پروژه پایانی ساماریوم

درس موبایل

مبینا اسمعیل پور- 99441029

زهرا محمود زاده -97522022

در این پروژه، هدف ما جمع‌آوری اطلاعات سیگنال موبایل کاربران و ذخیره آن‌ها در یک پایگاه داده SQLite با استفاده از کتابخانه Room بود. این اطلاعات شامل مکان جغرافیایی، نوع شبکه، و پارامترهای سیگنال مانند RSRP و RSRQ می‌باشد. در ادامه مراحل این پروژه را به صورت مرحله به مرحله توضیح می‌دهیم.

مرحله 1: تنظیمات اولیه پروژه

1. ایجاد پروژه جدید:

- پروژه جدیدی با نام MobileNetworkProject در Android Studio ایجاد کردیم.

2. تنظیم فایل‌های ساخت پروژه:

- وابستگی‌های مورد نیاز را در فایل build.gradle اضافه کردیم. این وابستگی‌ها شامل کتابخانه‌های Room, Lifecycle, و Location Services بودند.

```
dependencies {

    implementation(libs.androidx.core.ktx)
    implementation(libs.androidx.appcompat)
    implementation(libs.material)
    implementation(libs.androidx.activity)
    implementation(libs.androidx.constraintlayout)
    implementation(libs.androidx.room.runtime)
    implementation(libs.androidx.room.ktx)
    implementation(libs.play.services.ads)
    implementation(libs.androidx.activity.ktx)
    kotlin("kapt", "androidx.room:room-compiler:2.4.0")
    implementation("com.google.android.gms:play-services-maps:19.0.0")
    implementation("com.google.android.gms:play-services-location:21.3.0")
    implementation("org.jetbrains.kotlinx:kotlinx-coroutines-core:1.7.3")
    implementation("org.jetbrains.kotlinx:kotlinx-coroutines-android:1.7.3")
    testImplementation(libs.junit)
    androidTestImplementation(libs.androidx.junit)
    androidTestImplementation(libs.androidx.espresso.core)
}
```

مرحله 2: تعریف مدل داده‌ها و پایگاه داده

1. تعریف کلاس مدل `Measurement`:

- یک کلاس دیتابیس با استفاده از آنتوتیشن @Entity ایجاد کردیم که شامل فیلدهای مورد نیاز برای ذخیره اطلاعات سیگنال بود.

```kotlin
@Entity(tableName = "measurements")
data class Measurement(
    @PrimaryKey(autoGenerate = true) val id: Long = 0,
    val latitude: Double,
    val longitude: Double,
    val timestamp: Long,
    val technology: String,
    val plmnId: String,
    val lac: Int?,
    val rac: Int?,
    val tac: Int?,
    val cellId: Long,
    val rsrp: Int?,
    val rsrq: Int?,
    val rscp: Int?,
    val eCNo: Int?
```

2. تعریف DAO برای دسترسی به داده‌ها:

- یک اینترفیس DAO ایجاد کردیم تا عملیات CRUD روی پایگاه داده را انجام دهد.

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

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

3. تعریف پایگاه داده Room:

- یک کلاس دیتابیس ایجاد کردیم که به عنوان نقطه دسترسی اصلی به پایگاه داده عمل می‌کند.

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

    companion object {
        @Volatile
        private var INSTANCE: AppDatabase? = null

        fun getDatabase(context: Context): AppDatabase {
            return INSTANCE ?: synchronized( lock: this) {
                val instance = Room.databaseBuilder(
                    context.applicationContext,
                    AppDatabase::class.java,
                     name: "measurements_database"
                ).build()
                INSTANCE = instance
                instance
            }
        }
    }
}
```

مرحله 3: تنظیمات MainActivity

1. افزودن دسترسی‌ها:

- در فایل AndroidManifest.xml دسترسی‌های مورد نیاز را اضافه کردیم.

```xml
<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"/>
```

. تنظیم `MainActivity`:

- متغیرهای مورد نیاز را برای مدیریت مکان و سیگنال تعریف کردیم.

- بررسی و درخواست دسترسی‌ها را پیاده‌سازی کردیم.

`

```
class MainActivity : AppCompatActivity(), OnMapReadyCallback {

    //private lateinit var map: GoogleMap
    private var mGoogleMap: GoogleMap? = null
    private lateinit var telephonyManager: TelephonyManager
    private lateinit var db: AppDatabase
    private lateinit var fusedLocationClient: FusedLocationProviderClient

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val mapFragment = supportFragmentManager
            .findFragmentById(R.id.mapFragment) as SupportMapFragment
        mapFragment.getMapAsync( callback: this)
    }
    override fun onMapReady(googleMap: GoogleMap) {
        mGoogleMap = googleMap
    }
//    @RequiresApi(Build.VERSION_CODES.R)
//    override fun onCreate(savedInstanceState: Bundle?) {
//        super.onCreate(savedInstanceState)
//        setContentView(R.layout.activity_main)
//
//        db = AppDatabase.getDatabase(this)
//
//        if (!hasPermissions()) {
//            requestPermissions()
//        } else {
```

```
//            requestPermissions()
//        } else {
//            startTracking()
//        }
//
//        fusedLocationClient = LocationServices.getFusedLocationProviderClient(thi
//        telephonyManager = getSystemService(TELEPHONY_SERVICE) as TelephonyManage
//
//        val mapFragment = supportFragmentManager
//            .findFragmentById(R.id.map) as SupportMapFragment
//        mapFragment.getMapAsync(this)
//    }

    private fun hasPermissions(): Boolean {
        return ContextCompat.checkSelfPermission(
            context: this, Manifest.permission.ACCESS_FINE_LOCATION
        ) == PackageManager.PERMISSION_GRANTED &&
                ContextCompat.checkSelfPermission(
                    context: this, Manifest.permission.ACCESS_COARSE_LOCATION
                ) == PackageManager.PERMISSION_GRANTED &&
                ContextCompat.checkSelfPermission(
                    context: this, Manifest.permission.READ_PHONE_STATE
                ) == PackageManager.PERMISSION_GRANTED
    }

    private fun requestPermissions() {
        ActivityCompat.requestPermissions(
            activity: this,
            arrayOf(
```

```
private fun requestPermissions() {
    ActivityCompat.requestPermissions(
        activity: this,
        arrayOf(
            Manifest.permission.ACCESS_FINE_LOCATION,
            Manifest.permission.ACCESS_COARSE_LOCATION,
            Manifest.permission.READ_PHONE_STATE
        ),
        requestCode: 1
    )
}

@RequiresApi(Build.VERSION_CODES.R)
private fun startTracking() {
    if (ActivityCompat.checkSelfPermission(
            context: this,
            Manifest.permission.ACCESS_FINE_LOCATION
        ) != PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermissi
            context: this,
            Manifest.permission.ACCESS_COARSE_LOCATION
        ) != PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermissi
            context: this,
            Manifest.permission.READ_PHONE_STATE
        ) != PackageManager.PERMISSION_GRANTED
    ) {
        return
    }
    fusedLocationClient.lastLocation.addOnSuccessListener { location: Location? ->
        location?.let {
```

```kotlin
@RequiresApi(Build.VERSION_CODES.R)
private val requestPermissionLauncher = registerForActivityResult(
    ActivityResultContracts.RequestPermission()
) { isGranted: Boolean ->
    if (isGranted) {
        mGoogleMap?.let { onMapReady(it) }
    }
}

@RequiresApi(Build.VERSION_CODES.R)
private fun collectSignalData() {
    if (ActivityCompat.checkSelfPermission(
            context: this,
            Manifest.permission.READ_PHONE_STATE
        ) != PackageManager.PERMISSION_GRANTED
    ) {
        requestPermissionLauncher.launch(Manifest.permission.READ_PHONE_STATE)
        return
    }
    val cellInfoList = telephonyManager.allCellInfo
    for (cellInfo in cellInfoList) {
        when (cellInfo) {
            is CellInfoLte -> {
                val cellIdentityLte = cellInfo.cellIdentity as CellIdentityLte
                val cellSignalStrengthLte = cellInfo.cellSignalStrength as CellS
                fusedLocationClient.lastLocation.addOnSuccessListener { locatior
                    location?.let {
```

```
is CellInfoGsm -> {
    val cellIdentityGsm = cellInfo.cellIdentity as CellIdentityGsm
    val cellSignalStrengthGsm = cellInfo.cellSignalStrength as CellSign
    fusedLocationClient.lastLocation.addOnSuccessListener { location: L
        location?.let {
            val measurementGSM = Measurement(
                latitude = it.latitude,
                longitude = it.longitude,
                timestamp = System.currentTimeMillis(),
                technology = "GSM",
                plmnId = cellIdentityGsm.mccString + cellIdentityGsm.mn
                lac = cellIdentityGsm.lac,
                tac = null,
                cellId = cellIdentityGsm.cid.toLong(),
                rsrp = null,
                rsrq = null,
                rscp = null,
                eCNo = null,
                rac = null
            )
            lifecycleScope.launch {
                db.measurementDao().insert(measurementGSM)
            }
        }
    }
}
```

```
@SuppressLint("MissingPermission")
private fun getNetworkType(): String {
    return when (telephonyManager.networkType) {
        TelephonyManager.NETWORK_TYPE_GSM -> "GSM"
        TelephonyManager.NETWORK_TYPE_GPRS -> "GPRS"
        TelephonyManager.NETWORK_TYPE_EDGE -> "EDGE"
        TelephonyManager.NETWORK_TYPE_UMTS -> "UMTS"
        TelephonyManager.NETWORK_TYPE_HSPA -> "HSPA"
        TelephonyManager.NETWORK_TYPE_HSPAP -> "HSPA+"
        TelephonyManager.NETWORK_TYPE_LTE -> "LTE"
        TelephonyManager.NETWORK_TYPE_NR -> "5G"
        else -> "UNKNOWN"
    }
}

@RequiresApi(Build.VERSION_CODES.R)
override fun onRequestPermissionsResult(
    requestCode: Int, permissions: Array<out String>, grantResults: IntArray
) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults)
    if (requestCode == 1) {
        if (grantResults.isNotEmpty() && grantResults[0] == PackageManager.PE
            startTracking()
        }
    }
}

@RequiresApi(Build.VERSION_CODES.R)
fun onLocationChanged(location: Location) {
    collectSignalData()
```

مرحله 4: تنظیمات رابط کاربری

1. **تغییر فایل `activity_main.xml`:**

- یک دکمه برای شروع مکان‌یابی و نمایش اطلاعات به فایل XML اضافه کردیم.

```
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.a
    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">

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/mapFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        class = "com.google.android.gms.maps.SupportMapFragment"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        />



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

مرحله 5: تکمیل پروژه

- پروژه را کامپایل و اجرا کردیم تا مطمئن شویم که تمام مراحل به درستی کار می‌کنند. در صورت بروز خطا، آن‌ها را رفع کردیم و مجدداً اجرا کردیم.

- پس از اطمینان از عملکرد صحیح، اطلاعات جمع‌آوری شده در پایگاه داده ذخیره شدند و با استفاده از ViewModel و LiveData نمایش داده شدند.

جمع‌بندی

در این پروژه، با استفاده از تکنولوژی‌های مختلف Android، اطلاعات سیگنال موبایل و مکان جغرافیایی کاربران را جمع‌آوری و ذخیره کردیم. این اطلاعات می‌تواند برای تحلیل کیفیت شبکه و ارائه خدمات بهتر به کاربران استفاده شود.