Components Accessible Code

JSX

Copy

```
<TouchableOpacity
  accessibilityRole="button"
  accessibilityLabel="Submit form"
  accessibilityHint="Activates form
submission"
  style={{
    minHeight: 44,
    paddingHorizontal: 16,
    backgroundColor: '#007AFF',
    borderRadius: 8,
    justifyContent: 'center',
    alignItems: 'center',
  }}
>
  <Text style={{ color: '#fff' }}>
    Submit
  </Text>
</TouchableOpacity>
```

Accessibility Features

Minimum Touch Target

44x44 points minimum size ensures the button is easy to tap

Screen Reader Label

Clear description announces the button's purpose

Action Hint

Additional context about what happens on activation

Interactive Example

Submit

Try this button with VoiceOver/TalkBack enabled

Implementation

JSX

Copy

```
<TouchableOpacity
  accessibilityRole="button"
  accessibilityLabel="Submit form"
  accessibilityHint="Activates form
submission"
  style={{
    minHeight: 44,
    paddingHorizontal: 16,
    backgroundColor: '#007AFF',
    borderRadius: 8,
    justifyContent: 'center',
    alignItems: 'center',
  }}
>
  <Text style={{ color: '#fff' }}>
    Submit
  </Text>
</TouchableOpacity>
```

Accessibility Features

Components Accessible Code

Accessibility Components

Interactive examples of accessible React Native components with code samples and best practices.

Buttons & Touchables

Essential

Create accessible touch targets with proper sizing and feedback

Touch target sizing Haptic feedback

Form Controls

Complex

Implement accessible form inputs and controls

Error states Helper text

Media Content

Advanced

Make images and media content accessible

Aa Alt text Media controls

Modal Dialogs

Advanced

Implement accessible modal dialogs with proper focus management and screen reader support

Focus trapping Screen reader alerts

Interactive Example

Submit

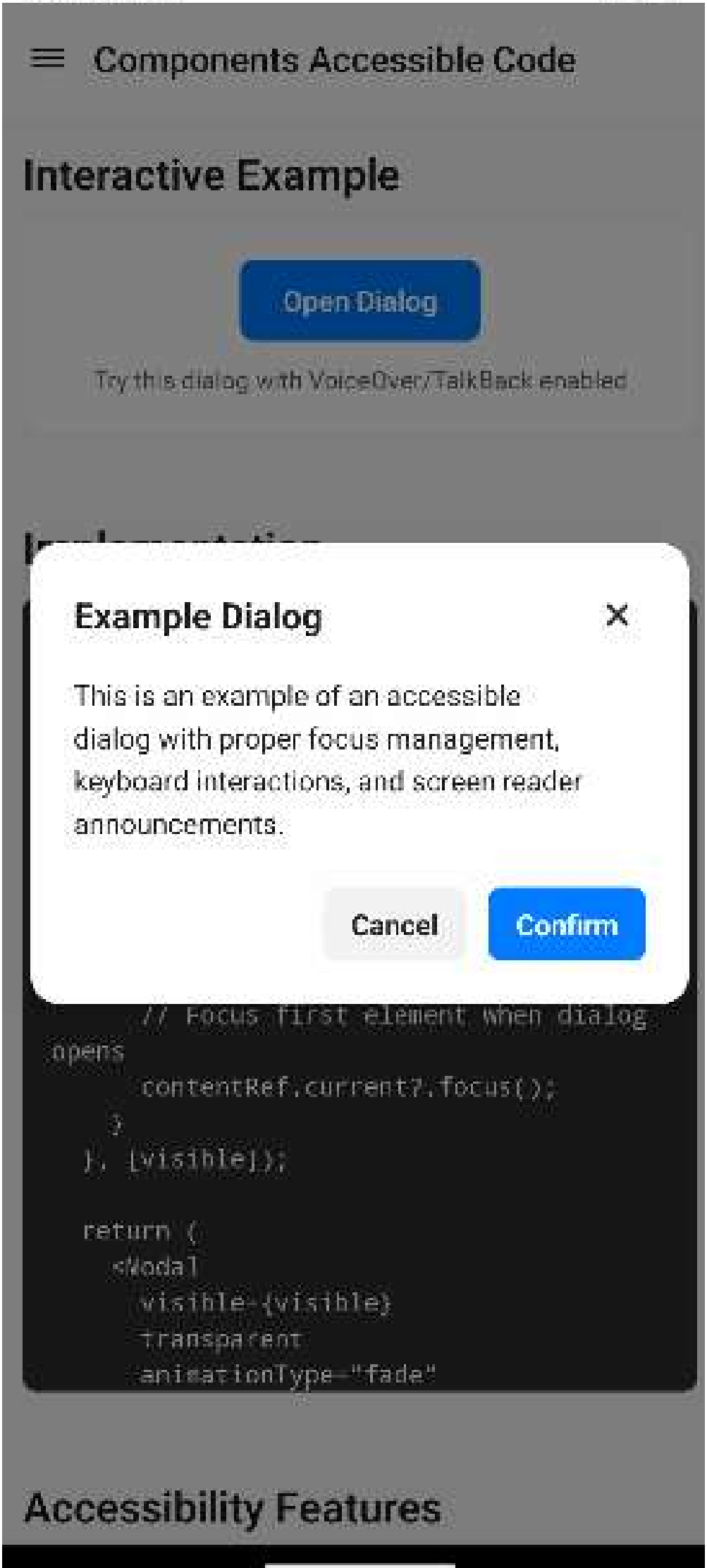
Try this button with VoiceOver/TalkBack enabled

Implementation

JSX ✓ Copied!

```
<TouchableOpacity
  accessibilityRole="button"
  accessibilityLabel="Submit form"
  accessibilityHint="Activates form
submission"
  style={{
    minHeight: 44,
    paddingHorizontal: 16,
    backgroundColor: '#007AFF',
    borderRadius: 8,
    justifyContent: 'center',
    alignItems: 'center',
  }}
>
  <Text style={{ color: '#fff' }}>
    Submit
  </Text>
</TouchableOpacity>
```

<TouchableOp
acity
accessibility
Role="button"
accessibilit...


Components Accessible Code
Interactive Example
Open Dialog
Try this dialog with VoiceOver/TalkBack enabled
Example Dialog
This is an example of an accessible dialog with proper focus management, keyboard interactions, and screen reader announcements.
Cancel
Confirm
// Focus first element when dialog opens
contentRef.current?.focus();
}, [visible]);
return (
<Modal
visible={visible}
transparent
animationType="fade"
Accessibility Features

Components Accessible Code

```
// Accessible Dialog Implementation
const AccessibleDialog = ({ visible,
onClose, title, children }) => {
  const closeRef = useRef(null);
  const contentRef = useRef(null);

  useEffect(() => {
    if (visible) {
      // Focus first element when dialog
opens
      contentRef.current?.focus();
    }
  }, [visible]);

  return (
    <Modal
      visible={visible}
      transparent
      animationType="fade"
```

Accessibility Features

Focus Management

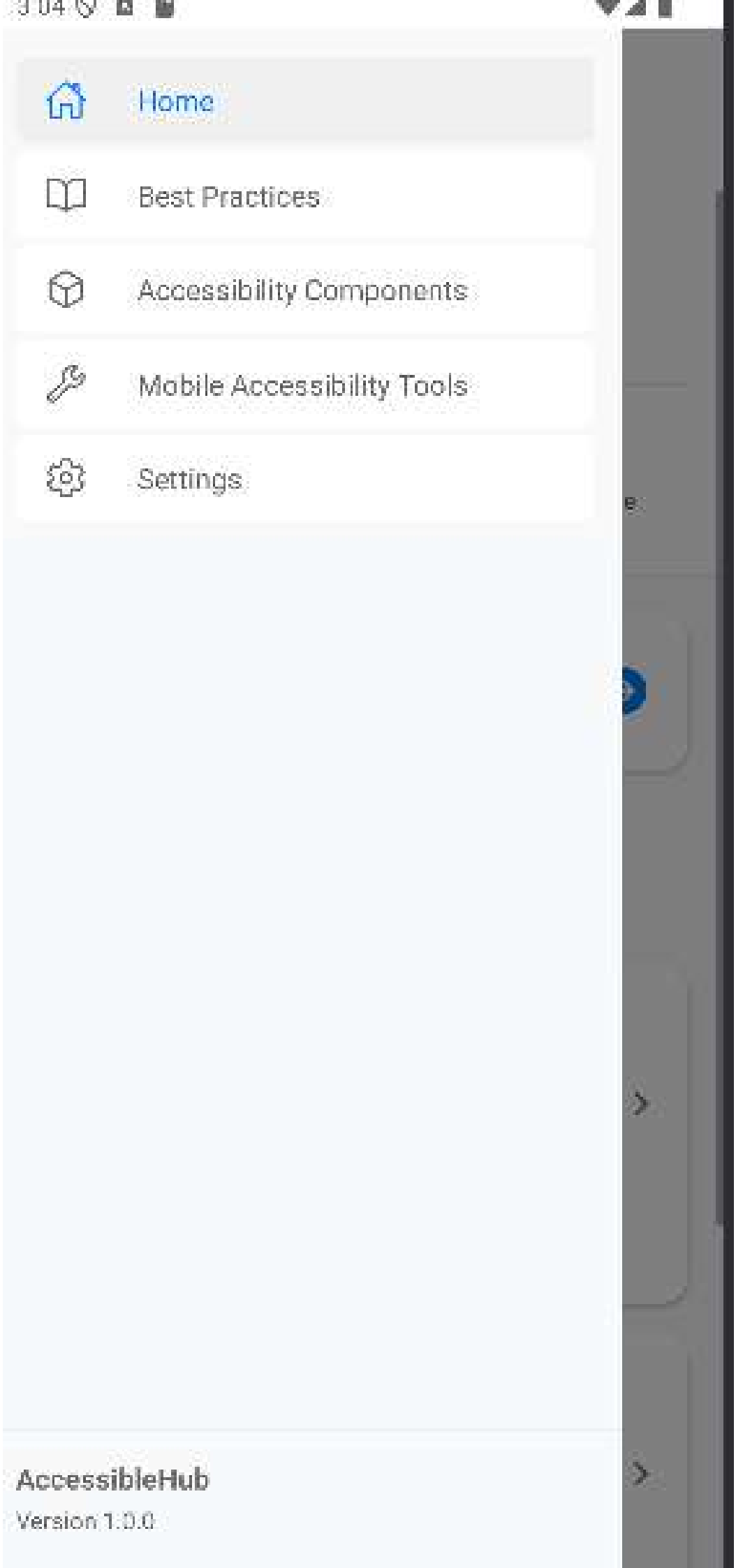
Proper focus trapping and restoration when dialog opens/closes

Keyboard Navigation

Full keyboard support including escape key to close

Screen Reader Support

Proper ARIA roles and live region announcements


Home
Best Practices
Accessibility Components
Mobile Accessibility Tools
Settings
AccessibleHub
Version 1.0.0

An accessibility testing manual for developers

A developer's guide to creating inclusive applications

15+	WCAG	100%
Components	2.2 Ready	Accessible

Quick Start

Begin with component examples

Explore topics

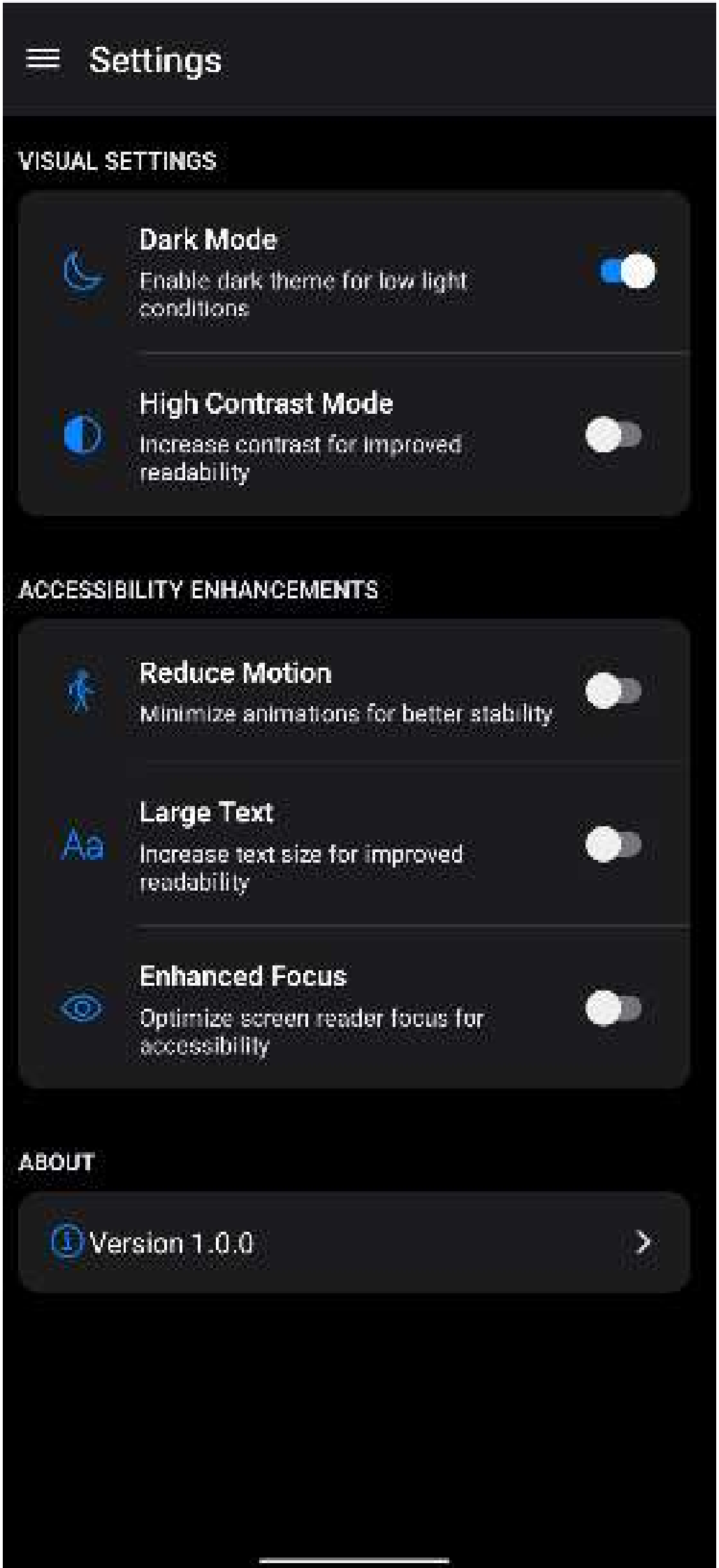
Best Practices

Learn how to implement WCAG guidelines in React Native

WCAG 2.2 Guidelines

Testing Tools

tools and methods to verify your app's accessibility


Settings
VISUAL SETTINGS
Dark Mode
Enable dark theme for low light conditions
High Contrast Mode
Increase contrast for improved readability
ACCESSIBILITY ENHANCEMENTS
Reduce Motion
Minimize animations for better stability
Large Text
Increase text size for improved readability
Enhanced Focus
Optimize screen reader focus for accessibility
ABOUT
Version 1.0.0

Implementation

JSX Copy

```
<View accessibilityRole="form">
  <Text style={styles.label}>Name</Text>
  <TextInput
    style={styles.input}
    accessibilityLabel="Enter your name"
    value={formData.name}
    onChangeText={(text) =>
setFormData(prev => ({ ...prev, name:
text }))}
  />

  <Text style={styles.label}>Email</Text>
  <TextInput
    style={styles.input}
    accessibilityLabel="Enter your email"
    value={formData.email}
```

Accessibility Features

Input Labels

Clear, descriptive labels that properly associate with form controls

Semantic Roles

Proper role assignments for form controls (radio, checkbox, button)

Error States

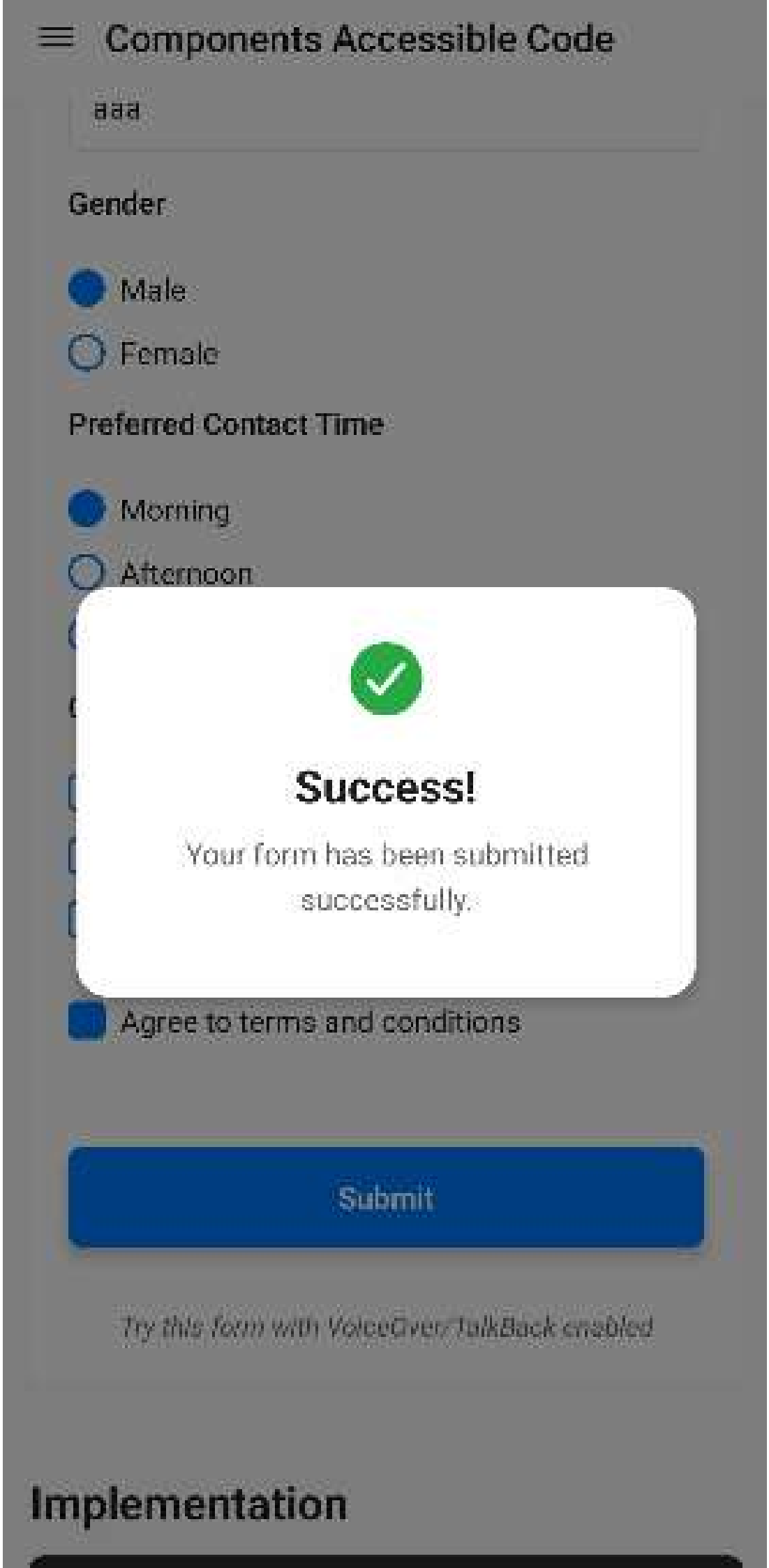
Clear error messages and validation feedback for screen readers

Touch Targets

Adequate sizing for interactive elements (minimum 44x44 points)

State Management

Proper state announcements for selection controls and submit button


Components Accessible Code
aad
Gender
Male
Female
Preferred Contact Time
Morning
Afternoon
Success!
Your form has been submitted successfully.
Agree to terms and conditions
Submit
Try this form with VoiceOver/TalkBack enabled
Implementation

Accessibility Gestures

Learn and practice the most common and reliable accessibility gestures

Single Tap

Select and announce items. Most basic and reliable gesture.

Double Tap

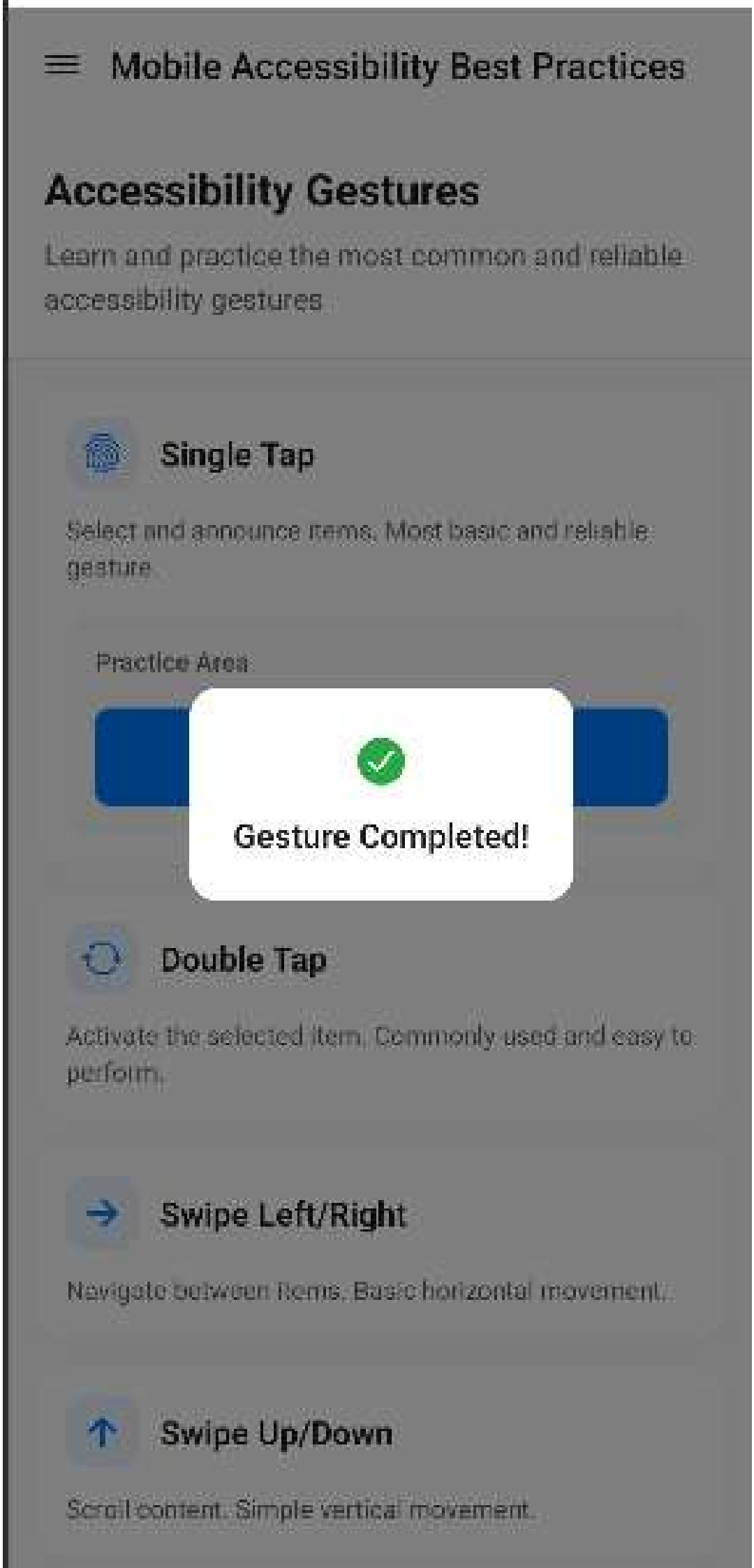
Activate the selected item. Commonly used and easy to perform.

Swipe Left/Right

Navigate between items. Basic horizontal movement.

Swipe Up/Down

Scroll content. Simple vertical movement.


Mobile Accessibility Best Practices
Accessibility Gestures
Learn and practice the most common and reliable accessibility gestures
Single Tap
Select and announce items. Most basic and reliable gesture.
Practice Area
Gesture Completed!
Double Tap
Activate the selected item. Commonly used and easy to perform.
Swipe Left/Right
Navigate between items. Basic horizontal movement.
Swipe Up/Down
Scroll content. Simple vertical movement.

WCAG 2.2 Guidelines

Essential guidelines for mobile accessibility

Perceivable

Information must be presentable to users in ways they can perceive.

- Text alternatives for non-text content
- Sufficient color contrast ratios
- Clear content structure and relationships

Operable

Interface components must be operable by all users.

- All functionality available via keyboard
- Sufficient time to read and use content
- No content that could cause seizures

Understandable

Information and interface operation must be understandable.

- Readable and understandable text content
- Predictable functionality
- Input assistance and error prevention

An accessibility testing manual for developers

A developer's guide to creating inclusive applications

15+	WCAG	100%
Components	2.2 Ready	Accessible

Quick Start

Begin with component examples

Explore topics

Best Practices

Learn how to implement WCAG guidelines in React Native

WCAG 2.2 Guidelines

Testing Tools

Quick Start

Begin with component examples

Explore topics

Best Practices

Learn how to implement WCAG guidelines in React Native

WCAG 2.2 Guidelines

Testing Tools

Tools and methods to verify your app's accessibility

TalkBack VoiceOver

Framework Comparison

Compare accessibility features across different mobile frameworks

React Native Flutter

```
  accessibilityLabel="Detailed
description of the image content"
  accessible={true}
  accessibilityRole="image"
  style={{
    width: 300,
    height: 200,
    borderRadius: 8,
  }}
/>

{/* For network images */}
<Image
  source={{ uri: 'https://your-domain.com
/image.jpg' }}
  accessibilityLabel="Detailed
description of the image content"
  accessible={true}
  accessibilityRole="image"
  style={{
    width: 300,
    height: 200,
    borderRadius: 8,
  }}
/>
```

Accessibility Features

Alternative Text

Descriptive text that conveys the content and function of the image

Role Announcement

Screen readers announce the element as an image

Interactive Example

Hide Alt Text

Alternative Text:

A placeholder image (first example)

Role: Interface example

Try this image component with VoiceOver/TalkBack enabled

Implementation

JSX Copy

```
<Image
  source={require('./path/to/image.png')}
  accessibilityLabel="Detailed
description of the image content"
  accessible={true}
  accessibilityRole="image"
  style={
```

Navigation & Focus

Guidelines for implementing effective keyboard and focus navigation

Focus Flow

Managing the order and flow of focus navigation

- Logical tab order
- Clear focus indicators
- Skip navigation patterns

Focus Management

Handling focus during interface changes

- Modal and dialog focus
- Focus restoration
- Dynamic content updates

Keyboard Navigation

Supporting keyboard-only navigation

- Keyboard shortcuts
- Focus trapping
- Custom key handlers

Accessible Practices

Understanding and implementing WCAG 2.2 guidelines in mobile apps

Success Criteria Examples

Semantic Structure

Code Examples

Creating meaningful and well-organized content hierarchies

Hierarchy Implementation

Gesture Tutorial

Interactive Guide

Learn and test common accessibility gestures

Gesture Patterns Interactive Demo

Screen Reader Support

Guidelines

Optimizing your app for VoiceOver and TalkBack

Platform-specific Gestures

Navigation & Focus

Interactive Guide

Managing focus and keyboard navigation effectively

Focus Flow Interactive Demo

Accessible Practices

Mobile Accessibility Best Practices

Essential guidelines for creating accessible React Native applications

WCAG Guidelines

2.2 Documentation

Understanding and implementing WCAG 2.2 guidelines in mobile apps

Success Criteria Examples

Semantic Structure

Code Examples

Creating meaningful and well-organized content hierarchies

Hierarchy Implementation

Gesture Tutorial

Interactive Guide

Learn and test common accessibility gestures

Gesture Patterns Interactive Demo

Screen Reader Support

Guidelines

Mobile Accessibility Best Practices

Screen Reader Guide

Comprehensive guide for optimizing your app for VoiceOver and TalkBack

VoiceOver (iOS) | TalkBack (Android)

Essential Gestures

Single tap

Select an item

Double tap

Activate selected item

Three finger swipe up/down

Scroll content

Three finger tap

Speak current page

Two finger swipe up

Read from current position

Two finger twist

Screen Reader Guide

Comprehensive guide for optimizing your app for VoiceOver and TalkBack

VoiceOver (iOS) | TalkBack (Android)

Implementation Guide

Semantic Structure

- Use proper heading hierarchy
- Implement meaningful landmarks
- Group related elements logically

View Code Examples →

Content Descriptions

- Provide clear accessibilityLabels
- Include meaningful hints
- Describe state changes

View Guidelines →

Interactive Elements

- Define proper roles
- Manage focus appropriately
- Handle custom actions

Semantic Structure

Building meaningful and well-organized content hierarchies

Content Hierarchy

Proper headings and landmarks help users understand content organization

```
// Good Example
<View accessibilityRole="header">
  <Text accessibilityRole="heading">
    Main Title
  </Text>
</View>

<View accessibilityRole="main">
  <Text accessibilityRole="heading">
    Section Title
  </Text>
</View>
```

Navigation Order

Logical tab order that matches visual layout improves navigation.

- Use natural reading order
- Group related elements
- Maintain consistent structure

Mobile Accessibility Best Practices

```
// Good Example
<View accessibilityRole="header">
  <Text accessibilityRole="heading">
    Main Title
  </Text>
</View>

<View accessibilityRole="main">
  <Text accessibilityRole="heading">
    Section Title
  </Text>
</View>
```

Navigation Order

Logical tab order that matches visual layout improves navigation.

- Use natural reading order
- Group related elements
- Maintain consistent structure

Landmarks & Regions

Define distinct areas of content to aid navigation and comprehension.

- Mark main content areas
- Identify navigation sections
- Label complementary content

Settings

VISUAL SETTINGS

Dark Mode
Enable dark theme for low light conditions

High Contrast Mode
Increase contrast for improved readability

ACCESSIBILITY ENHANCEMENTS

Reduce Motion
Minimize animations for better stability

Large Text
Increase text size for improved readability

Enhanced Focus
Optimize screen reader focus for accessibility

ABOUT

Version 1.0.0

Mobile Accessibility Tools

Development Tools

Accessibility Inspector

Built-in tool to inspect accessibility properties:

- Verify accessibility labels and hints
- Check navigation order
- Test screen reader announcements

Contrast Analyzer

Verify color contrast ratios for WCAG guidelines:

- Check text contrast ratios
- Verify UI component contrast
- Support for WCAG 2.2 standards

Testing Checklist

Automated Testing

Essential checks for accessibility testing:

- Run accessibility linter
- Verify accessibility props
- Check navigation order
- Test color contrast

Testing Tools

Essential tools for testing accessibility in your mobile applications

Screen Readers

TalkBack (Android) Built-in

Android's built-in screen reader. Essential gestures:

- Single tap: Select item
- Double tap: Activate selected item
- Swipe right/left: Next/previous item

VoiceOver (iOS) Built-in

iOS's integrated screen reader. Key gestures:

- Single tap: Select and speak
- Double tap: Activate item
- Three finger swipe: Scroll

Development Tools

Accessibility Inspector

Built-in tool to inspect accessibility properties: