برای دریافت کمیت و کیفیت سیگنال‌های سلولی در اندروید مانند پارامترهای RSRP و RSRQ در نسل چهارم و پارامترهای RSCP و EC/No در نسل سوم از کلاس‌های `TelephonyManager` و `CellInfo` استفاده می‌کنیم. در اینجا نمونه‌ای از کد در کاتلین به همراه توضیحات ارائه شده است.

ابتدا دسترسی‌های لازم را در فایل `AndroidManifest.xml` اضافه می‌کنیم:

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

سپس کد مربوط به دریافت کیفیت و کمیت سیگنال را در فایل کاتلین به صورت زیر پیاده‌سازی می‌کنیم:

```
import android.Manifest
import android.content.Context
import android.content.pm.PackageManager
import android.os.Build
import android.telephony.*
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat
import android.os.Bundle
import android.util.Log

class MainActivity : AppCompatActivity() {
```

```
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)


        requestPermissions()
        getCellInfo(this)
    }


    private fun requestPermissions() {
        if (ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED ||
            ContextCompat.checkSelfPermission(this,
Manifest.permission.READ_PHONE_STATE) !=
PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this,
arrayOf(Manifest.permission.ACCESS_FINE_LOCATION,
Manifest.permission.READ_PHONE_STATE), 1)
        }
    }


    private fun getCellInfo(context: Context) {
        val telephonyManager =
context.getSystemService(Context.TELEPHONY_SERVICE) as TelephonyManager
```

```
    if (ActivityCompat.checkSelfPermission(context,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
        return
    }

    val cellInfoList = telephonyManager.allCellInfo

    for (cellInfo in cellInfoList) {
        when (cellInfo) {
            is CellInfoLte -> {
                val cellSignalStrengthLte = cellInfo.cellSignalStrength
                val cellIdentityLte = cellInfo.cellIdentity

                val rsrp = cellSignalStrengthLte.rsrp
                val rsrq = cellSignalStrengthLte.rsrq

                Log.d("CellInfo", "LTE: RSRP = $rsrp, RSRQ = $rsrq")
            }
            is CellInfoWcdma -> {
                val cellSignalStrengthWcdma = cellInfo.cellSignalStrength
                val cellIdentityWcdma = cellInfo.cellIdentity

                val rscp = cellSignalStrengthWcdma.dbm
```

```
                val ecNo = cellSignalStrengthWcdma.ecNo

                Log.d("CellInfo", "WCDMA: RSCP = $rscp, EC/No = $ecNo")
            }
            // Add more cases for other network types if needed
            else -> {
                Log.d("CellInfo", "Unknown network type")
            }
        }
    }
  }
}
```

دسترسی‌ها

دسترسی‌های `ACCESS_FINE_LOCATION` و `READ_PHONE_STATE` برای دریافت اطلاعات سیگنال‌های سلولی ضروری هستند.

دریافت اطلاعات سلولی

با استفاده از `TelephonyManager` و متد `allCellInfo` لیستی از اطلاعات سلولی دریافت می‌شود.

سپس با بررسی نوع `CellInfo` مثل `CellInfoLte` و `CellInfoWcdma` پارامترهای مورد نیاز مانند RSRP، RSRQ، RSCPو EC/No استخراج و در لاگ ثبت می‌شوند.

درخواست دسترسی‌ها

بررسی می‌شود که دسترسی‌های لازم قبلا به برنامه داده شده‌اند یا نه و در صورت عدم وجود دسترسی، درخواست آنها از کاربر انجام می‌شود.

برای دریافت اطلاعات فناوری سلولی در اندروید و تشخیص نوع شبکه موبایل مثلLTE ، +HSPA، HSPA، UMTS، EDGE، GPRS، GSMاز کلاس‌های `TelephonyManager` و `NetworkInfo` استفاده می‌کنیم. در اینجا مثالی با استفاده از زبان برنامه‌نویسی کاتلین ارائه می‌دهیم.

ابتدا باید دسترسی‌های لازم را در فایل `AndroidManifest.xml` اضافه کنیم:

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

سپس کد مربوط به دریافت نوع شبکه در فایل کاتلین به صورت زیر خواهد بود:

```
import android.content.Context
import android.net.ConnectivityManager
import android.telephony.TelephonyManager
import android.util.Log

fun getNetworkClass(context: Context): String {
   val telephonyManager =
context.getSystemService(Context.TELEPHONY_SERVICE) as TelephonyManager

   when (telephonyManager.networkType) {
      TelephonyManager.NETWORK_TYPE_LTE -> return "LTE"
      TelephonyManager.NETWORK_TYPE_HSPAP -> return "HSPA+"
      TelephonyManager.NETWORK_TYPE_HSPA -> return "HSPA"
```

```
        TelephonyManager.NETWORK_TYPE_UMTS -> return "UMTS"
        TelephonyManager.NETWORK_TYPE_EDGE -> return "EDGE"
        TelephonyManager.NETWORK_TYPE_GPRS -> return "GPRS"
        TelephonyManager.NETWORK_TYPE_GSM -> return "GSM"
        // Other network types can be added here
        else -> return "UNKNOWN"
    }
}
```

این تابع نوع شبکه سلولی فعلی را برمی‌گرداند. برای استفاده از این تابع و نمایش نتیجه، می‌توانیم از کد زیر استفاده کنیم:

```
fun checkNetworkType(context: Context) {
    val networkType = getNetworkClass(context)
    Log.d("NetworkType", "Current network type is: $networkType")
}
```

```
import android.Manifest
import android.content.pm.PackageManager
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat

fun requestPermissions(activity: Activity) {
   if (ContextCompat.checkSelfPermission(activity,
Manifest.permission.READ_PHONE_STATE) !=
PackageManager.PERMISSION_GRANTED) {
      ActivityCompat.requestPermissions(activity,
arrayOf(Manifest.permission.READ_PHONE_STATE), 1)
   }
}
```

در نهایت، این کد را در جایی که نیاز داریم استفاده میکنیم. مثلا در اکتیویتی اصلی برنامه:

```
class MainActivity : AppCompatActivity() {
   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_main)

      requestPermissions(this)
      checkNetworkType(this)
```

```
    }
}
```

این مثال ساده نحوه دریافت نوع شبکه سلولی فعلی را در اندروید نشان می‌دهد.

بخش مکان کاربر:

نحوه دریافت موقعیت کاربری در اندروید:

بسیاری از برنامه‌ها در اندروید از مکان‌های کاربر استفاده می‌کنند، چه برای سفارش تاکسی یا تحویل غذا و اقلام. در اینجا یک برنامه اندرویدی ساده ساخته شده است که طول و عرض جغرافیایی کاربر را برمی گرداند. هنگامی که طول و عرض جغرافیایی مشخص شد، مکان دقیق در نقشه های گوگل با استفاده از عبارت زیر قابل مشاهده است:

https://www.google.com/maps/search/?api=1&query=,

مرحله کسب مجوز:

از آنجایی که استفاده از مجوز کاربر یک موضوع مربوط به حفظ حریم خصوصی بالا است، ابتدا با درخواست از کاربر برای استفاده از موقعیت مکانی خود، اجازه دریافت میکنیم.

ACCESS_COARSE_LOCATION: دقت مکان را در یک بلوک شهری ارائه می دهد .

```
<uses-permission
android:name=”android.permission.ACCESS_COARSE_LOCATION>/”
```

ACCESS_FINE_LOCATION: مکان دقیق تری را ارائه می دهد. توصیه می شود فقط زمانی که به مکان دقیق نیاز داریم از آن استفاده کنیم.

```
<uses-permission
android:name=”android.permission.ACCESS_FINE_LOCATION</ ”
```

در صورتی که برنامه نیاز به دسترسی به موقعیت مکانی کاربر در حالی که برنامه در پس‌زمینه اجرا می‌شود، باید مجوز زیر را به همراه موارد بالا اضافه کنیم:

```
<uses-permission
android:name=”android.permission.ACCESS_BACKGROUND_LOCATION”
/ >
```

ما باید همه این مجوزها را در AndroidManifest.xml اضافه کنیم. برای دسترسی به این فایل، نمای پروژه خود را به عنوان اندروید انتخاب میکنیم و به بخش زیر میرویم :

app->manifests->AndroidManifest.xml.

پس از افزودن تمام مجوزها، فایل AndroidManifest.xml به این صورت است:

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

    <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_BACKGROUND_LOCATION"
/>
```

```
    <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.GetUserLocation">
        <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>
```

مرحله طراحی طرح

از آنجایی که برنامه نسبتاً ساده است، فقط شامل MainActivity و بنابراین یک طرح اصلی واحد است. در طرح‌بندی، یک ImageView و دو TextView اضافه میکنیم که طول و عرض جغرافیایی کاربر را نشان می‌دهند. طول و عرض جغرافیایی که نمایش داده می شود از منطق MainActivity ما بازگردانده می شود که در ادامه مورد بحث قرار خواهد گرفت. در اینجا activity_ main.xmlبه نظر می رسد:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#4caf50"
    android:gravity="center"
    android:orientation="vertical">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingBottom="120dp"
        android:src="@drawable/gfgimage" />

    <TextView
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:fontFamily="sans-serif-black"
        android:text="Latitude:" />

    <TextView
        android:id="@+id/latTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Latitude will be here! "
        android:textColor="#f5f5f5" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:fontFamily="sans-serif-black"
        android:text="Longitude:" />

    <TextView
        android:id="@+id/lonTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:text="Longitude will be here! "
        android:textColor="#f5f5f5" />

/>LinearLayout>
```

مرحله نوشتن منطق

برای کار بر روی منطق اصلی برنامه خود، نکات کلیدی زیر را دنبال می کنیم :

بررسی کنیم که آیا مجوزهایی که درخواست می کنیم فعال هستند یا خیر.

در غیر این صورت مجوزها را درخواست کنیم.

اگر مجوزها پذیرفته شد و موقعیت مکانی فعال شد، آخرین مکان کاربر را دریافت کنیم.

برای به دست آوردن آخرین مکان کاربر، از کلاس عمومی جاوا FusedLocationProviderClientاستفاده کنیم. این در واقع یک سرویس مکان یابی است که موقعیت مکانی GPS و موقعیت مکانی شبکه را برای دستیابی به تعادل بین مصرف باتری و دقت ترکیب می کند. مکان GPS برای ارائه دقت و موقعیت شبکه برای دریافت موقعیت زمانی که کاربر در داخل خانه است استفاده می شود.

در ارتباط باFusedLocationProviderClient ، کلاس عمومی LocationRequest برای دریافت آخرین مکان شناخته شده استفاده می شود. در این شیLocationRequest ، روش‌های مختلفی را تنظیم میکنیم، مانند اولویت تعیین دقیق مکان یا در چند بازه زمانی که باید درخواست مکان انجام شود.

اگر دقت بسیار بالایی لازم است، از PRIORITY_HIGH_ACCURACY به عنوان آرگومان برای متد setPriority(int) استفاده میکنیم. برای دقت سطح شهر (دقت پایین)، از PRIORITY_LOW_POWERاستفاده میکنیم.

هنگامی که شی LocationRequest آماده شد، آن را روی شی
FusedLocationProviderClientتنظیم میکنیم تا مکان نهایی را بدست آوریم.

اکنون به فایل MainActivity.kt نگاه می کنیم:

```
import android.Manifest;
import android.annotation.SuppressLint;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.location.Location;
import android.location.LocationManager;
import android.os.Bundle;
import android.os.Looper;
import android.provider.Settings;
import android.widget.TextView;
import android.widget.Toast;

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

import com.google.android.gms.location.FusedLocationProviderClient;
import com.google.android.gms.location.LocationCallback;
```

```
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationResult;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;

public class MainActivity extends AppCompatActivity {

    // initializing
    // FusedLocationProviderClient
    // object
    FusedLocationProviderClient mFusedLocationClient;

    // Initializing other items
    // from layout file
    TextView latitudeTextView, longitTextView;
    int PERMISSION_ID = 44;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);

        latitudeTextView = findViewById(R.id.latTextView);
        longitTextView = findViewById(R.id.lonTextView);

        mFusedLocationClient =
LocationServices.getFusedLocationProviderClient(this);

        // method to get the location
        getLastLocation();
    }

    @SuppressLint("MissingPermission")
    private void getLastLocation() {
        // check if permissions are given
        if (checkPermissions()) {

            // check if location is enabled
            if (isLocationEnabled()) {

                // getting last
```

```
                    // location from
                    // FusedLocationClient
                    // object

        mFusedLocationClient.getLastLocation().addOnCompleteListener(
new OnCompleteListener<Location>() {
                        @Override
                        public void onComplete(@NonNull
Task<Location> task) {
                            Location location = task.getResult();
                            if (location == null) {
                                requestNewLocationData();
                            } else {

        latitudeTextView.setText(location.getLatitude() + "");

        longitTextView.setText(location.getLongitude() + "");
                            }
                        }
                    });
                } else {
                    Toast.makeText(this, "Please turn on" + " your
location...", Toast.LENGTH_LONG).show();
```

```
                        Intent intent = new
Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);
                        startActivity(intent);
                    }
                } else {
                    // if permissions aren't available,
                    // request for permissions
                    requestPermissions();
                }
            }

            @SuppressLint("MissingPermission")
            private void requestNewLocationData() {

                // Initializing LocationRequest
                // object with appropriate methods
                LocationRequest mLocationRequest = new
LocationRequest();

            mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_A
CCURACY);
                mLocationRequest.setInterval(5);
                mLocationRequest.setFastestInterval(0);
```

```
            mLocationRequest.setNumUpdates(1);


            // setting LocationRequest
            // on FusedLocationClient
            mFusedLocationClient =
LocationServices.getFusedLocationProviderClient(this);

        mFusedLocationClient.requestLocationUpdates(mLocationRequest
, mLocationCallback, Looper.myLooper());
        }


        private LocationCallback mLocationCallback = new
LocationCallback() {


            @Override
            public void onLocationResult(LocationResult locationResult) {
                Location mLastLocation =
locationResult.getLastLocation();
                latitudeTextView.setText("Latitude: " +
mLastLocation.getLatitude() + "");
                longitTextView.setText("Longitude: " +
mLastLocation.getLongitude() + "");
            }
        };
```

```
    // method to check for permissions
    private boolean checkPermissions() {
        return ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) ==
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED;


        // If we want background location
        // on Android 10.0 and higher,
        // use:
        // ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_BACKGROUND_LOCATION) ==
PackageManager.PERMISSION_GRANTED
    }


    // method to request for permissions
    private void requestPermissions() {
        ActivityCompat.requestPermissions(this, new String[]{
                Manifest.permission.ACCESS_COARSE_LOCATION,
                Manifest.permission.ACCESS_FINE_LOCATION},
PERMISSION_ID);
```

```
    }


    // method to check
    // if location is enabled
    private boolean isLocationEnabled() {
        LocationManager locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
        return
locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER)
||
locationManager.isProviderEnabled(LocationManager.NETWORK_PROVI
DER);
    }


    // If everything is alright then
    @Override
    public void
    onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode,
permissions, grantResults);


        if (requestCode == PERMISSION_ID) {
```

```
                    if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                        getLastLocation();
                    }
                }
            }

            @Override
            public void onResume() {
                super.onResume();
                if (checkPermissions()) {
                    getLastLocation();
                }
            }
}
```