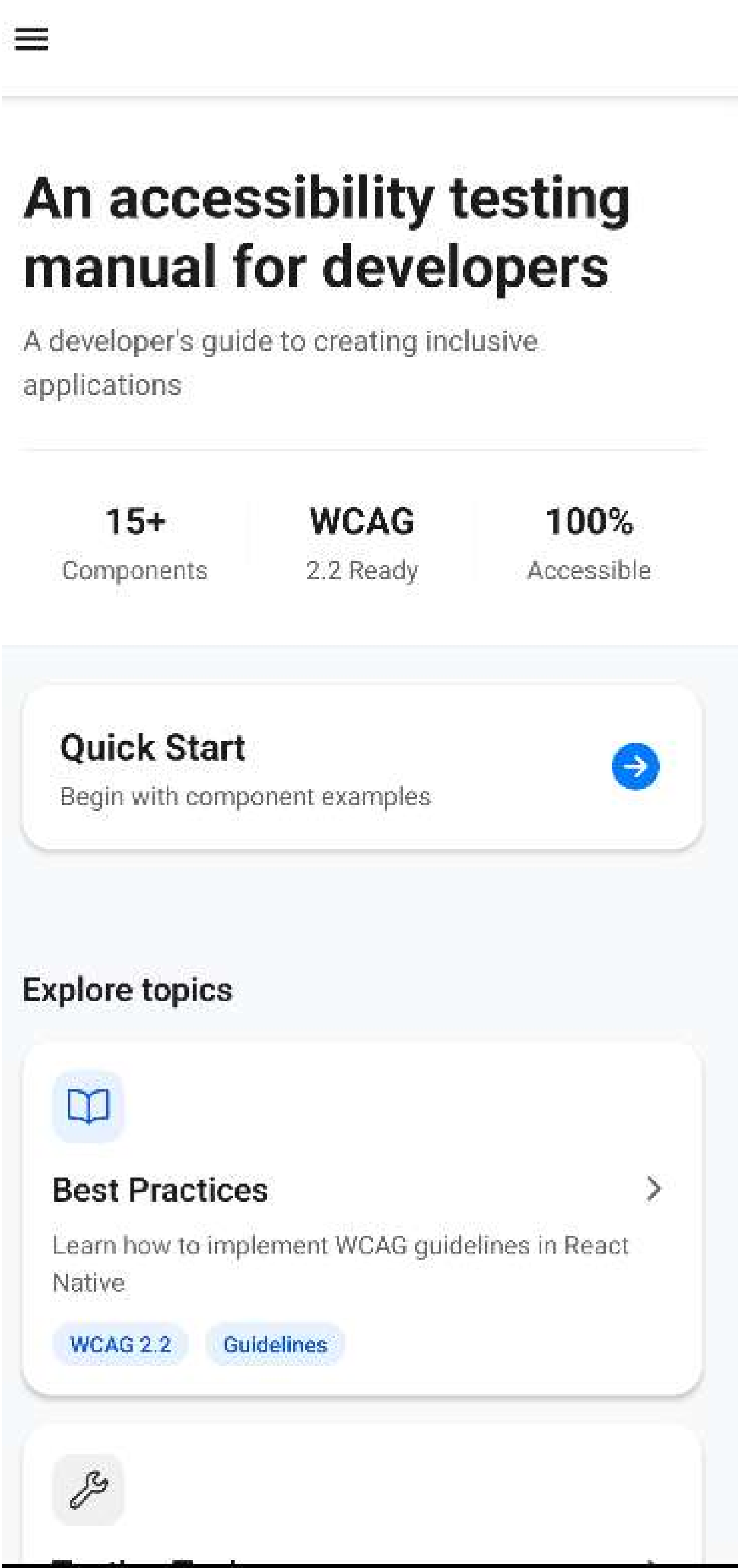


An accessibility testing manual for developers

A developer's guide to creating inclusive applications

15+	WCAG	100%
Components	2.2 Ready	Accessible

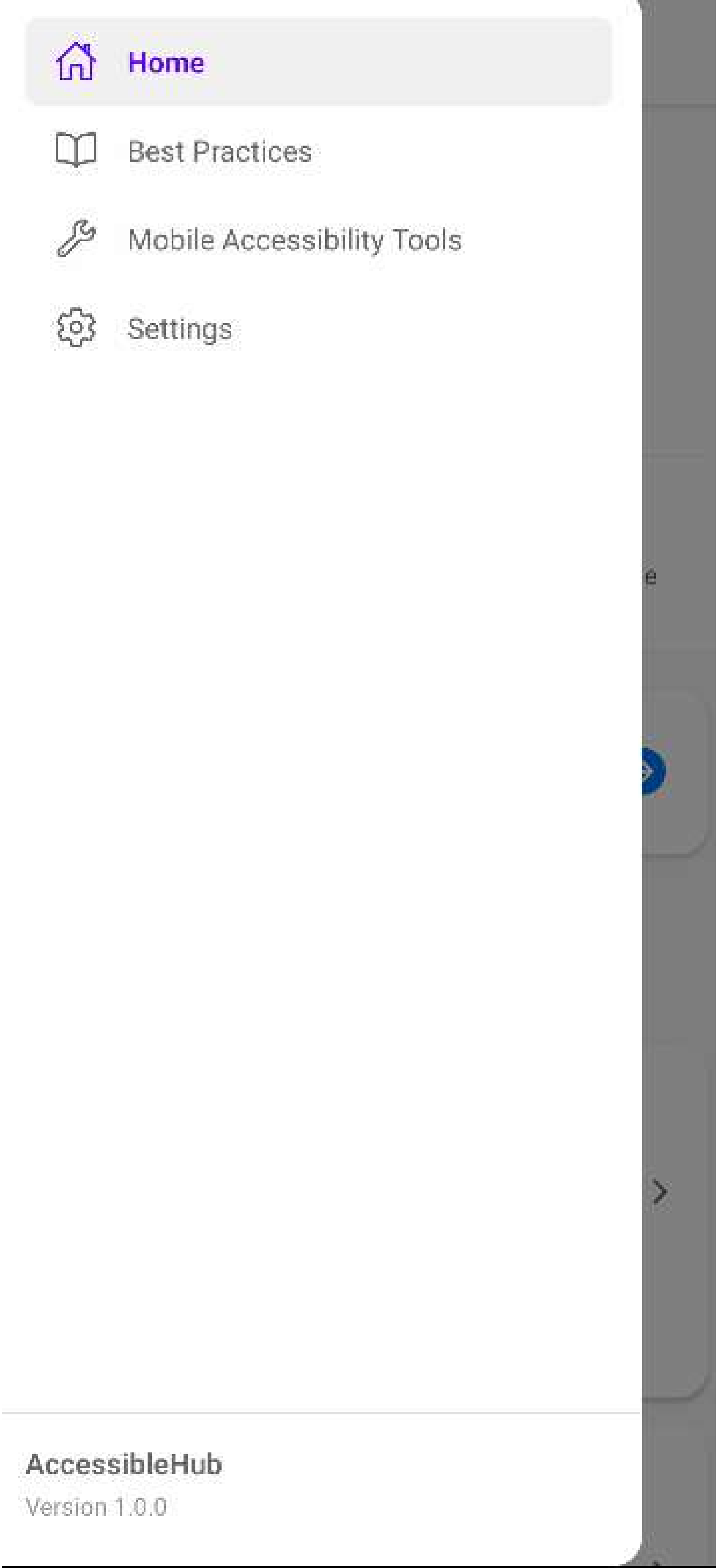
Quick Start

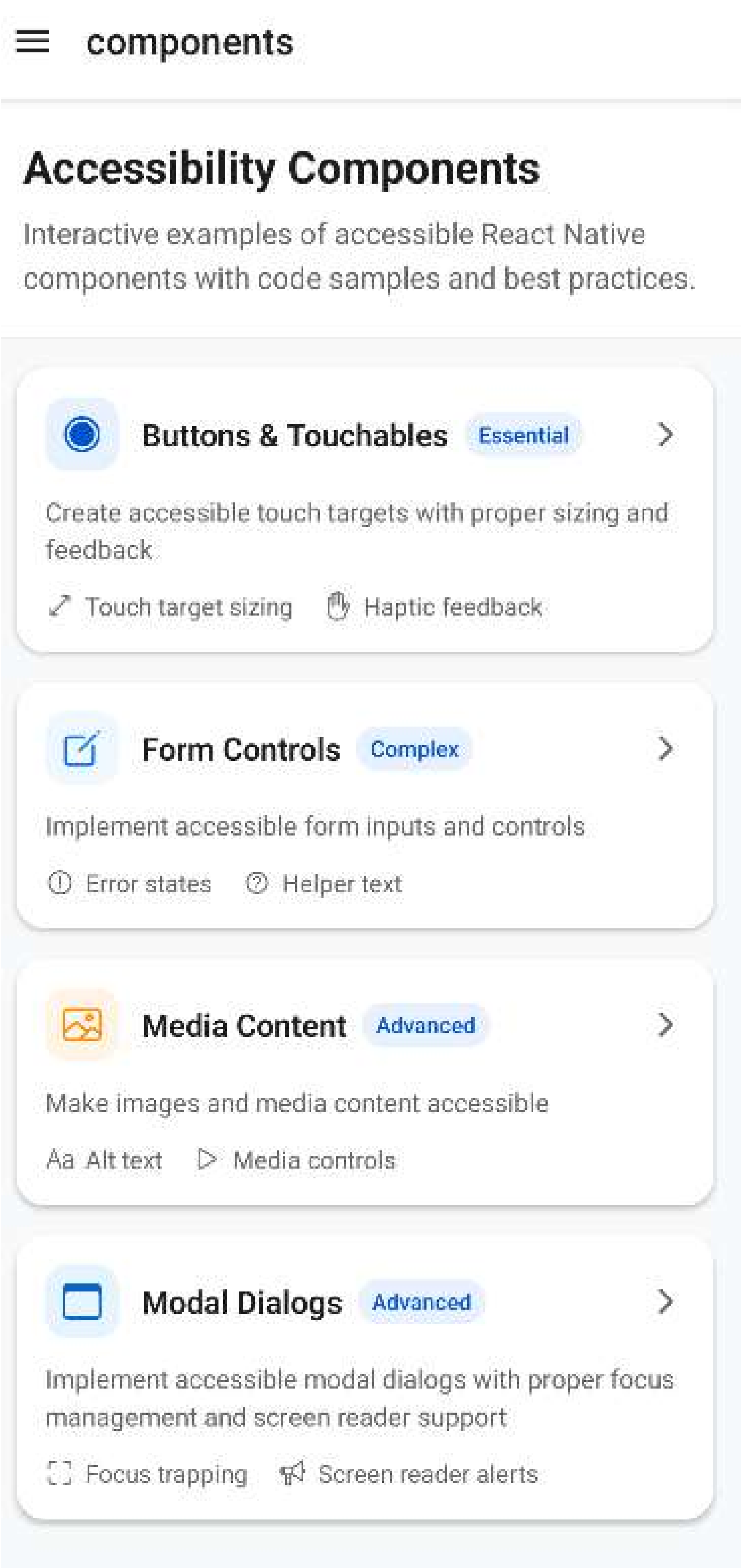
Begin with component examples

Explore topics

Best Practices

Learn how to implement WCAG guidelines in React Native


Home
Best Practices
Mobile Accessibility Tools
Settings
AccessibleHub
Version 1.0.0



☰ components

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

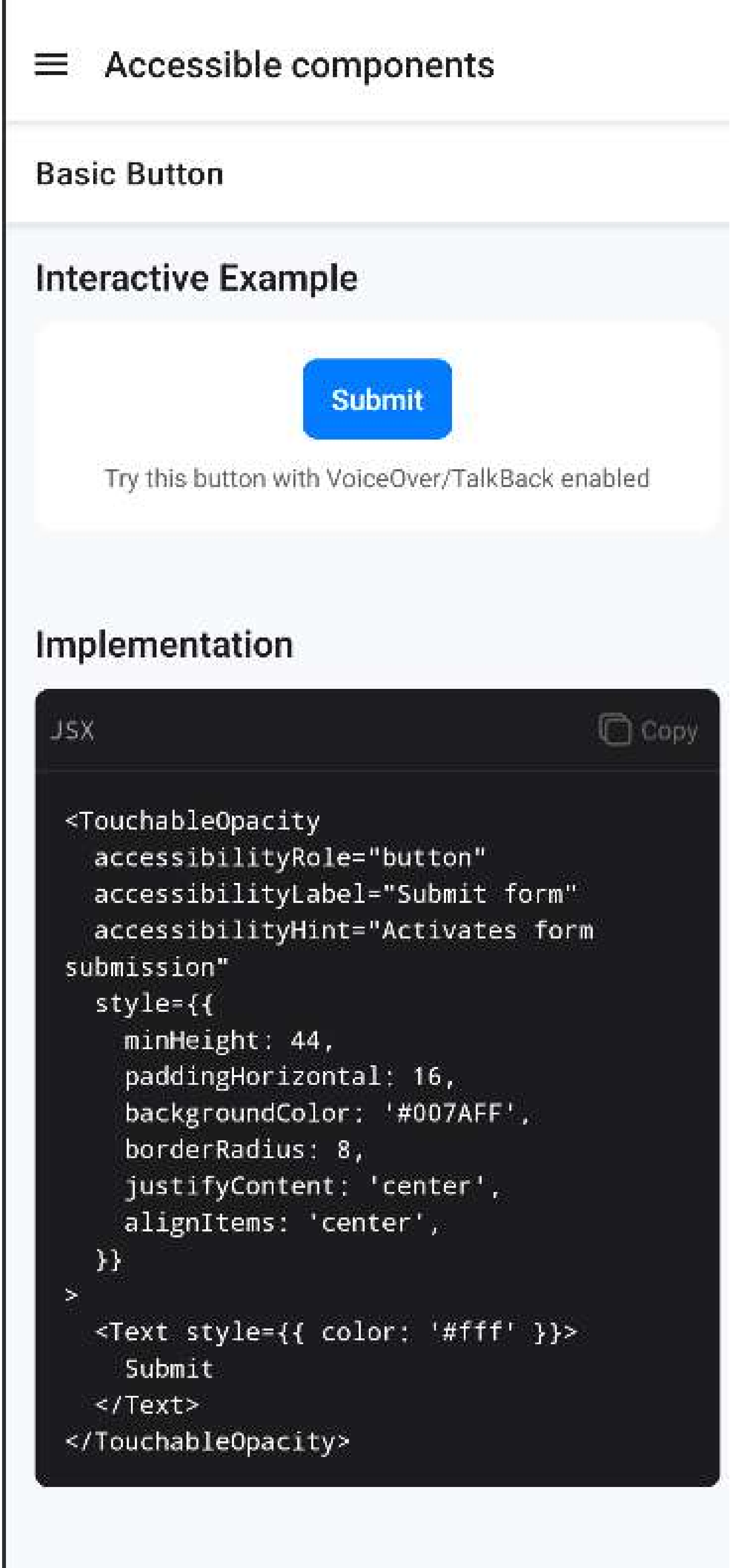
Aa Alt text   ▷ Media controls

Modal Dialogs

Advanced

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

⛶ Focus trapping   📢 Screen reader alerts



Accessible components

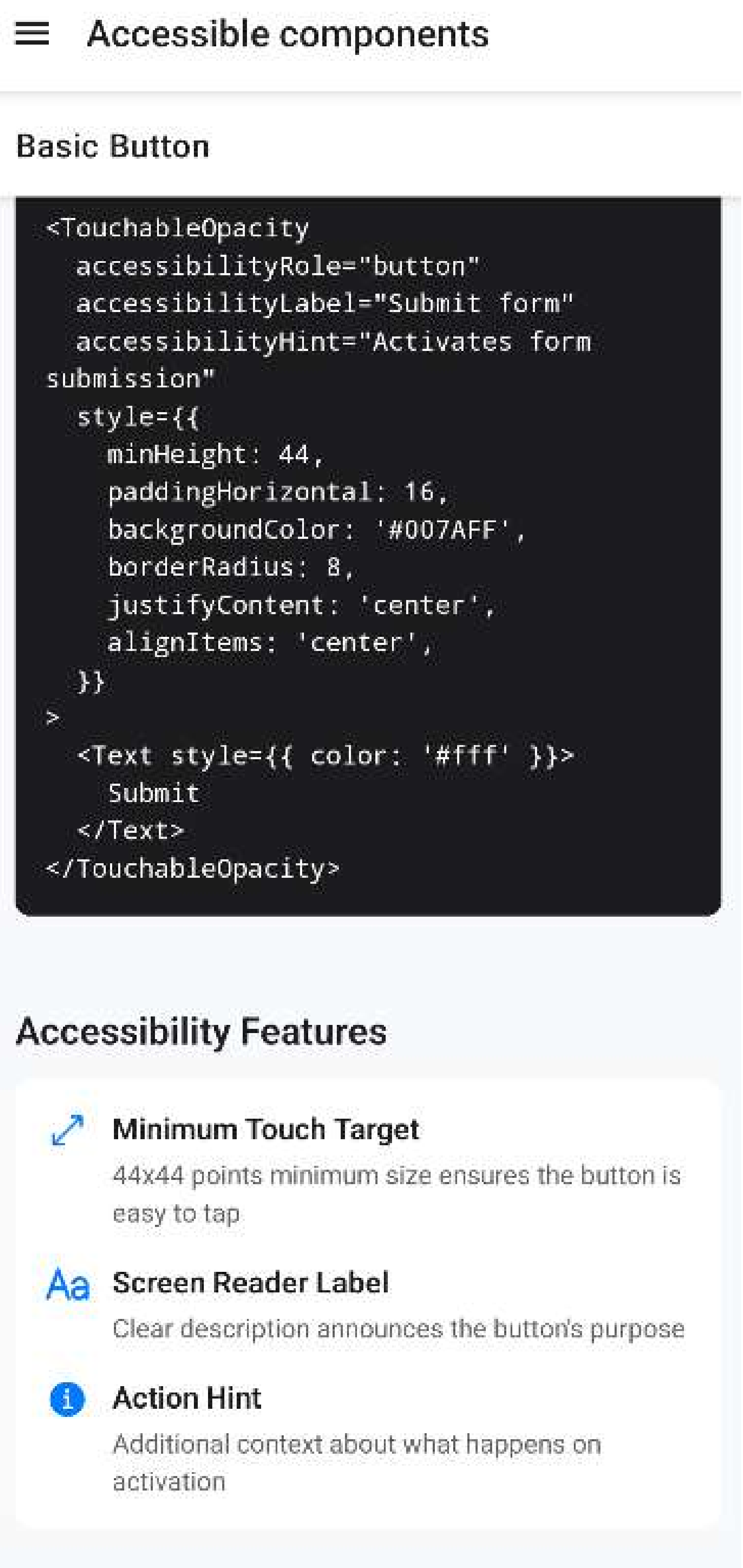
Basic Button

Interactive Example

Try this button with VoiceOver/TalkBack enabled

Implementation

```
<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>
```



☰ Accessible components

Basic Button

```
<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

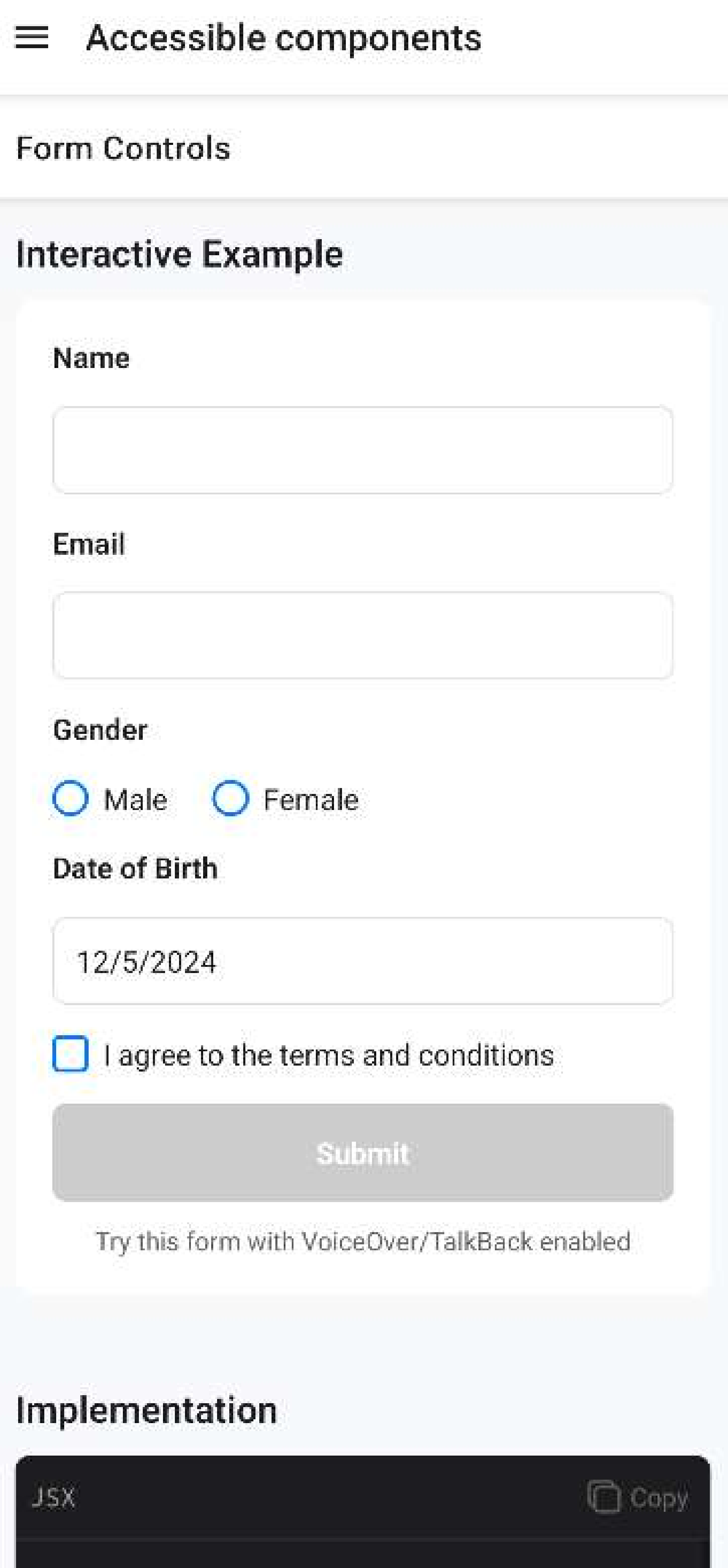
44x44 points minimum size ensures the button is easy to tap

Screen Reader Label

Clear description announces the button's purpose

Action Hint

Additional context about what happens on activation



☰ Accessible components

Form Controls

Interactive Example

Name

Email

Gender

○ Male ○ Female

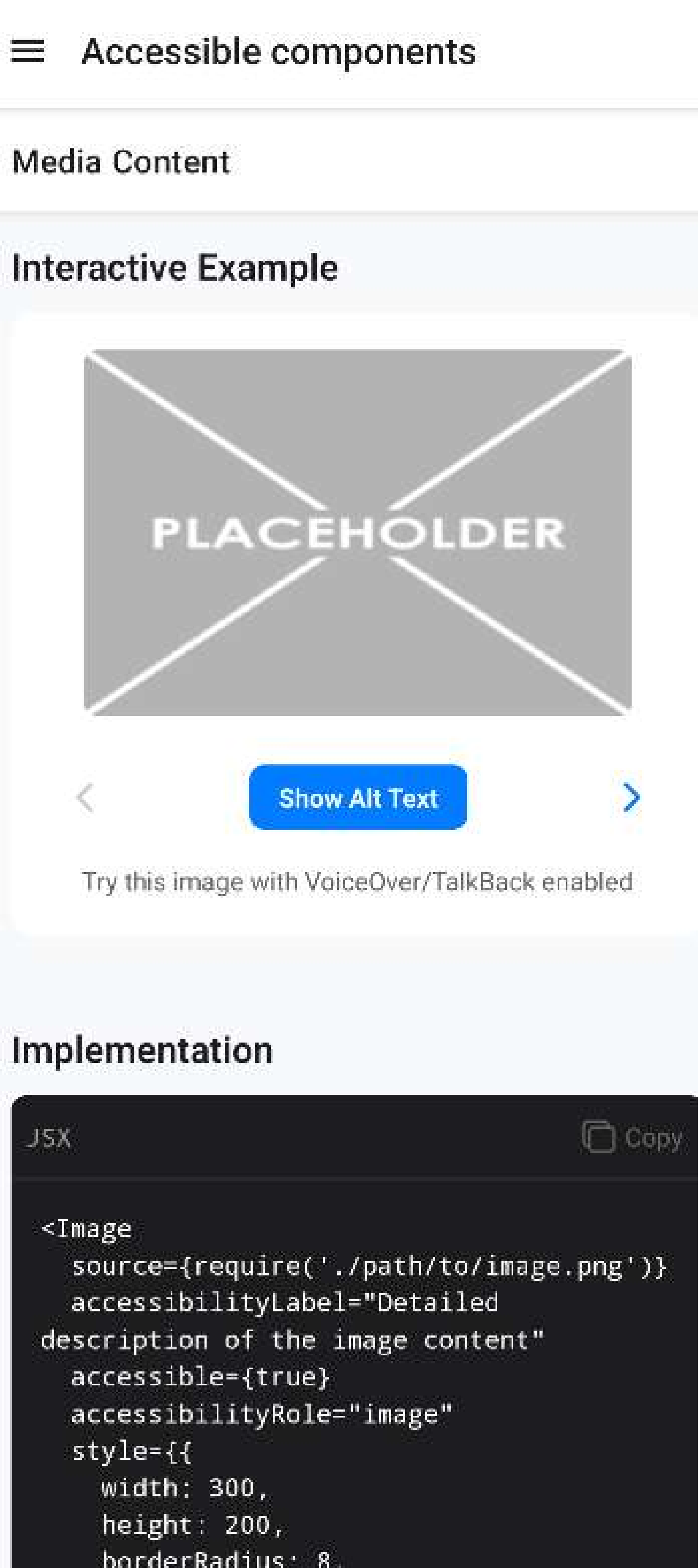
Date of Birth

12/5/2024

□ I agree to the terms and conditions

Try this form with VoiceOver/TalkBack enabled

Implementation



Accessible components

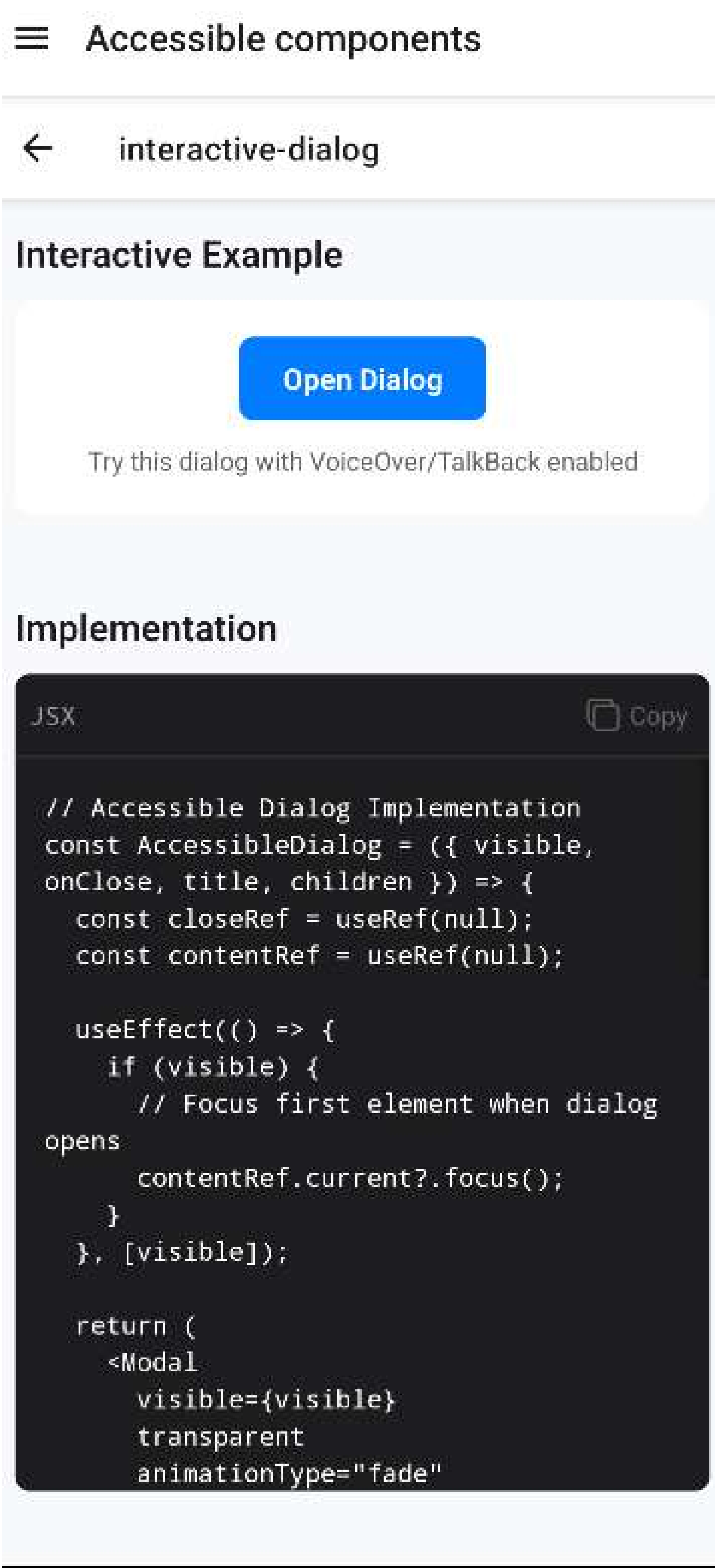
Media Content

Interactive Example

Try this image with VoiceOver/TalkBack enabled

Implementation

```jsx
<Image
  source={require('./path/to/image.png')}
  accessibilityLabel="Detailed
description of the image content"
  accessible={true}
  accessibilityRole="image"
  style={{
    width: 300,
    height: 200,
    borderRadius: 8,
```



Interactive Example

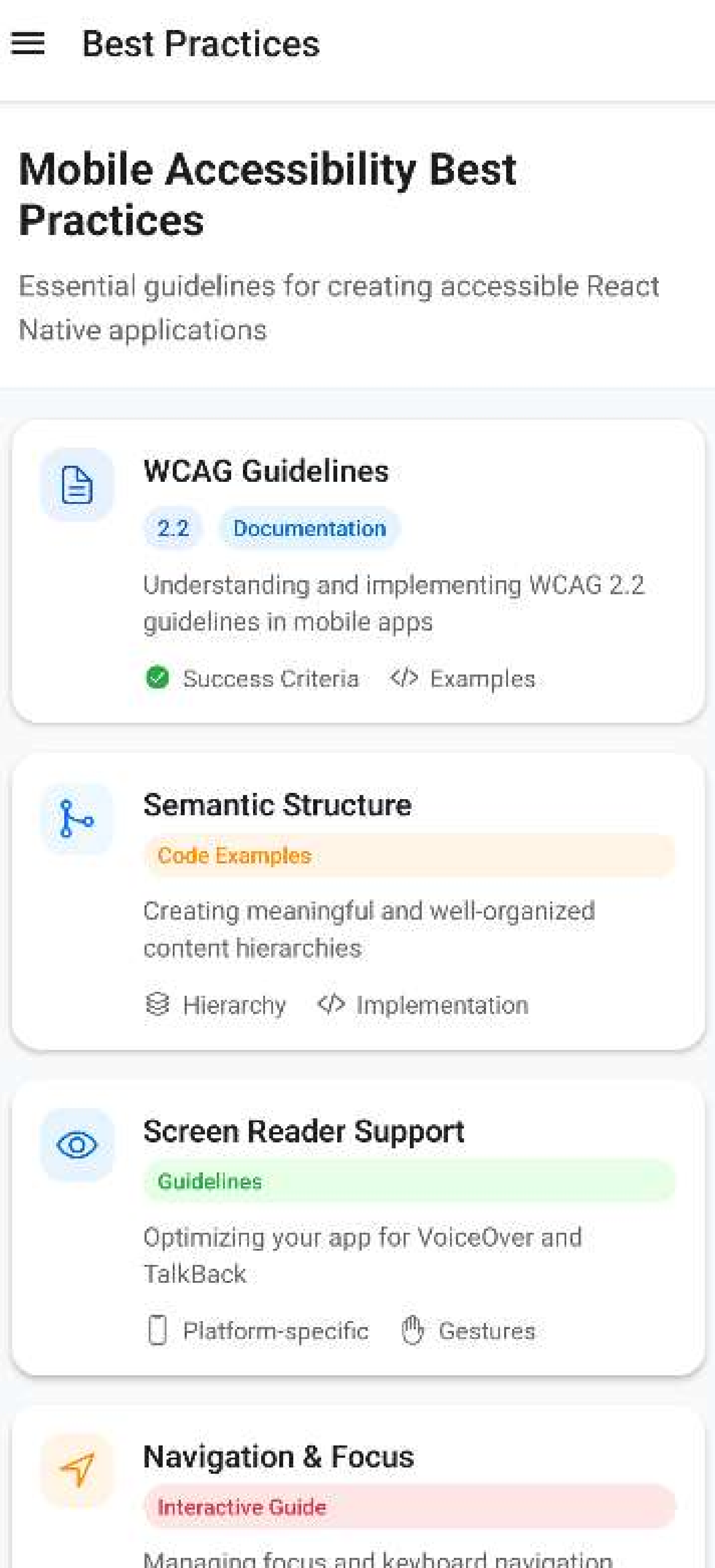
Try this dialog with VoiceOver/TalkBack enabled

Implementation

```
// 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"
```



Best 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

Screen Reader Support

Guidelines

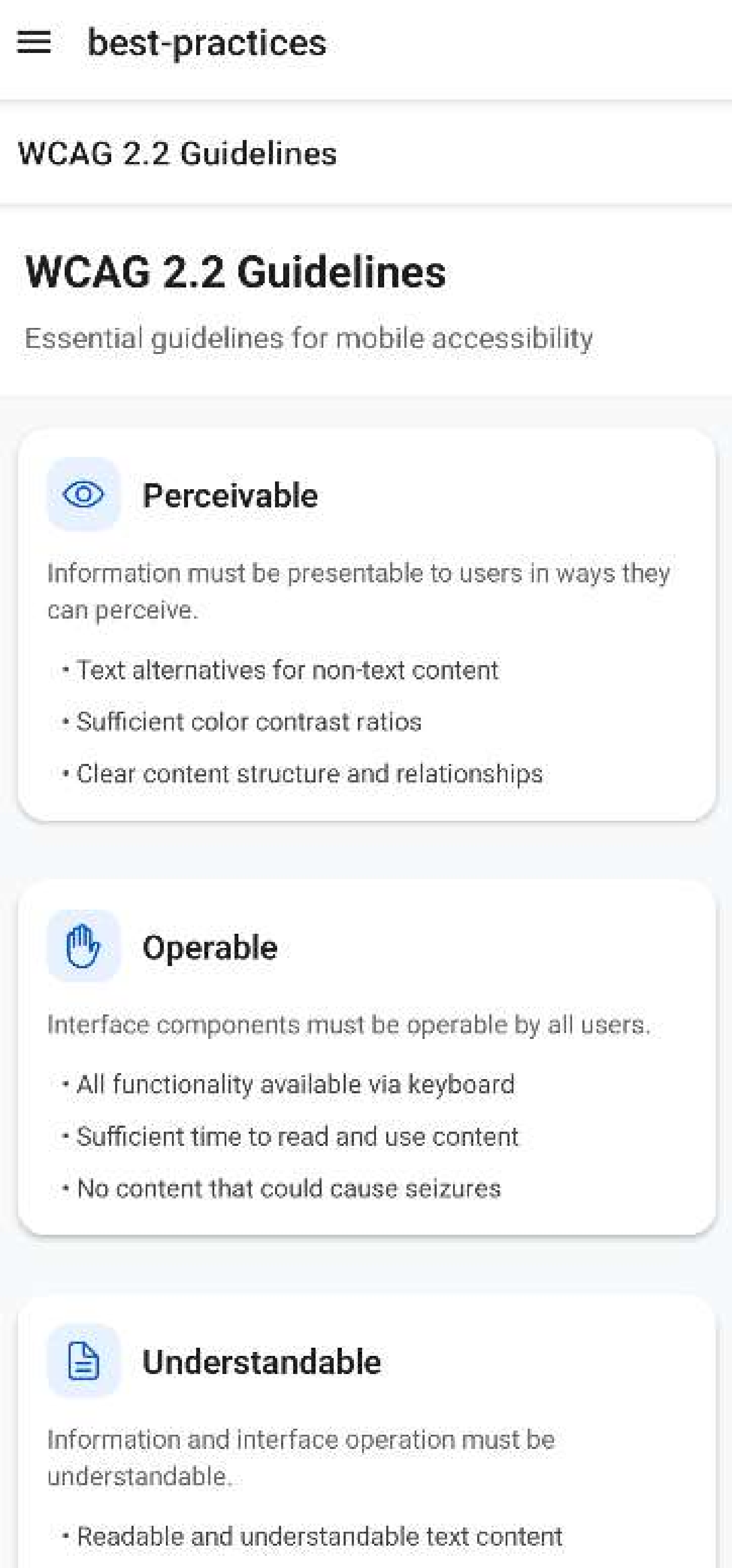
Optimizing your app for VoiceOver and TalkBack

Platform-specific Gestures

Navigation & Focus

Interactive Guide

Managing focus and keyboard navigation



☰ best-practices

WCAG 2.2 Guidelines

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

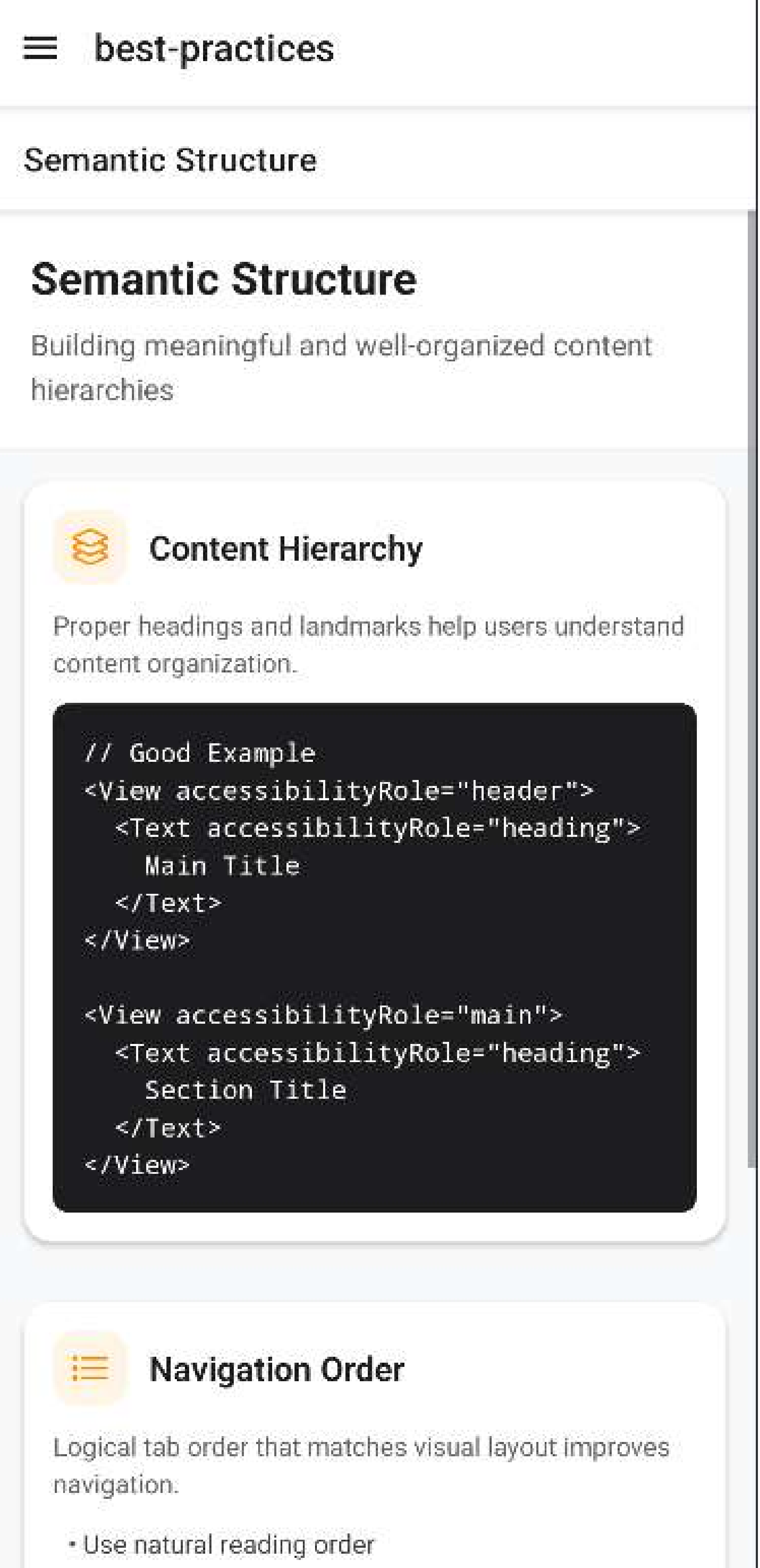
Interface components must be operable by all users.

- 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



☰ best-practices

Semantic Structure

Semantic Structure

Building meaningful and well-organized content hierarchies

Content Hierarchy

Proper headings and landmarks help users understand content organization.

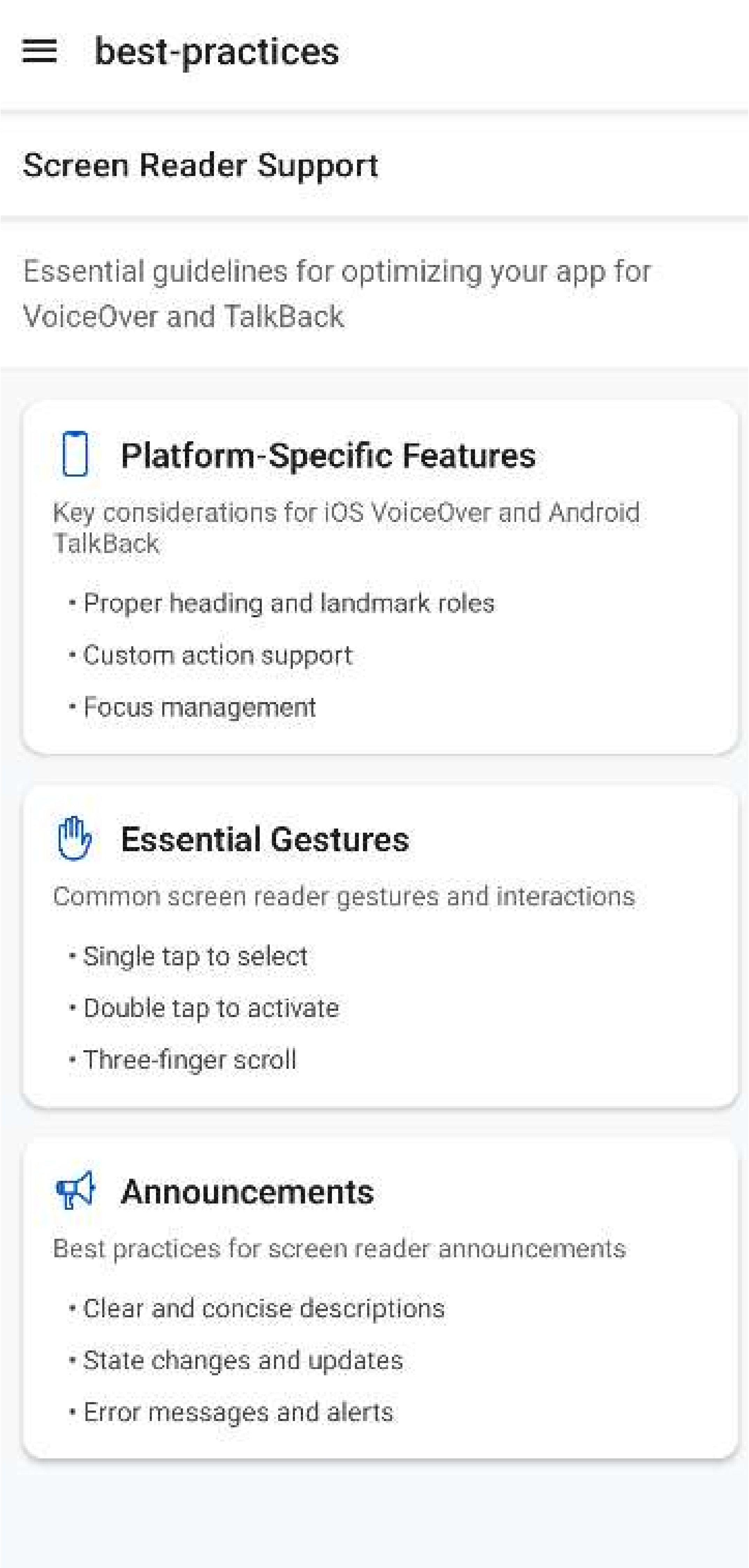
```
// Good Example
<View accessibilityRole="header">
  <Text accessibilityRole="heading">
    Main Title
  </Text>
</View>

<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



best-practices

Screen Reader Support

Essential guidelines for optimizing your app for VoiceOver and TalkBack

Platform-Specific Features

Key considerations for iOS VoiceOver and Android TalkBack

- Proper heading and landmark roles
- Custom action support
- Focus management

Essential Gestures

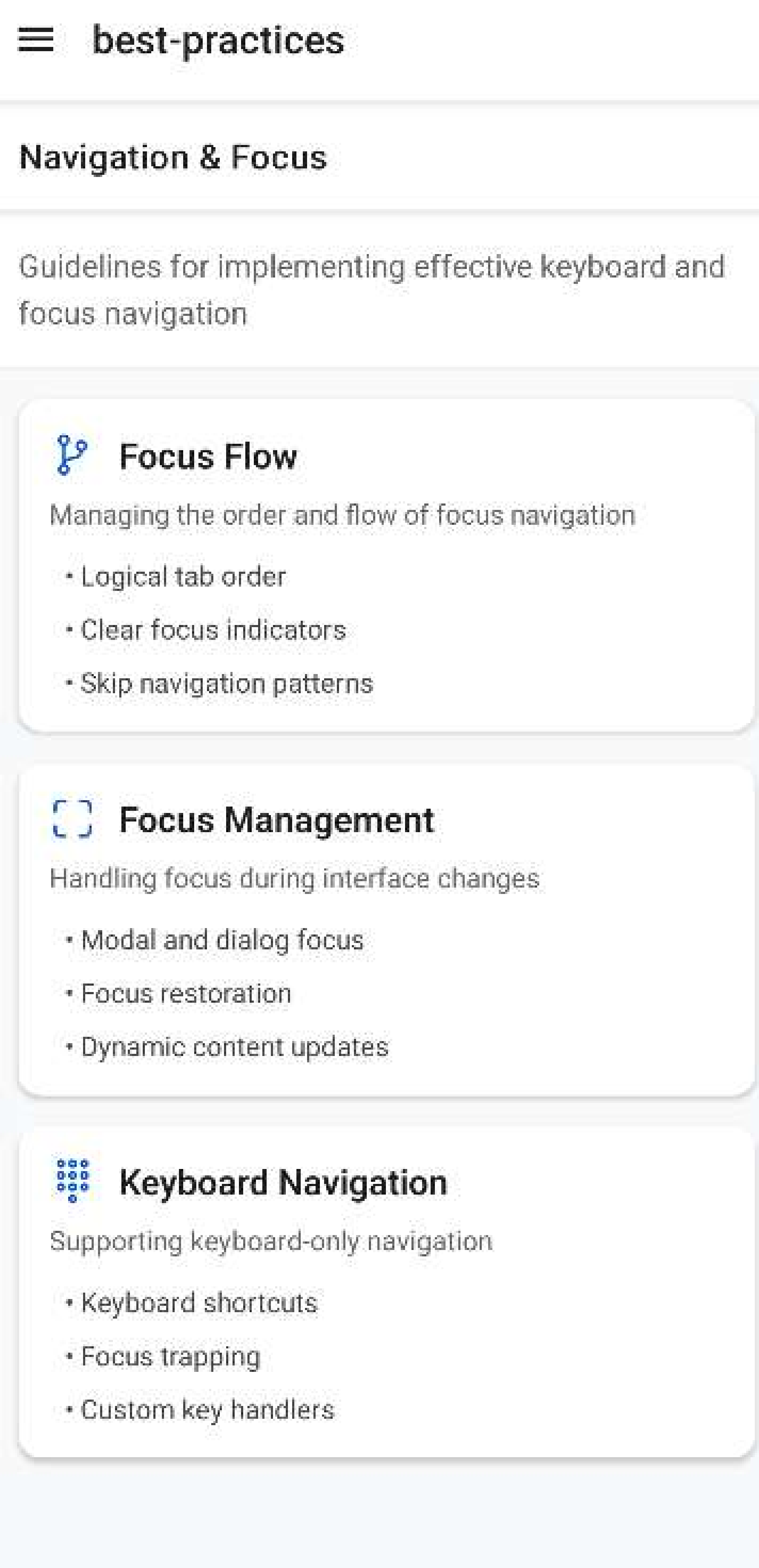
Common screen reader gestures and interactions

- Single tap to select
- Double tap to activate
- Three-finger scroll

Announcements

Best practices for screen reader announcements

- Clear and concise descriptions
- State changes and updates
- Error messages and alerts



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

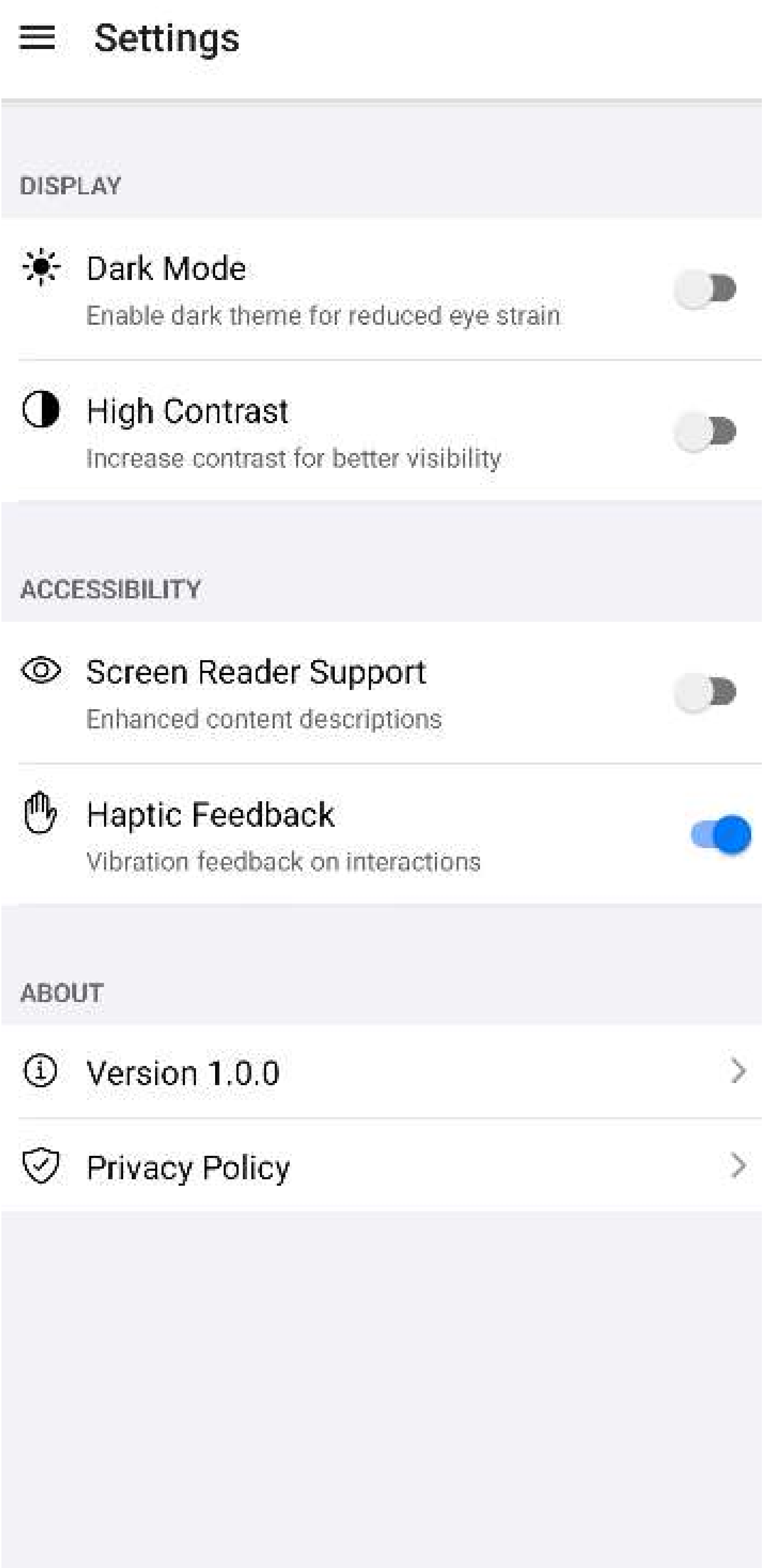
Handling focus during interface changes

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

Keyboard Navigation

Supporting keyboard-only navigation

- Keyboard shortcuts
- Focus trapping
- Custom key handlers



☰ Settings

DISPLAY

Dark Mode ○
Enable dark theme for reduced eye strain

High Contrast ○
Increase contrast for better visibility

ACCESSIBILITY

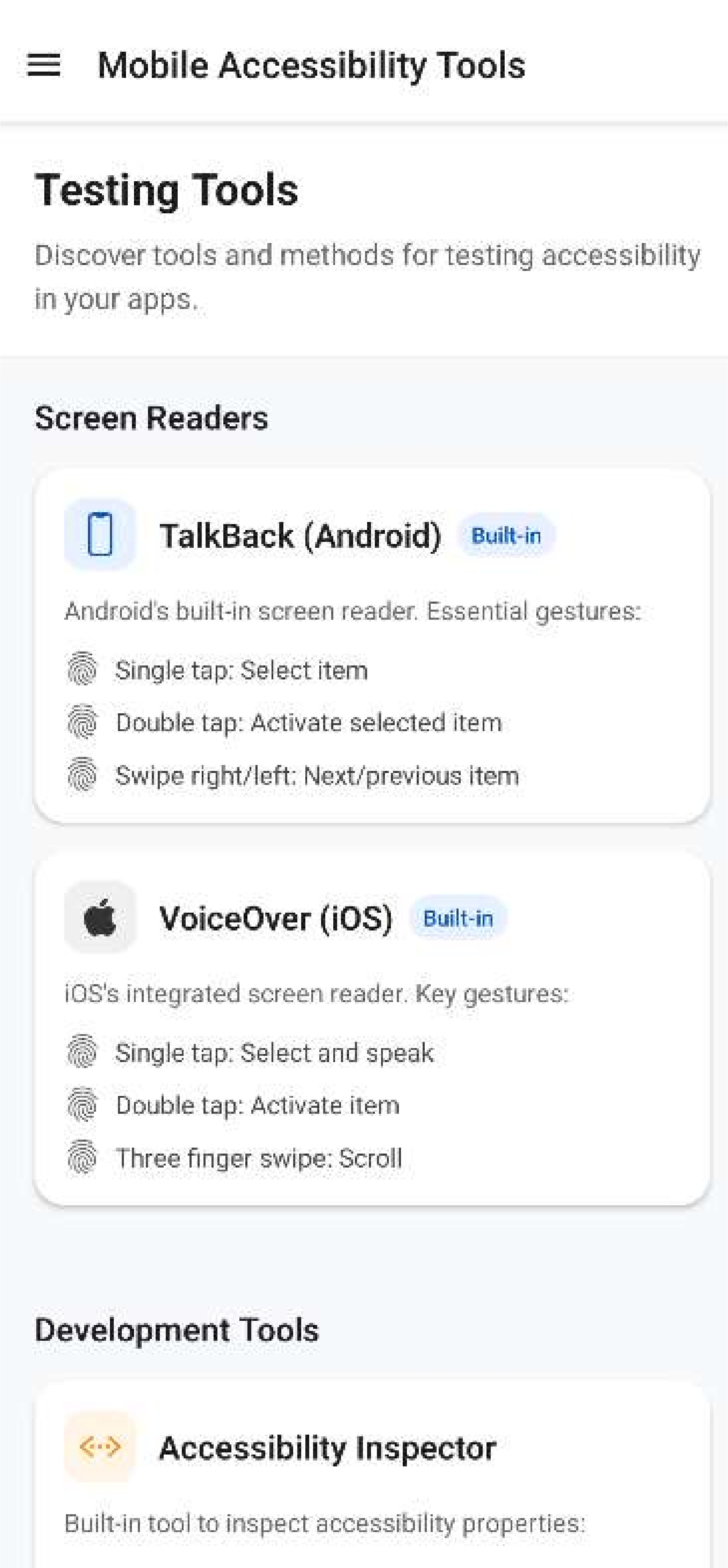
Screen Reader Support ○
Enhanced content descriptions

Haptic Feedback ●
Vibration feedback on interactions

ABOUT

Version 1.0.0 >

Privacy Policy >



Mobile Accessibility Tools

Testing Tools

Discover tools and methods for testing accessibility in your apps.

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: