Visvesvaraya Technological University

Belagavi-590 018, Karnataka

A Mini Project Report on

"QR and Barcode Scanner"

Mini Project Report submitted in partial fulfilment of the requirement for the Java for Mobile Application Laboratory with Mini Project [18AIL68]

BACHELOR OF ENGINEERING
IN
ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Submitted by
Dheeraj N Kashyap [1JT20AI007]

Under the guidance of

Prof. Deepthi Das V
Assistant Professor
Department of Artificial Intelligence and Machine Learning

Prof. Archana V R
Assistant Professor
Department of Artificial Intelligence and Machine Learning

Department of Artificial Intelligence and Machine Learning
Jyothy Institute of Technology Tataguni,
Bengaluru-560082

Jyothy Institute of Technology

Tataguni, Bengaluru-560082

Department of Artificial Intelligence and Machine Learning

CERTIFICATE

Certified that the mini project work entitled **"QR and Barcode Scanner"** carried out by **Dheeraj N Kashyap** [**1JT20AI007**] **and Varun G** [**1JT20AI049**] bonafide students of Jyothy Institute of Technology, in partial fulfilment for the award of **Bachelor of Engineering** in **Artificial Intelligence and Machine Learning** department of the **Visvesvaraya Technological University**, **Belagavi** during the year **2022- 2023**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

Prof. Deepthi Das V	**Prof. Archana V R**	**Dr. Madhu B R**
Guide, Asst. Professor	Guide, Asst. Professor	Professor & HOD
Dept. of AI&ML	Dept. of AI&ML	Dept. of AI&ML

External Examiner Signature with Date:

1.

2.

ACKNOWLEDGEMENT

Firstly, we are very grateful to this esteemed institution **"Jyothy Institute of Technology**" for providing us an opportunity to complete our project.

We express our sincere thanks to our **Principal Dr. K Gopalakrishna** for providing us with adequate facilities to undertake this project.

We would like to thank **Dr. Madhu B R**, **Professor and Head of AI&ML** Department for providing for his valuable support.

We would like to thank our guides **Prof. Deepthi Das V and Prof. Archana V R, Asst. Professors** for their keen interest and guidance in preparing this work.

Finally, we would thank all our friends who have helped us directly or indirectly in this project.

Dheeraj N Kashyap [1JT20AI007]

DECLARATION

I, Dheeraj N Kashyap & Varun G, 6th Semester student of Artificial Intelligence and Machine Learning, Jyothy Institute of Technology , Bangalore - 560 082, declare that the Java for Mobile Application Laboratory with Mini Project [18AIL68] is successfully completed.

This report is submitted in partial fulfillment of the requirements for award of Bachelor Degree in Artificial Intelligence and Machine Learning, during the academic year 2022-2023.

Date : 26-06-2023

Place : Bengaluru

Dheeraj N Kashyap [1JT20AI007]

ABSTRACT

This project focuses on developing an Android application that combines QR code and barcode scanning functionalities within a single, user-friendly interface. While existing apps predominantly offer QR code scanning, this novel app aims to bridge the gap by incorporating barcode recognition capabilities.

The proposed Android app provides a seamless solution for users to scan and decode various types of codes, offering enhanced convenience and versatility. By integrating both QR code and barcode scanning features, users can effortlessly extract information from a broader range of codes. The app utilizes the device's camera to capture codes and employs advanced image processing algorithms to accurately analyze and interpret the encoded data.

To ensure a smooth user experience, the app features an intuitive and user-friendly interface. Upon launch, users are presented with a live camera preview, enabling real-time code scanning. The app utilizes optimized algorithms to quickly identify and differentiate between QR codes and barcodes. Instant feedback through visual indicators and audio signals ensures successful code recognition.

In conclusion, the QR and Barcode Scanner Android App represents an innovative solution that combines QR code and barcode scanning within a user-friendly interface. By providing a comprehensive and versatile scanning experience, the app enhances the convenience and efficiency of code recognition. Advanced image processing algorithms ensure accurate and rapid decoding, empowering users to extract information from a wide range of codes. The app's intuitive interface and additional functionalities further enhance user experience, making it an indispensable tool for everyday code scanning and management.

TABLE OF CONTENT

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

1.1 INTRODUCTION TO ANDROID:

Android is an open-source operating system developed by Google primarily for mobile devices. It is built on the Linux kernel and designed to be used on touchscreen devices such as smartphones and tablets. Android offers a vast ecosystem of applications and services through the Google Play Store. It provides users with a customizable and user-friendly interface, allowing them to personalize their devices according to their preferences. Android supports a wide range of features, including seamless multitasking, notifications, voice commands, and access to various Google services. With its widespread adoption and continuous updates, Android has become one of the most popular and influential mobile operating systems worldwide.

1.2 INTRODUCTION TO ANDROID STUDIO:

Android Studio is the official integrated development environment (IDE) for developing Android applications. Developed by Google, it provides developers with a comprehensive set of tools and features to create, test, and debug Android apps efficiently. Android Studio offers a user-friendly interface that streamlines the app development process, allowing developers to focus on coding and building innovative mobile applications. It includes a code editor with intelligent code completion, syntax highlighting, and debugging capabilities. The IDE also integrates seamlessly with the Android Software Development Kit (SDK) and provides access to various libraries, templates, and emulators for efficient app development. Android Studio supports multiple programming languages, including Java and Kotlin, and offers robust tools for designing user interfaces and managing app resources. With its powerful capabilities and continuous updates, Android Studio remains the go-to choice for developers to create cutting-edge Android applications.

1.3 INTRODUCTION TO QR & BARCODE:

QR codes and barcodes are two widely used technologies for encoding and decoding information. QR codes are square-shaped codes that can store various types of data and are quickly readable by mobile devices. Barcodes consist of parallel lines or bars and are commonly found on product packaging, encoding information for retail and logistics purposes. Both QR codes and barcodes offer efficient ways to access and share information, playing crucial roles in inventory management, ticketing, payment systems, and advertising. Their simplicity and versatility have made them essential tools for quick and accurate data capture and exchange.

1.4 SCOPE AND IMPORTANCE OF WORK:

QR codes and barcodes have extensive scope and significant importance across various industries and applications. Their ability to encode and store information in a compact format makes them invaluable tools for efficient data capture. In inventory management, QR codes and barcodes enable accurate product tracking, reducing errors and improving operational efficiency. Retailers benefit from streamlined checkout processes and pricing accuracy, while logistics and supply chain management benefit from real-time tracking and enhanced visibility throughout the supply chain. Additionally, QR codes have emerged as powerful marketing tools, engaging customers through interactive experiences and providing valuable insights for targeted campaigns. In healthcare, ticketing systems, event management, identification cards, and loyalty programs, QR codes and barcodes find diverse applications, reflecting their versatility and widespread usage.

The significance of QR codes and barcodes lies in their ability to streamline data capture, improve operational processes, enhance customer experiences, and facilitate seamless information exchange. Their integration into various industries and applications promotes efficiency, accuracy, and convenience. As technology continues to evolve, the scope and importance of QR codes and barcodes are expected to expand further, driving innovation and enabling new possibilities in data management, customer engagement, and supply chain optimization.

.

CHAPTER 2

PROBLEM STATEMENT AND INPUT OUTPUT DESIGN

2. PROBLEM STATEMENT AND INPUT OUTPUT DESIGN

2.1 THEORY OF QR AND BARCODES:

QR codes, developed by Denso Wave in 1994, revolutionized barcode technology by enabling the storage of large amounts of data and high-speed decoding. Originally used for tracking parts in vehicle manufacturing, QR codes gained popularity in Japan for their ability to be scanned quickly using barcode readers and mobile phones. With the widespread adoption of smartphones equipped with cameras, QR codes became widely used for accessing websites, making payments, and initiating various actions. Today, QR codes are internationally recognized and utilized across industries such as marketing, advertising, retail, logistics, and healthcare. Their versatility, data capacity, and ease of scanning have made QR codes an essential tool for bridging the physical and digital worlds.

Barcodes have a significant history, with the development of the Universal Product Code (UPC) system in the 1970s marking a turning point. Barcodes, consisting of parallel lines or bars, revolutionized industries by enabling automated product identification and inventory management. They have become standardized across various sectors, serving as a crucial tool for data capture, supply chain management, and improving operational efficiency. Barcodes continue to play a vital role today, facilitating accurate tracking and streamlining processes in retail, logistics, healthcare, and other domains.

2.2 PROBLEM STATEMENT:

Current barcode scanning applications primarily focus on either QR codes or traditional barcodes, resulting in limited functionality for users who require both code recognition capabilities in a single app. The lack of a comprehensive solution hinders users from efficiently accessing and decoding information from various types of codes. Therefore, there is a need to develop an Android application that

seamlessly integrates QR code and barcode scanning functionalities, providing users with a versatile and user-friendly platform for capturing and decoding different codes. The application should offer fast and accurate recognition, robust scanning capabilities, and additional features such as code storage and sharing, aiming to enhance convenience, productivity, and information accessibility for users across industries.

2.3 INPUT OUTPUT DESIGN:

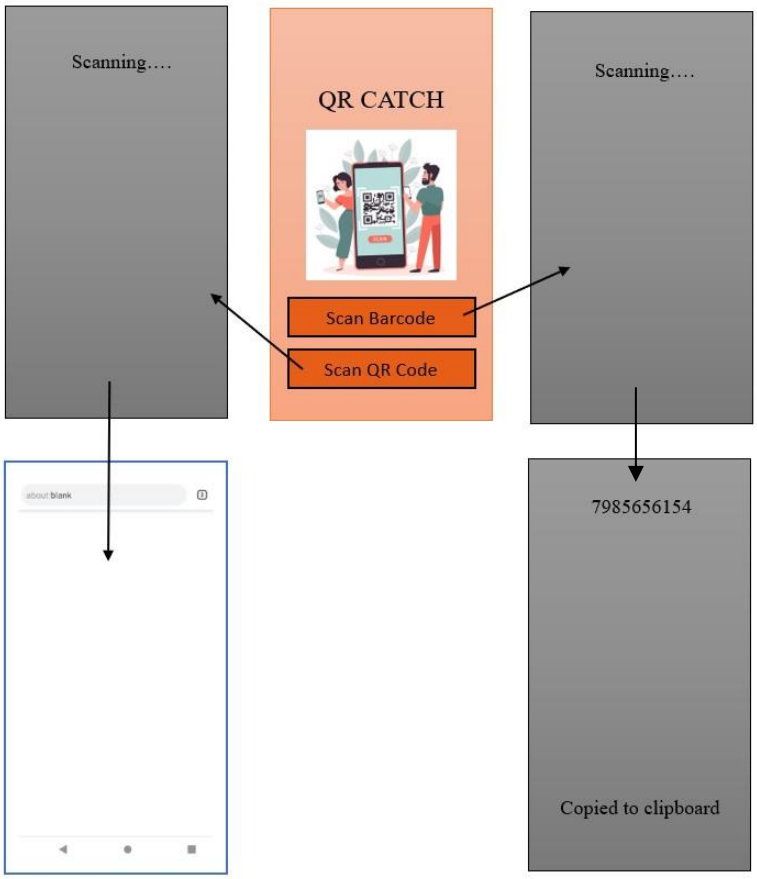

Fig 2.3 Input output Design

The app interface features two primary buttons: "Scan Barcode" and "Scan QR Code." When the user selects the "Scan Barcode" option, the app triggers the device's camera to activate. This enables the user to scan a barcode by capturing an image of it. The app analyzes the image in real-time, utilizing advanced barcode recognition algorithms to identify and decode the barcode. The decoded value is then promptly displayed on the screen, allowing the user to access the encoded information effortlessly. To further enhance usability, the app also automatically copies the decoded value to the device's clipboard. This feature enables users to conveniently paste and utilize the barcode data in other applications or processes without any additional steps.

Alternatively, when the user taps on the "Scan QR Code" button, the app initiates the camera functionality once again. This time, the app focuses on detecting and interpreting QR codes. As the camera captures the QR code, the app swiftly recognizes its content using specialized QR code decoding techniques. If the QR code contains a website URL, the app smoothly directs the user to the web browser, seamlessly opening the associated webpage. This unique aspect of the app enables users to effortlessly access web content directly from the scanned QR code, without the need for manual URL entry. For QR codes containing other types of information, such as contact details or text, the app may present the decoded content on the screen itself. This empowers users to view and interact with the extracted information as required.

In summary, the QR code and barcode scanning app simplifies the process of scanning and decoding barcodes, facilitating quick access to their encoded data. Moreover, it enriches the QR code scanning experience by seamlessly redirecting users to web content associated with the QR code, providing a seamless transition between physical objects and their digital counterparts. With its intuitive interface and efficient functionality, the app enhances user convenience and productivity when working with various types of codes, ensuring swift information extraction and effortless actions.

CHAPTER 3

XML CODE

(Front-end)

3.1 Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#FAD1B1"
    tools:context=".MainActivity"
    tools:showIn="@layout/activity_main">
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:layout_margin="10dp">

        <TextView
            android:textSize="40sp"
            android:textStyle="bold"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textAlignment="center"
            android:textColor="#F46538"
            android:text="QR CATCH"/>

        <ImageView
            android:layout_width="400dp"
            android:layout_height="300dp"
            android:src="@drawable/iiiii"
            android:layout_gravity="center_horizontal"
            android:padding="16dp"/>



        <Button
            android:id="@+id/buttonTakePicture"
            android:layout_width="match_parent"
            android:backgroundTint="#F46538"
            android:layout_height="wrap_content"
            android:outlineSpotShadowColor="@color/white"
            android:text="Scan BarCode" />
        <Button
```

```
            android:id="@+id/buttonScanBarcode"
             android:layout_width="match_parent"
             android:backgroundTint="#F46538"
             android:layout_height="wrap_content"
             android:layout_marginTop="10dp"
             android:outlineSpotShadowColor="@color/white"
             android:text="Scan QR code" />
    </LinearLayout>
</RelativeLayout>
```

3.2 Activity_Scanner_barcode.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ScannerBarcodeActivity">

    <SurfaceView
        android:id="@+id/surfaceView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_alignParentStart="true"
        android:layout_centerVertical="true" />


    <TextView
        android:id="@+id/txtBarcodeValue"
        android:layout_width="wrap_content"
        android:textAlignment="center"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginStart="25dp"
        android:layout_marginLeft="10dp"
        android:layout_marginTop="76dp"
        android:layout_marginEnd="25dp"
        android:layout_marginBottom="628dp"
        android:text="Scanning...."
        android:textColor="@color/black"
        android:textSize="20sp" />
</RelativeLayout>
```

3.3 AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-feature
        android:name="android.hardware.camera"
        android:required="false" />
    <uses-permission android:name="android.permission.CAMERA" />
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@drawable/qrcode_banner"
        android:label="@string/app_name"
        android:roundIcon="@drawable/qrcode_banner"
        android:supportsRtl="true"
        android:theme="@style/Theme.QRScan"
        tools:targetApi="31">
        <activity
            android:name=".PictureBarcodeActivity"
            android:exported="false" />
        <activity
            android:name=".ScannerBarcodeActivity"
            android:exported="false" />
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
              <action android:name="android.intent.action.MAIN" />

              <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

CHAPTER 4

JAVA CODE

(Back-end)

4.1 Mainactivity.java

```
package com.example.qr_scan;

import android.annotation.SuppressLint;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity implements
View.OnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        initComponents();
    }

    private void initComponents(){
        findViewById(R.id.buttonTakePicture).setOnClickListener(this);
        findViewById(R.id.buttonScanBarcode).setOnClickListener(this);
    }


    @SuppressLint("NonConstantResourceId")
    @Override
    public void onClick(View view) {
        int viewId = view.getId();
        if (viewId == R.id.buttonScanBarcode) {
            startActivity(new Intent(this, ScannerBarcodeActivity.class));
        } else if (viewId == R.id.buttonTakePicture) {
            startActivity(new Intent(this, PictureBarcodeActivity.class));
        }
    }
}
```

4.2 ScannerBarcodeActivity.java

```
package com.example.qr_scan;

import android.Manifest;
```

```
import android.content.ClipData;
import android.content.ClipboardManager;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.net.Uri;
import android.os.Bundle;
import android.util.SparseArray;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.widget.TextView;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import com.google.android.gms.vision.CameraSource;
import com.google.android.gms.vision.Detector;
import com.google.android.gms.vision.barcode.Barcode;
import com.google.android.gms.vision.barcode.BarcodeDetector;
import java.io.IOException;

public class ScannerBarcodeActivity extends AppCompatActivity {

    SurfaceView surfaceView;
    TextView textViewBarCodeValue;
    private BarcodeDetector barcodeDetector;
    private CameraSource cameraSource;
    private static final int REQUEST_CAMERA_PERMISSION = 201;
    String intentData = "";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_scanner_barcode);
        initComponents();
    }

    private void initComponents() {
        textViewBarCodeValue = findViewById(R.id.txtBarcodeValue);
        surfaceView = findViewById(R.id.surfaceView);
    }

    private void initialiseDetectorsAndSources() {
        Toast.makeText(getApplicationContext(), "QRcode scanner started",
Toast.LENGTH_SHORT).show();
        barcodeDetector = new BarcodeDetector.Builder(this)
```

```
                .setBarcodeFormats(Barcode.ALL_FORMATS)
                .build();

        cameraSource = new CameraSource.Builder(this, barcodeDetector)
                .setRequestedPreviewSize(1920, 1080)
                .setAutoFocusEnabled(true)
                .build();

        surfaceView.getHolder().addCallback(new SurfaceHolder.Callback() {
            @Override
            public void surfaceCreated(SurfaceHolder holder) {
                openCamera();
            }

            @Override
            public void surfaceChanged(SurfaceHolder holder, int format,
int width, int height) {
            }

            @Override
            public void surfaceDestroyed(SurfaceHolder holder) {
                cameraSource.stop();
            }
        });

        barcodeDetector.setProcessor(new Detector.Processor<Barcode>() {
            @Override
            public void release() {
                Toast.makeText(getApplicationContext(), "To prevent memory
leaks, the barcode scanner has been stopped", Toast.LENGTH_SHORT).show();
            }

            @Override
            public void receiveDetections(Detector.Detections<Barcode>
detections) {
                final SparseArray<Barcode> barCode =
detections.getDetectedItems();
                if (barCode.size() > 0) {
                    setBarCode(barCode);
                }
            }
        });
    }

    private void openCamera() {
        try {
```

```
            if
(ActivityCompat.checkSelfPermission(ScannerBarcodeActivity.this,
Manifest.permission.CAMERA) == PackageManager.PERMISSION_GRANTED) {
                cameraSource.start(surfaceView.getHolder());
            } else {

ActivityCompat.requestPermissions(ScannerBarcodeActivity.this, new
                        String[]{Manifest.permission.CAMERA},
REQUEST_CAMERA_PERMISSION);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void setBarCode(final SparseArray<Barcode> barCode) {
        textViewBarCodeValue.post(new Runnable() {
            @Override
            public void run() {
                intentData = barCode.valueAt(0).displayValue;
                textViewBarCodeValue.setText(intentData);
                redirectToWebPage(intentData);
            }
        });
    }

    private void redirectToWebPage(String url) {
        Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(url));
        startActivity(intent);
    }

    @Override
    protected void onPause() {
        super.onPause();
        cameraSource.release();
    }

    @Override
    protected void onResume() {
        super.onResume();
        initialiseDetectorsAndSources();
    }

    private void copyToClipBoard(String text) {
        ClipboardManager clipboard = (ClipboardManager)
getSystemService(Context.CLIPBOARD_SERVICE);
```

```
        ClipData clip = ClipData.newPlainText("QR code Scanner", text);
        clipboard.setPrimaryClip(clip);
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull
String[] permissions, @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
        if (requestCode == REQUEST_CAMERA_PERMISSION &&
grantResults.length > 0) {
            if (grantResults[0] == PackageManager.PERMISSION_DENIED)
                finish();
            else
                openCamera();
        } else
            finish();
    }
}
```

4.3 PictureBarcodeActivity.java

```
package com.example.qr_scan;

import android.Manifest;
import android.content.ClipData;
import android.content.ClipboardManager;
import android.content.Context;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.util.SparseArray;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;

import com.google.android.gms.vision.CameraSource;
import com.google.android.gms.vision.Detector;
import com.google.android.gms.vision.barcode.Barcode;
import com.google.android.gms.vision.barcode.BarcodeDetector;
```

```
import java.io.IOException;

public class PictureBarcodeActivity extends AppCompatActivity {

    SurfaceView surfaceView;
    TextView textViewBarCodeValue;
    private BarcodeDetector barcodeDetector;
    private CameraSource cameraSource;
    private static final int REQUEST_CAMERA_PERMISSION = 201;
    String intentData = "";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_scanner_barcode);
        initComponents();
    }

    private void initComponents() {
        textViewBarCodeValue = findViewById(R.id.txtBarcodeValue);
        surfaceView = findViewById(R.id.surfaceView);
    }

    private void initialiseDetectorsAndSources() {
        Toast.makeText(getApplicationContext(), "Barcode scanner started",
Toast.LENGTH_SHORT).show();
        barcodeDetector = new BarcodeDetector.Builder(this)
                .setBarcodeFormats(Barcode.ALL_FORMATS)
                .build();

        cameraSource = new CameraSource.Builder(this, barcodeDetector)
                .setRequestedPreviewSize(1920, 1080)
                .setAutoFocusEnabled(true) //you should add this feature
                .build();

        surfaceView.getHolder().addCallback(new SurfaceHolder.Callback() {
            @Override
            public void surfaceCreated(SurfaceHolder holder) {
                openCamera();
            }
            @Override
            public void surfaceChanged(SurfaceHolder holder, int format,
int width, int height) {
            }
            @Override
            public void surfaceDestroyed(SurfaceHolder holder) {
```

```
                cameraSource.stop();
            }
        });

        barcodeDetector.setProcessor(new Detector.Processor<Barcode>() {
            @Override
            public void release() {
                Toast.makeText(getApplicationContext(), "To prevent memory
leaks barcode scanner has been stopped", Toast.LENGTH_SHORT).show();
            }

            @Override
            public void receiveDetections(Detector.Detections<Barcode>
detections) {
                final SparseArray<Barcode> barCode =
detections.getDetectedItems();
                if (barCode.size() > 0) {
                    setBarCode(barCode);
                }
            }
        });
    }

    private void openCamera(){
        try {
            if
(ActivityCompat.checkSelfPermission(PictureBarcodeActivity.this,
Manifest.permission.CAMERA) == PackageManager.PERMISSION_GRANTED) {
                cameraSource.start(surfaceView.getHolder());
            } else {

ActivityCompat.requestPermissions(PictureBarcodeActivity.this, new
                        String[]{Manifest.permission.CAMERA},
REQUEST_CAMERA_PERMISSION);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void setBarCode(final SparseArray<Barcode> barCode){
        textViewBarCodeValue.post(new Runnable() {
            @Override
            public void run() {
                intentData = barCode.valueAt(0).displayValue;
                textViewBarCodeValue.setText(intentData);
```

```
                copyToClipBoard(intentData);
            }
        });
    }


    @Override
    protected void onPause() {
        super.onPause();
        cameraSource.release();
    }

    @Override
    protected void onResume() {
        super.onResume();
        initialiseDetectorsAndSources();
    }

    private void copyToClipBoard(String text){
        ClipboardManager clipboard = (ClipboardManager)
getSystemService(Context.CLIPBOARD_SERVICE);
        ClipData clip = ClipData.newPlainText("QR code Scanner", text);
        clipboard.setPrimaryClip(clip);
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull
String[] permissions, @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
        if(requestCode == REQUEST_CAMERA_PERMISSION &&
grantResults.length>0){
            if (grantResults[0] == PackageManager.PERMISSION_DENIED)
                finish();
            else
                openCamera();
        }else
            finish();
    }
}
```

CHAPTER 5

RESULT AND SCREENSHOTS

5.1 Home Page

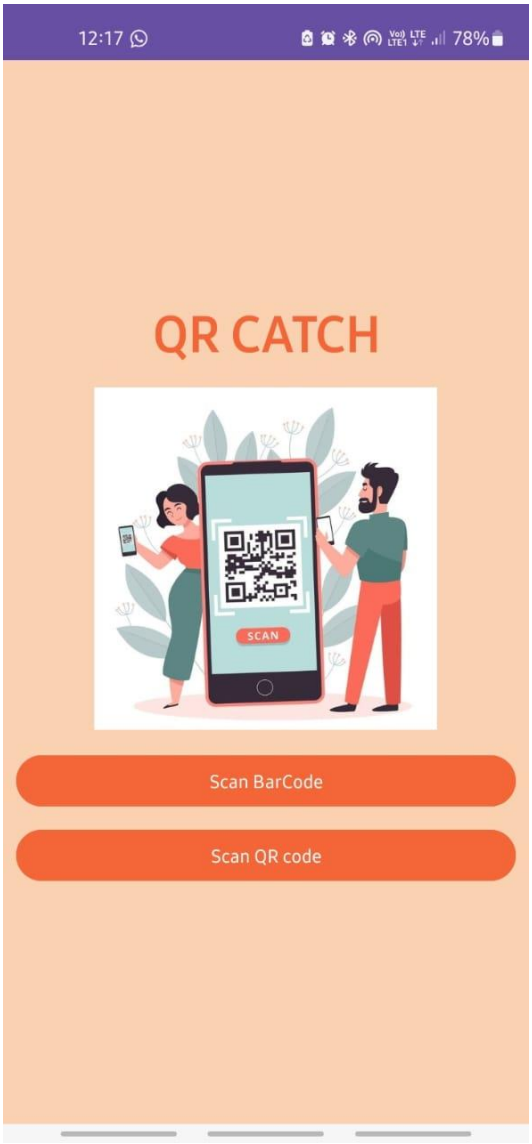

Fig 5.1 Home Page

The Home Activity serves as the main interface of the app, featuring two buttons: "Scan Barcode" and "Scan QR Code." This activity provides users with a convenient way to choose between barcode scanning or QR code scanning functionalities. By tapping on either button, users can navigate to the respective activities to perform the desired scanning operation.

5.2 Scan Barcode

Fig 5.2.1 Barcode Scanning

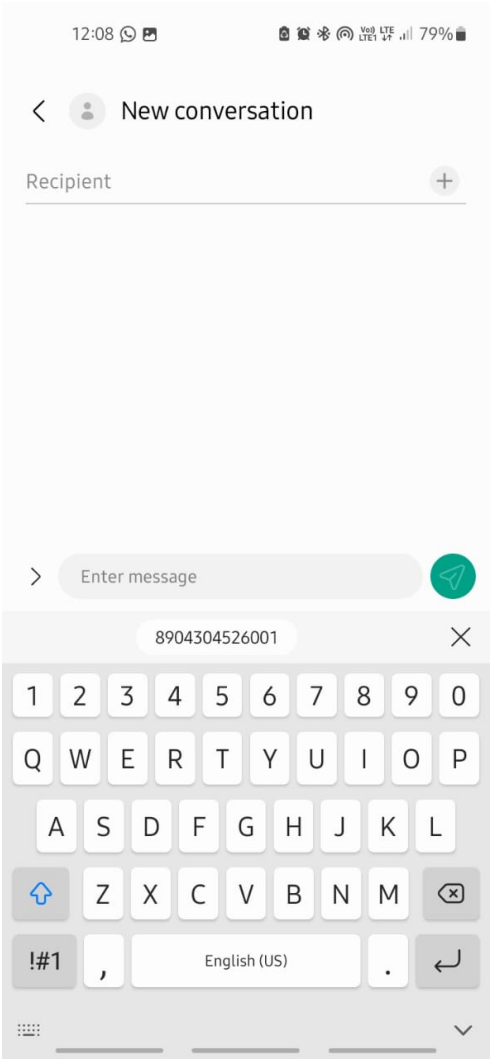

Fig 5.2.2 Barcode Number copied

In the Barcode Activity, the camera is activated, allowing users to scan barcodes. Once the barcode is successfully scanned, the app extracts the encoded number. This number is displayed on the screen, providing users with instant access to the information contained in the barcode. Additionally, the app automatically copies the extracted number to the clipboard, simplifying the process of using the barcode data in other applications or processes.

5.3 Scan QR Code

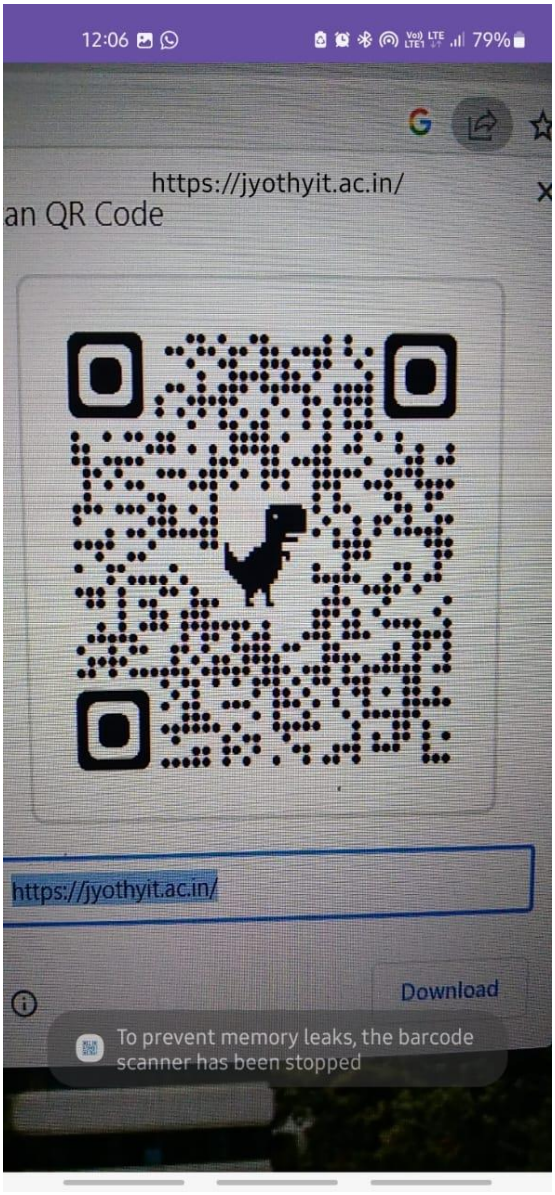

Fig 5.3.1 QR Scanning

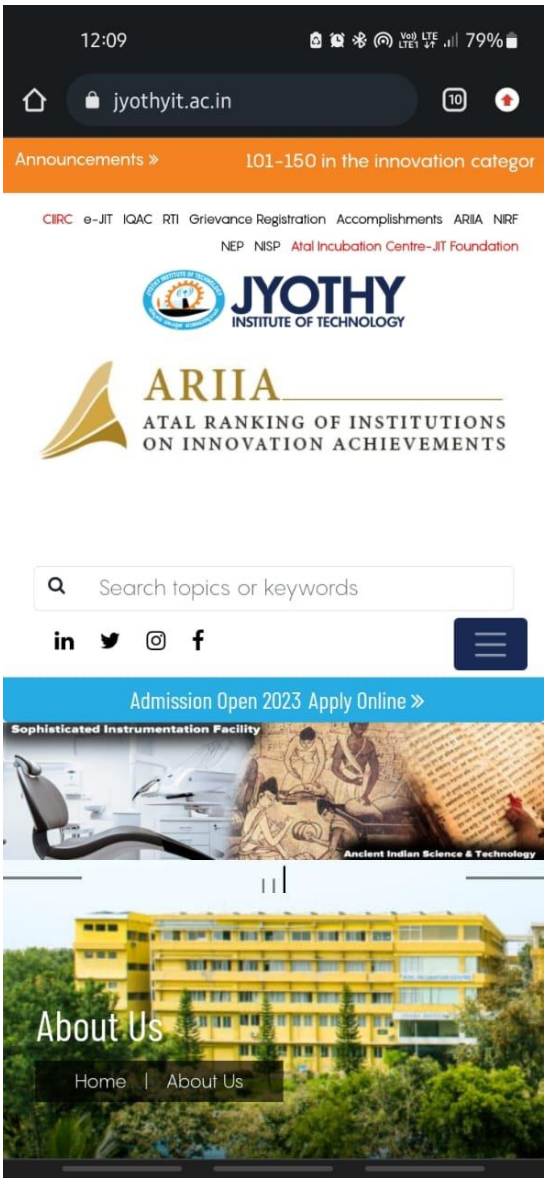

Fig 5.3.2 Redirecting to Webpage

The QR Code Activity opens the camera and enables users to scan QR codes. When a QR code is detected and recognized, the app triggers an action based on the content of the QR code. If the QR code contains a website URL, the app seamlessly redirects the user to the web browser, automatically opening the associated webpage. This functionality provides a convenient way for users to access web content directly from the scanned QR code. For QR codes containing other types of information, such as contact details or text, the app may display the decoded content on the screen, allowing users to view and interact with the information as required.

6. CONCLUSION

In conclusion, the QR code and barcode scanning app provides users with a comprehensive and efficient solution for capturing and utilizing different types of codes. With the ability to scan both barcodes and QR codes, the app offers a versatile platform that caters to a wide range of user needs. The intuitive user interface, featuring two main buttons for barcode and QR code scanning, ensures a seamless and straightforward experience for users.

The app's barcode scanning functionality accurately captures and decodes barcodes, providing users with immediate access to the encoded information. By automatically copying the decoded barcode value to the clipboard, the app streamlines the process of utilizing the barcode data in other applications or processes.

Additionally, the app's QR code scanning feature takes it a step further by seamlessly redirecting users to web content associated with QR codes containing website URLs. This unique aspect enhances user convenience, eliminating the need for manual URL entry and allowing for effortless access to online resources.

Overall, the QR code and barcode scanning app offers a user-friendly interface, efficient scanning capabilities, and seamless integration with other applications. It empowers users across various industries to efficiently capture, decode, and utilize barcode and QR code information, enhancing productivity and convenience. With its comprehensive functionality and user-centric design, the app proves to be a valuable tool for individuals and businesses alike in their barcode and QR code-related tasks.

REFERENCES

Text Books

1. "Android Programming: The Big Nerd Ranch Guide" by Bill Phillips and Brian Hardy.
2. “Android Programming with Android Studios” by J. F. DiMarzio
3. "Professional Android 4 Application Development" by Reto Meier.
4. "Android Programming for Beginners" by John Horton.
5. "Android Studio Development Essentials" by Neil Smyth.

Online Resources:

1. YouTube video: "Android App Development Tutorial for Beginners" by Derek Banas. Link: https://www.youtube.com/watch?v=8K-6gdTlGEA
2. YouTube video: "Android Development Tutorial - Full Course for Beginners" by freeCodeCamp. Link: https://www.youtube.com/watch?v=HF4aJ3ASyrk
3. GeeksforGeeks article: "Introduction to Android Development" by GeeksforGeeks. Link: https://www.geeksforgeeks.org/introduction-to-android-development/
4. Android Developer Documentation: Official documentation provided by Google for Android app development. Link: https://developer.android.com/docs
5. Codecademy: "Learn Android". Link: https://www.codecademy.com/learn/learn-android
6. YouTube Playlist: "Android App Development for Beginners" by thenewboston. Link: https://www.youtube.com/playlist?list=PL6gx4Cwl9DGBsvRxJJOzG4r4k_zLKrnxl