Faculty of Information and Communication Technology
BsC Hons in Applied Computer Science

3rd year

PHM
QR-Code Scanner

Submitted by:
LINGACHETTI Paavalen
Hachim Hassani Bacar

Submitted to:
Mr BEEHARRY Shiam

Date : 01/03/2024

Abstract

This report presents a detailed account of the development process and key features of QR code scanner mobile application designed to generate and scan QR codes. Here we are providing insights into the challenges we overcome during the design, implementation, and testing phases. The report begins with a comprehensive analysis of the requirements and features needed for this project.

The design phase outlines the user interface (UI) and user experience (UX) considerations, ensuring that the application offers an intuitive and visually appealing platform.The implementation section delves into the technical aspects such as the choice of development platform, language used and any library used for the project. The testing phase will show the challenges and problems we faced off and the solution used to overcome them.

In conclusion, this report not only offers a comprehensive overview of QR Code Scanner development to understanding the intricacies of creating a QR code mobile application.

Table of content

Prerequisites

Here are all the software and hardware requirements used for the development and utilisation of the app.

Software requirements

1. **Flutter SDK**

Install the Flutter SDK on the development machine. Developers are encouraged to follow the official Flutter installation guide tailored to their specific operating system.

2. **Dart**

The programming language seamlessly integrated with Flutter, constitutes another essential requirement. Typically bundled with Flutter, Dart eliminates the need for a separate installation.

3. **Integrated Development Environment (IDE)**

Use any of your preferred IDE, For this project we used Android studio with all dependencies required to develop with flutter.

Hardware Requirements

1. **Mobile Device**

A mobile device such as a phone and tablet running Android or IOS alternatively an android emulator are required for testing. These devices are required to have a camera.

2. **Computer**

A desktop or laptop running either Windows, Linux or Mac OS is necessary for development, and it should possess sufficient computing power to support the demands of running both the Integrated Development Environment (IDE) and emulator. In our specific case, we utilised laptops that match or exceed the recommended specifications for running Android Studio.

Installation

1. **Download the project files**

Clone the repository on github

2. **Install dependencies**

They are listed in the pubspec.yaml file

3. **Run the app**

Type the command flutter run in the terminal to run the app on an emulator or a connected device.

Introduction

The Quick Response (QR) code has revolutionised the way information is shared and accessed. Recognizing the increasing significance of this technology, our project has been to develop a QR code scanner mobile application that seamlessly integrates functionality and efficiency. This report provides an explanation of the developmental journey behind this app using the Flutter framework and Dart programming language.

Flutter, with its cross-platform capabilities, and Dart, as its accompanying language, offer an ideal platform for constructing a versatile and performant mobile application. This report aims to show the process of design, development, and testing used during the creation of our QR code scanner mobile app.

Objectives

The primary objective of our QR code scanner mobile app is to provide users with a fast, reliable, and user-friendly tool for effortlessly generating and decoding QR codes using flutter.

Main functionalities

- Generate QR codes

Enable the user to generate their own QR codes.

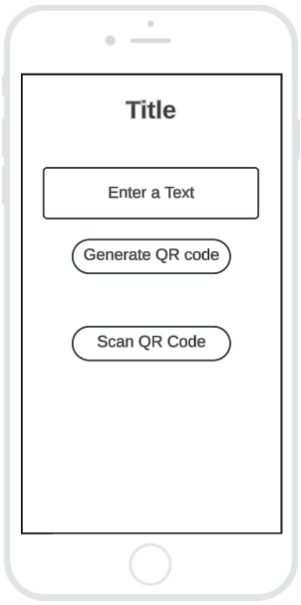
- Scan QR codes

Enable use to scan QR codes

Wireframe

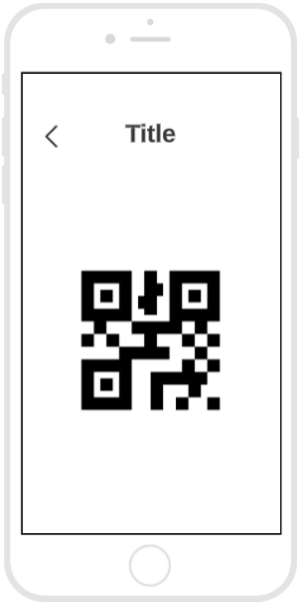
Screen 1

This is the first screen you will see after opening the app. Here you can enter a text or link then generate a QR code by pressing the first button or scan a QR code with the second button.



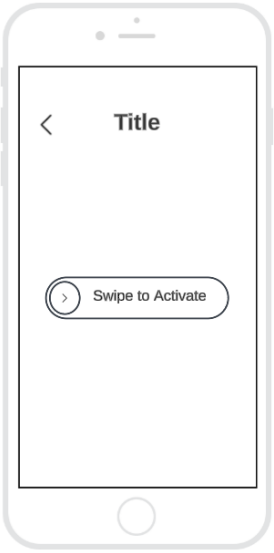
Screen 2

This screen shows the generated QR and a back button.



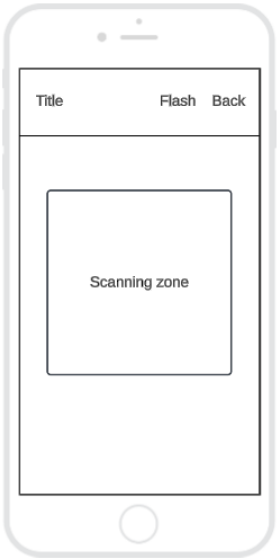
Screen 3

This screen appears when you want to scan a QR code. The user has to swipe the circle to start scanning.



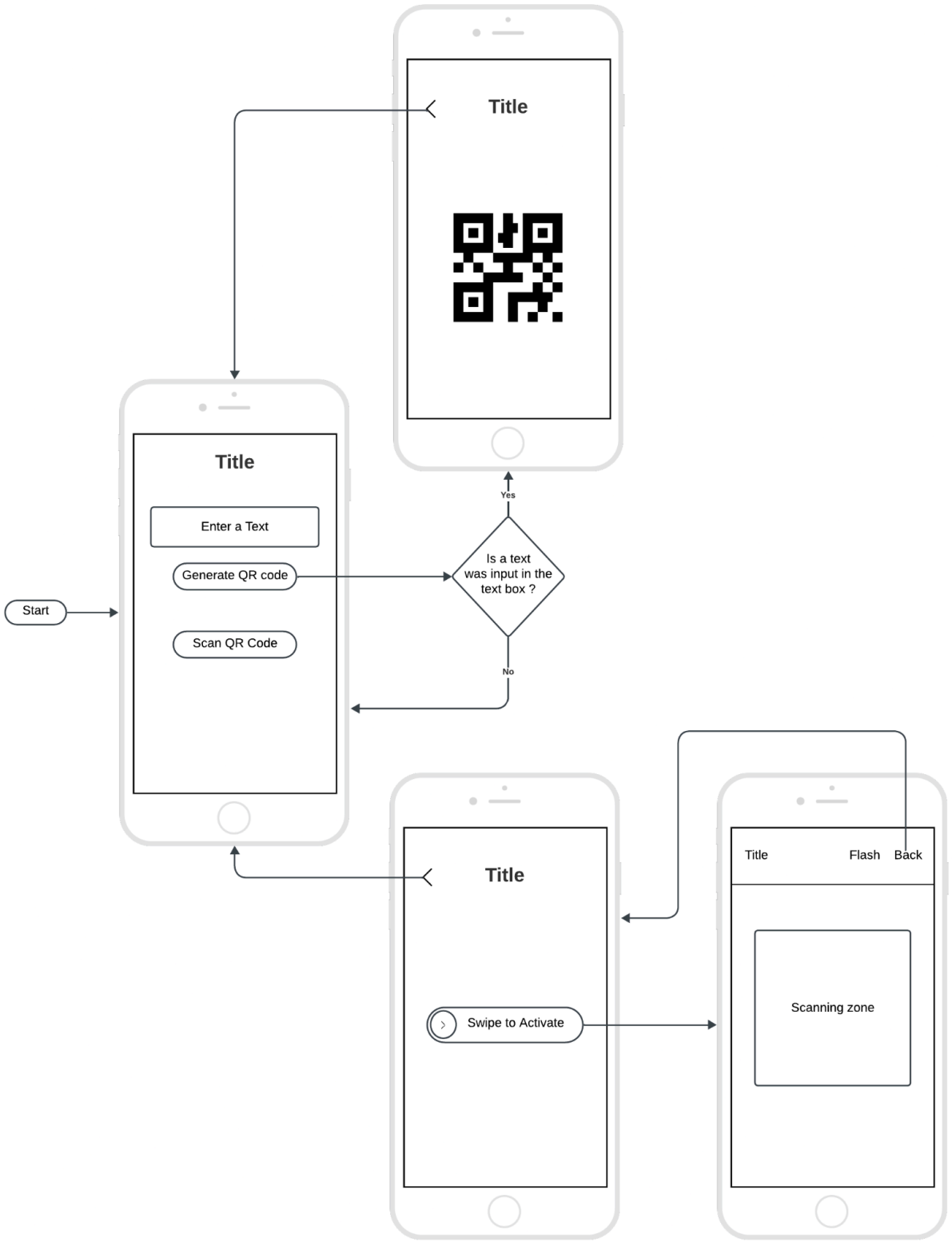
Screen 4

This screen shows the camera feed with the scanning zone highlighted.



Wire Flow

The figure below the wire flow shows how the user navigates in the app.



Each screens in the wire flow are the same as the one in section above, we will call then screen 1 to 4

As the wire flow shows the user will start on screen 1 when they open the app, here they are presented with a title, text box and 2 buttons.

The first button allows them to generate a QR code if they enter a text or link in the text box else nothing happens if they try to press the button.

The second button allows them to open the camera to scan a qr after they use the gesture button.

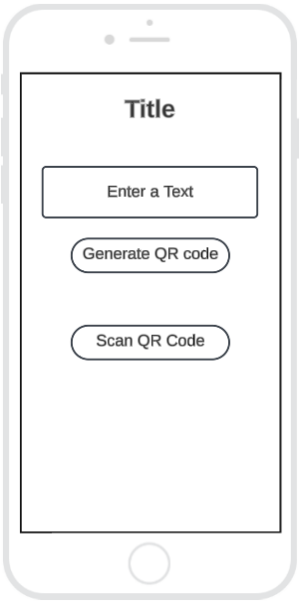
On screen 4 you can scan a qr code:
If it is a link the browser will open
If it is a text a toast with the text will appear and redirect you to screen 3.
There is also a button for back and activate the flash.

Final Design

Here is a comparison between the wireframe and the final design of the app.

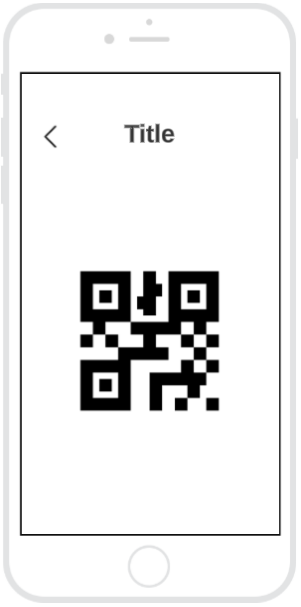
Screen 1

This is the main menu of the app where you are given the option to generate or scan a QR code

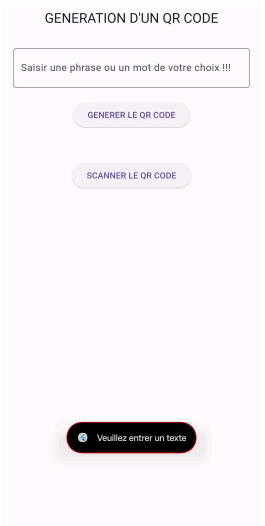


Screen 2

After inputting a text or link in the text box and press the generate button the Qr code is displayed

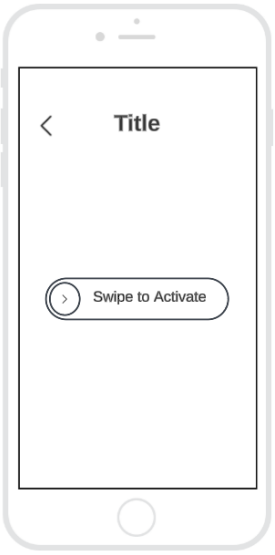


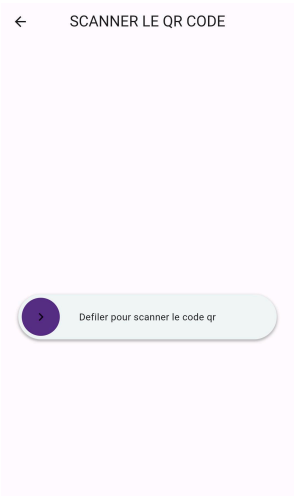
If nothing was entered in the text box this toast will appear



Screen 3

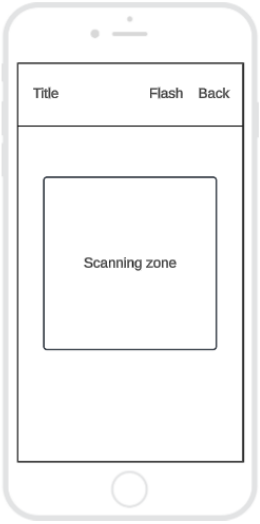
This screen show the gesture button to start the scanning

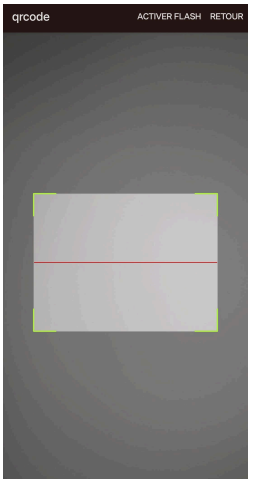




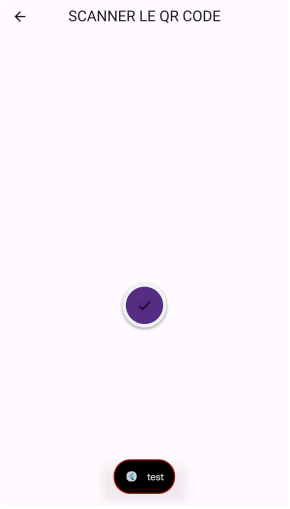
Screen 4

This screen shows the camera feed to scan a qr code, there's also a button to switch on the flash.

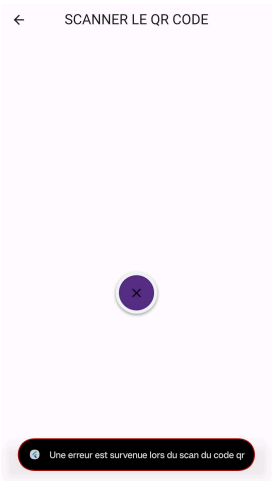




If the scan is successful and the QR code contains a text this screen is displayed with the text of the QR code in a toast.



If the user decides to cancel the scan or an error occurs this screen and toast is displayed.



Code explanation

The project's QR Code Generator and Scanner code is structured into multiple files and classes, each with a specific role. Here's a brief explanation of key parts:

1. **GenerateQRCode.dart:**
    - This file defines a user interface (UI) for generating QR codes.
    - It utilises packages like **flutter toast** for notifications and **qr_image** to display and share generated QR codes.
2. **QRCodeScanner.dart:**
    - This file implements the QR code scanning functionality.
    - It uses the **barcode_scan2** package to interact with the camera and scan codes.
    - The use of an Action Slider makes activating the scanner convenient.
3. **QRImage.dart:**
    - This class represents the screen displaying a generated QR code.
    - The **qr_flutter** package is used to generate the QR code from the provided text.

Testing

The project has undergone testing to ensure the proper functioning of its features. Tests covered the following aspects:

1. **QR Code Generation:**
    - Input of various texts and links, including edge cases.
    - Verification of correct QR code generation.
2. **QR Code Scanning:**
    - Simulating different lighting situations.

Scanning QR codes with various contents

Problems

During development, a few issues were identified:

1. **Permission Issues:**
    - Some users faced problems related to camera access permissions.
2. **Error Handling:**
    - Unexpected errors may occur during scanning, such as poor QR code image quality.

Solutions

To address the aforementioned problems, the following solutions were implemented:

1. **Camera Permissions:**
    - Addition of explanatory messages to guide users when facing authorization issues.
2. **Error Handling:**
    - Improved error handling to provide explicit messages in case of issues during scanning.

Reference list

Dart.dev. (2019). *Dart documentation*. [online] Available at: https://dart.dev/guides.

Flutter (2024). *Flutter documentation*. [online] docs.flutter.dev. Available at: https://docs.flutter.dev/.

flutter.dev. (2024). *Community*. [online] Available at: https://flutter.dev/community.

juliuscanute (2024). *juliuscanute/qr_code_scanner*. [online] GitHub. Available at: https://github.com/juliuscanute/qr_code_scanner [Accessed 1 Mar. 2024].

packages, using (2024). *Using packages*. [online] docs.flutter.dev. Available at: https://docs.flutter.dev/packages-and-plugins/using-packages.

Stack Overflow (2024). *Stack Overflow - Where Developers Learn, Share, & Build Careers*. [online] Stack Overflow. Available at: https://stackoverflow.com/.