Mobile Application Development

Week8　Use Location data in Android

Yi SUN
Kobe Institute of Computing

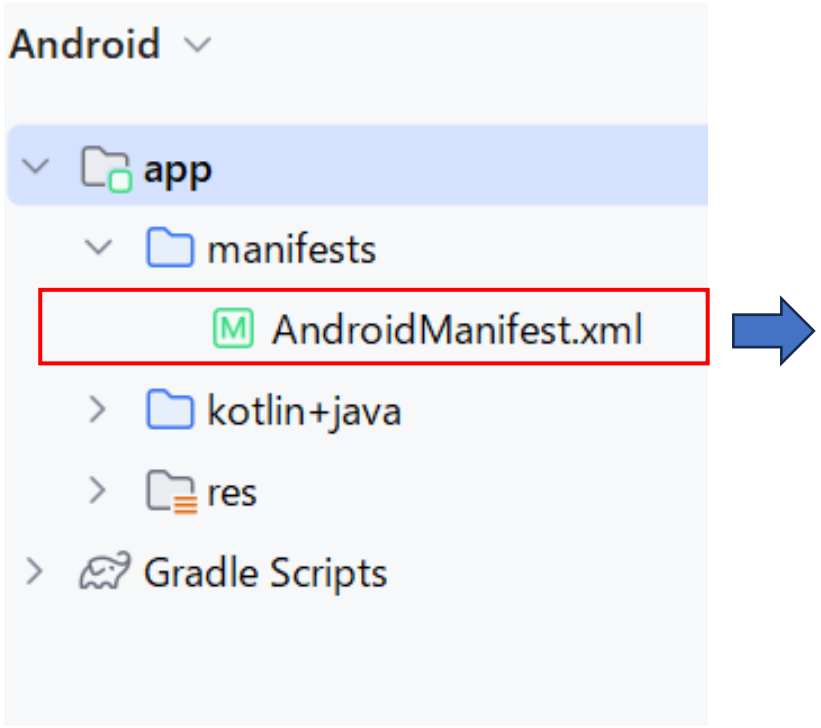
Understand Location Services by simple sample

- We are building a simple greeting app.

1. Open AndroidStudio
2. Select `File > New > NewProject`
3. Choose `Empty Activity`
4. Name: LocationSample

Add Permissons for Location Serves in AndroidManifest.xml

- Edit the AndroidManifest.xml file in app/manifests/ folder



AndroidManifest.xml

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

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


    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
```

https://developer.android.com/reference/android/Manifest.permission

Check we have the permissions to access the location services

Create a New class file LocationUtils to check the App have the permissions to access the location services. This code is a utility for managing runtime location permissions in an Android app. By abstracting the permission check into a reusable method (***hasLocationPermission***), developers can easily verify if the necessary location access has been granted before performing location-dependent operations.

LocationUtils.kt

It takes a Context object, allowing it to interact with Android's permission system.

```
class LocationUtils(val context: Context) {
    fun hasLocationPermission(context: Context): Boolean{
        if (ContextCompat.checkSelfPermission(
            context,Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED
            &&
            ContextCompat.checkSelfPermission(
                context,Manifest.permission.ACCESS_COARSE_LOCATION) == PackageManager.PERMISSION_GRANTED
        ){
            return true
        }else{
            return false
        }
    }
}
```

uses ContextCompat.checkSelfPermission() to check each permission.

The UI of check the permission

MainActivity.kt

The Android context used for permission checks and UI operations such as showing Toasts or launching permission requests.

```
@Composable
fun LocationDisplay(
    locationUtils: LocationUtils,
    context: Context
){
    val requestPermissionLauncher = rememberLauncherForActivityResult(
        contract = ActivityResultContracts.RequestMultiplePermissions(),
        onResult = { permissions ->
            if(permissions[Manifest.permission.ACCESS_COARSE_LOCATION] == true
                && permissions[Manifest.permission.ACCESS_FINE_LOCATION] == true){
                // I have access to location
            }else{
                // Ask for permission
            }
        }
    )
    Column(modifier = Modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center){
        Text(text =" Location not avalilabe")
        Button(onClick = {
            if(locationUtils.hasLocationPermission(context))
            {
                //Permission already granted update the location
            }else{
                //request location permission
            }
        }) {
            Text(text="Get Location")
        }
}
```

The callback receives a map Keys: Permission names (e.g., Manifest.permission.ACCESS_FINE_LOCATION).Values: Booleans indicating whether each permission was granted.

This function creates a launcher that can requesting permissions. It uses the ***ActivityResultContracts.RequestMultiplePermissions()*** contract, which allows you to request several permissions at once.

UI Layout

The code calls ***locationUtils.hasLocationPermission(context)*** to determine if the required location permissions are already granted.

Permission Request Launcher

MainActivity.kt

```
onResult = { permissions ->
    if(permissions[Manifest.permission.ACCESS_COARSE_LOCATION] == true
        && permissions[Manifest.permission.ACCESS_FINE_LOCATION] == true){
        // I have access to location
    }else{
        val rationaleRequired = ActivityCompat.shouldShowRequestPermissionRationale(
            context as MainActivity,
            Manifest.permission.ACCESS_FINE_LOCATION
        ) || ActivityCompat.shouldShowRequestPermissionRationale(
            context as MainActivity,
            Manifest.permission.ACCESS_COARSE_LOCATION
        )
        if(rationaleRequired){
            Toast.makeText(context,
                "Location permission is required for this feature to work",
                Toast.LENGTH_LONG).show()
        }else{
            Toast.makeText(context,
                "Location permission is required.please enable it in the Android Settings",
                Toast.LENGTH_LONG).show()
        }
    }
```

Granted Case:
If both **ACCESS_COARSE_LOCATION** and **ACCESS_FINE_LOCATION** are granted (true), the code can proceed with location-related functionality (placeholder comment indicates where additional logic would go).

Denied Case & Rationale Check:
If one or both permissions are not granted, it checks whether the app should display a rationale using ***ActivityCompat.shouldShowRequestPermissionRationale***.

With Rationale:A Toast is shown explaining that location permission is needed for the feature to work.

Without Rationale:A different Toast informs the user to enable the permission via Android Settings.

The Button Action

MainActivity.kt

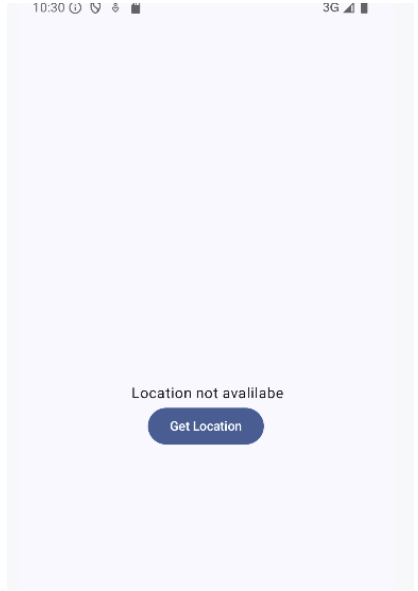
```
Column(modifier = Modifier.fillMaxSize(),
    horizontalAlignment = Alignment.CenterHorizontally,
    verticalArrangement = Arrangement.Center){
    Text(text =" Location not avalilabe")
    Button(onClick = {
        if(locationUtils.hasLocationPermission(context))
        {
            //Permission already granted update the location
        }else{
            requestPermissionLauncher.launch(
                arrayOf(
                    Manifest.permission.ACCESS_FINE_LOCATION,
                    Manifest.permission.ACCESS_COARSE_LOCATION
                )
            )
        }
    }) {
        Text(text="Get Location")
    }
```

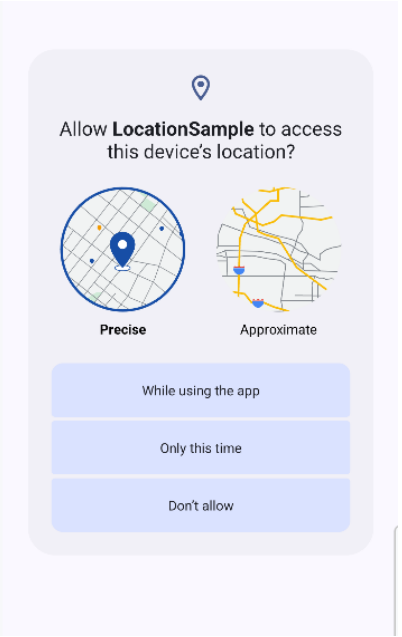
If permissions are not granted, it triggers the ***requestPermissionLauncher*** to request both fine and coarse location permissions.

Load the app to the MainActivity

MainActivity.kt

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            LocationSampleTheme {
                Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
                    LocationApp(
                        modifier = Modifier.padding(innerPadding)
                    )
                }
            }
        }
    }
}

@Composable
fun LocationApp(modifier: Modifier){
    val context = LocalContext.current
    val locationUtils = LocationUtils(context)
    LocationDisplay(locationUtils=locationUtils, context = context)
}
```





Add a Data class for location data

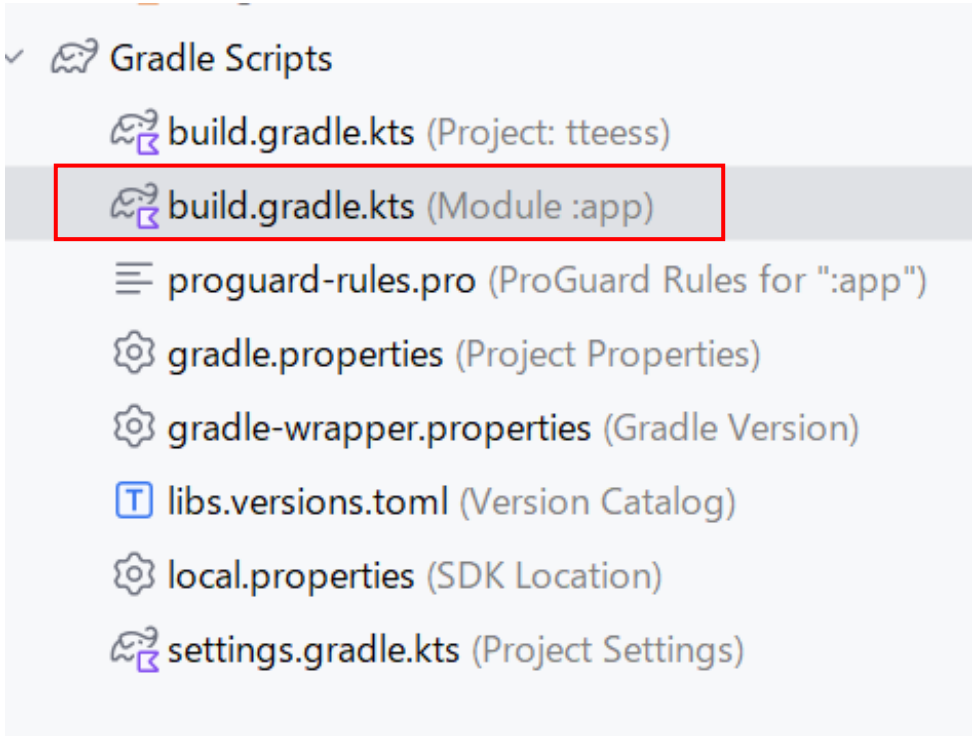
Create a New data class file **LocationData** to store the location data include the latitude data and longitude data.

LocationData.kt

```
data class LocationData (
    val latitude: Double,
    val longitude: Double
)
```

Add the Dependency

- Make sure you've added the correct dependency in your build.gradle (app-level) file:




```
dependencies {

    implementation("androidx.lifecycle:lifecycle-viewmodel-compose:2.8.7")
    implementation("com.google.android.gms:play-services-location:21.3.0")
    implementation(libs.androidx.core.ktx)
    implementation(libs.androidx.lifecycle.runtime.ktx)
    implementation(libs.androidx.activity.compose)
```

- Sync the Project:After making these changes, sync your project in Android Studio to download the dependency.
- Cache/Index Issues:If everything seems correct but the dependency still isn't found, try cleaning the project and rebuilding it. You can also invalidate caches via File > Invalidate Caches / Restart... in Android Studio.

https://developers.google.com/android/guides/setup

Add a ViewModel for Location data

Create a New file **LocationViewModel** to present the location

LocationViewModel.kt

```
class LocationViewModel: ViewModel() {
    private val _location = mutableStateOf<LocationData?>(null)
    val location: State<LocationData?> = _location

    fun updateLocation(newLocation: LocationData){
        _location.value = newLocation
    }
}
```

Add the viewModel to MainActiviity

MainActivity.kt

```
        enableEdgeToEdge()
        setContent {
            val viewModel: LocationViewModel = viewModel()
            LocationSampleTheme {
                Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->
                    LocationApp(
                        modifier = Modifier.padding(innerPadding),
                        viewModel
                    )
……
}

@Composable
fun LocationApp(modifier: Modifier, viewModel: LocationViewModel){
    val context = LocalContext.current
    val locationUtils = LocationUtils(context)
    LocationDisplay(locationUtils=locationUtils,viewModel, context = context)
}

@Composable
fun LocationDisplay(
    locationUtils: LocationUtils,
    viewModel: LocationViewModel,
    context: Context
){
    val requestPermissionLauncher = rememberLauncherForActivityResult(
```

Create an instance of the Fused Location Provider Client to take the location data

LocationUtils.kt

```
class LocationUtils(val context: Context) {

    private  val _fusedLocationClient: FusedLocationProviderClient = LocationServices.getFusedLocationProviderClient(context)

    @SuppressLint("MissingPermission")
    fun requestLocationUpdates(viewModel: LocationViewModel){
        val locationCallback = object: LocationCallback(){
            override fun onLocationResult(locationResult: LocationResult) {
                super.onLocationResult(locationResult)
                locationResult.lastLocation?.let {
                    val location = LocationData(latitude = it.latitude, longitude = it.longitude)
                    viewModel.updateLocation(location)
                }
            }
        }

        val locationRequest = LocationRequest.Builder(Priority.PRIORITY_HIGH_ACCURACY,1000).build()

        _fusedLocationClient.requestLocationUpdates(locationRequest,locationCallback, Looper.getMainLooper())
    }
```

Initializes a client to interact with Google's location services. The Fused Location Provider API efficiently combines signals from GPS, Wi-Fi, and cellular networks to provide accurate location data.

This annotation tells the compiler to ignore warnings about missing location permission checks.

Defines a callback that will be triggered when new location data is available.

Priority:Uses Priority.PRIORITY_HIGH_ACCURACY to request the most precise location available.
Interval:The second parameter, 1000, represents the desired interval in milliseconds (i.e., 1 second) for location updates.

Initiates the request for location updates using the configured request and callback.
locationRequest: Specifies the configuration for the updates.
locationCallback: The callback that handles incoming location data.
Looper.getMainLooper(): Ensures that the callback is executed on the main thread, which is necessary for updating UI elements or interacting with view models

https://developer.android.com/develop/sensors-and-location/location/retrieve-current

Request the location data to MainActivity and show it

MainActivity.kt

```
fun LocationDisplay(
    locationUtils: LocationUtils,
    viewModel: LocationViewModel,
    context: Context
){
    val location = viewModel.location.value
    val requestPermissionLauncher = rememberLauncherForActivityResult(
        contract = ActivityResultContracts.RequestMultiplePermissions(),
        onResult = { permissions ->
            if(permissions[Manifest.permission.ACCESS_COARSE_LOCATION] == true
                && permissions[Manifest.permission.ACCESS_FINE_LOCATION] == true){

                locationUtils.requestLocationUpdates(viewModel=viewModel)

            }else{
                val rationaleRequired = ActivityCompat.shouldShowRequestPermissionRationale(
                    context as MainActivity,
                    Manifest.permission.ACCESS_FINE_LOCATION
                ) || ActivityCompat.shouldShowRequestPermissionRationale(
```

Retrieves the current location value from the view model.

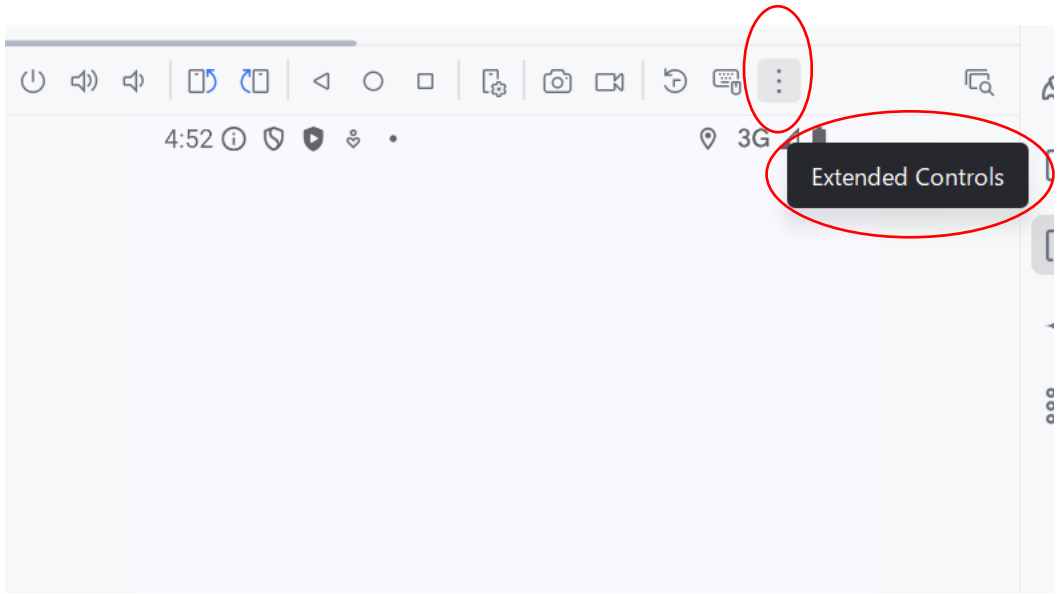
If both permissions are granted, the function ***locationUtils.requestLocationUpdates(viewModel = viewModel)*** is called. This method likely starts the process of receiving location updates, passing along the view model to update the location data accordingly.

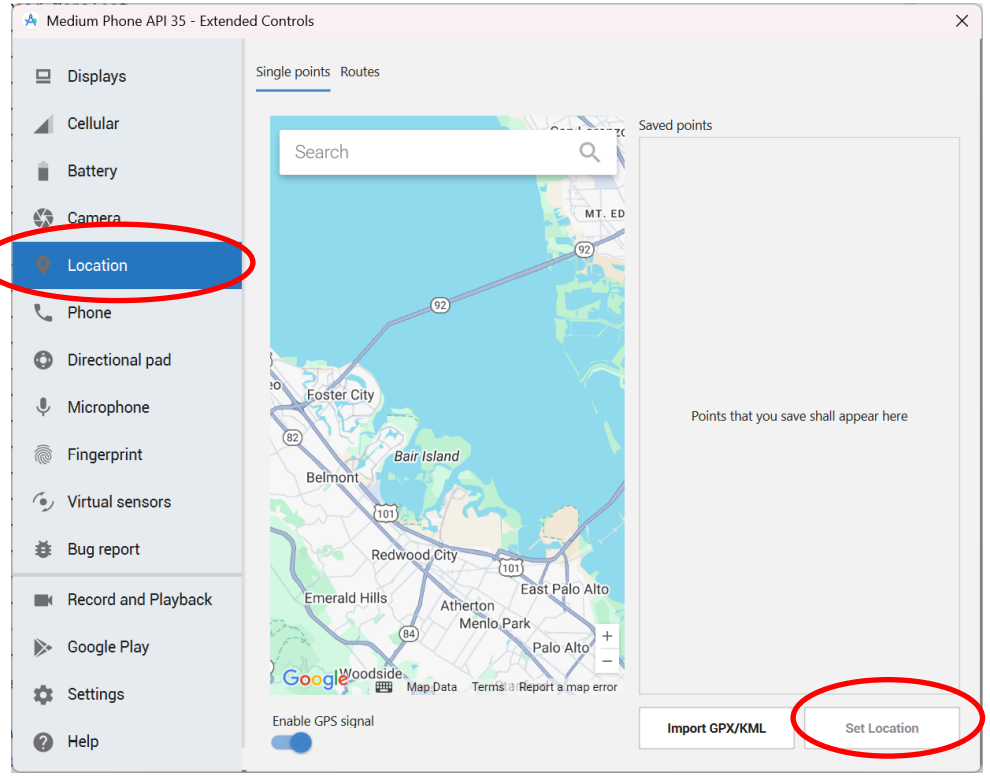
MainActivity.kt

```
Column(modifier = Modifier.fillMaxSize(),
    horizontalAlignment = Alignment.CenterHorizontally,
    verticalArrangement = Arrangement.Center){

    if(location != null){
        Text("Address: ${location.latitude} ${location.longitude}")
    }else{
        Text(text =" Location not avalilabe")
    }

    Button(onClick = {
```

When location is not null, it displays a Text composable showing the latitude and longitude:

Use the Emulator to test the location



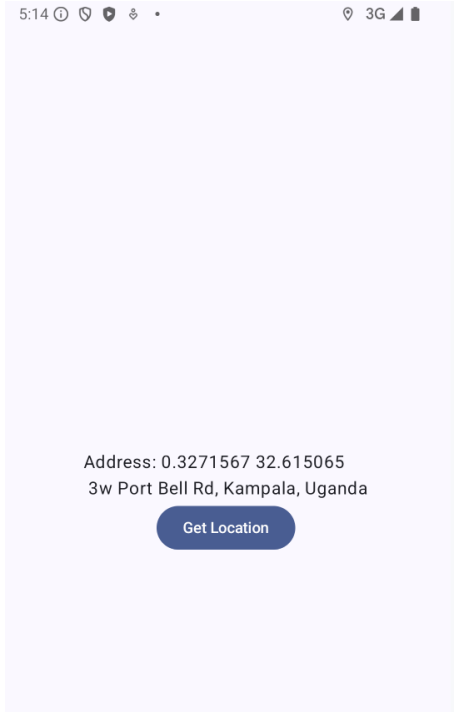


Show the address based on location data

- Add a reverse function in LocationUtils.kt file

LocationUtils.kt

```
fun reverseGeocodeLocation(location: LocationData): String{
    val geocoder = Geocoder(context, Locale.getDefault())
    val coordinate = LatLng(location.latitude, location.longitude)
    val addresses:MutableList<Address>? = geocoder.getFromLocation(coordinate.latitude,coordinate.longitude,1)
    return if(addresses?.isNotEmpty() == true){
        addresses[0].getAddressLine(0)
    }else{
        "Address not found"
    }
}
```



MainActivity.kt

```
val location = viewModel.location.value
val address = location?.let{
    locationUtils.reverseGeocodeLocation(location)
}
```

MainActivity.kt

```
if(location != null){
    Text("Address: ${location.latitude} ${location.longitude} ¥n $address")
}else{
    Text(text =" Location not avalilabe")
}
```