A PROJECT REPORT (IB403)

on

AUTHENTICATION SYSTEM USING QR CODE & OTP

This report is submitted for the partial fulfilment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

by

Utkarsh Gupta-180110001- CSE(CCV)

Dhananjay Pratap Singh-180110004- CSE(CCV)

Piyush Sachdeva -180110005- CSE(CCV)

Vikas Singh Mehta-180110007- CSE(CCV)

Under the Guidance of

Rohit Kamboj

IBM Faculty, Department of Computer Science & Engineering

SCHOOL OF COMPUTING

DIT UNIVERSITY, DEHRADUN

(State Private University through State Legislature Act No. 10 of 2013 of Uttarakhand and approved by UGC)

Mussoorie Diversion Road, Dehradun, Uttarakhand - 248009, India.

Nov,2021

CANDIDATES DECLARATION

We hereby certify that the work, which is being presented in the Report, entitled E-Authentication System using QR Code and OTP, in partial fulfilment of the requirement for the award of the Degree of Bachelor of Technology and submitted to the DIT University is an authentic record of my work carried out during the period August 2021 to November 2021 under theguidance of IBM CSE Faculty, Rohit Kamboj

[illegible] Dhananjay P. Singh

Piyush Sachdeva Dhananjay Pratap Singh

[illegible] Utkarsh

Vikas Singh Mehta Utkarsh Gupta

Date: 13th October 2021 **Signature of the Candidates**

This is to certify that the above statement made by the candidate is correct to the best of our knowledge.

Date: **Signature of the Supervisor**

ACKNOWLEDGEMENT

"Whatever your acknowledgement gains the legitimacy to exist within your world."

- Steven Redhead

We want to take this opportunity to thank all the people who continuously helped us strive towards the successful completion of my Summer Training Project. To name a few among many, I would first like to express my gratitude to my highly esteemed guide, IBM CSE Faculty, Rohit Kamboj, and the D.I.T University, Dehradun, for guiding us students on the right path time after time. In addition to this, we would like to thank our friends and family who were constantly there, supporting us and pushing us bit by bit to make this amazing project possible.

Utkarsh Gupta
Dhananjay Pratap Singh
Piyush Sachdeva
Vikas Singh Mehta

<u>**ABSTRACT**</u>

In this gizmo world, as technology is advancing day by day at breakneck speed, there are new apps being developed every day. Not only logins to these apps but also to bank websites and other important websites need to be incredibly safe and easy for the users. Over the years we have come up with password pattern restrictions to maximize the possible combinations, leading to decrease in the cases risk of information theft. It must also be noted that remembering these long passwords and emails often becomes a tedious task. Therefore, adding an extra layer of protection would go the distance way.

Both the ideas of a Quick Response Code (QR code) and One-Time Password (OTP) date back to the times when the modern computer was still in the development phase. QR code has been used for a long time now to mark manufactured products, dating back to its origin in Japan ' automotive industry. But with the growth of smart and sophisticated machines in the last decade, their use has become very common. They tend to avoid several shortcomings that are associated with traditional (static) password-based authentication. Their implementation of incorporating a two-factor authentication system is the most menial implementation we see in our day to day lives.

Table Of Content

List of Figures

List Of Abbreviations

1. UI : User Interface

2. OTP : One Time Password

3. QR : Quick Response

4. iOS : iPhone Operating System

5. IDE: Integrated Development Environment

6. JDK : Java Development Kit

7. GB : Giga Byte

8. RAM : Random Access Memory

9. GUI : Graphical User Interface

10. App : Application

11. E- Authentication : Electronic- Authentication

12. MVC : Model View Controller

13. OOP: Object Oriented Programming

14. ZXing : Zebra Crossing

Chapter -1

Introduction

To introduce our project, first I would like to talk about the idea behind this app. The thought behind our project is very easily understandable. We thought, that by the medium of this app, want to make this process of generation of OTP and QR code (for a limited time use mostly) very efficient and user friendly. This makes it easier for anyone to generate random codes as per their requirements. This is going to help users automate a lot of tasks and more than that, add another security layer to applications and things of day-to-day use.

As it is rightly said these days, "Information is Power." I am sure it is going to be the norm for the coming generations. And therefore, having a means to secure that information takes precedence over the information itself. This thought often gets overlooked by a lot of people. But as young individuals in the progressive field of Computer Science, it is our duty to contribute to the field of security.

With increase in field of technology, humans have made a great achievement, but this field also has a dark side. Some developers use un-authenticate practices and take out personal information. So, there is a need for a layer that only rightful person can access the account. Now, the need for authentication has arrived.

1.1 UI(User Interface):

Alan Kay, Douglas Engelbart were the main researchers of developing the first UI at Xerox PARC in 1981.The first computer with GUI was introduced by Apple named Lisa in 1983, as GUI is an important moment for the evolution of UI. As time changes more and more applications are introducing in the market. Since, the first impression of the application must be attractive and user friendly.

The main and most important feature of any UI is that it must be user friendly. Suppose UI of an application is not user friendly, then user will feel very difficult to operate and it is not a correct sign for any App Developer.

1.2 E-Authentication:

The need for authentication has been prevalent throughout history. In ancient times, people would identify each other through eye contact and physical appearance. .Now in this era of technology new ways to authenticate people are being developed and are termed as E-authentication. It is the process of electronic verification of the identity of an entity. The entity may be a person using a computer/mobile, a computer/mobile itself or a computer/mobile program.

Indian government has even developed a dedicated framework for E-Authentication, this framework enables various government departments and agencies to address the access management and authorization requirements associated with the deployment of E-governance applications and services.

Studies shows that electronic authentication helps to build confidence and trust in online transactions and encourages the use of the electronic environment as a channel for service delivery. In online transactions, data is communicated electronically through internet and mobile applications. During the delivery of online public services, it is not only important to authenticate the user for her identity, but it is also important to authenticate the Web site that the user is accessing for availing various public services. Lack of appropriate security measures in ensuring the authenticity of Web sites may lead to the user revealing her personal credentials over a fake Web site.

Chapter -2

Overall Project Description

2.1 Purpose:

The application's main purpose is to generate QR codes and OTPs for people, according to their needs. As these days data security is the utmost concern in this world and many technologies in the world have added these codes to provide an extra layer of security to the users.
Through the medium of this app, users can add this extra layer wherever they want, and it is not just restricted to the applications that provide this. Given the adoption of this method of security by banks, railways, and various other government sectors, this boosts the utility of the app.

2.2 Motivation:

As mentioned above, the concern of data privacy is rising rapidly, we all as a team wanted to make sure that the power of data should only be possessed by its creator. Therefore, to make sure that the user has complete control over his/her data and can secure it according to his needs, is what pushed us harder to pursue this idea.

2.3 Problem Statement:

In today's era, with great achievements of technology, people can use unauthenticated practices to breach in other's personal system. This model will give an extra layer of security to it.
If a person wants a security layer for a small competition or activity, then this application will have a lot to offer by generating OTP and QR code.
QR Code and OTP generated by this model can be applied on the following:

- Small Competition
- Creation of account of new applications
- Food Products
- Student Activities

2.4 Project Perspective:

The perspective of this project is simple. It is meant to make trivial things like securing user data easy for the user. Without any restrictions, it aims to provide the user with a simplistic user-friendly UI to make his/her life easy.

2.5 System Requirements:

- **Software Requirements:**

  For Windows and Linux

  - Windows 7 and above
  - Linux 18.04 and above
  - Android Studio 4 and above
  - JDK 8 and above

  For Mac

  - High Sierra and above (10.13)
  - Android Studio 4 and above
  - JDK 8 and above.
  - XCode 11 and above

- **Hardware Requirements:**

- Intel core i5
- 8 GB RAM

Chapter -3

Methods and Materials

3.1 Software for UI designing:

Android Studio: Android Studio is the official IDE for Android app development, design of UI. Above UI is also designed in this software which is based on IntelliJ IDEA. It offers even more features that enhance your productivity when building Android apps.

3.2 Various language for backend process:

JDK and Java:

JDK is a software or platform used to run java language-based programs; this software is mainly organized by Oracle.
Java here is a class-based, object-oriented, general-purpose programming language that is designed to implement various complex procedures in the world of Computer Science.

MVC:

MVC is an app design paradigm. It is used for developing backend of online transaction and Android applications user interfaces that divides the related program logic into three interconnected elements namely Model, View and Controller.

OOPs:

It is a very common programming paradigm where there is a notion of classes and objects to provide us with the required abstraction. An object represents a real-life entity which has attributes and behaviors as variables and methods.

Cryptography:

Generally, cryptography is about constructing and analyzing protocols that prevent third parties or the public from reading private messages.

XCode:

XCode is a type of IDE for mac operating system containing a suite of software development tools. It is used to develop software for operating system of mac, iPad, watch, and tv by Apple organization.

Swift:

Swift is one type of programming language just like Java but for iOS, iPadOS, macOS, watchOS, tvOS, and Linux developed by Apple organization. On Apple platforms, like java programming language it uses the Object C runtime library which allows different types of languages like C, Object-C, C++, and Swift based programs to run within one program.

ZXing:

ZXing is a barcode image processing library implemented in Java, with ports to other languages. It has support for 1D product, 1D industrial, and 2D barcodes. ZXing is used by web search to make millions of barcodes on the web indexable. It also forms the basis of Android's Barcode Scanner app and is integrated into Google Product and Book Search.

Virtual Device:

A Virtual Machine (VM) uses software to run programs and deploy apps instead of physical machine or resource. Even in the same host device, every virtual machine uses its own operating system independent on others virtual machines, since one or more virtual "guest" machines can run on a single physical "host" machine for example, a virtual MacOS virtual machine can run on a physical PC.

Chapter -4

Implementation Modules And Screen Shots

4.1 iOS:

4.1.1 UI Design:

The idea always has been to have a simplistic UI, which is very user-friendly, making it easy for a wide class people to use the app efficiently. Keeping the same thought in mind we all, in the team decided to keep things smooth, crisp to minimize clutter which is beautiful to look at and easy for the eyes.

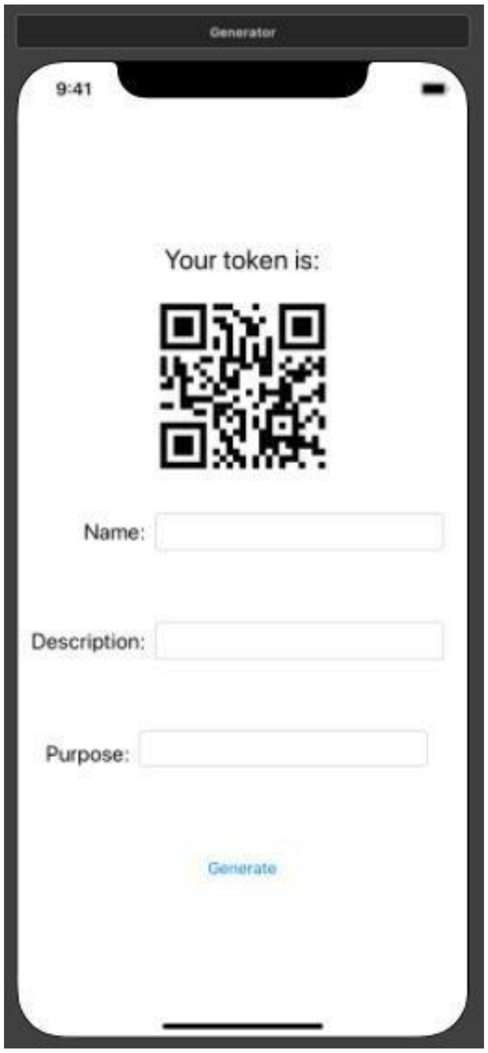


Figure 1

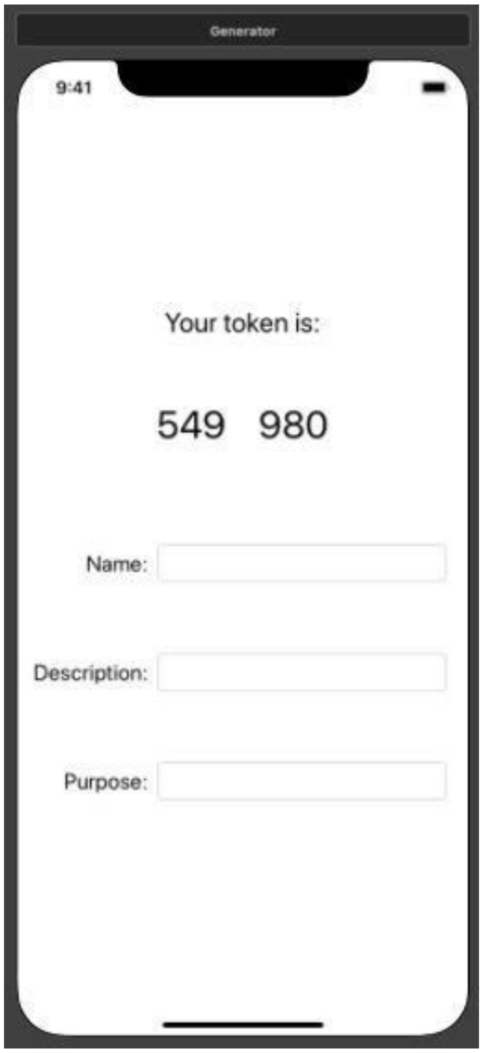


Figure 2

4.1.2 OTP and QR code generator:

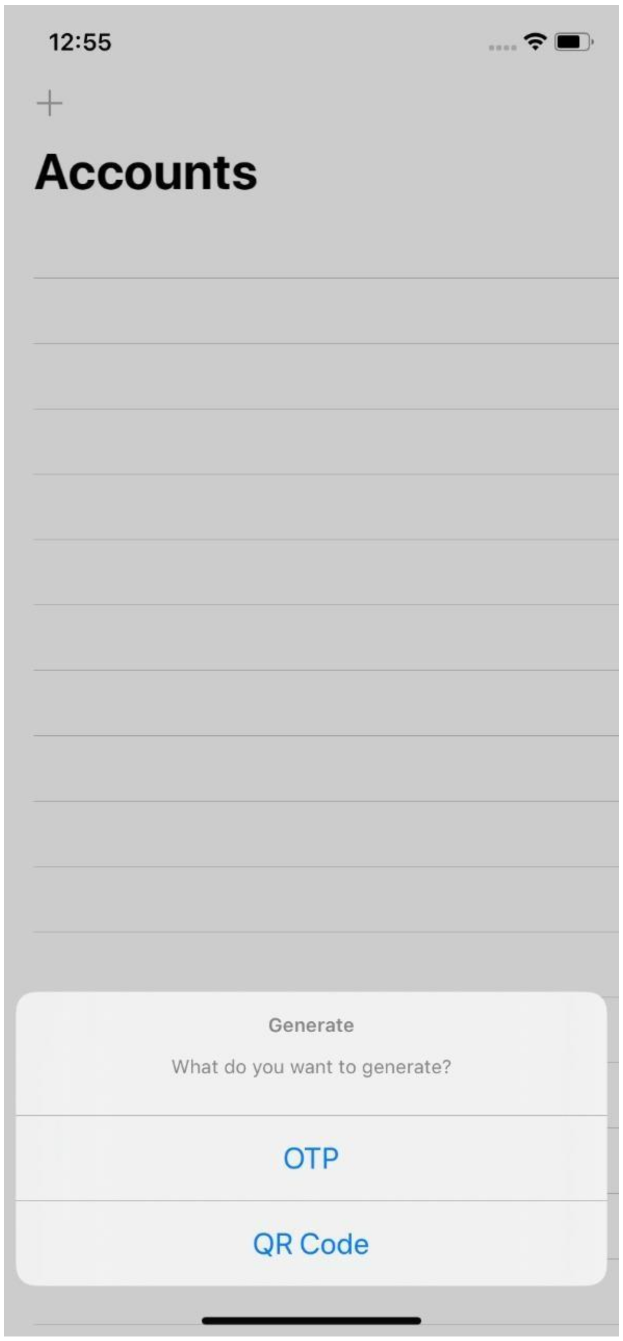
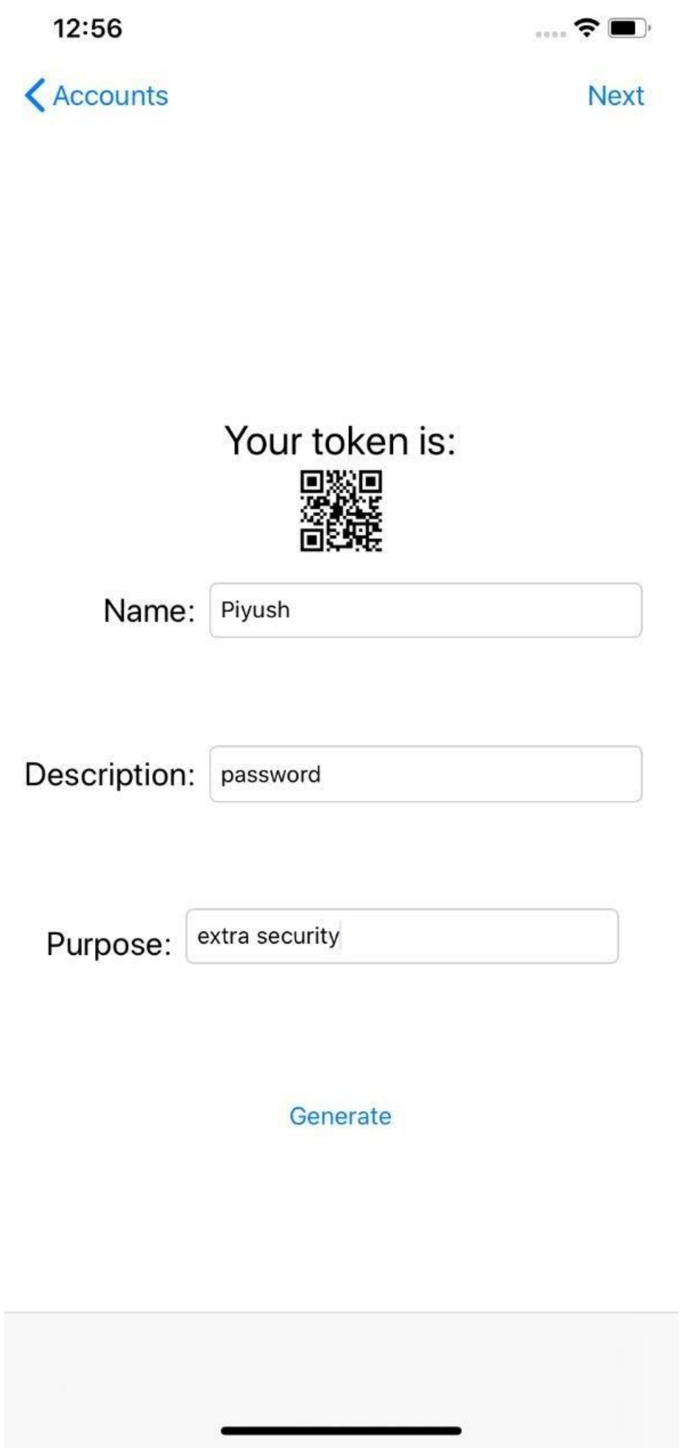


Figure 3



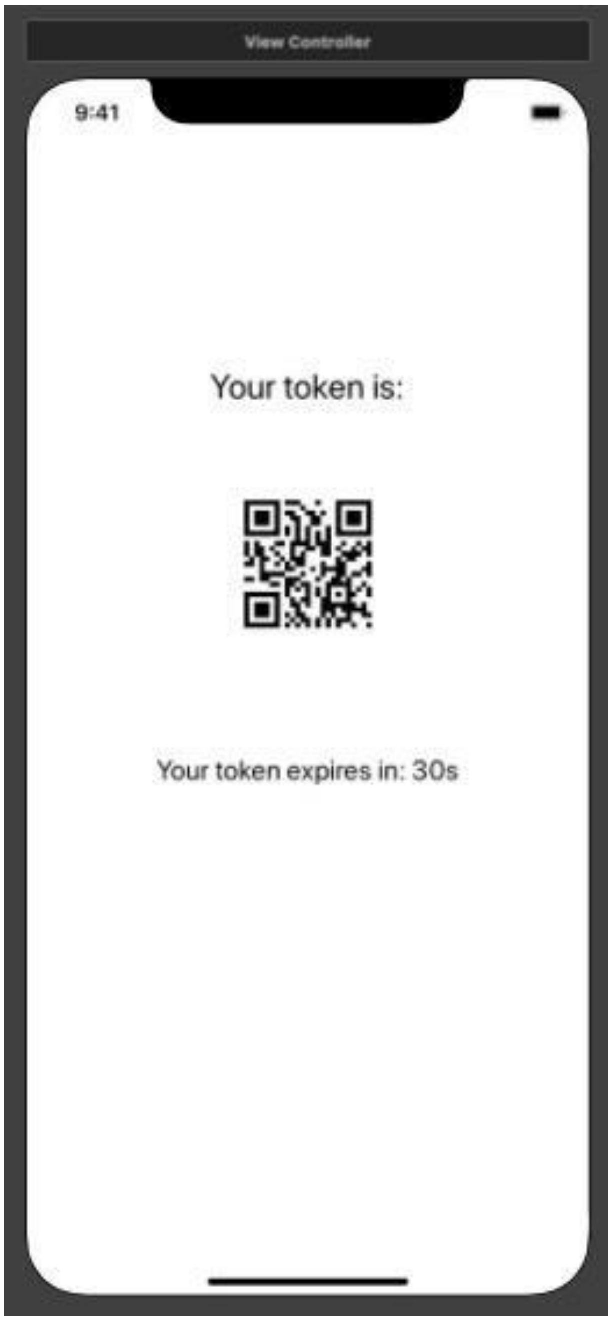
Your token is:

Name: Piyush

Description: password

Purpose: extra security

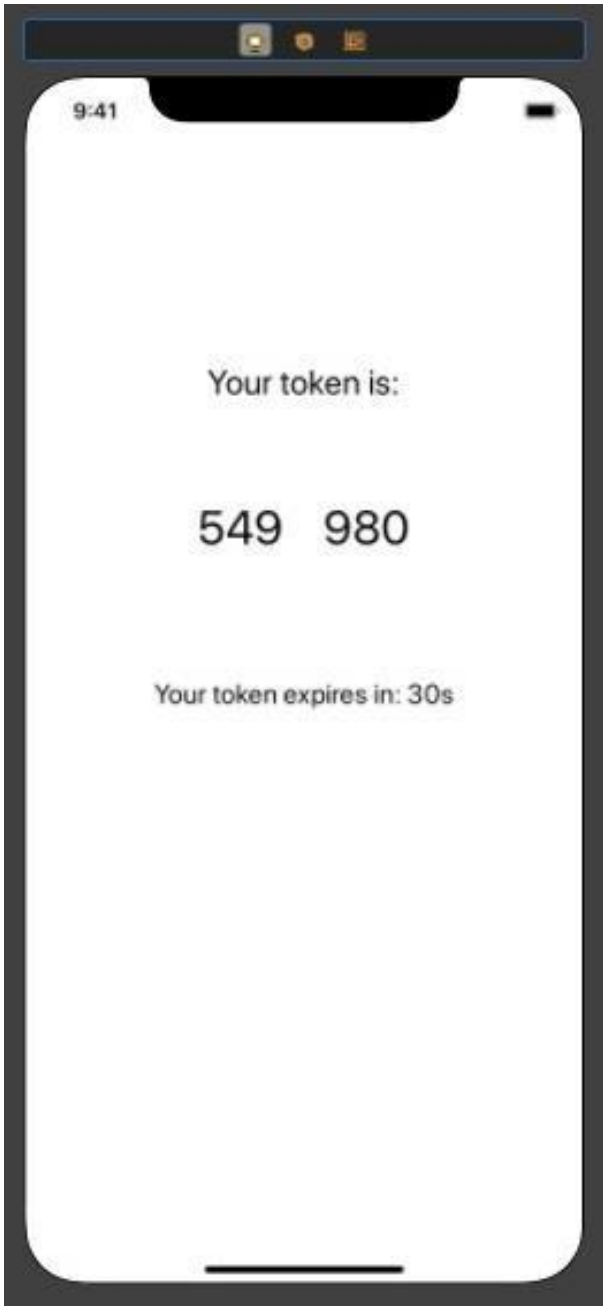
Figure 4



Your token is:

Your token expires in: 30s

Figure 5



Your token is:

549 980

Your token expires in: 30s

Figure 6

4.1.3 Blueprint of UI:

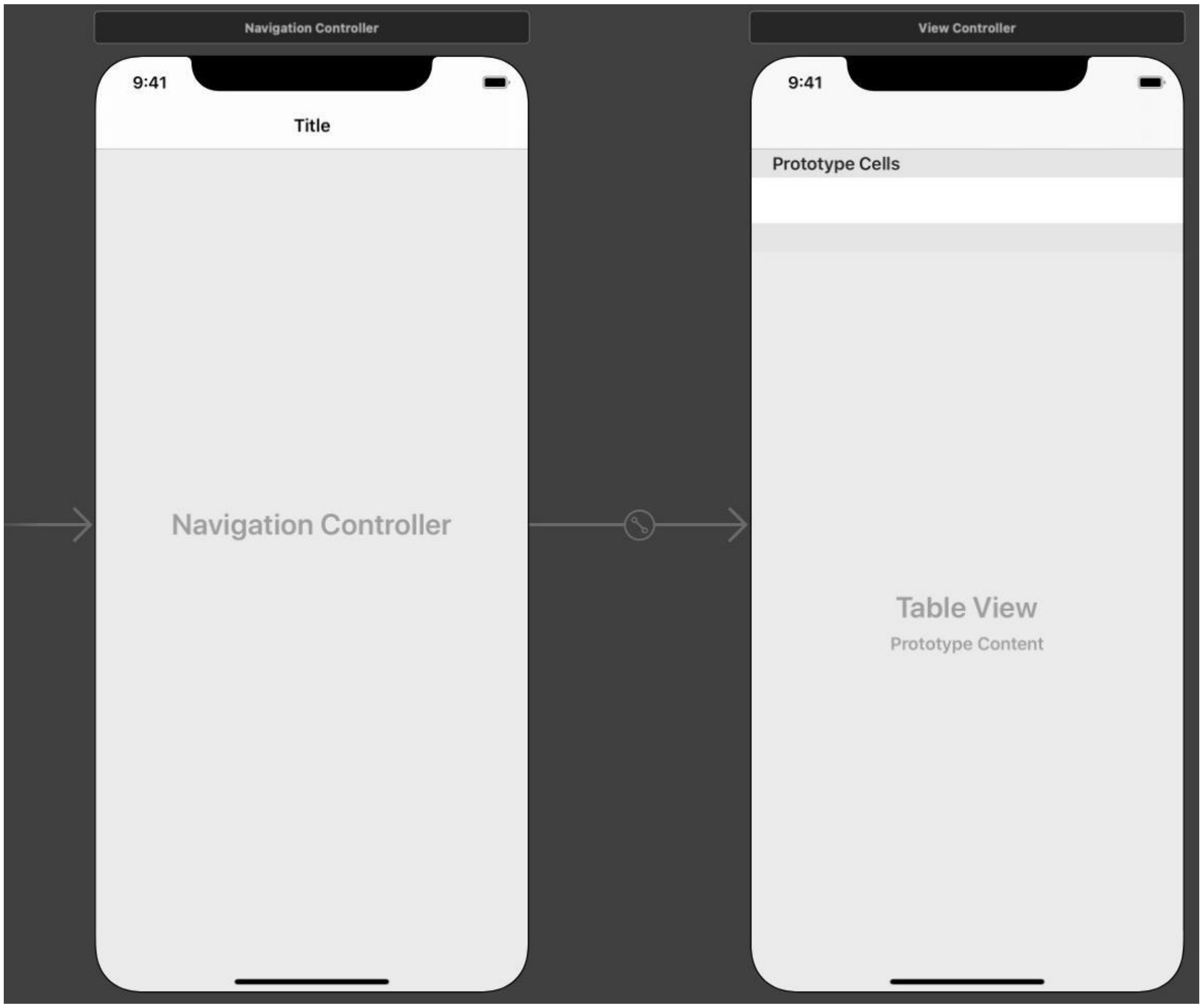


Figure 7

4.2 Android:

4.2.1 QR Code Generator:

It is a combination of black and white pixels that generates a specific pattern in a square box and every time it is generated it will be different from the previous one.

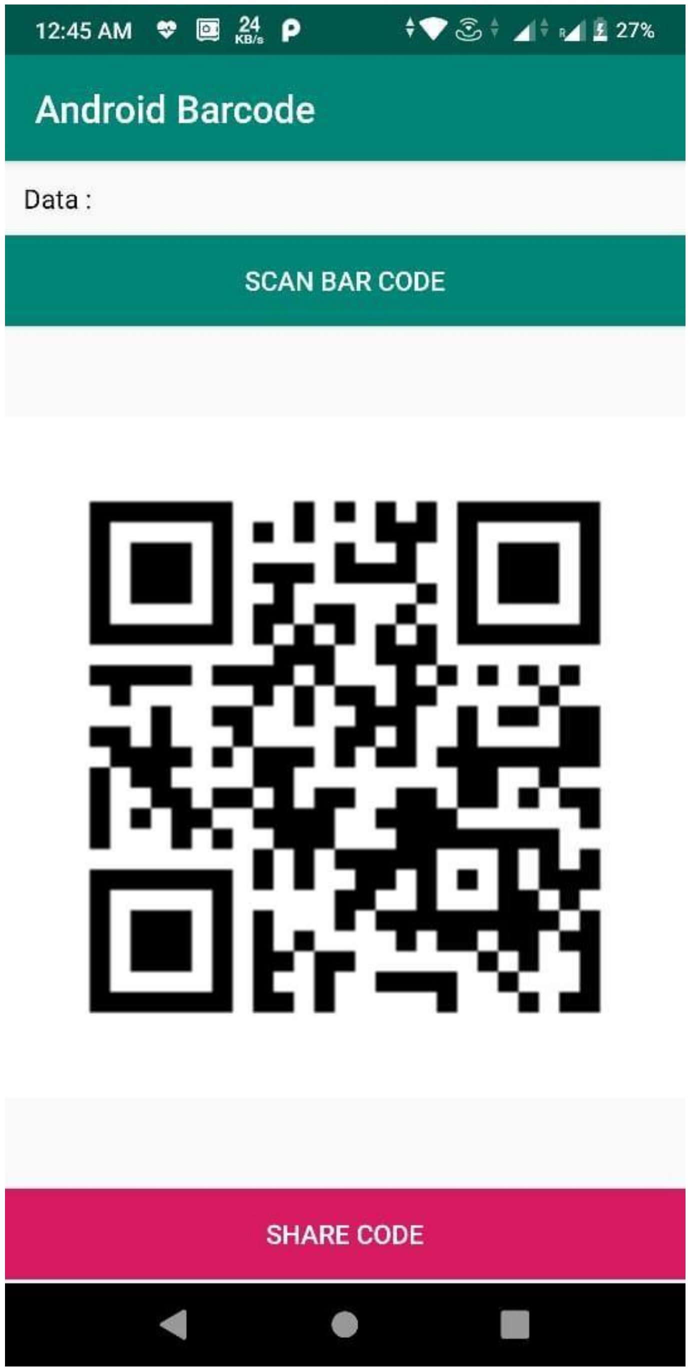


Figure 8

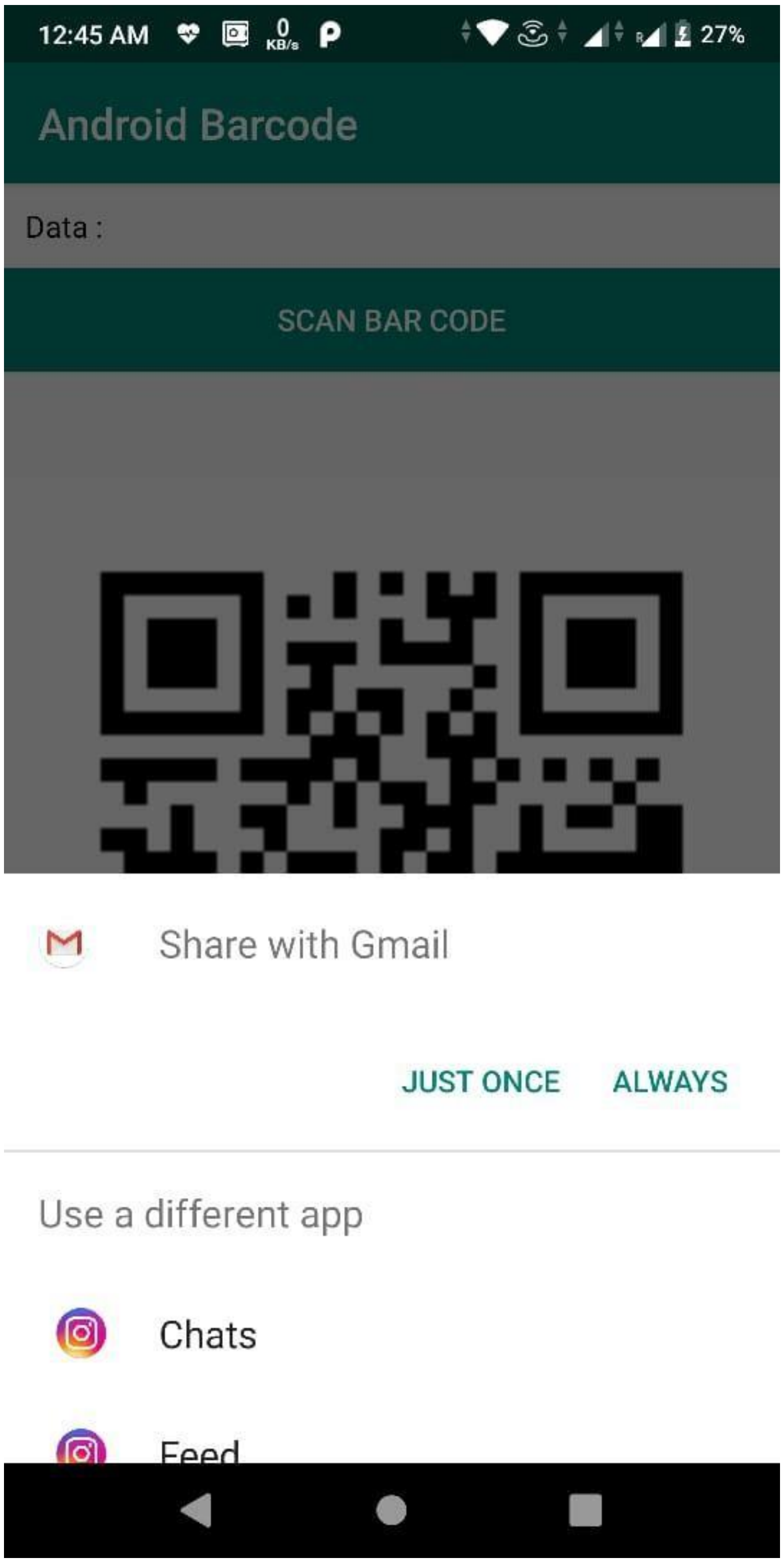

12:45 AM
27%
Android Barcode
Data :
SCAN BAR CODE
Share with Gmail
JUST ONCE
ALWAYS
Use a different app
Chats
Feed

Figure 9

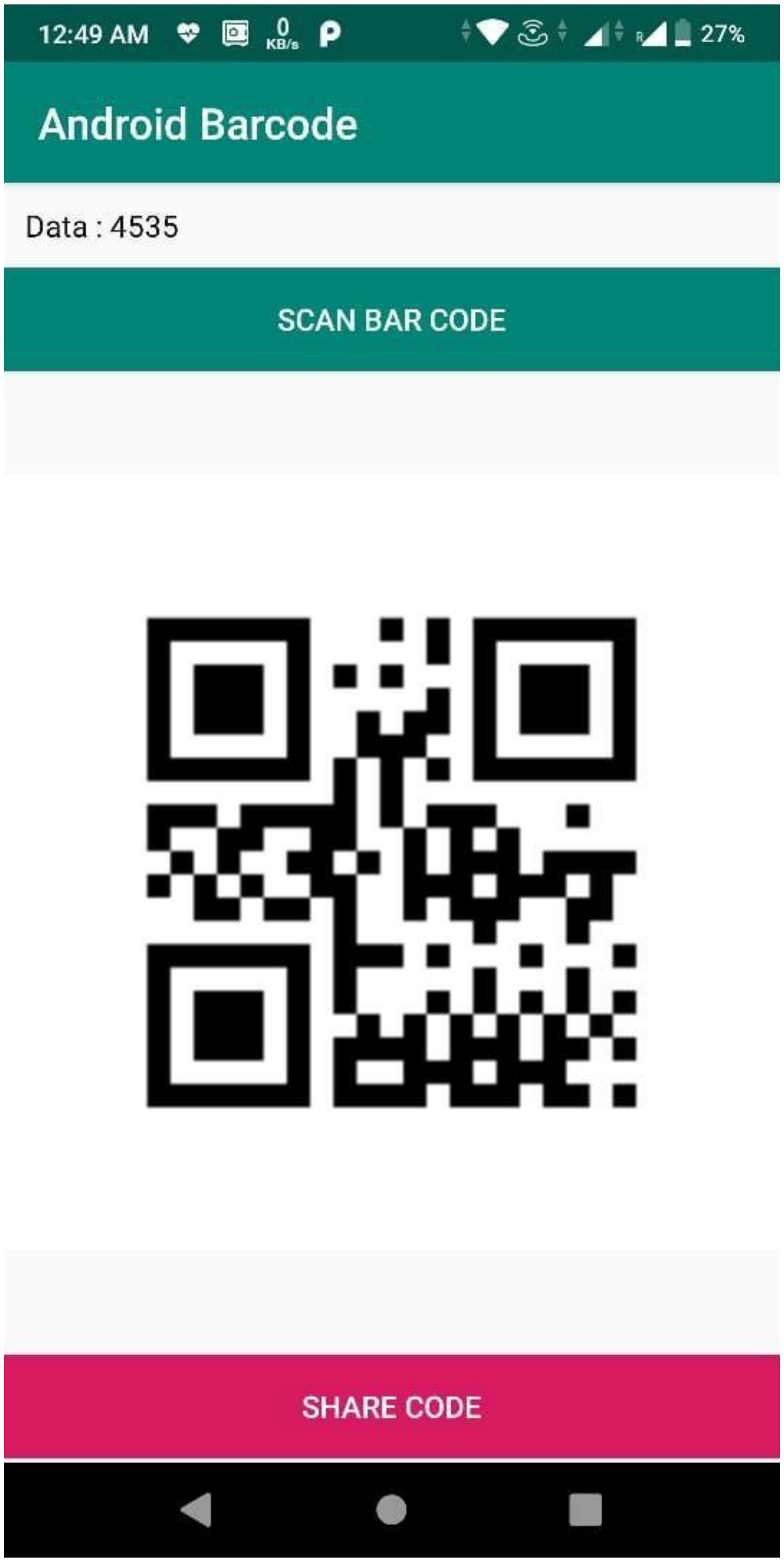

12:49 AM
27%
Android Barcode
Data : 4535
SCAN BAR CODE
SHARE CODE

Figure 10

4.2.2 OTP Generator:

It is a specific number of codes that is generated for a particular time interval and every time a new code generates by the user it will be different from the previous one.

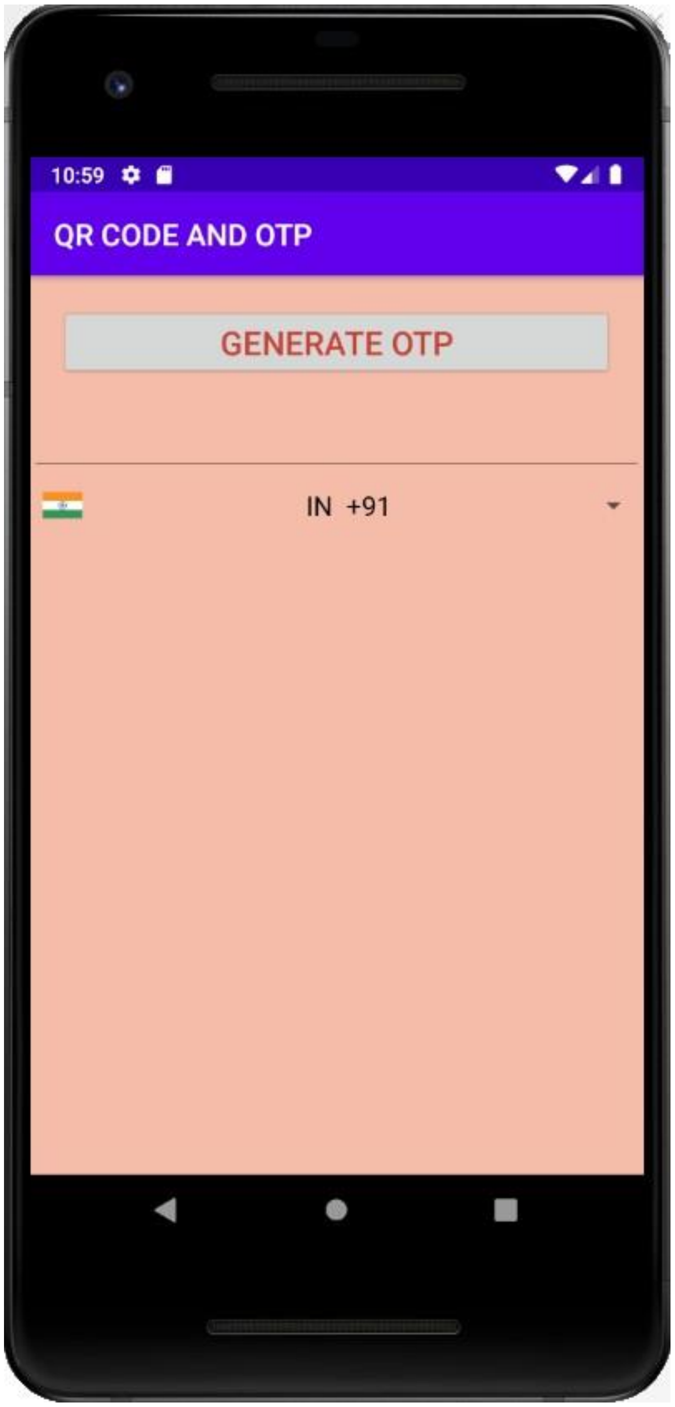
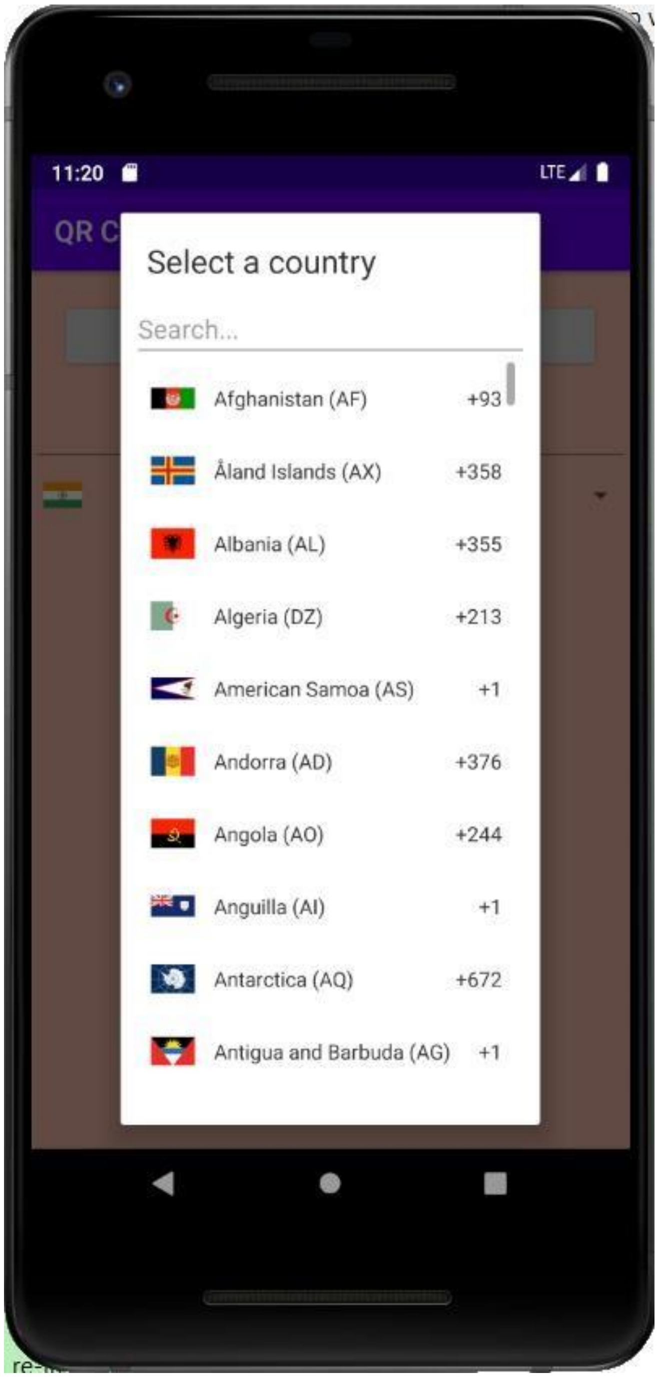


Figure 11

Country Code Peeker:



11:20

QR C

Select a country

Search...

Afghanistan (AF)	+93
Åland Islands (AX)	+358
Albania (AL)	+355
Algeria (DZ)	+213
American Samoa (AS)	+1
Andorra (AD)	+376
Angola (AO)	+244
Anguilla (AI)	+1
Antarctica (AQ)	+672
Antigua and Barbuda (AG)	+1

Figure 12

OTP Generator for Android:

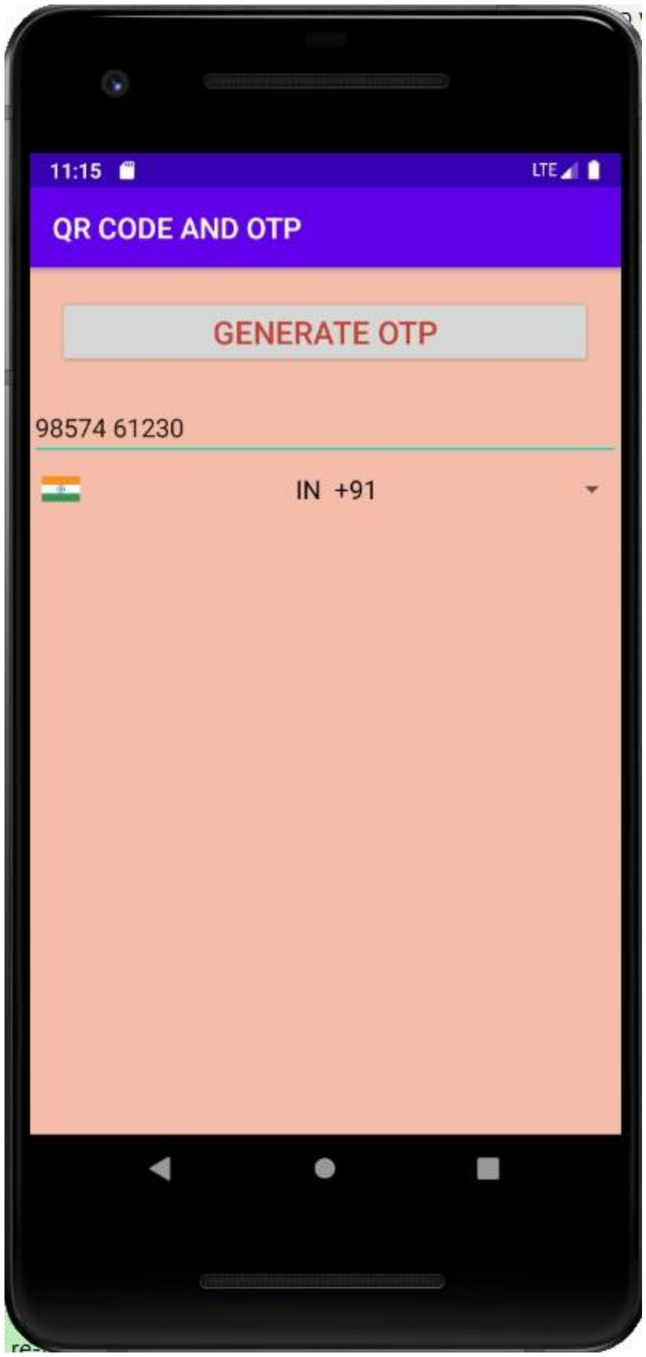


Figure 13

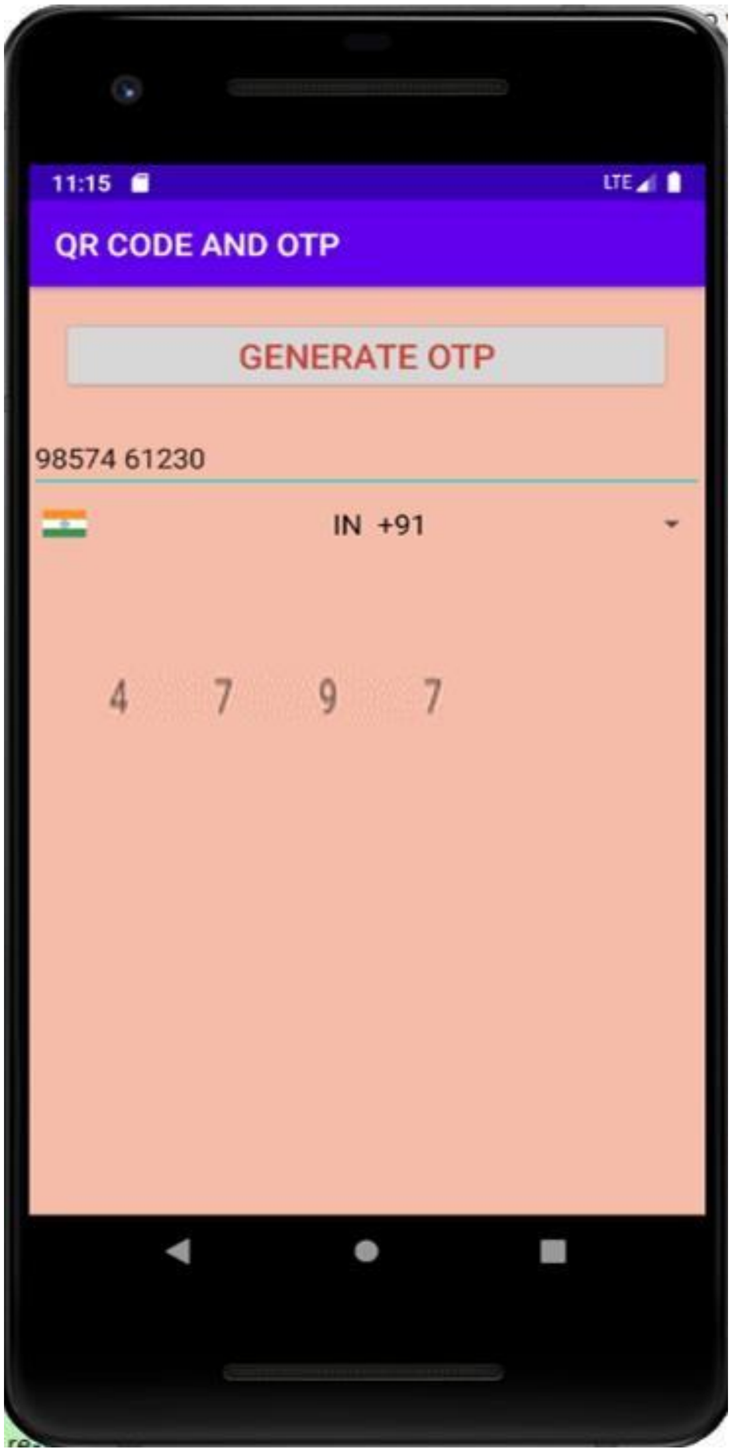
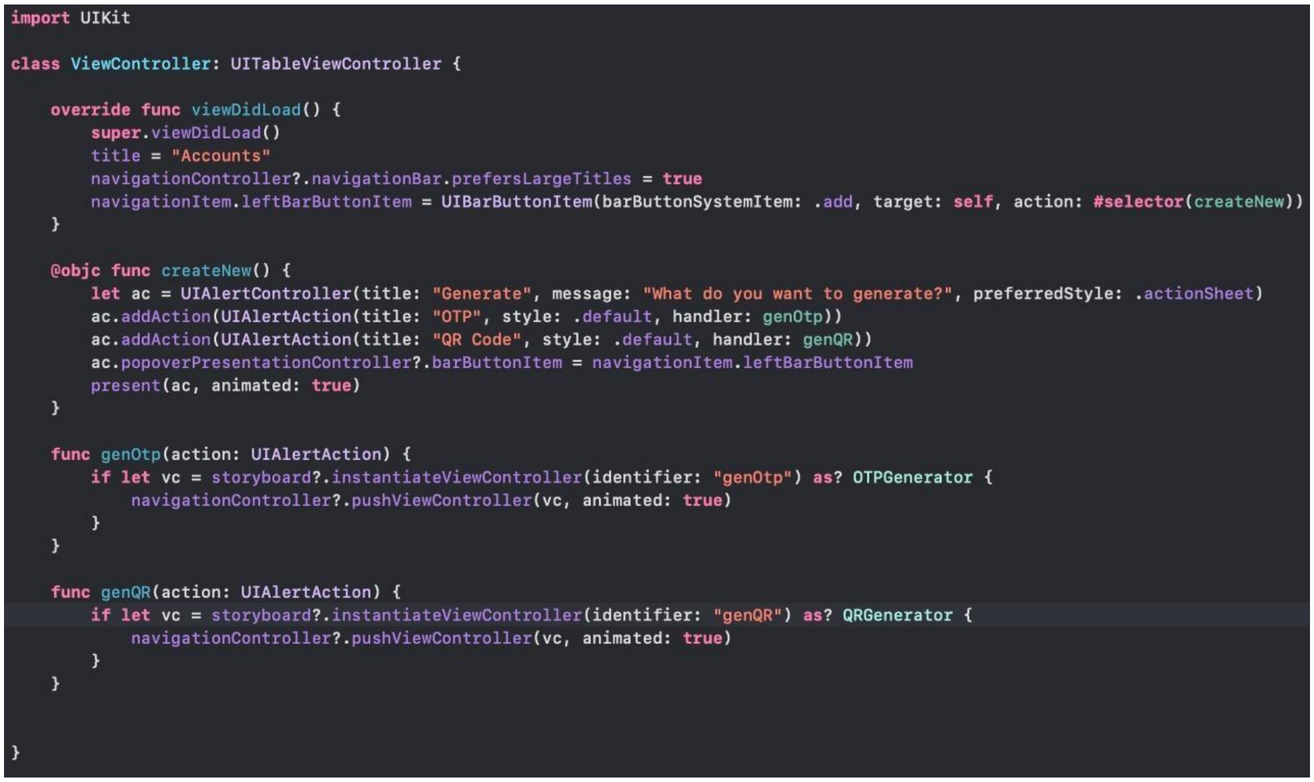


Figure 14

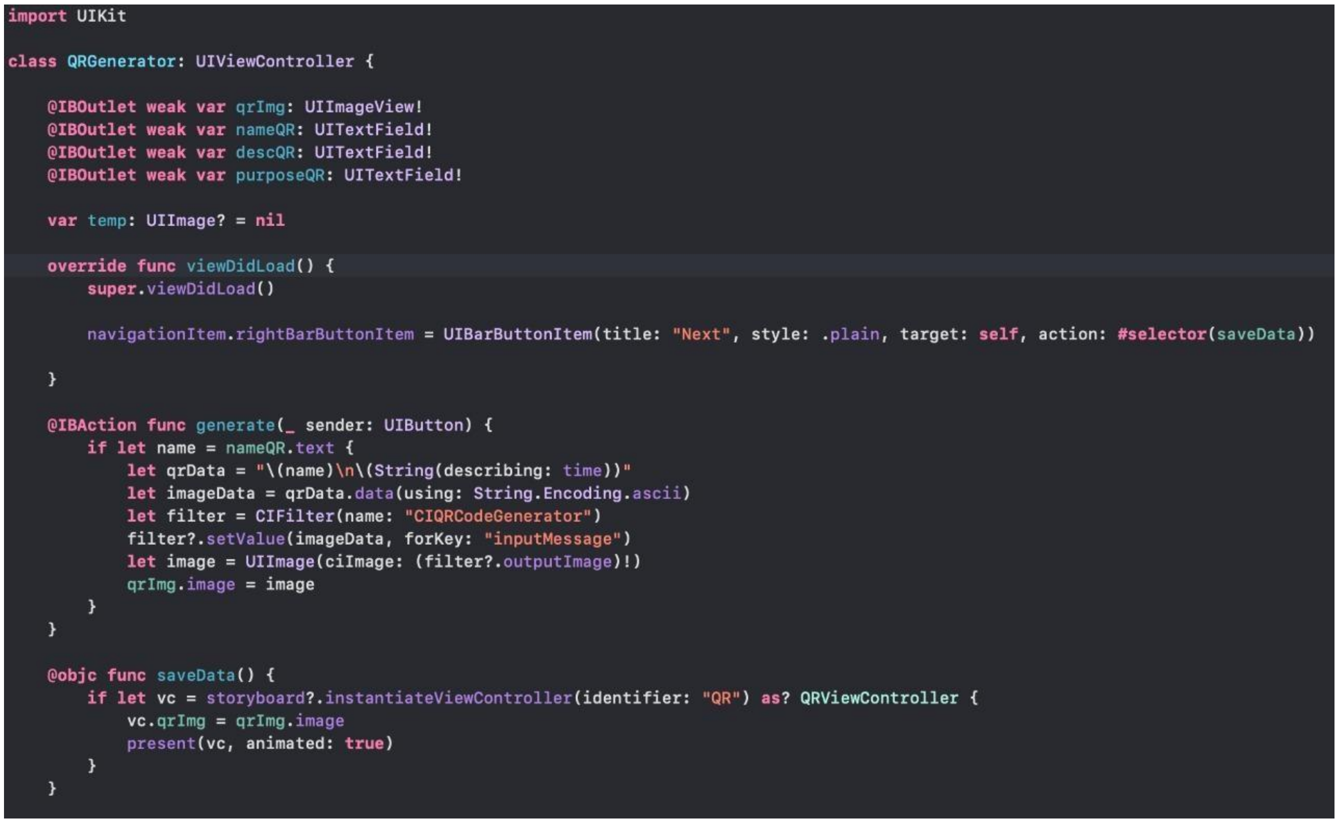
4.3 Code:

4.3.1 iOS Code:



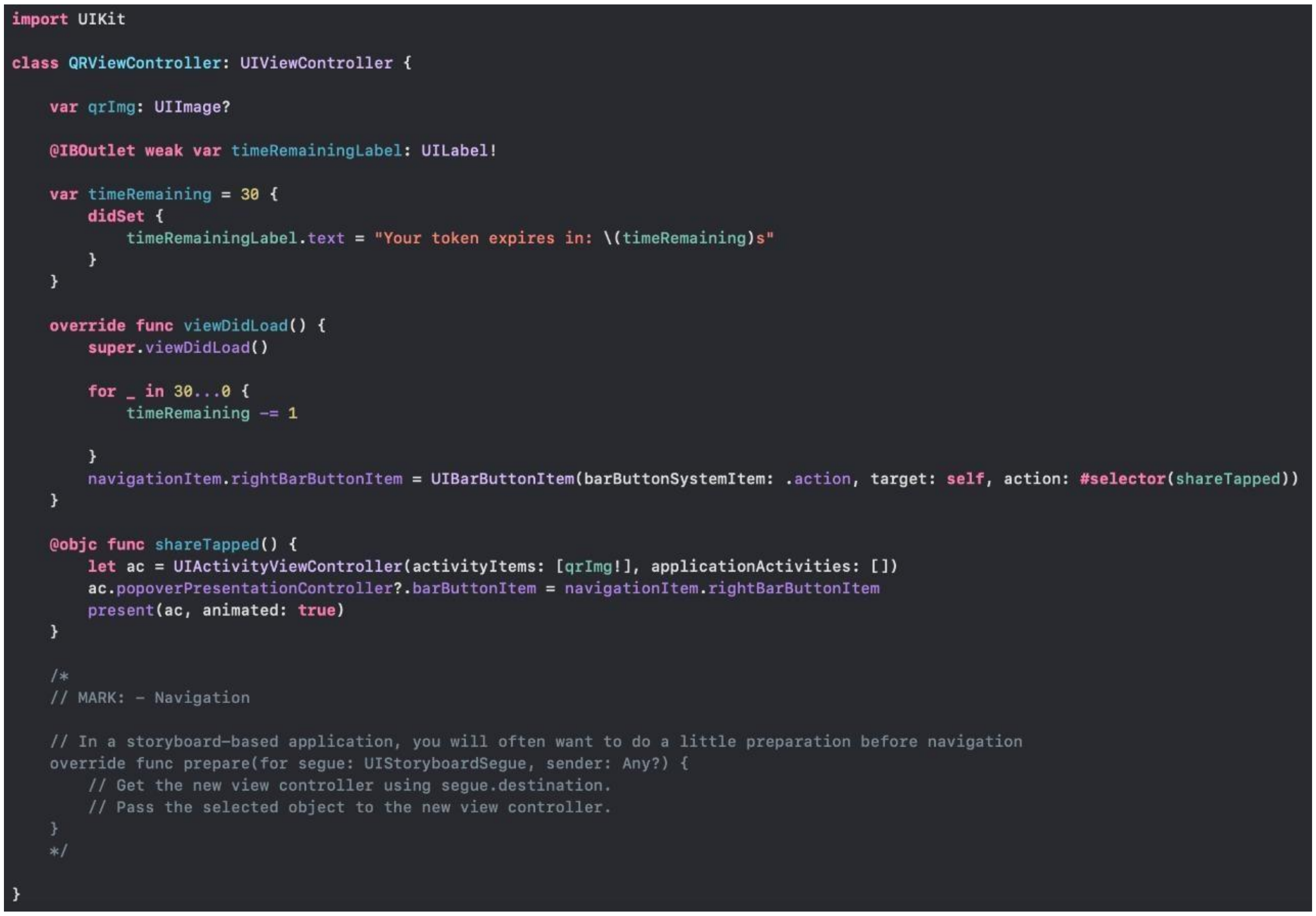
```swift
import UIKit

class ViewController: UITableViewController {

    override func viewDidLoad() {
        super.viewDidLoad()
        title = "Accounts"
        navigationController?.navigationBar.prefersLargeTitles = true
        navigationItem.leftBarButtonItem = UIBarButtonItem(barButtonSystemItem: .add, target: self, action: #selector(createNew))
    }

    @objc func createNew() {
        let ac = UIAlertController(title: "Generate", message: "What do you want to generate?", preferredStyle: .actionSheet)
        ac.addAction(UIAlertAction(title: "OTP", style: .default, handler: genOtp))
        ac.addAction(UIAlertAction(title: "QR Code", style: .default, handler: genQR))
        ac.popoverPresentationController?.barButtonItem = navigationItem.leftBarButtonItem
        present(ac, animated: true)
    }

    func genOtp(action: UIAlertAction) {
        if let vc = storyboard?.instantiateViewController(identifier: "genOtp") as? OTPGenerator {
            navigationController?.pushViewController(vc, animated: true)
        }
    }

    func genQR(action: UIAlertAction) {
        if let vc = storyboard?.instantiateViewController(identifier: "genQR") as? QRGenerator {
            navigationController?.pushViewController(vc, animated: true)
        }
    }


}
```

Figure 15



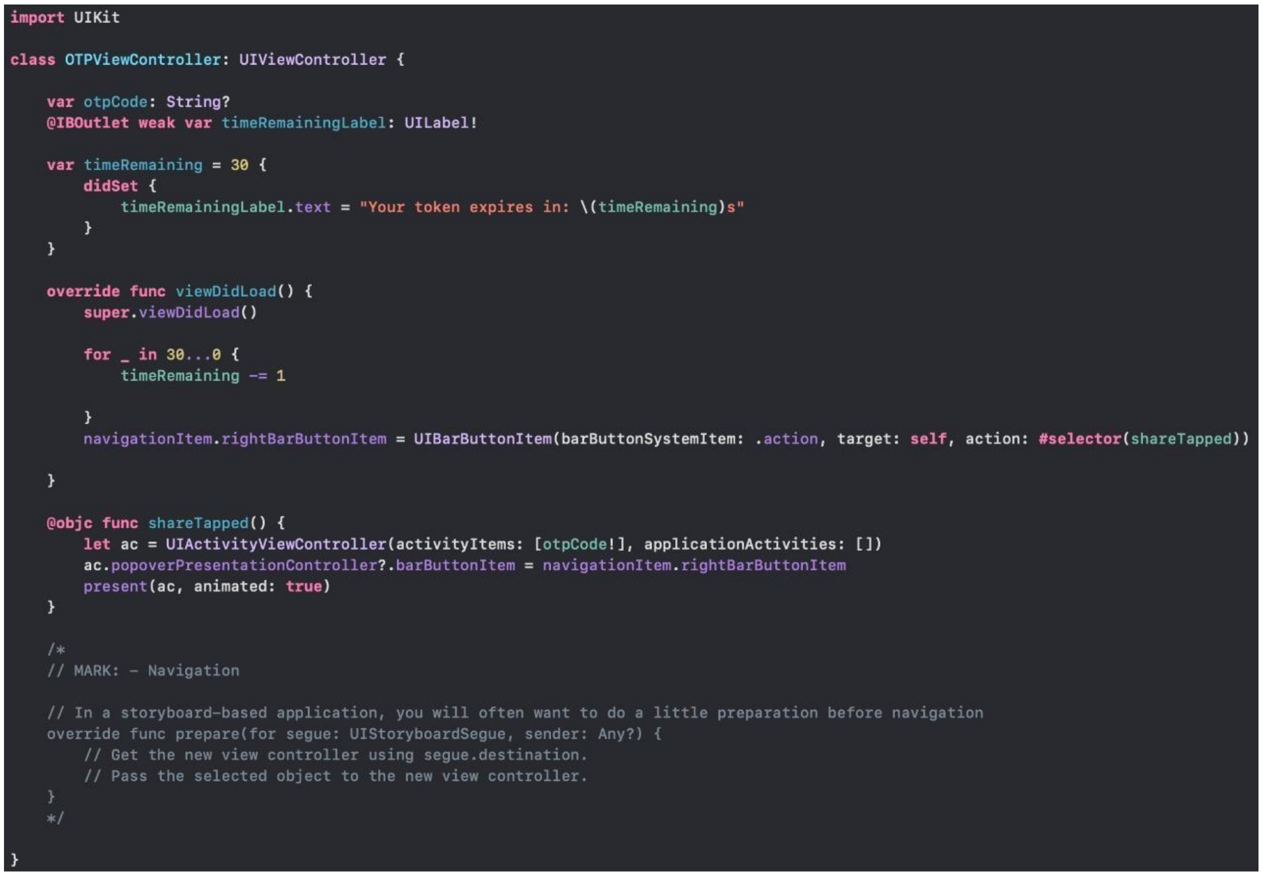
```swift
import UIKit

class QRGenerator: UIViewController {

    @IBOutlet weak var qrImg: UIImageView!
    @IBOutlet weak var nameQR: UITextField!
    @IBOutlet weak var descQR: UITextField!
    @IBOutlet weak var purposeQR: UITextField!

    var temp: UIImage? = nil

    override func viewDidLoad() {
        super.viewDidLoad()

        navigationItem.rightBarButtonItem = UIBarButtonItem(title: "Next", style: .plain, target: self, action: #selector(saveData))

    }

    @IBAction func generate(_ sender: UIButton) {
        if let name = nameQR.text {
            let qrData = "\(name)\n\(String(describing: time))"
            let imageData = qrData.data(using: String.Encoding.ascii)
            let filter = CIFilter(name: "CIQRCodeGenerator")
            filter?.setValue(imageData, forKey: "inputMessage")
            let image = UIImage(ciImage: (filter?.outputImage)!)
            qrImg.image = image
        }
    }

    @objc func saveData() {
        if let vc = storyboard?.instantiateViewController(identifier: "QR") as? QRViewController {
            vc.qrImg = qrImg.image
            present(vc, animated: true)
        }
    }
```

Figure 16



```swift
import UIKit

class QRViewController: UIViewController {

    var qrImg: UIImage?

    @IBOutlet weak var timeRemainingLabel: UILabel!

    var timeRemaining = 30 {
        didSet {
            timeRemainingLabel.text = "Your token expires in: \(timeRemaining)s"
        }
    }

    override func viewDidLoad() {
        super.viewDidLoad()

        for _ in 30...0 {
            timeRemaining -= 1

        }
        navigationItem.rightBarButtonItem = UIBarButtonItem(barButtonSystemItem: .action, target: self, action: #selector(shareTapped))
    }

    @objc func shareTapped() {
        let ac = UIActivityViewController(activityItems: [qrImg!], applicationActivities: [])
        ac.popoverPresentationController?.barButtonItem = navigationItem.rightBarButtonItem
        present(ac, animated: true)
    }

    /*
    // MARK: - Navigation

    // In a storyboard-based application, you will often want to do a little preparation before navigation
    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        // Get the new view controller using segue.destination.
        // Pass the selected object to the new view controller.
    }
    */

}
```

Figure 17

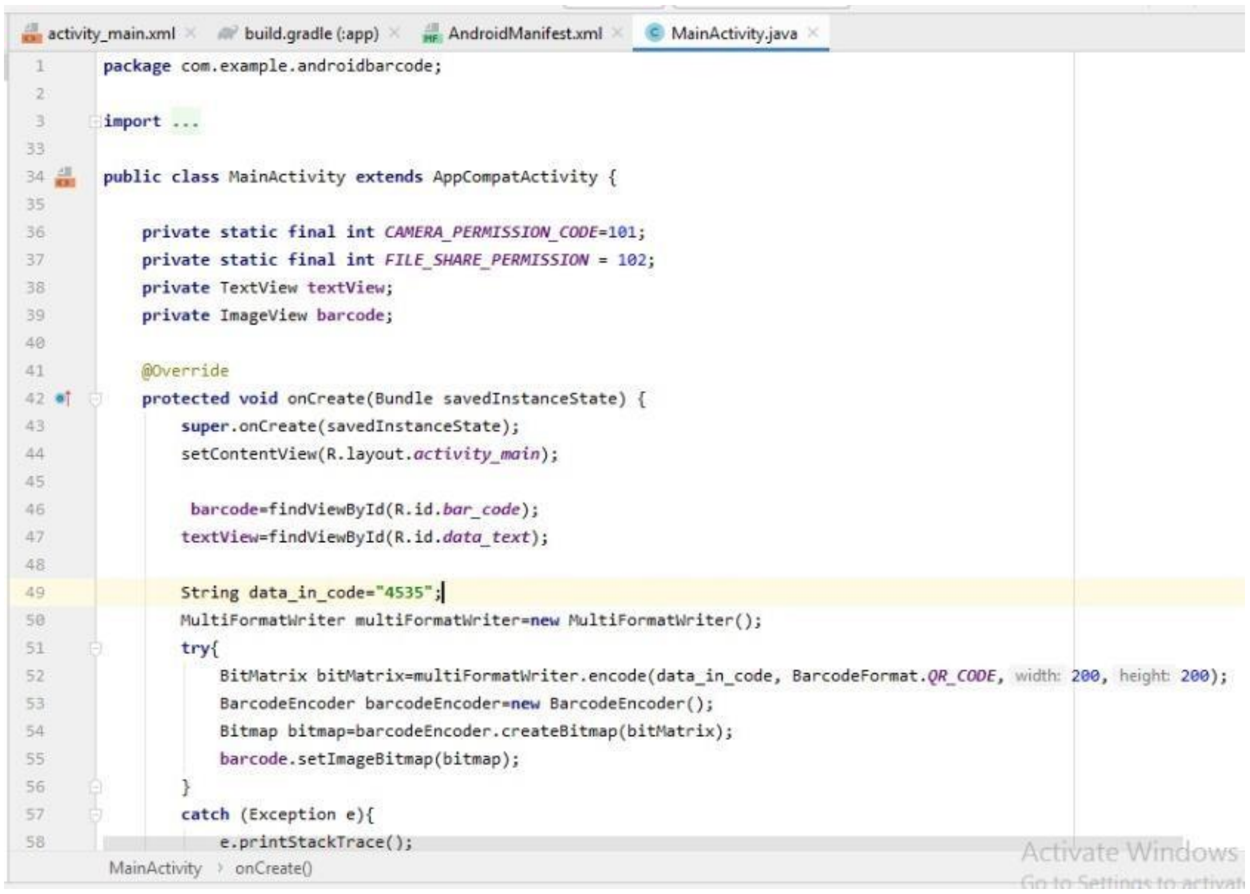


```
import UIKit

class OTPViewController: UIViewController {

    var otpCode: String?
    @IBOutlet weak var timeRemainingLabel: UILabel!

    var timeRemaining = 30 {
        didSet {
            timeRemainingLabel.text = "Your token expires in: \(timeRemaining)s"
        }
    }

    override func viewDidLoad() {
        super.viewDidLoad()

        for _ in 30...0 {
            timeRemaining -= 1

        }
        navigationItem.rightBarButtonItem = UIBarButtonItem(barButtonSystemItem: .action, target: self, action: #selector(shareTapped))

    }

    @objc func shareTapped() {
        let ac = UIActivityViewController(activityItems: [otpCode!], applicationActivities: [])
        ac.popoverPresentationController?.barButtonItem = navigationItem.rightBarButtonItem
        present(ac, animated: true)
    }

    /*
    // MARK: - Navigation

    // In a storyboard-based application, you will often want to do a little preparation before navigation
    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        // Get the new view controller using segue.destination.
        // Pass the selected object to the new view controller.
    }
    */

}
```

Figure 18

4.3.2 Android QR code Generator:

MainActivity.java:



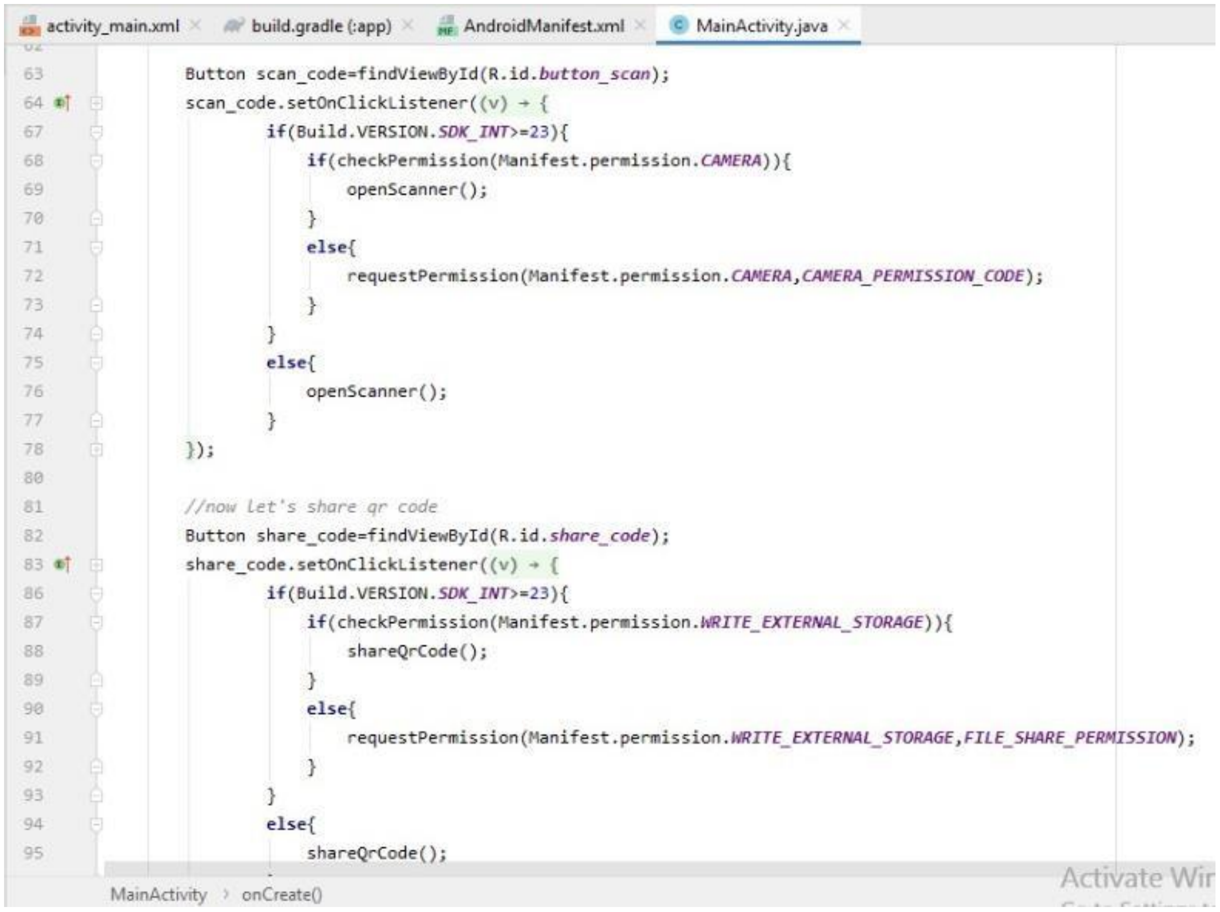
activity_main.xml × build.gradle (:app) × AndroidManifest.xml × MainActivity.java ×

```java
package com.example.androidbarcode;

import ...

public class MainActivity extends AppCompatActivity {

    private static final int CAMERA_PERMISSION_CODE=101;
    private static final int FILE_SHARE_PERMISSION = 102;
    private TextView textView;
    private ImageView barcode;

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

         barcode=findViewById(R.id.bar_code);
        textView=findViewById(R.id.data_text);

        String data_in_code="4535";
        MultiFormatWriter multiFormatWriter=new MultiFormatWriter();
        try{
            BitMatrix bitMatrix=multiFormatWriter.encode(data_in_code, BarcodeFormat.QR_CODE, width: 200, height: 200);
            BarcodeEncoder barcodeEncoder=new BarcodeEncoder();
            Bitmap bitmap=barcodeEncoder.createBitmap(bitMatrix);
            barcode.setImageBitmap(bitmap);
        }
        catch (Exception e){
            e.printStackTrace();
```

MainActivity › onCreate()

Activate Windows
Go to Settings to activat

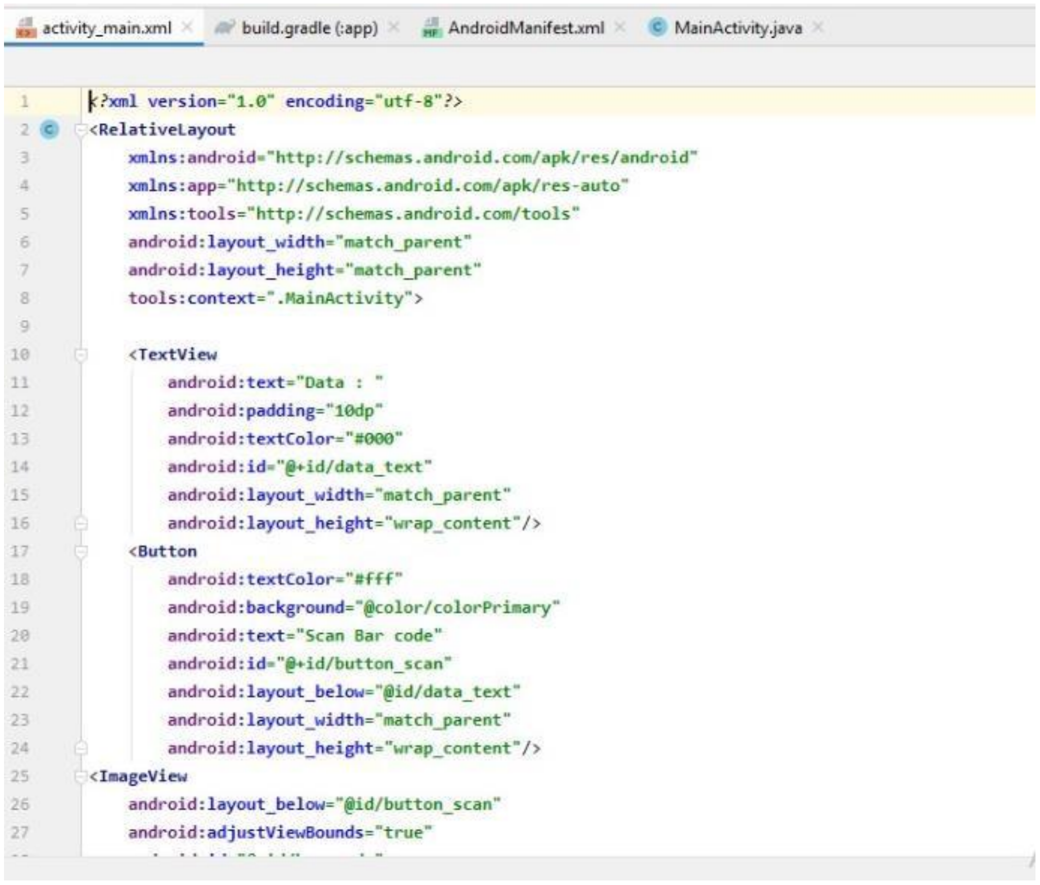
Figure 19



```java
        Button scan_code=findViewById(R.id.button_scan);
        scan_code.setOnClickListener((v) → {
                if(Build.VERSION.SDK_INT>=23){
                    if(checkPermission(Manifest.permission.CAMERA)){
                        openScanner();
                    }
                    else{
                        requestPermission(Manifest.permission.CAMERA,CAMERA_PERMISSION_CODE);
                    }
                }
                else{
                    openScanner();
                }
        });

        //now let's share qr code
        Button share_code=findViewById(R.id.share_code);
        share_code.setOnClickListener((v) → {
                if(Build.VERSION.SDK_INT>=23){
                    if(checkPermission(Manifest.permission.WRITE_EXTERNAL_STORAGE)){
                        shareQrCode();
                    }
                    else{
                        requestPermission(Manifest.permission.WRITE_EXTERNAL_STORAGE,FILE_SHARE_PERMISSION);
                    }
                }
                else{
                    shareQrCode();
```

Figure 20

UI code:

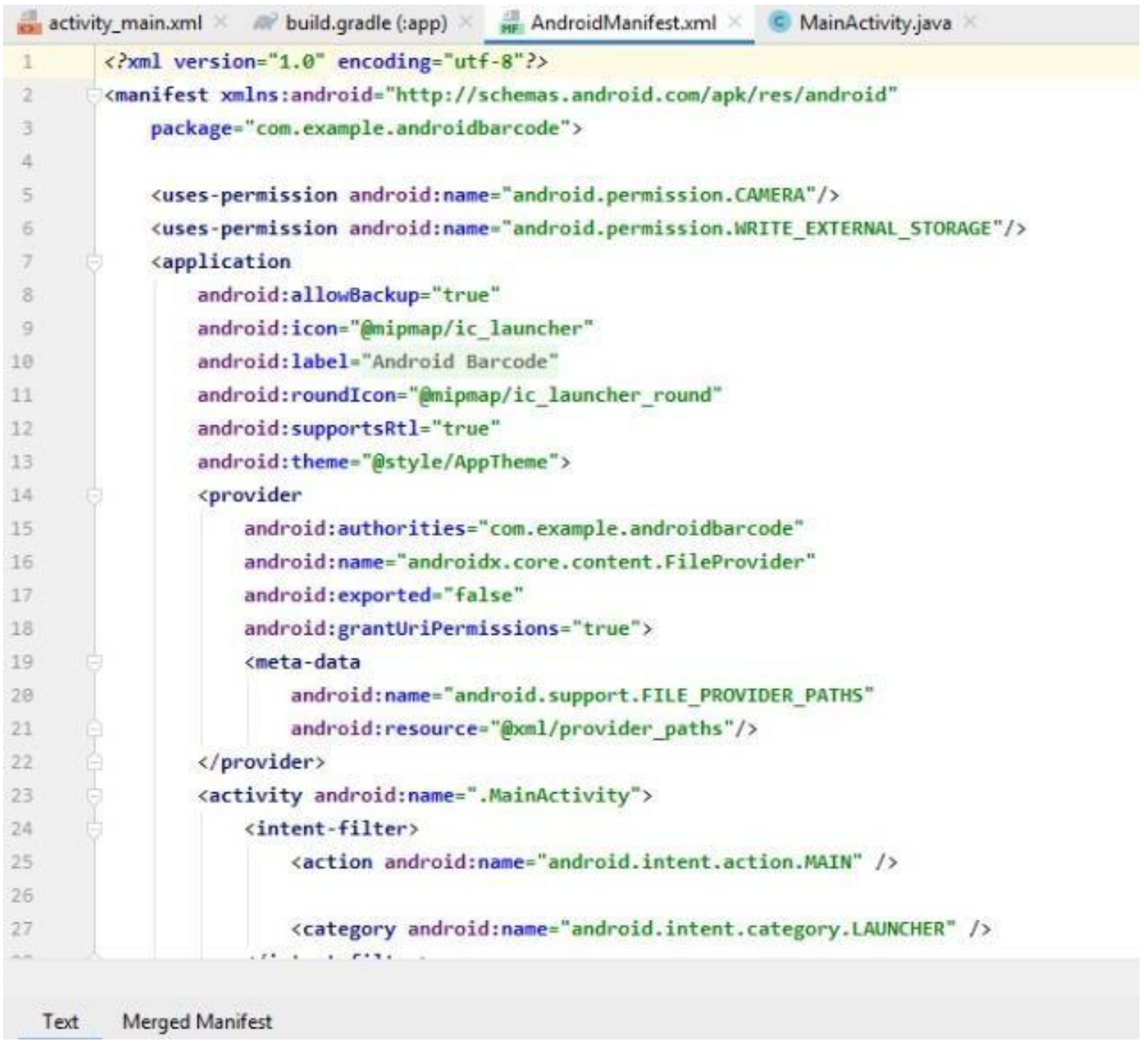


activity_main.xml × build.gradle (:app) × AndroidManifest.xml × MainActivity.java ×

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

    <TextView
        android:text="Data : "
        android:padding="10dp"
        android:textColor="#000"
        android:id="@+id/data_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
    <Button
        android:textColor="#fff"
        android:background="@color/colorPrimary"
        android:text="Scan Bar code"
        android:id="@+id/button_scan"
        android:layout_below="@id/data_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
<ImageView
    android:layout_below="@id/button_scan"
    android:adjustViewBounds="true"
```

Figure 21

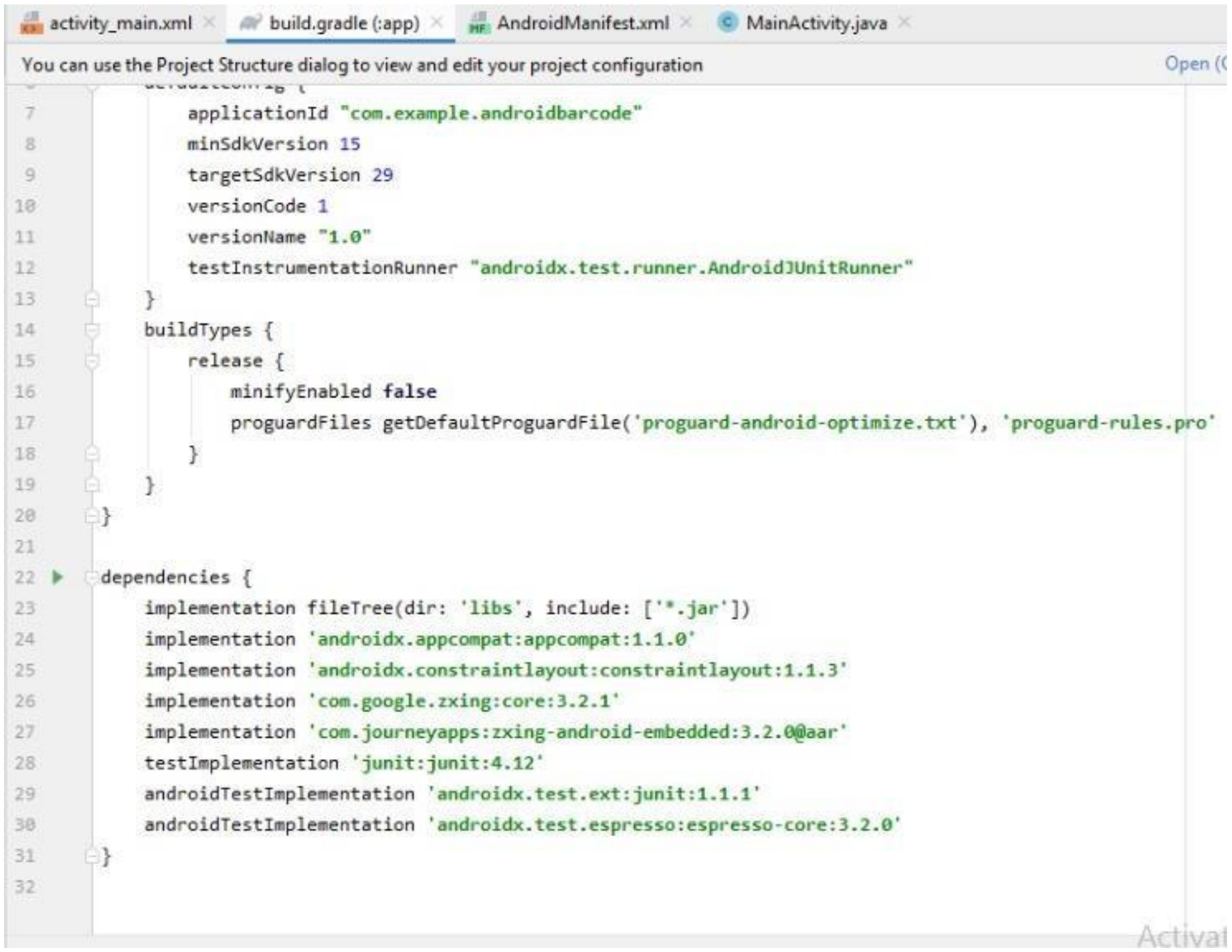
Manifest.xml file:



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

    <uses-permission android:name="android.permission.CAMERA"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Android Barcode"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <provider
            android:authorities="com.example.androidbarcode"
            android:name="androidx.core.content.FileProvider"
            android:exported="false"
            android:grantUriPermissions="true">
            <meta-data
                android:name="android.support.FILE_PROVIDER_PATHS"
                android:resource="@xml/provider_paths"/>
        </provider>
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
```

Figure 22

Inbuilt dependencies:



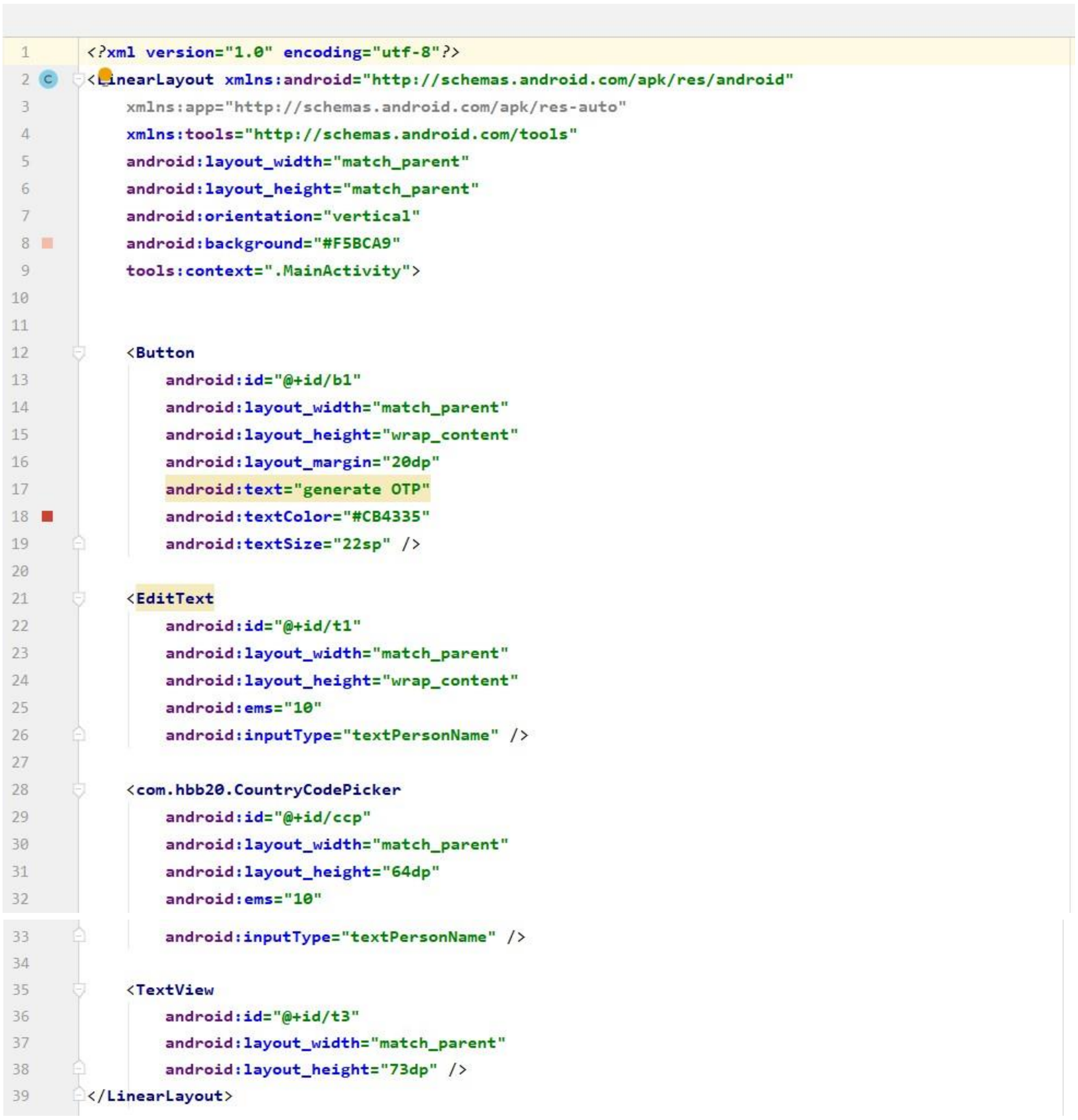
activity_main.xml
build.gradle (:app)
AndroidManifest.xml
MainActivity.java
You can use the Project Structure dialog to view and edit your project configuration
Open (

```
        applicationId "com.example.androidbarcode"
        minSdkVersion 15
        targetSdkVersion 29
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    implementation 'com.google.zxing:core:3.2.1'
    implementation 'com.journeyapps:zxing-android-embedded:3.2.0@aar'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.1'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
}
```

Figure 23

4.3.3 Android OTP Generator

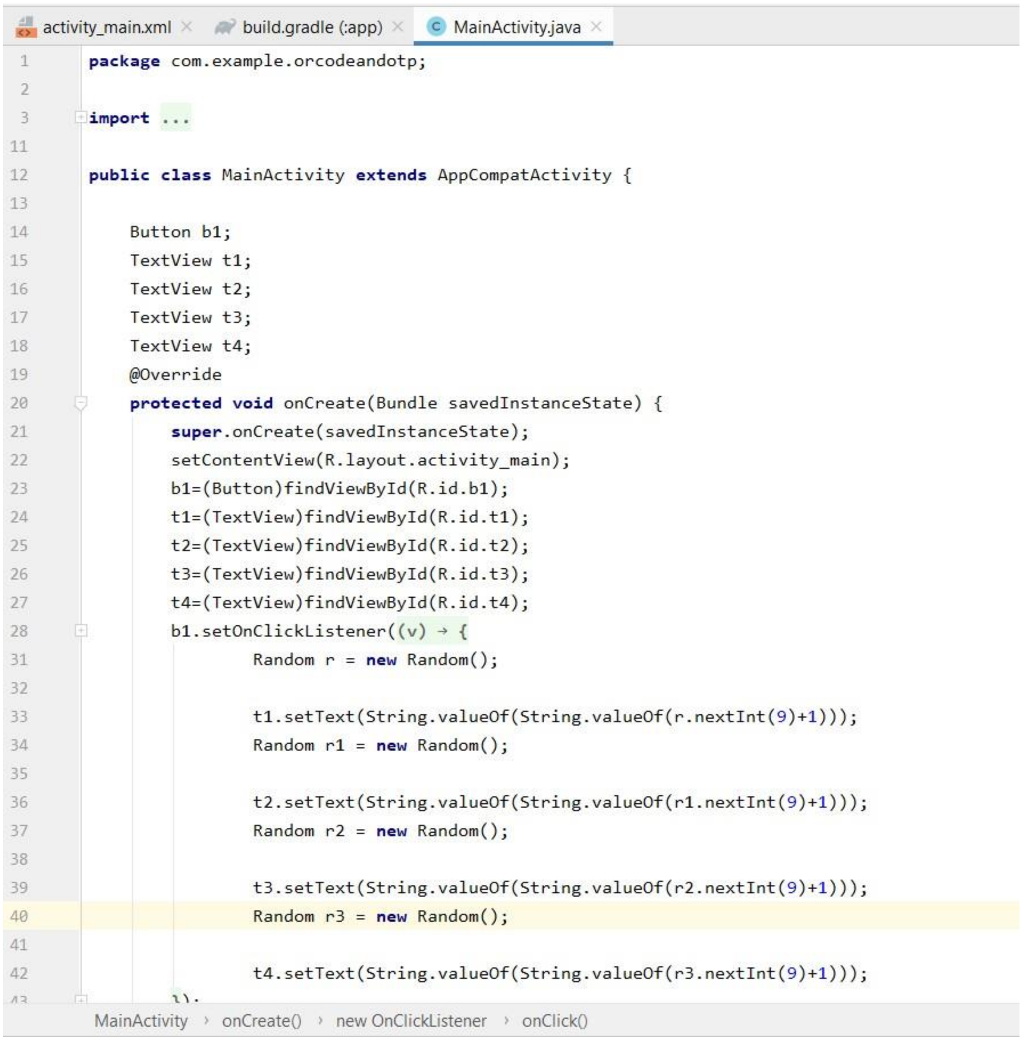
UI code:



```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="#F5BCA9"
    tools:context=".MainActivity">


    <Button
        android:id="@+id/b1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:text="generate OTP"
        android:textColor="#CB4335"
        android:textSize="22sp" />

    <EditText
        android:id="@+id/t1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="textPersonName" />

    <com.hbb20.CountryCodePicker
        android:id="@+id/ccp"
        android:layout_width="match_parent"
        android:layout_height="64dp"
        android:ems="10"
        android:inputType="textPersonName" />

    <TextView
        android:id="@+id/t3"
        android:layout_width="match_parent"
        android:layout_height="73dp" />
</LinearLayout>
```

Figure 24

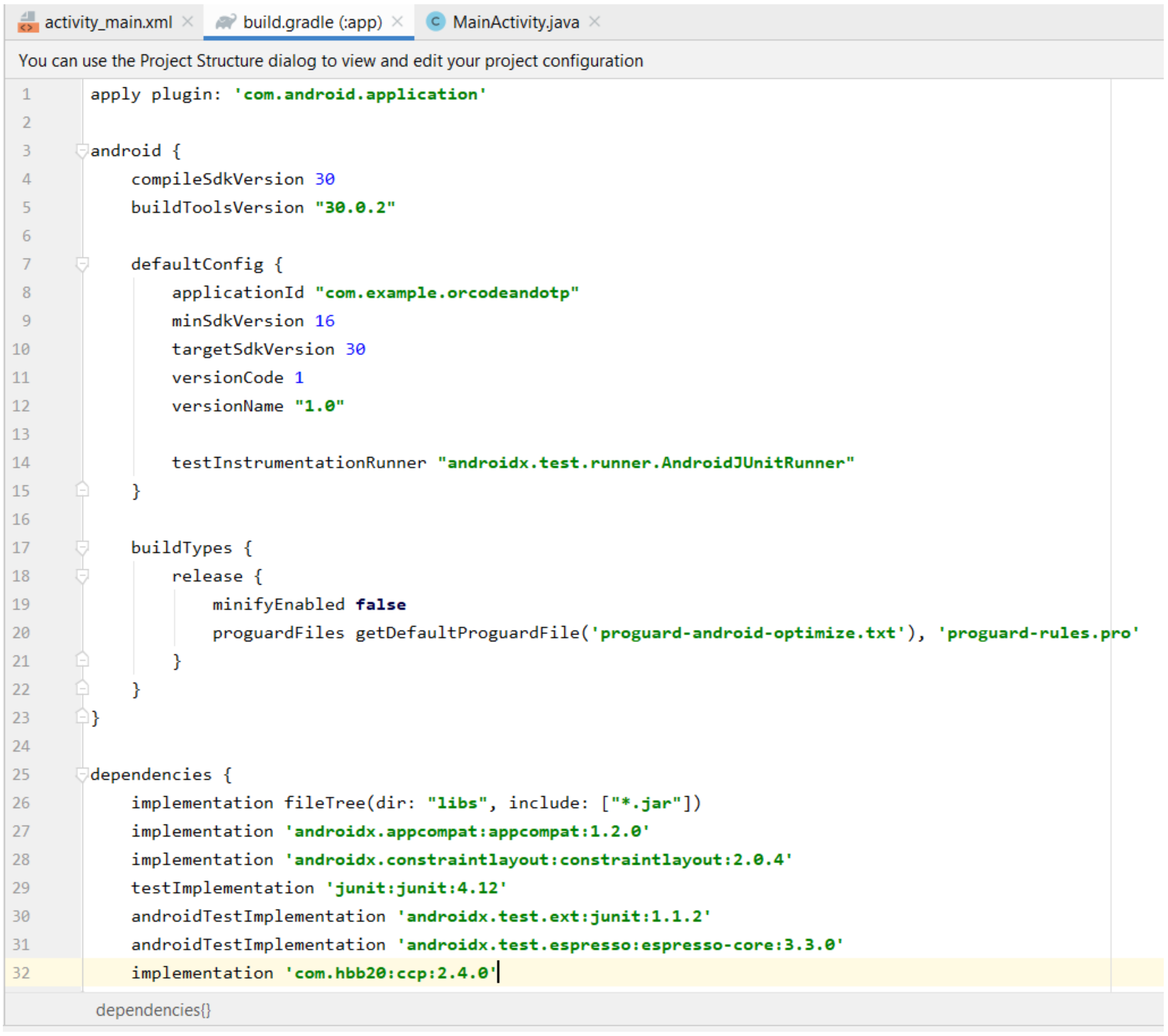
MainActitvity.java:



```java
package com.example.orcodeandotp;

import ...

public class MainActivity extends AppCompatActivity {

    Button b1;
    TextView t1;
    TextView t2;
    TextView t3;
    TextView t4;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        b1=(Button)findViewById(R.id.b1);
        t1=(TextView)findViewById(R.id.t1);
        t2=(TextView)findViewById(R.id.t2);
        t3=(TextView)findViewById(R.id.t3);
        t4=(TextView)findViewById(R.id.t4);
        b1.setOnClickListener((v) → {
                Random r = new Random();

                t1.setText(String.valueOf(String.valueOf(r.nextInt(9)+1)));
                Random r1 = new Random();

                t2.setText(String.valueOf(String.valueOf(r1.nextInt(9)+1)));
                Random r2 = new Random();

                t3.setText(String.valueOf(String.valueOf(r2.nextInt(9)+1)));
                Random r3 = new Random();

                t4.setText(String.valueOf(String.valueOf(r3.nextInt(9)+1)));
```

Figure 25

Inbuilt Dependencies:



activity_main.xml build.gradle (:app) MainActivity.java

You can use the Project Structure dialog to view and edit your project configuration

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 30
    buildToolsVersion "30.0.2"

    defaultConfig {
        applicationId "com.example.orcodeandotp"
        minSdkVersion 16
        targetSdkVersion 30
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(dir: "libs", include: ["*.jar"])
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'
    implementation 'com.hbb20:ccp:2.4.0'
```

dependencies{}

Figure 26

Chapter -5

Conclusion & Work Done in all 3 Phases

5.1 Phases:

In 5th semester, we got the great opportunity to build our own project. So, we decided to build a project on the topic “Authentication System Using QR Code & OTP”. We decided to divide our project in three phases in first phase we design the flow of the project and basic UI design, then in second phase we build an iOS application of our project and little part of android application and in third phase we fully build our android application.

5.1.1 Phase I:

In this phase, we firstly understand the meaning of our project and decided the control flow of our project, then we understand the technologies that will be used in our project, after understanding and chosen the right technologies, firstly we build a prototype of our project and finally we build an actual UI(User Interface) of our project’s android application.

5.1.2 Phase II:

In this phase, we tried to build our project’s iOS version as well as android version, we firstly build an UI(User Interface) of iOS in swift and then also build our fully working iOS application. Simultaneously, we build our android application and we just left over with some connectivity and some bugs which will be completed in Phase 3.

5.1.3 Phase III:

In this phase, we fully build our working android application and we tried to improve some stuff in our iOS application. So, our both android as well as iOS application of our project is ready to function in this phase and by this phase, we have completed our project.

5.2 Conclusion:

Towards the end of this report, on behalf of my team, I would just like to say that we count ourselves lucky to have taken this first step of this journey. It has enriched our knowledge by many folds, and we can already feel our horizons broadened. Such projects have the power to change the life of a student and can give them new innovations.

This phase of the project has especially been the most incredible phase. With all the heavy lifting like, continuous debugging and tireless efforts towards the successful making the application, done. Now at the end of this phase, we all felt really relaxed and a lot more satisfied. But the thing is that the journey is not over. And we have a very important part ahead of us. Where we will try to remove all the runtime bugs which may arrive in the day-to-day working of the application. Along with that it is our plan to optimize the algorithm used by and make the application more efficient. We all are really excited in the anticipation of this upcoming phase.

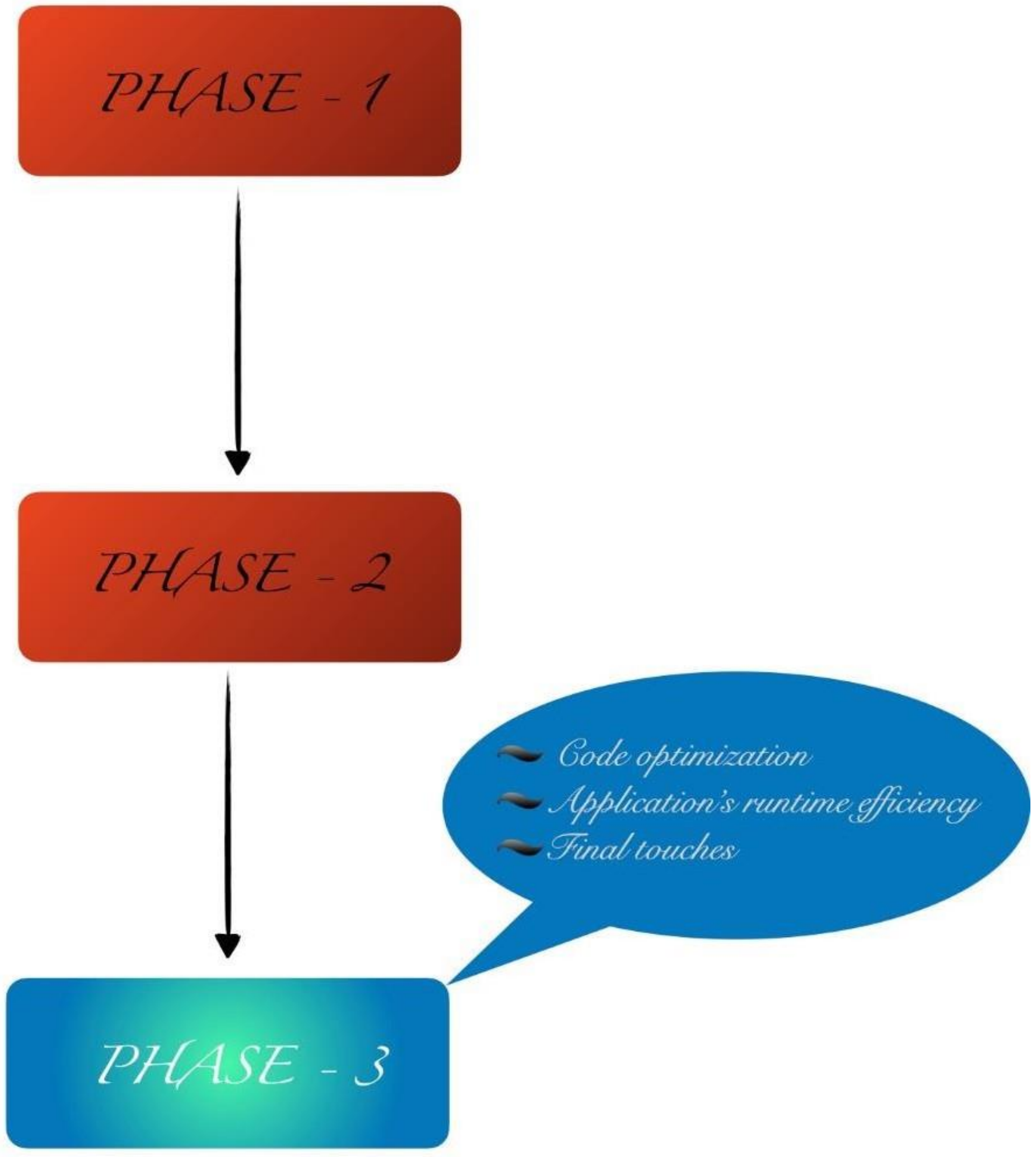

PHASE - 1
PHASE - 2
Code optimization
Application's runtime efficiency
Final touches
PHASE - 3

Figure 27

Bibliography

1. https://en.wikipedia.org/wiki/RSA_(cryptosystem)

2. https://en.wikipedia.org/wiki/Cryptography

3. https://www.wikipedia.org

4. https://developer.android.com/docs

5. https://www.oracle.com/java/technologies/javase-jdk8-doc-downloads.html

6. https://docs.swift.org/swift-book/

7. https://developer.apple.com/documentation/xcode/

8. Core Java by Cay S.Horstmann

IBMG4 Final Plag

ORIGINALITY REPORT

20%	19%	2%	10%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	egovernance.gov.in Internet Source	4%
2	Submitted to DIT university Student Paper	3%
3	www.coursehero.com Internet Source	2%
4	en.wikipedia.org Internet Source	1%
5	state.1barcode.com Internet Source	1%
6	baadalsg.inflibnet.ac.in Internet Source	1%
7	Submitted to Nottingham Trent University Student Paper	1%
8	Submitted to Uttar Pradesh Technical University Student Paper	1%
9	www.vmware.com Internet Source	1%

10	baixardoc.com Internet Source	1 %
11	www.mdpi.com Internet Source	1 %
12	Submitted to Middlesex University Student Paper	1 %
13	Submitted to National University of Ireland, Galway Student Paper	1 %
14	ijarcs.info Internet Source	<1 %
15	towardsdatascience.com Internet Source	<1 %
16	Submitted to North Warwickshire & Hinckley College Student Paper	<1 %
17	www.intechopen.com Internet Source	<1 %
18	www.itmsoc.org Internet Source	<1 %
19	www.semanticscholar.org Internet Source	<1 %
20	www.scribd.com Internet Source	<1 %