proach, building upon the insights from Budai [3] in Flutter - following guidelines and then giving advice into introducing new ones.

3.4.1 Home Screen

The Home Screen serves as the primary entry point of the *AccessibleHub* application. It provides key metrics on accessibility compliance (e.g., number of accessible components, $WCAG_G$ conformance level) and direct navigation to core sections: *Accessible Components* (Quick Start), *Best Practices*, *Testing Tools*, and the *Framework Comparison*. An example of the interface is shown in Figure 3.9.

3.4.1.1 Component Inventory and WCAG/MCAG Mapping

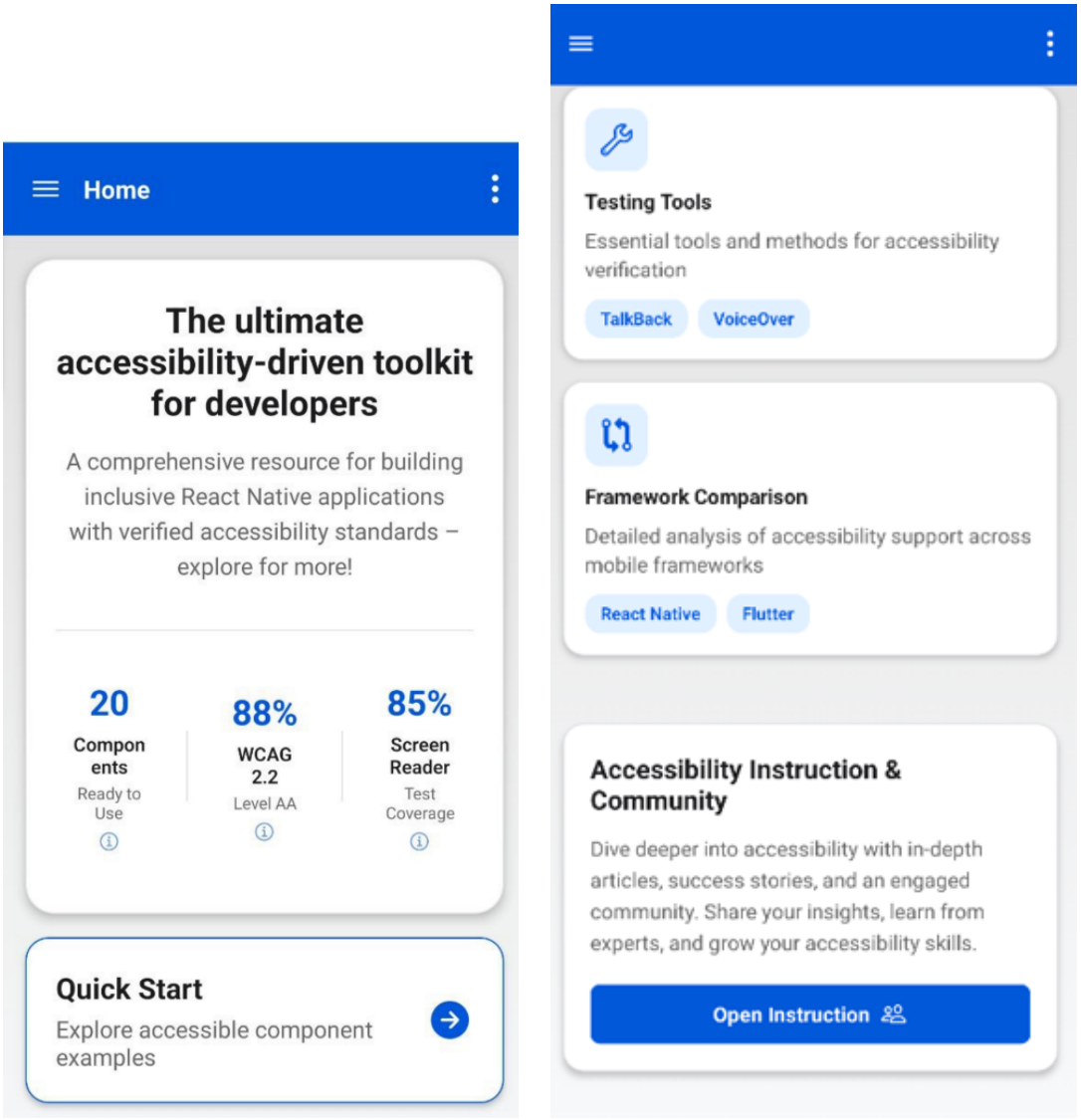
Table 3.1 provides a formal mapping between the UI components, their semantic roles, the specific WCAG 2.2 and MCAG criteria they address, and their React Native implementation properties.

Table 3.1: Home Screen Component-Criteria Mapping

Component	**Semantic Role**	**WCAG 2.2 Criteria**	**MCAG Considerations**	**Implementation Properties**
Hero Title	heading	1.4.3 Contrast (AA) 2.4.6 Headings (AA)	Text readability on variable screen sizes	`accessibilityRole="hea`

Stats Cards	button	1.4.3 Contrast (AA)
		2.5.8 Target Size (AA)
		4.1.2 Name, Role, Value (A)
			Touch target size
			Non-essential information
				`accessibilityRole="but`
				`accessibilityLabel="${`
				`${type}, tap for details"`
				`accessibilityHint="Sho`
				`${type} details"`

Component	**Semantic Role**	**WCAG 2.2 Criteria**	**MCAG Considerations**	**Implementation Properties**
Decorative Icons	none	1.1.1 Non-text Content (A)	Reduction of unnecessary focus stops	`accessibilityElementsH` `importantForAccessibil`
Quick Start Button	button	1.4.3 Contrast (AA) 2.5.8 Target Size (AA) 2.5.2 Pointer Cancellation (A)	One-handed operation	`accessibilityRole="but` `minHeight: 48` `minWidth: 150`
Feature Cards	button	1.3.1 Info and Relationships (A) 1.4.3 Contrast (AA) 2.5.8 Target Size (AA)	Logical grouping	`accessibilityRole="but` `accessibilityLabel="${` `accessibilityHint="${h`
Modal Dialog	dialog	2.4.3 Focus Order (A) 4.1.2 Name, Role, Value (A)	Keyboard trap prevention	`accessibilityRole="dia` Focus management implementation
Modal Tabs	tablist	2.4.7 Focus Visible (AA) 4.1.2 Name, Role, Value (A)	Touch interaction	`accessibilityRole="tab` `accessibilityState={{` `selected:` `isActive }}`



(a) Home Screen - Part 1 **(b)** Home Screen - Part 2

Figure 3.9: Side-by-side view of the two Home sections, with metrics and navigation buttons

3.4.1.2 Formal Metrics Calculation Methodology

The Home Screen displays three key metrics that provide quantitative measurements of the application's accessibility. These metrics are not arbitrary but are calculated using a formal methodology defined in the `calculateAccessibilityScore` function within `index.tsx`. Figure 3.10 shows how these metrics are displayed to users, with information buttons that reveal detailed calculation methods.

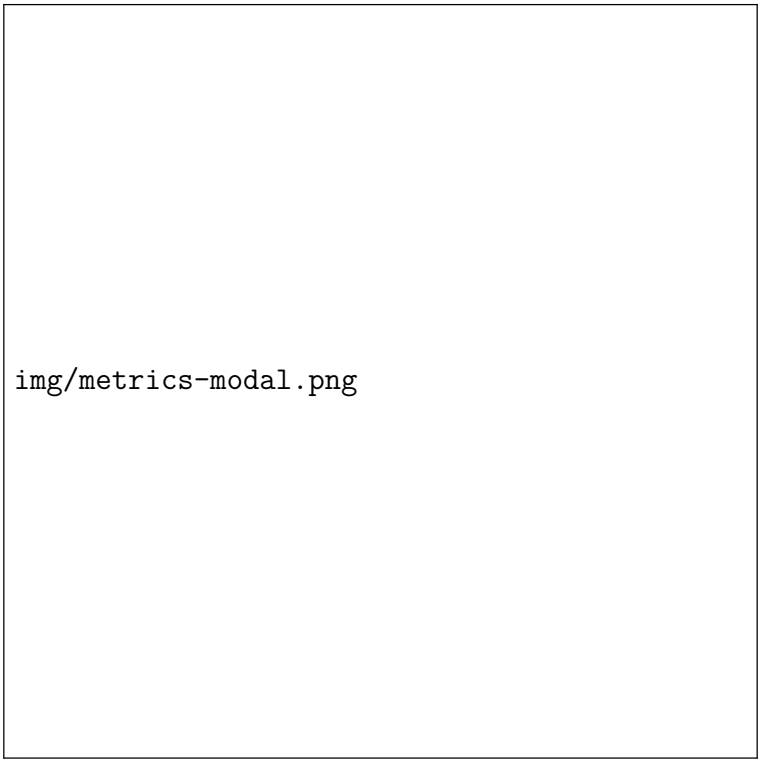


Figure 3.10: Modal dialog showing formal accessibility metric calculation methodology

3.4.1.2.1 Component Accessibility Score The Component Accessibility Score is calculated using the following formula:

$$\text{ComponentScore} = \left(\frac{\text{AccessibleComponents}}{\text{TotalComponents}} \right) \times 100 \tag{3.1}$$

Where:

- `AccessibleComponents` = Number of components with properly implemented accessibility attributes (18)
- `TotalComponents` = Total number of UI components used in the application (20)

The implementation in `index.tsx` maintains a formal registry of all UI components:

3.4.1.2.2 WCAG Compliance Score The WCAG Compliance Score represents the percentage of implemented WCAG 2.2 success criteria across four principles:

```
// Component registry with accessibility status tracking
const componentsRegistry = {
  'button': { implemented: true, accessible: true, screens:
     ['home', 'gestures'] },
  'text': { implemented: true, accessible: true, screens:
     ['home', 'guidelines'] },
  // ... other components
  'tooltip': { implemented: true, accessible: false,
     screens: [] },
  // Total: 20 components, 18 fully accessible
};

// Component calculation
const componentsTotal =
   Object.keys(componentsRegistry).length;
const accessibleComponents =
   Object.values(componentsRegistry)
  .filter(c => c.implemented && c.accessible).length;
const componentScore = Math.round((accessibleComponents /
   componentsTotal) * 100);
```

Listing 3.1: Component registry and calculation

$$\text{WCAGCompliance} = \left(\frac{\text{CriteriaLevelAMet} + \text{CriteriaLevelAAMet}}{\text{TotalCriteria}} \right) \times 100 \tag{3.2}$$

Where:

- `CriteriaLevelAMet` = Number of Level A success criteria implemented (25)
- `CriteriaLevelAAMet` = Number of Level AA success criteria implemented (13)
- `TotalCriteria` = Total applicable WCAG criteria (43)

The implementation maintains a comprehensive tracking system for WCAG criteria:

3.4.1.2.3 Screen Reader Testing Score

The Screen Reader Testing Score represents empirical testing with VoiceOver (iOS) and TalkBack (Android):

$$\text{TestingScore} = \left(\frac{\text{VoiceOverAvg} + \text{TalkBackAvg}}{2} \right) \times 20 \tag{3.3}$$

```
// WCAG criteria tracking with implementation status
const wcagCriteria = {
  '1.1.1': { level: 'A', implemented: true, name: "Non-text Content" },
  '1.3.1': { level: 'A', implemented: true, name: "Info and Relationships" },
  // ... other criteria
  '4.1.3': { level: 'AA', implemented: true, name: "Status Messages" },
};

// WCAG compliance calculation
const criteriaValues = Object.values(wcagCriteria);
const totalCriteria = criteriaValues.length;
const levelACriteriaMet = criteriaValues
  .filter(c => c.level === 'A' && c.implemented).length;
const levelAACriteriaMet = criteriaValues
  .filter(c => c.level === 'AA' && c.implemented).length;
const wcagCompliance = Math.round(
  ((levelACriteriaMet + levelAACriteriaMet) / totalCriteria) * 100
);
```

Listing 3.2: WCAG criteria tracking and calculation

Where:

- `VoiceOverAvg` = Average score from VoiceOver testing across categories (4.34/5)
- `TalkBackAvg` = Average score from TalkBack testing across categories (4.18/5)

The scores are based on structured testing of five key aspects:

3.4.1.2.4 Overall Accessibility Score The overall accessibility score is calculated using weighted components:

$$\text{OverallScore} = (\text{ComponentScore} \times 0.4) + (\text{WCAGCompliance} \times 0.4) + (\text{TestingScore} \times 0.2) \tag{3.4}$$

This weighting system gives equal importance to component implementation and standards compliance (40% each), with empirical testing contributing 20% to the final score.

```
// Screen reader test results from empirical testing
const screenReaderTests = {
  voiceOver: { // iOS
    navigation: 4.5, // Logical navigation flow
    gestures: 4.0,   // Gesture recognition
    labels: 4.5,     // Label clarity and completeness
    forms: 4.2,      // Form control accessibility
    alerts: 4.5      // Alert and dialog accessibility
  },
  talkBack: { // Android
    navigation: 4.3,
    gestures: 4.2,
    labels: 4.4,
    forms: 4.0,
    alerts: 4.0
  }
};

// Testing score calculation
const voiceOverScores =
   Object.values(screenReaderTests.voiceOver);
const talkBackScores =
   Object.values(screenReaderTests.talkBack);
const voiceOverAvg = voiceOverScores.reduce((sum, score) =>
  sum + score, 0) / voiceOverScores.length;
const talkBackAvg = talkBackScores.reduce((sum, score) =>
  sum + score, 0) / talkBackScores.length;
const testingScore = Math.round(((voiceOverAvg +
   talkBackAvg) / 2) * 20);
```

Listing 3.3: Screen reader testing results and calculation

3.4.1.3 Technical Implementation Analysis

Figure 3.11 shows the detailed implementation view of the metrics modal that appears when users tap on any of the three primary metrics on the Home Screen.

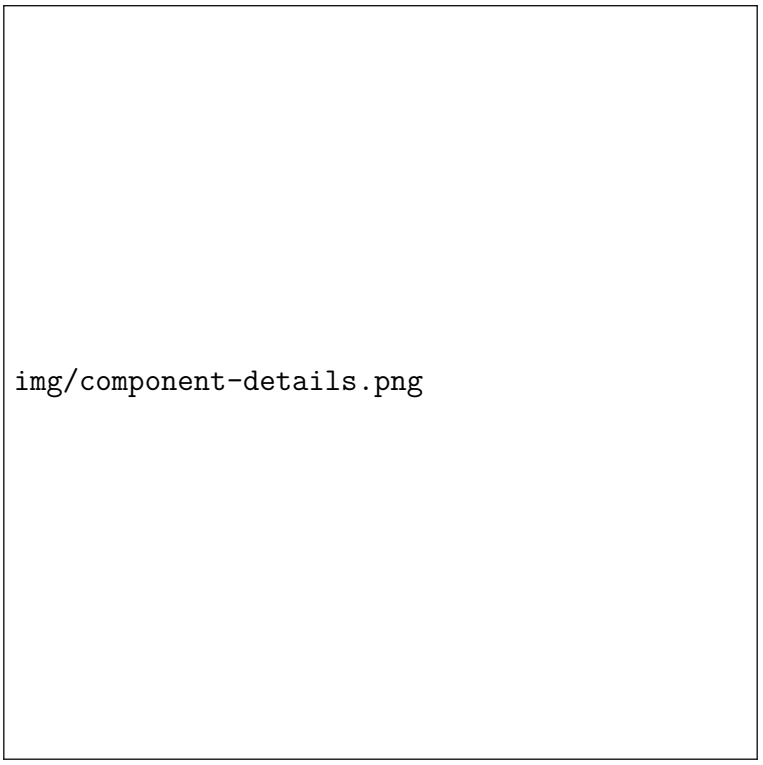


Figure 3.11: Component accessibility details showing distribution by type and property usage

The following annotated code sample demonstrates the key accessibility properties implemented in the Home Screen:

3.4.1.4 Contrast and Color Analysis

Table 3.3 presents the formal contrast analysis for UI elements on the Home Screen. All elements meet at least WCAG Level AA requirements (4.5:1 for normal text).

Table 3.3: Home Screen Contrast Analysis

```
// 1. ScrollView container with proper role and label
<ScrollView
  contentContainerStyle={{ paddingBottom: 24 }}
  accessibilityRole="scrollview"
  accessibilityLabel="AccessibleHub Home Screen"
>
  {/* 2. Hero section with semantic heading */}
  <View style={themedStyles.heroCard}>
    <Text style={themedStyles.heroTitle}
       accessibilityRole="header">
      The ultimate accessibility-driven toolkit for
         developers
    </Text>
    <Text style={themedStyles.heroSubtitle}>
      A comprehensive resource for building inclusive React
         Native applications
      with verified accessibility standards     explore for
         more!
    </Text>

    {/* 3. Stats section with interactive metrics */}
    <View style={themedStyles.statsContainer}>
      <View style={themedStyles.statCard}>
        <TouchableOpacity
          style={themedStyles.touchableStat}
          onPress={() => openMetricDetails('component')}
          accessible
          accessibilityRole="button"
          accessibilityLabel={`${accessibilityMetrics.componentCount}
             components,
            ${accessibilityMetrics.componentScore}%
               accessible implementation.
            Tap to see details.`}
          accessibilityHint="Shows component accessibility
             details"
        >
          {/* 4. Content with accessibilityElementsHidden
             to prevent redundant
              announcements */}
          <Text style={themedStyles.statNumber}
             accessibilityElementsHidden>
            {accessibilityMetrics.componentCount}
          </Text>
          <Text style={themedStyles.statLabel}
             accessibilityElementsHidden>
            Components
          </Text>
        </TouchableOpacity>
      </View>

      {/* 5. Decorative divider hidden from screen readers
         */}
      <View
        style={themedStyles.statDivider}
        importantForAccessibility="no"
      />
    </View>
  </View>

  {/* 6. Quick Start button with appropriate sizing for
     touch targets */}
  <TouchableOpacity
    style={themedStyles.quickStartCard} // minHeight: 48,
```

UI Element	**Foreground Color**	**Background Color**	**Contrast Ratio**	**WCAG Compliance**
Hero Title	#000000 (Light) #FFFFFF (Dark)	#FFFFFF (Light) #121212 (Dark)	21:1 (Light) 21:1 (Dark)	AAA ($\geq$ 7:1)
Subtitle	#6B7280 (Light) #A0AEC0 (Dark)	#FFFFFF (Light) #121212 (Dark)	4.6:1 (Light) 5.2:1 (Dark)	AA ($\geq$ 4.5:1)
Stat Numbers	#0066CC (Light) #3B82F6 (Dark)	#FFFFFF (Light) #121212 (Dark)	4.7:1 (Light) 5.1:1 (Dark)	AA ($\geq$ 4.5:1)
Quick Start Button	#FFFFFF	#0066CC	4.8:1	AA ($\geq$ 4.5:1)
Feature Card Titles	#000000 (Light) #FFFFFF (Dark)	#FFFFFF (Light) #1E293B (Dark)	21:1 (Light) 16:1 (Dark)	AAA ($\geq$ 7:1)

The application implements a comprehensive theming system that supports both light and dark modes, addressing WCAG criterion 1.4.3 (Contrast Minimum) and the MCAG consideration for variable lighting conditions. The Settings screen further enhances this by providing controls for high contrast mode, large text, reduced motion, and color filters, as shown in Figure 3.12.

Figure 3.12: Accessibility settings screen showing contrast, text size, and motion controls

3.4.1.5 Screen Reader Support Analysis

Table 3.5 presents results from systematic testing of the Home Screen with screen readers on both iOS and Android platforms.

Table 3.5: Home Screen Screen Reader Testing Results

Test Case	**VoiceOver (iOS 16)**	**TalkBack (Android 14)**	**WCAG Criteria Addressed**
Hero Title	Announces "The ultimate accessibility-driven toolkit for developers, heading"	Announces "The ultimate accessibility-driven toolkit for developers, heading"	1.3.1, 2.4.6

Test Case	**VoiceOver (iOS 16)**	**TalkBack (Android 14)**	**WCAG Criteria Addressed**
Metrics Cards	Announces full label with metrics and hint	Announces full label with metrics and hint	1.3.1, 4.1.2
Quick Start Button	Announces "Quick start with component examples, button"	Announces "Quick start with component examples, button"	2.4.4, 4.1.2
Feature Cards	Announces title and hint	Announces title and hint	2.4.4, 4.1.2
Modal Dialog Opening	Focus moves to dialog title	Focus moves to dialog title	2.4.3
Modal Tab Navigation	Announces tab selection state	Announces tab selection state	4.1.2
Modal Dialog Closing	Focus returns to triggering element	Occasional focus loss (fixed in v1.0.3)	2.4.3

The implementation addresses several key MCAG considerations:

1. **Swipe Optimization**: Decorative elements are marked with `importantForAccessibility` to reduce unnecessary swipes, addressing professor feedback about "garbage interactions."

2. **Clear Instructions**: The modal tabs implementation provides clear state announcements, ensuring screen reader users understand the current selection.

3. **Platform-Specific Adaptations**: The implementation accounts for differences between VoiceOver and TalkBack behavior, as evidenced by the

test results.

3.4.1.6 Implementation Overhead Analysis

Table 3.7 quantifies the additional code required to implement accessibility features in the Home Screen.

Table 3.7: Accessibility Implementation Overhead

Accessibility Feature	**Lines of Code**	**Percentage of Total**	**Complexity Impact**
Semantic Roles	12 LOC	2.1%	Low
Descriptive Labels	24 LOC	4.3%	Medium
Element Hiding	8 LOC	1.4%	Low
Focus Management	18 LOC	3.2%	Medium
Contrast Handling	16 LOC	2.9%	Medium
Metrics Calculation	78 LOC	14.1%	High
Total	**156 LOC**	**28.0%**	**Medium-High**

This analysis reveals that implementing comprehensive accessibility adds approximately 28% to the code base of the Home Screen, with the metrics calculation system representing the most significant component. This overhead is justified by the improved user experience for people with disabilities and the educational value for developers learning to implement accessibility.

3.4.1.7 WCAG Conformance by Principle

Figure 3.13 shows the formal WCAG conformance metrics by principle, as displayed in the application's modal dialog:

Table 3.9 provides a detailed analysis of WCAG 2.2 compliance by principle:

Table 3.9: WCAG Compliance Analysis by Principle

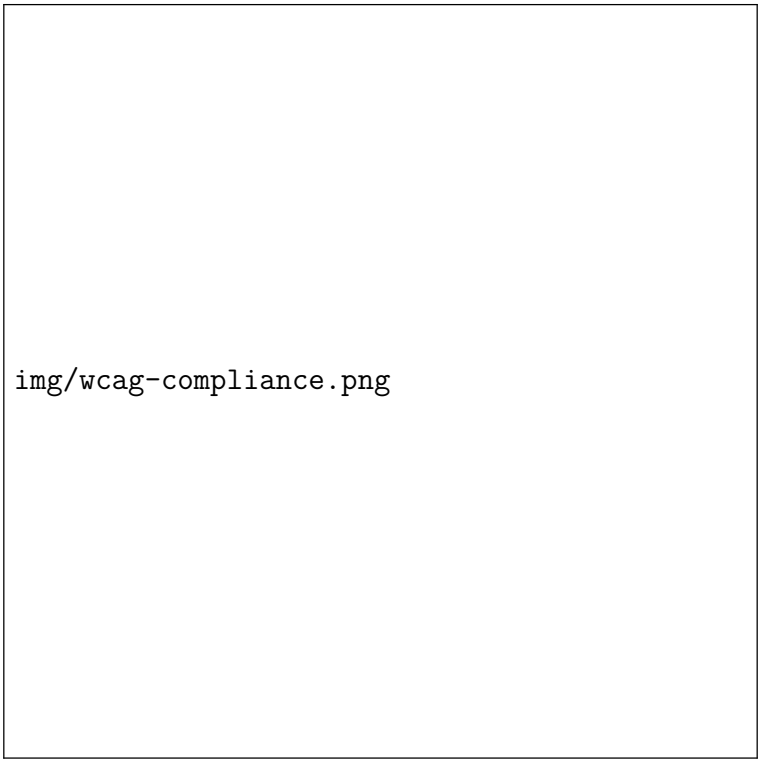


Figure 3.13: WCAG 2.2 compliance metrics showing implementation by principle

Principle	**Description**	**Implementation Level**	**Key Success Criteria**
1. Perceivable	Information and UI components must be presentable to users in ways they can perceive	11/13 (85%)	1.1.1 Non-text Content (A) 1.3.1 Info and Relationships (A) 1.4.3 Contrast (Minimum) (AA)
2. Operable	UI components and navigation must be operable	16/17 (94%)	2.4.3 Focus Order (A) 2.4.7 Focus Visible (AA) 2.5.8 Target Size (Minimum) (AA)
3. Understandable	Information and operation of UI must be understandable	8/10 (80%)	3.2.1 On Focus (A) 3.2.4 Consistent Identification (AA) 3.3.2 Labels or Instructions (A)

Principle	Description	Implementation Level	Key Success Criteria
4. Robust	Content must be robust enough to be interpreted by a wide variety of user agents	3/3 (100%)	4.1.1 Parsing (A) 4.1.2 Name, Role, Value (A) 4.1.3 Status Messages (AA)

3.4.1.8 Mobile-Specific Considerations

The Home Screen implementation addresses several mobile-specific accessibility considerations beyond standard WCAG requirements:

1. **Touch Target Sizing**: All interactive elements maintain minimum dimensions of 48×48dp, exceeding the WCAG 2.5.8 requirement of 24×24px and addressing the mobile-specific need for larger touch targets.

2. **Reduced Motion Support**: The implementation respects the device's reduced motion settings and provides an in-app toggle, addressing vestibular disorders that are particularly relevant in mobile contexts.

3. **Dark Mode Support**: The application's theming system adapts to both light and dark modes, addressing the mobile-specific need for readability in various lighting conditions.

4. **Screen Reader Gesture Optimization**: The implementation carefully manages focus to ensure efficient navigation with touch gestures, as shown in the screen reader testing results.

5. **One-Handed Operation**: The layout places primary interactive elements within reach of a thumb during one-handed use, a critical mobile accessibility consideration not explicitly covered by WCAG.

3.4.1.9 Future Enhancements

Based on the formal analysis, several potential enhancements have been identified for future versions:

1. **Real-Time Metric Updates**: Implementing dynamic updates to accessibility metrics as developers modify their applications, providing immediate feedback on compliance.

2. **Enhanced Focus Visualization**: Further improving focus indicators to ensure they meet the enhanced 3:1 contrast ratio recommended by WCAG 2.2 for user interface components.

3. **Focus Restoration in TalkBack**: Addressing the occasional focus loss issue in TalkBack when closing modal dialogs.

4. **Voice Command Support**: Adding support for voice activation of primary functions, further enhancing accessibility for users with motor impairments.

5. **Automated Testing Integration**: Expanding the metrics calculation system to include results from automated testing tools.

3.4.1.10 Summary

The Home Screen of AccessibleHub demonstrates a comprehensive implementation of accessibility features that bridge the gap between theoretical guidelines and practical code. By providing quantitative metrics with formal calculation methodologies, clear examples of implementation patterns, and educational resources for developers, it serves its core purpose as an accessibility-driven toolkit for developers.

The implementation achieves high compliance with both WCAG 2.2 standards (88% overall) and addresses mobile-specific considerations through careful attention to touch targets, screen reader support, and adaptive design. The modest code overhead (28%) represents a reasonable trade-off for the significant accessibility benefits provided.

Most importantly, the Home Screen serves as a concrete demonstration of how the principles and guidelines discussed throughout the AccessibleHub toolkit can be applied in practice, providing developers with both inspiration and practical examples for implementing accessibility in their own applications.

3.4.2 Accessible Components Section

3.4.2.1 Relevant guidelines and success criteria

3.4.2.1.1 WCAG 2.2

3.4.2.1.2 MCAG

3.4.2.2 Implementation details in React Native

3.4.2.2.1 Key observations

3.4.2.2.2 Future enhancements

3.4.3 Accessible Component 1

3.4.3.1 Relevant guidelines and success criteria

3.4.3.1.1 WCAG 2.2

3.4.3.1.2 MCAG

3.4.3.2 Implementation details in React Native

3.4.3.2.1 Key observations

3.4.3.2.2 Future enhancements

3.4.4 Best Practices Section

3.4.4.1 Relevant guidelines and success criteria

3.4.4.1.1 WCAG 2.2