibility-checker.

Microsoft Design. (2024, May). Inclusive 101. Retrieved 16 May 2024, from Microsoft Inclusive Design website: https://inclusive.microsoft.design.

Monteiro, T., Abrantes, S., & Ratinho, M. (2024). A Solution for Submitting Expenses. Procedia Computer Science, 237, 20–27. https://doi.org/10.1016/j.procs.2024.05.075.

Moon, N. W., Baker, P. M. A., & Goughnour, K. (2019). Designing wearable technologies for users with disabilities: Accessibility, usability, and connectivity factors. Journal of Rehabilitation and Assistive Technologies Engineering, 6, 2055668319862137. https://doi.org/10.1177/2055668319862137.

Morse, C. A. (1986). Square One — An Introduction to Software. Measurement and Control, 19(4), 140–147. https://doi.org/10.1177/002029408601900402.

Neal, D., Gaber, S., Joddrell, P., Brorsson, A., Dijkstra, K., & Dröes, R.-M. (2023). Read and accepted? Scoping the cognitive accessibility of privacy policies of health apps and websites in three European countries. DIGITAL HEALTH, 9, 20552076231152160. https://doi.org/10.1177/20552076231152162.

Nisbet, P. D. (2020). Assistive technologies to access print resources for students with visual impairment: Implications for accommodations in high stakes assessments. British Journal of Visual Impairment, 38(2), 222–247. https://doi.org/10.1177/0264619619899678.

Ogbuju, E., Ayodeji, B., & Azeez, A. (2022). Performance and Accessibility Evaluation of University Websites in Nigeria. 2022 5th Information Technology for Education and Development (ITED), 1–7. https://doi.org/10.1109/ITED56637.2022.10051461.

Olariu, C., Gogan, M., & Rennung, F. (2016). Switching the Center of Software Development from IT to Business Experts Using Intelligent Business Process Management Suites. In V. E. Balas, L. C. Jain, & B. Kovačević (Eds.), Soft Computing Applications (pp. 993–1001). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-18416-6_79.

Olds, B. M., Moskal, B. M., & Miller, R. L. (2005). Assessment in Engineering Education: Evolution, Approaches and Future Collaborations. Journal of Engineering Education, 94(1), 13–25. https://doi.org/10.1002/j.2168-9830.2005.tb00826.x.

Palmer, Z. B., & Palmer, R. H. (2018). Legal and Ethical Implications of Website Accessibility. Business and Professional Communication Quarterly, 81(4), 399–420. https://doi.org/10.1177/2329490618802418.

Pedley, P. (2003). Implementing an Intranet in a Global Organization. Business Information Review, 20(3), 136–143. https://doi.org/10.1177/02663821030203003.

Perera, M. (2022). Inclusion: The Final Frontier. ITNOW, 64(1), 16–17. https://doi.org/10.1093/itnow/bwac007.

Rajaram, A., Olory, C., Leduc, V., Evaristo, G., Coté, K., Isenberg, J., … Fiset, P. O. (2022a). An integrated virtual pathology education platform developed using Microsoft Power Apps and Microsoft Teams. Journal of Pathology Informatics, 13, 100117. https://doi.org/10.1016/J.JPI.2022.100117.

Rajaram, A., Olory, C., Leduc, V., Evaristo, G., Coté, K., Isenberg, J., … Fiset, P. O. (2022b). An integrated virtual pathology education platform developed using Microsoft Power Apps and Microsoft Teams. Journal of Pathology Informatics, 13, 100117. https://doi.org/10.1016/j.jpi.2022.100117.

Ramkumar, S. (2025, March 9). COGA: Enhancing Web Accessibility for Cognitive and Learning Disabilities | User Vision. https://uservision.co.uk/thoughts/the-cognitive-and-learning-disabilities-accessibility-task-force-coga.

Runciman, B. (2023). Experiences of Disability and Neurodiversity in IT. ITNOW, 65(2), 27. https://doi.org/10.1093/combul/bwad050.

Saunders, M., Lewis, P., Thornhill, A., & Bristow, A. (2019). 'Research Methods for Business Students' Chapter 4: Understanding research philosophy and approaches to theory development.

Schmutz, S., Sonderegger, A., & Sauer, J. (2016). Implementing Recommendations From Web Accessibility Guidelines: Would They Also Provide Benefits to Nondisabled Users. Human Factors, 58(4), 611–629. https://doi.org/10.1177/0018720816640962.

Shani, A. B. (Rami), Coghlan, D., & Alexander, B. N. (2019). Rediscovering Abductive Reasoning in Organization Development and Change Research. The Journal of Applied Behavioral Science, 56(1), 60–72. https://doi.org/10.1177/0021886319893016.

Sharma, U., & Gupta, D. (2021). Email Ingestion Using Robotic Process Automation for Online Travel Agency. 2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), 1–5. https://doi.org/10.1109/ICRITO51393.2021.9596472.

Taylor, Z. W., & Burnett, C. A. (2019). Hispanic-Serving Institutions and Web Accessibility: Digital Equity for Hispanic Students With Disabilities in the 21st Century. Journal of Hispanic Higher Education, 20(4), 402–421. https://doi.org/10.1177/1538192719883966.

Teixeira, P., Eusébio, C., & Teixeira, L. (2021). How diverse is hotel website accessibility? A study in the central region of Portugal using web diagnostic tools. Tourism and Hospitality Research, 22(2), 180–195. https://doi.org/10.1177/14673584211022797.

Thiessen, P., & Russell, E. (2009). WAI-ARIA live regions and channels: ReefChat as a case example. Disability and Rehabilitation: Assistive Technology, 4(4), 276–287. https://doi.org/10.1080/17483100902903325.

Uslan, M. M., Burton, D. M., & Clements, C. W. (2008). Blood Glucose Meters That Are Accessible to Blind and Visually Impaired Persons. Journal of Diabetes Science and Technology, 2(2), 284–287. https://doi.org/10.1177/193229680800200219.

Vo, J., Hewitt, D. H., & He, Y. (2023). Web Accessibility: A Revisit of U.S. State and Territory COVID-19 Websites After Two Years. Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 67(1), 1084–1089. https://doi.org/10.1177/21695067231192275.

W3C. (2024a). https://wave.webaim.org.

W3C. (2024b). Web Accessibility Evaluation Tools. Retrieved 26 June 2024, https://wave.webaim.org

Warunek, L. N., Gruver, B., Bartko, L., & Blair, J. (2024). Assessing intradisciplinary pharmacy communication related to transitions of care. Exploratory Research in Clinical and Social Pharmacy, 14, 100438. https://doi.org/10.1016/j.rcsop.2024.100438.

Wentz, B., Pham, D. (June), & Tressler, K. (2017). Exploring the accessibility of banking and finance systems for blind users. First Monday, 22(3). https://doi.org/10.5210/fm.v22i3.7036.

Whitehead, D., Taket, A., & Smith, P. (2003). Action research in health promotion. Health Education Journal, 62(1), 5–22. https://doi.org/10.1177/001789690306200102.

Youngblood, N. E., & Lysaght, R. (2015). Accessibility and Use of Online Video Captions by Local Television News Websites. Electronic News, 9(4), 242–256. https://doi.org/10.1177/1931243115604885.

Youngblood, S. A. (2012). Communicating Web Accessibility to the Novice Developer: From User Experience to Application. Journal of Business and Technical Communication, 27(2), 209–232. https://doi.org/10.1177/1050651912458924

9. Appendices

9.1. Appendix A: Literature Review

Table 12: Online WCAG accessibility checkers in the literature

Tool	Studies mentioned
WAVE	(Alim, 2021; Bounajim et al., 2022; Sonnenberg, 2020; Vo et al., 2023; N. E. Youngblood & Lysaght, 2015; S. A. Youngblood, 2012)
TAW	(Alim, 2021; Singh et al., 2023; Teixeira et al., 2021)
Elll Page Checker	(Alim, 2021)
Mauve ++	(Ismailova & Inal, 2022; Vo et al., 2023)
Bobby	(N. E. Youngblood & Lysaght, 2015)
AChecker	(Ismailova & Inal, 2022; McGill et al., 2023; Schmoelz, 2023; Vo et al., 2023; N. E. Youngblood & Lysaght, 2015)
Vamola Validator	(Ismailova & Inal, 2022; Teixeira et al., 2021)
Access Monitor	(Ismailova & Inal, 2022; Teixeira et al., 2021)
Examinator	(Ismailova & Inal, 2022)
Cynthia Says	(Ismailova & Inal, 2022)

9.2. Appendix B: Results & Analysis Tables

Table 13: Results of Lighthouse accessibility audit on both staff directories

Lighthouse Category	Lighthouse feedback
Additional item to check manually	Interactive controls are keyboard focusable
Additional item to check manually	Interactive elements indicate their purpose and state
Additional item to check manually	The page has a logical tab order

Lighthouse Category	Lighthouse feedback
Additional item to check manually	Visual order on the page follows DOM order
Additional item to check manually	User focus is not accidentally trapped in a region
Additional item to check manually	The user's focus is directed to new content added to the page
Additional item to check manually	HTML5 landmark elements are used to improve navigation
Additional item to check manually	Offscreen content is hidden from assistive technology
Additional item to check manually	Custom controls have associated labels
Additional item to check manually	Custom controls have ARIA roles
Passed Audits	[aria-*] attributes match their roles
Passed Audits	[aria-hidden="true"] is not present on the document <body>
Passed Audits	[role]s have all required [aria-*] attributes
Passed Audits	[aria-*] attributes have valid values
Passed Audits	[aria-*] attributes are valid and not misspelled
Passed Audits	Buttons have an accessible name
Passed Audits	Image elements have [alt] attributes
Passed Audits	[user-scalable="no"] is not used in the <meta name="viewport"> element and the [maximum-scale] attribute is not less than 5
Passed Audits	ARIA attributes are used as specified for the element's role
Passed Audits	[aria-hidden="true"] elements do not contain focusable descendants
Passed Audits	Elements use only permitted ARIA attributes
Passed Audits	[role] values are valid
Passed Audits	Background and foreground colors have a sufficient contrast ratio
Passed Audits	Document has a <title> element
Passed Audits	<frame> or <iframe> elements have a title
Passed Audits	<html> element has a [lang] attribute
Passed Audits	<html> element has a valid value for its [lang] attribute
Passed Audits	No element has a [tabindex] value greater than 0
Passed Audits	Touch targets have sufficient size and spacing
Passed Audits	Uses ARIA roles only on compatible elements
Passed Audits	Deprecated ARIA roles were not used
Passed Audits	Image elements do not have [alt] attributes that are redundant text
Not Applicable	[accesskey] values are unique
Not Applicable	button, link, and menuitem elements have accessible names
Not Applicable	Elements with role="dialog" or role="alertdialog" have accessible names.
Not Applicable	ARIA input fields have accessible names
Not Applicable	ARIA meter elements have accessible names
Not Applicable	ARIA progressbar elements have accessible names
Not Applicable	Elements with an ARIA [role] that require children to contain a specific [role] have all required children

Lighthouse Category	Lighthouse feedback
Not Applicable	[role]s are contained by their required parent element
Not Applicable	Elements with the role=text attribute do not have focusable descendents.
Not Applicable	ARIA toggle fields have accessible names
Not Applicable	ARIA tooltip elements have accessible names
Not Applicable	ARIA treeitem elements have accessible names
Not Applicable	The page contains a heading, skip link, or landmark region
Not Applicable	<dl>'s contain only properly-ordered <dt> and <dd> groups, <script>, <template> or <div> elements
Not Applicable	Definition list items are wrapped in <dl> elements
Not Applicable	ARIA IDs are unique
Not Applicable	No form fields have multiple labels
Not Applicable	Heading elements appear in a sequentially descending order
Not Applicable	<html> element has an [xml:lang] attribute with the same base language as the [lang] attribute
Not Applicable	Input buttons have discernible text
Not Applicable	<input type="image"> elements have [alt] text
Not Applicable	Form elements have associated labels
Not Applicable	Links are distinguishable without relying on colour
Not Applicable	Links have a discernible name
Not Applicable	Lists contain only <li> elements and script supporting elements (<script> and <template>)
Not Applicable	List items (<li>) are contained within <ul>, <ol> or <menu> parent elements
Not Applicable	The document does not use <meta http-equiv="refresh">
Not Applicable	<object> elements have alternate text
Not Applicable	Select elements have associated label elements
Not Applicable	Skip links are focusable
Not Applicable	Tables have different content in the summary attribute and <caption>
Not Applicable	Cells in a <table> element that use the [headers] attribute refer to table cells within the same table
Not Applicable	<th> elements and elements with [role= "columnheader"/"rowheader"] have data cells they describe
Not Applicable	[lang] attributes have a valid value
Not Applicable	<video> elements contain a <track> element with [kind="captions"]

Table14: Age demographic for accessible staff directory

Age range	Count	Percentage
18-24	0	0.00
25-34	3	17.65
35-44	6	35.29
45-54	5	29.41
55-64	3	17.65
64+	0	0.00
Prefer not to say	0	0.00
Total	17	100.00

Table 15: Age demographics for inaccessible staff directory

Age range	Count	Percentage
18-24	0	0.00
25-34	3	17.65
35-44	7	41.18
45-54	6	35.29
55-64	1	5.88
64+	0	0.00
Prefer not to say	0	0.00
Total	17	100.00

Table 16: Gender demographics for accessible staff directory

Gender	Count	Percentage
Man	5	29.41
Woman	12	70.59
Non-binary	0	0.00
Prefer not to say	0	0.00
Total	17	100.00

Table 17: Gender demographics for inaccessible staff directory

Gender	Count	Percentage
Man	7	41.18
Woman	10	58.82
Non-binary	0	0.00
Prefer not to say	0	0.00
Total	17.00	100.00

Table 18: Difficulty of T1 for accessible staff directory

Difficulty of task	Count	Percentage
1	9	52.94
2	2	11.76
3	3	17.65
4	1	5.88
5	2	11.76
Total	17	100.00

Table19: Difficulty of T1 for inaccessible staff directory

Difficulty of task	Count	Percentage
1	9	52.94
2	3	17.65
3	0	0.00
4	1	5.88
5	4	23.53
Total	17	100.00

Table 20: Difficulty of T2 for accessible staff directory

Difficulty of task	Count	Percentage
1	3	17.65
2	5	29.41
3	3	17.65
4	4	23.53
5	2	11.76
Total	17	100.00

Table 21: Difficulty of T2 for inaccessible staff directory

Difficulty of task	Count	Percentage
1	6	35.29
2	6	35.29
3	2	11.76
4	3	17.65
5	0	0.00
Total	17	100.00

Table22: Difficulty of T2 for inaccessible staff directory

Difficulty of task	Count	Percentage
1	9	52.94
2	3	17.65
3	0	0.00
4	1	5.88
5	4	23.53
Total	17	100.00

Table 23: Difficulty of T3 for accessible staff directory

Difficulty of task	Count	Percentage
1	3	17.65
2	3	17.65
3	3	17.65
4	4	23.53
5	4	23.53
Total	17	100.00

Table 24: Difficulty of T3 for inaccessible staff directory

Difficulty of task	Count	Percentage
1	4	23.53
2	6	35.29
3	2	11.76
4	4	23.53
5	1	5.88
Total	17	100.00

© Journal of Accessibility and Design for All (JACCES), Volume 15, Issue 2, 2025, ISSN: 2013-7087
DOI: https://doi.org/10.17411/jacces.v15i2.590

How to cite this article

Andrews S. (2025). Low-code solution development and accessibility. Journal of Accessibility and Design for All, 15(2), 1-29. https://doi.org/10.17411/jacces.v15i2.590.