Objective

The objective of this document is to define the end-to-end approach for integrating the Lepton GIS hybrid application—built using React, Vite, deck.gl, Tailwind, and Capacitor—into the HSC native mobile app. This integration should enable HSC users to access Lepton's GIS capabilities within the HSC app.

IT & Build Environment Prerequisites

- **Network Whitelisting (to be handled by IT team)**
  The following URLs must be allowlisted to enable dependency installation and native builds:
  - https://registry.npmjs.org/ — npm packages
  - https://maven.google.com/ — Android libraries
  - https://repo1.maven.org/maven2/ — Maven Central
  - https://services.gradle.org/ — Gradle services
- **TLS / Secure Socket Configuration**
  - Ensure **TLS sockets / HTTPS traffic** are enabled and not blocked by corporate network policies, as secure connections are required for dependency resolution, build services, and runtime communications.
- **Build Tooling Configuration**
  - **Yarn** must be available in the build environment
  - Required **Yarn environment variables** should be added to the **global environment variables** by the IT team

Prerequisites

Before proceeding, ensure that the following prerequisites are met:

- The **Lepton Android codebase**, named `hsc-android`, exists at the same directory level as `deal_wf_mcsa-develop`.
- The `hsc-android` project has been successfully built by running the following commands in sequence:

```
yarn install
yarn build
npx cap sync
```

Exact Steps for integration:

Step 1: Update Root `build.gradle.kts`

File:
`deal_wf_mcsa-develop/build.gradle.kts`

Changes to Apply

1. **Add the following import at the top of the file:**
    - `import com.android.build.gradle.LibraryExtension`

2. **Apply Capacitor variables file:**
    - Add the following line **after the `plugins {}` block**:
        - `apply(from = "variables.gradle")`

3. **Add a namespace override for the Capacitor file picker plugin:**

    - Append the following **at the bottom of the file**:
        - A `subprojects {}` block that sets the `namespace` for the `capacitor-file-picker` module to comply with AGP 8+ requirements.

Reason

- `variables.gradle` provides required SDK versions to all Capacitor modules.
- During the **initial attempt to integrate the `capacitor-file-picker` plugin**, a **"Namespace not specified"** build error was encountered when using **Android Gradle Plugin (AGP) 8+**.
  The namespace override was added to resolve this issue.

Step 2: Update `app/build.gradle.kts`

File Modified:
`deal_wf_mcsa-develop/app/build.gradle.kts`

Changes Made

1. **Enable MultiDex** *(inside `android { defaultConfig { ... } }`)*:

```
defaultConfig {
    // ... existing configs ...
    multiDexEnabled = true
}
```

2. **Disable R8 in debug** *(inside `android { buildTypes { debug { ... } } }`)*:

```
buildTypes {
    debug {
        // Disable R8 in debug to prevent plugin loading issues
        isMinifyEnabled = false
    }
}
```

3. **Add required dependencies** *(keep existing ones; add only if missing) (inside `dependencies { ... }`)*:

```
dependencies {
    // Capacitor core
    implementation(project(":capacitor-android"))

    // Capacitor plugins
    implementation(project(":capacitor-app"))
    implementation(project(":capacitor-filesystem"))
    implementation(project(":capacitor-geolocation"))
    implementation(project(":capacitor-preferences"))
    implementation(project(":capacitor-share"))
    implementation(project(":capacitor-file-picker"))

    // Cordova compatibility
    implementation(project(":capacitor-cordova-android-plugins"))

    // Geolocation dependencies
    implementation("io.ionic.libs:iongeolocation-android:1.0.0")
    implementation("com.google.android.gms:play-services-location:21.3.0")
    implementation("org.jetbrains.kotlinx:kotlinx-coroutines-play-services:1.6.4")

    // WebView support
    implementation("androidx.webkit:webkit:1.12.1")
    // Local tile server
    implementation("org.nanohttpd:nanohttpd:2.3.1")


    // MultiDex runtime
    implementation("androidx.multidex:multidex:2.0.1")
}
```

Reason

Required configuration and dependencies to support **Capacitor core/plugins**, **geolocation services**, **WebView enhancements**, **local tile server functionality**, and **MultiDex** for projects that exceed the Android method limit.

Step 3: Update `AndroidManifest.xml`

File Modified:
`deal_wf_mcsa-develop/app/src/main/AndroidManifest.xml`

Changes Made

1) Permissions Added

(Add these immediately after the existing `ACCESS_NETWORK_STATE` permission)

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

2) Permissions Updated

Update `READ_EXTERNAL_STORAGE`

```
<!-- BEFORE -->

<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />



<!-- AFTER -->

<uses-permission

    android:name="android.permission.READ_EXTERNAL_STORAGE"

    android:maxSdkVersion="32" />
```

Update `MANAGE_EXTERNAL_STORAGE`

```
<!-- BEFORE -->

<uses-permission android:name="android.permission.MANAGE_EXTERNAL_STORAGE" />



<!-- AFTER -->

<uses-permission

    android:name="android.permission.MANAGE_EXTERNAL_STORAGE"

    tools:ignore="ScopedStorage" />
```

Update `WRITE_EXTERNAL_STORAGE`

```


<!-- BEFORE -->

<uses-permission

    android:name="android.permission.WRITE_EXTERNAL_STORAGE"

    android:maxSdkVersion="28" />
```

```
<!-- AFTER -->
<uses-permission
    android:name="android.permission.WRITE_EXTERNAL_STORAGE"
    android:maxSdkVersion="29" />
```

Already present (no changes required):

- `INTERNET`
- `ACCESS_NETWORK_STATE`
- `READ_MEDIA_IMAGES`, `READ_MEDIA_VIDEO`, `READ_MEDIA_AUDIO`

3) Application Attributes Added

(Add the following attributes to the existing `<application>` tag — place them before `tools:targetApi="31"`)

```
android:requestLegacyExternalStorage="true"
android:usesCleartextTraffic="true"
```

4) Add FileProvider for Capacitor

*(Add this provider **after the existing FileProvider** and **before** the closing `</application>` tag)*

```
<provider
    android:name="androidx.core.content.FileProvider"
    android:authorities="${applicationId}.fileprovider"
    android:exported="false"
    android:grantUriPermissions="true">
    <meta-data
```

```
        android:name="android.support.FILE_PROVIDER_PATHS"

        android:resource="@xml/file_paths" />

</provider>
```

Note: The app already contains a FileProvider using `${applicationId}.provider` for existing functionality. This is an **additional** FileProvider required specifically for Capacitor.

Step 4: Create `file_paths.xml`

File Created:
`deal_wf_mcsa-develop/app/src/main/res/xml/file_paths.xml`

File Content:

```
<?xml version="1.0" encoding="utf-8"?>

<paths>

    <external-path name="external" path="." />

    <external-files-path name="external_files" path="." />

    <cache-path name="cache" path="." />

    <files-path name="files" path="." />

</paths>
```

Reason

This file is required by the Capacitor FileProvider added in Step 3 to define the file system locations that can be safely shared with other apps**.**

Step 5: Copy Custom Plugins from hsc-android

1). MemberActionPlugin.kt

2). NativeUploaderPlugin.kt

3). OfflineTileServerPlugin.kt

4). UdpPlugin.kt

5). ZipFolderPlugin.kt

From File location:
`hsc-android/kt-msca-plugins/`

To file location:
`deal_wf_mcsa-develop/app/src/main/java/org/deal/mcsa/plugins/`

Step 6: Copy GisCapacitorFragment.kt

From File location:
hsc-android/mcsa-fragment-files/

To file location:
deal_wf_mcsa-develop/app/src/main/java/org/deal/mcsa/GisCapacitorFragment.kt

Step 7: Copy GisScreen.kt (Compose Screen)

From File location:
hsc-android/mcsa-screen-files/

To file location:
deal_wf_mcsa-develop/app/src/main/java/org/deal/mcsa/presentation/ui/screens/GisScreen.kt

Step 8: Add GisScreen to Navigation

File to Modify:
deal_wf_mcsa-develop/app/src/main/java/org/deal/mcsa/presentation/ui/screens/HomeScreen.kt

Code to add :

```
AppDestination.Gis -> GisScreen()
```

Step 9: Copy Web Assets

Assets to Copy

- **Web build output**
  Copy: hsc-android/dist/*
  To: deal_wf_mcsa-develop/app/src/main/assets/public/

- **Capacitor plugins metadata**
  Copy: hsc-android/android/app/src/main/assets/capacitor.plugins.json
  To: deal_wf_mcsa-develop/app/src/main/assets/

- **Capacitor runtime config**
  Copy: hsc-android/android/app/src/main/assets/capacitor.config.json
  To: deal_wf_mcsa-develop/app/src/main/assets/

Important

1. These assets must be copied **after building hsc-android** (yarn build).
2. If hsc-android is rebuilt, **copy the assets again** to ensure the latest web bundle and Capacitor metadata are included in the app.

Step 10: Additional Required File Changes

File 1: Update BaseActivity.kt

File Modified:
deal_wf_mcsa-develop/app/src/main/java/org/deal/mcsa/presentation/ui/activity/BaseActivity.kt

Change Required: Update the class declaration.

```
// BEFORE

open class BaseActivity : ComponentActivity()



// AFTER

open class BaseActivity : AppCompatActivity()
```

Reason: Capacitor's Bridge requires AppCompatActivity (which extends FragmentActivity), not ComponentActivity.

File 2: Update themes.xml

File Modified:
`deal_wf_mcsa-develop/app/src/main/res/values/themes.xml`

Change Required: Update the theme parent to an AppCompat theme.

```
<style name="Theme.DealMCSA" parent="Theme.AppCompat.Light.NoActionBar" />
```

Reason: Since `BaseActivity` is updated to `AppCompatActivity`, the app theme must inherit from an **AppCompat** parent. This avoids theme/runtime issues that occur when using a **Material** parent theme with `AppCompatActivity`.

File 3: Update HomeActivity.kt

File Modified:
deal_wf_mcsa-develop/app/src/main/java/org/deal/mcsa/presentation/ui/activity/HomeActivity.kt

Changes Made (Code)

1) Add these imports (along with existing imports)

```
import android.content.Intent                 // ADDED (for storage permission)

import android.net.Uri                        // ADDED (for storage permission)

import android.os.Build                       // ADDED (for storage permission)

import android.os.Environment                 // ADDED (for storage permission)

import android.provider.Settings              // ADDED (for storage permission)

import androidx.appcompat.app.AlertDialog   // ADDED (for storage permission
dialog)
```

2) Update onCreate() to call storage permission check

```
override fun onCreate(savedInstanceState: Bundle?) {

    super.onCreate(savedInstanceState)

    // ... existing code ...

    checkPermissions()

    checkStoragePermission()   // ADDED (call storage permission check)

    // ... rest of existing code ...

}
```

3) Update `checkPermissions()` to include location permissions

```
private fun checkPermissions() {

  val permissions = mutableListOf<String>()



  // Camera, Audio, Bluetooth permissions (existing - unchanged)
```

```
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.CAMERA)

        != android.content.pm.PackageManager.PERMISSION_GRANTED) {

        permissions.add(Manifest.permission.CAMERA)

    }

    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.RECORD_AUDIO)

        != android.content.pm.PackageManager.PERMISSION_GRANTED) {

        permissions.add(Manifest.permission.RECORD_AUDIO)

    }

    if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.BLUETOOTH_CONNECT)

        != android.content.pm.PackageManager.PERMISSION_GRANTED) {

        permissions.add(Manifest.permission.BLUETOOTH_CONNECT)

    }

    // ADDED: Location permissions (required for Geolocation plugin)

    if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION)

        != android.content.pm.PackageManager.PERMISSION_GRANTED) {

        permissions.add(Manifest.permission.ACCESS_FINE_LOCATION)

    }

    if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION)
```

```
        != android.content.pm.PackageManager.PERMISSION_GRANTED) {

        permissions.add(Manifest.permission.ACCESS_COARSE_LOCATION)

    }



    if (permissions.isNotEmpty()) {

        ActivityCompat.requestPermissions(this, permissions.toTypedArray(), 0)

    }

}
```

4) Add storage permission methods (for offline tile server)

```


private fun checkStoragePermission() {

    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {

        if (!Environment.isExternalStorageManager()) {

            showStoragePermissionDialog()

        } else {

            Log.d("HomeActivity", "Storage permission already granted")

        }

    }

}
```

```
private fun showStoragePermissionDialog() {

    AlertDialog.Builder(this)

        .setTitle("Storage Permission Required")

        .setMessage(

            "This app needs access to manage all files for the GIS map tile server.\n\n" +

            "Without this permission, the map cannot:\n" +

            "• Load offline map tiles\n" +

            "• Access tile files\n" +

            "• Display the map correctly\n\n" +

            "Please tap 'Open Settings' and enable 'Allow access to manage all files'."

        )

        .setPositiveButton("Open Settings") { _, _ ->

            requestManageStoragePermission()

        }

        .setNegativeButton("Cancel") { dialog, _ ->

            dialog.dismiss()

            Toast.makeText(this, "Map may not work without storage permission",
Toast.LENGTH_LONG).show()

        }

        .setIcon(android.R.drawable.ic_dialog_alert)

        .setCancelable(false)

        .show()
```

```
}


private fun requestManageStoragePermission() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
        try {
            val intent = Intent(Settings.ACTION_MANAGE_APP_ALL_FILES_ACCESS_PERMISSION)
            intent.data = Uri.parse("package:$packageName")
            startActivity(intent)
        } catch (e: Exception) {
            try {
                val intent = Intent(Settings.ACTION_MANAGE_ALL_FILES_ACCESS_PERMISSION)
                startActivity(intent)
            } catch (e2: Exception) {
                Log.e("HomeActivity", "Failed to open storage settings: ${e2.message}", e2)
                Toast.makeText(
                    this,
                    "Could not open settings. Please grant storage permission manually.",
                    Toast.LENGTH_LONG
                ).show()
            }
        }
```

```
  }

}
```

Reason

Runtime permissions are required for:

- **Capacitor Geolocation plugin** (location permissions)
- **Offline GIS tile server** (Android 11+ “Manage All Files” flow to access tile files)

File 4: Update `MCSADealApp.kt`

File Modified:
`deal_wf_mcsa-develop/app/src/main/java/org/deal/mcsa/MCSADealApp.kt`

---

1) Add the following imports *(top of file with existing imports)*

```
import androidx.multidex.MultiDex                       // ADDED

import androidx.multidex.MultiDexApplication            // ADDED (if you change the
base class)

import android.app.Application                          // ADDED (if you keep
Application as base class)

import android.content.Context                          // ADDED
```

2) Update the application class to install MultiDex

Preferred (if you can change the base class):

```
class MCSADealApp : MultiDexApplication() {

    // ... existing code ...
    override fun attachBaseContext(base: Context) {

        super.attachBaseContext(base)

        MultiDex.install(this)

    }

}
```

If you must keep extending `Application`:

```
class MCSADealApp : Application() {

    // ... existing code ...



    override fun attachBaseContext(base: Context) {

        super.attachBaseContext(base)

        MultiDex.install(this)

    }

}
```

Reason

This is required once multiDexEnabled = true is enabled in app/build.gradle.kts, to ensure the app can load methods across multiple DEX files at runtime.

File 5: Update `proguard-rules.pro`

File Modified:
`deal_wf_mcsa-develop/app/proguard-rules.pro`

Change Required

Add the following rules at the **end of the file**:

```
# Keep attributes needed for reflection
-keepattributes *Annotation*
-keepattributes Signature
-keepattributes Exceptions
-keepattributes InnerClasses
-keepattributes EnclosingMethod

# Capacitor Plugin Rules - Keep all Capacitor plugins from being obfuscated
-keep class com.capacitorjs.plugins.** { *; }
-keep class com.getcapacitor.** { *; }
-keepclassmembers class * extends com.getcapacitor.Plugin {
    <init>(...);
    *;
}

# Specifically keep GeolocationPlugin and its dependencies
-keep class com.capacitorjs.plugins.geolocation.** { *; }
-keep class com.capacitorjs.plugins.geolocation.GeolocationPlugin {
    <init>(...);
    *;
}
```

```
# Keep all plugin classes referenced in capacitor.plugins.json
-keep class com.capacitorjs.plugins.app.AppPlugin { *; }
-keep class com.capacitorjs.plugins.filesystem.FilesystemPlugin { *; }
-keep class com.capacitorjs.plugins.geolocation.GeolocationPlugin { *; }
-keep class com.capacitorjs.plugins.preferences.PreferencesPlugin { *; }
-keep class com.capacitorjs.plugins.share.SharePlugin { *; }
-keep class com.epicshaggy.filepicker.FilePicker { *; }

# Keep custom plugins
-keep class org.deal.mcsa.plugins.** { *; }

# Keep PluginHandle and Bridge classes that load plugins
-keep class com.getcapacitor.PluginHandle { *; }
-keep class com.getcapacitor.Bridge { *; }
-keep class com.getcapacitor.Bridge$Builder { *; }
-keep class com.getcapacitor.PluginManager { *; }

# Keep all plugin constructors (critical for reflection-based instantiation)
-keepclassmembers class com.capacitorjs.plugins.** {
    public <init>();
    public <init>(android.content.Context);
}



# Keep all public methods in plugins (needed for JavaScript bridge)

-keepclassmembers class com.capacitorjs.plugins.** {

    public *;

}



# Keep all classes that implement Plugin interface

-keep class * implements com.getcapacitor.Plugin {

    <init>(...);

    *;

}



# Don't obfuscate plugin method names (needed for JavaScript bridge)

-keepclassmembers class com.capacitorjs.plugins.** {

    @com.getcapacitor.annotation.PluginMethod *;

}
```

Reason

Prevents R8/ProGuard from obfuscating Capacitor plugin classes and methods, which can otherwise lead to runtime failures such as **"Unable to load plugin instance"** due to reflection-based plugin loading.

File 6: Fix Screenshot to Capture WebView (`HomeScreen.kt`)

File Modified:

`deal_wf_mcsa-develop/app/src/main/java/org/deal/mcsa/presentation/ui/screens/HomeScreen.kt`

Problem

The screenshot button captures a blank area where the Capacitor WebView should appear. This occurs because WebViews are hardware-accelerated and are not captured using the standard `view.draw(Canvas)` approach.

Solution

Use Android's `PixelCopy` API to capture the actual rendered window surface (equivalent to a system screenshot).

Changes Made (Code)

1) Add Imports

Add the following imports (as indicated, after line 7 / alongside existing imports):

```
import android.graphics.Canvas

import android.view.PixelCopy         // ADDED

import android.os.Handler             // ADDED

import android.os.Looper              // ADDED

import android.view.Window            // ADDED

import android.util.Log
```

2) Replace `onClick` Handler in `ScreenshotButton` (lines ~424–436)

BEFORE (WebView not captured)

```
onClick = {

    isVisible = false

    val view = activity.window.decorView.rootView

    val bitmap = createBitmap(activity.window.decorView.width,
activity.window.decorView.height)

    val canvas = Canvas(bitmap)

    view.draw(canvas)  // This does not capture hardware-accelerated WebView

    file.value = saveScreenshotToCache(activity, bitmap)

    isVisible = true

    if (file.value!!.exists()) {

        Log.d("Screenshot", "Saved at: ${file.value!!.absolutePath}")

        showConfirmDialog.value = true

    }

}
```

AFTER (captures WebView using `PixelCopy`)

```
onClick = {
    isVisible = false

    val window = activity.window
    val width = window.decorView.width
    val height = window.decorView.height
    val bitmap = Bitmap.createBitmap(width, height, Bitmap.Config.ARGB_8888)

    // Use PixelCopy to capture the actual rendered window surface
    PixelCopy.request(window, bitmap, { result ->
        if (result == PixelCopy.SUCCESS) {
            // Successfully captured the window surface (including WebView)
            file.value = saveScreenshotToCache(activity, bitmap)
            isVisible = true
            if (file.value!!.exists()) {
                Log.d("Screenshot", "Saved at: ${file.value!!.absolutePath}")
                showConfirmDialog.value = true
            }
        } else {
            // Fallback to old method if PixelCopy fails

            Log.e("Screenshot", "PixelCopy failed: $result")

            val view = activity.window.decorView.rootView

            val fallbackBitmap = createBitmap(width, height)

            val canvas = Canvas(fallbackBitmap)

            view.draw(canvas)

            file.value = saveScreenshotToCache(activity, fallbackBitmap)

            isVisible = true

            if (file.value!!.exists()) {

                Log.d("Screenshot", "Saved at: ${file.value!!.absolutePath}")

                showConfirmDialog.value = true

            }

        }

    }, Handler(Looper.getMainLooper()))

}
```

Summary of Changes

- **Imports added:** `PixelCopy`, `Handler`, `Looper`, `Window`
- **Screenshot capture updated:** replaced `view.draw(canvas)` with `PixelCopy.request()` to capture the rendered surface
- **Fallback included:** uses the previous capture method if PixelCopy fails

Reason

`PixelCopy` captures the actual rendered window surface (similar to a system screenshot), including hardware-accelerated WebView content. This ensures the Capacitor WebView is visible in screenshots.

Note

`PixelCopy` requires Android 7.0+ (API 24+). The fallback ensures compatibility if `PixelCopy` fails on certain devices/configurations.

Final Steps: Build and Test

1. **Sync Gradle**
   - `File → Sync Project with Gradle Files`
2. **Build Project**
   - `Build → Clean Project`
   - `Build → Rebuild Project`
3. **Run App**
   - `Run → Run 'app'`

These steps ensure all configuration changes are applied, dependencies are resolved correctly, and the app is built and launched with the updated Capacitor integration.