



گزارش پروژه آشنایی با شبکه های تلفن همراه
نیمسال ۴۰۲۲

عنوان پروژه: ساماریم

اعضای گروه: نیایش خانی، فرگل فریدونی

نام گروه: PacketPartners

در ابتدا به بررسی توابع **MainActivity** می پردازیم:

:onCreate

این تابع هنگامی که (Activity) ایجاد می شود، فراخوانی خواهد شد. در این تابع، پایگاه داده ایجاد (DatabaseHelper) و بررسی می شود که آیا دستگاه اجازه دسترسی به مجوزهای READ_PHONE_STATE و ACCESS_FINE_LOCATION را دارد یا خیر. اگر دسترسی داده نشده باشد، یک درخواست برای دریافت این مجوزها ارسال می شود. اگر دسترسی داده شده باشد، تابع getCellInfo() فراخوانی می شود تا اطلاعات سلولی دستگاه را دریافت و نمایش دهد.

```

33     }
34
35     # niayesh-khani
36     override fun onCreate(savedInstanceState: Bundle?) {
37         super.onCreate(savedInstanceState)
38         setContentView(R.layout.activity_main)
39
40         dbHelper = DatabaseHelper(this)
41
42         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
43             if (ContextCompat.checkSelfPermission(this, Manifest.permission.READ_PHONE_STATE) != PackageManager.PERMISSION_GRANTED ||
44                 ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
45                 requestPermissionLauncher.launch(arrayOf(Manifest.permission.READ_PHONE_STATE, Manifest.permission.ACCESS_FINE_LOCATION))
46             } else {
47                 getCellInfo()
48             }
49         } else {
50             getCellInfo()
51         }
52     }

```

:getCellInfo

این تابع اطلاعات مربوط به سلول (Cellular Information) را از TelephonyManager گرفته و در پایگاه داده ذخیره می‌کند. برای هر نوع سلولی (LTE، WCDMA، GSM)، اطلاعاتی مانند TAC، PLMN ID (یا LAC برای WCDMA)، Cell ID، قدرت سیگنال (RSSI، Ec/NO، RSCP، RSRQ، RSRP) و فناوری (LTE، WCDMA، GSM) را جمع‌آوری می‌کند و آنها را به پایگاه داده محلی ذخیره می‌کند. در نهایت، این اطلاعات به صورت متنی در TextView با id cellInfoTextView نمایش داده می‌شود.

```

53 private fun getCellInfo() {
54     val telephonyManager = getSystemService(Context.TELEPHONY_SERVICE) as TelephonyManager
55
56     if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
57         return
58     }
59
60     val cellInfoList: List<CellInfo> = telephonyManager.allCellInfo
61     val cellInfoText = StringBuilder()
62
63     val networkType = getNetworkType(telephonyManager.networkType)
64     cellInfoText.append("Network Type: $networkType\n\n")
65
66     val db = dbHelper.writableDatabase
67
68     for (cellInfo in cellInfoList) {
69         val contentValues = ContentValues()
70
71         when (cellInfo) {
72             is CellInfoLte -> {
73                 val cellIdentityLte = cellInfo.cellIdentity
74                 val cellSignalStrengthLte = cellInfo.cellSignalStrength
75
76                 val plmnId = if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.P) {
77                     cellIdentityLte.mobileNetworkOperator
78                 } else {
79                     "N/A"
80                 }

```

:requestPermissionLauncher

در این تابع از `registerForActivityResult` استفاده شده است. موظف است که پس از درخواست دسترسی به `Manifest.permission.READ_PHONE_STATE` و `Manifest.permission.ACCESS_FINE_LOCATION`، نتیجه درخواست را بررسی کند. اگر دسترسی‌ها داده شده باشند، تابع `getCellInfo()` را فراخوانی می‌کند تا اطلاعات سلولی را دریافت و نمایش دهد.

```
13 private lateinit var dbHelper: DatabaseHelper
14
15 private val requestPermissionLauncher = registerForActivityResult(
16     ActivityResultContracts.RequestMultiplePermissions()
17 ) { permissions ->
18     if (permissions[Manifest.permission.READ_PHONE_STATE] == true && permissions[Manifest.permission.ACCESS_FINE_LOCATION] == true) {
19         getCellInfo()
20     } else {
21         // Handle permission denial
22     }
23 }
24
25 niayesh-khani
```

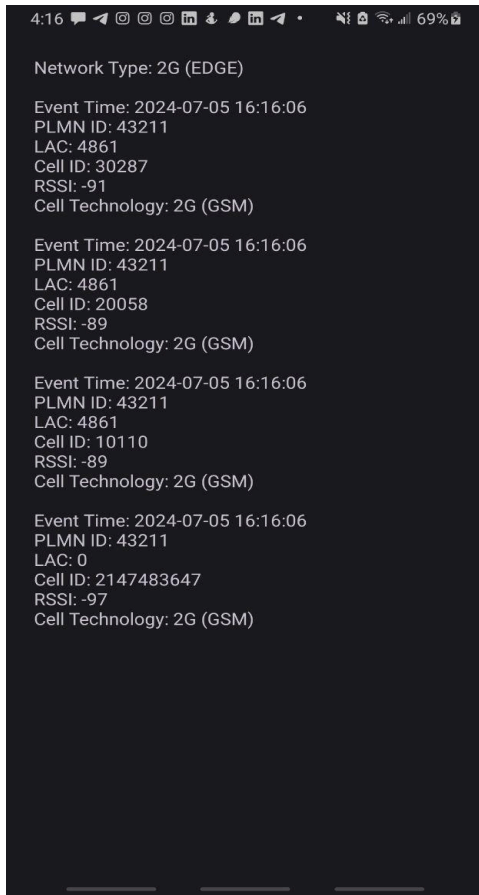
:getNetworkType

این تابع بر اساس مقدار `networkType` که توسط `TelephonyManager.networkType` ارائه می‌شود، نوع شبکه تلفنی را تشخیص می‌دهد. از `when expression` استفاده می‌کند تا بر اساس مقدار `networkType`، مقدار مناسب را برگرداند. مقادیر بازگشتی شامل "2G"، "3G (UMTS)"، "3G (HSPA)"، "3G (HSPA+)"، "5G"، "4G (LTE)"، "2G (GPRS)"، "(EDGE)" و "Unknown" خواهند بود.

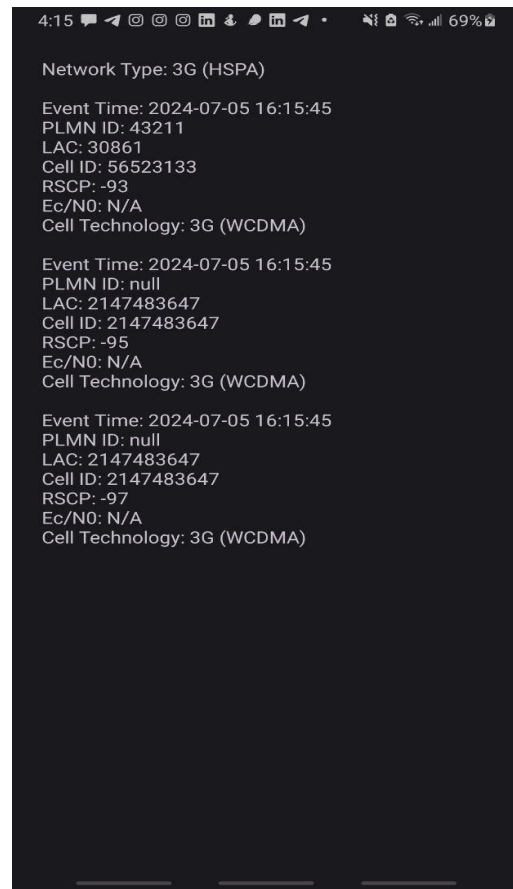
```
191
192 private fun getNetworkType(networkType: Int): String {
193     return when (networkType) {
194         TelephonyManager.NETWORK_TYPE_LTE -> "4G (LTE)"
195         TelephonyManager.NETWORK_TYPE_NR -> "5G"
196         TelephonyManager.NETWORK_TYPE_HSPAP -> "3G (HSPA+)"
197         TelephonyManager.NETWORK_TYPE_HSPA -> "3G (HSPA)"
198         TelephonyManager.NETWORK_TYPE_UMTS -> "3G (UMTS)"
199         TelephonyManager.NETWORK_TYPE_EDGE -> "2G (EDGE)"
200         TelephonyManager.NETWORK_TYPE_GPRS -> "2G (GPRS)"
201         else -> "Unknown"
202     }
203 }
204
205
```

نتایج و خروجی MainActivity در تکنولوژی های متفاوت شبکه های تلفن همراه:

2G:



3G:



4G:



MapsActivity:

:onCreate

این تابع هنگام ایجاد Activity فراخوانی می‌شود. فایل بایندینگ (ActivityMapsBinding) را تنظیم می‌کند و نمای مرتبط با آن را به عنوان محتوای اصلی Activity تنظیم و SupportMapFragment را دریافت می‌کند تا زمانی که نقشه آماده شد، اعلان دریافت خواهد کرد. در نهایت یک نمونه از FusedLocationProviderClient را برای دسترسی به مکان فعلی دستگاه ایجاد می‌کند.

```

3
7  @
1  override fun onCreate(savedInstanceState: Bundle?) {
2      super.onCreate(savedInstanceState)
3
4      binding = ActivityMapsBinding.inflate(layoutInflater)
5      setContentView(binding.root)
6
7      // Obtain the SupportMapFragment and get notified when the map is ready to be used.
8      val mapFragment = supportFragmentManager
9          .findFragmentById(R.id.map) as SupportMapFragment
10     mapFragment.getMapAsync(callback: this)
11
12     // Initialize the FusedLocationProviderClient
13     fusedLocationClient = LocationServices.getFusedLocationProviderClient(activity: this)
14 }
15
16 override fun onMapReady(googleMap: GoogleMap) {

```

:onMapReady

این تابع زمانی که نقشه آماده استفاده است، فراخوانی می‌شود. تنظیمات اولیه نقشه، مانند فعال کردن کنترل‌های زوم را انجام می‌دهد. همچنین بررسی می‌کند که آیا اجازه دسترسی به مکان دقیق کاربر داده شده است یا خیر. اگر اجازه داده شده باشد، مکان فعلی کاربر روی نقشه نمایش داده می‌شود. اگر اجازه داده نشده باشد، درخواست دسترسی به مکان ارسال می‌شود.

```

    }

    override fun onMapReady(googleMap: GoogleMap) {
        mMap = googleMap
        mMap.uiSettings.isZoomControlsEnabled = true

        // Check for location permission and get the current location
        if (ContextCompat.checkSelfPermission(context: this, Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED) {
            mMap.isMyLocationEnabled = true
            getCurrentLocation()
        } else {
            // Request location permission
            ActivityCompat.requestPermissions(activity: this, arrayOf(Manifest.permission.ACCESS_FINE_LOCATION), LOCATION_PERMISSION_REQUEST_CODE)
        }
    }
}

```

:getCurrentLocation

این تابع مکان فعلی کاربر را دریافت می‌کند. اگر اجازه دسترسی به مکان داده شده باشد، از FusedLocationProviderClient برای دریافت آخرین مکان شناخته شده استفاده می‌کند. در صورت یافتن مکان، یک pointer در مکان فعلی کاربر روی نقشه اضافه می‌کند و دوربین نقشه را به آن مکان منتقل می‌کند.

```

56     }
57
58     private fun getCurrentLocation() {
59         if (ActivityCompat.checkSelfPermission(
60             context: this,
61             Manifest.permission.ACCESS_FINE_LOCATION
62         ) != PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(
63             context: this,
64             Manifest.permission.ACCESS_COARSE_LOCATION
65         ) != PackageManager.PERMISSION_GRANTED
66         ) {
67             // TODO: Consider calling
68             //     ActivityCompat#requestPermissions
69             // here to request the missing permissions, and then overriding
70             //     public void onRequestPermissionsResult(int requestCode, String[] permissions,
71             //                                     int[] grantResults)
72             // to handle the case where the user grants the permission. See the documentation
73             // for ActivityCompat#requestPermissions for more details.
74             return
75         }
76         fusedLocationClient.lastLocation.addOnSuccessListener { location: Location? ->
77             if (location != null) {
78                 val currentLatLng = LatLng(location.latitude, location.longitude)
79                 mMap.addMarker(MarkerOptions().position(currentLatLng).title("You are here"))
80                 mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(currentLatLng, zoom: 15f))
81             }
82         }
83     }
84 }

```

onRequestPermissionsResult

:

این تابع نتیجه درخواست دسترسی به مکان را مدیریت می‌کند. بررسی می‌کند که آیا دسترسی به مکان داده شده است یا خیر. اگر اجازه داده شده باشد، مکان فعلی کاربر روی نقشه نمایش داده می‌شود. اگر اجازه داده نشده باشد، اقدامات مناسب انجام می‌شود.

```
}

@+
override fun onRequestPermissionsResult(requestCode: Int, permissions: Array<out String>, grantResults: IntArray) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults)
    if (requestCode == LOCATION_PERMISSION_REQUEST_CODE) {
        if ((grantResults.isNotEmpty() && grantResults[0] == PackageManager.PERMISSION_GRANTED)) {
            // Permission granted, get the current location
            if (ContextCompat.checkSelfPermission(context: this, Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED) {
                mMap.isMyLocationEnabled = true
                getCurrentLocation()
            }
        } else {
            // Permission denied, handle appropriately
        }
    }
}
```

با کمک این توابع میتوان نقشه‌ای را نمایش داد که مکان فعلی کاربر را نشان می‌دهد و به کاربر اجازه می‌دهیم تا با نقشه تعامل داشته باشد.

همچنین handover نیز هندل خواهد شد.

نحوه دریافت و استفاده از Api_Key:

افزودن dependency های موردنیاز:


```
implementation(libs.material)
implementation(libs.androidx.activity)
implementation(libs.androidx.constraintlayout)
implementation(libs.play.services.maps)
implementation(libs.play.services.location)
testImplementation(libs.junit)
```

دریافت API Key نقشه‌های گوگل:

در کنسول [Google Cloud Platform](#) یک پروژه جدید ایجاد کرده‌ایم. در بخش "APIs & Services" نیز API Maps SDK for Android و برخی موارد ضروری دیگر را فعال می‌کنیم. یک API Key ایجاد و کپی خواهیم کرد.

افزودن API Key به فایل [google_maps_api](#):

در مسیر `res/values` یک فایل به نام `google_maps_api.xml` ایجاد کردیم و `API Key` خود را درون آن قرار خواهیم داد:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3   <string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">AIzaSyCRQeVZ80egYM6omGKysJ06bF-LUZTpkBU</string>
4   
5 </resources>
```

اضافه کردن مجوزها در فایل `AndroidManifest`:

مجوزهای لازم برای دسترسی به اینترنت و مکان کاربر را اضافه کرده ایم:

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

ایجاد فایل `layout` برای نقشه:

در فایل `res/layout/activity_maps.xml` یک `SupportMapFragment` اضافه میکنیم:

```
<?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=".MapsActivity">

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/map"
        android:name="com.google.android.gms.maps.SupportMapFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</RelativeLayout>
```

عملکرد:

اکتیویته `MapsActivity` ایجاد شده و نمایش داده می شود.

SupportMapFragment به نقشه‌های گوگل متصل می‌شود و زمانی که نقشه آماده است، تابع onMapReady فراخوانی می‌شود.

در تابع onMapReady، اگر دسترسی به مکان داده شده باشد، مکان فعلی کاربر دریافت و نشانگر روی نقشه قرار می‌گیرد. در غیر این صورت، درخواست دسترسی به مکان ارسال می‌شود. در صورتی که دسترسی به مکان داده شود، مکان کاربر دریافت و نشانگر روی نقشه قرار می‌گیرد.

FirstFragment:

onCreateView برای ایجاد `onViewCreated` و تنظیم رویدادهای UI (برای مثال کلیک بر روی دکمه) از `FirstFragment.kt` استفاده خواهد کرد.

یکی از اهداف اصلی این `Fragment`، نمایش یک دکمه است که با فشردن آن، کاربر به `Fragment` دیگری هدایت می‌شود.

onCreateView():

این تابع برای ایجاد و برگرداندن `Fragment` فراخوانی می‌شود. از `LayoutInflater` برای ایجاد نمای `Fragment` از فایل XML استفاده می‌کند. یک شیء از `FragmentFirstBinding` را ایجاد کرده و از آن برای دسترسی به (UI) استفاده می‌کند.

```
20     private val binding get() = _binding!!
21
22     override fun onCreateView(
23         inflater: LayoutInflater, container: ViewGroup?,
24         savedInstanceState: Bundle?
25     ): View {
26
27         _binding = FragmentFirstBinding.inflate(inflater, container, false)
28         return binding.root
29
30     }
31
```

onViewCreated

():

این تابع بعد از اینکه نمای `Fragment` توسط `onCreateView` ایجاد شد، فراخوانی می‌شود. از `binding` برای دسترسی به عناصر UI استفاده و یک `Listener` برای دکمه `buttonFirst` تعریف می‌کند. هنگامی که دکمه فشرده می‌شود، `(findNavController()).navigate(R.id.action_FirstFragment_to_SecondFragment)` را فراخوانی می‌کند تا کاربر را به `Fragment` دوم هدایت کند.

```

@ niayesh-khani
override fun onCreateView(view: View, savedInstanceState: Bundle?) {
    super.onCreateView(view, savedInstanceState)

    binding.buttonFirst.setOnClickListener {
        findNavController().navigate(R.id.action_FirstFragment_to_SecondFragment)
    }
}

```

:(onDestroyView

این تابع زمانی که نمای Fragment از بین می‌رود، فراخوانی می‌شود. در اینجا `_binding` به `null` تنظیم می‌شود تا از نشت حافظه جلوگیری شود.

```

@ niayesh-khani
override fun onDestroyView() {
    super.onDestroyView()
    _binding = null
}
}

```

این بخش، یک متغیر خصوصی برای نگهداری `FragmentFirstBinding` است که به `Fragment` متصل می‌شود.

```
private var _binding: FragmentFirstBinding? = null
```

در خط ذکر شده نیز یک ویژگی (property) که به `_binding` اشاره می‌کند و دسترسی ایمن به `binding` را تضمین می‌کند.

```

@ niayesh-khani
private val binding get() = _binding!!

```

در مقایسه **SecondFragment** با **FirstFragment** میتوان گفت، هر دو **Fragment** عملکرد مشابهی دارند و از روش‌های یکسانی برای ایجاد و مدیریت نمایی (UI) استفاده می‌کنند. تفاوت اصلی آنها در **destination** و شناسه‌های منابع استفاده شده در **Listener** های **button** ها است. این تفاوت‌ها مشخص می‌کنند که کدام **Fragment** به کدام مقصد خواهد رفت.

(هر دو از **FragmentBinding** (یعنی **FragmentFirstBinding** و **FragmentSecondBinding**) برای اتصال (UI) استفاده می‌کنند. هر دو به این صورت تعریف شده‌اند که **binding** فقط بین **onCreateView** و **onDestroyView** (معتبر خواهد بود).

DatabaseHelper:

عملکرد کلی فایل **DatabaseHelper** مدیریت پایگاه داده **SQLite** می‌باشد. این کلاس با توابع **onCreate** و **onUpgrade** که از کلاس **SQLiteOpenHelper** به ارث برده شده‌اند، مدیریت بروزرسانی پایگاه داده را انجام می‌دهد. این کلاس شامل تعریف جداول و ستون‌ها برای ذخیره اطلاعات سلولی است. جدول **cellinfo** را با ستون‌های مربوط به اطلاعات سلولی ایجاد می‌کند. در صورت نیاز به بروزرسانی پایگاه داده، جدول قدیمی حذف شده و جدول جدید ایجاد می‌شود. این ساختار به ذخیره و بازیابی اطلاعات سلولی که توسط اپلیکیشن جمع‌آوری می‌شوند، کمک می‌کند.

این کلاس از **SQLiteOpenHelper** به ارث برده شده و وظیفه مدیریت پایگاه داده را دارد.

```
class DatabaseHelper(context: Context) : SQLiteOpenHelper(context, DATABASE_NAME, null, DATABASE_VERSION) {
```

companion object

:

در این بخش **constants** هایی تعریف شده‌اند که برای نام و نسخه پایگاه داده و همچنین نام جدول و ستون‌ها استفاده می‌شوند. **DATABASE_NAME** و **DATABASE_VERSION** به ترتیب نام و نسخه پایگاه داده را مشخص می‌کنند. **TABLE_NAME** و نام ستون‌ها برای ایجاد و مدیریت جدول اطلاعات سلولی استفاده می‌شوند.

```
companion object {
    const val DATABASE_NAME = "cellinfo.db"
    const val DATABASE_VERSION = 1
    const val TABLE_NAME = "cellinfo"
    const val COLUMN_ID = "id"
    const val COLUMN_EVENT_TIME = "event_time"
    const val COLUMN_PLMN_ID = "plmn_id"
    const val COLUMN_TAC = "tac"
    const val COLUMN_CELL_ID = "cell_id"
    const val COLUMN_RSRP = "rsrp"
    const val COLUMN_RSRQ = "rsrq"
    const val COLUMN_TECHNOLOGY = "technology"
}
```

:onCreate

این تابع زمانی که پایگاه داده برای اولین بار ایجاد می‌شود، فراخوانی می‌شود. در اینجا، یک رشته SQL برای ایجاد جدول cellinfo با ستون‌های تعریف شده ساخته شده و با استفاده از db.execSQL اجرا می‌شود. ستون‌ها شامل: id (به عنوان کلید اصلی)، rsrp، rsrq، cell_id، tac، plmn_id، event_time و technology هستند.

```
override fun onCreate(db: SQLiteDatabase) {
    val createTable = "CREATE TABLE $TABLE_NAME (" +
        "$COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT, " +
        "$COLUMN_EVENT_TIME TEXT, " +
        "$COLUMN_PLMN_ID TEXT, " +
        "$COLUMN_TAC INTEGER, " +
        "$COLUMN_CELL_ID INTEGER, " +
        "$COLUMN_RSRP TEXT, " +
        "$COLUMN_RSRQ TEXT, " +
        "$COLUMN_TECHNOLOGY TEXT)"
    db.execSQL(createTable)
}
```

:onUpgrade

این تابع زمانی که نسخه پایگاه داده تغییر می‌کند، فراخوانی می‌شود. ابتدا جدول موجود با استفاده از ("db.execSQL("DROP TABLE IF EXISTS \$TABLE_NAME حذف می‌شود. سپس تابع onCreate() دوباره فراخوانی می‌شود تا جدول جدید ایجاد شود.

```
override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {
    db.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")
    onCreate(db)
}
```

AndroidManifest:

در فایل **AndroidManifest.xml** تنظیمات و دسترسی‌های ضروری اپلیکیشن و فعالیت‌های اصلی آن را مشخص کرده‌ایم. این فایل به سیستم عامل اندروید کمک می‌کند تا اپلیکیشن را به درستی مدیریت و اجرا کند. همچنین یکی از موارد مهم این فایل که می‌توان به آن اشاره کرد: کلید api برای Google Maps می‌باشد که در تگ <application> قرار دارد.

```

android:fullBackupContent="@xml/backup_rules"
android:icon="@mipmap/ic_launcher"
android:label="Google Map"
android:roundIcon="@mipmap/ic_launcher_round"
android:supportRtl="true"
android:theme="@style/Theme.GoogleMap"
tools:targetApi="31">

<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="AIzaSyCRQeVZB0egYM6om6KysJ06bF-LUZTpKBU" />

<activity
    android:name=".MapsActivity"
    android:exported="true"
    android:label="MAP">

    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />

    </intent-filter>
</activity>
<activity
    android:name=".MainActivity"
    android:exported="true">

</activity>

```

در نهایت نتیجه بدین صورت خواهد بود:

