App Name: MySMS

Introduction:

An app that receives SMS messages and displays the message body in a toast message within the app.

Requirements:
1. The app should be compatible with Android devices running on API level 21 (Android 5.0) and above.
2. Upon receiving an SMS, the app should display a toast message showing the content of the received SMS.
3. Implement a user-friendly interface to showcase the received messages.
4. Handle potential permissions and security considerations related to SMS access.
5. Ensure proper error handling and edge case scenarios.
6. The app should be stable, efficient, and well-optimized for performance.
7. Properly handle any potential exceptions and crashes.

What I am Delivering:

1. Full source code of the Android application.
2. A brief documentation of how the code works, including an explanation of any significant design choices or considerations.
3. APK file for testing the app on various devices

Process:

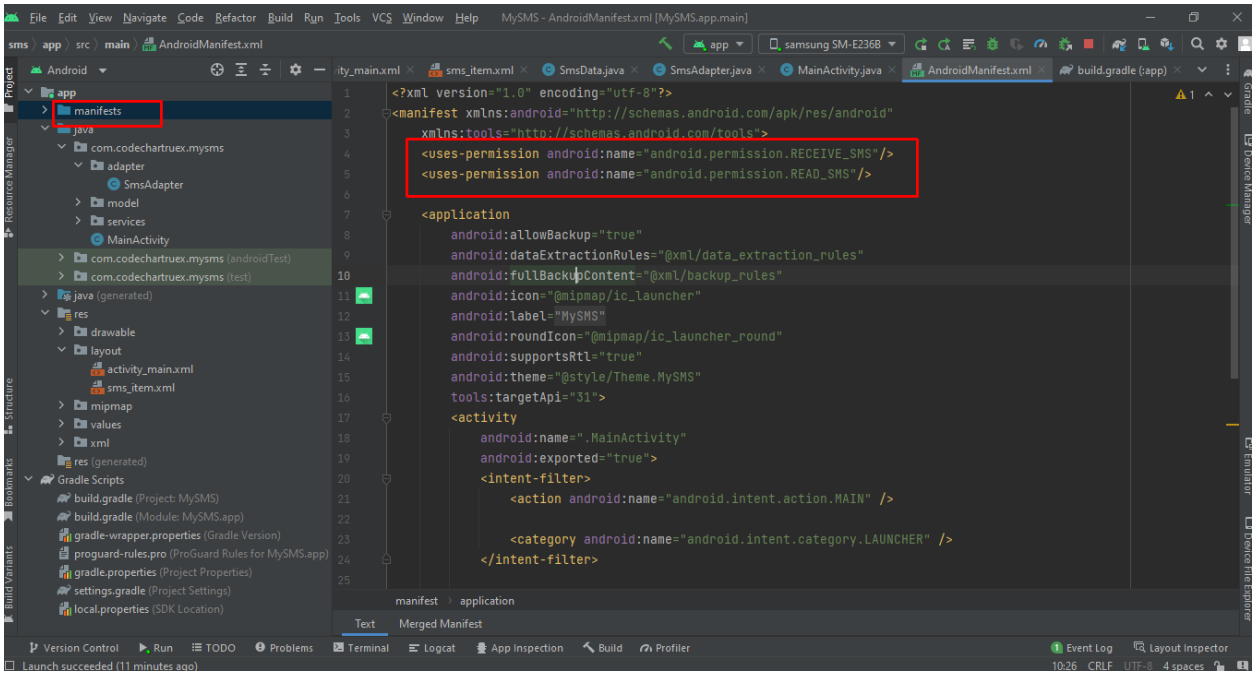
Android app that receives SMS messages and displays the message body in a toast message within the app involves several steps. Below is a step-by-step guide on how to implement this functionality using Java in Android:

1. **Create a New Android Project:**
Start by creating a new Android project in Android Studio

2. Request SMS Permissions:
Adding the following permissions to AndroidManifest.xml file to request permission to read SMS messages:

Android Manifest.xml in android studio project:

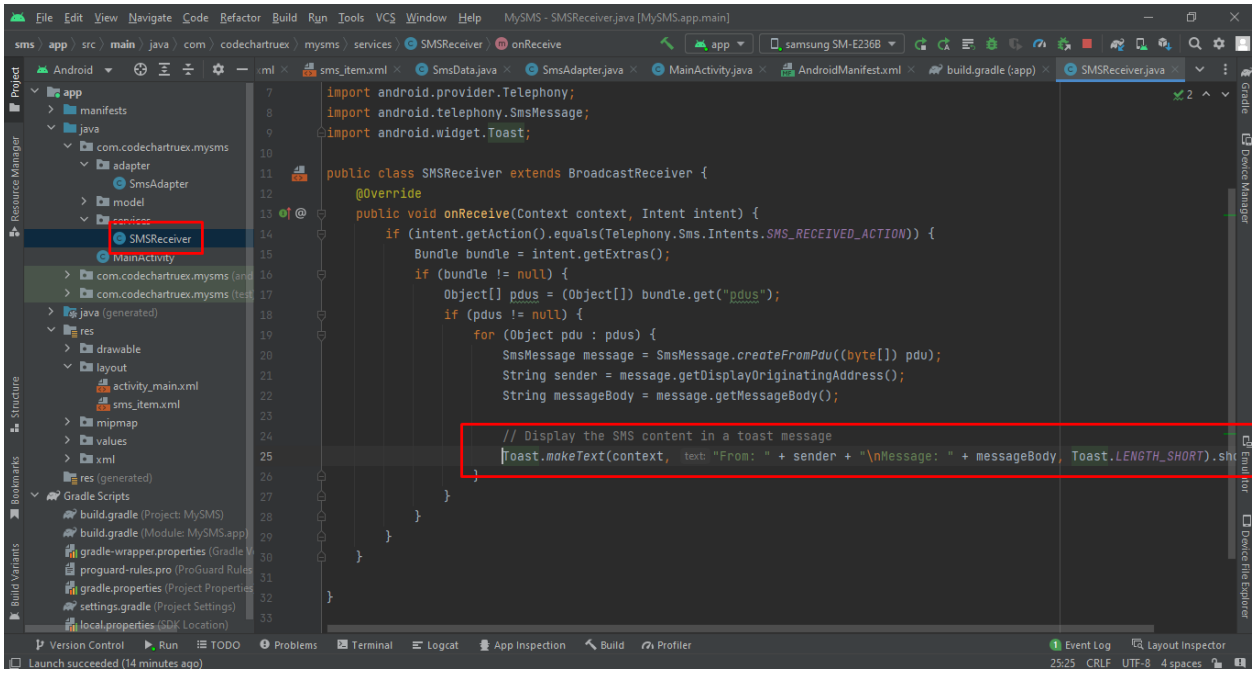
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
<uses-permission android:name="android.permission.READ_SMS"/>



We also need to request runtime permissions from the user, as Android 6.0 and above require runtime permission requests. We can use the ActivityCompat.requestPermissions method to request these permissions.

3. Create a Broadcast Receiver:

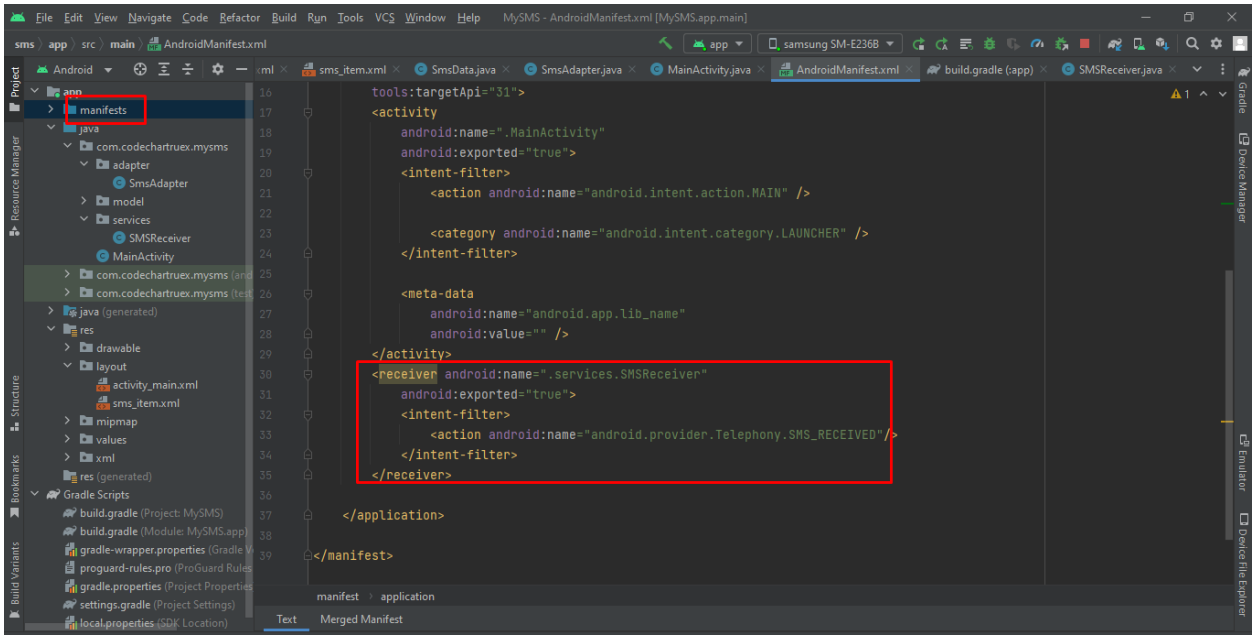
Create a BroadcastReceiver to listen for incoming SMS messages. And a Java class that extends BroadcastReceiver:



```
import android.provider.Telephony;
import android.telephony.SmsMessage;
import android.widget.Toast;

public class SMSReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        if (intent.getAction().equals(Telephony.Sms.Intents.SMS_RECEIVED_ACTION)) {
            Bundle bundle = intent.getExtras();
            if (bundle != null) {
                Object[] pdus = (Object[]) bundle.get("pdus");
                if (pdus != null) {
                    for (Object pdu : pdus) {
                        SmsMessage message = SmsMessage.createFromPdu((byte[]) pdu);
                        String sender = message.getDisplayOriginatingAddress();
                        String messageBody = message.getMessageBody();

                        // Display the SMS content in a toast message
                        Toast.makeText(context,  text: "From: " + sender + "\nMessage: " + messageBody, Toast.LENGTH_SHORT).sh
                    }
                }
            }
        }

}
```

4.Register the BroadcastReceiver:

In Our AndroidManifest.xml file, register the BroadcastReceiver to listen for SMS messages. Add the following code within the <application> section:



```
        tools:targetApi="31">
        <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>

            <meta-data
                android:name="android.app.lib_name"
                android:value="" />
        </activity>
        <receiver android:name=".services.SMSReceiver"
            android:exported="true">
            <intent-filter>
                <action android:name="android.provider.Telephony.SMS_RECEIVED" />
            </intent-filter>
        </receiver>

    </application>

</manifest>
```

Query and Display Existing SMS Messages

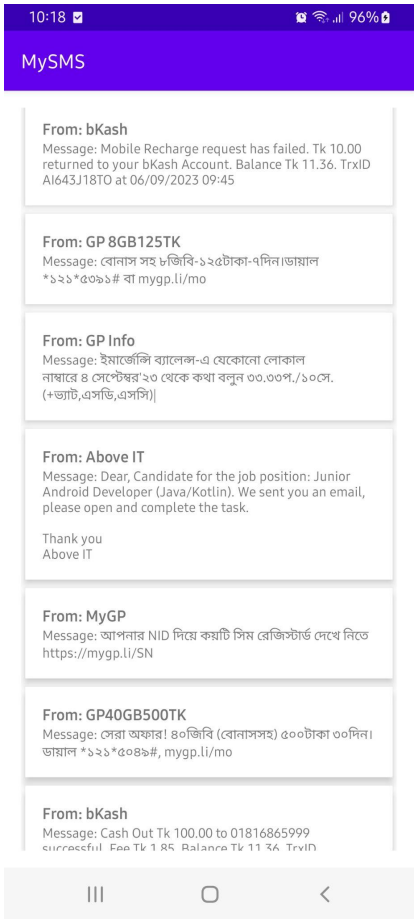
In the onCreate method of our MainActivity, query the SMS content provider to retrieve existing SMS messages and add them to the smsList:

```
if (checkSmsPermission()) {
    registerSmsReceiver();
    loadExistingSmsMessages(); // Add this line
} else {
    requestSmsPermission();
}
```

Create a method called loadExistingSmsMessages to query and load the existing messages:

```
private void loadExistingSmsMessages() {
    // Query the SMS content provider to retrieve existing SMS messages
    Cursor cursor = getContentResolver().query(
        Uri.parse("content://sms/inbox"),
        new String[]{"_id", "address", "body", "date"},
        null,
        null,
        null
    );

    if (cursor != null && cursor.moveToFirst()) {
        do {
            String sender = cursor.getString(cursor.getColumnIndex("address"));
            String messageBody = cursor.getString(cursor.getColumnIndex("body"));
            SmsData smsData = new SmsData(sender, messageBody);
            smsList.add(smsData);
        } while (cursor.moveToNext());
        cursor.close();
        smsAdapter.notifyDataSetChanged();
    }
}
```

App Look after Install:



Problem Facing while testing:

Old sms is only but when new incoming sms is coming its not responding but i noticed the sms on my default sms app.

Solution:
If new incoming SMS messages are not showing in the notification bar or toast messages, it's likely because our default SMS app is intercepting and handling incoming SMS messages. Android allows only one app at a time to be the default SMS app for handling incoming messages.

To receive new SMS messages in our app, we will need to do the following:

Remove Default SMS App Setting:
Make sure that our app is not set as the default SMS app on our device. If it is set as the default SMS app, Android will route incoming SMS messages to our app and not show them in the notification bar or default SMS app.

Check Default SMS App:

To check which app is currently set as the default SMS app

Call method in onCreate to check if our app is currently set as the default SMS app. If it returns true, our app is the default SMS app and will receive incoming SMS.

```
private boolean isDefaultSmsApp() {
    return getPackageName().equals(Telephony.Sms.getDefaultSmsPackage(this));
}
```

Prompt to Change Default SMS App:

prompt the user to change it by launching the default SMS app's settings page:

```
private void promptToChangeDefaultSmsApp() {
    Intent intent = new Intent(Telephony.Sms.Intents.ACTION_CHANGE_DEFAULT);
    intent.putExtra(Telephony.Sms.Intents.EXTRA_PACKAGE_NAME,
getPackageName());
    startActivity(intent);
}
```

We can call promptToChangeDefaultSmsApp() when our app detects that it is set as the default SMS app. This way, we give the user the option to change the default SMS app to a different one.

handling SMS messages requires careful handling of permissions, and we should follow Android's guidelines on SMS handling, especially when it comes to changing the default SMS app.

How i handle :

Error Handling and Edge Cases:
null checks and permission checks, to make our app robust and stable.

Optimization and Efficiency:
using a background service to process incoming SMS messages

Exception Handling:

Handle any potential exceptions and crashes gracefully by implementing try-catch blocks and logging.

Thanks For Reading the documentation —MD.Rakibul hasan