

QRCS Accessible Donation Prototype – Handover Manual

Client: Qatar Red Crescent Society (QRCS)

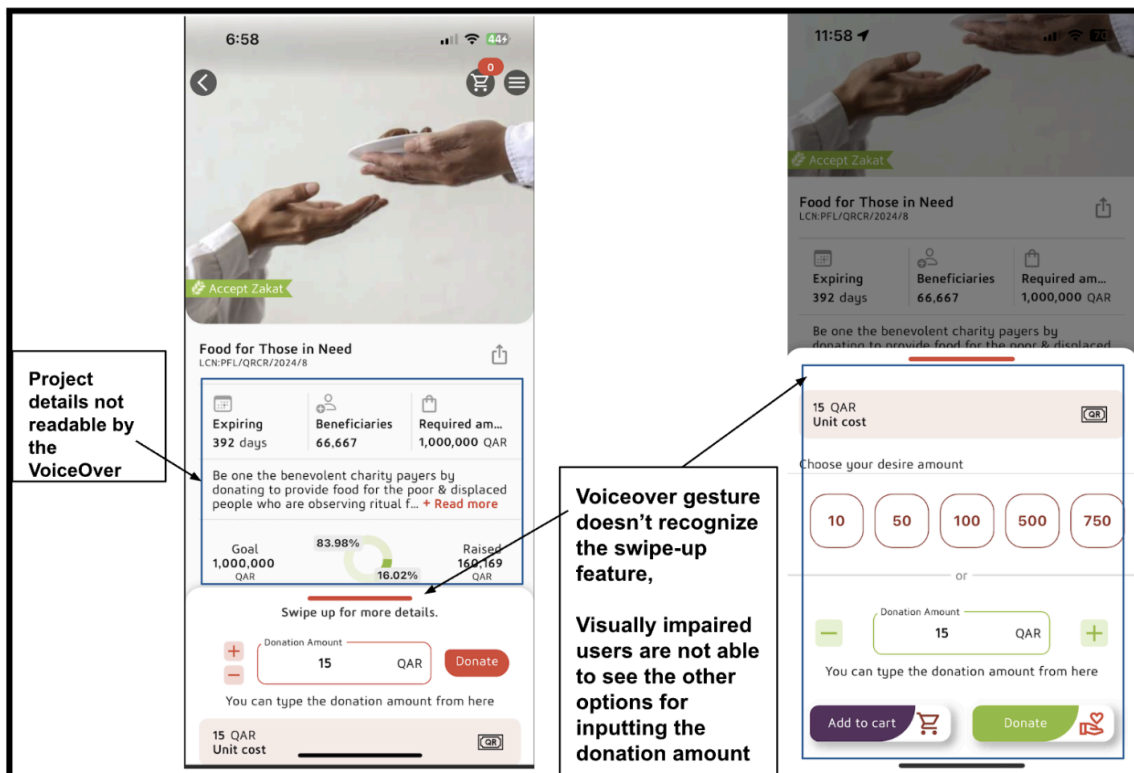
Team: VisionForAll (Al Anoud Al Khulaifi, Fatima Al-Haddad, Deema Al-Mohanadi)

Advisor: Professor Abdul Salam Knio

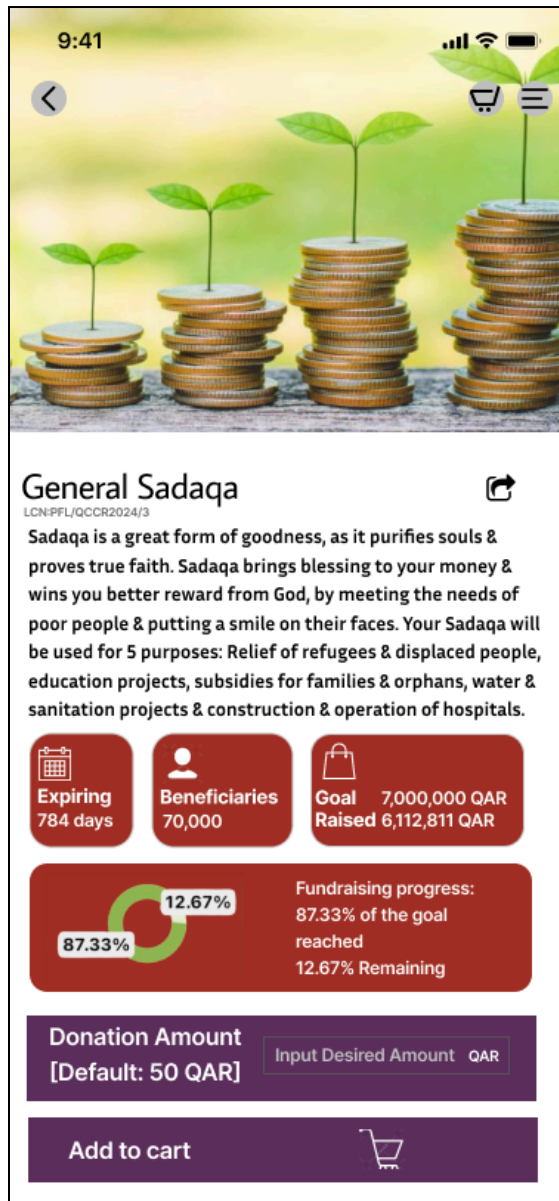
Purpose of This Manual

This handover manual is intended to provide QRCS developers and technical staff with a comprehensive overview of the accessible donation prototype developed by Team VisionForAll. It documents accessibility standards followed, changes made during design and implementation, testing procedures, and tools for future development.

Before (Current QRCS mobile application):



After (Low-fidelity WireFrame on Figma):





Accessibility Guidelines Followed:

Color Contrast

- Follows WCAG Success Criterion 1.4.3 (Minimum Contrast)
- Checked using: <https://accessibleweb.com/color-contrast-checker/>

Box to background and Text to box:

Foreground Color
 #FFFFFF

Background Color
 #5C315E

Contrast Ratio
10.21:1


WCAG Compliance Results


ELEMENT TYPE	AA	AAA
Small Text	✓ Pass	✓ Pass
Large Text	✓ Pass	✓ Pass
UI Components	✓ Pass	✓ Pass


WCAG AA and AAA Results

SMALL sample text: 14pt (18.5px)

LARGE sample text: 18pt (24px)



Foreground Color
 #000000

Background Color
 #C0C0C0

Contrast Ratio
11.54:1


WCAG Compliance Results


ELEMENT TYPE	AA	AAA
Small Text	✓ Pass	✓ Pass
Large Text	✓ Pass	✓ Pass
UI Components	✓ Pass	✓ Pass


WCAG AA and AAA Results

SMALL sample text: 14pt (18.5px)

LARGE sample text: 18pt (24px)



Foreground Color
 #FFFFFF

Background Color
 #A03123

Contrast Ratio
7.10:1


WCAG Compliance Results

ELEMENT TYPE	AA	AAA
Small Text	✓ Pass	✓ Pass
Large Text	✓ Pass	✓ Pass
UI Components	✓ Pass	✓ Pass

WCAG AA and AAA Results

SMALL sample text: 14pt (18.5px)

LARGE sample text: 18pt (24px)



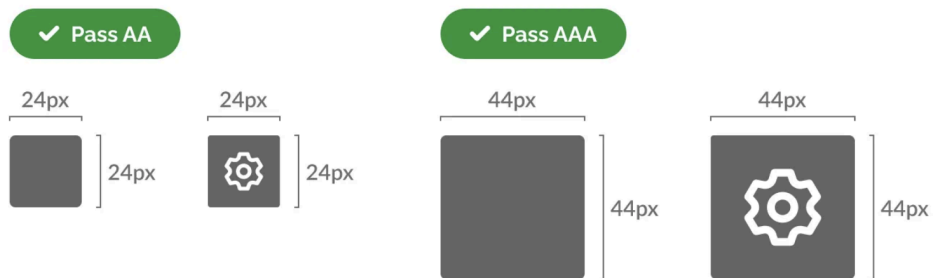
Text Styling

- Follows WCAG 2.1 Success Criterion 1.4.12 (Text Spacing) and 1.4.4 (Resize Text)
- Font: Inter Sans
- Line height (line spacing) to at least 1.5 times the font size.
- Letter spacing (tracking) to at least 0.12 times the font size.
- Word spacing to at least 0.16 times the font size.
- Sizing: should be at least 12

Target Size

- Follows WCAG Success Criterion 2.5.5
- Icons: Ensured that the size of the icons to be at least 24x24
- Ensured the position of the buttons so that they don't overlap

Especially, in WCAG 2.2 released in October 2023, there was an update on more deep about target sizes that were not mentioned in WCAG 2.1.



Changes made to the Donation/Information Page Layout:

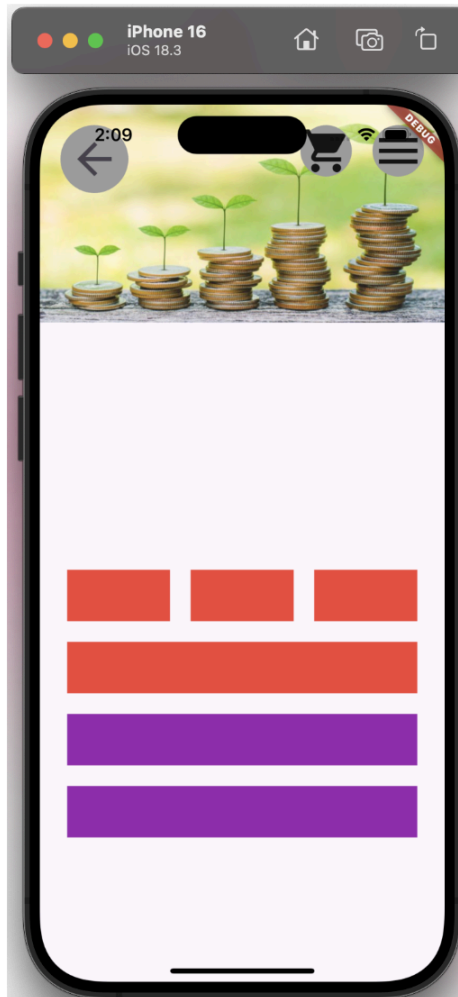
Project Information Section:

- Avoided hiding content: removed the “read more”
- Removed unnecessary repeated information
- Fixed the color contrast icons

Donation Process:

- **Initially:** Too many ways to add a donation amount/ steps to complete donation
- **Updated:**
 - **Provided the users with the simple action of inputting their own amount** through a clear and accessible text field.
 - Screen reader compatibility: Text fields are easier for screen readers to navigate and announce, compared to buttons.
 - Flexible control: Users can donate exactly what they want without being limited to preset amounts.
 - Large, clear input field: A single, well-labeled field minimizes confusion and reduces the number of elements they need to navigate through.
 - Big, high-contrast text: Ensure the input field has large, dark text on a light background for readability.
 - Clear labels and instructions: The field should be clearly labeled ("Input Desired Amount QAR") and announced properly by screen readers.

Flutter MVP (High-Fidelity Progress):




Button Design Ownership

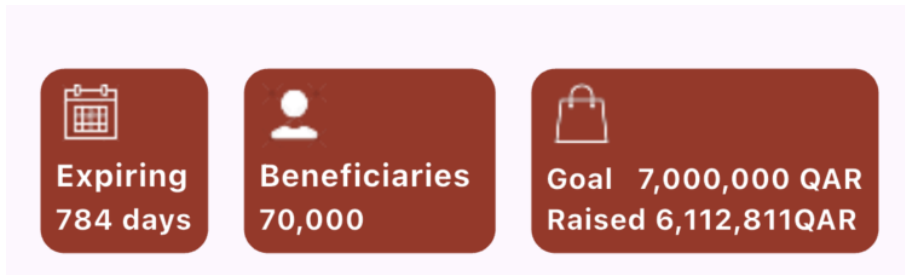
Al-Anoud's Purple Buttons:

Donation Amount
 ₪Default: 50 QAR₪

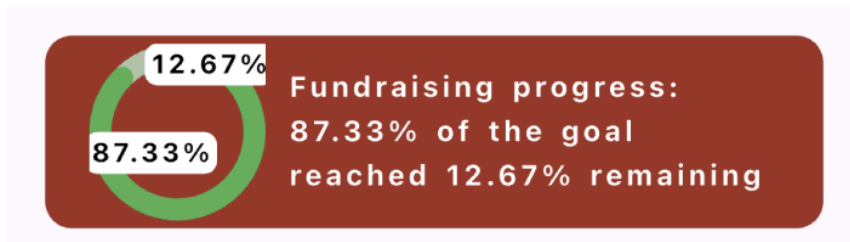
Input Desired
 Amount QAR

Add to Cart
 

Deema's 3 Red Buttons:



Fatima's 1 long Red Button:



Automated Flutter Testing Tools

- [accessibility_tools](#)
 - This is the automated flutter tool we used for testing. It is a Flutter developer package with checkers and tools to ensure that our app is accessible by checking the interface as we build it.
- [Flutter Accessibility Docs](#)
 - Official Flutter documentation that outlines key accessibility best practices, internationalization tips, and configuration for screen reader compatibility.
- [Flutter Accessibility Widgets](#)
 - A list and guide of Flutter widgets specifically designed to enhance app accessibility.
- [Comprehensive Flutter Guide](#)
 - This website shows simple snippets of code for how to implement certain accessibility features in Flutter: Semantic labels, contrast, large touch targets, Alt text for images

Screen Reader Compatibility Testing - Documentation:

Each Tester Should Include:

1. **Feature/Component Tested**
 - Name the component (e.g., "Submit Button", "Home Icon", "Welcome Text").
2. **Screen Reader Output**
 - A transcription of what the screen reader reads aloud.
 - Optionally, screen reader logs or screen recordings.
3. **Success Criteria**
 - Based on WCAG 2.1 (Level AA where applicable).
 - Examples:
 - Text is announced clearly and matches what's visually displayed.
 - Buttons/icons have semantic labels and are properly described.
 - Elements can be navigated in a logical order using swipe gestures.
4. **Screenshots**
 - Highlight the element being tested.
 - Mark areas where issues occur.
5. **Observed Issues**
 - Mislabeling, empty announcements, incorrect focus order, etc.
 - Include device info and OS version.
6. **Suggested Fixes**
 - Propose changes in Flutter (e.g., adding `semanticsLabel`, using `Semantics()` widget).
7. **Overall Status**
 - Pass / Fail / Needs Review

Shared Testing Checklist (WCAG-Aligned)

Item	Check	Notes
1. All non-text UI elements have semantic labels		
2. Text content is read correctly		
3. Buttons and icons are identified as actionable		
4. No focus traps exist		
5. All custom widgets expose semantics		
6. Role and state are conveyed (e.g., switch on/off)		
7. Contrast issues noted (optional)		

Deploying the Prototype to a Physical iOS Device

To test or demonstrate the Flutter prototype on a physical iOS device, follow these steps:

1. Apple Developer Account

- You do not need a paid Apple Developer account for testing.
- Create an Apple ID if you don't have one: [Create Apple ID](#)

2. Set Up Your iOS Device

- Connect your iPhone to your Mac via USB.
- Tap "**Trust this computer**" when prompted.
- Unlock your iOS device.
- Enable **Developer Mode** (iOS 16 or later):
 - Go to **Settings > Privacy & Security > Developer Mode**
 - Enable it and restart your device
 - Tap **Turn On** when prompted after reboot

3. Enable Developer Code Signing in Xcode

- Open Xcode > Preferences > Accounts > Add Apple ID
- Go to **File > Open...** and open your project: `ios/Runner.xcworkspace`
- In the left panel, select **Runner** under Targets
- Go to **Signing & Capabilities > Signing** tab
- Enable **Automatically manage signing**
- Select your personal Apple ID team under Team dropdown
- Specifically, the one that is stored within the code is `com.deema.prototypeApp` but it can be switched to your personal and unique Bundle Identifier to launch the app.

4. Trust Certificates on Device

- On your iPhone: Go to **Settings > General > Profiles & Device Management**
- Trust the Developer Certificate if prompted
- If asked to allow key access on your Mac, enter your Mac password and choose **Always Allow**

VoiceOver Setup & Usage

On iOS (iPhone or iPad)

1. Go to **Settings > Accessibility > VoiceOver**

2. Toggle VoiceOver **ON**
3. Use the following gestures:
 - **Swipe right/left** to move through elements
 - **Double-tap** to activate an item
 - **Three-finger swipe up/down** to scroll