**Emergency Alert System for Organizations**

**Scenario:**

A company wants an Android app for employees that can send pre-defined emergency SMS alerts to a group of contacts and automatically dial a specific number in case of an emergency, such as a fire or medical crisis. The app should also provide an option to detect the network status before sending the alert.

**Requirements:**

1. **Send SMS**: Send a pre-configured emergency message to a list of contacts.
2. **Make a Call**: Automatically dial an emergency contact number when a button is pressed.
3. **Check Network Status**: Ensure the device has network connectivity before sending SMS or making calls.
4. **Permissions Handling**: Prompt the user to grant necessary permissions (SMS, Call, Network State).
5. **Simple UI**: A single screen with buttons to trigger the SMS and call actions.

**Step-by-Step Implementation:**

**1. Setup Project**

* Create a new Android project in Android Studio.
* Use **Java** as the programming language.

**2. Permissions in AndroidManifest.xml**

xml

Copy code

<uses-feature  
 android:name="android.hardware.telephony"  
 android:required="false" />

<uses-permission android:name="android.permission.SEND\_SMS" />

<uses-permission android:name="android.permission.CALL\_PHONE" />

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />

**3. Layout (activity\_main.xml)**

xml

Copy code

<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:padding="16dp">

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Emergency Alert System"

android:textSize="18sp"

android:gravity="center"

android:padding="8dp" />

<Button

android:id="@+id/btnSendSms"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Send Emergency SMS" />

<Button

android:id="@+id/btnMakeCall"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Call Emergency Contact"

android:layout\_marginTop="16dp" />

</LinearLayout>

**4. Java Code (MainActivity.java)**

java

Copy code

package com.example.emergencyalert;

import android.Manifest;

import android.content.Intent;

import android.content.pm.PackageManager;

import android.net.ConnectivityManager;

import android.net.NetworkInfo;

import android.os.Bundle;

import android.telephony.SmsManager;

import android.view.View;

import android.widget.Button;

import android.widget.Toast;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.app.ActivityCompat;

import androidx.core.content.ContextCompat;

public class MainActivity extends AppCompatActivity {

private static final int PERMISSION\_REQUEST\_CODE = 101;

private String emergencyMessage = "Emergency! Please help. Location: Office.";

private String emergencyContact = "+1234567890";

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

Button btnSendSms = findViewById(R.id.btnSendSms);

Button btnMakeCall = findViewById(R.id.btnMakeCall);

btnSendSms.setOnClickListener(view -> sendEmergencySms());

btnMakeCall.setOnClickListener(view -> makeEmergencyCall());

}

private void sendEmergencySms() {

if (checkPermissions(Manifest.permission.SEND\_SMS)) {

if (isNetworkAvailable()) {

SmsManager smsManager = SmsManager.getDefault();

smsManager.sendTextMessage(emergencyContact, null, emergencyMessage, null, null);

Toast.makeText(this, "Emergency SMS sent!", Toast.LENGTH\_SHORT).show();

} else {

Toast.makeText(this, "No network available!", Toast.LENGTH\_SHORT).show();

}

} else {

requestPermissions(new String[]{Manifest.permission.SEND\_SMS});

}

}

private void makeEmergencyCall() {

if (checkPermissions(Manifest.permission.CALL\_PHONE)) {

Intent callIntent = new Intent(Intent.ACTION\_CALL);

callIntent.setData(android.net.Uri.parse("tel:" + emergencyContact));

startActivity(callIntent);

} else {

requestPermissions(new String[]{Manifest.permission.CALL\_PHONE});

}

}

private boolean isNetworkAvailable() {

ConnectivityManager cm = (ConnectivityManager) getSystemService(CONNECTIVITY\_SERVICE);

NetworkInfo activeNetwork = cm.getActiveNetworkInfo();

return activeNetwork != null && activeNetwork.isConnected();

}

private boolean checkPermissions(String permission) {

return ContextCompat.checkSelfPermission(this, permission) == PackageManager.PERMISSION\_GRANTED;

}

private void requestPermissions(String[] permissions) {

ActivityCompat.requestPermissions(this, permissions, PERMISSION\_REQUEST\_CODE);

}

@Override

public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {

if (requestCode == PERMISSION\_REQUEST\_CODE) {

if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION\_GRANTED) {

Toast.makeText(this, "Permission granted!", Toast.LENGTH\_SHORT).show();

} else {

Toast.makeText(this, "Permission denied!", Toast.LENGTH\_SHORT).show();

}

}

}

}

**Key Features:**

1. **Emergency SMS**: Sends a pre-configured message using SmsManager.
2. **Emergency Call**: Initiates a call using Intent.ACTION\_CALL.
3. **Network Check**: Ensures network connectivity using ConnectivityManager.
4. **Runtime Permissions**: Dynamically requests and handles permissions for SMS and call features.

**Testing the App:**

1. **SMS**:  
   Use a test device or emulator with proper configuration (SIM and network availability).
2. **Call**:  
   Test on a physical device. The emulator does not support direct calls.
3. **Permissions**:  
   Deny permissions and verify if the app requests them again.

This is a foundational example for building more complex telephony-based apps, like group alerts or location-based emergency systems. Let me know if you’d like to extend it further