آموزش برنامه نویسی اندروید
در محیط اندروید استودیو

ارسال پیامک در برنامه اندرویدی

مدرس : سیدمهدی مطهری

www.android-studio.ir

به نام خدا

یکی از قابلیت‌هایی که در برخی از برنامه‌های اندرویدی مورد استفاده قرار می‌گیرد، ارسال پیامک است. مواردی مانند ارسال داده‌های کم حجم، ارسال موقعیت جغرافیایی و... از کاربردهای این قابلیت بشمار می‌رود. در این جلسه به نحوه ارسال پیامک (SMS) در برنامه نویسی اندروید می‌پردازیم.

همچنین بررسی وضعیت سیم کارت (فعال یا غیر فعال بودن و...) و دریافت گزارش ارسال و گزارش تحویل پیامک از دیگر مواردی است که در این جلسه به آن پرداخته خواهد شد.

کاربردهای ارسال پیامک از طریق اپلیکیشن

ارسال پیامک از طریق برنامه‌های اندرویدی کاربردهای متفاوتی می‌تواند داشته باشد. برای مثال در برخی از همراه بانک‌ها کاربر امکان انتخاب اینترنت یا پیامک (SMS) را به عنوان بستر ارسال و دریافت اطلاعات در اختیار دارد.

همچنین یکی دیگر از کاربردهای ارسال پیامک (SMS) در برنامه نویسی اندروید ارسال موقعیت جغرافیایی فرد (بر اساس مختصات GPS) به یک شماره خاص است. قابلیتی که به فرد این امکان را می‌دهد تا در شرایط اضطراری، با لمس یک دکمه موقعیت جغرافیایی خود را برای شماره موبایلی که قبلا در برنامه تعریف کرده ارسال کند. یا برنامه‌ای که بر روی دستگاه اندرویدی کودکان نصب می‌شود تا در فاصله‌های زمانی معین، مختصات جغرافیایی او را به صورت خودکار به شماره همراه والدینش ارسال می‌کند.

ارسال پیامک به مرکز سرویس دهنده‌ی اپلیکیشن به جهت تائید شماره همراه کاربر در هنگام عضویت یکی دیگر از کاربردهای این قابلیت است.

قابلیت ارسال پیامک از اپلیکیشن کاربردهای متعدد دیگری می‌تواند داشته باشد که استفاده از آن به ماهیت برنامه بستگی دارد.

نحوه ارسال پیامک در اندروید

ارسال پیامک (SMS) در برنامه نویسی اندروید به دو طریق امکان پذیر است. روش نخست استفاده از intent است که در صورت پیاده سازی این روش در برنامه، کاربر برای ارسال پیامک به برنامه مدیریت پیامک پیش فرض دستگاه خود منتقل خواهد شد.

اما در روش دوم با استفاده از کلاس SmsManager فرایند ارسال پیامک درون خود برنامه و در پشت صحنه انجام شده و کاربر از جابجایی بین برنامه اصلی و برنامه مدیریت SMS خود بی نیاز خواهد بود. علاوه بر آن دسترسی به محتوای پیامک و امکان ویرایش آن نیز مقدور نیست.

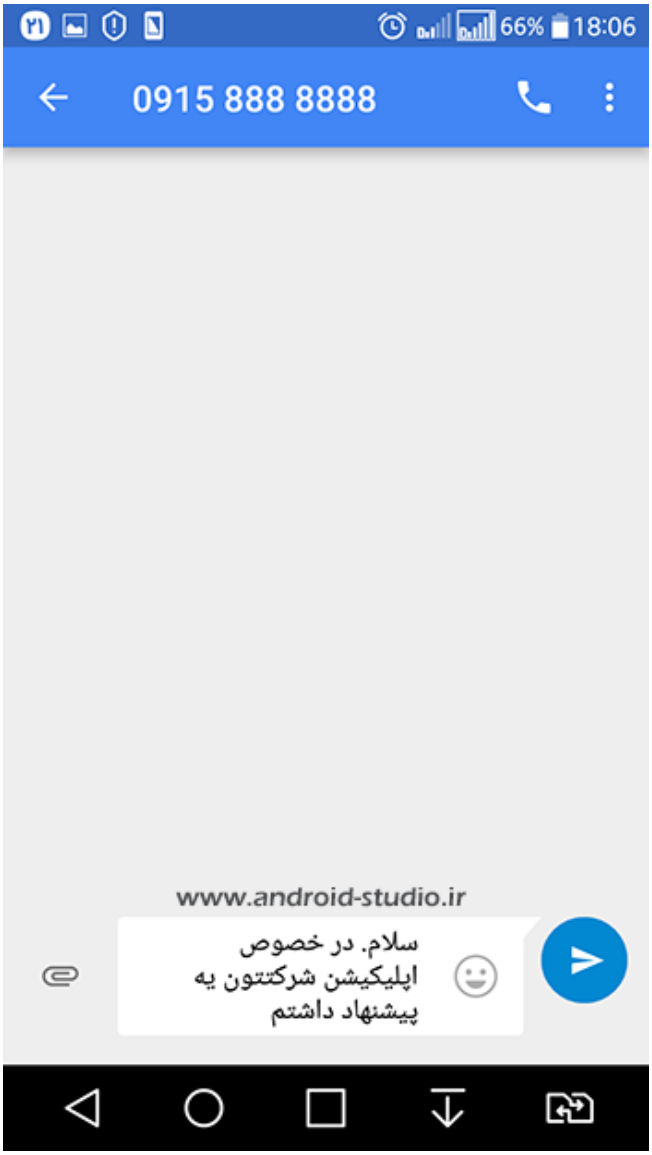
در ادامه جلسه به بررسی هردو روش می‌پردازیم.

ارسال پیامک (SMS) از طریق Intent

قبلا در جلسه آموزش کار با intent در اندروید با کاربرد اینتنت‌ها در برنامه نویسی اندروید آشنا شدیم. ضمن اینکه در همان جلسه یکی از مثال‌های کاربرد intent به قابلیت ارسال پیامک از طریق برنامه مدیریت SMS پیش فرض دستگاه اختصاص یافته بود که توضیحات کامل را می‌توانید مطالعه کنید. بنابراین از پرداختن مجدد به توضیحات این روش اجتناب می‌کنم و صرفا جهت یاداوری، کد اینتنت مربوط به ارسال پیامک از طریق برنامه مدیریت پیامک را در اینجا ذکر می‌کنم:

```
public void contactMessage(View v) {
    Intent cnt_msg = new Intent();
    cnt_msg.setAction(Intent.ACTION_VIEW);
    cnt_msg.setData(Uri.parse("sms:+989158888888"));
    cnt_msg.putExtra("sms_body", "سلام . در خصوص اپلیکیشن شرکتتون یه پیشنهاد داشتم");
    startActivity(cnt_msg);
}
```

با اجرای تابع فوق، کاربر به برنامه پیامک پیش فرض دیوایس اندرویدی منتقل شده و شماره‌ی گیرنده پیام و همچنین متن پیامک به صورت خودکار اضافه می‌شود:



در اینجا کاربر در صورت تمایل قبل از ارسال پیامک می‌تواند محتوای پیام را ویرایش کند.

اما این شیوه ارسال پیامک به دلیل طولانی بودن فرایند ارسال و همچنین دسترسی کاربر به محتوای آن کاربرد چندانی نداشته و برنامه نویسان اندرویدی عموما روش دوم را بکار می‌گیرند که در ادامه مبحث معرفی می‌کنم.

ارسال پیامک (SMS) از طریق کلاس SmsManager

بر خلاف روش قبل یعنی ارسال پیامک از طریق intent در این روش و با استفاده از کلاس SmsManager پیامک به طور مستقیم و درون خود برنامه به مقصد ارسال شده و کاربر دسترسی به محتوای پیامک نخواهد داشت. به عبارت دیگر، کاربر از فرایند ارسال پیامک آگاه نخواهد شد.

با استفاده از SmsManager می‌توان ۴ نوع پیام را ارسال کرد که لازم است کاربرد هرکدام را توضیح دهم:

sendTextMessage: همانطور که از نام این متد پیداست، برای ارسال پیامک (SMS) معمولی بکار می‌رود. یعنی ارسال پیامک متنی در قالب یک پیام و به مقصد از قبل تعیین شده. درست همان چیزی که در این جلسه نیاز داریم.

sendMultipartTextMessage: عملیات مشابه متد قبل را انجام می‌دهد با این تفاوت که می‌توان پیام را در قالب چند پیام جداگانه به مقصد ارسال کرد.

sendDataMessage: از این متد برای ارسال پیام داده محور و در یک پورت مشخص استفاده می‌شود.

sendMultimediaMessage: چنانچه قصد ارسال پیام مولتی مدیا یا همان MMS (مانند ارسال عکس) داشته باشیم باید از این متد استفاده کنیم.

همانطور که در توضیحات قبل اشاره شد، در این جلسه می‌خواهیم یک پیامک متنی ساده و تک قسمتی را ارسال کنیم که متد sendTextMessage مناسب این کار است. با ادامه جلسه همراه باشید.

ساخت پروژه اندرویدی ارسال پیامک

مطابق مبحث آموزش ساخت پروژه در اندروید استودیو یک پروژه اندرویدی با نام SendSMS می‌سازم. اکتیویتی را از نوع Empty Activity و زبان را Java انتخاب کردم.

در این پروژه قصد داریم با استفاده از کلاس SmsManager بصورت درون برنامه‌ای و بدون نیاز به برنامه مدیریت پیامک، یک پیام کوتاه به مقصدی مشخص ارسال کنیم. همچنین نحوه فعالسازی گزارش ارسال (sent) و گزارش تحویل (delivery) پیامک و در دسترس بودن یا نبودن سیم کارت جهت ارسال SMS را نیز بررسی خواهیم کرد.

در رابط کاربری این پروژه تنها به یک دکمه برای ارسال متن پیامک از پیش تعیین شده نیاز داریم. بنابراین در layout پیش فرض اکتیویتی پروژه کافیست TextView موجود در مرکز ویو گروپ ConstraintLayout را به Button تبدیل کرده و یک id به آن اختصاص دهم:

activity_main.xml

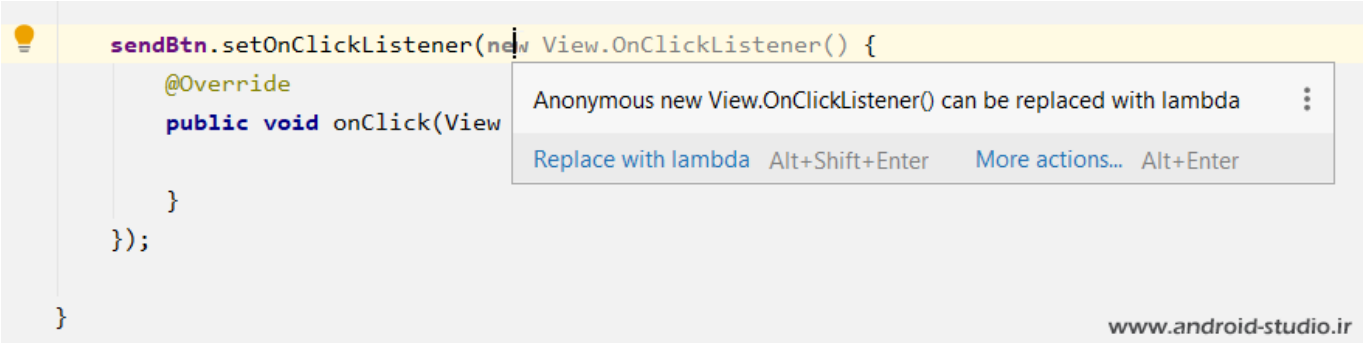
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
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

    <Button
        android:id="@+id/sms_btn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="ارسال پیامک"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

اگر با این layout آشنایی ندارید آموزش کار با ConstraintLayout را مطالعه کنید. حالا رویداد مربوط به دکمه را در اکتیویتی تعریف می‌کنم:

MainActivity.java

```
package ir.android_studio.sendsms;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {

    Button sendBtn;

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

        sendBtn = findViewById(R.id.sms_btn);

        sendBtn.setOnClickListener(view -> {

        });

    }
}
```

نکته: در نسخه‌های اخیر اندروید استودیو، هنگام اضافه کردن متد setOnClickListener پیشنهادی مبنی بر جایگزینی View.OnClickListener با lambda داده می‌شود که با زدن alt+Enter یا کلیک روی آیکون لامپ، جایگزینی انجام می‌شود. تفاوت در کوتاه شدن کدهاست.



پس از تبدیل، کد رویداد دکمه به اینصورت تبدیل می‌شود:

```
sendBtn.setOnClickListener(view -> {

});
```

تعریف مجوز ارسال پیامک

در قدم اول لازم است مجوز یا Permission ارسال پیامک (SMS) را در برنامه تعریف کنیم. ابتدا پرمیشن SEND_SMS را در مانیفست پروژه تعریف می‌کنم:

```
<uses-permission android:name="android.permission.SEND_SMS"/>
```

AndroidManifest.xml

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

    <uses-permission android:name="android.permission.SEND_SMS"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.SendSMS">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

```
    </application>

</manifest>
```

همچنین مطابق جلسه آموزش پیاده سازی Runtime Permission برای دریافت مجوز در دستگاه‌های اندرویدی ۶ و به بالا کدهای لازم را به پروژه اضافه می‌کنم:

MainActivity.java

```
package ir.android_studio.sendsms;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

import android.Manifest;
import android.content.DialogInterface;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    Button sendBtn;
    private final int SMS_REQUEST_CODE = 100;

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

        sendBtn = findViewById(R.id.sms_btn);

        sendBtn.setOnClickListener(view -> {

            if (ContextCompat.checkSelfPermission(MainActivity.this,
Manifest.permission.SEND_SMS) != PackageManager.PERMISSION_GRANTED) {

                requestSendSMSpermission();

            } else {

                sendMessage();

            }

        });

    }
```

```
    private void requestSendSMSpermission() {

        if (ActivityCompat.shouldShowRequestPermissionRationale(MainActivity.this,
Manifest.permission.SEND_SMS)) {

            new AlertDialog.Builder(this)
                    .setTitle("درخواست مجوز")
                    .setMessage(" برای عملکرد صحیح برنامه باید دسترسی به ارسال پیامک
تایید شود")
                    .setPositiveButton("موافقم", new DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialogInterface, int i) {

                            reqPermission();

                        }
                    })
                    .setNegativeButton("لغو", new DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialogInterface, int i) {

                            dialogInterface.dismiss();

                        }
                    })
                    .create()
                    .show();

        } else {

            reqPermission();

        }

    }

    private void reqPermission() {

        ActivityCompat.requestPermissions(MainActivity.this, new String[]
{Manifest.permission.SEND_SMS}, SMS_REQUEST_CODE);

    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
@NonNull int[] grantResults) {

        if (requestCode == SMS_REQUEST_CODE) {

            if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {

                sendMessage();

            } else {

                Toast.makeText(this, "مجوز رد شد", Toast.LENGTH_SHORT).show();

            }
```

```
        }

    }

}
```

پیاده سازی کلاس SmsManager

با اضافه شدن کدهای فوق، پس از لمس دکمه "ارسال پیامک" چنانچه کاربر مجوز ارسال SMS را به برنامه اعطا کند، تابع sendMessage فراخوانی خواهد شد. تابعی با همین نام درون اکتیویتی و بعد از متد onCreate اضافه کرده و کدهای مربوط به ارسال پیامک را درون آن می‌نویسم:

```
private void sendMessage() {

    try {

        SmsManager smsManager = SmsManager.getDefault();
        smsManager.sendTextMessage("+989158888888", null, "تست ارسال پیامک", null, null);

        Toast.makeText(MainActivity.this, "پیامک ارسال شد ", Toast.LENGTH_SHORT).show();

    } catch (Exception e) {

        Toast.makeText(MainActivity.this, "پیامک ارسال نشد ", Toast.LENGTH_SHORT).show();

    }

}
```

کدهای مربوط به ارسال پیامک را درون یک try catch قرار داده‌ام تا چنانچه به هر دلیلی فرایند ارسال SMS انجام نشد، برنامه کرش نکند.

در کد فوق ابتدا یک نمونه (شیء) از کلاس SmsManager با نام دلخواه smsManager ایجاد شده که برای مقدار دهی آن، متد getDefault فراخوانی شده است. این متد شناسه سیم کارت پیش فرض دستگاه اندرویدی را دریافت می‌کند که برای ارسال پیامک ضروری است. توصیه می‌کنم توضیحات تکمیلی اینجا و اینجا را در خصوص دستگاه‌های دو سیم کارت مطالعه کنید.

در خط دوم، متد sendTextMessage تعریف شده که ۴ پارامتر ورودی می‌پذیرد. در تصویر زیر جنس هر ورودی قابل تشخیص است:

```
smsManager.sendTextMessage( destinationAddress: "+989158888888", scAddress: null, text: "تست ارسال پیامک", sentIntent: null, deliveryIntent: null);
```

destinationAddress: پارامتر نخست از جنس String بوده که همان شماره گیرنده پیام می‌باشد.

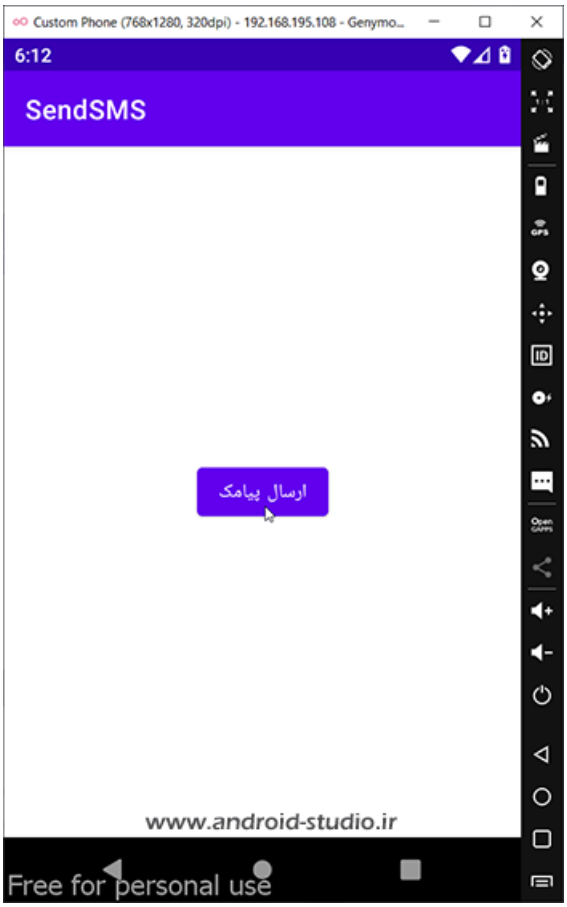
scAddress: پارامتر دوم مربوط به آدرس یا شماره‌ی Service Center پیامک اپراتوری است که کاربر از سیم کارت آن استفاده می‌کند. به طور معمول این شماره توسط کلاینت تعیین نمی‌شود و به عهده‌ی خود اپراتور است. بنابراین مقدار null را وارد می‌کنیم.

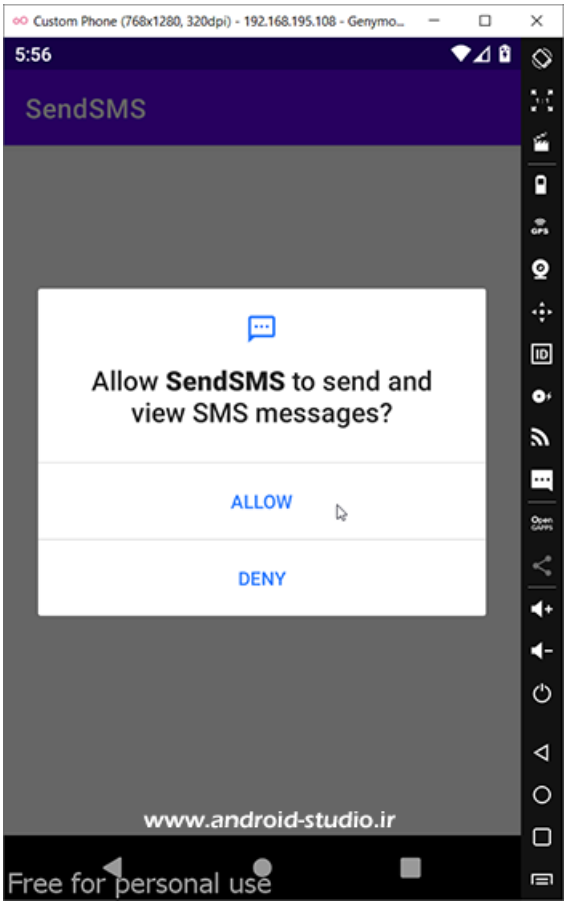
text: واضح است که این پارامتر، محتوای پیام را شامل می‌شود.

sentIntent و deliveryIntent: به ترتیب برای دریافت گزارش ارسال و گزارش تحویل پیامک استفاده می‌شود که در این مرحله به جهت درک بهتر عملکرد اصلی کلاس، برای هردو مقدار null درنظر گرفتم. بنابراین در این مرحله به نتیجه‌ی ارسال و یا تحویل پیامک به مقصد دسترسی نخواهیم داشت.

نگران نباشید! به این دو مورد هم در ادامه جلسه می‌پردازیم.

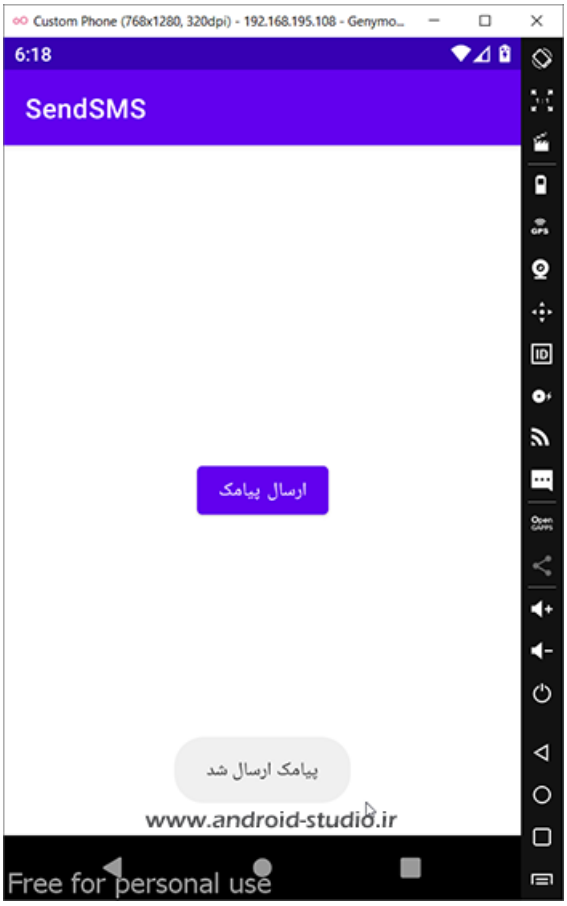
در همین مرحله پروژه را اجرا می‌کنم تا عملکرد کلاس SmsManager را بررسی کنیم.



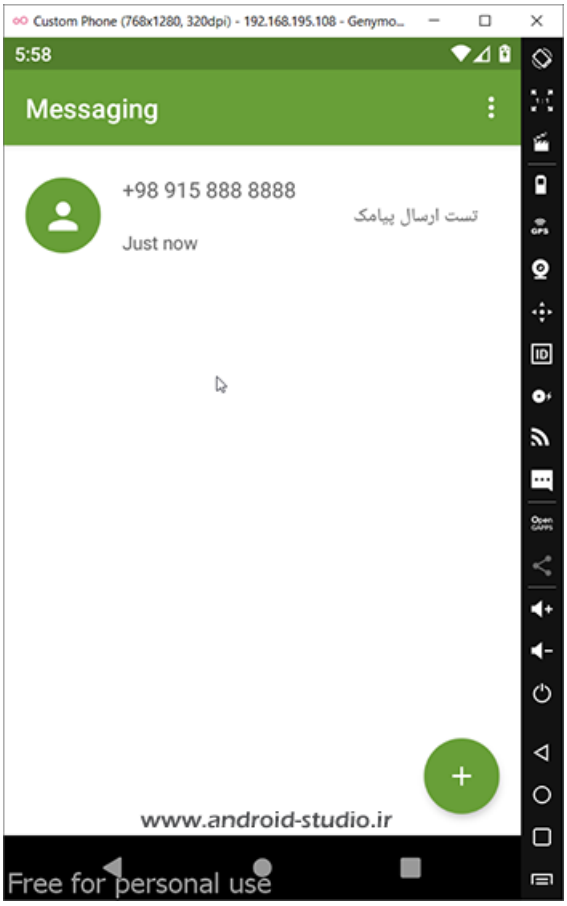


با توجه به اینکه یک دیوایس با Android 10 را روی شبیه ساز (امولاتور) اندرویدی اجرا کرده‌ام، با کلیک روی دکمه ارسال پیامک ابتدا باید مجوز ارسال پیامک را تایید کنم.

با تایید مجوز، پیغام "پیامک ارسال شد" روی صفحه ظاهر شد که نشان می‌دهد عملکرد برنامه تا اینجای کار درست بوده:



پیامک ارسال شده در برنامه مدیریت SMS پیش فرض دستگاه نیز قابل مشاهده است:



دریافت گزارش ارسال و گزارش تحویل پیامک

در قسمت قبل صرفا یک SMS توسط کلاس SmsManager ارسال کردیم و دسترسی به گزارش ارسال و تحویل پیامک نداشتیم و برای هردو پارامتر sentIntent و deliveryIntent مقدار null داده بودیم.

اما در این قسمت و در ادامه‌ی آموزش ارسال پیامک (SMS) در برنامه نویسی اندروید با استفاده از BroadcastReceiver و PendingIntent رویدادهای مربوط به وضعیت ارسال و تحویل پیامک را شنود کرده و پیغام مربوط آن را چاپ می‌کنیم.

ابتدا در بدنه اصلی کلاس اکتیویتی دو متغیر از جنس String با نام و مقدار دلخواه تعریف می‌کنم. از این دو متغیر برای برقراری ارتباط بین BroadcastReceiver و PendingIntent مربوط به یکدیگر استفاده خواهیم کرد:

```
String SMS_SENT = "SMS_SENT";
String SMS_DELIVERED = "SMS_DELIVERED";
```

سپس مطابق جلسه آموزش شنود رویدادها در اندروید توسط BroadcastReceiver دو برودکست را درون متد sendMessage داخل بلاک try و قبل از smsManager، رجیستر می‌کنم:

```
//Broadcast for Sent SMS
registerReceiver(new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {

        String state = "";
        switch (getResultCode()) {
            case Activity.RESULT_OK:
                state = "پیامک ارسال شد";
                break;
            case SmsManager.RESULT_ERROR_GENERIC_FAILURE:
                state = "یک خطای عمومی رخ داد";
                break;
            case SmsManager.RESULT_ERROR_NO_SERVICE:
                state = "اپراتور در دسترس نیست";
                break;
            case SmsManager.RESULT_ERROR_NULL_PDU:
                state = "در دسترس نیست PDU پروتکل";
                break;
            case SmsManager.RESULT_ERROR_RADIO_OFF:
                state = "سیم کارت در دسترس نیست";
                break;
        }
        Toast.makeText(context, state, Toast.LENGTH_SHORT).show();
    }
}, new IntentFilter(SMS_SENT));

//Broadcast for Delivered SMS
registerReceiver(new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        String state = "";
        switch (getResultCode()) {
            case Activity.RESULT_OK:
                state = "پیامک تحویل داده شد";
                break;
            case Activity.RESULT_CANCELED:
                state = "پیامک تحویل داده نشد";
                break;
        }
        Toast.makeText(context, state, Toast.LENGTH_SHORT).show();
    }
}, new IntentFilter(SMS_DELIVERED));
```

برودکست اول برای شنود وضعیت ارسال پیامک استفاده خواهد شد. چنانچه پیامک با موفقیت به سرویس دهنده (اپراتور) ارسال شود، پیغام "پیامک ارسال شد" و در غیر اینصورت سایر پیغام‌های مناسب وضعیت رخ داده نمایش داده خواهد شد.

برای حالت عدم ارسال پیامک، ۴ حالت مختلف تعریف شده که با توجه به وضعیت دریافتی از SmsManager، نوع خطا را به کاربر اعلام می‌کند. خطای دریافتی می‌تواند یک خطای عمومی بوده یا مربوط به در دسترس نبودن اپراتور، در دسترس نبودن سیم کارت و یا پروتکل PDU باشد.

برودکست دوم نیز برای شنود وضعیت تحویل یا عدم تحویل SMS به گیرنده بکار می‌رود.

بعد از برودکست‌ها دو PendingIntent به صورت زیر تعریف می‌کنم:

```
PendingIntent sentSMS = PendingIntent.getBroadcast(this, 0, new Intent(SMS_SENT), 0);
PendingIntent deliverSMS = PendingIntent.getBroadcast(this, 0, new Intent(SMS_DELIVERED),
0);
```

قبلا در جلسه آموزش ساخت نوتیفیکیشن‌های حرفه‌ای در اندروید با کاربرد PendingIntent ها آشنا شدیم. با استفاده از متد getBroadcast، برودکست موردنظر در PendingIntent تعریف می‌شود.

پارامتر سوم getBroadcast از جنس اینتنت بوده که شامل اکشن‌های SMS_SENT و SMS_DELIVERED می‌باشد که قبلا در برودکست‌ها تعیین کرده بودیم. همانطور که از نحوه نامگذاری PendingIntent ها پیداست، مورد اول برای اجرای برودکست ارسال پیامک و مورد دوم برای اجرای برودکست تحویل پیامک استفاده خواهد شد.

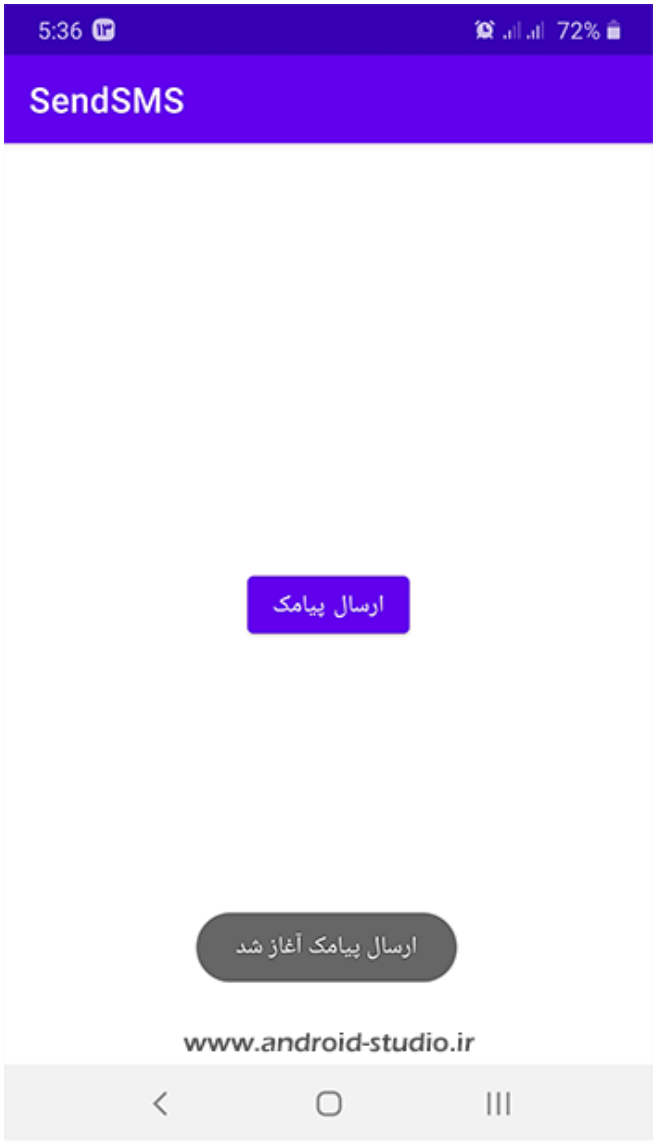
در انتها لازم است این دو PendingIntent در smsManager جایگزین مقدارهای null شوند:

```
smsManager.sendTextMessage("+989158888888", null, "تست ارسال پیامک", sentSMS, deliverSMS);
```

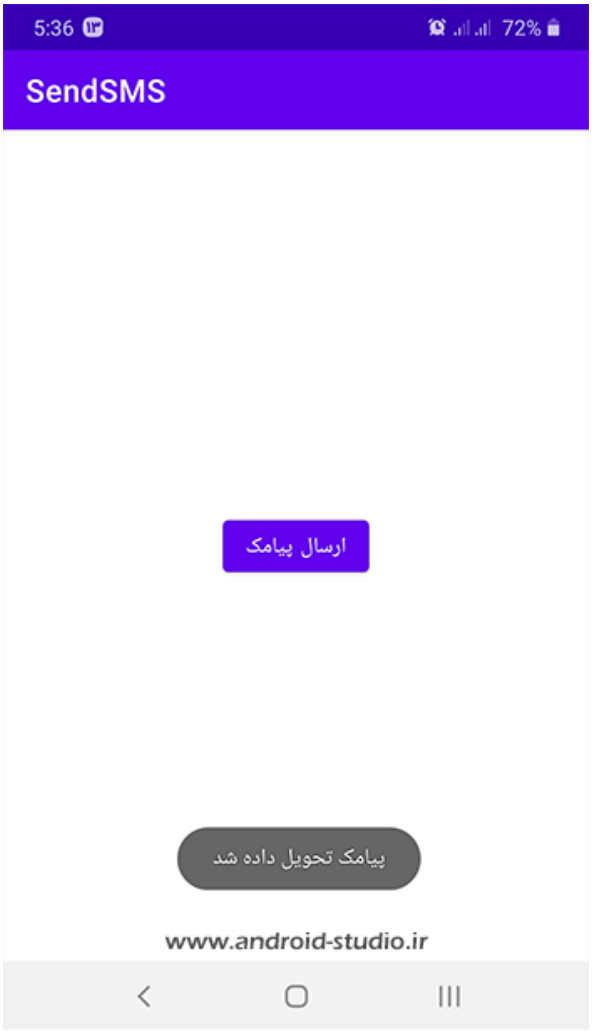
بنابراین با ارسال پیامک توسط SmsManager در آینده و هر زمانی که گزارش مربوط به وضعیت ارسال و همچنین تحویل پیامک توسط سیستم عامل منتشر شود، توسط BroadcastReceiver ها دریافت شده و پیغام مناسب روی صفحه ظاهر خواهد شد.

در نهایت، متن پیغام Toast که بعد از smsManager قرار گرفته را از "پیامک ارسال شد" به "ارسال پیامک آغاز شد" تغییر می‌دهم تا از پیغامی که در برودکست ارسال پیامک تعریف شده قابل تشخیص باشد.

مجدد پروژه را اجرا کرده و روی دکمه ارسال پیامک کلیک می‌کنم:


5:36
72%
SendSMS
ارسال پیامک
ارسال پیامک آغاز شد
www.android-studio.ir


5:36
72%
SendSMS
ارسال پیامک
پیامک ارسال شد
www.android-studio.ir



مشاهده می‌کنید گزارش مربوط به ارسال و تحویل پیامک توسط برنامه دریافت و پیغام آن نمایش داده شد.

کد نهایی اکتیویتی تا اینجای کار:

نکته: در صورت نیاز به کپی کدها بهتر است از فایل سورس کد ضمیمه آموزش استفاده کنید.

MainActivity.java

```
package ir.android_studio.sendsms;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import android.Manifest;
import android.app.Activity;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
```

```
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    Button sendBtn;
    private final int SMS_REQUEST_CODE = 100;
    String SMS_SENT = "SMS_SENT";
    String SMS_DELIVERED = "SMS_DELIVERED";

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

        sendBtn = findViewById(R.id.sms_btn);

        sendBtn.setOnClickListener(view -> {

            if (ContextCompat.checkSelfPermission(MainActivity.this,
Manifest.permission.SEND_SMS) != PackageManager.PERMISSION_GRANTED) {

                requestSendSMSpermission();

            } else {

                sendMessage();

            }

        });

    }

    private void sendMessage() {

        try {

            //Broadcast for Sent SMS
            registerReceiver(new BroadcastReceiver() {
                @Override
                public void onReceive(Context context, Intent intent) {

                    String state = "";
                    switch (getResultCode()) {
                        case Activity.RESULT_OK:
                            state = "پیامک ارسال شد";
                            break;
                        case SmsManager.RESULT_ERROR_GENERIC_FAILURE:
                            state = "یک خطای عمومی رخ داد";
                            break;
                        case SmsManager.RESULT_ERROR_NO_SERVICE:
```

```
                        state = "اپراتور در دسترس نیست";
                        break;
                    case SmsManager.RESULT_ERROR_NULL_PDU:
                        state = "در دسترس نیست PDU پروتکل";
                        break;
                    case SmsManager.RESULT_ERROR_RADIO_OFF:
                        state = "سیم کارت در دسترس نیست";
                        break;
                }
                Toast.makeText(context, state, Toast.LENGTH_SHORT).show();
            }
        }, new IntentFilter(SMS_SENT));

        //Broadcast for Delivered SMS
        registerReceiver(new BroadcastReceiver() {
            @Override
            public void onReceive(Context context, Intent intent) {
                String state = "";
                switch (getResultCode()) {
                    case Activity.RESULT_OK:
                        state = "پیامک تحویل داده شد";
                        break;
                    case Activity.RESULT_CANCELED:
                        state = "پیامک تحویل داده نشد";
                        break;
                }
                Toast.makeText(context, state, Toast.LENGTH_SHORT).show();
            }
        }, new IntentFilter(SMS_DELIVERED));

        PendingIntent sentSMS = PendingIntent.getBroadcast(this, 0, new
Intent(SMS_SENT), 0);
        PendingIntent deliverSMS = PendingIntent.getBroadcast(this, 0, new
Intent(SMS_DELIVERED), 0);

        SmsManager smsManager = SmsManager.getDefault();
        smsManager.sendTextMessage("+989158888888", null, "تست ارسال پیامک", sentSMS,
deliverSMS);

        Toast.makeText(MainActivity.this, "ارسال پیامک آغاز شد",
Toast.LENGTH_SHORT).show();

    } catch (Exception e) {

        Toast.makeText(MainActivity.this, "پیامک ارسال نشد",
Toast.LENGTH_SHORT).show();

    }

}

private void requestSendSMSpermission() {

    if (ActivityCompat.shouldShowRequestPermissionRationale(MainActivity.this,
Manifest.permission.SEND_SMS)) {

        new AlertDialog.Builder(this)
                .setTitle("درخواست مجوز")
                .setMessage("برای عملکرد صحیح برنامه باید دسترسی به ارسال پیامک
```

```
تایید شود")
                    .setPositiveButton("موافقم", new DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialogInterface, int i) {

                            reqPermission();

                        }
                    })
                    .setNegativeButton("لغو", new DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialogInterface, int i) {

                            dialogInterface.dismiss();

                        }
                    })
                    .create()
                    .show();

        } else {

            reqPermission();

        }

    }

    private void reqPermission() {

        ActivityCompat.requestPermissions(MainActivity.this, new String[]
{Manifest.permission.SEND_SMS}, SMS_REQUEST_CODE);

    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
@NonNull int[] grantResults) {

        if (requestCode == SMS_REQUEST_CODE) {

            if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {

                sendMessage();

            } else {

                Toast.makeText(this, "مجوز رد شد ", Toast.LENGTH_SHORT).show();

            }

        }

    }

}
```

بررسی وضعیت سیم کارت قبل از ارسال پیامک

در قسمت قبل پس از ارسال پیامک و توسط کلاس SmsManager وضعیت سیم کارت بررسی می‌شد و در صورت عدم امکان ارسال SMS علت عدم ارسال توسط BroadcastReceiver مشخص می‌شد.

اما با استفاده از کلاس TelephonyManager و متد getSimState می‌توانیم قبل از اقدام به ارسال SMS هم وضعیت سیم کارت را از سیستم عامل دریافت کرده و بر اساس آن عملیاتی انجام داد. برای مثال تا زمانی که سیم کارت در دسترس نباشد، دکمه ارسال پیامک در حالت غیر فعال (Disable) قرار گیرد و یا سایر روش‌های آگاه سازی کاربر به در دسترس نبودن سیم کارت که موضوع این جلسه نیست.

البته کاربرد getSimState محدود به قبل از ارسال پیامک نمی‌شود. برنامه ما ممکن است به هر دلیلی نیاز به فعال بودن سیم کارت داشته باشد؛ مانند دریافت رمز یکبار مصرف. بنابراین می‌توانیم کاربر را از وضعیت سیم کارت دستگاه مطلع کنیم.

یک متد با نام دلخواه simState درون اکتیویتی تعریف و به صورت زیر تکمیل می‌کنم:

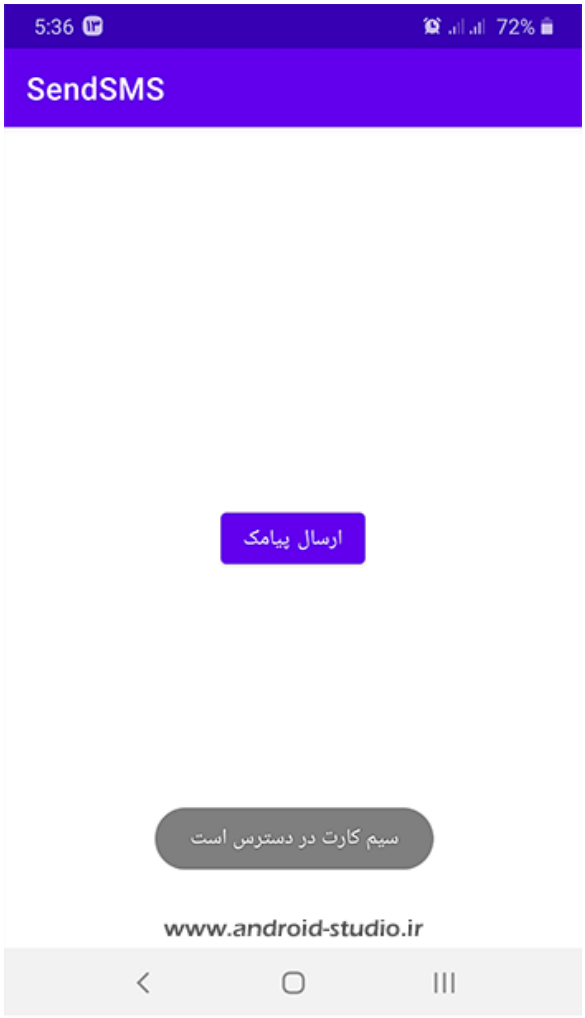
```
public void simState() {
    TelephonyManager telephonyManager = (TelephonyManager)
getSystemService(Context.TELEPHONY_SERVICE);
    int SIM_STATE = telephonyManager.getSimState();

    if (SIM_STATE == TelephonyManager.SIM_STATE_READY) {
        Toast.makeText(this, "سیم کارت در دسترس است", Toast.LENGTH_SHORT).show();
    }
    else {
        String state = "";
        switch (SIM_STATE) {
            case TelephonyManager.SIM_STATE_ABSENT:
                state = "Sim absent";
                break;
            case TelephonyManager.SIM_STATE_NETWORK_LOCKED:
                state = "Network locked";
                break;
            case TelephonyManager.SIM_STATE_PIN_REQUIRED:
                state = "Pin required";
                break;
            case TelephonyManager.SIM_STATE_PUK_REQUIRED:
                state = "PUK required";
                break;
            case TelephonyManager.SIM_STATE_UNKNOWN:
                state = "Unknown";
                break;
        }
        Toast.makeText(this, state, Toast.LENGTH_SHORT).show();
    }
}
```

در کد فوق با استفاده از متد getSimState وضعیت سیم کارت بررسی می‌شود. چنانچه سیم کارت در دسترس باشد حالت SIM_STATE_READY برگردانده می‌شود. در غیر اینصورت یکی از ۵ حالت تعریف

شده در قسمت دوم شرط برگردانده شده و پیغام مربوط به آن حالت در Toast نمایش داده خواهد شد.

برای اجرای متد ()simState کافیست آنرا در متد ()onCreate اکتیویتی فراخوانی کنیم.

پروژه را یکبار در حالتی که سیم کارت درون شیار دستگاه قرار دارد و بار دوم در حالتی که سیم کارت موجود نیست اجرا می‌کنم:



MainActivity.java

```
package ir.android_studio.sendsms;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import android.Manifest;
import android.app.Activity;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.telephony.TelephonyManager;
import android.widget.Button;
```

```
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    Button sendBtn;
    private final int SMS_REQUEST_CODE = 100;
    String SMS_SENT = "SMS_SENT";
    String SMS_DELIVERED = "SMS_DELIVERED";

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

        simState();

        sendBtn = findViewById(R.id.sms_btn);

        sendBtn.setOnClickListener(view -> {

            if (ContextCompat.checkSelfPermission(MainActivity.this,
Manifest.permission.SEND_SMS) != PackageManager.PERMISSION_GRANTED) {

                requestSendSMSpermission();

            } else {

                sendMessage();

            }

        });

    }

    private void sendMessage() {

        try {

            //Broadcast for Sent SMS
            registerReceiver(new BroadcastReceiver() {
                @Override
                public void onReceive(Context context, Intent intent) {

                    String state = "";
                    switch (getResultCode()) {
                        case Activity.RESULT_OK:
                            state = "پیامک ارسال شد";
                            break;
                        case SmsManager.RESULT_ERROR_GENERIC_FAILURE:
                            state = "یک خطای عمومی رخ داد";
                            break;
                        case SmsManager.RESULT_ERROR_NO_SERVICE:
                            state = "اپراتور در دسترس نیست";
                            break;
                        case SmsManager.RESULT_ERROR_NULL_PDU:
                            state = "در دسترس نیست PDU پروتکل";
                            break;
                        case SmsManager.RESULT_ERROR_RADIO_OFF:
```

```
                        state = "سیم کارت در دسترس نیست";
                        break;
                }
                Toast.makeText(context, state, Toast.LENGTH_SHORT).show();
            }
        }, new IntentFilter(SMS_SENT));

        //Broadcast for Delivered SMS
        registerReceiver(new BroadcastReceiver() {
            @Override
            public void onReceive(Context context, Intent intent) {
                String state = "";
                switch (getResultCode()) {
                    case Activity.RESULT_OK:
                        state = "پیامک تحویل داده شد";
                        break;
                    case Activity.RESULT_CANCELED:
                        state = "پیامک تحویل داده نشد";
                        break;
                }
                Toast.makeText(context, state, Toast.LENGTH_SHORT).show();
            }
        }, new IntentFilter(SMS_DELIVERED));

        PendingIntent sentSMS = PendingIntent.getBroadcast(this, 0, new
Intent(SMS_SENT), 0);
        PendingIntent deliverSMS = PendingIntent.getBroadcast(this, 0, new
Intent(SMS_DELIVERED), 0);

        SmsManager smsManager = SmsManager.getDefault();
        smsManager.sendTextMessage("+989158888888", null, "تست ارسال پیامک", sentSMS,
deliverSMS);

        Toast.makeText(MainActivity.this, "ارسال پیامک آغاز شد",
Toast.LENGTH_SHORT).show();

    } catch (Exception e) {

        Toast.makeText(MainActivity.this, "پیامک ارسال نشد",
Toast.LENGTH_SHORT).show();

    }

}

private void requestSendSMSpermission() {

    if (ActivityCompat.shouldShowRequestPermissionRationale(MainActivity.this,
Manifest.permission.SEND_SMS)) {

        new AlertDialog.Builder(this)
                .setTitle("درخواست مجوز")
                .setMessage("برای عملکرد صحیح برنامه باید دسترسی به ارسال پیامک
تایید شود")
                .setPositiveButton("موافقم", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int i) {

                        reqPermission();
```

```
                    }
                })
                .setNegativeButton("لغو", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int i) {

                        dialogInterface.dismiss();

                    }
                })
                .create()
                .show();

        } else {

            reqPermission();

        }

    }

    private void reqPermission() {

        ActivityCompat.requestPermissions(MainActivity.this, new String[]
{Manifest.permission.SEND_SMS}, SMS_REQUEST_CODE);

    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
@NonNull int[] grantResults) {

        if (requestCode == SMS_REQUEST_CODE) {

            if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {

                sendMessage();

            } else {

                Toast.makeText(this, "مجوز رد شد ", Toast.LENGTH_SHORT).show();

            }

        }

    }

    public void simState() {
    TelephonyManager telephonyManager = (TelephonyManager)
getSystemService(Context.TELEPHONY_SERVICE);
    int SIM_STATE = telephonyManager.getSimState();

    if (SIM_STATE == TelephonyManager.SIM_STATE_READY) {
        Toast.makeText(this, "سیم کارت در دسترس است", Toast.LENGTH_SHORT).show();
    }
    else {
```

```
        String state = "";
        switch (SIM_STATE) {
            case TelephonyManager.SIM_STATE_ABSENT:
                state = "Sim absent";
                break;
            case TelephonyManager.SIM_STATE_NETWORK_LOCKED:
                state = "Network locked";
                break;
            case TelephonyManager.SIM_STATE_PIN_REQUIRED:
                state = "Pin required";
                break;
            case TelephonyManager.SIM_STATE_PUK_REQUIRED:
                state = "PUK required";
                break;
            case TelephonyManager.SIM_STATE_UNKNOWN:
                state = "Unknown";
                break;

            }
        Toast.makeText(this, state, Toast.LENGTH_SHORT).show();
        }
    }

}
```

در این جلسه ارسال پیامک (SMS) در برنامه نویسی اندروید را به همراه نحوه‌ی دریافت گزارش ارسال و تحویل و همچنین وضعیت سیم کارت دیوایس اندرویدی بررسی کردیم. امیدوارم مفید واقع شود.

موفق و پیروز باشید.

مطالعه بیشتر:

https://developer.android.com/reference/android/telephony/SmsManager

https://developer.android.com/reference/android/app/PendingIntent

توجه: سورس پروژه درون پوشه Exercises قرار دارد