### **Full WCAG 2.1 AA Accessibility Testing Coverage with WAVE and Axe DevTools**

Below is a detailed, step-by-step **test coverage plan** for ensuring compliance with **WCAG 2.1 AA** standards using **WAVE** and **Axe DevTools**. This guide includes specific **test cases**, tools’ usage instructions, and the expected outcomes for comprehensive accessibility testing.

## **1. WCAG 2.1 AA Test Cases with WAVE**

WAVE is a browser extension used to visually identify accessibility issues on a webpage. It provides overlays and a report of errors, warnings, and best practices.

### **Test Setup for WAVE**

1. Install the WAVE browser extension for Chrome or Firefox from https://wave.webaim.org/extension/.
2. Open the target webpage in your browser.
3. Click the WAVE icon in the toolbar to analyze the page.

### **Detailed Test Cases for WAVE**

| **Test Case ID** | **Test Case** | **Steps to Execute** | **Expected Result** |
| --- | --- | --- | --- |
| **WAVE-001** | Missing Alternative Text | 1. Open WAVE.  2. Look for red icons labeled "Missing Alternative Text" on images. | All images should have descriptive alt attributes unless they are decorative (use alt="" for those). |
| **WAVE-002** | Empty Links | 1. Look for red icons labeled "Empty Link".  2. Hover over flagged icons for details. | All links must have meaningful text describing their destination or function. |
| **WAVE-003** | Text Contrast | 1. Open the "Contrast" tab in WAVE.  2. Review flagged text elements with low contrast. | Text should have a contrast ratio of at least 4.5:1 for normal text and 3:1 for large text or UI components. |
| **WAVE-004** | Heading Structure | 1. Go to the "Outline" tab in WAVE.  2. Inspect the hierarchy of heading levels (H1 to H6). | Heading levels must follow a logical order without skipping levels (e.g., no H1 > H3). |
| **WAVE-005** | Form Field Labels | 1. Check flagged form elements for missing or incomplete labels. | All form inputs must have descriptive labels that are programmatically associated. |
| **WAVE-006** | ARIA Roles and Landmarks | 1. Inspect ARIA roles or landmarks flagged in the WAVE report. | ARIA roles and landmarks should be used appropriately and consistently (e.g., main, nav). |
| **WAVE-007** | Navigation Accessibility | 1. Use the "Structure" tab to review the logical order of interactive elements.  2. Test tab navigation. | Interactive elements must follow a logical tab order and highlight visibly on focus. |
| **WAVE-008** | Language Attribute | 1. Verify if WAVE flags a missing or incorrect lang attribute in the HTML tag. | The lang attribute must be present and correctly identify the page's language (e.g., lang="en"). |
| **WAVE-009** | Link Purpose | 1. Check links flagged under "Features". | All links should provide clear context about their purpose (avoid “Click here”). |
| **WAVE-010** | Dynamic Content Accessibility | 1. Look for dynamic content issues flagged in WAVE, like missing ARIA live roles. | ARIA live regions should notify users about important dynamic changes on the page. |

## **2. WCAG 2.1 AA Test Cases with Axe DevTools**

Axe DevTools is a more developer-focused accessibility testing tool. It provides detailed test results and code fixes directly within browser DevTools.

### **Test Setup for Axe DevTools**

1. Install the Axe DevTools extension for your browser from https://www.deque.com/axe/devtools/.
2. Open the target webpage and right-click > Inspect to open browser DevTools.
3. Navigate to the Axe tab in DevTools and click "Analyze."

### **Detailed Test Cases for Axe DevTools**

| **Test Case ID** | **Test Case** | **Steps to Execute** | **Expected Result** |
| --- | --- | --- | --- |
| **AXE-001** | Analyze for Accessibility Violations | 1. Open Axe DevTools.  2. Click "Analyze".  3. Review categorized violations (Critical, Serious, Moderate). | All violations should be identified with actionable fixes. |
| **AXE-002** | Image Alt Text | 1. Check flagged images in the "Images" section.  2. Verify missing or inappropriate alt attributes. | All images must have meaningful alt text or be explicitly marked as decorative (alt=""). |
| **AXE-003** | Text Contrast | 1. Review contrast-related violations in the Axe report.  2. Inspect affected elements and suggested ratios. | Text should meet contrast ratios: 4.5:1 for normal text, 3:1 for large text. |
| **AXE-004** | Heading Order | 1. Review flagged heading issues in the "Headings" section.  2. Inspect the heading hierarchy. | Headings should be semantic and follow a logical sequence without skipping levels. |
| **AXE-005** | Form Labels and Instructions | 1. Check flagged form elements in the "Forms" section.  2. Inspect missing labels and accessible error messages. | All form inputs must have labels, instructions, and clear error feedback visible to assistive technologies. |
| **AXE-006** | Focus Management | 1. Use the keyboard (Tab) to navigate the page.  2. Inspect flagged focusable elements in Axe. | All focusable elements should have a visible focus indicator and follow a logical navigation order. |
| **AXE-007** | Link Purpose | 1. Check flagged links for unclear or duplicate link text.  2. Inspect suggested fixes in Axe. | Links should provide meaningful context about their destination or function. |
| **AXE-008** | Language Attribute | 1. Review the "HTML" section for violations related to missing langattributes. | The lang attribute must correctly specify the primary language of the page content (e.g., lang="en"). |
| **AXE-009** | ARIA Roles and Live Regions | 1. Inspect ARIA violations flagged by Axe.  2. Verify appropriate use of ARIA roles and live regions. | ARIA roles should enhance accessibility without being misused (e.g., role="button" for interactive elements). |
| **AXE-010** | Dynamic Content Accessibility | 1. Inspect flagged dynamic content elements.  2. Verify ARIA live roles are used to notify screen readers. | Dynamic content changes must notify users with appropriate ARIA live roles (e.g., aria-live="polite"). |

## **3. Manual Accessibility Test Cases**

While WAVE and Axe DevTools cover many automated checks, manual testing is essential for certain WCAG criteria that require human judgment.

| **Test Case ID** | **Test Case** | **Steps to Execute** | **Expected Result** |
| --- | --- | --- | --- |
| **MAN-001** | Keyboard Navigation | 1. Use the Tab and Shift+Tab keys to navigate.  2. Check focus visibility and navigation order. | The focus indicator should be visible, and focus order must follow a logical sequence. |
| **MAN-002** | Screen Reader Compatibility | 1. Test with screen readers (e.g., NVDA, JAWS, VoiceOver).  2. Navigate through headings, forms, and links. | All interactive elements should have meaningful labels and work seamlessly with screen readers. |
| **MAN-003** | Responsive Design | 1. Resize the browser window or zoom to 200%.  2. Verify if content adjusts without loss of functionality. | Content should remain usable and readable, with no horizontal scrolling introduced. |
| **MAN-004** | Multimedia Accessibility | 1. Play videos and verify captions.  2. Check for audio descriptions. | All videos should have accurate captions and audio descriptions where applicable. |
| **MAN-005** | Skip Navigation | 1. Check for a "Skip to Content" link at the top of the page.  2. Verify its functionality. | "Skip to Content" should move focus to the main content area. |

## **4. Best Practices for Comprehensive Testing**

1. **Combine Automated Tools**:
   * Use **WAVE** for a visual overview of issues and **Axe DevTools** for deeper code-level analysis.
2. **Retest After Fixes**:
   * After resolving issues, rerun tests with both tools to validate fixes.
3. \*\*Conduct Manual