Build gradle kts  
  
  
plugins **{** alias(*libs*.*plugins*.*android*.*application*)  
 alias(*libs*.*plugins*.*kotlin*.*android*)  
 alias(*libs*.*plugins*.*kotlin*.*compose*)  
**}***android* **{** namespace = "com.example.myapplication"  
 compileSdk = 35  
  
 defaultConfig **{** applicationId = "com.example.myapplication"  
 minSdk = 24  
 targetSdk = 35  
 versionCode = 1  
 versionName = "1.0"  
  
 testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"  
 **}** buildTypes **{** *release* **{** isMinifyEnabled = false  
 proguardFiles(  
 getDefaultProguardFile("proguard-android-optimize.txt"),  
 "proguard-rules.pro"  
 )  
 **}  
 }** compileOptions **{** sourceCompatibility = JavaVersion.*VERSION\_11* targetCompatibility = JavaVersion.*VERSION\_11* **}** *kotlinOptions* **{** jvmTarget = "11"  
 **}** buildFeatures **{** compose = true  
 **}** composeOptions **{** kotlinCompilerExtensionVersion = "1.5.3"  
 **}  
}***dependencies* **{** // Core dependencies  
 *implementation*(*libs*.*androidx*.*core*.*ktx*)  
 *implementation*(*libs*.*androidx*.*appcompat*)  
 *implementation*(*libs*.*material*)  
 *implementation*(*libs*.*play*.*services*.*location*)  
  
 // Lifecycle & Activity  
 *implementation*(*libs*.*androidx*.*lifecycle*.*runtime*.*ktx*)  
 *implementation*(*libs*.*androidx*.*activity*.*compose*)  
  
 // Jetpack Compose BOM (Manages all Compose versions)  
 *implementation*(platform(*libs*.*androidx*.*compose*.*bom*))  
 *implementation*(*libs*.*androidx*.*ui*)  
 *implementation*(*libs*.*androidx*.*ui*.*tooling*.*preview*)  
 *implementation*(*libs*.*androidx*.*material3*)  
  
 // Testing  
 *testImplementation*(*libs*.*junit*)  
 *androidTestImplementation*(*libs*.*androidx*.*junit*)  
 *androidTestImplementation*(*libs*.*androidx*.*espresso*.*core*)  
 *androidTestImplementation*(platform(*libs*.*androidx*.*compose*.*bom*))  
 *androidTestImplementation*(*libs*.*androidx*.*ui*.*test*.*junit4*)  
 *debugImplementation*(*libs*.*androidx*.*ui*.*tooling*)  
 *debugImplementation*(*libs*.*androidx*.*ui*.*test*.*manifest*)  
**}**

Main activity kt  
  
  
@file:Suppress("DEPRECATION")  
  
package com.example.myapplication  
  
import android.Manifest  
import android.annotation.SuppressLint  
import android.app.NotificationChannel  
import android.app.NotificationManager  
import android.content.Context  
import android.content.Intent  
import android.content.pm.PackageManager  
import android.location.Location  
import android.location.LocationListener  
import android.location.LocationManager  
import android.net.wifi.WifiManager  
import android.os.Build  
import android.os.Bundle  
import android.provider.Settings  
import android.util.Log  
import android.widget.Button  
import android.widget.TextView  
import android.widget.Toast  
import androidx.appcompat.app.AppCompatActivity  
import androidx.core.app.ActivityCompat  
import java.io.OutputStream  
import java.net.Socket  
  
class MainActivity : AppCompatActivity(), LocationListener {  
 private lateinit var wifiManager: WifiManager  
 private lateinit var locationManager: LocationManager  
 private lateinit var textView: TextView  
 private lateinit var btnStartSending: Button  
 private val espHost = "192.168.4.1"  
 private val espPort = 4210  
 private var socket: Socket? = null  
 private var outputStream: OutputStream? = null  
 private var sendingData = false  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setTheme(R.style.*Theme\_MyApplication*)  
 setContentView(R.layout.*activity\_main*)  
  
 textView = findViewById(R.id.*textView*)  
 btnStartSending = findViewById(R.id.*btnStartSending*)  
 wifiManager = *applicationContext*.getSystemService(Context.*WIFI\_SERVICE*) as WifiManager  
 locationManager = getSystemService(Context.*LOCATION\_SERVICE*) as LocationManager  
  
 createNotificationChannel()  
 requestPermissions()  
  
 if (!isGpsEnabled()) {  
 promptEnableGps()  
 }  
  
 enableWiFi()  
  
 btnStartSending.setOnClickListener **{** sendingData = !sendingData  
 if (sendingData) {  
 connectToESP32()  
 startLocationUpdates()  
 startSendingGpsData()  
 btnStartSending.*text* = getString(R.string.*stop\_sending\_gps*)  
 } else {  
 disconnectFromESP32()  
 stopLocationUpdates()  
 btnStartSending.*text* = getString(R.string.*start\_sending\_gps*)  
 }  
 **}** }  
  
 @SuppressLint("SetTextI18n")  
 override fun onLocationChanged(location: Location) {  
 if (!sendingData) return  
  
 val latitude = location.*latitude* val longitude = location.*longitude* val gpsData = "GPS Data|$latitude,$longitude"  
  
 Log.d("GPS\_DEBUG", "GPS Data Received: $latitude, $longitude") // ✅ Debug log to check if phone gets updates  
  
 runOnUiThread **{** textView.*text* = getString(R.string.*location\_prefix*) + " $latitude, $longitude"  
 **}** sendDataToESP(gpsData)  
 }  
  
 @Synchronized  
 private fun connectToESP32() {  
 Thread **{** *synchronized*(this) **{** try {  
 if (socket != null && !socket!!.*isClosed*) {  
 Log.d("ESP\_COMM", "Already connected to ESP32")  
 return@synchronized  
 }  
  
 Log.d("ESP\_COMM", "Attempting connection to ESP32...")  
 socket = Socket(espHost, espPort)  
  
 if (socket!!.*isConnected*) {  
 outputStream = socket!!.getOutputStream()  
 Log.d("ESP\_COMM", "Successfully connected to ESP32 via TCP")  
  
 // Send initial message to keep connection alive  
 sendDataToESP("Hello from Android")  
 } else {  
 Log.e("ESP\_COMM", "Connection failed!")  
 }  
  
 } catch (e: Exception) {  
 Log.e("ESP\_COMM", "Error connecting to ESP32: ${e.message}", e)  
 socket = null  
 outputStream = null  
 }  
 **}  
 }**.start()  
 }  
  
 private fun sendDataToESP(data: String) {  
 Thread **{** try {  
 if (socket == null || socket!!.*isClosed*) {  
 Log.e("ESP\_COMM", "Socket is closed. Reconnecting...")  
 connectToESP32()  
 Thread.sleep(2000) // Wait for reconnection  
 }  
  
 outputStream?.write((data + "\n").*toByteArray*(Charsets.UTF\_8))  
 outputStream?.flush()  
 Log.d("ESP\_COMM", "TCP data sent: $data")  
  
 } catch (e: Exception) {  
 Log.e("ESP\_COMM", "Error sending TCP data: ${e.message}", e)  
 }  
 **}**.start()  
 }  
  
 private fun disconnectFromESP32() {  
 Thread **{** *synchronized*(this) **{** try {  
 outputStream?.close()  
 socket?.close()  
 outputStream = null  
 socket = null  
 Log.d("ESP\_COMM", "Disconnected from ESP32")  
 } catch (e: Exception) {  
 Log.e("ESP\_COMM", "Error disconnecting: ${e.message}", e)  
 }  
 **}  
 }**.start()  
 }  
  
 private fun enableWiFi() {  
 if (!wifiManager.*isWifiEnabled*) {  
 wifiManager.*isWifiEnabled* = true  
 Toast.makeText(this, getString(R.string.*wifi\_enabled*), Toast.*LENGTH\_SHORT*).show()  
 } else {  
 Toast.makeText(this, getString(R.string.*wifi\_already\_on*), Toast.*LENGTH\_SHORT*).show()  
 }  
 }  
  
 private fun requestPermissions() {  
 val permissions = *mutableListOf*(  
 Manifest.permission.*ACCESS\_FINE\_LOCATION*,  
 Manifest.permission.*ACCESS\_COARSE\_LOCATION*,  
 Manifest.permission.*INTERNET*,  
 Manifest.permission.*CHANGE\_NETWORK\_STATE* )  
  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*Q*) {  
 permissions.add(Manifest.permission.*ACCESS\_BACKGROUND\_LOCATION*)  
 }  
  
 ActivityCompat.requestPermissions(this, permissions.*toTypedArray*(), 100)  
 }  
  
 @SuppressLint("MissingPermission")  
 private fun startLocationUpdates() {  
 if (ActivityCompat.checkSelfPermission(this, Manifest.permission.*ACCESS\_FINE\_LOCATION*) == PackageManager.*PERMISSION\_GRANTED*) {  
 locationManager.requestLocationUpdates(LocationManager.*GPS\_PROVIDER*, 2000, 1f, this)  
 Log.d("GPS\_DEBUG", "Started GPS Updates")  
 }  
 }  
  
 private fun stopLocationUpdates() {  
 if (ActivityCompat.checkSelfPermission(  
 this,  
 Manifest.permission.*ACCESS\_FINE\_LOCATION* ) == PackageManager.*PERMISSION\_GRANTED* ) {  
 locationManager.removeUpdates(this)  
 }  
 }  
  
 private fun isGpsEnabled(): Boolean {  
 return locationManager.isProviderEnabled(LocationManager.*GPS\_PROVIDER*)  
 }  
  
 private fun promptEnableGps() {  
 startActivity(Intent(Settings.*ACTION\_LOCATION\_SOURCE\_SETTINGS*))  
 }  
  
 private fun createNotificationChannel() {  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*O*) {  
 val channel = NotificationChannel(  
 "wifi\_channel", "Wi-Fi Alerts",  
 NotificationManager.*IMPORTANCE\_HIGH* ).*apply* **{** *description* = "Notifications for Wi-Fi status changes" **}** val notificationManager = getSystemService(NotificationManager::class.*java*)  
 notificationManager.createNotificationChannel(channel)  
 }  
 }  
  
 override fun onDestroy() {  
 super.onDestroy()  
 stopLocationUpdates()  
 disconnectFromESP32()  
 }  
  
 @SuppressLint("MissingPermission")  
 private fun startSendingGpsData() {  
 Thread **{** while (sendingData) {  
 if (ActivityCompat.checkSelfPermission(  
 this,  
 Manifest.permission.*ACCESS\_FINE\_LOCATION* ) == PackageManager.*PERMISSION\_GRANTED* ) {  
 val location = locationManager.getLastKnownLocation(LocationManager.*GPS\_PROVIDER*)  
  
 if (location != null) {  
 val data = "GPS Data|${location.*latitude*},${location.*longitude*}"  
 sendDataToESP(data)  
 Log.d("ESP\_COMM", "Sent GPS Data: $data")  
 } else {  
 Log.d("GPS\_ERROR", "Location is null")  
 sendDataToESP("KEEP\_ALIVE") // Prevent ESP32 timeout  
 Log.d("ESP\_COMM", "Sent KEEP\_ALIVE message")  
 }  
 } else {  
 Log.e("GPS\_ERROR", "Location permission not granted")  
 }  
  
 Thread.sleep(5000) // Send data every 5 seconds  
 }  
 **}**.start()  
 }  
  
}

Android manifest xml  
  
  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.example.myapplication">  
  
 <!-- Permissions for Wi-Fi scanning -->  
 <uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE"/>  
 <uses-permission android:name="android.permission.CHANGE\_WIFI\_STATE"/>  
  
 <!-- Permissions for Location Services -->  
 <uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION"/>  
 <uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION"/>  
 <uses-permission android:name="android.permission.ACCESS\_BACKGROUND\_LOCATION"/>  
 <uses-permission android:name="android.permission.CHANGE\_NETWORK\_STATE"/>  
 <uses-permission android:name="android.permission.INTERNET"/>  
 <uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE"/>  
 <uses-permission android:name="android.permission.CHANGE\_WIFI\_STATE"/>  
 <uses-permission android:name="android.permission.WRITE\_SETTINGS"/>  
  
 <uses-permission android:name="android.permission.WAKE\_LOCK"/>  
  
 <uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION"/>  
 <uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION"/>  
 <uses-permission android:name="android.permission.ACCESS\_BACKGROUND\_LOCATION"/>  
  
  
 <!-- Permissions for Network Communication -->  
 <uses-permission android:name="android.permission.INTERNET"/>  
 <uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE"/>  
  
 <!-- Permissions for Notifications (Android 13+) -->  
 <uses-permission android:name="android.permission.POST\_NOTIFICATIONS"/>  
  
 <application  
 android:allowBackup="true"  
 android:usesCleartextTraffic="true"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="My Application"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/Theme.MyApplication">  
  
 <activity  
 android:name=".MainActivity"  
 android:exported="true">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN"/>  
 <category android:name="android.intent.category.LAUNCHER"/>  
 </intent-filter>  
 </activity>  
  
 </application>  
  
</manifest>

Activity main xml  
  
  
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:gravity="center"  
 android:padding="16dp">  
  
 <TextView  
 android:id="@+id/textView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="@string/scanning\_wifi\_networks"  
 android:textSize="18sp"  
 android:textStyle="bold"  
 android:gravity="center"  
 android:paddingBottom="20dp" />  
  
 <Button  
 android:id="@+id/btnEnableWifi"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="@string/turn\_on\_wifi"  
 android:padding="10dp"  
 android:layout\_marginBottom="10dp" />  
  
 <Button  
 android:id="@+id/btnStartSending"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="@string/start\_sending\_gps"  
 android:padding="10dp" />  
  
</LinearLayout>

Strings  
  
<resources>  
 <string name="app\_name">MyApplication</string>  
 <string name="scanning\_wifi\_networks">Scanning Wi-Fi Networks…</string>  
 <string name="turn\_on\_wifi">Turn On Wi-Fi</string>  
 <string name="start\_sending\_gps">Start Sending GPS Data</string>  
 <string name="stop\_sending\_gps">Stop Sending GPS Data</string>  
 <string name="wifi\_enabled">Wi-Fi Enabled</string>  
 <string name="wifi\_already\_on">Wi-Fi is already ON</string>  
 <string name="gps\_permission\_required">Permissions required for GPS and Wi-Fi</string>  
 <string name="location\_prefix">Location:</string>  
</resources>

colors xml  
  
  
<?xml version="1.0" encoding="utf-8"?>  
<resources>  
 <color name="purple\_200">#FFBB86FC</color>  
 <color name="purple\_500">#FF6200EE</color>  
 <color name="purple\_700">#FF3700B3</color>  
 <color name="teal\_200">#FF03DAC5</color>  
 <color name="teal\_700">#FF018786</color>  
 <color name="black">#FF000000</color>  
 <color name="white">#FFFFFFFF</color>  
</resources>

themes  
  
  
<resources xmlns:tools="http://schemas.android.com/tools">  
 <style name="Theme.MyApplication" parent="Theme.AppCompat.Light.DarkActionBar">  
 <!-- Customize your theme here -->  
 <item name="colorPrimary">@color/purple\_500</item>  
 <item name="colorPrimaryVariant">@color/purple\_700</item>  
 <item name="colorOnPrimary">@android:color/white</item>  
 <item name="colorSecondary">@color/teal\_200</item>  
 <item name="colorOnSecondary">@android:color/black</item>  
 </style>  
</resources>

launcher background  
  
<?xml version="1.0" encoding="utf-8"?>  
<shape xmlns:android="http://schemas.android.com/apk/res/android"  
 android:shape="rectangle">  
 <solid android:color="#FFFFFF"/> <!-- Change to your preferred background color -->  
</shape>

launcher foreground  
  
<?xml version="1.0" encoding="utf-8"?>  
<vector xmlns:android="http://schemas.android.com/apk/res/android"  
 android:width="108dp"  
 android:height="108dp"  
 android:viewportWidth="108"  
 android:viewportHeight="108">  
 <path  
 android:fillColor="#000000"  
 android:pathData="M54,4A50,50 0,1,1 4,54A50,50 0,1,1 54,4Z" />  
</vector>

wifi xml  
  
<?xml version="1.0" encoding="utf-8"?>  
<vector xmlns:android="http://schemas.android.com/apk/res/android"  
 android:width="24dp"  
 android:height="24dp"  
 android:viewportWidth="24"  
 android:viewportHeight="24"  
 android:tint="?attr/colorPrimary">  
  
 <!-- Wi-Fi Signal Curves -->  
 <path  
 android:fillColor="@android:color/white"  
 android:pathData="M2,8C6,4 18,4 22,8L20,10C16,6 8,6 4,10Z" />  
  
 <path  
 android:fillColor="@android:color/white"  
 android:pathData="M5,12C9,8 15,8 19,12L17,14C14,11 10,11 7,14Z" />  
  
 <path  
 android:fillColor="@android:color/white"  
 android:pathData="M8,16C10,14 14,14 16,16L14,18C13,17 11,17 10,18Z" />  
  
 <!-- Wi-Fi Dot (Strong Signal Indicator) -->  
 <path  
 android:fillColor="@android:color/white"  
 android:pathData="M11,19h2v2h-2z" />  
</vector>

esp32 code  
  
#include <WiFi.h>

const char\* ssid = "ESP32\_AP";

const char\* password = "12345678";

WiFiServer server(4210);

void setup() {

    Serial.begin(115200);

    WiFi.softAP(ssid, password);

    Serial.println("ESP32 AP Started");

    server.begin();

}

void loop() {

    WiFiClient client = server.available();

    if (client) {

        Serial.println("Client connected");

        unsigned long lastDataTime = millis(); // Track last received data time

        while (client.connected()) {

            if (client.available()) {

                String data = client.readStringUntil('\n');

                Serial.println("Received: " + data);

                client.println("ACK: " + data);

                lastDataTime = millis(); // Reset timeout

            }

            // Disconnect client if no data is received for 10 seconds

            if (millis() - lastDataTime > 10000) {

                Serial.println("Client timeout, disconnecting...");

                break;

            }

        }

        client.stop();

        Serial.println("Client disconnected");

    }

}