برای انجام پروژه باید مورد زیر را بدانیم:

چگونه با استفاده از Kotlin مکان GPS فعلی را به صورت برنامه ای در اندروید دریافت کنیم؟

این مثال نشان می دهد که چگونه می توان مکان GPS فعلی را به صورت برنامه ریزی شده در اندروید با استفاده از Kotlin دریافت کرد.

مرحله 1 - یک پروژه جدید در Android Studio ایجاد میکنیم، به File ⇒ New Project میرویم و تمام جزئیات مورد نیاز را برای ایجاد یک پروژه جدید پر میکنیم.

مرحله 2 - کد زیر را به res/layout/activity_main.xml 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"
  android:padding="4dp"
  tools:context=".MainActivity">
  <TextView
    android:id="@+id/text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="70dp"
    android:background="#008080"
    android:padding="5dp"
    android:text="TutorialsPoint"
    android:textColor="#fff"
```

```
    android:textSize="24sp"
    android:textStyle="bold" />
  <TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"
    android:text="Current GPS Location"
    android:textColor="@color/colorPrimary"
    android:textSize="24sp"
    android:textStyle="bold" />
<Button
  android:id="@+id/getLocation"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_below="@+id/text"
  android:layout_centerInParent="true"
  android:layout_marginTop="40dp"
  android:text="Get location" />
/>RelativeLayout<
```

مرحله 3 - کد زیر را به src/MainActivity.kt اضافه میکنیم.

```
import android.Manifest
import android.content.Context
import android.content.pm.PackageManager
import android.location.Location
import android.location.LocationListener
import android.location.LocationManager
import android.os.Bundle
import android.widget.Button
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat
class MainActivity : AppCompatActivity(), LocationListener {
  private lateinit var locationManager: LocationManager
  private lateinit var tvGpsLocation: TextView
  private val locationPermissionCode = 2
  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    title = "KotlinApp"
    val button: Button = findViewById(R.id.getLocation)
    button.setOnClickListener {
      getLocation()
```

```
    }
  }
  private fun getLocation() {
    locationManager = getSystemService(Context.LOCATION_SERVICE) as
LocationManager
    if ((ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED)) {
      ActivityCompat.requestPermissions(this,
arrayOf(Manifest.permission.ACCESS_FINE_LOCATION),
locationPermissionCode)
    }
    locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
5000, 5f, this)
  }
  override fun onLocationChanged(location: Location) {
    tvGpsLocation = findViewById(R.id.textView)
    tvGpsLocation.text = "Latitude: " + location.latitude + " , Longitude: " +
location.longitude
  }
  override fun onRequestPermissionsResult(requestCode: Int, permissions:
Array<out String>, grantResults: IntArray) {
    if (requestCode == locationPermissionCode) {
      if (grantResults.isNotEmpty() && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
        Toast.makeText(this, "Permission Granted",
Toast.LENGTH_SHORT).show()
```

```
    }
    else {
      Toast.makeText(this, "Permission Denied", Toast.LENGTH_SHORT).show()
    }
  }
}
```

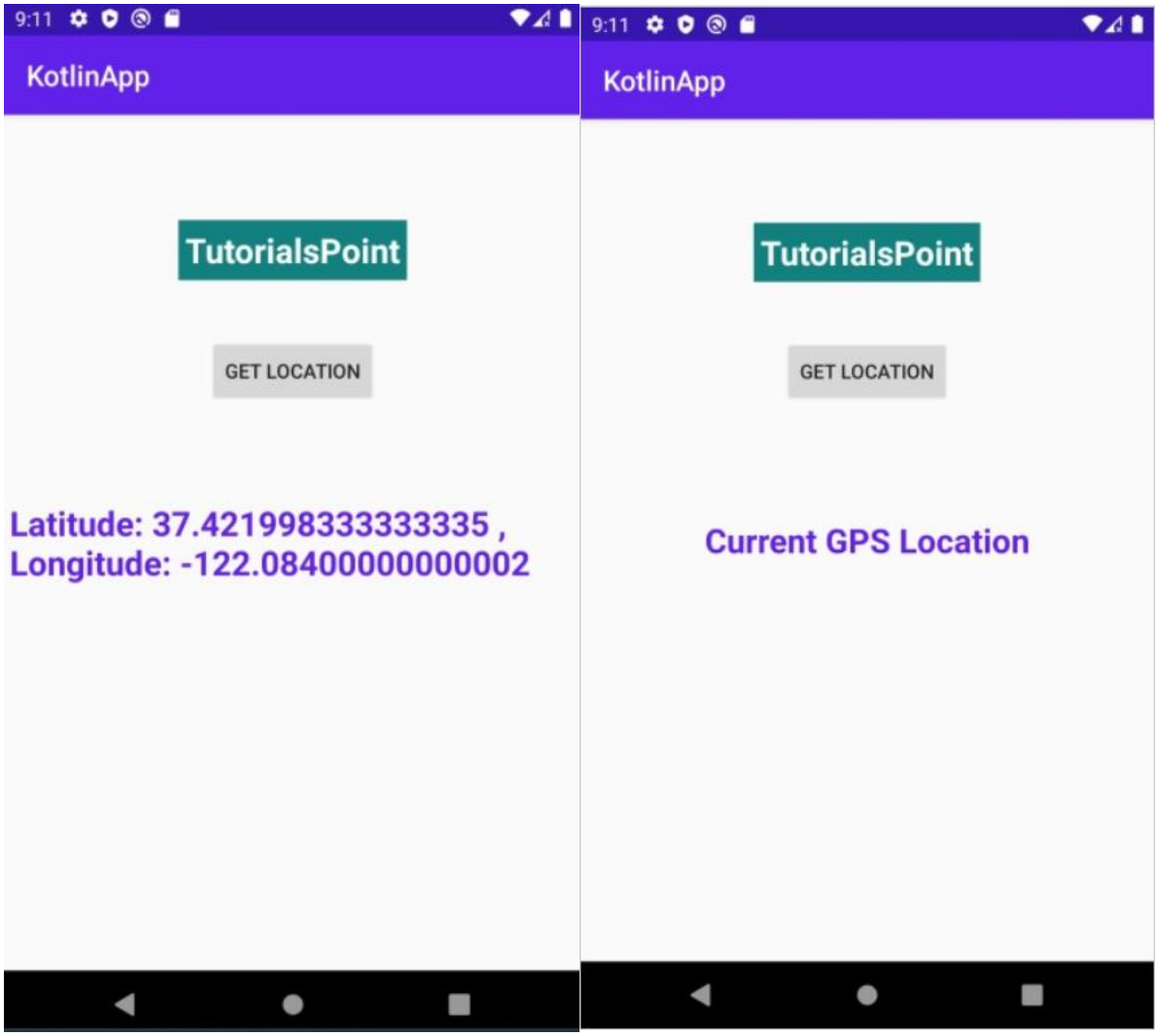
{

مرحله 4 - کد زیر را به androidManifest.xml اضافه میکنیم.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.q11">
<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_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/AppTheme">
    <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<
```

حال تصاویر اجرای برنامه به صورت زیر است که طول و عرض جغرافیایی را نشان میدهد:



توضیح پروژه:

میتوان به صورت زیر عمل کرد:

برای نوشتن برنامه‌ای در زبان Kotlin که پارامترهای مختلف شبکه را اندازه‌گیری و در یک پایگاه داده SQLiteذخیره کند، مراحل زیر را دنبال می‌کنیم:

مراحل کلی پروژه:

جمع‌آوری داده‌های شبکه:

مکان جغرافیایی کاربر (عرض و طول جغرافیایی)

زمان ثبت رخداد

فناوری سلولی که گوشی بر روی آن قرار دارد مثلاGSM ، UMTS، LTE، 5 Gو...

شناسه‌های مکانی سلولPLMN-Id ، LAC، RAC، TACو...

کیفیت سیگنالRSRP ، RSRQ، RSSI، RSCPو...

ذخیره‌سازی داده‌ها در یک پایگاه داده:SQLite

ایجاد یک پایگاه دادهSQLite

ایجاد جداول لازم برای ذخیره داده‌ها

درج داده‌ها در جداول

در Kotlinمعمولاً کدهای مختلف در فایل‌های متفاوت قرار می‌گیرند. برای این برنامه نیز، کدهای مربوط به موارد مختلف در فایل‌های مختلف قرار می‌گیرند. این فایل‌ها عبارتند از:

فایل `MainActivity.kt` این فایل شامل کد اصلی برنامه است. در اینجا، کدهای مربوط به دریافت موقعیت جغرافیایی و اضافه کردن دسترسی‌های لازم در`AndroidManifest.xml` ، و همچنین فراخوانی توابع مربوطه برای ذخیره‌سازی در پایگاه داده وجود دارد.

فایل `CellInfoHelper.kt` این فایل شامل کد مربوط به اطلاعات شبکه سلولی است. کلاس `CellInfoHelper`در اینجا تعریف شده است که وظیفه دریافت اطلاعات شبکه سلولی را دارد.

`DatabaseHelper.kt`این فایل شامل کد مربوط به مدیریت پایگاه داده SQLite است. کلاس `DatabaseHelper`در اینجا تعریف شده است که وظیفه ایجاد و به‌روزرسانی پایگاه داده و همچنین اجرای دستورات SQL بر روی آن را بر عهده دارد.

`DataManager.kt`این فایل شامل کد مربوط به مدیریت داده‌ها است. کلاس `DataManager` در اینجا تعریف شده است که وظیفه ذخیره داده‌ها در پایگاه داده را دارد.

در کل، هر فایل در این پروژه نقش خاص خود را در فرآیند انجام کار دارد.

برای نوشتن برنامه‌ای به زبان کاتلین که پارامترهای مربوط به مکان و شبکه سلولی کاربر را اندازه‌گیری کند و در پایگاه داده ذخیره کند، نیاز است ازAPI های اندروید استفاده کنیم. این برنامه شامل دسترسی به موقعیت جغرافیایی و اطلاعات شبکه سلولی است. همچنین باید از یک پایگاه داده محلی مثل SQLite برای ذخیره‌سازی داده‌ها استفاده کنیم. در اینجا به صورت مرحله‌ای نحوه انجام این کار توضیح داده می‌شود:

تنظیم مجوزهای لازم در`AndroidManifest.xml`

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.cellinfotracker">

    <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_NETWORK_STATE"/>

    <application
        android:allowBackup="true"
        android:label="@string/app_name"
        android:icon="@mipmap/ic_launcher"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
```

```
    android:theme="@style/AppTheme">
    <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>
```

دریافت اطلاعات موقعیت جغرافیایی

برای دریافت اطلاعات موقعیت جغرافیایی از `FusedLocationProviderClient` استفاده می‌کنیم:

```
import android.Manifest
import android.content.pm.PackageManager
import android.location.Location
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import androidx.core.app.ActivityCompat
import com.google.android.gms.location.FusedLocationProviderClient
import com.google.android.gms.location.LocationServices
import com.google.android.gms.tasks.Task
```

```
class MainActivity : AppCompatActivity() {

    private lateinit var fusedLocationClient: FusedLocationProviderClient

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

        fusedLocationClient =
LocationServices.getFusedLocationProviderClient(this)

        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(
                this, Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this,
arrayOf(Manifest.permission.ACCESS_FINE_LOCATION,
Manifest.permission.ACCESS_COARSE_LOCATION), 1)
            return
        }

        fusedLocationClient.lastLocation.addOnSuccessListener { location :
Location? ->
```

```
            location?.let {
                // ذخیره موقعیت در پایگاه داده
                saveLocationToDatabase(location)
            }
        }
    }

    private fun saveLocationToDatabase(location: Location) {
        // کد برای ذخیره مکان در پایگاه داده
    }
}
```

دریافت اطلاعات شبکه سلولی

برای دریافت اطلاعات شبکه سلولی باید از `TelephonyManager` استفاده کنیم:

```
import android.content.Context
import android.telephony.TelephonyManager
import android.telephony.CellInfo
import android.telephony.CellSignalStrength

class CellInfoHelper(private val context: Context) {
```

```
    fun getCellInfo(): List<CellInfo> {
        val telephonyManager =
context.getSystemService(Context.TELEPHONY_SERVICE) as TelephonyManager
        return telephonyManager.allCellInfo
    }

    fun getCellSignalStrength(cellInfo: CellInfo): CellSignalStrength {
        // استخراج قدرت سیگنال از CellInfo
    }
}
```

ذخیره داده‌ها درSQLite

برای ذخیره داده‌ها در پایگاه داده SQLite از کلاس `SQLiteOpenHelper` استفاده می‌کنیم:

```
import android.content.Context
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper

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

```
    override fun onCreate(db: SQLiteDatabase) {
        val createTable = "CREATE TABLE cell_info (" +
                "id INTEGER PRIMARY KEY AUTOINCREMENT," +
                "latitude REAL," +
                "longitude REAL," +
                "timestamp TEXT," +
                "network_type TEXT," +
                "signal_strength INTEGER," +
                "cell_id TEXT," +
                "plmn_id TEXT," +
                "lac INTEGER," +
                "tac INTEGER," +
                "rnc INTEGER)"
        db.execSQL(createTable)
    }

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

    companion object {
        private const val DATABASE_NAME = "cell_info.db"
```

```
        private const val DATABASE_VERSION = 1
    }
}
```

ذخیره اطلاعات در پایگاه داده

```
import android.content.ContentValues

class DataManager(private val context: Context) {

    private val dbHelper = DatabaseHelper(context)

    fun saveData(location: Location, networkType: String, signalStrength: Int,
cellId: String, plmnId: String, lac: Int, tac: Int, rnc: Int) {
        val db = dbHelper.writableDatabase

        val values = ContentValues().apply {
            put("latitude", location.latitude)
            put("longitude", location.longitude)
            put("timestamp", System.currentTimeMillis().toString())
            put("network_type", networkType)
            put("signal_strength", signalStrength)
            put("cell_id", cellId)
```

```
            put("plmn_id", plmnId)
            put("lac", lac)
            put("tac", tac)
            put("rnc", rnc)
        }

        db.insert("cell_info", null, values)
    }
}
```

جمع‌بندی

در این برنامه، ابتدا مجوزهای لازم در فایل `AndroidManifest.xml` تعریف شده است. سپس در `MainActivity` اطلاعات مکان کاربر و در کلاس `CellInfoHelper` اطلاعات شبکه سلولی دریافت می‌شود. در نهایت داده‌های بدست آمده در پایگاه داده SQLite توسط `DatabaseHelper` ذخیره می‌شود.

از دیتابیس Room هم میتوانیم استفاده کنیم.

برای تکمیل پروژه و تمامی موارد به صورت زیر عمل میکنیم:

برای پیاده‌سازی برنامه‌ای که پارامترهای مشخص شده را اندازه‌گیری و در یک پایگاه داده محلی ذخیره کند، مراحل زیر را دنبال میکنیم.

افزودن مجوزها به`AndroidManifest.xml`

در فایل`AndroidManifest.xml` مجوزهای مورد نیاز را اضافه میکنیم:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.measurementapp">

    <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.INTERNET"/>

    <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.MeasurementApp">
        <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>
```

افزودن وابستگی‌ها به فایل `build.gradle` ماژول app :

در فایل `build.gradle` (ماژول اپلیکیشن)، وابستگی‌های لازم را اضافه میکنیم:

```
dependencies {
    implementation "androidx.room:room-runtime:2.3.0"
    kapt "androidx.room:room-compiler:2.3.0"
    implementation "com.google.android.gms:play-services-location:18.0.0"
    implementation "androidx.core:core-ktx:1.6.0"
    implementation "androidx.appcompat:appcompat:1.3.1"
    implementation "com.google.android.material:material:1.4.0"
    implementation "androidx.constraintlayout:constraintlayout:2.1.0"
}
```

تنظیمات KAPT کامپایلر برای Annotations در فایل `build.gradle` ماژول app :

```
apply plugin: 'kotlin-kapt'
```

ایجاد کلاس Entity برای ذخیره اطلاعات

یک فایل جدید به نام `Measurement.kt` ایجاد میکنیم:

```
package com.example.measurementapp

import androidx.room.Entity
import androidx.room.PrimaryKey

@Entity
data class Measurement(
    @PrimaryKey(autoGenerate = true) val id: Int = 0,
    val latitude: Double,
    val longitude: Double,
    val timestamp: Long,
    val networkType: String,
    val cellId: Int,
    val tac: Int,
    val rac: Int,
    val lac: Int,
```

```
    val plmnId: String,
    val signalStrength: Int,
    val rsrq: Int,
    val rsrp: Int,
    val rscp: Int,
    val ecno: Int
)
```

ایجاد DAO برای دسترسی به داده‌ها

یک فایل جدید به نام `MeasurementDao.kt` ایجاد میکنیم:

```
package com.example.measurementapp

import androidx.room.Dao
import androidx.room.Insert
import androidx.room.Query

@Dao
interface MeasurementDao {
    @Insert
    fun insert(measurement: Measurement)

    @Query("SELECT * FROM Measurement")
    fun getAll(): List<Measurement>
```

```
}
```

ایجاد دیتابیس

یک فایل جدید به نام `AppDatabase.kt` ایجاد میکنیم:

```
package com.example.measurementapp

import androidx.room.Database
import androidx.room.RoomDatabase

@Database(entities = [Measurement::class], version = 1)
abstract class AppDatabase : RoomDatabase() {
   abstract fun measurementDao(): MeasurementDao
}
```

تنظیم مکان و نوع شبکه و ذخیره‌سازی داده‌ها

در فایل`MainActivity.kt` ، تغییرات زیر را اعمال میکنیم:

```
package com.example.measurementapp

import android.Manifest
import android.content.pm.PackageManager
import android.location.Location
import android.os.Bundle
```

```
import android.telephony.*
import android.util.Log
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat
import androidx.room.Room
import com.google.android.gms.location.FusedLocationProviderClient
import com.google.android.gms.location.LocationServices

class MainActivity : AppCompatActivity() {

    private lateinit var fusedLocationClient: FusedLocationProviderClient
    private lateinit var db: AppDatabase

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

        fusedLocationClient = LocationServices.getFusedLocationProviderClient(this)
        db = Room.databaseBuilder(
            applicationContext,
            AppDatabase::class.java, "measurement-database"
        ).build()
```

```
        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)
        } else {
            getLastLocationAndSave()
        }
    }


    override fun onRequestPermissionsResult(requestCode: Int, permissions:
Array<String>, grantResults: IntArray) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults)
        if (requestCode == 1 && grantResults.isNotEmpty() && grantResults[0] ==
PackageManager.PERMISSION_GRANTED && grantResults[1] ==
PackageManager.PERMISSION_GRANTED) {
            getLastLocationAndSave()
        }
    }


    private fun getLastLocationAndSave() {
        fusedLocationClient.lastLocation
            .addOnSuccessListener { location: Location? ->
```

```
            location?.let {
                val latitude = it.latitude
                val longitude = it.longitude
                val timestamp = System.currentTimeMillis()
                val networkInfo = getNetworkInfo()

                networkInfo?.let { info ->
                    saveMeasurement(latitude, longitude, timestamp, info)
                }
            }
        }
    }

    private fun getNetworkInfo(): NetworkInfo? {
        val telephonyManager = getSystemService(TELEPHONY_SERVICE) as
TelephonyManager
        val allCellInfo = telephonyManager.allCellInfo

        if (allCellInfo.isNotEmpty()) {
            val cellInfo = allCellInfo[0]

            return when (cellInfo) {
                is CellInfoLte -> {
                    val cellIdentity = cellInfo.cellIdentity
```

```
    val cellSignalStrength = cellInfo.cellSignalStrength
    NetworkInfo(
        networkType = "LTE",
        cellId = cellIdentity.ci,
        tac = cellIdentity.tac,
        rac = -1,
        lac = -1,
        plmnId = "${cellIdentity.mcc}${cellIdentity.mnc}",
        signalStrength = cellSignalStrength.dbm,
        rsrq = cellSignalStrength.rsrq,
        rsrp = cellSignalStrength.rsrp,
        rscp = -1,
        ecno = -1
    )
}
is CellInfoWcdma -> {
    val cellIdentity = cellInfo.cellIdentity
    val cellSignalStrength = cellInfo.cellSignalStrength
    NetworkInfo(
        networkType = "UMTS",
        cellId = cellIdentity.cid,
        tac = -1,
        rac = -1,
        lac = cellIdentity.lac,
```

```
                plmnId = "${cellIdentity.mcc}${cellIdentity.mnc}",
                signalStrength = cellSignalStrength.dbm,
                rsrq = -1,
                rsrp = -1,
                rscp = cellSignalStrength.dbm,
                ecno = cellSignalStrength.ecNo
            )
        }
        else -> null
      }
    }
    return null
  }


  private fun saveMeasurement(latitude: Double, longitude: Double, timestamp:
Long, networkInfo: NetworkInfo) {
    val measurement = Measurement(
      latitude = latitude,
      longitude = longitude,
      timestamp = timestamp,
      networkType = networkInfo.networkType,
      cellId = networkInfo.cellId,
      tac = networkInfo.tac,
      rac = networkInfo.rac,
      lac = networkInfo.lac,
```

```
            plmnId = networkInfo.plmnId,
            signalStrength = networkInfo.signalStrength,
            rsrq = networkInfo.rsrq,
            rsrp = networkInfo.rsrp,
            rscp = networkInfo.rscp,
            ecno = networkInfo.ecno
        )

        Thread {
            db.measurementDao().insert(measurement)
        }.start()
    }
}

data class NetworkInfo(
    val networkType: String,
    val cellId: Int,
    val tac: Int,
    val rac: Int,
    val lac: Int,
    val plmnId: String,
    val signalStrength: Int,
    val rsrq: Int,
    val rsrp: Int,
```

```
    val rscp: Int,
    val ecno: Int
)
```

توضیحات کد:

مجوزها: مجوزهای لازم برای دسترسی به مکان و اطلاعات شبکه اضافه کردیم.

وابستگی‌ها: کتابخانه‌های لازم برای Room و موقعیت مکانی اضافه کردیم.

Entity -کلاس `Measurement` نمایانگر ساختار جدول در پایگاه داده است.

DAO -این کلاس شامل متدهایی برای دسترسی به داده‌ها در پایگاه داده است.

Database -این کلاس دیتابیس را ایجاد و DAO را ارائه می‌دهد.

MainActivity -شامل منطق برای دریافت مکان و اطلاعات شبکه و ذخیره آن‌ها در پایگاه داده است.

دریافت مجوزها: درخواست مجوزهای لازم از کاربر.

دریافت مکان: استفاده از `FusedLocationProviderClient` برای دریافت آخرین مکان.

دریافت اطلاعات شبکه: استفاده از `TelephonyManager` برای دریافت اطلاعات شبکه و سیگنال.

ذخیره‌سازی: ذخیره داده‌ها در پایگاه داده با استفاده ازRoom .

نحوه اجرا کردن پروژه:

اتصال دستگاه یا شبیه‌ساز اندروید: باید مطمئن شویم که دستگاه یا شبیه‌ساز اندروید ما به درستی پیکربندی شده باشد.

کامپایل و اجرا :

- از منویRun ، گزینه 'Run 'app را انتخاب میکنیم.
- یا از میانبر Shift + F10 استفاده میکنیم.

این مراحل ما را قادر می‌سازد تا اطلاعات موقعیت مکانی، زمان و نوع شبکه کاربر را در پایگاه داده محلی ذخیره کنیم.

گزارش نهایی:

برای پوشش دادن تمامی موارد خواسته شده شامل اطلاعات مکان، زمان، نوع شبکه، شناسه سلول و پارامترهای سیگنال، برنامه باید به طور کامل این داده‌ها را جمع‌آوری و ذخیره کند. در زیر نحوه انجام این کار با استفاده از زبان کاتلین و Room database در اندروید توضیح داده شده است:

افزودن مجوزها به AndroidManifest.xml

در فایل `AndroidManifest.xml` ، مجوزهای مورد نیاز را اضافه میکنیم:

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

افزودن وابستگی‌ها به فایل `build.gradle` ماژول app :

در فایل `build.gradle` (ماژول اپلیکیشن)، وابستگی‌های لازم را اضافه میکنیم:

```
dependencies {
    implementation "androidx.room:room-runtime:2.3.0"
    kapt "androidx.room:room-compiler:2.3.0"
    implementation "com.google.android.gms:play-services-location:18.0.0"
    implementation "androidx.core:core-ktx:1.6.0"
    implementation "androidx.appcompat:appcompat:1.3.1"
    implementation "com.google.android.material:material:1.4.0"
    implementation "androidx.constraintlayout:constraintlayout:2.1.0"
}
apply plugin: 'kotlin-kapt'
```

ایجاد کلاس Entity برای ذخیره اطلاعات

یک فایل جدید به نام `Measurement.kt` ایجاد میکنیم:

```
package com.example.networkinfo

import androidx.room.Entity
import androidx.room.PrimaryKey
```

```
@Entity
data class Measurement(
    @PrimaryKey(autoGenerate = true) val id: Int = 0,
    val latitude: Double,
    val longitude: Double,
    val timestamp: Long,
    val networkType: String,
    val cellId: String,
    val signalStrength: Int,
    val signalQuality: Int
)
```

ایجاد DAO برای دسترسی به داده‌ها

یک فایل جدید به نام `MeasurementDao.kt` ایجاد میکنیم:

```
package com.example.networkinfo

import androidx.room.Dao
import androidx.room.Insert
```

```
import androidx.room.Query

@Dao
interface MeasurementDao {
    @Insert
    fun insert(measurement: Measurement)

    @Query("SELECT * FROM Measurement")
    fun getAll(): List<Measurement>
}
```

ایجاد دیتابیس

یک فایل جدید به نام `AppDatabase.kt` ایجاد میکنیم:

```
package com.example.networkinfo

import androidx.room.Database
import androidx.room.RoomDatabase

@Database(entities = [Measurement::class], version = 1)
```

```
abstract class AppDatabase : RoomDatabase() {
   abstract fun measurementDao(): MeasurementDao
}
```

تنظیم مکان، نوع شبکه و ذخیره‌سازی داده‌ها

در فایل`MainActivity.kt` ، تغییرات زیر را اعمال کنید:

```
package com.example.networkinfo

import android.Manifest
import android.content.Context
import android.content.pm.PackageManager
import android.location.Location
import android.os.Bundle
import android.telephony.CellInfo
import android.telephony.TelephonyManager
import android.telephony.SignalStrength
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat
```

```
import androidx.room.Room
import com.google.android.gms.location.FusedLocationProviderClient
import com.google.android.gms.location.LocationServices
import com.google.android.gms.tasks.OnSuccessListener

class MainActivity : AppCompatActivity() {

    private lateinit var fusedLocationClient: FusedLocationProviderClient
    private lateinit var db: AppDatabase
    private lateinit var telephonyManager: TelephonyManager

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

        fusedLocationClient =
LocationServices.getFusedLocationProviderClient(this)
        db = Room.databaseBuilder(
            applicationContext,
            AppDatabase::class.java, "measurement-database"
        ).build()
```

```
        telephonyManager =
getSystemService(Context.TELEPHONY_SERVICE) as TelephonyManager


        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)
        } else {
            getLastLocationAndSave()
        }
    }


    override fun onRequestPermissionsResult(requestCode: Int,
permissions: Array<String>, grantResults: IntArray) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults)
        if (requestCode == 1 && grantResults.isNotEmpty() &&
grantResults[0] == PackageManager.PERMISSION_GRANTED &&
grantResults[1] == PackageManager.PERMISSION_GRANTED) {
            getLastLocationAndSave()
```

```
        }
    }

    private fun getLastLocationAndSave() {
        fusedLocationClient.lastLocation
            .addOnSuccessListener { location: Location? ->
                location?.let {
                    val latitude = it.latitude
                    val longitude = it.longitude
                    val timestamp = System.currentTimeMillis()
                    val networkType = getNetworkType()
                    val cellId = getCellId()
                    val signalStrength = getSignalStrength()
                    val signalQuality = getSignalQuality()

                    saveMeasurement(latitude, longitude, timestamp,
networkType, cellId, signalStrength, signalQuality)
                }
            }
    }

    private fun getNetworkType(): String {
```

```
    return when (telephonyManager.networkType) {
        TelephonyManager.NETWORK_TYPE_LTE -> "LTE"
        TelephonyManager.NETWORK_TYPE_HSPAP -> "HSPA+"
        TelephonyManager.NETWORK_TYPE_HSPA -> "HSPA"
        TelephonyManager.NETWORK_TYPE_UMTS -> "UMTS"
        TelephonyManager.NETWORK_TYPE_EDGE -> "EDGE"
        TelephonyManager.NETWORK_TYPE_GPRS -> "GPRS"
        TelephonyManager.NETWORK_TYPE_GSM -> "GSM"
        else -> "Unknown"
    }
}

private fun getCellId(): String {
    val cellInfoList = telephonyManager.allCellInfo
    var cellId = "Unknown"
    if (cellInfoList.isNotEmpty()) {
        val cellInfo = cellInfoList[0]
        cellId = when (cellInfo) {
            is CellInfoLte -> cellInfo.cellIdentity.ci.toString()
            is CellInfoWcdma -> cellInfo.cellIdentity.cid.toString()
            is CellInfoGsm -> cellInfo.cellIdentity.cid.toString()
            else -> "Unknown"
```

```
            }
        }
        return cellId
    }

    private fun getSignalStrength(): Int {
        val signalStrength = telephonyManager.signalStrength
        return signalStrength?.level ?: 0
    }

    private fun getSignalQuality(): Int {
        val signalStrength = telephonyManager.signalStrength
        // Assuming that signal quality is represented by the level of the signal strength
        return signalStrength?.level ?: 0
    }

    private fun saveMeasurement(latitude: Double, longitude: Double, timestamp: Long, networkType: String, cellId: String, signalStrength: Int, signalQuality: Int) {
        val measurement = Measurement(
            latitude = latitude,
```

```
            longitude = longitude,
            timestamp = timestamp,
            networkType = networkType,
            cellId = cellId,
            signalStrength = signalStrength,
            signalQuality = signalQuality
        )

        Thread {
            db.measurementDao().insert(measurement)
        }.start()
    }
}
```

نحوه اجرا کردن پروژه

باید مطمئن شویم که دستگاه یا شبیه‌ساز اندروید ما به درستی پیکربندی شده باشد.

پروژه را کامپایل و اجرا میکنیم:

- از منوی`Run` ، گزینه `Run 'app'` را انتخاب میکنیم.
- یا از میانبر `Shift + F10` استفاده میکنیم.

این کد تمامی موارد خواسته شده شامل اطلاعات مکان، زمان، نوع شبکه، شناسه سلول و پارامترهای سیگنال را جمع‌آوری و در یک پایگاه داده محلی ذخیره می‌کند.