1. Create a Flutter Module:

- Use the following command to create a new Flutter module: flutter create --template module my_flutter
- This creates a Flutter module with the necessary structure and configurations.

2. Build the Flutter Module:

- Inside the my_flutter module directory, run: flutter build aar
- This generates an .aar (Android Archive) file, which contains your Flutter code and is ready to be integrated into the Android app.

3. Set Up Repositories in Android Project:

- In the **settings.gradle.kts** of the Android app project, add the path to the Flutter module's repository: pluginManagement {
  repositories {
  google()
  maven { url = uri("/path/to/flutter/module/build/host/outputs/repo") }
  maven { url = uri("https://storage.googleapis.com/download.flutter.io") }
  }
  }

4. Modify build.gradle Files:

- In the **app/build.gradle** of your Android app, add the following to include the Flutter module dependencies: dependencies {
  debugImplementation 'com.example.my_flutter:flutter_debug:1.0'
  releaseImplementation 'com.example.my_flutter:flutter_release:1.0'

}

- Ensure you have the correct Flutter module paths and dependencies configured.

5. Add Flutter Activity to AndroidManifest.xml:

- Inside your **AndroidManifest.xml**, declare the FlutterActivity: <activity
  android:name="io.flutter.embedding.android.FlutterActivity"
  android:exported="true"
  android:theme="@style/Theme.AppCompat.Light.NoActionBar"/>

- This allows the Android app to launch the Flutter UI when needed.

6. Initialize FlutterEngine in the Host App:

- In your
  MainActivity
  of the Android app:
  - Initialize the FlutterEngine and make sure Flutter is ready to launch: lateinit var flutterEngine: FlutterEngine
    override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    flutterEngine = FlutterEngine(this)
    flutterEngine.dartExecutor.executeDartEntrypoint(
    DartExecutor.DartEntrypoint.createDefault()
)
    }

7. Trigger Flutter UI from Native Android:

- Use an Intent to launch the FlutterActivity from your native Android code when needed: val flutterIntent = FlutterActivity.createDefaultIntent(this)
  startActivity(flutterIntent)

8. Run the App:

- Once the integration is done, build and run the Android app.
- Test that your Flutter UI is displayed properly when the FlutterActivity is triggered.

Key Notes:

- **Flutter Module:** The Flutter module acts as a separate codebase that you integrate into your native Android project. It's bundled into .aar format to be included as a dependency.
- **FlutterActivity:** This is the bridge to render Flutter content within the native Android app.
- **Build and Repositories:** You need to build the .aar and ensure your repositories are set up to find the generated dependencies.
- **FlutterEngine:** Manages the Flutter runtime to execute Dart code and render the Flutter UI within the app.

Optional Additional Steps:

- If you need to pass data between your Flutter and Android code, you can use **platform channels**.

- Customize the appearance and behavior of the Flutter UI using themes and other Flutter configurations.

Conclusion:

By following these steps, you've successfully added Flutter to your Android project using the **Add-to-App** approach. This allows you to use Flutter for parts of your app while keeping the native Android code intact.

Feel free to reach out if you need help with further customizations or run into any issues. Happy coding!