`

Cloud Based Accessibility Testing Service Definition

GCloud15

`

Introduction

Accessibility Testing helps organisations ensure digital services are usable by everyone and compliant with WCAG 2.2 AA and public sector accessibility requirements. The service combines automated testing and expert accessibility assessment to identify barriers across web and digital applications. Testing is integrated into agile and DevOps delivery to support early detection, clear remediation guidance, and ongoing compliance. The service provides structured reporting and evidence to support assurance, audit, and safe transition to live service operations while improving overall usability and inclusion.

The Challenges of Accessibility Testing

Delivering Accessibility Testing

Delivering effective accessibility testing presents a number of technical, process, and organisational challenges. While accessibility standards such as WCAG 2.2 AA provide clear success criteria, consistently identifying, prioritising, and remediating accessibility issues across evolving digital services requires discipline, tooling, and expertise

AI is used wherever possible and this can increase quality and decrease time to deliver.

One of the primary challenges is the **complexity of accessibility requirements themselves**. WCAG success criteria cover a wide range of considerations including keyboard interaction, screen reader compatibility, colour contrast, focus management, semantic structure, and dynamic content behaviour. Many accessibility issues are contextual and cannot be reliably identified through a single testing approach. Automated testing alone is insufficient, while manual testing without structure can be inconsistent and difficult to scale.

A related challenge is the **limitations of automated tooling**. Open-source tools such as Axe Core, Pa11y, Lighthouse, HTML CodeSniffer, and WAVE-based tooling can identify a significant subset of accessibility issues, including missing labels, incorrect ARIA usage, colour contrast failures, and structural HTML problems. However, these tools typically detect only 30–50% of potential accessibility issues. They cannot reliably assess usability, appropriateness of alternative text, logical focus order, or whether content is understandable to users of assistive technologies. This means automated findings must be interpreted and supplemented with expert judgement.

Integrating accessibility testing into **modern delivery pipelines** introduces further challenges. Running accessibility checks within CI/CD pipelines requires careful configuration to avoid false positives, brittle tests, or excessive noise that teams may learn to ignore. Open-source tools such as Axe CLI and Pa11y CI can be embedded into pipelines, but thresholds, reporting formats, and failure conditions must be defined sensibly. Without clear prioritisation, pipelines may either block delivery unnecessarily or fail to highlight genuinely critical accessibility risks.

Maintaining an **accessibility regression test pack** adds additional complexity. Accessibility issues can reappear as user interfaces change, frameworks are upgraded, or content is modified. Creating a regression pack that is stable, meaningful, and maintainable requires selecting appropriate pages, components, and user journeys, and ensuring tests remain aligned to the evolving application. Pipeline-based regression testing must balance coverage with execution time to remain viable within agile delivery cycles.

Another challenge lies in **dynamic and component-based user interfaces**. Modern web applications often rely heavily on JavaScript frameworks, client-side rendering, and reusable UI components. Accessibility issues may only surface after user interaction, state changes, or

`

asynchronous updates. Open-source tools can struggle to fully capture these behaviours unless carefully scripted. Ensuring accessibility coverage across dynamic states requires deeper understanding of the application and deliberate test design.

Assistive technology compatibility also presents challenges. While automated tools can flag code-level issues, validating real-world usability with screen readers, keyboard-only navigation, or high-contrast modes requires expertise and time. Differences between assistive technologies and browsers can lead to inconsistent behaviour that is difficult to reproduce and diagnose. Managing this variability while remaining pragmatic is a key challenge in accessibility testing.

A further challenge is **prioritisation and remediation guidance**. Accessibility testing often identifies a large number of issues, not all of which carry the same user impact or delivery risk. Without clear prioritisation aligned to WCAG conformance levels and user impact, teams may struggle to address the most important issues first. Providing actionable remediation guidance that developers can implement efficiently is essential but requires accessibility expertise beyond tool output.

Organisational maturity also affects accessibility outcomes. Accessibility testing is most effective when treated as an ongoing quality practice rather than a one-off compliance exercise. Teams unfamiliar with accessibility concepts may initially find results difficult to interpret or implement. Embedding accessibility testing into regular delivery cycles, supported by clear standards and repeatable processes, helps mitigate this challenge but requires sustained commitment.

The use of **open-source tooling** introduces both benefits and challenges. Open-source tools reduce licensing costs, improve transparency, and avoid vendor lock-in, but they may require additional configuration, integration effort, and ongoing maintenance. Keeping tools such as Axe Core, Pa11y, and Lighthouse up to date and aligned with evolving WCAG guidance is an ongoing responsibility.

Finally, accessibility testing must support **assurance and audit requirements**. Public sector services often require clear evidence of accessibility testing, decision-making, and remediation. Producing repeatable, auditable outputs from automated tools and pipeline-based regression packs requires consistent reporting formats and disciplined record keeping.

Overall, the challenge of accessibility testing lies in balancing automation with expert insight, integrating testing into delivery pipelines without friction, and maintaining accessibility as a continuous practice. By using open-source tools, structured regression packs, and pipeline-based execution, accessibility testing can be scaled effectively, but success depends on thoughtful implementation, prioritisation, and sustained integration into delivery processes.

Key Benefits

Some of the benefits are;

- Improves accessibility and usability for users with disabilities across digital services.
- Supports compliance with WCAG 2.2 AA and public sector accessibility obligations.
- Identifies accessibility issues early, reducing remediation cost and delivery risk.
- Combines automated and expert testing for broader, more reliable accessibility coverage.
- Open-source tooling avoids licensing costs and vendor lock-in.

`

- Pipeline-based accessibility regression testing prevents reintroduction of known issues.

- Clear prioritisation helps teams focus on highest-impact accessibility barriers first.
- Improves confidence in accessibility readiness for live service release.
- Provides repeatable evidence to support assurance, audit, and compliance reporting.
- Embeds accessibility into agile and DevOps delivery practices for sustained compliance.

Return on Investment

The return on investment includes the following.

- Early identification of accessibility issues reduces expensive late-stage remediation and rework.
- Open-source testing tools eliminate licensing costs while maintaining effective accessibility coverage.
- Pipeline-based regression testing prevents recurring accessibility defects, reducing ongoing fix effort.
- Improved accessibility lowers risk of legal challenge and reputational damage.
- Reduced production defects decrease support and incident management costs.
- Clear remediation guidance shortens developer fix time and improves delivery efficiency.
- Accessible services reduce user abandonment and improve task completion rates.
- Continuous accessibility testing reduces assurance and audit preparation effort.

Approach

The following activities are available with the Accessibility Testing.

Accessibility testing activities begin with discovery and planning to ensure testing is aligned to the service scope, delivery model, and accessibility obligations. This includes reviewing user journeys, supported browsers and devices, target WCAG version (typically WCAG 2.2 AA), and any relevant organisational or regulatory requirements. Key pages, components, and workflows are identified, along with known accessibility risks and dependencies such as third-party components.

A test tooling proof of concept is commonly carried out early in the engagement. Open-source tools such as Axe Core, Pa11y, Lighthouse, HTML CodeSniffer, and WAVE-based tooling are evaluated against the application to confirm suitability, coverage, reporting formats, and ease of integration. The proof of concept validates how tools behave against dynamic content, authentication flows, and protected routes, and helps establish realistic expectations around automated coverage and limitations.

Detailed accessibility test planning follows. This defines test scope, success criteria, prioritisation approach, execution strategy, and reporting standards. The plan also sets out how manual and automated testing will be combined, how defects will be categorised by WCAG criteria and impact, and how accessibility testing will align with sprint or release cycles.

`

Accessibility test cases are then designed and documented in a test management tool. Test cases are mapped to WCAG success criteria and written in a clear, repeatable format. They cover automated checks, manual validation steps, and assistive technology considerations. Documenting test cases ensures traceability, consistency, and reusability across releases and supports audit and assurance requirements.

Test data preparation is an important activity. Data is prepared to support meaningful accessibility testing, including realistic content lengths, form inputs, error states, validation messages, and dynamic content variations. Where authentication or role-based access is required, appropriate test accounts are configured to allow accessibility testing across different user journeys.

Manual accessibility testing and validation is performed before automation. This includes keyboard-only navigation, focus management, semantic structure validation, and screen reader interaction checks. Manual testing validates whether automated findings are accurate and identifies issues that tools cannot detect, such as inappropriate alternative text, confusing instructions, or illogical reading order. Manual validation ensures only meaningful, stable tests are selected for automation.

Once validated, suitable test cases are automated using open-source tooling. Automation focuses on repeatable checks with clear pass/fail outcomes, such as missing labels, ARIA misuse, colour contrast failures, heading structure, and landmark usage. Automated tests are structured to minimise false positives and avoid fragile selectors.

An accessibility regression test pack is created from the automated test cases. This pack targets critical pages, components, and journeys and is designed to run reliably as part of continuous integration pipelines. Regression packs help prevent previously fixed accessibility issues from being reintroduced as the application evolves.

Pipeline integration is then implemented. Accessibility tests are integrated into CI/CD pipelines using tools such as Pa11y CI or Axe CLI. Thresholds and failure conditions are defined so that critical accessibility failures are visible early without unnecessarily blocking delivery. Test results are produced in consistent, machine-readable formats to support reporting and analysis.

Defect management and reporting activities ensure that findings are actionable. Accessibility issues are logged with clear descriptions, WCAG references, severity, user impact, and remediation guidance. Reports prioritise issues by impact and conformance level, helping teams focus on the most significant barriers first.

Retesting and validation is carried out after fixes are applied. Automated and manual tests are re-run to confirm issues are resolved and no regressions have been introduced. Updated evidence is captured for assurance and audit purposes.

Finally, the service supports continuous improvement. Test cases, regression packs, and tooling configurations are reviewed and updated as the application, standards, or delivery approach evolve. Accessibility testing becomes an embedded, repeatable part of delivery rather than a one-off activity, supporting sustained compliance and improved user experience over time.

Training

Training is available and will take approximately 10 days.

`

Support

Support is available from 08:00 to 18:00 on working days in England. 1st

Line Support

1st line support will be available between the hours of 09:00 – 17:00 via the following channels;

- Web
- Phone
- email

2nd line support which will operate between the hours of 09:00 – 17:00.

Definition of SLA's (Will provide list)

- 0 – Critical – Within 1 Hour – Call back
- 1 – Severe – Within 2 Hours – Call back
- 2- High – 24 hours
- 3 – Medium – 48 hours
- 4 – Cosmetic – 72 hours