ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

-----🕮-----

BÁO CÁO ĐỒ ÁN MÔN HỌC
NHẬP MÔN ỨNG DỤNG DI ĐỘNG

QR Code Scanner

Sinh viên thực hiện:

Lê Tuấn Anh - 18520450

Giảng viên hướng dẫn:

Ths. Huỳnh Tuấn Anh

TP.HCM, ngày 26/01/2021

MỤC LỤC

I. Giới thiệu đề tài. 3

1. Giới thiệu đề tài. 3
2. Mục tiêu đề tài. 3
a. Mục tiêu. 3
b. Yêu cầu. 3
c. Đối tượng người dùng. 3
3. Các chức năng chính. 3

II. Những điều cần biết về QR Code. 3

1. Mã QR là gì ? 3
2. Lịch sử phát triển 4
3. Các tính năng quan trọng của QR Code. 5
4. Những hạn chế 7
5. Cách tạo ra mã QR Code 7

III. Công việc liên quan. 10

1. Thêm các thư viện cần thiết. 10
2. Thiết kế giao diện. 10
3. Thiết kế các lớp ứng dụng. 10
4. Thực hiện các hàm xử lí thông tin QR code. 10
5. Thiết kế cơ sở dữ liệu, truy xuất, lưu thông tin vào cơ sở dữ liệu. 10
6. Cài đặt kiểm tra. 10

IV. Phương pháp thực hiện. 10

1. Thêm các thư viện cần thiết. 10
2. Thiết kế giao diện. 11
3. Thiết kế các lớp ứng dụng. 12
4. Thực hiện các hàm xử lí thông tin QR code. 13
5. Thiết kế cơ sở dữ liệu, truy xuất, lưu thông tin vào cơ sở dữ liệu. 17
6. Cài đặt và kiểm tra. 19

V. Kết quả đạt được. 19

1. Kết quả. 19
2. Ưu điểm. 19
3. Nhược điểm. 19
4. Hướng mở rộng. 19

VI. Kết luận: 20

Tài liệu tham khảo 21

I. Giới thiệu đề tài.

1. Giới thiệu đề tài.

Ngày nay khi nhìn trên nhãn mác hàng hóa chúng ta thường thấy ô vuông lớn bên trong có nhiều ô vuông xếp liên tiếp thành các kỹ tự lạ giống như ma trận, ô vuông đó được gọi là QR Code. Hoặc có thể là những đường kẻ màu trắng đen được gọi là Bar code.

Chỉ bằng cách sử dụng các camera của điện thoại, những thiết bị đọc mã, ta đã có thể đọc được thông tin từ QR Code và làm được nhiều việc như kiểm tra, nhận thông tin về các sản phẩm, thực hiện thanh toán tự động, truy cập trang web, thêm bạn bè trên mạng xã hội.. và còn rất nhiều những lợi ích mà QR Code mang lại cho chúng ta.

Để đáp ứng nhu cầu truy xuất thông tin từ QR Code, nhóm đã phát triển ứng QR Code để thực hiện các chức năng trên.

2. Mục tiêu đề tài.

a. Mục tiêu.

Hỗ trợ người dùng Smart phone Android có mong muốn truy xuất thông tin từ QR Code bằng Camera điện thoại, tạo QR Code với nội dung tùy chọn. Lưu thông tin các QR Code

b. Yêu cầu.

Tính tiện dụng: Ứng dụng phải dễ học, dễ dùng. Thiết kế phải mới (modern), tuy nhiên không quá cầu kỳ phức tạp gây khó chịu cho người dùng.

Tính đúng đắn: Ứng dụng chạy không lỗi.

Tính thích nghi: Ứng dụng có thể chạy tốt trên nhiều thiết bị với cấu hình phần cứng khác nhau và thiết kế kiến trúc thiết bị khác nhau.

Tính tiến hoá: Ứng dụng phải dễ dàng được phát triển thêm tính năng mà không gây ảnh hưởng đến những tính năng đã phát triển trước đó.

c. Đối tượng người dùng.

Người dùng Android có nhu cầu quét, tạo QR Code.

3. Các chức năng chính.

Quét, truy xuất nội dung QR Code: Cho phép người dùng quét QR code, nhận thông tin, sao chép nội dung, lưu mã.

Tạo QR Code: Cho phép người dùng tạo mã với nội dung mong muốn, lưu mã vào thiết bị, chia sẽ mã bằng nhiều cách.

II. Những điều cần biết về QR Code.

1. Mã QR là gì ?

Mã QR một mã ma trận hay còn gọi là mã hai chiều (Two-Dimensional) được phát triển bởi công ty Denso Wave (Nhật Bản) vào năm 1994. “QR" là viết tắt của từ "Quick Response", trong tiếng Anh có nghĩa là đáp ứng nhanh, vì người tạo ra nó có ý định cho phép mã được giải mã ở tốc độ cao. Mã QR chứa thông tin theo cả hướng dọc và ngang , trong khi mã vạch (Barcode) chỉ chứa dữ liệu theo một hướng đồng thời QRcode còn có thể chứa khối lượng lớn hơn thông tin so với mã vạch thông thường.

2. Lịch sử phát triển

Sơ khai của QRcode là Barcode – một loại mã vạch được phát minh năm 1973 và sữ dụng rộng rãi trong nhiều lĩnh vực trên thế giới. Barcode bùng nỗ và phát triển khá mạnh cho đến hiện tại do tốc độ đọc, độ chính xác và các tính năng ưu việt khác. Tuy nhiên, với sự hạn chế về nhiều mặt đặt biệt là về khối lượng thông tin mà BarCode có thể lưu trữ (chỉ khoảng 20 số hoặc chữ số) nên các loại mã 2-D đã được sinh ra để giải quyết những vấn đề đó.

Barcode *2D Code with stacked bar codes* *2D Code (matrix type)*

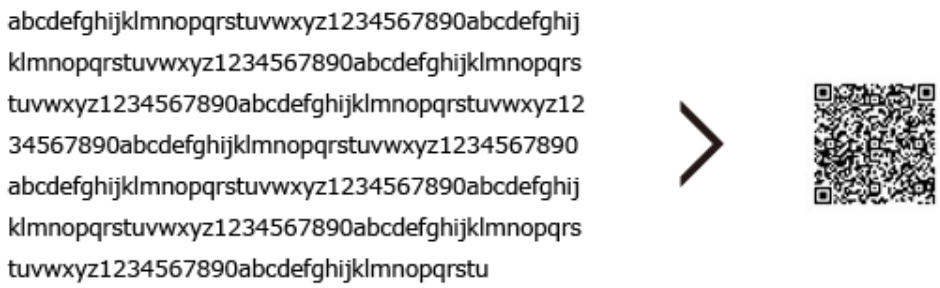
Mã 2D (Two-Dimensional) ban đầu thật chất là những mã Barcode xếp chồng lên nhau tạo thành (stacked Barcode) và sau đó phát triển thêm thành mã nhiều loại mã 2D khác nhau với phương pháp ma trận để lưu trữ nhiều thông tin hơn. Trong đó có nhiều loại mã được sữ dụng phổ biến hiện nay trong các lĩnh vực công nghệ khoa học như : PDF417, DataMatrix, Mã Maxi... nhưng thông dụng hơn cả là mã QRCode.

Hệ thống mã QR được Denso Wave phát minh măm 1994. Mục đích chính là theo dõi xe cộ trong quá trình sản xuất; nó được thiết kể để cho phép quét các bộ phận với tốc độ cao. Mặc dù những ứng dụng ban đầu chỉ để theo dõi các bộ phận của xe, nhưng hiện nay mã QR được ứng dụng trong nhiều ngữ cảnh khác nhau bao gồm cả các ứng dụng theo dõi thương mại và ứng dụng hướng tới sự tiện lợi cho những người sử dụng điện thoại di động. Mã QR có thể được sử dụng để hiển thị chữ cho người sử dụng, thời gian diễn ra một sự kiện, để thêm danh thiếp vCard vào thiết bị của

người sử dụng, để mở URI, để viết e-mail hoặc tin nhắn hay thậm chí là thông tin định vị vị trí địa lý. Người sử dụng có thể tạo và in mã QR của riêng họ cho những người khác quét và sử dụng để ghé thăm một trong các trang phải trả tiền và miễn phí thông qua mã QR. Nó hiện trở thành một trong những kiểu sử dụng nhiều nhất trong nhóm mã vạch hai chiều.

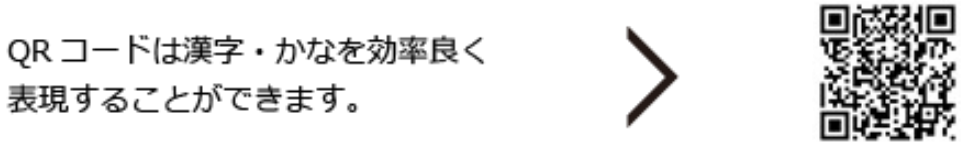
3. Các tính năng quan trọng của QR Code.

- QR code có thể lưu giữ thông tin chứa vài chục đến vài nghìn ký tự và chữ số vượt trội hơn nhiều so với Barcode truyền thống chỉ lưu trữ được khoảng 20 kí tự. Tối đa 7,089 ký tự có thể được mã hoá trong một biểu tượng.



Biểu tượng Mã QR có kích thước này có thể mã hoá 300 ký tự chữ và số.

- QR Code có khả năng xử lý nhiều loại dữ liệu phức tạp, chẳng hạn như các ký tự số và chữ, Kanji, Hiragana, chữ Khmer, chữ Hán, chữ Hebrew... ký hiệu, mã nhị phân và control codes.



Một đoạn chữ tiếng Nhật được mã hóa bởi mã QR

***Khả năng lưu trữ dữ liệu của mã QR**

- Số đơn thuần: Tối đa 7.089 kí tự
- Số và chữ cái: Tối đa 4.296 kí tự
- Số nhị phân (8 bit): Tối đa 2.953 byte
- Kanji/Kana: Tối đa 1.817 kí tự

- Kích thước nhỏ gọn tiện dụng hơn BarCode : Vì QR Code mang thông tin theo chiều ngang và cả chiều dọc nên QR Code có khả năng mã hoá cùng một lượng dữ liệu trong khoảng một phần mười kích thước so với BarCode. Mặc

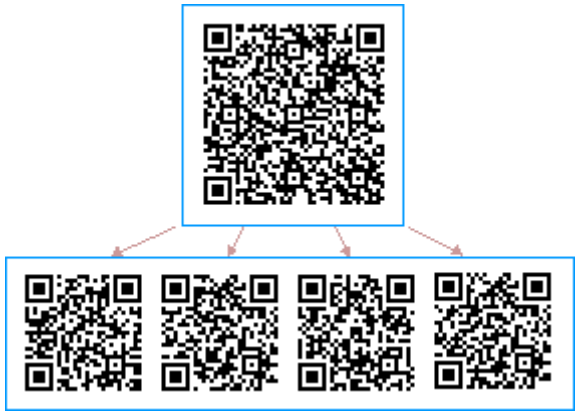
khác thì mã QR còn có dạng Vi mã QR (Micro QRcode) với kích thước rất nhỏ và tiện dụng.

Kích thước BarCode so với QRCode khi lưu cùng 1 dữ liệu

- Dù nằm ở vị trí nào dù ngược – xuôi – ngang – dọc thì việc đọc code vẫn rất dễ dàng mà không cần phải xoay chỉnh thiết bị đọc hay mã code cho đúng định dạng. Khả năng này có được chủ yếu là nhờ cấu trúc của mã QR có các hoa văn định vị nên mã QR có thể được đọc ở 360°.

Cấu trúc cơ bản của 1 mã QR

- Tính năng Structured Append: tính năng này cho phép Mã QR có thể được chia thành nhiều vùng dữ liệu. Ngược lại, thông tin được lưu trữ trong nhiều biểu tượng Mã QR có thể được tái tạo lại dưới dạng các ký hiệu dữ liệu đơn.



Nhiều kí tự số của 1 mã QR được chia thành nhiều mã QR

- Ngoài ra QR còn được phát triển thành nhiều loại với nhiều ứng dụng riêng đang được sữ dụng trong khá nhiều lĩnh vực qua những đặt tính riêng chẳng hạn như Micro QR Code, IQR Code, SQR code, Frame QR...

4. Những hạn chế

Bên cạnh những tính năng vô cùng hửu ích mã QR cũng có một vài hạn chế rất nhỏ như sau:

- Phải có thiết bị đọc mã QR hoặc smartphone có phần mềm hổ trợ tính năng đọc mã QR.

- Mức độ phổ biến trong thực tế so với Barcode còn thấp hơn khá nhiều lần mặc khác các doanh nghiệp, tổ chức cá nhân đã khá hài lòng với Barcode nên sẽ chưa sẳn sàng bỏ ra chi phí đầu tư cho các thiết bị đọc mã QRcode.

5. Cách tạo ra mã QR Code

Các bước cơ bản trong thuật toán tạo mã QR code:

- *Bước 1: Data Analysis*

Chuẩn QR có bốn chế độ mã hóa văn bản: số, chữ số, nhị phân, và chữ Kanji. Mỗi chế độ mã hoá văn bản như một chuỗi nhị phân (1 và 0), nhưng lại sử dụng một phương pháp khác nhau để chuyển đổi văn bản thành chuỗi nhị phân, và mỗi phương pháp mã hóa được tối ưu hoá để mã hóa dữ liệu với chuỗi ngắn nhất có thể. Do đó, bước đầu tiên của chúng ta là thực hiện phân tích dữ liệu để xác định xem văn bản của chúng ta có thể được mã hoá ở chế chỉ số, chữ số, nhị phân, hoặc chữ Kanji, sau đó chọn chế độ tối ưu cho văn bản của chúng ta.

- *Bước 2: Data Encoding*

Bây giờ chúng ta đã chọn chế độ mã hoá thích hợp cho văn bản của chúng ta, bước tiếp theo là mã hoá văn bản. Kết quả của bước này là một chuỗi các bit được chia thành các từ mã dữ liệu có chiều dài 8 bit.

VD: Đoạn chử Hello World được mã hóa với QR code cho ra kết quả:

00100000 01011011 00001011 01111000 11010001 01110010 11011100
01001101 01000011 01000000 11101100 00010001 11101100+

- *Bước 3: Error Correction Coding*

Mã QR sử dụng bộ sửa lỗi Reed-Solomon . Điều này có nghĩa là sau khi chúng ta tạo một chuỗi nhị phân dữ liệu đại diện cho văn bản của mình, chúng ta phải sử dụng các chuỗi này để tạo mã số hiệu chỉnh lỗi bằng cách sử dụng một quá trình gọi là Reed-Solomon error correction.

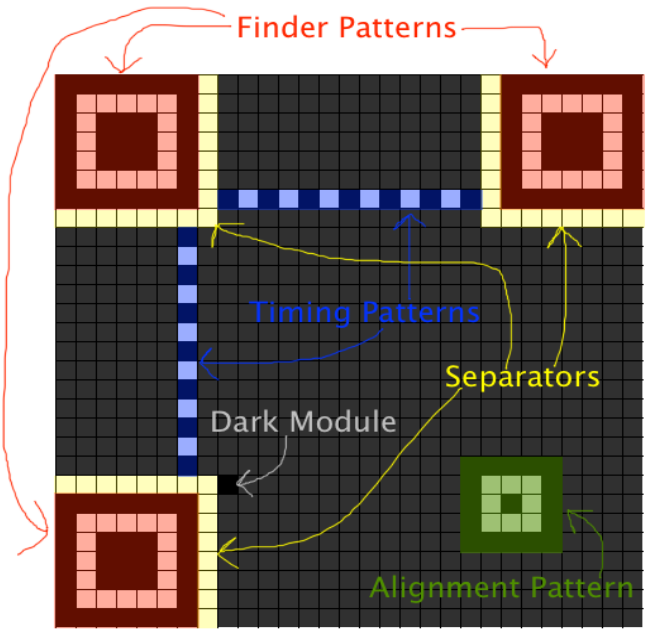
Máy quét QR đọc cả mã số dữ liệu và từ mã hoá sửa lỗi. Bằng cách so sánh cả hai , máy quét có thể xác định xem nó đọc dữ liệu có chính xác hay không, và nó có thể sửa lỗi nếu đọc dữ liệu không chính xác.

- *Bước 4: Structure Final Message*

Các dữ liệu mã hoá và sửa lỗi được tạo ra trong các bước trước đây phải được sắp xếp theo thứ tự thích hợp. Đối với mã QR lớn, dữ liệu và mã hiệu chỉnh lỗi được tạo ra trong các khối, và các khối này phải được xen kẽ theo các QR code specification.

- *Bước 5: Module Placement in Matrix*

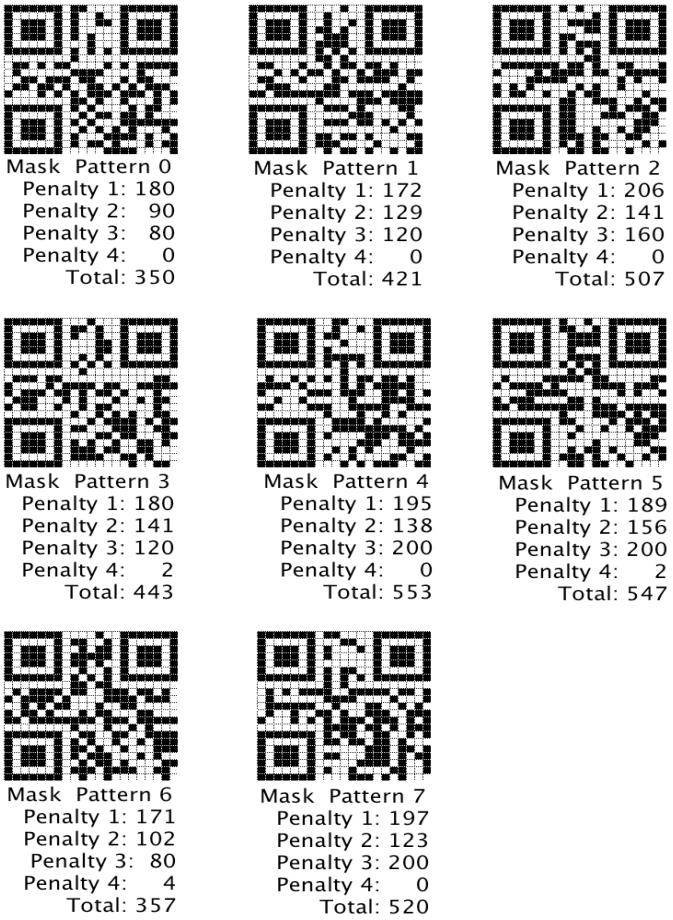
Sau khi tạo mã dữ liệu và mã hiệu chỉnh lỗi và sắp xếp chúng theo đúng thứ tự, chúng ta phải đặt các bit trong ma trận mã QR. Các từ mã được sắp xếp trong ma trận một cách cụ thể. Trong bước này, chúng ta cũng sẽ đặt các patterns cho tất cả mã QR, chẳng hạn như các boxes ở ba góc



- Finder patterns(FP): Là ba khối ở các góc của mã QR ở trên cùng bên trái, trên cùng bên phải và dưới cùng bên trái.
- Separators: Là các khoảng trắng bên cạnh FP.
- Alignment patterns: Tương tự như các FP, nhưng nhỏ hơn, và được đặt trong suốt.
- Timing patterns: Là các đường chấm chấm kết nối các FP.
- Dark module: Là những module màu đen.

- *Bước 6: Data Masking*

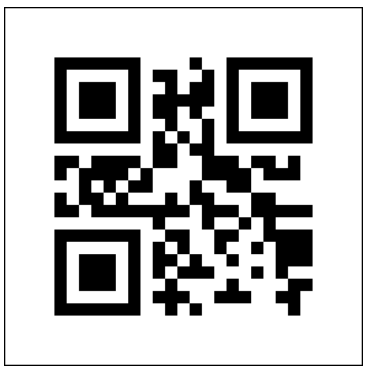
Một số patterns trong ma trận mã QR có thể khiến máy quét mã QR khó đọc mã chính xác. Để chống lại việc này, QR Code Specification đã xác định 8 mask pattern, mỗi mã này sẽ thay đổi mã QR theo một particular pattern. Chúng ta phải xác định pattern nào trong số các mask pattern này sẽ dẫn đến mã QR có ít đặc điểm không mong muốn. Điều này được thực hiện bằng cách đánh giá mỗi ma trận được che dấu dựa trên bốn penalty rules. Mã QR cuối cùng của bạn phải sử dụng mẫu mask pattern dẫn đến penalty score thấp nhất.



Những hình ảnh trên cho thấy tám mã QR, mỗi hình một mask pattern. Tất cả tám mã QR trong ví dụ này mã hóa cùng một dữ liệu. Như hình trên, mask pattern có điểm số thấp nhất là mask pattern 0. Do đó, trong ví dụ này, bộ mã hóa QR nên sử dụng mask pattern 0 khi xuất ra mã QR cuối cùng.

- *Bước 7: Định dạng và Thông tin Phiên bản*

Bước cuối cùng là thêm định dạng và thông tin về phiên bản vào mã QR bằng cách thêm các điểm ảnh trong các khu vực particular của phần mã để trống trong các bước trước đó. Các điểm ảnh định dạng xác định mức độ hiệu chỉnh lỗi và mask pattern được sử dụng trong mã QR này.



Mã QR đã được tạo hoàn tất

III. Công việc liên quan.

1. Thêm các thư viện cần thiết.

- Thực hiện thêm các thư viện hỗ trợ cần dùng cho ứng dụng.

2. Thiết kế giao diện.

- Thiết kế, mô phỏng giao diện thân thiện với người dùng.
- Giao diện đồng nhất màu sắc, kiểu thiết kế giữa các cửa sổ.
- Mang lại trải nghiệm tốt cho người dùng.

3. Thiết kế các lớp ứng dụng.

- Phân tích yêu cầu ứng dụng tạo, liên kết giữa các lớp.
- Đồng bộ hoạt động giữa các lớp ứng dụng.
- Tổng quát hóa, khai thác các lớp.

4. Thực hiện các hàm xử lí thông tin QR code.

- Xây dựng các hàm có chức năng giải mã các QR Code dựa trên các thư viện có sẵn.
- Xây dựng các hàm có chức năng mã hóa dữ liệu, tạo QR Code từ dữ liệu người dùng.

5. Thiết kế cơ sở dữ liệu, truy xuất, lưu thông tin vào cơ sở dữ liệu.

- Tạo cơ sở dữ liệu lưu trữ thông tin lịch sử của các QR code đã quét.
- Tạo các lớp, hàm thực hiện thêm, xóa, sửa dữ liệu. Truy vấn thông tin từ cơ sở dữ liệu.

6. Cài đặt kiểm tra.

- Chạy thử ứng dụng kiểm tra hoạt động, tính đúng đắn của ứng dụng.
- Kiểm tra các lỗi, hoạt động của ứng dụng.

IV. Phương pháp thực hiện.

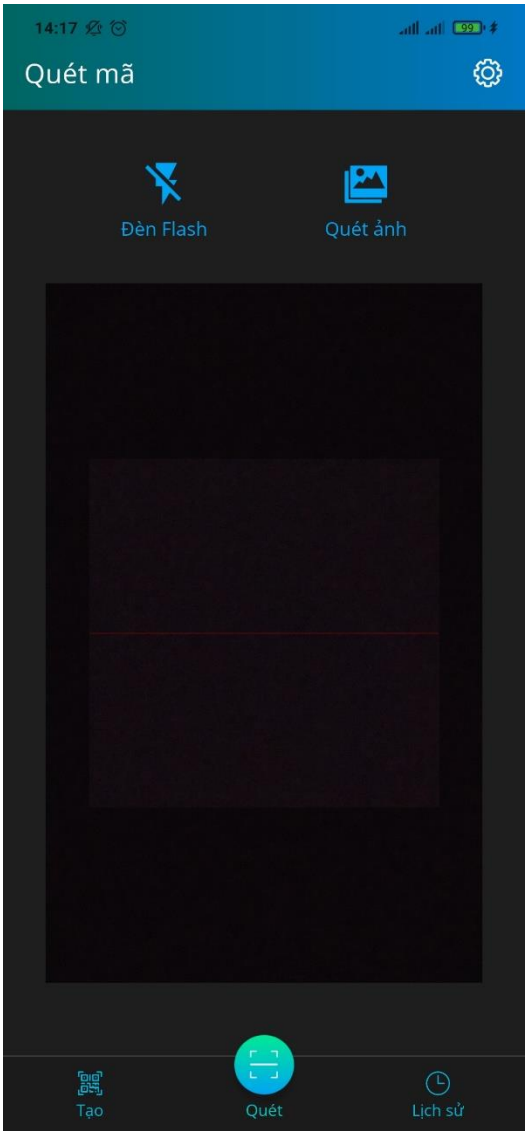
1. Thêm các thư viện cần thiết.

```
dependencies {
    def room_version = "2.1.0-alpha03"

    implementation fileTree(include: ['*.jar'], dir: 'libs')

    // Support library
    implementation 'androidx.appcompat:appcompat:1.0.2'
    implementation 'com.google.android.material:material:1.0.0'

    // Constraint layout
    implementation
'androidx.constraintlayout:constraintlayout:1.1.3'

    // Glide
    implementation 'com.github.bumptech.glide:glide:4.8.0'

    // Room
    implementation "androidx.room:room-runtime:$room_version"
    annotationProcessor "androidx.room:room-
compiler:$room_version"
    implementation "androidx.room:room-rxjava2:$room_version"

    /*
    * Reactive Extensions (Rx)
    * */
```

```
    implementation 'io.reactivex.rxjava2:rxandroid:2.1.0'
    implementation 'io.reactivex.rxjava2:rxjava:2.2.4'

    // MultiDex
    implementation 'androidx.multidex:multidex:2.0.1'

    // Bar code generation
    implementation 'com.google.zxing:core:3.3.0'
    implementation('com.journeyapps:zxing-android-
embedded:3.6.0') { transitive = false }

    // Pdf generation
    implementation 'com.itextpdf:itextg:5.5.10'

    // Google Admob
    implementation 'com.google.android.gms:play-services-
ads:17.1.2'
    implementation 'com.github.pwittchen:reactivenetwork-
rx2:3.0.2'
    implementation 'com.agilie:swipe2delete:1.0'


    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test:runner:1.1.0'
    androidTestImplementation 'androidx.test.espresso:espresso-
core:3.1.0'
}
```

2. Thiết kế giao diện.

- Giao diện chính của ứng dụng.
- Ứng dụng có 3 mục chính với các chức năng như Quét, Tạo QR Code, Lịch sử.

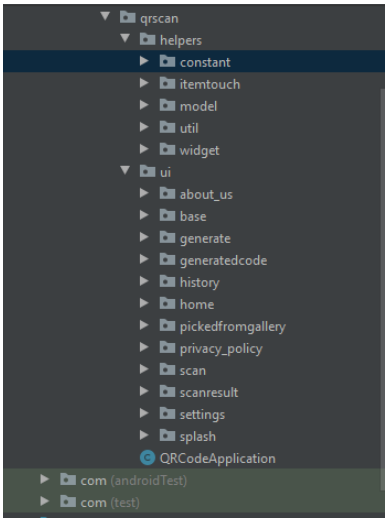


- Các layout cần có của ứng dụng.

STT	Tên layout	Chức năng
1	activity_about_us.xml	Chứa thông tin tác giả
2	activity_generated_code.xml	Giao diện kết quả tạo QR code
3	activity_home.xml	Màn hình chính của App
4	activity_picked_from_gallery.xml	Chọn ảnh từ bộ nhớ
5	activity_privay_policy.xml	Giao diện Policy
6	activity_scan_result.xml	Hiển thị kết quả scan
7	activity_settings.xml	Giao diện cài đặt
8	activity_splash.xml	Giao diện khởi động App

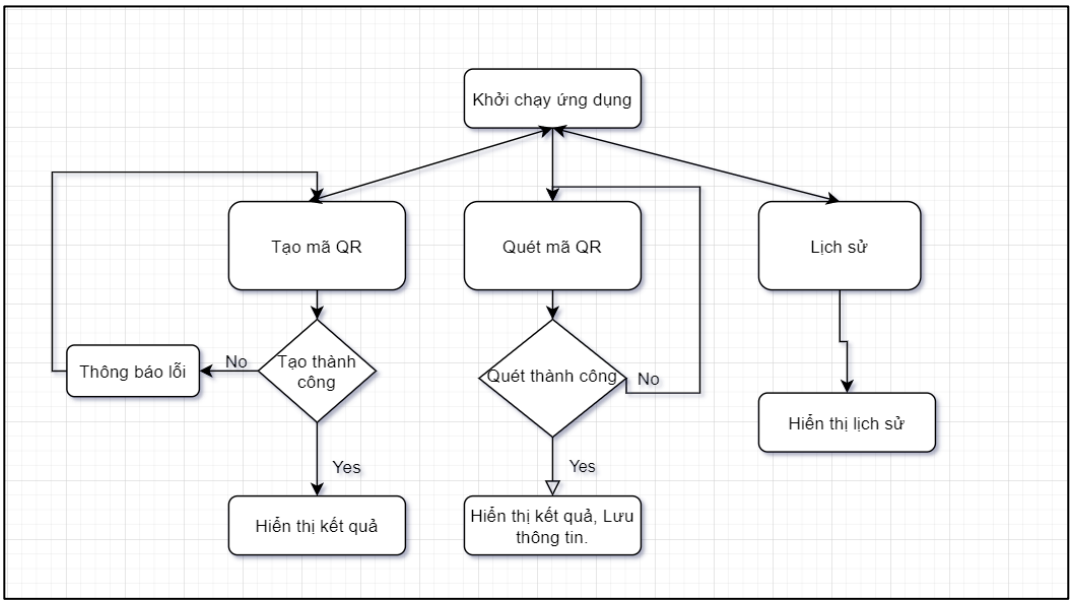
3. Thiết kế các lớp ứng dụng.

- Xây dựng các lớp ứng dụng tương ứng với yêu cầu.
- Chia thành 2 nhóm chính: nhóm hỗ trợ các chức năng, nhóm giao diện



4. Thực hiện các hàm xử lí thông tin QR code.

- Mô hình ứng dụng:



- Hàm thực hiện quét mã QR Code: Khi bắt được sự kiện mã được quét, ứng dụng sẽ gọi hàm thực hiện phát âm thanh cho người dùng biết việc quét thực hiện hoàn tất. Sau đó hiện thị thông tin kết quả cho người dùng, dừng việc quét lại, đồng thời lưu mã vào lịch sử.

```
private void doScan() {
    mBarcodeView.decodeSingle(new BarcodeCallback() {
        @Override
        public void barcodeResult(BarcodeResult result) {
            mBarcodeView.pause();
            mBeepManager.playBeepSoundAndVibrate();

            if (result != null
                    && !TextUtils.isEmpty(result.getText())
                    &&
!TextUtils.isEmpty(result.getBarcodeFormat().name())) {

                Code code;
```

```
                if (result.getBitmap() != null) {
                    int typeIndex =
result.getBarcodeFormat().name().toLowerCase().startsWith("qr")
                            ? Code.QR_CODE : Code.BAR_CODE;
                    String type =
getResources().getStringArray(R.array.code_types)[typeIndex];

                    File codeImageFile =
FileUtil.getEmptyFile(mContext, AppConstants.PREFIX_IMAGE,
                            String.format(Locale.ENGLISH,
getString(R.string.file_name_body),
                                    type.substring(0, type.indexOf("
Code")),

String.valueOf(System.currentTimeMillis())),
                            AppConstants.SUFFIX_IMAGE,
                            Environment.DIRECTORY_PICTURES);

                    if (codeImageFile != null) {
                        try (FileOutputStream out = new
FileOutputStream(codeImageFile)) {

result.getBitmap().compress(Bitmap.CompressFormat.PNG, 100, out);

                            code = new Code(result.getText(),

result.getBarcodeFormat().name().toLowerCase().startsWith("qr")
                                            ? Code.QR_CODE :
Code.BAR_CODE,
                                    codeImageFile.getPath(),
result.getResult().getTimestamp());
                        } catch (IOException e) {
                            if (!TextUtils.isEmpty(e.getMessage())) {
                                Log.e(getClass().getSimpleName(),
e.getMessage());
                            }

                            code = new Code(result.getText(),

result.getBarcodeFormat().name().toLowerCase().startsWith("qr")
                                            ? Code.QR_CODE :
Code.BAR_CODE, result.getResult().getTimestamp());
                        }
                    } else {
                        code = new Code(result.getText(),

result.getBarcodeFormat().name().toLowerCase().startsWith("qr")
                                        ? Code.QR_CODE :
Code.BAR_CODE, result.getResult().getTimestamp());
                    }
                } else {
                    code = new Code(result.getText(),

result.getBarcodeFormat().name().toLowerCase().startsWith("qr")
                                    ? Code.QR_CODE : Code.BAR_CODE,
result.getResult().getTimestamp());
                }
```

```
                Intent intent = new Intent(mContext,
ScanResultActivity.class);
                intent.putExtra(IntentKey.MODEL, code);
                startActivity(intent);
            } else {
                mBarcodeView.resume();
                doScan();
                Toast.makeText(mContext,
getString(R.string.error_occured_while_scanning),
                        Toast.LENGTH_SHORT).show();
            }
        }

        @Override
        public void possibleResultPoints(List<ResultPoint>
resultPoints) {

        }
    });
}
```

- Hàm thực hiện việc tạo mã QR code. Ứng dụng đưa ra hai lại chọn cho người dùng, từ đó tùy theo nhu cầu người dùng có thể lựa chọn tạo ra Bar Code hay QR Code với nội dung mong muốn. Với mỗi loại mã, ứng dụng sẽ kiểm tra nội dung người dùng nhập với định dạng, kích thước tương ứng với từng loại mã.

```
-  private void loadQRCode() {
        Intent intent = getIntent();

        if (intent != null) {
            Bundle bundle = intent.getExtras();

            if (bundle != null &&
    bundle.containsKey(IntentKey.MODEL)) {

    setCurrentCode(bundle.getParcelable(IntentKey.MODEL));
            }
        }

        if (getCurrentCode() != null) {
            ProgressDialogUtil.on().showProgressDialog(this);


    mBinding.textViewContent.setText(String.format(Locale.ENGLISH,
                    getString(R.string.content),
    getCurrentCode().getContent()));


    mBinding.textViewType.setText(String.format(Locale.ENGLISH,
    getString(R.string.code_type),

    getResources().getStringArray(R.array.code_types)[getCurrentCode
    ().getType()]));

            BarcodeFormat barcodeFormat;
            switch (getCurrentCode().getType()) {
                case Code.BAR_CODE:
                    barcodeFormat = BarcodeFormat.CODE_128;
```

```
                break;

            case Code.QR_CODE:
                barcodeFormat = BarcodeFormat.QR_CODE;
                break;

            default:
                barcodeFormat = null;
                break;
        }

        if (barcodeFormat != null) {
            try {
                BarcodeEncoder barcodeEncoder = new
BarcodeEncoder();
                Bitmap bitmap =
barcodeEncoder.encodeBitmap(getCurrentCode().getContent(),
                        barcodeFormat, 1000, 1000);

mBinding.imageViewGeneratedCode.setImageBitmap(bitmap);
                mCurrentGeneratedCodeBitmap = bitmap;
            } catch (Exception e) {
                if (!TextUtils.isEmpty(e.getMessage())) {
                    Log.e(getClass().getSimpleName(),
e.getMessage());
                }
            }
        }

        ProgressDialogUtil.on().hideProgressDialog();
    }
}
```

- Sau khi tạo mã ứng dụng sẽ tiến hành hiển thị kết quả cho người dùng. Người dùng có thể lựa chọn việc lưu mã vào thiết bị, chia sẽ mã bằng những ứng dụng chia sẽ khác có trong SmartPhone. Với lựa chọn lưu mã, ứng dụng sẽ kiểm tra xem mã đã lưu trước đó hay chưa và xuất thông báo cho người dùng. Với sự kiện lưu mã thất bại ứng dụng sẽ trả về thông báo tạo mã thất bại.

```
- public void onRequestPermissionsResult(int requestCode, @NonNull
String[] permissions, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions,
grantResults);

    boolean isValid = true;

    for (int i = 0; i < permissions.length; i++) {
        if (grantResults[i] !=
PackageManager.PERMISSION_GRANTED) {
            isValid = false;
        }
    }

    switch (requestCode) {
        case REQUEST_CODE_TO_SAVE:
            if (isValid) {
                if (getCurrentCodeFile() == null) {
```

```
                    storeCodeImage(true);
                } else {
                    Toast.makeText(this,
getString(R.string.generated_qr_code_already_exists),
                            Toast.LENGTH_SHORT).show();
                }
            }
            break;

        case REQUEST_CODE_TO_PRINT:
            if (isValid) {
                if (getCurrentPrintedFile() == null) {
                    storeCodeDocument();
                } else {
                    Toast.makeText(this,
getString(R.string.generated_qr_code_already_exists),
                            Toast.LENGTH_SHORT).show();
                }
            }
            break;

        case REQUEST_CODE_TO_SHARE:
            if (isValid) {
                if (getCurrentCodeFile() == null) {
                    storeCodeImage(false);

                    if (getCurrentCodeFile() != null) {
                        shareCode(getCurrentCodeFile());
                    } else {
                        Toast.makeText(this,
getString(R.string.failed_to_share_the_code),
Toast.LENGTH_SHORT).show();
                    }
                } else {
                    shareCode(getCurrentCodeFile());
                }
            }
            break;

        default:
            break;
    }
}
```

5. Thiết kế cơ sở dữ liệu, truy xuất, lưu thông tin vào cơ sở dữ liệu.

- Ứng dụng sử dụng Room Database.
- Room Database được phát triển và cải tiến từ sqlite. Room database giúp đơn giản hoá việc code,và giảm thiểu các công đoạn liên quan đến cơ sở dữ liệu. Bản chất Room database là abstract layer gồm cơ sở dữ liệu chuẩn SQLite được Android thông qua.Với 3 thành phần chính là: Database, DAO(Data Access Object) và entity. Mỗi thành phần đều có nhiệm vụ và chức năng riêng.

- Ứng dụng sử dụng một bảng có tabble_name = CODES, với các cột id, mContent, mtype, mCodeImagePath, mTimeStamp tương ứng với các thông tin của mã là id, nội dung, loại mã, đường dẫn lưu mã, thời gian tạo mã.
- Thực hiện các thao tác với dữ liệu như thêm, xóa, sửa, thay thế.

```
-  public CodeDao_Impl(RoomDatabase __db) {
     this.__db = __db;
     this.__insertionAdapterOfCode = new
   EntityInsertionAdapter<Code>(__db) {
       @Override
       public String createQuery() {
         return "INSERT OR REPLACE INTO
   `CODES`(`id`,`mContent`,`mType`,`mCodeImagePath`,`mTimeStamp`)
   VALUES (nullif(?, 0),?,?,?,?)";
       }

       @Override
       public void bind(SupportSQLiteStatement stmt, Code value) {
         stmt.bindLong(1, value.mId);
         if (value.getContent() == null) {
           stmt.bindNull(2);
         } else {
           stmt.bindString(2, value.getContent());
         }
         stmt.bindLong(3, value.getType());
         if (value.getCodeImagePath() == null) {
           stmt.bindNull(4);
         } else {
           stmt.bindString(4, value.getCodeImagePath());
         }
         stmt.bindLong(5, value.getTimeStamp());
       }
     };
     this.__deletionAdapterOfCode = new
   EntityDeletionOrUpdateAdapter<Code>(__db) {
       @Override
       public String createQuery() {
         return "DELETE FROM `CODES` WHERE `id` = ?";
       }

       @Override
       public void bind(SupportSQLiteStatement stmt, Code value) {
         stmt.bindLong(1, value.mId);
       }
     };
     this.__updateAdapterOfCode = new
   EntityDeletionOrUpdateAdapter<Code>(__db) {
       @Override
       public String createQuery() {
         return "UPDATE OR REPLACE `CODES` SET `id` = ?,`mContent`
   = ?,`mType` = ?,`mCodeImagePath` = ?,`mTimeStamp` = ? WHERE `id`
   = ?";
       }

       @Override
       public void bind(SupportSQLiteStatement stmt, Code value) {
         stmt.bindLong(1, value.mId);
```

```
        if (value.getContent() == null) {
          stmt.bindNull(2);
        } else {
          stmt.bindString(2, value.getContent());
        }
        stmt.bindLong(3, value.getType());
        if (value.getCodeImagePath() == null) {
          stmt.bindNull(4);
        } else {
          stmt.bindString(4, value.getCodeImagePath());
        }
        stmt.bindLong(5, value.getTimeStamp());
        stmt.bindLong(6, value.mId);
      }
    };
  }
```

6. Cài đặt và kiểm tra.

- Môi trường cài đặt: Các thiết bị Android từ version 4.2.x trở lên
- Kết quả: ứng dụng hoạt động ổn định, cho tốc độ phản hồi nhanh, kết quả tương đối chính xác. Một số dịch vụ không sử dụng được đối với các phiên bản Android quá cũ

V. Kết quả đạt được.

1. Kết quả.

- Tìm hiểu hoàn thành ứng dụng tương đối hoàn thiện. Ứng dụng có thể đáp ứng được tốt yêu cầu của người dùng.
- Áp dụng được các kiến thức đã học trong môn học, hiểu biết thêm một số thư viện mới trong lập trình ứng dụng Android.

2. Ưu điểm.

- Ứng dụng có tốc độ xử lí nhanh
- Giao diện dễ sử dụng, thân thiện với người dùng.
- Nhu cầu quét tạo,mã của người dùng đang tăng dần, nên tiềm năng phát triển của ứng dụng rất lớn. Phù hợp với đại đa số người dùng.
- Áp dụng công nghệ, phương pháp lập trình mới hiện đại giúp ứng dụng dễ dàng nâng cấp, sửa lỗi.

3. Nhược điểm.

- Chưa tối ưa với các phiên bản Android cũ(API<28)
- Giao diện người dùng còn chưa bắt mắt, thu hút người dùng.
- Chưa hỗ trợ hết tất cả các loại Code hiện nay.

4. Hướng mở rộng.

- Tối ưu hóa ứng dụng. Cải thiện tốc độ đọc, tạo mã, truy xuất thông tin.
- Đăng tải ứng dụng lên các trang tải ứng dụng như Play Store.
- Cải thiện giao diện UI/UX mang lại trải nghiệm tốt cho người dùng.
- Phát triển ứng dụng tương tự chạy trên iOS

VI. Kết luận:

Qua môn học, giúp có một cách nhìn tổng quan về lập trình ứng dụng di động. Tổng quan về các nền tảng di động như iOS, Android. Áp dụng các công nghệ, thư viện vào thực tiễn, lập trình được ứng dụng Android với yêu cầu mong muốn. Nhận thức được xu thế phát triển của các ứng dụng di động, từ đó tập trung tìm hiểu, nghiên cứu các công nghệ mới, tiếp thu ý kiến từ người dùng, tạo ra những ứng dụng có tính ứng dụng cao, mang lại hiệu quả tốt.

Tài liệu tham khảo

1. Mã QR: https://vi.wikipedia.org/wiki/M%C3%A3_QR
2. Zxing: https://github.com/zxing/zxing
3. Room DataBase https://developer.android.com/training/data-storage/room
4. QR Code Java https://viblo.asia/p/tao-ma-qr-code-trong-java-voi-zxing-4P856grLKY3
5. Stackoverflow https://stackoverflow.com/questions/8831050/android-how-to-read-qr-code-in-my-application