Name: Nahush Patil

Individual Software Accessibility Assignment 3: Automated Testing

Tools used - QualWeb and Wave for Web Automation and Motorease for Mobile Automation

USPS - QUALWEB

For each accessibility issue found in the previous assignment, was it detected by any of the two automated tools chosen? Why or why not?

Incorrect Link Labeling (Text Scams vs. Email Scams)

Detected - Yes
Explanation - This rule checks whether each link has a descriptive label accessible to screen readers. Since incorrect or missing link text affects the accessible name, the tool likely detected it. However, it does not verify the semantic accuracy of the label (i.e., whether "Text Scams" actually leads to that page), so the tool can catch the presence of the label but not necessarily whether it's the right label.

Unclear Button Borders in Top Navigation Bar

Detected - No
Explanation - This is a visual design issue related to contrast and borders, which may not trigger a rule unless the text contrast itself is too low (e.g., ACT-R37 or ACT-R76). If the buttons have no text or sufficient color contrast numerically, automated tools won't flag them — even if they appear visually ambiguous to users.

Screen Reader Language Doesn't Change After Switching Language

Detected - No
Explanation - Most automated tools cannot simulate screen reader behavior or detect language switching behavior across dynamic content. This issue requires human judgment and testing with assistive technologies to confirm whether the lang attribute changes appropriately when the page content changes.

Name: Nahush Patil

For each automated tool, mention and explain (i.e., issue type, location, etc.) no more than three accessibility issues found by the tool that you think is not feasible/possible to be found with manual testing. Why do you think the accessibility issue cannot be found with manual testing? On the other hand, if no accessibility issue was found by the automated tool, explain why you think that is.

1) **ARIA required context role (ACT-R33)**

   Issue Description: Elements with ARIA roles (e.g., menuitem, cell) must be nested within required parent roles (e.g., menu, row respectively).

   Why Manual Testing Fails: These roles and relationships are not visually apparent in the UI. A tester would need to inspect the raw HTML and fully understand ARIA specifications — something even experienced developers can overlook.

2) **ARIA required owned elements (ACT-R38)**

   Issue Description: Some parent ARIA roles must contain specific child roles (e.g., list must have list item children).

   Why Manual Testing Fails: A screen reader might simply skip such content or read it in a misleading way — but without source code or specialized tools, the issue isn't obvious.

3) **Meta viewport does not prevent zoom (ACT-R14)**

   Issue Description: Pages must not disable user zooming via <meta name="viewport">.

   Why Manual Testing Fails: This issue cannot be detected just by looking at or using the page. You must inspect the <meta> tag's user-scalable attribute, which is hidden from view. Users with low vision rely on zoom; disabling it violates WCAG.

Name: Nahush Patil

For each automated tool, choose no more than five issues detected that were not found manually in the previous assignment, and state whether you agree that it is an accessibility issue. Explain why or why not.

1) Adding a link at the top of each page that goes directly to the main content area

   This technique, often implemented as a "Skip to main content" link, is essential for users navigating via keyboard or screen readers. Without it, users must repeatedly tab through navigation menus on every page. It's easy to miss in manual testing unless testers specifically check for keyboard navigability at the start of the page.

2) ARIA required context role

   This rule ensures that ARIA roles like menuitem, cell, or listitem are placed in valid parent containers (menu, row, or list). It directly impacts screen reader interpretation and page structure. These relationships are difficult to verify without parsing the code, so they're often missed during manual checks.

3) meta viewport does not prevent zoom
   If zoom is disabled via the viewport meta tag (e.g., user-scalable=no), users with low vision cannot enlarge the text. This issue is not visible to manual testers unless they inspect the HTML or attempt to zoom and notice it doesn't work. It's a critical issue for mobile accessibility.

4) Using percent, em, or named values for font sizes

   Using relative units (em, %) ensures that text resizes with browser and system settings. Fixed units like px can prevent users from adjusting font sizes for readability. It's rarely checked manually unless code is reviewed directly, making automated detection valuable

5) Text has enhanced contrast

   This checks whether text meets enhanced contrast ratios (7:1), which benefit users with significant visual impairments. While not required under WCAG AA, it's a strong inclusive design practice. This issue is hard to spot manually without using contrast analysis tools, especially if color differences are subtle.

Name: Nahush Patil

USPS - WAVE

Incorrect Link Labeling (Text Scams vs. Email Scams)

Detected - No
Explanation - Automated tools like WAVE can check if a link has an accessible name (i.e., text or aria-label), but they cannot determine if the link label is semantically accurate or contextually misleading. In this case, the labels were incorrect (e.g., "Text Scams" actually pointed to "Email Scams"), which is a content-related error requiring human review. Only manual testing can catch such mismatches.

Unclear Button Borders in Top Navigation Bar

Detected - Yes
Explanation - AVE flags elements that lack sufficient color contrast, focus visibility, or clear interactive cues. Buttons with missing or faint borders may be flagged under contrast errors or keyboard accessibility issues. If these buttons also use tabindex="-1" or are not focusable, it makes them both visually and interactively inaccessible, especially for users navigating via keyboard.

Screen Reader Language Doesn't Change After Switching Language

Detected - No
Explanation - This issue involves the failure to update the lang attribute dynamically when the site's language is changed. Tools like WAVE only detect the static value of the lang attribute in the HTML <html lang="..."> tag. They cannot simulate user interaction to check whether the language changes dynamically and if assistive technologies are notified of the change. This must be verified manually using a screen reader.

For each automated tool, mention and explain (i.e., issue type, location, etc.) no more than three accessibility issues found by the tool that you think is not feasible/possible to be found with manual testing. Why do you think the accessibility issue cannot be found with manual testing? On the other hand, if no accessibility issue was found by the automated tool, explain why you think that is

Contrast Errors – Very Low Contrast

Name: Nahush Patil

Issue Type: Contrast (Visual Design, ACT-R37 / ACT-R76 equivalent)

Why Not Feasible Manually:
It is extremely difficult to determine contrast ratios by eye. Even skilled testers cannot accurately assess whether text meets the WCAG 2.1 minimum contrast thresholds (4.5:1 for normal text or 3:1 for large text) without a contrast checking tool. Automated tools like WAVE use precise color values from CSS and compute the exact ratio, ensuring reliability.

Empty Heading
Issue Type: Semantic Structure

Why Not Feasible Manually:
Empty headings are often invisible to sighted users unless reviewing the page's HTML directly. They can easily be missed unless a tester is specifically scanning the DOM structure or using a screen reader. Automated tools detect these quickly and reliably by parsing heading tags and checking their content.

Linked Image Missing Alternative Text
Issue Type: Image Accessibility

Why Not Feasible Manually:
Manually checking whether a linked image has a meaningful alt attribute requires viewing the source code or using developer tools for every image. This becomes impractical for large pages. WAVE automates this by scanning every link-image combination.

For each automated tool, choose no more than five issues detected that were not found manually in the previous assignment, and state whether you agree that it is an accessibility issue. Explain why or why not.

Linked image missing alternative text

Images that act as links must have meaningful alternative text so screen reader users understand the purpose of the link. Without it, the link is effectively invisible to them.

Empty heading
Headings are used by assistive technologies for navigation. An empty heading creates confusion or false structure and should be avoided.

Empty link
Links with no text or accessible name provide no context to screen readers, making them useless and frustrating for users relying on assistive tech.

Very low contrast
Low contrast between text and background can make content unreadable for users with low vision or color blindness. This is a critical accessibility failure.

Suspicious alternative text
Wrong description of the image can be misleading for the screen reader thereby confusing the user.

WALMART - QUALWEB

For each accessibility issue found in the previous assignment, was it detected by any of the two automated tools chosen? Why or why not?

Arrow Keys Do Not Navigate Sub-Menus (e.g., Departments)

Name: Nahush Patil

Detected - No
Explanation - This issue relates to keyboard accessibility, specifically keyboard event handling (e.g., using arrow keys to navigate menus). Automated tools like QualWeb primarily check for static code violations and WCAG violations related to markup, such as missing ARIA roles or improper focus management. However, dynamic behavior triggered by JavaScript, such as improper event listeners or failure to handle key events, typically requires manual or script-based testing. Therefore, QualWeb cannot detect this kind of interactive keyboard navigation issue.

Search Bar and Icon Not Always Recognized by Screen Reader

Detected - No
Explanation - Screen reader compatibility issues often arise from missing or misused ARIA labels, roles, or alt attributes, or from incorrect DOM labeling. If the search bar and icon have some semantic HTML but are still not announced properly due to dynamic rendering or poor labeling logic, QualWeb might not flag them unless the relevant attributes are explicitly missing or incorrect in the static DOM snapshot. These issues often require live interaction and screen reader testing, which automated tools cannot fully simulate.

Unresponsive Mouse Navigation

Detected - No
Explanation - This is a functional issue related to event handling for mouse interactions—such as clicks or hovers that don’t trigger the intended UI changes. QualWeb and similar tools don't interact with the page as a user would. They analyze DOM structure, accessibility tree, and code semantics, but not the actual UI behavior during interaction. As such, issues involving non-working buttons, broken click listeners, or hover states are typically not detectable unless they stem from improper HTML elements or attributes.

Out-of-Stock Items Not Read Aloud by Screen Reader

Detected - No
Explanation - This issue is about dynamic content updates and screen reader announcement of state changes, which require ARIA live regions or similar mechanisms. If Walmart is updating the DOM visually (e.g., showing “Out of Stock”) without using appropriate ARIA live attributes or roles, screen readers might not announce the change. QualWeb can detect missing ARIA live regions in some cases, but

it cannot simulate or verify live announcements the way a screen reader can in real time, so it may miss this kind of contextual accessibility failure.

For each automated tool, mention and explain (i.e., issue type, location, etc.) no more than three accessibility issues found by the tool that you think is not feasible/possible to be found with manual testing. Why do you think the accessibility issue cannot be found with manual testing? On the other hand, if no accessibility issue was found by the automated tool, explain why you think that is.

1) id attribute value is unique (WCAG-T35)

Issue Description: This issue checks whether all elements have unique id values across the page. It's not feasible to detect manually on large pages because tracking every id across potentially hundreds of nodes would require painstaking inspection of the HTML source**.**

Why Manual Testing Fails: Manual testers would have to scan and cross-compare all id attributes line-by-line, which is both time-consuming and error-prone. Automated tools can scan the full DOM and instantly flag duplicates with high accuracy**.**

2) Failure due to specifying foreground colors without specifying background colors (WCAG-T31)

Issue Description: This issue arises when only one of the two (foreground or background) colors is explicitly defined. This can cause poor contrast visibility when the page is viewed with user-defined styles (e.g., in high-contrast mode).

Why Manual Testing Fails: Unless testers manually inspect all CSS rules, including inheritance and browser defaults, it's nearly impossible to confirm whether both values are explicitly declared. Automated tools can analyze the computed styles and inheritance tree in milliseconds.

3) Visible label is part of accessible name (ACT-R30)

Issue Description: This rule checks that the text visible to users matches the element's accessible name (used by screen readers). If a button says "Buy Now" visually but the accessible name is only "Buy", this can confuse screen reader users.

Why Manual Testing Fails: Manual testers would need to run a screen reader and inspect the accessibility tree for each element—something that's both labor-intensive and easy to overlook. Automated tools directly compare the visible label and the programmatic name, catching mismatches quickly.

For each automated tool, choose no more than five issues detected that were not found manually in the previous assignment, and state whether you agree that it is an accessibility issue. Explain why or why not.

1) id attribute value is unique (WCAG-T35)

Duplicate id values can break assistive technologies (like screen readers or scripts that rely on unique element targeting). For example, if multiple elements share the same id, the aria-labelledby or for attributes may not behave as expected. This affects navigability and semantic accuracy.

2) Combining adjacent image and text links for the same resource (WCAG-T10)

Disagree - If the image and text are visually distinct for design reasons or enhance recognition, separating them might be justifiable — as long as each link is clearly labeled. Still, combining them is generally best practice for clarity.

3) Failure due to only foreground or background color being specified (WCAG-T31)

Users relying on high contrast modes or custom color schemes may struggle if only the foreground or background is defined — causing content to become invisible or unreadable when default styles clash.

4) Visible label is part of accessible name (ACT-R30)

When an interactive element (like a button) has a visible label that doesn't match its programmatic name, screen reader users may receive confusing or incomplete cues. For instance, a button showing "Pay Now" but being read as "Submit" may lead to hesitation or misunderstanding.

5) Providing submit buttons (WCAG-T19)

Forms must have explicit submit buttons to allow keyboard users and screen readers to activate them reliably. Implicit submissions (e.g., onchange or JavaScript triggers) can confuse or disorient users relying on accessible controls.

Name: Nahush Patil

WALMART - WAVE

For each accessibility issue found in the previous assignment, was it detected by any of the two automated tools chosen? Why or why not?

Arrow Keys Do Not Navigate Sub-Menus (e.g., Departments)

Detected - No
Explanation - Automated tools cannot simulate keyboard interactions or dynamic behavior like pressing arrow keys to navigate dropdown menus. This type of issue involves JavaScript event handling and focus management, which can only be tested manually using a keyboard.

Search Bar and Icon Not Always Recognized by Screen Reader

Detected - No
Explanation - Tools like WAVE can check if a field or icon has an accessible label, but they don't simulate screen reader behavior. If labeling is done dynamically or is context-dependent (e.g., via placeholder text only or ARIA misuse), it may not be flagged. Only real screen reader testing can detect whether the element is properly announced.

Unresponsive Mouse Navigation

Detected - No
Explanation - Automated tools focus on HTML structure, ARIA attributes, and CSS properties — not event-driven interactivity like mouse hovers or clicks that fail to work. This issue requires manual interaction testing and possibly browser debugging tools to verify click/hover functionality.

Out-of-Stock Items Not Read Aloud by Screen Reader

Detected - No
Explanation - This is a semantic communication issue. If the "out of stock" status is visually styled but not programmatically exposed (e.g., via aria-hidden="false" or aria-label="Out of Stock"), automated tools won't catch it. Only a screen reader test can reveal whether that information is announced correctly.

Name: Nahush Patil

For each automated tool, mention and explain (i.e., issue type, location, etc.) no more than three accessibility issues found by the tool that you think is not feasible/possible to be found with manual testing. Why do you think the accessibility issue cannot be found with manual testing? On the other hand, if no accessibility issue was found by the automated tool, explain why you think that is.

Redundant Alternative Text
Issue Type: Alert – Image Accessibility

Why Not Feasible Manually:
Manually reviewing every image and comparing its alt attribute to surrounding visible content is time-consuming and often overlooked. Automated tools like WAVE can instantly detect redundancy, helping avoid unnecessary repetition for screen reader users.

Suspicious Alternative Text

Issue Type: Alert – Image Accessibility

Why Not Feasible Manually:
Whether alt text is “suspicious” involves scanning hundreds of images and judging based on naming patterns or redundancy. Automated tools apply rules and heuristics to flag likely problems quickly.

Image with Title but No Alt Text
Issue Type: Alert – Image Accessibility

Why Not Feasible Manually:
Visual inspection won’t show whether an image has an alt tag unless the HTML source is closely examined. This detail is often missed in manual reviews, especially when the image appears decorative.

Name: Nahush Patil

For each automated tool, choose no more than five issues detected that were not found manually in the previous assignment, and state whether you agree that it is an accessibility issue. Explain why or why not.

Redundant Title Text
Disagree - If the redundancy is harmless and does not affect functionality or clarity (e.g., repeating simple labels), it may be considered low impact. Also, title attributes are often ignored by screen readers.

<noscript> Element Present
This might indicate that important content is only available through JavaScript, which can limit access for users with JavaScript disabled or using browsers that don't support it well. It also reflects incomplete progressive enhancement.

Redundant Link
Having adjacent links that point to the same URL can be confusing for screen reader users, as the links are announced separately. It also adds unnecessary tab stops for keyboard navigation.

Possible Heading
Disagree - Sometimes text is styled for emphasis, not structural navigation, so not all bold or large text needs to be a heading. It depends on the intent — context matters.

Image with Title but No Alt Text
The alt attribute is critical for screen reader users, while the title is inconsistently supported. An image with no alt but only a title is likely inaccessible to assistive tech.

MOTOREASE

UTDALLAS MOBILE APP

For each accessibility issue found in the previous assignment, was it detected by MotorEase? Why or why not?

1) Screen Reader Mispronounces Terms:

Name: Nahush Patil

MotorEase Detection: MotorEase doesn't specifically report on issues related to screen reader pronunciation, as there is no direct mention of text-to-speech mispronunciations or screen reader behavior in the provided report. The tool primarily focuses on interactive elements, touch targets, and visual issues (e.g., "Touch Target Detector" or "Expanding Sections Detector").

Why: MotorEase is designed to detect interactive accessibility issues like touch targets, expanding sections, and layout violations. Issues like screen reader mispronunciations typically require specific screen reader testing or a more specialized tool focused on text-to-speech behavior.

2) Login Button Redirects Incorrectly:

MotorEase Detection: The report does not mention specific behavior regarding button actions like "Login Button Redirects Incorrectly." However, it detects visual elements such as touch target violations (e.g., "Interactive Elements: 28 | Violating Elements: 28" for some screenshots).

Why: MotorEase focuses on visual and layout elements rather than functional testing like button actions or redirects. This kind of issue typically requires functional testing to ensure proper action on button clicks, which automated tools like MotorEase aren't designed to handle.

3) Tab Key Does Not Show Focus:

MotorEase Detection: The report does not mention anything about focus visibility (e.g., "Tab Key Does Not Show Focus"). The tool checks for visual and interactive elements but doesn't directly report on keyboard navigation or focus management.

Why: MotorEase does not provide feedback on keyboard accessibility features like focus management, which is essential for users relying on keyboard navigation. This issue would need to be manually tested with assistive technologies like keyboard-only navigation.

Name: Nahush Patil

Mention and explain (i.e., issue type, location, etc.) no more than three accessibility issues found by MotorEase that you think is not feasible/possible to be found with manual testing. Why do you think the accessibility issue cannot be found with manual testing?

1) Touch Target Detection:

MotorEase Report: In the screenshot edu.utdallas.utd_Top_Down_32.png, it reports 28 interactive elements with 28 violating elements.

Why Not Feasible Manually: It's hard to manually assess every touch target for size and accessibility in real-time, especially when there are a large number of interactive elements. MotorEase automates this process and can quickly detect whether interactive elements meet accessibility standards for size and spacing, something that would be tedious and error-prone in manual testing.

2) Icon Distance Violations:

MotorEase Report: In edu.utdallas.utd_Top_Down_7.png, the report flags icon distance results with violations.

Why Not Feasible Manually: Manually evaluating the spacing and distance between icons is subjective and can be difficult to measure precisely without automation tools. MotorEase uses algorithms to calculate exact distances between icons, ensuring consistent measurements across the entire app.

3) Persisting Elements Detection:

MotorEase Report: The tool detects the presence of elements across multiple screens and flags if any screen elements persist incorrectly (as seen in some of the images like edu.utdallas.utd_Top_Down_22.png).

Why Not Feasible Manually: This type of issue can be difficult to detect manually because it involves checking whether elements persist across different screens or states in the app. MotorEase can detect and track these persistent elements programmatically, while manually, testers might miss these subtleties, especially in a dynamic environment where multiple screens interact.

Choose no more than five issues detected that were not found manually in the previous assignment, and state whether you agree that it is an accessibility issue. Explain why or why not.

1) Touch Target Violations (e.g., edu.utdallas.utd_Top_Down_32.png):

MotorEase Report: 28 interactive elements with 28 violations.

This is definitely an accessibility issue. Ensuring that interactive elements meet the required size and spacing is crucial for users with motor impairments. Small or improperly spaced touch targets can make navigation difficult, which would be challenging to detect manually without tools like MotorEase.

2) Icon Distance Violations (e.g., edu.utdallas.utd_Top_Down_7.png):

MotorEase Report: Icon distance violations.

This is an accessibility issue. Proper spacing between icons is essential for users who rely on screen readers, magnifiers, or have limited dexterity. Manual testing might overlook these issues, especially when there are many icons across the app.

3) Interactive Element Violations in edu.utdallas.utd_Top_Down_22.png:

MotorEase Report: 1 interactive element with violations.

This is an accessibility issue. Each interactive element should be appropriately designed for easy interaction, and violations indicate poor design that can hinder the app's usability for users with disabilities

Name: Nahush Patil

4) Expanding Sections Missing in Multiple Screens (e.g., edu.utdallas.utd_Top_Down_8.png):

MotorEase Report: "Expanding elements: No Expanding Detected."

The lack of expandable sections in a user interface can limit how users navigate through content, especially if the content is dynamic. MotorEase detected this, and it is a valid accessibility issue, particularly for users with cognitive or visual impairments.

5) Non-Detectable Touch Target Violations (e.g., edu.utdallas.utd_Top_Down_4.png):

MotorEase Report: No interactive elements or violations detected, which is the ideal case.

The fact that no violations were found in some screenshots is a positive sign that the app is accessible. Manual testers might miss these cases due to fatigue or oversight, so having an automated check is beneficial.

---

iHeart Radio MOBILE APP

For each accessibility issue found in the previous assignment, was it detected by MotorEase? Why or why not?

1) Screen Reader Looping to Same Page:

MotorEase Detection: The report doesn't explicitly mention any screen reader behavior or focus management issues, such as looping focus. MotorEase primarily detects visual and interactive elements like touch targets and expanding sections, but doesn't check for focus order or screen reader behavior.

Why: MotorEase is not equipped to test screen reader behavior or focus management, which requires specific assistive technology or manual testing with screen readers.

2) Shift+Tab Acts Like Tab (Cannot Go Back):

MotorEase Detection: The MotorEase report does not mention keyboard navigation or focus visibility issues, including Shift+Tab behavior.

Name: Nahush Patil

Why: MotorEase focuses on visual elements like touch targets and expanding sections. It does not evaluate keyboard navigation behaviors, which would require manual testing or specialized accessibility tools.

3) Music Time Bar Cannot Be Controlled by Arrow Keys:

MotorEase Detection: There is no mention of this specific issue related to keyboard control of the music time bar in the report.

Why: MotorEase primarily checks for touch target violations, not keyboard or media control behaviors. This type of issue would require functional testing or keyboard-specific accessibility tools.

Mention and explain (i.e., issue type, location, etc.) no more than three accessibility issues found by MotorEase that you think is not feasible/possible to be found with manual testing. Why do you think the accessibility issue cannot be found with manual testing?

1) Touch Target Violations (e.g., com.clearchannel.iheartradio.controller_Bottom_Up_55.png):

MotorEase Report: The report lists 2 interactive elements with 2 violating elements.

Why Not Feasible Manually: Detecting touch target violations requires precise measurement of interactive elements' size and spacing. Manual testing of each interactive element's accessibility is time-consuming and prone to error, especially on complex screens. MotorEase automates this process efficiently, ensuring that touch targets meet size and spacing requirements.

2) Icon Distance Violations (e.g., com.clearchannel.iheartradio.controller_Bottom_Up_57.png):

MotorEase Report: The report detects violations regarding icon distances.

Why Not Feasible Manually: Manually measuring distances between icons or elements on a screen is difficult without the aid of automation tools. MotorEase uses algorithms to calculate these distances precisely, making it a better approach for identifying such violations across multiple screens.

3) Expanding Sections Not Detected (e.g., com.clearchannel.iheartradio.controller_Top_Down_57.png):

MotorEase Report: Expanding elements detection shows "No Expanding Detected."

Why Not Feasible Manually: Manual testing might miss cases where content should expand or collapse based on user actions. MotorEase provides an automated way to track these dynamic behaviors across multiple screens, which might be tedious to test manually, especially on a large scale or for highly dynamic content.

Choose no more than five issues detected that were not found manually in the previous assignment, and state whether you agree that it is an accessibility issue. Explain why or why not.

1) Touch Target Violations (e.g., com.clearchannel.iheartradio.controller_Bottom_Up_55.png):

MotorEase Report: 2 interactive elements with violations.

This is an accessibility issue. Small or improperly spaced touch targets make navigation difficult for users with motor impairments. Ensuring proper size and spacing for touch targets is critical for usability and accessibility. Manual testing might miss these violations, especially on larger screens with multiple interactive elements.

2) Icon Distance Violations (e.g., com.clearchannel.iheartradio.controller_Top_Down_2.png):

MotorEase Report: Detects distance issues between icons.

This is an accessibility issue. Proper spacing between interactive elements like icons is essential for users with cognitive or motor impairments. Manual testing may not reliably identify such violations, especially if the spacing is subtle or inconsistent across different devices.

3) Expanding Sections Not Detected (e.g., com.clearchannel.iheartradio.controller_Top_Down_7.png):

MotorEase Report: No expanding elements detected.

If content is supposed to expand dynamically (e.g., showing additional options or content), and it doesn't work as expected, this creates a usability barrier for all users, including those with disabilities. This issue is hard to detect manually, especially when there are many dynamic components.

4) Non-Detectable Touch Target Violations (e.g., com.clearchannel.iheartradio.controller_Top_Down_2.png):

MotorEase Report: Detects 1 interactive element with violations.

This is an accessibility issue, as any touch target that is too small or poorly positioned can cause difficulties for users with limited dexterity. Manual testing is prone to oversight, especially on complex layouts, making MotorEase a helpful tool for detecting such violations.

5) Violations in Button Placement or Interaction (e.g., com.clearchannel.iheartradio.controller_Top_Down_12.png):

MotorEase Report: Detects touch target violations in button placement.

Improper button placement can make navigation difficult for all users, especially those relying on assistive technologies or having motor impairments. Manual testing can easily overlook such violations, particularly in apps with multiple screen states and dynamic elements.