Ex. No. 5	Exploring Built-in Content Providers in Android Development
Date of Exercise	10 - 09 - 2024

Aim

The aim of this experiment is to make a message accessing application that uses content providers to display, add, update and delete contents in android.

Program

MainActivity.kt

```
package com.example.exp_5
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import android.Manifest
import android.annotation.SuppressLint
import android.content.pm.PackageManager
import android.provider.Telephony
import android.telephony.SmsManager
import android.widget.ArrayAdapter
import android.widget.Button
import android.widget.EditText
import android.widget.ListView
import android.widget.Toast
import androidx.appcompat.app.AlertDialog
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat
class MainActivity : AppCompatActivity() {
    private val REQUEST_SMS_PERMISSION = 123
    private lateinit var smsList: ArrayList<String>
```

```
private lateinit var smsIds: ArrayList<String>
private lateinit var adapter: ArrayAdapter<String>

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    smsList = ArrayList()
    smsIds = ArrayList()

    if (ContextCompat.checkSelfPermission(this, Manifest.permission.READ_SMS)
        != PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions(
            this,
            arrayOf(Manifest.permission.READ_SMS, Manifest.permission.SEND_SMS),
            REQUEST_SMS_PERMISSION
        )
    }

    findViewById<Button>(R.id.button_load_sms).setOnClickListener {
        loadSmsMessages()
    }

    findViewById<Button>(R.id.button_add_sms).setOnClickListener {
        showSendSmsDialog()
    }
}

private fun loadSmsMessages() {
    smsList.clear()
    smsIds.clear()
```

```
val uri = Telephony.Sms.CONTENT_URI
val cursor = contentResolver.query(uri, null, null, null, null)

if (cursor != null && cursor.moveToFirst()) {
    val indexBody = cursor.getColumnIndex(Telephony.Sms.BODY)
    val indexAddress = cursor.getColumnIndex(Telephony.Sms.ADDRESS)
    val indexId = cursor.getColumnIndex(Telephony.Sms._ID)

    do {
        val body = cursor.getString(indexBody)
        val address = cursor.getString(indexAddress)
        val id = cursor.getString(indexId)
        smsList.add("From: $address\nMessage: $body")
        smsIds.add(id)
    } while (cursor.moveToNext())

    cursor.close()
}

adapter = ArrayAdapter(this, android.R.layout.simple_list_item_1, smsList)
val listView = findViewById<ListView>(R.id.list_view_sms)
listView.adapter = adapter

listView.setOnItemLongClickListener { _, _, position, _ ->
    smsList.removeAt(position)
    smsIds.removeAt(position)
    adapter.notifyDataSetChanged()
    Toast.makeText(this, "Message removed", Toast.LENGTH_SHORT).show()
    true
}
```

```
    }

    private fun sendSmsMessage(phoneNumber: String, message: String) {
        val smsManager = SmsManager.getDefault()
        smsManager.sendTextMessage(phoneNumber, null, message, null, null)
        Toast.makeText(this, "SMS Sent", Toast.LENGTH_SHORT).show()
    }

    @SuppressLint("MissingInflatedId")
    private fun showSendSmsDialog() {
        val builder = AlertDialog.Builder(this)
        builder.setTitle("Send SMS")

        val dialogView = layoutInflater.inflate(R.layout.dialog_send_sms, null)
        builder.setView(dialogView)

        val phoneNumberEditText =
dialogView.findViewById<EditText>(R.id.edit_text_phone_number)
        val messageEditText = dialogView.findViewById<EditText>(R.id.edit_text_message)

        builder.setPositiveButton("Send") { _, _ ->
            val phoneNumber = phoneNumberEditText.text.toString()
            val message = messageEditText.text.toString()

            if (phoneNumber.isNotEmpty() && message.isNotEmpty()) {
                sendSmsMessage(phoneNumber, message)
            } else {
                Toast.makeText(this, "Phone number and message cannot be empty",
Toast.LENGTH_SHORT).show()
            }
        }
```

```
        builder.setNegativeButton("Cancel") { dialog, _ ->
            dialog.dismiss()
        }

        builder.create().show()
    }

    override fun onRequestPermissionsResult(requestCode: Int, permissions: Array<out
String>, grantResults: IntArray) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults)

        if (requestCode == REQUEST_SMS_PERMISSION && grantResults.isNotEmpty()
&& grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            loadSmsMessages()
        } else {
            Toast.makeText(this, "SMS Permission Denied", Toast.LENGTH_SHORT).show()
        }
    }
}
```

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:padding="16dp">
    <Button
        android:id="@+id/button_load_sms"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Load SMS" />

    <Button
        android:id="@+id/button_add_sms"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Send SMS" />


    <ListView
        android:id="@+id/list_view_sms"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</LinearLayout>
```

dialogue_send_sms.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="wrap_content"
    android:orientation="vertical"
    android:padding="16dp">

    <EditText
        android:id="@+id/edit_text_phone_number"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

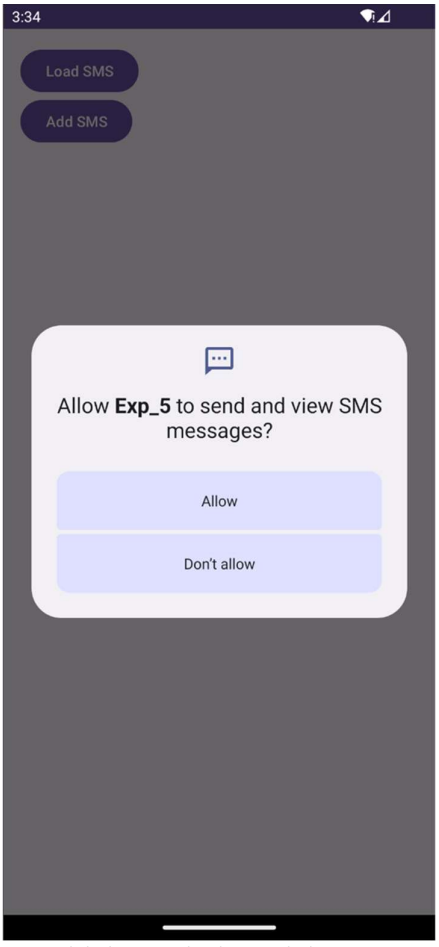
```
        android:hint="Phone Number"
        android:inputType="phone" />

    <EditText
        android:id="@+id/edit_text_message"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Message"
        android:inputType="textMultiLine"
        android:lines="4"
        android:maxLines="5" />
</LinearLayout>
```

AndroidManifest.xml

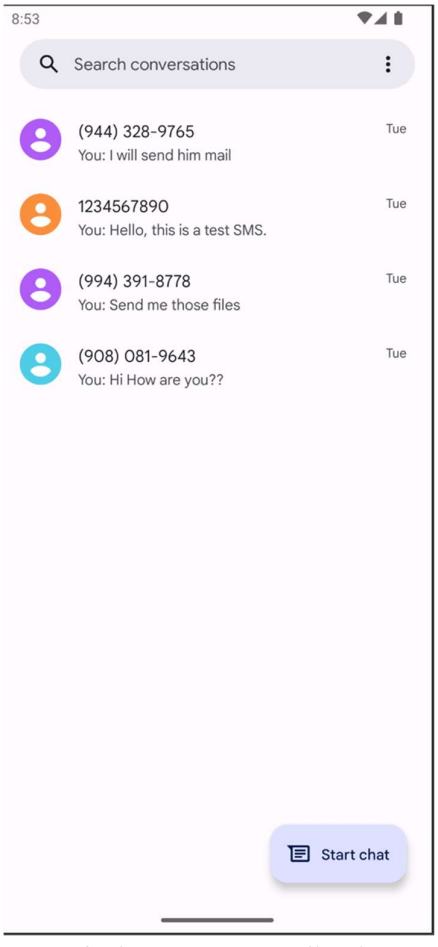
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <uses-permission android:name="android.permission.READ_SMS"/>
    <uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-permission android:name="android.permission.WRITE_SMS"/>
    <uses-feature android:name="android.hardware.telephony" android:required="true"/>
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Exp_5"
```

```
        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>
        </activity>
    </application>
</manifest>
```

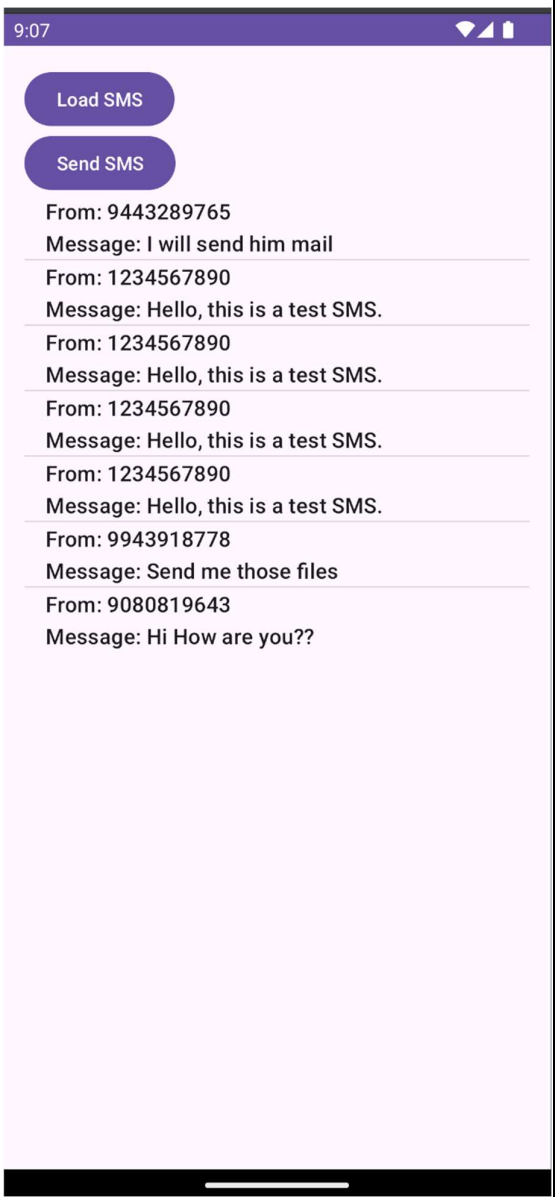
Output Screenshots



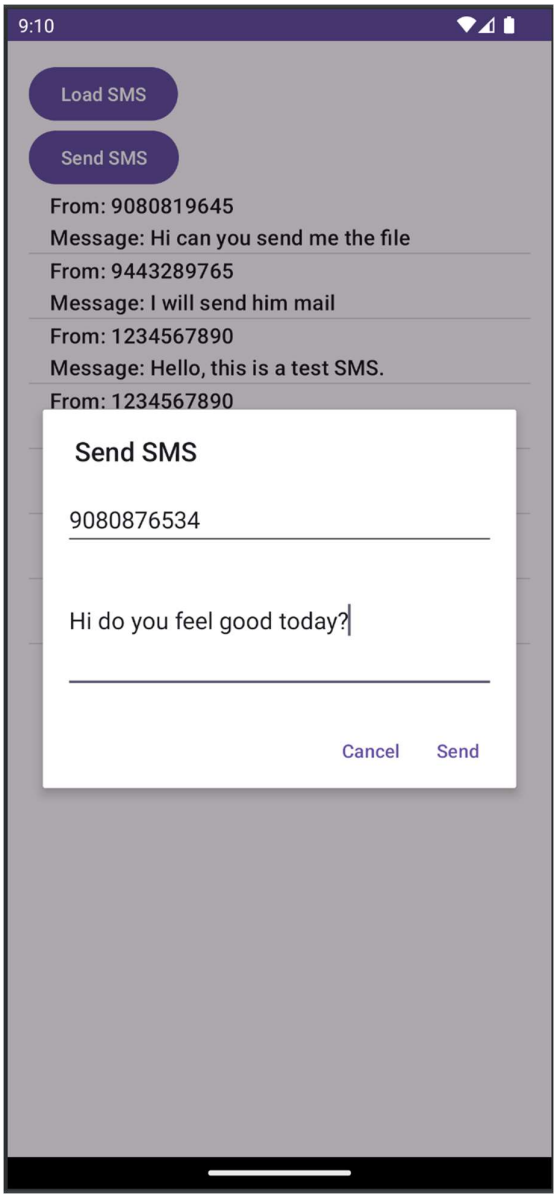
Initial Permission Dialogue



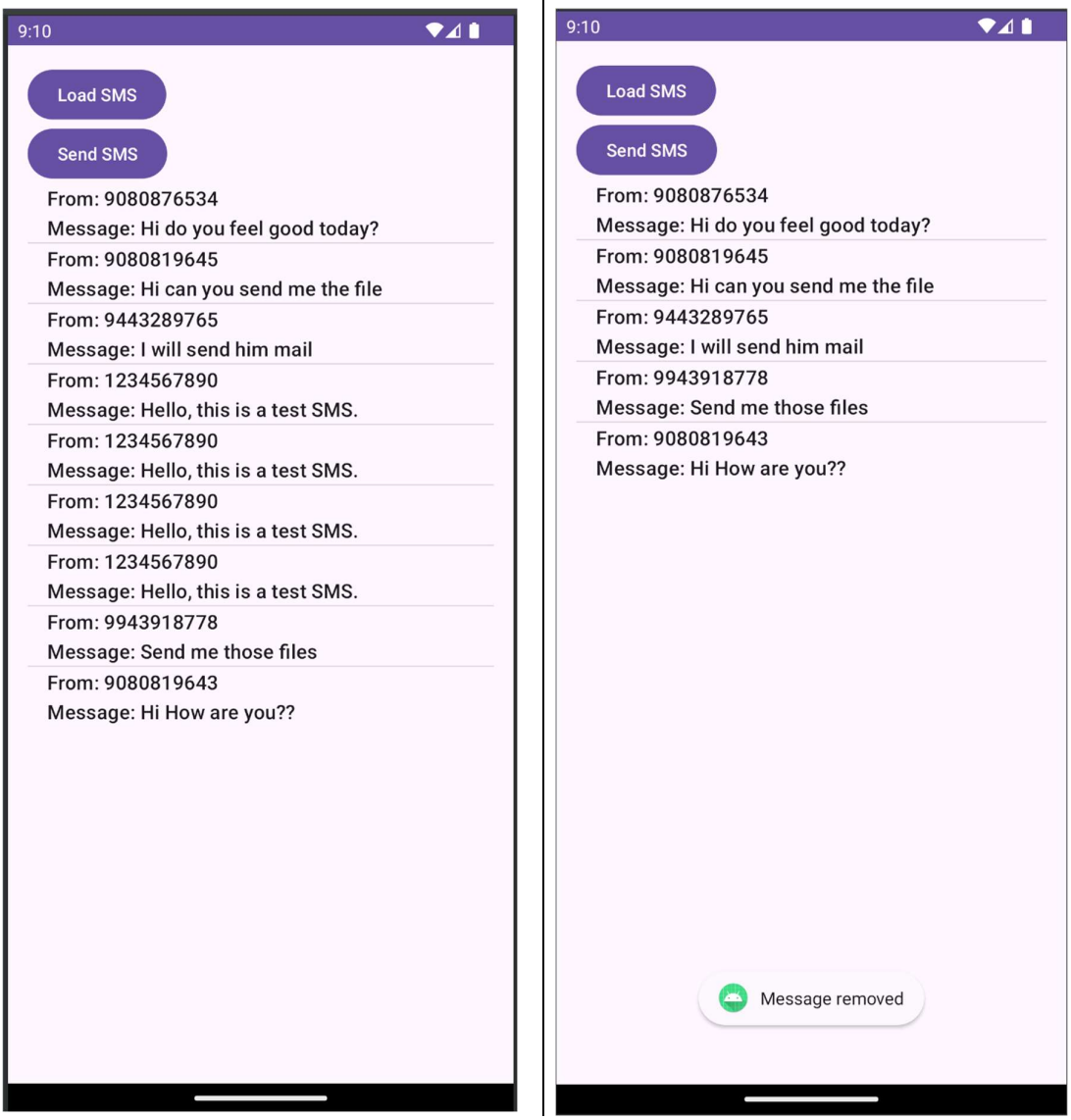
Default Message Application



Loading the Messages



Sending Message



Removing by Long Pressing

Result

Thus, an android application was developed to access messages, to display them, to send them and to update them using the content providers and the same was verified successfully.