QR Craft Milestone #1 Requirements

This document outlines the requirements for the first development milestone (Milestone #1) of the QR Craft application.

Each milestone will be accompanied by a dedicated requirements document and an updated Figma file. All current milestones and design assets are available in the project's Discord community channel.

The mockups define the intended appearance and behavior of the app and its components. The app uses a custom visual theme and does not require support for light or dark modes.

While the mockups define layout and design, you are free to decide how to technically implement certain elements — for example, how camera permission dialogs, loading indicators, or error messages are displayed.

You can find the QR Craft **mockups here**:

> https://www.figma.com/design/cyboRIV1lOA5UQmhLGDAlE/QR-Craft?node-id=0-1

Milestone #1 Goal

Deliver a fast, smooth, camera-first scanning experience.

Your task is to build the main **scanning screen**, which serves as the app's primary entry point. This includes a live camera view, real-time QR code detection, and a result screen with basic actions (copy, share). You also need to implement **camera permission handling**.

Icons

You may use Material Design icons where appropriate. If a suitable Material icon is not available, use the custom icons provided in the Figma mockups.

Adaptive Layouts

The mockups provide two layout versions:

- for mobile devices (**up to 600dp**)
- for wider screens (**from 600dp and above**)

Both versions must be implemented in the app as shown in the provided designs.

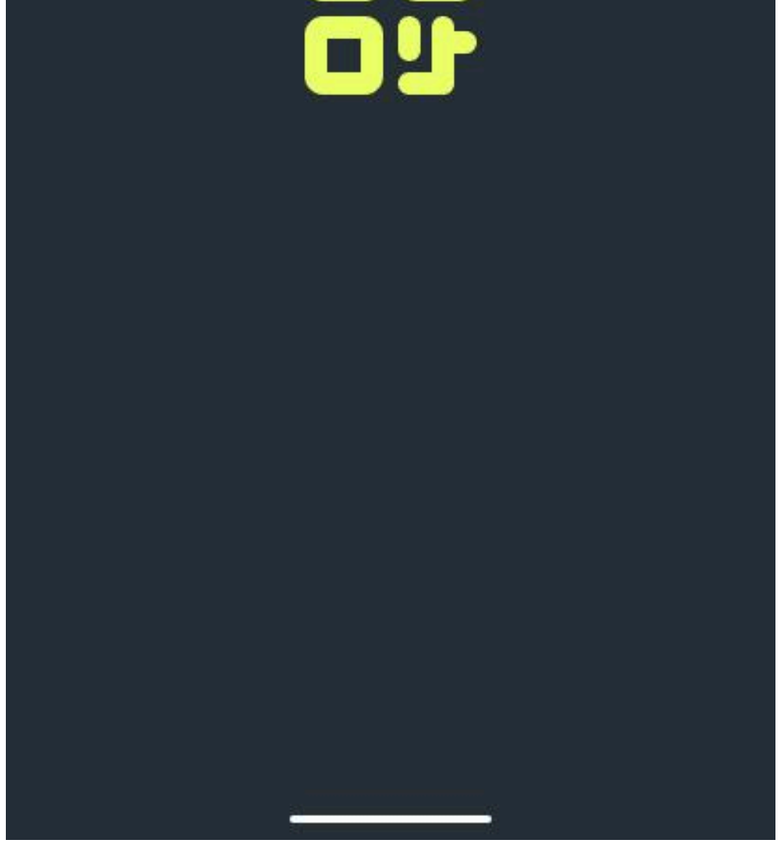
Technical Requirements

🚀 Splash Screen

- Background: solid dark gray with no gradients or images.
- Bright yellow QR Craft logo centered on the screen.
- The background should fill the entire screen (edge to edge).

📷 Main Screen — QR Code Scanning

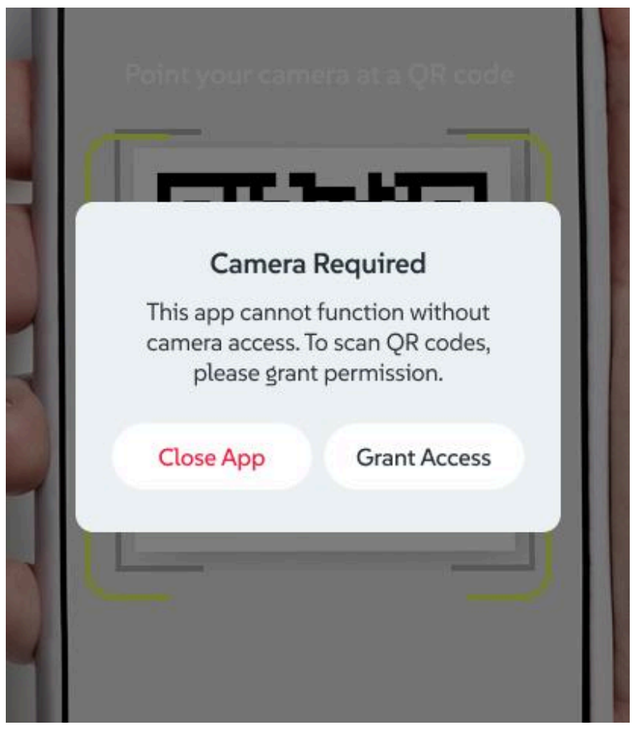
- On app launch, the app must first check for camera permission.
- If the permission has not been granted, a permission dialog is shown.
- If the permission is granted, the camera starts with a live preview and scanning frame.
- A hint is displayed above the frame: "**Point your camera at a QR code**"



🔒 Camera Permission Dialog

If the user has not granted camera access:

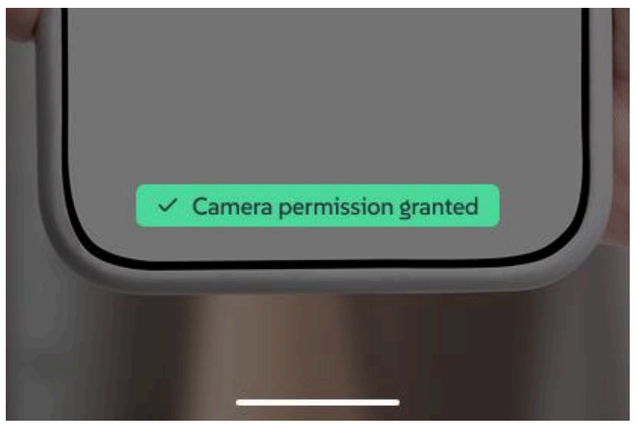
- A modal dialog appears over the scanning screen.
- Title: "**Camera Required**"
- Message: "*This app cannot function without camera access. To scan QR codes, please grant permission.*"
- Two buttons:
  - **Close App** — closes the application.
  - **Grant Access** — triggers the system permission request dialog.



🟢 Snackbar: Permission Confirmation

Once the camera permission is granted, a snackbar appears at the bottom of the screen:

- Text: "**Camera permission granted**"
- Snackbar background color: green.
- The snackbar disappears automatically after a few seconds.



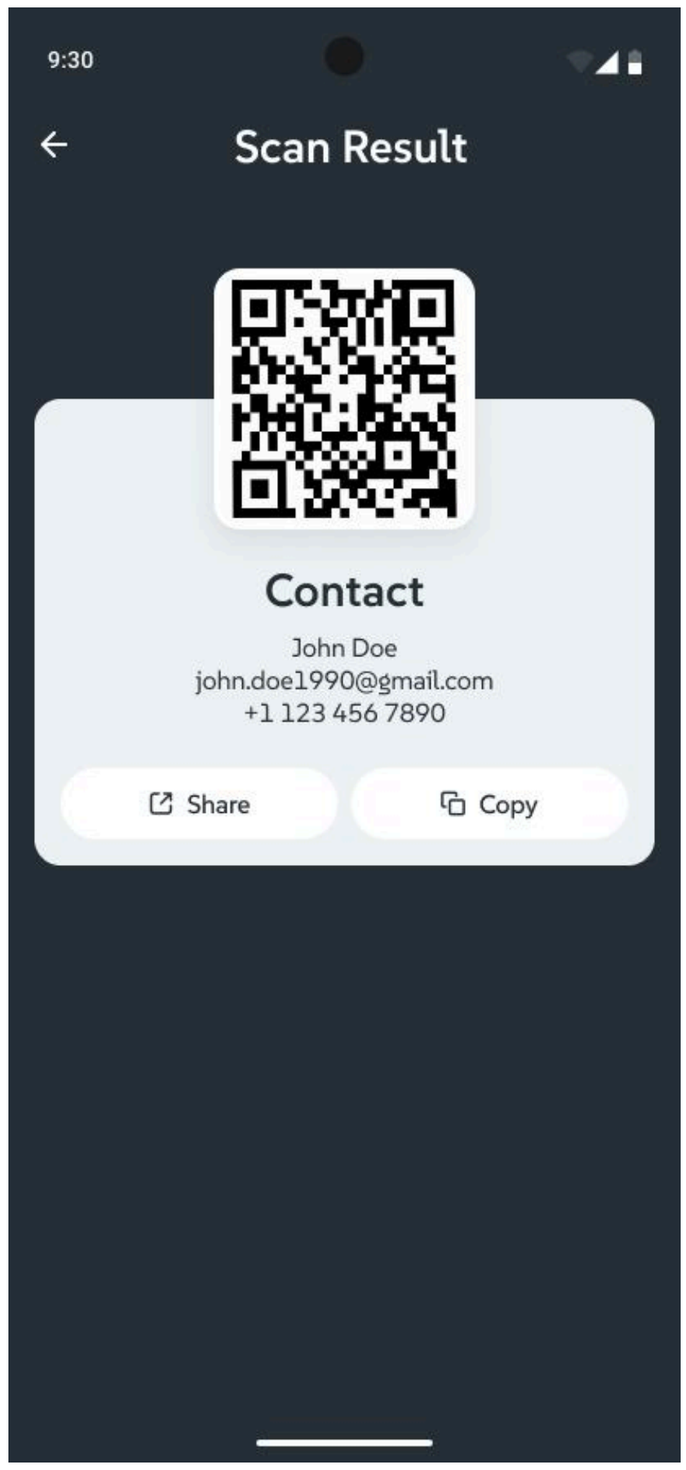
⏳ Scan State — Loading

- When a QR code enters the camera's focus, a loading animation appears directly over the scanner area.
- A centered spinner with the label: "**Loading...**" is shown.
- At this stage, the app is processing the camera image to detect a QR code.
- If the QR code is successfully read:
  - The app navigates to the **ScanResult** screen.

📄 Scan Result Screen (ScanResult)

After a QR code is successfully scanned, the Scan Result screen is displayed with the following structure:

- **Top bar:**
  - The screen title "**Scan Result**" is centered.
  - On the left side of the top bar, there is a back arrow button — tapping it returns the user to the previous screen.
  - When navigating back, the Scan Result screen should be **removed from the navigation stack**.
- **QR code** — displayed at the top of the main content block, centered horizontally.
- **Scanned content block:**
  - Shows the content type label (e.g., Text, Link, Contact, etc.).
  - Below the label, the scanned value is displayed.
  - All types, except for Text, should be **center-aligned**.
  - For the Text type specifically, the **value should be left-aligned** to ensure better readability for longer content.
  - For the Link type, the link should be rendered with a **highlighted background color** (e.g., lime yellow #EBFF69) to make it visually stand out as tappable.



Special Handling for the Text Type

- The scanned value is displayed as a paragraph with **left-aligned** text.
- If the content is long, it is automatically **truncated to 6 lines**.
- A "**Show more**" button appears below the text.
  - This button is styled with a noticeable dark-blue color (#505F6A) to differentiate it from regular text.
- When the user taps "**Show more**":
  - The full text is expanded and displayed.
  - The button changes to "**Show less**".
  - The "Show less" button has a lighter tone (#8C99A2) to reduce visual emphasis.
- Tapping "**Show less**" collapses the text back to 6 lines.

Content Type Detection After Scanning

After scanning a QR code, the result is received as a plain text string. To display the appropriate result screen, the app must detect the type of content based on its structure or matching patterns.

The scanner should support automatic recognition of the following types:

- **Link**: if the text starts with http:// or https://, it is treated as a URL. The result will be displayed as a clickable link.
- **Contact**: if the string starts with BEGIN:VCARD, the content is interpreted as vCard contact information.
- **Phone Number:** if the text matches a phone number format (e.g., starts with + or contains only digits, spaces, parentheses, or dashes).
- **Geolocation:** if the text contains coordinates in the format latitude, longitude.
- **Wi-Fi:** if the string starts with WIFI:, it is interpreted as Wi-Fi connection data (SSID, password, encryption type).
- **Text:** if none of the above patterns match, the content is treated as plain text.

This detection logic allows the app to render the correct visualization on the result screen. All type handling should be implemented centrally in a dedicated logic module.

> 💡 For more reliable content type detection, you can use third-party libraries such as **ZXing** or **Google ML Kit**. These libraries return structured content types (e.g., URL, ContactInfo, GeoPoint) out of the box, which simplifies the processing logic and reduces the need for manual parsing.

Action Buttons

At the bottom of the content block, two action buttons are always shown:

- **Share** — opens the system sharing sheet (e.g., WhatsApp, Gmail, etc.).
- **Copy** — copies the QR code content to the clipboard.

Adaptation for Larger Screens

- Regardless of screen width, the content block must have a **fixed width of 480dp.**
- This ensures consistent UI on tablets and large-screen devices.
- The content block should be horizontally centered.