

**BỘ NÔNG NGHIỆP VÀ MÔI TRƯỜNG
TRƯỜNG ĐẠI HỌC THỦY LỢI**



BÀI TẬP THỰC HÀNH SỐ 5
PHÁT TRIỂN ÚNG DỤNG CHO THIẾT BỊ DI ĐỘNG
NỘI DUNG BỔ SUNG: ÚNG DỤNG VỚI CHỦ ĐỀ NÂNG CAO

STT	Mã sinh viên	Họ và tên	Lớp
1	2251061739	Phạm Tiến Đạt	64CNTT1

Hà Nội, năm 2025

BÀI TẬP 1: Content Providers

Mục tiêu:

- Tìm hiểu cách sử dụng Content Providers để truy cập dữ liệu từ ứng dụng khác (ứng dụng Danh bạ).
- Hiển thị danh sách tên các liên hệ trong danh bạ lên ứng dụng của mình.

Các bước thực hiện:

1. Thiết lập quyền truy cập:

- Mở file `AndroidManifest.xml` của ứng dụng.
- Thêm quyền `READ_CONTACTS` để xin phép ứng dụng được đọc dữ liệu danh bạ.

XML

```
<uses-permission android:name="android.permission.READ_CONTACTS" />
```

2. Thiết kế giao diện (layout):

- Tạo một `ListView` trong file layout (ví dụ: `activity_main.xml`) để hiển thị danh sách tên liên hệ.

XML

```
<ListView  
    android:id="@+id/listViewContacts"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent" />
```

3. Đọc dữ liệu từ Content Provider:

- Trong Activity chính (ví dụ: `MainActivity.kt`), sử dụng `ContentResolver` để truy vấn dữ liệu từ Content Provider của ứng dụng Danh bạ.
- URI của Content Provider Danh bạ là:
`ContactsContract.Contacts.CONTENT_URI`
- Sử dụng phương thức `query()` của `ContentResolver` để lấy dữ liệu. Phương thức này trả về một `Cursor` chứa kết quả truy vấn.
- Duyệt `Cursor` để lấy tên của từng liên hệ và lưu vào một `ArrayList<String>`.

4. Hiển thị dữ liệu lên ListView:

- Tạo một `ArrayAdapter<String>` để đưa dữ liệu từ `ArrayList<String>` lên `ListView`.
- Gán `ArrayAdapter<String>` cho `ListView`.

Code ví dụ (`MainActivity.kt`):

Kotlin

```
import android.Manifest  
import android.content.pm.PackageManager  
import android.database.Cursor  
import android.os.Bundle  
import android.provider.ContactsContract  
import android.widget.ArrayAdapter  
import android.widget.ListView  
import androidx.appcompat.app.AppCompatActivity  
import androidx.core.app.ActivityCompat  
import androidx.core.content.ContextCompat
```

```

class MainActivity : AppCompatActivity() {

    private lateinit var listViewContacts: ListView
    private lateinit var contactsList: ArrayList<String>
    private lateinit var contactsAdapter: ArrayAdapter<String>

    private val REQUEST_READ_CONTACTS_PERMISSION = 100

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

        listViewContacts = findViewById(R.id.listViewContacts)
        contactsList = ArrayList()

        // Kiểm tra và xin quyền READ_CONTACTS
        if (ContextCompat.checkSelfPermission(
                this,
                Manifest.permission.READ_CONTACTS
            ) != PackageManager.PERMISSION_GRANTED
        ) {
            ActivityCompat.requestPermissions(
                this,
                arrayOf(Manifest.permission.READ_CONTACTS),
                REQUEST_READ_CONTACTS_PERMISSION
            )
        } else {
            loadContacts()
        }
    }

    override fun onRequestPermissionsResult(
        requestCode: Int,
        permissions: Array<String>,
        grantResults: IntArray
    ) {
        super.onRequestPermissionsResult(requestCode, permissions,
        grantResults)
        if (requestCode == REQUEST_READ_CONTACTS_PERMISSION) {
            if (grantResults.isNotEmpty() && grantResults[0] ==
            PackageManager.PERMISSION_GRANTED) {
                loadContacts()
            } else {
                // Xử lý trường hợp người dùng từ chối cấp quyền
                // Ví dụ: Hiển thị thông báo cho người dùng
                // Giải thích lý do cần quyền truy cập danh bạ
                Toast.makeText(this, "Permission denied",
                Toast.LENGTH_SHORT).show()
            }
        }
    }

    private fun loadContacts() {
        // Lấy dữ liệu từ Content Provider
        val cursor: Cursor? = contentResolver.query(
            ContactsContract.Contacts.CONTENT_URI,
            null,
            null,
            null,
            null
        )

        cursor?.use {

```

```

        if (it.count > 0) {
            while (it.moveToNext()) {
                val nameIndex =
                    it.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME)
                val name = it.getString(nameIndex)
                contactsList.add(name)
            }
        }
    }

    // Hiển thị dữ liệu lên ListView
    contactsAdapter = ArrayAdapter(this,
        android.R.layout.simple_list_item_1, contactsList)
    listViewContacts.adapter = contactsAdapter
}
}

```

Giải thích chi tiết (Kotlin):

- `Manifest.permission.READ_CONTACTS`: Khai báo quyền truy cập danh bạ trong `AndroidManifest.xml`.
- `ContextCompat.checkSelfPermission()`: Kiểm tra xem ứng dụng đã được cấp quyền `READ_CONTACTS` hay chưa.
- `ActivityCompat.requestPermissions()`: Xin quyền `READ_CONTACTS` từ người dùng nếu chưa được cấp.
- `onRequestPermissionsResult()`: Xử lý kết quả trả về khi người dùng cấp hoặc từ chối quyền.
- `contentResolver`: Đối tượng `ContentResolver` cho phép ứng dụng tương tác với Content Providers.
- `ContactsContract.Contacts.CONTENT_URI`: URI của Content Provider chứa dữ liệu về các liên hệ.
- `Cursor`: Một interface đại diện cho tập kết quả của một truy vấn cơ sở dữ liệu. Trong trường hợp này, nó chứa dữ liệu từ Content Provider.
- `cursor?.use {}`: Sử dụng `use` block để tự động đóng `Cursor` sau khi sử dụng, tránh rò rỉ tài nguyên.
- `it.count`: Lấy số lượng hàng trong `Cursor`.
- `it.moveToNext()`: Di chuyển đến hàng tiếp theo trong `Cursor`.
- `it.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME)`: Lấy chỉ số của cột chứa tên hiển thị của liên hệ.
- `it.getString(nameIndex)`: Lấy giá trị kiểu String từ cột tại chỉ số đã cho.
- `ArrayAdapter<String>`: Adapter để hiển thị danh sách các chuỗi (tên liên hệ) lên `ListView`.
- `listViewContacts.adapter`: Gán `ArrayAdapter` cho `ListView` để hiển thị dữ liệu.

Hướng dẫn Bài tập 01: Chụp lại mà hình từng bước thực hiện (tương tự cho các Bài tập bên dưới) để học sinh lớp 10 có thể thực hiện lại theo được :D

1. Thiết lập quyền truy cập:

- Mở file `AndroidManifest.xml` của ứng dụng.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="Content Providers"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.ContentProviders"
        tools:targetApi="31">

        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="Content Providers"
            android:theme="@style/Theme.ContentProviders">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        
```

- Thêm quyền READ_CONTACTS để xin phép ứng dụng được đọc dữ liệu danh bạ.

XML

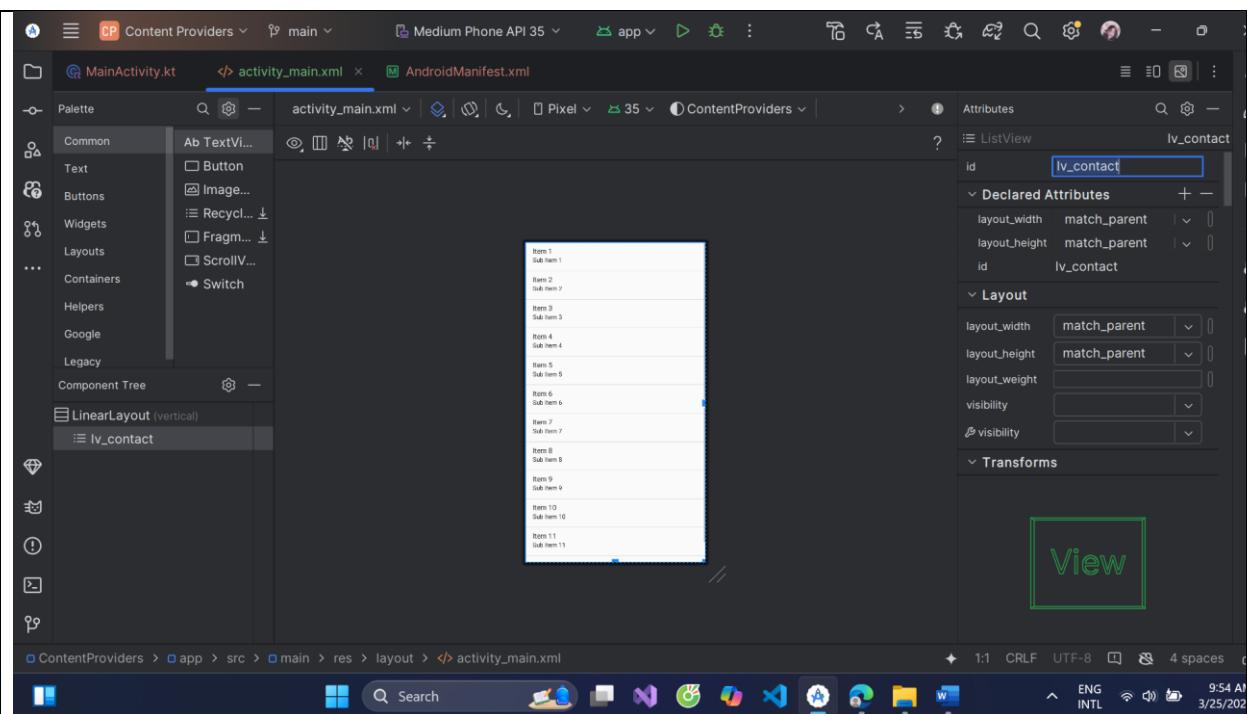
```
<uses-permission android:name="android.permission.READ_CONTACTS" />
```

```
<?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_CONTACTS" />
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="Content Providers"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.ContentProviders"
        tools:targetApi="31">

        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="Content Providers"
            android:theme="@style/Theme.ContentProviders">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        
```

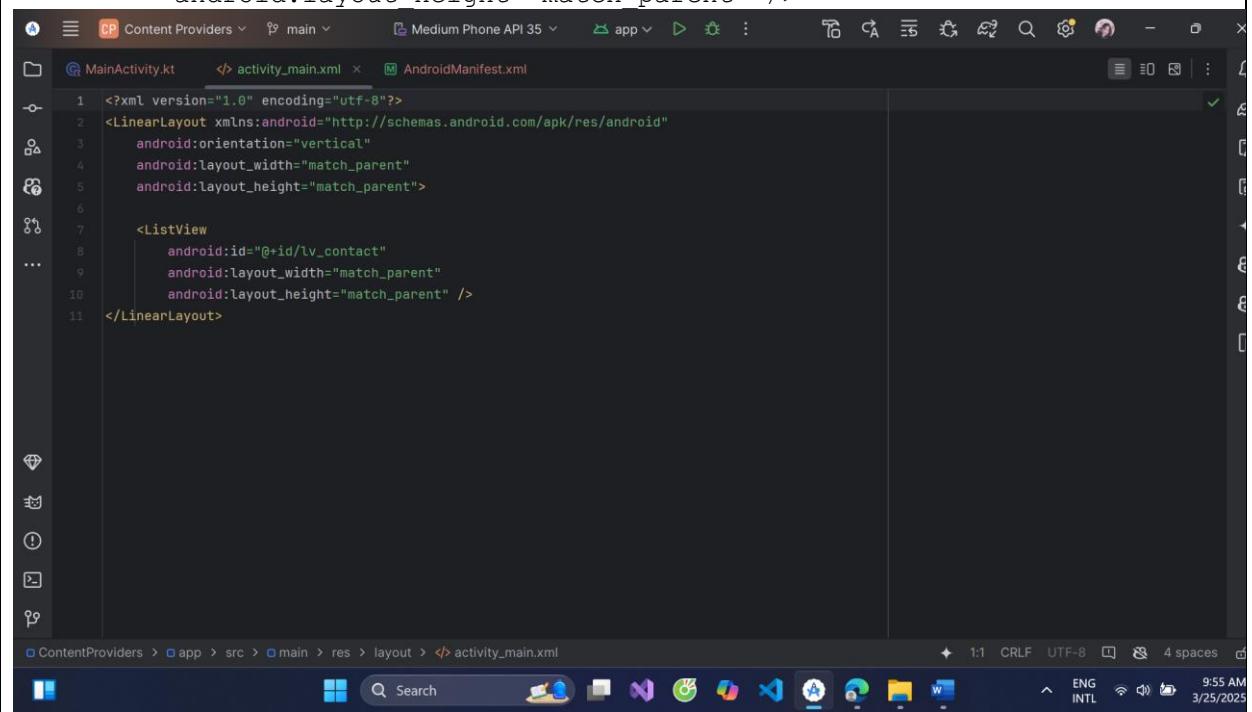
2. Thiết kế giao diện (layout):

- Tạo một ListView trong file layout (ví dụ: activity_main.xml) để hiển thị danh sách tên liên hệ.



XML

```
<ListView
    android:id="@+id/listViewContacts"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```



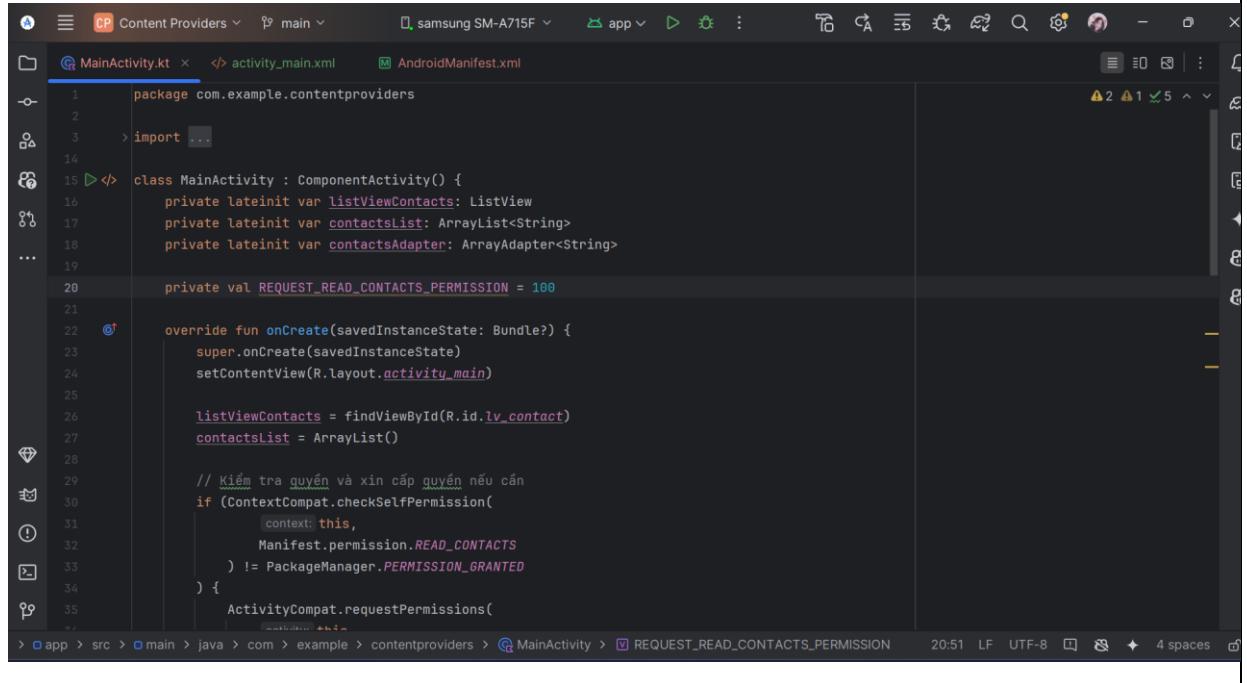
3. Đọc dữ liệu từ Content Provider:

- Trong Activity chính (ví dụ: MainActivity.kt), sử dụng ContentResolver để truy vấn dữ liệu từ Content Provider của ứng dụng Danh bạ.
- URI của Content Provider Danh bạ là:
ContactsContract.Contacts.CONTENT_URI
- Sử dụng phương thức query() của ContentResolver để lấy dữ liệu. Phương thức này trả về một Cursor chứa kết quả truy vấn.

- Duyệt Cursor để lấy tên của từng liên hệ và lưu vào một ArrayList<String>.

4. Hiển thị dữ liệu lên ListView:

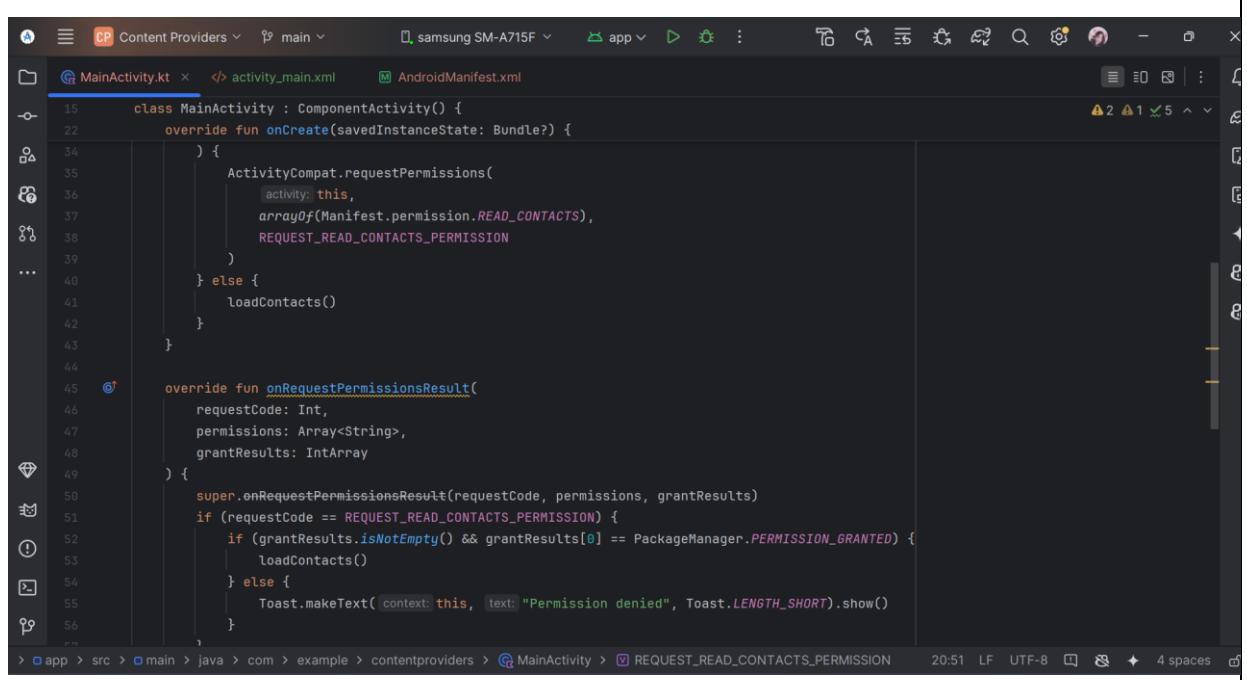
- Tạo một ArrayAdapter<String> để đưa dữ liệu từ ArrayList<String> lên ListView.
- Gán ArrayAdapter<String> cho ListView.



```

1 package com.example.contentproviders
2
3 import ...
4
5 class MainActivity : ComponentActivity() {
6     private lateinit var listViewContacts: ListView
7     private lateinit var contactsList: ArrayList<String>
8     private lateinit var contactsAdapter: ArrayAdapter<String>
9
10    private val REQUEST_READ_CONTACTS_PERMISSION = 100
11
12    override fun onCreate(savedInstanceState: Bundle?) {
13        super.onCreate(savedInstanceState)
14        setContentView(R.layout.activity_main)
15
16        listViewContacts = findViewById(R.id.lv_contact)
17        contactsList = ArrayList()
18
19        // Kiểm tra quyền và xin cấp quyền nếu cần
20        if (ContextCompat.checkSelfPermission(
21            context, Manifest.permission.READ_CONTACTS
22        ) != PackageManager.PERMISSION_GRANTED) {
23            ActivityCompat.requestPermissions(
24                this, arrayOf(Manifest.permission.READ_CONTACTS),
25                REQUEST_READ_CONTACTS_PERMISSION
26            )
27        }
28    }
29
30    override fun onRequestPermissionsResult(
31        requestCode: Int,
32        permissions: Array<String>,
33        grantResults: IntArray
34    ) {
35        super.onRequestPermissionsResult(requestCode, permissions, grantResults)
36        if (requestCode == REQUEST_READ_CONTACTS_PERMISSION) {
37            if (grantResults.isNotEmpty() && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
38                loadContacts()
39            } else {
40                Toast.makeText(context, "Permission denied", Toast.LENGTH_SHORT).show()
41            }
42        }
43    }
44
45    private fun loadContacts() {
46        contactsList.clear()
47        contactsAdapter.notifyDataSetChanged()
48    }
49}

```



```

1 package com.example.contentproviders
2
3 import ...
4
5 class MainActivity : ComponentActivity() {
6     private lateinit var listViewContacts: ListView
7     private lateinit var contactsList: ArrayList<String>
8     private lateinit var contactsAdapter: ArrayAdapter<String>
9
10    private val REQUEST_READ_CONTACTS_PERMISSION = 100
11
12    override fun onCreate(savedInstanceState: Bundle?) {
13        super.onCreate(savedInstanceState)
14        setContentView(R.layout.activity_main)
15
16        listViewContacts = findViewById(R.id.lv_contact)
17        contactsList = ArrayList()
18
19        // Kiểm tra quyền và xin cấp quyền nếu cần
20        if (ContextCompat.checkSelfPermission(
21            context, Manifest.permission.READ_CONTACTS
22        ) != PackageManager.PERMISSION_GRANTED) {
23            ActivityCompat.requestPermissions(
24                this, arrayOf(Manifest.permission.READ_CONTACTS),
25                REQUEST_READ_CONTACTS_PERMISSION
26            )
27        }
28    }
29
30    override fun onRequestPermissionsResult(
31        requestCode: Int,
32        permissions: Array<String>,
33        grantResults: IntArray
34    ) {
35        super.onRequestPermissionsResult(requestCode, permissions, grantResults)
36        if (requestCode == REQUEST_READ_CONTACTS_PERMISSION) {
37            if (grantResults.isNotEmpty() && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
38                loadContacts()
39            } else {
40                Toast.makeText(context, "Permission denied", Toast.LENGTH_SHORT).show()
41            }
42        }
43    }
44
45    private fun loadContacts() {
46        contactsList.clear()
47        contactsAdapter.notifyDataSetChanged()
48
49        val cursor = contentResolver.query(
50            ContactsContract.Contacts.CONTENT_URI,
51            null,
52            null,
53            null,
54            null
55        )
56
57        if (cursor != null) {
58            while (cursor.moveToNext()) {
59                val name = cursor.getString(cursor.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME))
60                contactsList.add(name)
61            }
62            cursor.close()
63        }
64
65        contactsAdapter.notifyDataSetChanged()
66    }
67}

```

```
15     class MainActivity : ComponentActivity() {
16         override fun onRequestPermissionsResult(
17             requestCode: Int, permissions: Array<String>, grantResults: IntArray) {
18             if (requestCode == REQUEST_READ_CONTACTS_PERMISSION) {
19                 if (grantResults.all { it == PackageManager.PERMISSION_GRANTED }) {
20                     loadContacts()
21                 } else {
22                     Toast.makeText(this, "Permission denied", Toast.LENGTH_SHORT).show()
23                 }
24             }
25         }
26     }
27
28     private fun loadContacts() {
29         val cursor: Cursor? = contentResolver.query(
30             ContactsContract.Contacts.CONTENT_URI,
31             arrayOf(ContactsContract.Contacts.DISPLAY_NAME), // Chỉ lấy cột tên
32             selection: null,
33             selectionArgs: null,
34             sortOrder: ContactsContract.Contacts.DISPLAY_NAME + " ASC" // Sắp xếp theo tên
35         )
36
37         cursor?.use {
38             if (it.count > 0) {
39                 while (it.moveToNext()) {
40                     val name = it.getString(columnIndex: 0) // Lấy tên từ cột đầu tiên
41                     contactsList.add(name)
42                 }
43             }
44         }
45     }
46
47     // Hiển thị danh sách tên lên ListView
48     contactsAdapter = ArrayAdapter(context: this, android.R.layout.simple_list_item_1, contactsList)
49     listViewContacts.adapter = contactsAdapter
50 }
```

```
51     }
52
53     // Hiển thị danh sách tên lên ListView
54     contactsAdapter = ArrayAdapter(context: this, android.R.layout.simple_list_item_1, contactsList)
55     listViewContacts.adapter = contactsAdapter
56 }
```

BÀI TẬP 2: Ứng dụng tự động trả lời tin nhắn cuộc gọi nhỡ

Mục tiêu:

- Sử dụng Broadcast Receiver để lắng nghe sự kiện cuộc gọi đến.
- Sử dụng Telephony API để lấy thông tin về cuộc gọi (số điện thoại).
- Sử dụng SMS API để gửi tin nhắn SMS.

Mô tả:

Ứng dụng sẽ tự động gửi một tin nhắn SMS đến số điện thoại của người gọi nhỡ, với nội dung thông báo rằng bạn đang bận và sẽ gọi lại sau.

Các bước thực hiện:

1. Khai báo quyền:

- Thêm các quyền cần thiết vào AndroidManifest.xml:
 - android.permission.READ_PHONE_STATE (để theo dõi trạng thái cuộc gọi)
 - android.permission.SEND_SMS (để gửi tin nhắn SMS)
 - android.permission.RECEIVE_SMS (nếu muốn xử lý cả tin nhắn đến)

2. Tạo Broadcast Receiver:

- Tạo một class kế thừa BroadcastReceiver để lắng nghe sự kiện android.intent.action.PHONE_STATE.
- Trong phương thức onReceive():
 - Kiểm tra trạng thái cuộc gọi (TelephonyManager.EXTRA_STATE_RINGING).
 - Nếu là cuộc gọi đến, lấy số điện thoại người gọi (intent.getStringExtra(TelephonyManager.EXTRA_INCOMING_NUMBER)).
 - Nếu cuộc gọi bị nhỡ (có thể theo dõi thêm sự kiện TelephonyManager.CALL_STATE_IDLE sau khi đổ chuông), gửi tin nhắn SMS đến số điện thoại đó.

3. Gửi tin nhắn SMS:

- Sử dụng SmsManager để gửi tin nhắn SMS.
- SmsManager.getDefault().sendTextMessage() để gửi tin nhắn.

4. Đăng ký Broadcast Receiver:

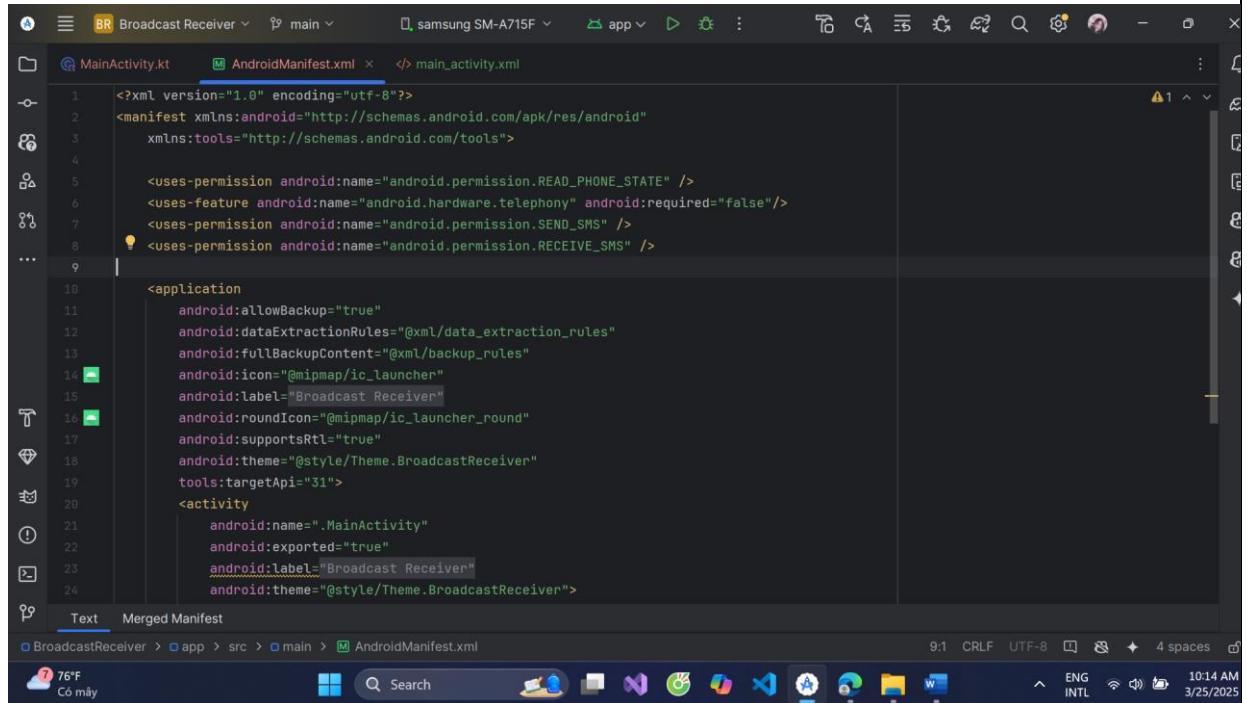
- Đăng ký receiver trong AndroidManifest.xml.

Hướng dẫn Bài tập 02: Chụp lại mà hình từng bước thực hiện (tương tự cho các Bài tập bên dưới) để học sinh lớp 10 có thể thực hiện lại theo được :D

1. Khai báo quyền:

- Thêm các quyền cần thiết vào AndroidManifest.xml:
 - android.permission.READ_PHONE_STATE (để theo dõi trạng thái cuộc gọi)
 - android.permission.SEND_SMS (để gửi tin nhắn SMS)
 - android.permission.RECEIVE_SMS (nếu muốn xử lý cả tin nhắn đến)

- Thêm dòng: <uses-feature android:name="android.hardware.telephony" android:required="false"/> để đảm bảo thiết bị không có phần cứng đó thì vẫn dùng được ứng dụng bởi không phải thiết bị nào cũng có loại phần cứng đó



```

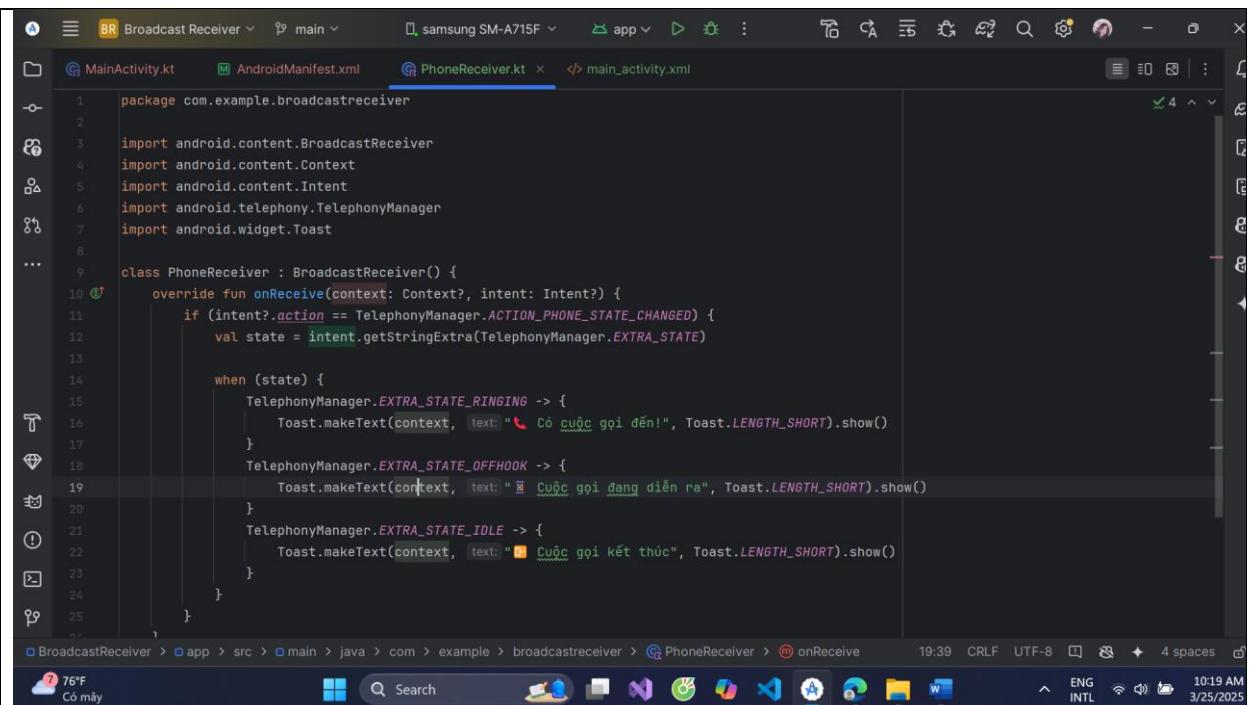
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

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

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="Broadcast Receiver"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.BroadcastReceiver"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="Broadcast Receiver"
            android:theme="@style/Theme.BroadcastReceiver">
    
```

1. Tạo Broadcast Receiver:

- Tạo một class kế thừa BroadcastReceiver để lắng nghe sự kiện android.intent.action.PHONE_STATE.



```
package com.example.broadcastreceiver

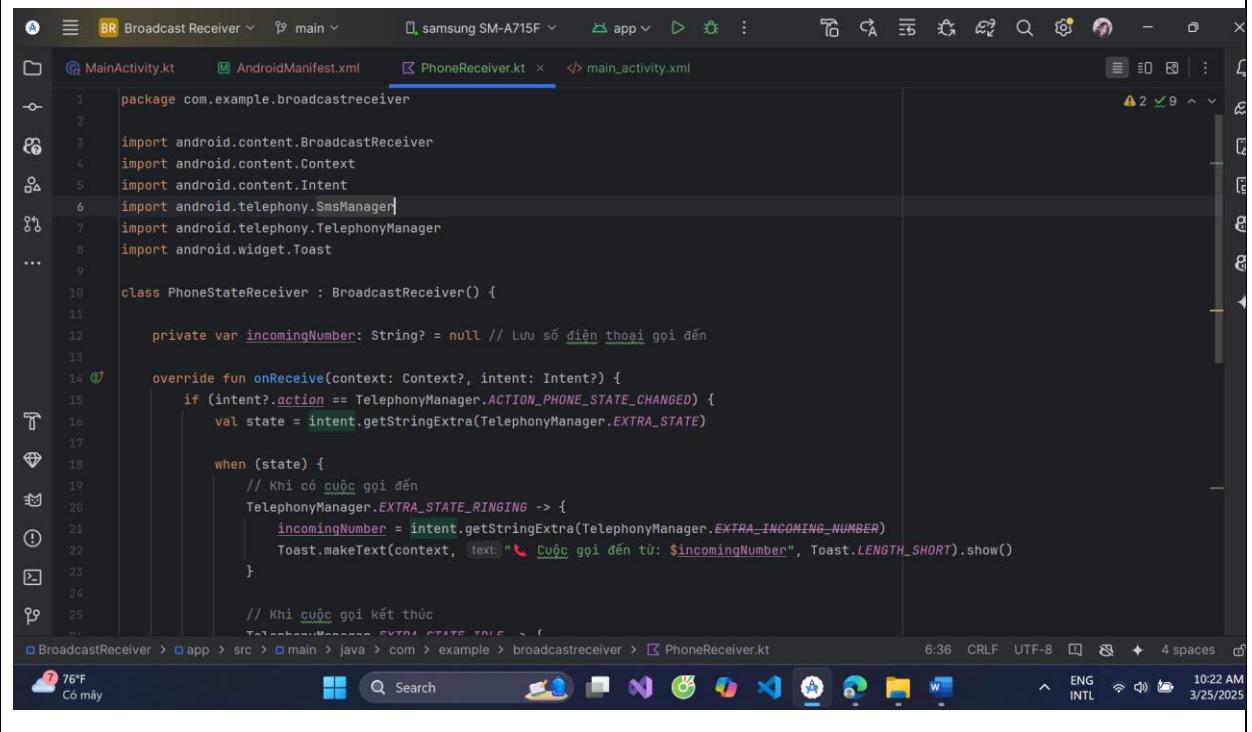
import android.content.BroadcastReceiver
import android.content.Context
import android.content.Intent
import android.telephony.TelephonyManager
import android.widget.Toast

class PhoneReceiver : BroadcastReceiver() {
    override fun onReceive(context: Context?, intent: Intent?) {
        if (intent?.action == TelephonyManager.ACTION_PHONE_STATE_CHANGED) {
            val state = intent.getStringExtra(TelephonyManager.EXTRA_STATE)

            when (state) {
                TelephonyManager.EXTRA_STATE_RINGING -> {
                    Toast.makeText(context, "来电提醒!", Toast.LENGTH_SHORT).show()
                }
                TelephonyManager.EXTRA_STATE_OFFHOOK -> {
                    Toast.makeText(context, "正在通话", Toast.LENGTH_SHORT).show()
                }
                TelephonyManager.EXTRA_STATE_IDLE -> {
                    Toast.makeText(context, "通话结束", Toast.LENGTH_SHORT).show()
                }
            }
        }
    }
}
```

o Trong phương thức onReceive():

- Kiểm tra trạng thái cuộc gọi (TelephonyManager.EXTRA_STATE_RINGING).
- Nếu là cuộc gọi đến, lấy số điện thoại người gọi (intent.getStringExtra(TelephonyManager.EXTRA_INCOMING_NUMBER)).
- Nếu cuộc gọi bị nhỡ (có thẻ theo dõi thêm sự kiện TelephonyManager.CALL_STATE_IDLE sau khi đổ chuông), gửi tin nhắn SMS đến số điện thoại đó.



```
package com.example.broadcastreceiver

import android.content.BroadcastReceiver
import android.content.Context
import android.content.Intent
import android.telephony.SmsManager
import android.telephony.TelephonyManager
import android.widget.Toast

class PhoneStateReceiver : BroadcastReceiver() {

    private var incomingNumber: String? = null // Lưu số điện thoại gọi đến

    override fun onReceive(context: Context?, intent: Intent?) {
        if (intent?.action == TelephonyManager.ACTION_PHONE_STATE_CHANGED) {
            val state = intent.getStringExtra(TelephonyManager.EXTRA_STATE)

            when (state) {
                // Khi có cuộc gọi đến
                TelephonyManager.EXTRA_STATE_RINGING -> {
                    incomingNumber = intent.getStringExtra(TelephonyManager.EXTRA_INCOMING_NUMBER)
                    Toast.makeText(context, "来电号码: $incomingNumber", Toast.LENGTH_SHORT).show()
                }
                // Khi cuộc gọi kết thúc
                TelephonyManager.EXTRA_STATE_IDLE -> {
                    // Handle call end logic
                }
            }
        }
    }
}
```

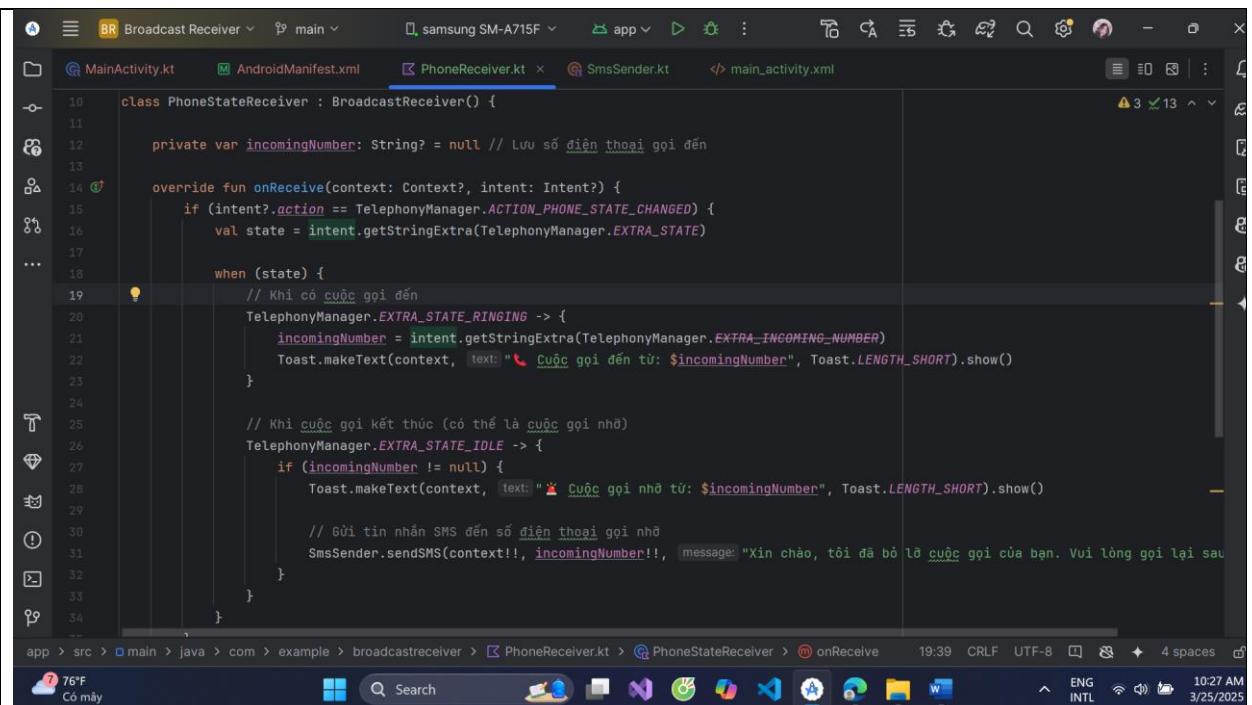
```
BR Broadcast Receiver ✓ main ✓ samsung SM-A715F ✓ app ✓ PhoneReceiver.kt x main_activity.xml

10 class PhoneStateReceiver : BroadcastReceiver() {
11     override fun onReceive(context: Context?, intent: Intent?) {
12         ...
13         ...
14         ...
15         ...
16         ...
17         ...
18         ...
19         ...
20         ...
21         ...
22         ...
23         ...
24         ...
25         ...
26         ...
27         ...
28         ...
29         ...
30         ...
31         ...
32         ...
33         ...
34     }
35
36     ...
37     private fun sendMissedCallSMS(context: Context?, phoneNumber: String) {
38         try {
39             val smsManager = SmsManager.getDefault()
40             val message = "Xin chào, tôi đã bỏ lỡ cuộc gọi của bạn. Vui lòng gọi lại sau nhé!"
41             smsManager.sendTextMessage(phoneNumber, null, message, null, null)
42             Toast.makeText(context, text: "SMS đã gửi đến: $phoneNumber", Toast.LENGTH_SHORT).show()
43         } catch (e: Exception) {
44             Toast.makeText(context, text: "Gửi SMS thất bại!", Toast.LENGTH_SHORT).show()
45         }
46     }
47 }
```

2. Gửi tin nhắn SMS:

- Sử dụng SmsManager để gửi tin nhắn SMS.
 - SmsManager.getDefault().sendTextMessage() để gửi tin nhắn.

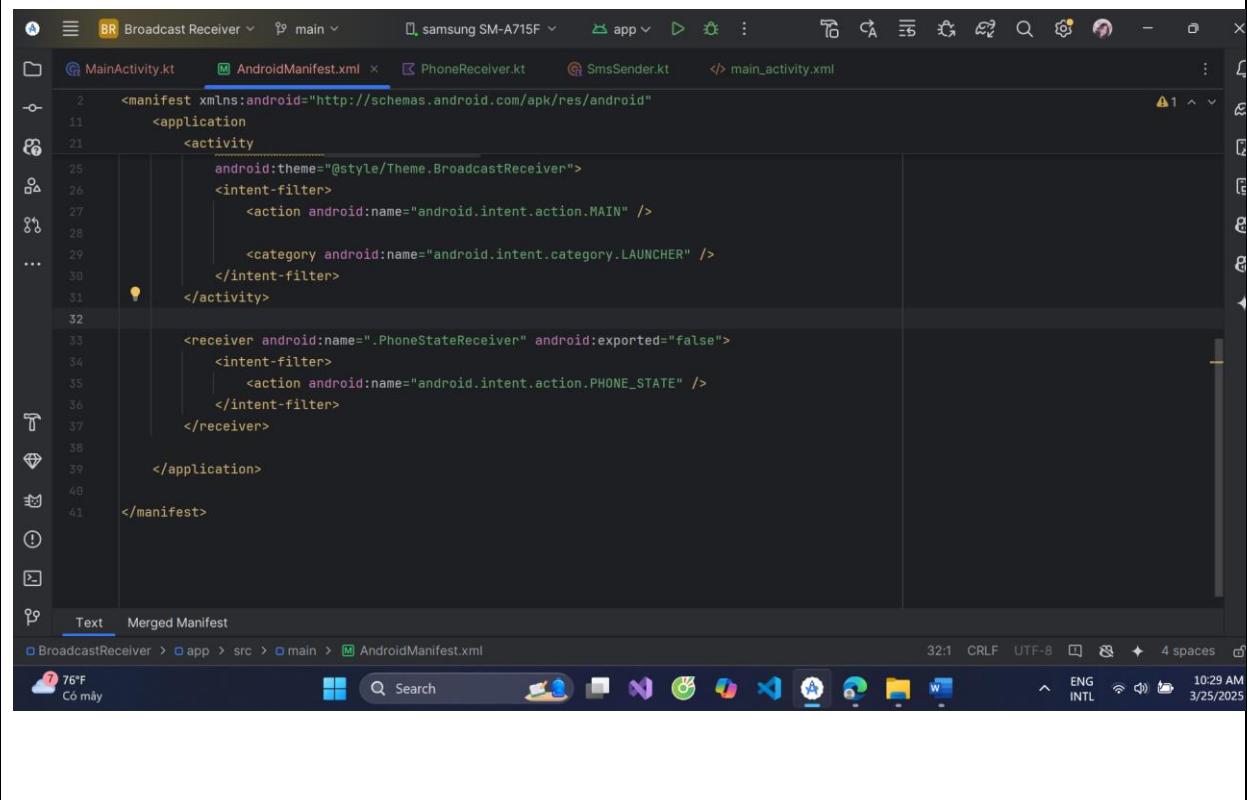
- File PhoneReceiver cập nhật để phát hiện cuộc gọi nhỡ sau khi có SmsSender



```
10 class PhoneStateReceiver : BroadcastReceiver() {
11
12     private var incomingNumber: String? = null // Lưu số điện thoại gọi đến
13
14     override fun onReceive(context: Context?, intent: Intent?) {
15         if (intent?.action == TelephonyManager.ACTION_PHONE_STATE_CHANGED) {
16             val state = intent.getStringExtra(TelephonyManager.EXTRA_STATE)
17
18             when (state) {
19                 // Khi có cuộc gọi đến
20                 TelephonyManager.EXTRA_STATE_RINGING -> {
21                     incomingNumber = intent.getStringExtra(TelephonyManager.EXTRA_INCOMING_NUMBER)
22                     Toast.makeText(context, "来电号码: $incomingNumber", Toast.LENGTH_SHORT).show()
23                 }
24
25                 // Khi cuộc gọi kết thúc (có thể là cuộc gọi nhỡ)
26                 TelephonyManager.EXTRA_STATE_IDLE -> {
27                     if (incomingNumber != null) {
28                         Toast.makeText(context, "已挂断: $incomingNumber", Toast.LENGTH_SHORT).show()
29
30                         // Gửi tin nhắn SMS đến số điện thoại gọi nhỡ
31                         SmsSender.sendSMS(context!!, incomingNumber!!, "Xin chào, tôi đã bỏ lỡ cuộc gọi của bạn. Vui lòng gọi lại sau")
32                     }
33                 }
34             }
35         }
36     }
37 }
```

3. Đăng ký Broadcast Receiver:

- Đăng ký receiver trong AndroidManifest.xml.



```
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     <application
4         <activity
5             android:theme="@style/Theme.BroadcastReceiver"
6             <intent-filter>
7                 <action android:name="android.intent.action.MAIN" />
8
9             <category android:name="android.intent.category.LAUNCHER" />
10        </intent-filter>
11    </activity>
12
13    <receiver android:name=".PhoneStateReceiver" android:exported="false">
14        <intent-filter>
15            <action android:name="android.intent.action.PHONE_STATE" />
16        </intent-filter>
17    </receiver>
18
19    </application>
20
21 </manifest>
```

BÀI TẬP 3: Ứng dụng chặn cuộc gọi theo số điện thoại

Mục tiêu:

- Sử dụng Broadcast Receiver để lắng nghe sự kiện cuộc gọi đến.

- Sử dụng Telephony API để lấy thông tin về cuộc gọi (số điện thoại).
- (Nâng cao) Tìm hiểu cách chặn cuộc gọi (có thể cần các phương pháp không chính thức hoặc API riêng của nhà sản xuất điện thoại).

Mô tả:

Ứng dụng sẽ tự động từ chối hoặc ngắt kết nối các cuộc gọi đến từ một danh sách các số điện thoại bị chặn.

Các bước thực hiện:

1. Khai báo quyền:

- Thêm quyền android.permission.READ_PHONE_STATE vào AndroidManifest.xml.
- (Có thể cần thêm các quyền liên quan đến xử lý cuộc gọi tùy theo phương pháp chặn)

2. Tạo Broadcast Receiver:

- Tạo một class kế thừa BroadcastReceiver để lắng nghe sự kiện android.intent.action.PHONE_STATE.
- Trong phương thức onReceive():
 - Kiểm tra trạng thái cuộc gọi (TelephonyManager.EXTRA_STATE_RINGING).
 - Nếu là cuộc gọi đến, lấy số điện thoại người gọi.
 - Kiểm tra xem số điện thoại đó có nằm trong danh sách chặn hay không.
 - Nếu có trong danh sách chặn, thực hiện hành động chặn cuộc gọi.

3. Chặn cuộc gọi:

- (Phần này có thể phức tạp và phụ thuộc vào phiên bản Android và nhà sản xuất điện thoại)
- Có thể cần sử dụng TelephonyManager hoặc các API khác để thực hiện chặn cuộc gọi.
- Lưu ý rằng việc chặn cuộc gọi có thể bị hạn chế hoặc không được hỗ trợ trên một số thiết bị.

4. Đăng ký Broadcast Receiver:

- Đăng ký receiver trong AndroidManifest.xml.

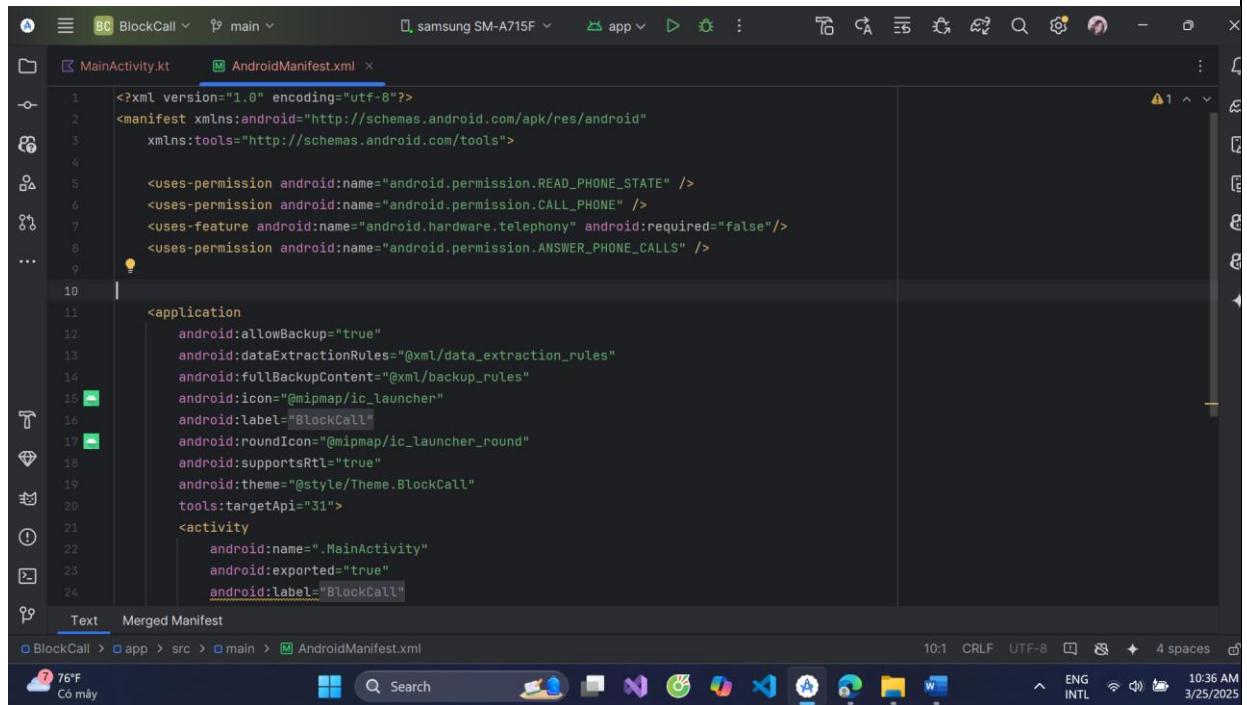
Lưu ý quan trọng:

- **Xử lý bất đồng bộ trong onReceive():** Tránh thực hiện các tác vụ tốn thời gian trong phương thức onReceive() của Broadcast Receiver. Nếu cần thực hiện các tác vụ dài, hãy sử dụng Service.
- **Quyền (Permissions):** Các thao tác liên quan đến Telephony và SMS đều yêu cầu các quyền đặc biệt. Đảm bảo bạn đã khai báo đầy đủ các quyền trong AndroidManifest.xml và xử lý việc xin quyền từ người dùng một cách thích hợp (đặc biệt là trên các phiên bản Android mới).
- **Hạn chế của Telephony API:** Một số chức năng liên quan đến Telephony có thể bị hạn chế hoặc không được hỗ trợ trên một số thiết bị hoặc phiên bản Android.
- **SMS PDU:** Khi nhận SMS, dữ liệu thường ở định dạng PDU. Cần xử lý để giải mã và đọc nội dung tin nhắn.

Hướng dẫn bài tập 3:

1. Khai báo quyền:

- Thêm quyền android.permission.READ_PHONE_STATE vào 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_PHONE_STATE" />
    <uses-permission android:name="android.permission.CALL_PHONE" />
    <uses-feature android:name="android.hardware.telephony" android:required="false"/>
    <uses-permission android:name="android.permission.ANSWER_PHONE_CALLS" />

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="BlockCall"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.BlockCall"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="BlockCall">
    
```

- (Có thể cần thêm các quyền liên quan đến xử lý cuộc gọi tùy theo phương pháp chặn)

2. Tạo Broadcast Receiver:

- Tạo một class kế thừa BroadcastReceiver để lắng nghe sự kiện android.intent.action.PHONE_STATE.
- Trong phương thức onReceive():
 - Kiểm tra trạng thái cuộc gọi (TelephonyManager.EXTRA_STATE_RINGING).
 - Nếu là cuộc gọi đến, lấy số điện thoại người gọi.
 - Kiểm tra xem số điện thoại đó có nằm trong danh sách chặn hay không.
 - Nếu có trong danh sách chặn, thực hiện hành động chặn cuộc gọi.

```
1 package com.example.blockcall
2
3 import android.content.BroadcastReceiver
4 import android.content.Context
5 import android.content.Intent
6 import android.telephony.TelephonyManager
7 import android.util.Log
8
9 class PhoneStateReceiver : BroadcastReceiver() {
10     override fun onReceive(context: Context?, intent: Intent?) {
11         if (intent?.action == TelephonyManager.ACTION_PHONE_STATE_CHANGED) {
12             val state = intent.getStringExtra(TelephonyManager.EXTRA_STATE)
13             val incomingNumber = intent.getStringExtra(TelephonyManager.EXTRA_INCOMING_NUMBER)
14
15             if (state == TelephonyManager.EXTRA_STATE_RINGING && incomingNumber != null) {
16                 Log.d(tag: "PhoneStateReceiver", msg: "Cuộc gọi đến từ: $incomingNumber")
17
18                 // Kiểm tra số điện thoại có trong danh sách chặn không
19                 if (isBlockedNumber(incomingNumber, context!!)) {
20                     Log.d(tag: "PhoneStateReceiver", msg: "Chặn cuộc gọi từ: $incomingNumber")
21                     endCall(context)
22                 }
23             }
24         }
25     }
26 }
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
```

The screenshot shows the Android Studio interface with the PhoneReceiver.kt file open. The code implements a BroadcastReceiver for the ACTION_PHONE_STATE_CHANGED intent. It checks if the phone state is RINGING and if there is an incoming number. It then calls a private function to check if the incoming number is blocked. If it is, it logs a message and ends the call using reflection on the TelephonyManager service.

3. Chặn cuộc gọi:

- (Phần này có thể phức tạp và phụ thuộc vào phiên bản Android và nhà sản xuất điện thoại)
- Có thể cần sử dụng TelephonyManager hoặc các API khác để thực hiện chặn cuộc gọi.
- Lưu ý rằng việc chặn cuộc gọi có thể bị hạn chế hoặc không được hỗ trợ trên một số thiết bị.

```
1 package com.example.blockcall
2
3 import android.content.BroadcastReceiver
4 import android.content.Context
5 import android.content.Intent
6 import android.telephony.TelephonyManager
7 import android.util.Log
8
9 class PhoneStateReceiver : BroadcastReceiver() {
10     override fun onReceive(context: Context?, intent: Intent?) {
11         if (intent?.action == TelephonyManager.ACTION_PHONE_STATE_CHANGED) {
12             val state = intent.getStringExtra(TelephonyManager.EXTRA_STATE)
13             val incomingNumber = intent.getStringExtra(TelephonyManager.EXTRA_INCOMING_NUMBER)
14
15             if (state == TelephonyManager.EXTRA_STATE_RINGING && incomingNumber != null) {
16                 Log.d(tag: "PhoneStateReceiver", msg: "Cuộc gọi đến từ: $incomingNumber")
17
18                 // Kiểm tra số điện thoại có trong danh sách chặn, bạn có thể thay bằng dữ liệu từ SharedPreferences hoặc Database
19                 if (isBlockedNumber(incomingNumber, context!!)) {
20                     Log.d(tag: "PhoneStateReceiver", msg: "Chặn cuộc gọi từ: $incomingNumber")
21                     endCall(context)
22                 }
23             }
24         }
25     }
26
27     private fun isBlockedNumber(number: String, context: Context): Boolean {
28         // Giả lập danh sách chặn, bạn có thể thay bằng dữ liệu từ SharedPreferences hoặc Database
29         val blockedNumbers = listOf("+84901234567", "+84876543210")
30         return blockedNumbers.contains(number)
31     }
32
33     private fun endCall(context: Context) {
34         try {
35             val telephonyManager = context.getSystemService(Context.TELEPHONY_SERVICE) as TelephonyManager
36             val clazz = Class.forName(telephonyManager.javaClass.name)
37             val method = clazz.getDeclaredMethod(name: "getITelephony")
38             method.isAccessible = true
39             val telephonyInterface = method.invoke(telephonyManager)
40             val endCallMethod = telephonyInterface.javaClass.getMethod(name: "endCall")
41             endCallMethod.invoke(telephonyInterface)
42         } catch (e: Exception) {
43             Log.e(tag: "PhoneStateReceiver", msg: "Không thể chặn cuộc gọi: ${e.message}")
44         }
45     }
46 }
```

The screenshot shows the same Android Studio interface with additional code added to the endCall() method. This code uses reflection to get the ITelephony interface from the TelephonyManager service and then invoke the endCall() method on it. This is necessary because the endCall() method is protected.

4. Đăng ký Broadcast Receiver:

- Đăng ký receiver trong AndroidManifest.xml.

The screenshot shows the Android Studio interface with the code editor open to the `AndroidManifest.xml` file. The manifest defines an application with a launcher icon, a round icon, and supports RTL. It includes a receiver for phone state changes and an activity for the main application. The code is as follows:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="BlockCall"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.BlockCall"
        tools:targetApi="31">
        <receiver android:name=".PhoneStateReceiver" android:exported="false">
            <intent-filter>
                <action android:name="android.intent.action.PHONE_STATE" />
            </intent-filter>
        </receiver>
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="BlockCall"
            android:theme="@style/Theme.BlockCall">
            <intent-filter>

```

BÀI TẬP 4: Ứng dụng tải và hiển thị ảnh từ Internet

Mục tiêu:

- Sử dụng `AsyncTask` để thực hiện tải ảnh từ một URL trên Internet trong background.
- Hiển thị ảnh đã tải xuống lên `ImageView` trong UI Thread.
- Hiển thị progress bar trong khi tải ảnh.

Mô tả:

Ứng dụng cho phép người dùng nhập một URL ảnh. Sau khi người dùng nhấn nút, ứng dụng sẽ hiển thị một progress bar và bắt đầu tải ảnh từ URL đó trong background. Khi tải xong, ứng dụng sẽ ẩn progress bar và hiển thị ảnh lên `ImageView`.

Các bước thực hiện:

1. Thiết kế giao diện:

- Một `EditText` để người dùng nhập URL.
- Một `ImageView` để hiển thị ảnh.
- Một `ProgressBar` để hiển thị tiến trình tải.
- Một `Button` để kích hoạt quá trình tải.

2. Tạo `AsyncTask`:

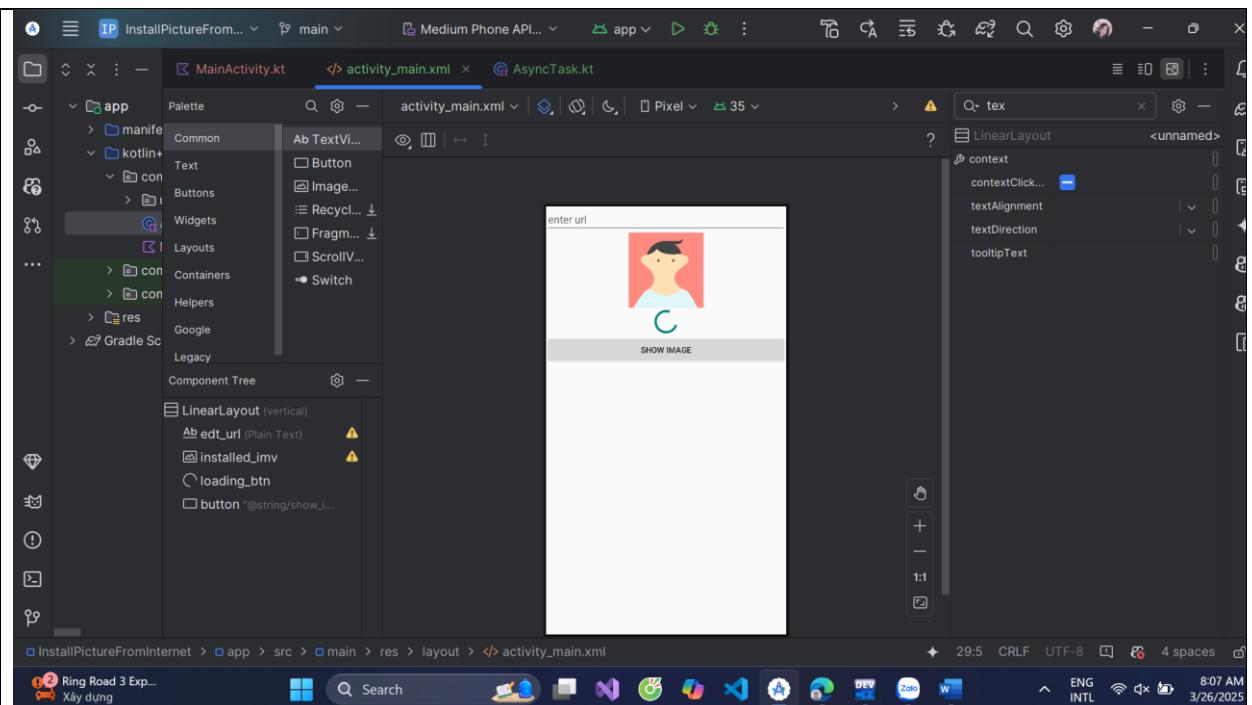
- Tạo một class kế thừa `AsyncTask<String, Integer, Bitmap>`.
 - `String`: URL của ảnh.
 - `Integer`: Tiến trình tải (ví dụ: phần trăm hoàn thành).
 - `Bitmap`: Ảnh đã tải.
- Implement các phương thức:
 - `onPreExecute()`: Hiển thị progress bar.
 - `doInBackground(String... urls)`:
 - Tải ảnh từ URL (sử dụng các thư viện như `HttpURLConnection` hoặc `OkHttp`).
 - Trong quá trình tải, gọi `publishProgress()` để cập nhật tiến trình (ví dụ: phần trăm tải).
 - Trả về `Bitmap` của ảnh đã tải.
 - `onProgressUpdate(Integer... values)`: Cập nhật progress bar trên UI thread.
 - `onPostExecute(Bitmap result)`:
 - Ẩn progress bar.
 - Hiển thị ảnh lên `ImageView` (nếu tải thành công).

3. Xử lý sự kiện Button click:

- Khi người dùng click nút, lấy URL từ `EditText`.
- Tạo một instance của `AsyncTask` và gọi `execute(url)`.

1. Thiết kế giao diện:

- Một `EditText` để người dùng nhập URL.
- Một `ImageView` để hiển thị ảnh.
- Một `ProgressBar` để hiển thị tiến trình tải.
- Một `Button` để kích hoạt quá trình tải.



2.Tạo AsyncTask:

- Tạo một class kế thừa `AsyncTask<String, Integer, Bitmap>`.
 - String: URL của ảnh.
 - Integer: Tiến trình tải (ví dụ: phần trăm hoàn thành).
 - Bitmap: Ảnh đã tải.

```

1 import android.graphics.BitmapFactory
2 import android.os.AsyncTask
3 import android.view.View
4 import android.widget.ImageView
5 import android.widget.ProgressBar
6 import java.io.InputStream
7 import java.net.HttpURLConnection
8 import java.net.URL
9
10
11 class DownloadImageTask(
12     private val imageView: ImageView,
13     private val progressBar: ProgressBar
14 ) : AsyncTask<String, Int, Bitmap?>() {
15
16     override fun onPreExecute() {
17         super.onPreExecute()
18         progressBar.visibility = View.VISIBLE // Hiển thị progress bar
19     }
20
21     override fun doInBackground(vararg params: String?): Bitmap? {
22         val urlString = params[0] ?: return null
23         var bitmap: Bitmap? = null
24         try {
25             val url = URL(urlString)
26             val connection = url.openConnection() as HttpURLConnection
27             connection.connect()
28             bitmap = BitmapFactory.decodeStream(connection.inputStream)
29         } catch (e: Exception) {
30             e.printStackTrace()
31         }
32         return bitmap
33     }
34
35     override fun onPostExecute(result: Bitmap?) {
36         imageView.setImageBitmap(result)
37         progressBar.visibility = View.GONE
38     }
39 }

```

- Implement các phương thức:
 - `onPreExecute()`: Hiển thị progress bar.
 - `doInBackground(String... urls)`:
 - Tải ảnh từ URL (sử dụng các thư viện như `HttpURLConnection` hoặc `OkHttp`).

- Trong quá trình tải, gọi `publishProgress()` để cập nhật tiến trình (ví dụ: phần trăm tải).
- Trả về `Bitmap` của ảnh đã tải.
- `onProgressUpdate(Integer... values)`: Cập nhật progress bar trên UI thread.
- `onPostExecute(Bitmap result)`:
 - Ẩn progress bar.
 - Hiển thị ảnh lên ImageView (nếu tải thành công).

```

11  class DownloadImageTask(
12
13    override fun doInBackground(vararg params: String?): Bitmap? {
14      val urlString = params[0] ?: return null
15      var bitmap: Bitmap? = null
16      try {
17          val url = URL(urlString)
18          val connection = url.openConnection() as HttpURLConnection
19          connection.doInput = true
20          connection.connect()
21
22          val fileLength = connection.contentLength // Lấy kích thước file
23          val inputStream: InputStream = connection.inputStream
24          val byteArray = ByteArray(fileLength)
25          var totalBytesRead = 0
26          var bytesRead: Int
27
28          while (inputStream.read(byteArray, totalBytesRead, byteArray.size - totalBytesRead)
29                  .also { bytesRead = it } != -1
30          ) {
31              totalBytesRead += bytesRead
32              publishProgress(...values: (totalBytesRead * 100) / fileLength) // Cập nhật tiến trình
33          }
34
35          bitmap = BitmapFactory.decodeByteArray(byteArray, offset: 0, totalBytesRead)
36          inputStream.close()
37      } catch (e: Exception) {
38          e.printStackTrace()
39      }
40
41      return bitmap
42  }
43
44  override fun onProgressUpdate(vararg values: Int?) {
45      super.onProgressUpdate(*values)
46      val progress = values[0] ?: 0
47      progressBar.progress = progress // Cập nhật progress bar
48  }
49
50  override fun onPostExecute(result: Bitmap?) {
51      super.onPostExecute(result)
52      progressBar.visibility = View.GONE // Ẩn progress bar
53      if (result != null) {
54          imageView.setImageBitmap(result) // Hiển thị ảnh lên ImageView
55      }
56  }
57
58
59
60
61
62
63
64

```

```

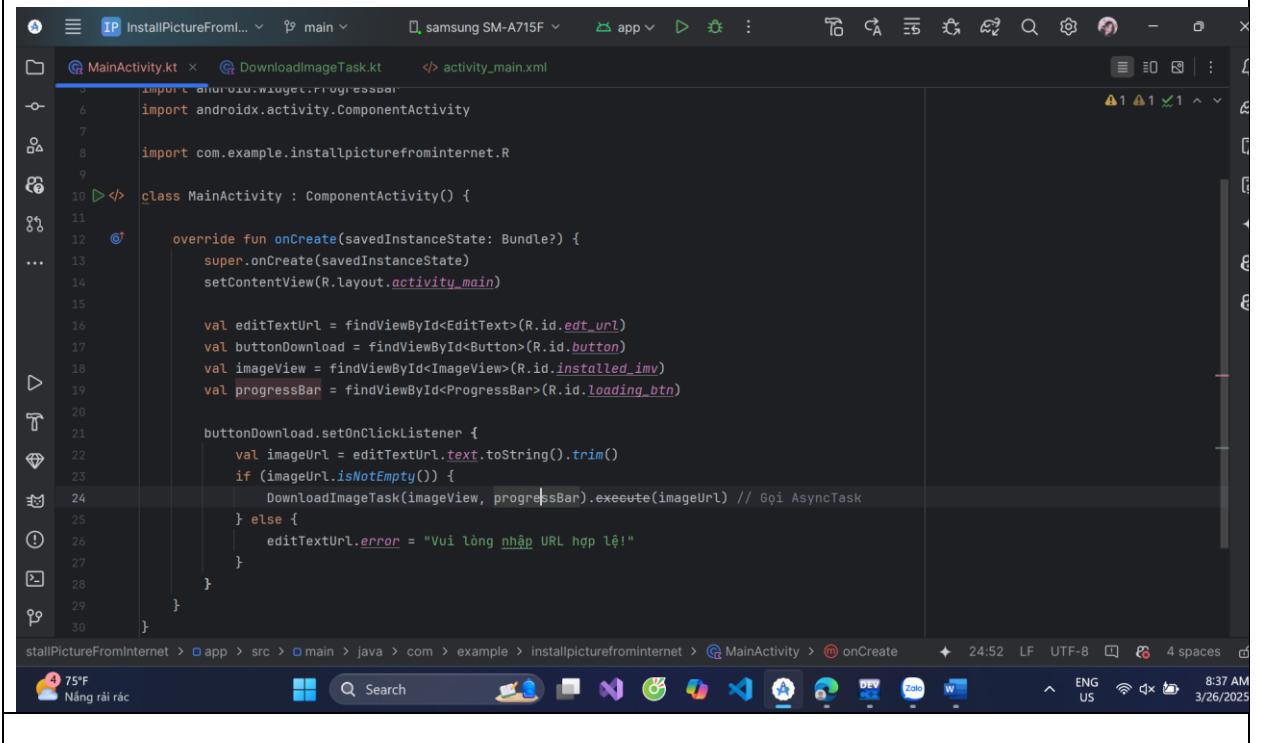
11  class DownloadImageTask(
12    override fun doInBackground(vararg params: String?): Bitmap? {
13
14        bitmap = BitmapFactory.decodeByteArray(byteArray, offset: 0, totalBytesRead)
15        inputStream.close()
16    } catch (e: Exception) {
17        e.printStackTrace()
18    }
19
20    return bitmap
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64

```

3.Xử lý sự kiện Button click:

- Khi người dùng click nút, lấy URL từ EditText.

- Tạo một instance của AsyncTask và gọi execute(url).



The screenshot shows the Android Studio interface with the code editor open. The file is `MainActivity.kt`. The code implements an `onCreate` method that finds views by ID and sets up a click listener for the download button. The listener checks if the URL entered in the edit text is not empty, then creates an instance of `DownloadImageTask` and calls its `execute` method with the URL. If the URL is empty, it sets an error message in the edit text.

```
import android.widget.ProgressBar
import androidx.appcompat.app.AppCompatActivity
import com.example.installpicturefrominternet.R
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val editTextUrl = findViewById<EditText>(R.id.edt_url)
        val buttonDownload = findViewById<Button>(R.id.button)
        val imageView = findViewById<ImageView>(R.id.installed_imv)
        val progressBar = findViewById<ProgressBar>(R.id.loading_btn)

        buttonDownload.setOnClickListener {
            val imageUrl = editTextUrl.text.toString().trim()
            if (imageUrl.isNotEmpty()) {
                DownloadImageTask(imageView, progressBar).execute(imageUrl) // Gọi AsyncTask
            } else {
                editTextUrl.error = "Vui lòng nhập URL hợp lệ!"
            }
        }
    }
}
```

BÀI TẬP 5: Ứng dụng đếm giờ và cập nhật giao diện

Mục tiêu:

- Sử dụng Handler và Runnable để cập nhật UI định kỳ từ một thread khác.
- Hiển thị thời gian đã trôi qua trên TextView.

Mô tả:

Ứng dụng hiển thị một TextView để hiển thị thời gian đã trôi qua (ví dụ: số giây). Một thread nền sẽ tăng giá trị thời gian và sử dụng Handler để gửi thông tin cập nhật lên UI thread để hiển thị.

Các bước thực hiện:

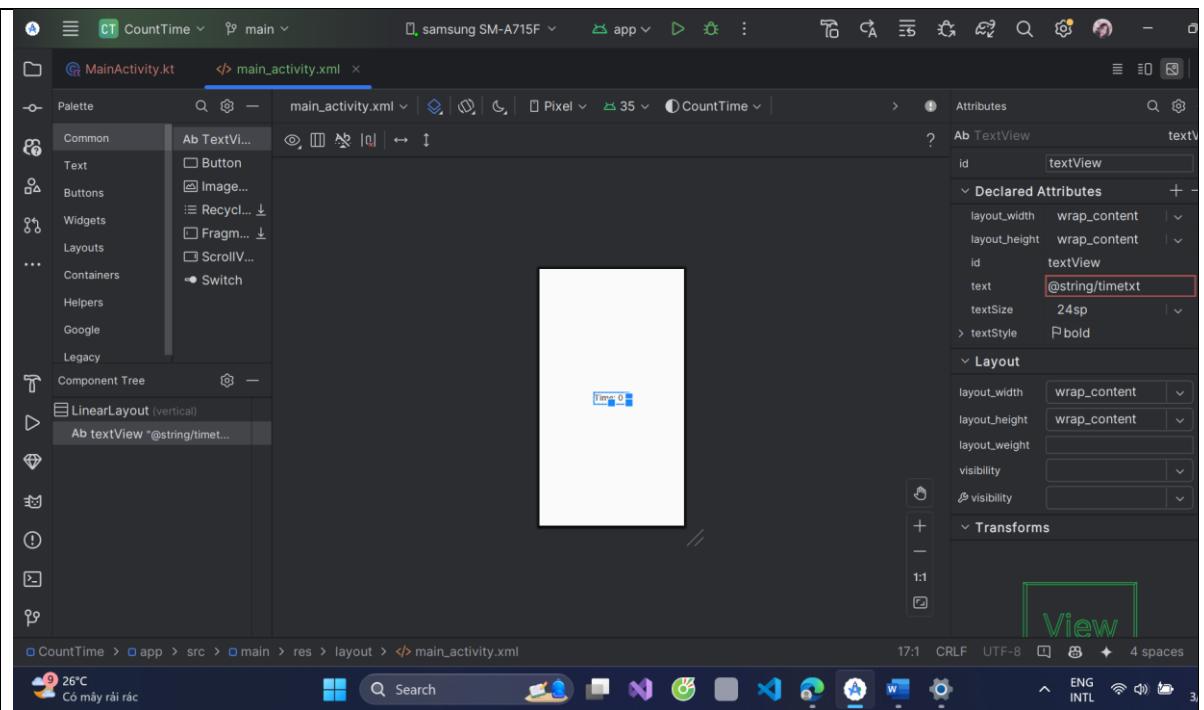
1. **Thiết kế giao diện:**
 - Một TextView để hiển thị thời gian.
2. **Tạo Handler:**
 - Tạo một Handler trong Activity chính.
 - Override phương thức `handleMessage()` (nếu dùng Message) hoặc tạo một Runnable.
3. **Tạo Thread:**
 - Tạo một Thread mới.
 - Trong `run()` method:
 - Lặp lại việc tăng giá trị thời gian.
 - Sử dụng `Handler.post(runnable)` để gửi một Runnable lên UI thread để cập nhật TextView.
4. **Cập nhật TextView:**
 - Trong Runnable, cập nhật `TextView.setText()` với giá trị thời gian hiện tại.

Lưu ý:

- **UI Thread:** Chỉ có UI thread mới được phép trực tiếp cập nhật các thành phần giao diện người dùng.
- **Tránh các tác vụ dài trên UI Thread:** Các tác vụ tốn thời gian (ví dụ: tải dữ liệu mạng, xử lý ảnh) nên được thực hiện trong background thread để tránh làm treo ứng dụng.
- **Sử dụng AsyncTask cho các tác vụ đơn giản:** AsyncTask là một cách dễ dàng để thực hiện các tác vụ nền đơn giản và tương tác với UI thread.
- **Sử dụng Handler và Thread cho các tác vụ phức tạp hơn:** Handler và Thread cung cấp sự linh hoạt cao hơn cho các tác vụ nền phức tạp hoặc cần kiểm soát thread chi tiết hơn.

1.Thiết kế giao diện:

- Một TextView để hiển thị thời gian.



2. Tạo Handler:

- Tạo một Handler trong Activity chính.
- Override phương thức handleMessage() (nếu dùng Message) hoặc tạo một Runnable.

```

package com.example.counttime
import ...
class MainActivity : ComponentActivity() {
    private lateinit var textView: TextView
    private var seconds = 0
    private var running = false
    private val handler = Handler(Looper.getMainLooper())

    private val runnable = object : Runnable {
        @SuppressLint("SetTextI18n")
        override fun run() {
            if (running) {
                seconds++
                textView.text = "Time: $seconds s"
                handler.postDelayed(this, delayMillis: 1000)
            }
        }
    }

    @SuppressLint("MissingInflatedId")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
    }
}

```

3.Tạo Thread:

- Tạo một Thread mới.
- Trong run() method:
 - Lặp lại việc tăng giá trị thời gian.
 -

- Sử dụng Handler.post(runnable) để gửi một Runnable lên UI thread để cập nhật TextView.

```

class MainActivity : ComponentActivity() {

    @SuppressLint("MissingInflatedId")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.main_activity)

        textView = findViewById(R.id.textView)
        running = true
        handler.post(runnable)
    }

    override fun onDestroy() {
        super.onDestroy()
        running = false
        handler.removeCallbacks(runnable)
    }
}

```

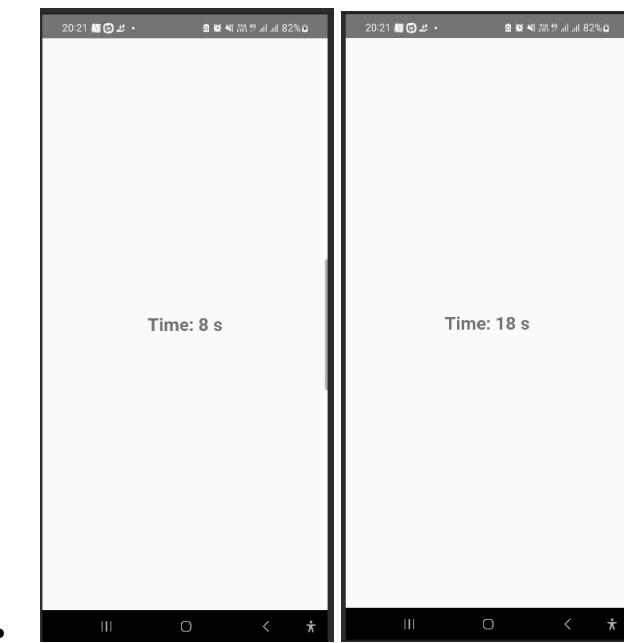
4.Cập nhật TextView:

- Trong Runnable, cập nhật TextView.setText() với giá trị thời gian hiện tại.

```

private val runnable = object : Runnable {
    @SuppressLint("SetTextI18n")
    override fun run() {
        if (running) {
            seconds++
            textView.text = "Time: $seconds s"
            handler.postDelayed(this, delayMillis: 1000)
        }
    }
}

```



BÀI TẬP 6: Ứng dụng ghi âm và phát lại

Mục tiêu:

- Sử dụng `MediaRecorder` để ghi âm từ microphone của thiết bị.
- Lưu file ghi âm vào `MediaStore` để các ứng dụng khác có thể truy cập.
- Sử dụng `MediaPlayer` để phát lại file ghi âm.
- Hiển thị danh sách các file ghi âm đã lưu trong `MediaStore`.

Mô tả:

Ứng dụng cho phép người dùng ghi âm, xem danh sách các bản ghi đã thực hiện và phát lại chúng.

Các bước thực hiện:

1. Ghi âm:

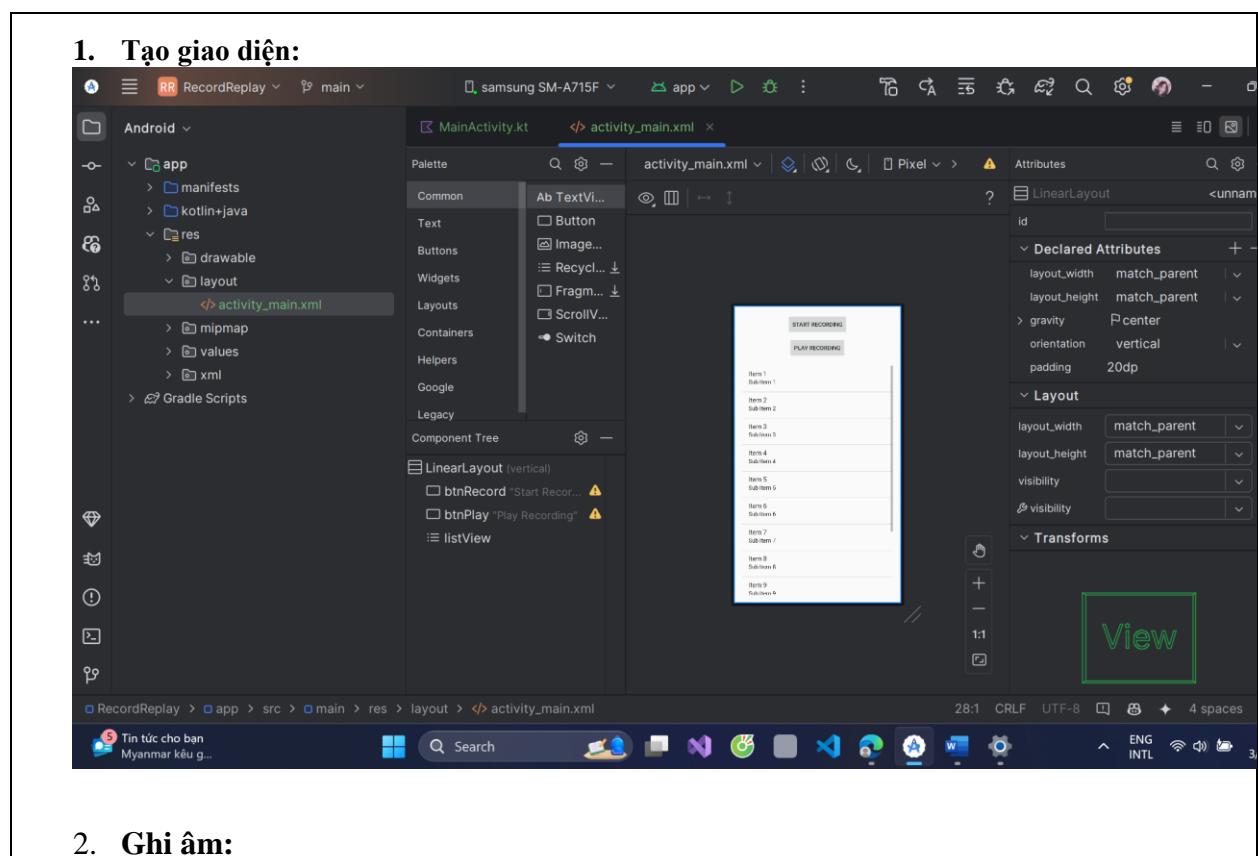
- Sử dụng `MediaRecorder` để ghi âm.
- Thiết lập các thông số cần thiết như nguồn âm thanh, định dạng file, nơi lưu trữ.
- Lưu file ghi âm vào một vị trí cụ thể.
- Sử dụng `ContentValues` để thêm thông tin về file ghi âm vào `MediaStore`.

2. Phát lại:

- Sử dụng `MediaPlayer` để phát lại file ghi âm.
- Có thể phát từ file hoặc từ một URI trong `MediaStore`.

3. Hiển thị danh sách file ghi âm:

- Sử dụng `ContentResolver` để truy vấn `MediaStore` và lấy danh sách các file âm thanh.
- Hiển thị danh sách này lên giao diện người dùng (ví dụ: `ListView`).



2. Ghi âm:

- a. Sử dụng MediaRecorder để ghi âm.
- b. Thiết lập các thông số cần thiết như nguồn âm thanh, định dạng file, nơi lưu trữ.
- c. Lưu file ghi âm vào một vị trí cụ thể.
- d. Sử dụng ContentValues để thêm thông tin về file ghi âm vào MediaStore.

```

<?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.RECORD_AUDIO"/>
    <uses-permission android:name="android.permission.READ_MEDIA_AUDIO"/>

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="RecordReplay"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.RecordReplay"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="RecordReplay"
            android:theme="@style/Theme.RecordReplay">
            <intent-filter>
        
```

- Thêm quyền vào AndroidManifest để xin quyền ghi âm và lưu file:

3. Phát lại:

- a. Sử dụng MediaPlayer để phát lại file ghi âm.
- b. Có thể phát từ file hoặc từ một URI trong MediaStore.

4. Hiển thị danh sách file ghi âm:

- a. Sử dụng ContentResolver để truy vấn MediaStore và lấy danh sách các file âm thanh.
- b. Hiển thị danh sách này lên giao diện người dùng (ví dụ: ListView).

Screenshot of Android Studio showing the code for MainActivity.kt. The code initializes MediaRecorder and MediaPlayer, sets up button listeners, and handles permissions.

```
7 import android.media.MediaRecorder
8 import android.net.Uri
9 import android.os.Bundle
10 import android.provider.MediaStore
11 import android.widget.ArrayAdapter
12 import android.widget.Button
13 import android.widget.ListView
14 import android.widget.Toast
15 import androidx.activity.ComponentActivity
16 import java.io.IOException
17
18 class MainActivity : ComponentActivity() {
19     private var mediaRecorder: MediaRecorder? = null
20     private var mediaPlayer: MediaPlayer? = null
21     private var audioUri: Uri? = null
22     private var isRecording = false
23
24     override fun onCreate(savedInstanceState: Bundle?) {
25         super.onCreate(savedInstanceState)
26         setContentView(R.layout.activity_main)
27
28         val btnRecord = findViewById<Button>(R.id.btnRecord)
29         val btnPlay = findViewById<Button>(R.id.btnPlay)
30         val listView = findViewById<ListView>(R.id.listView)
31
32         btnRecord.setOnClickListener { toggleRecording() }
33         btnPlay.setOnClickListener { startPlaying() }
34
35         requestPermissions()
36     }
37
38     private fun requestPermissions() {
39         val permissions = arrayOf(
40             Manifest.permission.RECORD_AUDIO,
41             Manifest.permission.READ_MEDIA_AUDIO // Android 13+
42         )
43
44         if (!permissions.all { checkSelfPermission(it) == PackageManager.PERMISSION_GRANTED }) {
45             requestPermissions(permissions, requestCode: 0)
46         }
47     }
48
49     private fun toggleRecording() {
50         if (isRecording) {
51             stopRecording()
52         } else {
53             startRecording()
54         }
55     }
56
57     private fun startRecording() {
58         if (mediaRecorder == null) {
59             mediaRecorder = MediaRecorder()
60             mediaRecorder?.setAudioSource(MediaRecorder.AudioSource.MIC)
61             mediaRecorder?.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP)
62             mediaRecorder?.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB)
63             mediaRecorder?.setOutputFile(audioUri)
64             mediaRecorder?.start()
65         }
66     }
67
68     private fun stopRecording() {
69         if (mediaRecorder != null) {
70             mediaRecorder?.stop()
71             mediaRecorder?.release()
72             mediaRecorder = null
73         }
74     }
75
76     private fun startPlaying() {
77         if (mediaPlayer == null) {
78             mediaPlayer = MediaPlayer()
79             mediaPlayer?.setDataSource(audioUri)
80             mediaPlayer?.start()
81         }
82     }
83
84     private fun stopPlaying() {
85         if (mediaPlayer != null) {
86             mediaPlayer?.stop()
87             mediaPlayer?.release()
88             mediaPlayer = null
89         }
90     }
91 }
```

Screenshot of Android Studio showing the code for MainActivity.kt. The code now includes permission requests and checks using PackageManager.

```
18 class MainActivity : ComponentActivity() {
19     override fun onCreate(savedInstanceState: Bundle?) {
20         val listView = findViewById<ListView>(R.id.listView)
21
22         btnRecord.setOnClickListener { toggleRecording() }
23         btnPlay.setOnClickListener { startPlaying() }
24
25         requestPermissions()
26     }
27
28     private fun requestPermissions() {
29         val permissions = arrayOf(
30             Manifest.permission.RECORD_AUDIO,
31             Manifest.permission.READ_MEDIA_AUDIO // Android 13+
32         )
33
34         if (!permissions.all { checkSelfPermission(it) == PackageManager.PERMISSION_GRANTED }) {
35             requestPermissions(permissions, requestCode: 0)
36         }
37     }
38
39     private fun toggleRecording() {
40         if (isRecording) {
41             stopRecording()
42         } else {
43             startRecording()
44         }
45     }
46
47     private fun startRecording() {
48         if (mediaRecorder == null) {
49             mediaRecorder = MediaRecorder()
50             mediaRecorder?.setAudioSource(MediaRecorder.AudioSource.MIC)
51             mediaRecorder?.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP)
52             mediaRecorder?.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB)
53             mediaRecorder?.setOutputFile(audioUri)
54             mediaRecorder?.start()
55         }
56     }
57
58     private fun stopRecording() {
59         if (mediaRecorder != null) {
60             mediaRecorder?.stop()
61             mediaRecorder?.release()
62             mediaRecorder = null
63         }
64     }
65
66     private fun startPlaying() {
67         if (mediaPlayer == null) {
68             mediaPlayer = MediaPlayer()
69             mediaPlayer?.setDataSource(audioUri)
70             mediaPlayer?.start()
71         }
72     }
73
74     private fun stopPlaying() {
75         if (mediaPlayer != null) {
76             mediaPlayer?.stop()
77             mediaPlayer?.release()
78             mediaPlayer = null
79         }
80     }
81 }
```

The screenshot shows the Android Studio interface with the code editor open to the file `MainActivity.kt`. The code implements a `ComponentActivity` that handles audio recording. It uses `ContentValues` to set the display name and MIME type for the recorded file, inserts it into the `MediaStore`, and then creates a `MediaRecorder` to start recording from the microphone. The recorded audio is saved in AMR_NB format.

```
18     class MainActivity : ComponentActivity() {
19         ...
20         private fun toggleRecording() {
21             if (isRecording) {
22                 stopRecording()
23             } else {
24                 startRecording()
25             }
26         }
27
28         private fun startRecording() {
29             val values = ContentValues().apply {
30                 put(MediaStore.Audio.Media.DISPLAY_NAME, "recording_${System.currentTimeMillis()}.3gp")
31                 put(MediaStore.Audio.Media.MIME_TYPE, "audio/3gpp")
32             }
33
34             val resolver = contentResolver
35             audioUri = resolver.insert(MediaStore.Audio.Media.EXTERNAL_CONTENT_URI, values)
36
37             try {
38                 val fileDescriptor = contentResolver.openFileDescriptor(audioUri!!, "w")?.fileDescriptor ?: return
39                 mediaRecorder = MediaRecorder().apply {
40                     setAudioSource(MediaRecorder.AudioSource.MIC)
41                     setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP)
42                     setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB)
43                 }
44             } catch (e: Exception) {
45                 e.printStackTrace()
46             }
47         }
48
49         private fun stopRecording() {
50             mediaRecorder?.apply {
51                 stop()
52                 release()
53             }
54         }
55
56         ...
57     }
```

The second screenshot shows the same code after some modifications. The `startRecording()` method now uses a `try-with-resources` block to handle the `MediaRecorder` and `FileDescriptor` automatically. The `mediaRecorder` variable is no longer explicitly released in the `stopRecording()` method.

```
18     class MainActivity : ComponentActivity() {
19         ...
20         private fun startRecording() {
21             try {
22                 val fileDescriptor = contentResolver.openFileDescriptor(audioUri!!, "w")?.fileDescriptor ?: return
23                 mediaRecorder = MediaRecorder().apply {
24                     setAudioSource(MediaRecorder.AudioSource.MIC)
25                     setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP)
26                     setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB)
27                     setOutputFile(fileDescriptor)
28                     prepare()
29                     start()
30                 }
31                 isRecording = true
32                 Toast.makeText(context, "Recording started", Toast.LENGTH_SHORT).show()
33             } catch (e: Exception) {
34                 e.printStackTrace()
35             }
36         }
37
38         private fun stopRecording() {
39             mediaRecorder?.apply {
40                 stop()
41                 release()
42             }
43         }
44
45         ...
46     }
```

MainActivity.kt

```
18     class MainActivity : ComponentActivity() {
19         private fun startRecording() {
20             ...
21             private fun stopRecording() {
22                 mediaRecorder?.apply {
23                     stop()
24                     release()
25                 }
26                 mediaRecorder = null
27                 isRecording = false
28                 Toast.makeText(context, "Recording saved", Toast.LENGTH_SHORT).show()
29                 loadRecordings()
30             }
31
32             private fun startPlaying() {
33                 if (audioUri == null) {
34                     Toast.makeText(context, "No recording found", Toast.LENGTH_SHORT).show()
35                     return
36                 }
37
38                 mediaPlayer = MediaPlayer().apply {
39                     try {
40                         setDataSource(applicationContext, audioUri!!)
41                     ...
42                 }
43             }
44
45             private fun loadRecordings() {
46                 val recordings = mutableListOf<String>()
47                 val projection = arrayOf(MediaStore.Audio.Media._ID, MediaStore.Audio.Media.DISPLAY_NAME)
48                 val cursor = contentResolver.query(
49                     MediaStore.Audio.Media.EXTERNAL_CONTENT_URI,
50                     projection, selection: null, selectionArgs: null, sortOrder: MediaStore.Audio.Media.DATE_ADDED + " DESC"
51                 )
52             }
53
54         }
55
56         private fun startPlaying() {
57             ...
58             mediaPlayer = MediaPlayer().apply {
59                 try {
60                     setDataSource(applicationContext, audioUri!!)
61                     prepare()
62                     start()
63                     Toast.makeText(context: this@MainActivity, text: "Playing Audio", Toast.LENGTH_SHORT).show()
64                 } catch (e: IOException) {
65                     e.printStackTrace()
66                 }
67             }
68
69             ...
70         }
71
72         private fun loadRecordings() {
73             val recordings = mutableListOf<String>()
74             val projection = arrayOf(MediaStore.Audio.Media._ID, MediaStore.Audio.Media.DISPLAY_NAME)
75             val cursor = contentResolver.query(
76                 MediaStore.Audio.Media.EXTERNAL_CONTENT_URI,
77                 projection, selection: null, selectionArgs: null, sortOrder: MediaStore.Audio.Media.DATE_ADDED + " DESC"
78             )
79         }
80
81         ...
82     }
83
84     ...
85
86     ...
87
88     ...
89
89     ...
90
91     ...
92
93
94
95     ...
96
97     ...
98
99     ...
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
129
130
131
132
133
134
135
136
137
138
139
139
140
141
142
143
144
145
146
147
148
149
149
150
151
152
153
154
155
156
157
158
159
159
160
161
162
163
164
165
166
167
168
169
169
170
171
172
173
174
175
176
177
178
179
179
180
181
182
183
184
185
186
187
188
189
189
190
191
192
193
194
195
196
197
198
199
199
200
201
202
203
204
205
206
207
207
208
209
209
210
211
212
213
214
214
215
216
216
217
217
218
218
219
219
220
220
221
221
222
222
223
223
224
224
225
225
226
226
227
227
228
228
229
229
230
230
231
231
232
232
233
233
234
234
235
235
236
236
237
237
238
238
239
239
240
240
241
241
242
242
243
243
244
244
245
245
246
246
247
247
248
248
249
249
250
250
251
251
252
252
253
253
254
254
255
255
256
256
257
257
258
258
259
259
260
260
261
261
262
262
263
263
264
264
265
265
266
266
267
267
268
268
269
269
270
270
271
271
272
272
273
273
274
274
275
275
276
276
277
277
278
278
279
279
280
280
281
281
282
282
283
283
284
284
285
285
286
286
287
287
288
288
289
289
290
290
291
291
292
292
293
293
294
294
295
295
296
296
297
297
298
298
299
299
300
300
301
301
302
302
303
303
304
304
305
305
306
306
307
307
308
308
309
309
310
310
311
311
312
312
313
313
314
314
315
315
316
316
317
317
318
318
319
319
320
320
321
321
322
322
323
323
324
324
325
325
326
326
327
327
328
328
329
329
330
330
331
331
332
332
333
333
334
334
335
335
336
336
337
337
338
338
339
339
340
340
341
341
342
342
343
343
344
344
345
345
346
346
347
347
348
348
349
349
350
350
351
351
352
352
353
353
354
354
355
355
356
356
357
357
358
358
359
359
360
360
361
361
362
362
363
363
364
364
365
365
366
366
367
367
368
368
369
369
370
370
371
371
372
372
373
373
374
374
375
375
376
376
377
377
378
378
379
379
380
380
381
381
382
382
383
383
384
384
385
385
386
386
387
387
388
388
389
389
390
390
391
391
392
392
393
393
394
394
395
395
396
396
397
397
398
398
399
399
400
400
401
401
402
402
403
403
404
404
405
405
406
406
407
407
408
408
409
409
410
410
411
411
412
412
413
413
414
414
415
415
416
416
417
417
418
418
419
419
420
420
421
421
422
422
423
423
424
424
425
425
426
426
427
427
428
428
429
429
430
430
431
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375

```

Screenshot of Android Studio showing the code for `MainActivity.kt` and three screenshots of the app's recording interface.

```

18     class MainActivity : ComponentActivity() {
19         private fun loadRecordings() {
20             val cursor = contentResolver.query(
21                 MediaStore.Audio.Media.EXTERNAL_CONTENT_URI,
22                 projection, selection: null, selectionArgs: null, sortOrder: MediaStore.Audio.Media.DATE_ADDED + " DESC"
23             )
24
25             cursor?.use {
26                 val nameIndex = it.getColumnIndex(MediaStore.Audio.Media.DISPLAY_NAME)
27                 while (it.moveToNext()) {
28                     recordings.add(it.getString(nameIndex))
29                 }
30             }
31         }
32
33         val adapter = ArrayAdapter(context: this, android.R.layout.simple_list_item_1, recordings)
34         findViewById<ListView>(R.id.listView).adapter = adapter
35     }

```

The app's interface consists of two buttons: "START RECORDING" and "PLAY RECORDING". Below these buttons is a list view displaying recorded files. The first screenshot shows a recording in progress, indicated by a red notification bar at the top. The second and third screenshots show the list after recording has stopped, with a green notification bar at the top indicating the recording was saved.

BÀI TẬP 7: Ứng dụng chơi video đơn giản

Mục tiêu:

- Sử dụng VideoView và MediaController để phát video.
- Cho phép người dùng chọn video từ MediaStore hoặc từ một URL.

Mô tả:

Ứng dụng cho phép người dùng xem video từ các nguồn khác nhau trên thiết bị hoặc từ Internet.

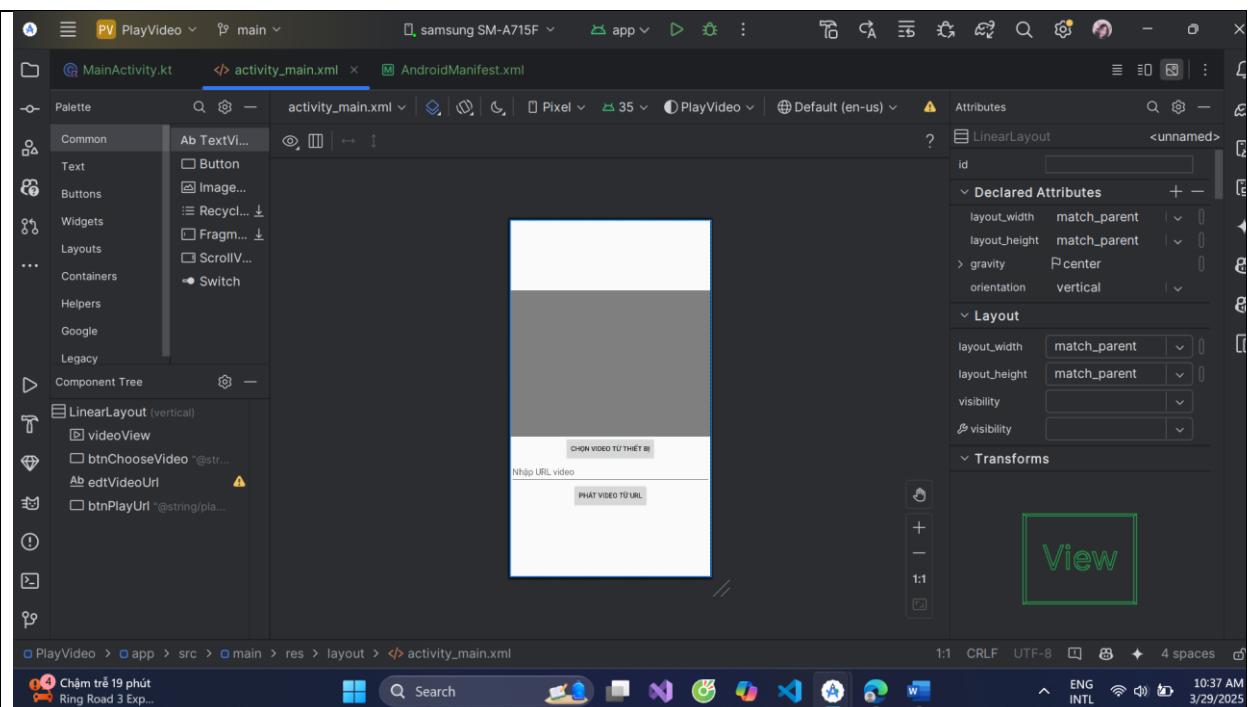
Các bước thực hiện:

1. **Thêm VideoView và MediaController vào layout.**
2. **Chọn video:**
 - Cho phép người dùng chọn video từ MediaStore bằng cách sử dụng Intent.ACTION_PICK và MediaStore.Video.Media.EXTERNAL_CONTENT_URI.
 - Hoặc cho phép người dùng nhập URL của video.
3. **Phát video:**
 - Sử dụng VideoView.setVideoURI() để thiết lập nguồn video.
 - Sử dụng MediaController để cung cấp các điều khiển phát lại video (play, pause, stop,...).
 - VideoView.start() để bắt đầu phát video.

Lưu ý:

- **Quyền (Permissions):** Đừng quên khai báo các quyền cần thiết trong AndroidManifest.xml, chẳng hạn như android.permission.RECORD_AUDIO, android.permission.READ_EXTERNAL_STORAGE, android.permission.WRITE_EXTERNAL_STORAGE, và android.permission.INTERNET.
- **Xử lý lỗi:** Cần xử lý các trường hợp lỗi có thể xảy ra, chẳng hạn như không tìm thấy file, lỗi khi ghi âm, lỗi khi phát lại, v.v.
- **Vòng đời (Lifecycle):** Quản lý các tài nguyên Media một cách chính xác trong các phương thức lifecycle của Activity/Fragment để tránh rò rỉ bộ nhớ và các vấn đề khác. Ví dụ, cần release() MediaPlayer và MediaRecorder khi không còn sử dụng.
- **MediaStore:** Làm quen với cách truy vấn và sử dụng MediaStore để lấy thông tin về các file media trên thiết bị.

1. **Thêm VideoView và MediaController vào layout.**



2. Chọn video:

- Cho phép người dùng chọn video từ MediaStore bằng cách sử dụng Intent.ACTION_PICK
MediaStore.Video.Media.EXTERNAL_CONTENT_URI.
- Hoặc cho phép người dùng nhập URL của video.
⇒ Vào androidManifest để thêm

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

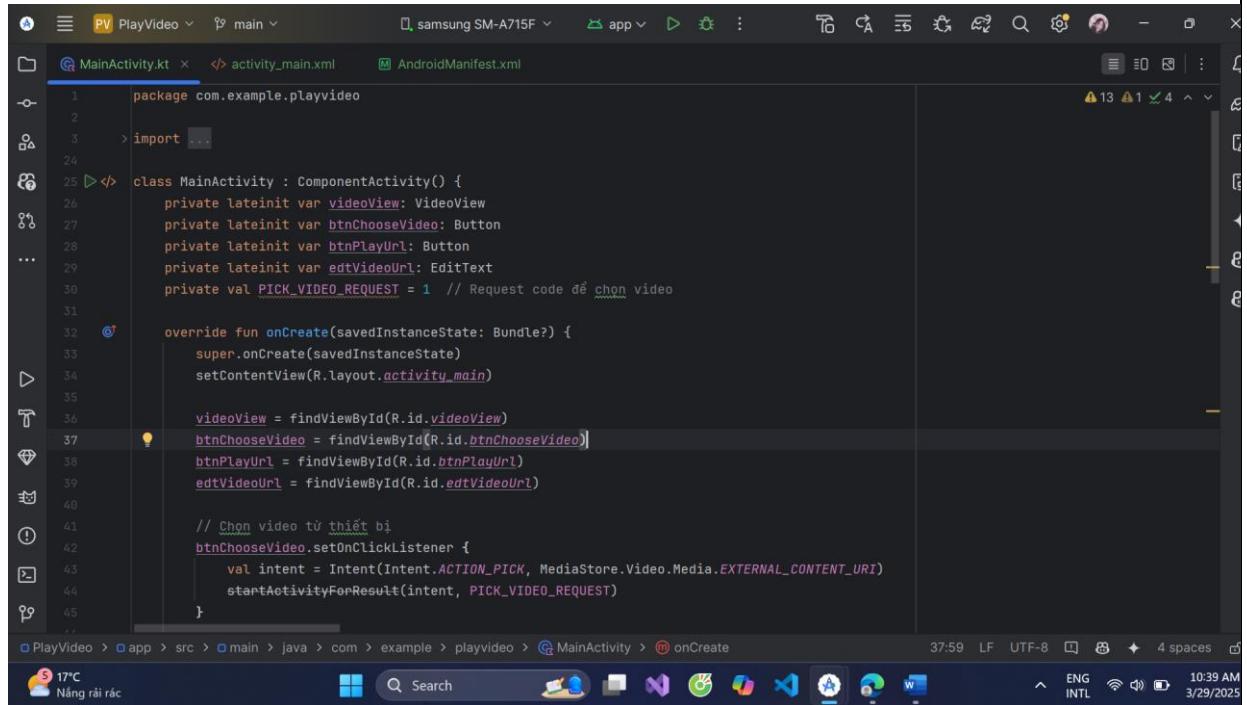
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"
        android:maxSdkVersion="32" />
    <uses-permission android:name="android.permission.INTERNET"/>

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="PlayVideo"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.PlayVideo"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="PlayVideo"
            android:theme="@style/Theme.PlayVideo">
            <intent-filter>
```

3. Phát video:

- Sử dụng VideoView.setVideoURI () để thiết lập nguồn video.

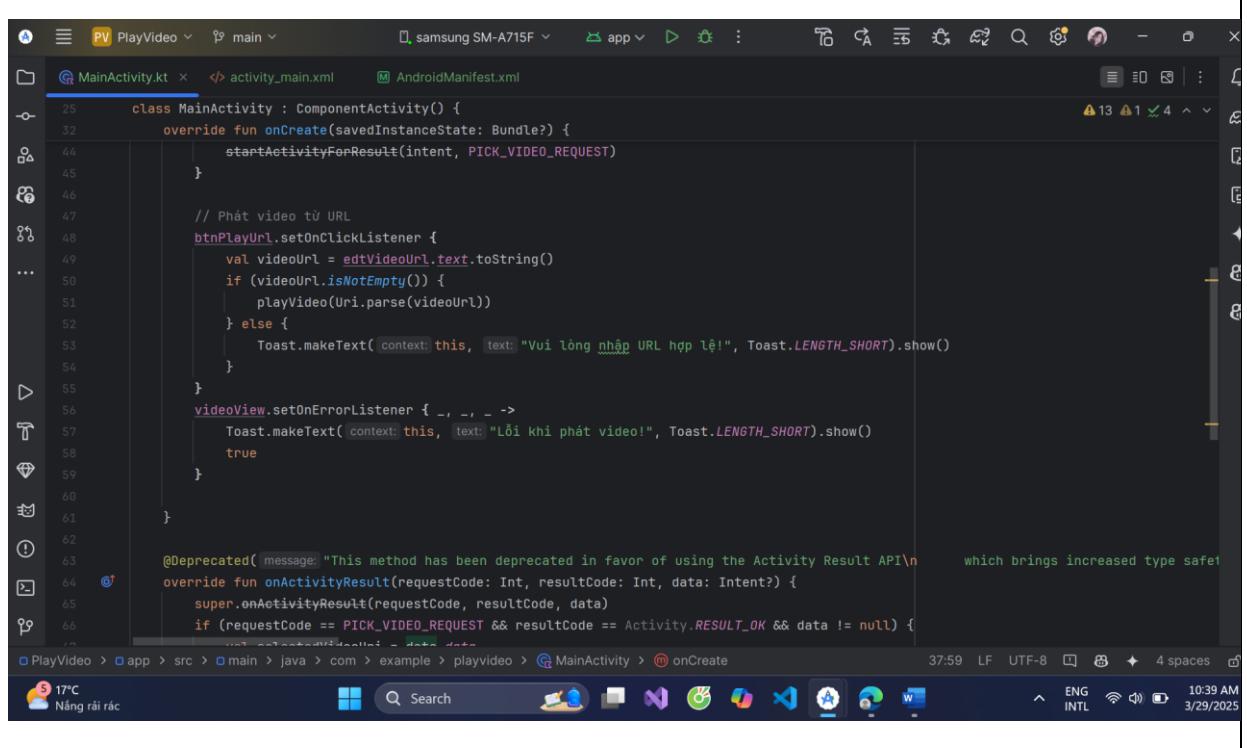
- Sử dụng MediaController để cung cấp các điều khiển phát lại video (play, pause, stop,...).
- VideoView.start() để bắt đầu phát video.



```

1 package com.example.playvideo
2
3 import ...
4
5 class MainActivity : ComponentActivity() {
6     private lateinit var videoView: VideoView
7     private lateinit var btnChooseVideo: Button
8     private lateinit var btnPlayUrl: Button
9     private lateinit var edtVideoUrl: EditText
10    private val PICK_VIDEO_REQUEST = 1 // Request code để chọn video
11
12    override fun onCreate(savedInstanceState: Bundle?) {
13        super.onCreate(savedInstanceState)
14        setContentView(R.layout.activity_main)
15
16        videoView = findViewById(R.id.videoView)
17        btnChooseVideo = findViewById(R.id.btnChooseVideo)
18        btnPlayUrl = findViewById(R.id.btnPlayUrl)
19        edtVideoUrl = findViewById(R.id.edtVideoUrl)
20
21        // Chọn video từ thiết bị
22        btnChooseVideo.setOnClickListener {
23            val intent = Intent(Intent.ACTION_PICK, MediaStore.Video.Media.EXTERNAL_CONTENT_URI)
24            startActivityForResult(intent, PICK_VIDEO_REQUEST)
25    }
26
27    override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
28        super.onActivityResult(requestCode, resultCode, data)
29
30        if (requestCode == PICK_VIDEO_REQUEST & resultCode == Activity.RESULT_OK & data != null) {
31            val videoUri = data.data
32            if (videoUri != null) {
33                playVideo(Uri.parse(videoUri))
34            } else {
35                Toast.makeText(context, text: "Vui lòng nhập URL hợp lệ!", Toast.LENGTH_SHORT).show()
36            }
37        }
38
39        videoView.setOnErrorListener { _, _, _ ->
40            Toast.makeText(context, text: "Lỗi khi phát video!", Toast.LENGTH_SHORT).show()
41            true
42        }
43
44    }
45
46    @Deprecated(message: "This method has been deprecated in favor of using the Activity Result API\nwhich brings increased type safety")
47    override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
48        super.onActivityResult(requestCode, resultCode, data)
49
50        if (requestCode == PICK_VIDEO_REQUEST & resultCode == Activity.RESULT_OK & data != null) {
51            val videoUri = data.data
52            if (videoUri != null) {
53                playVideo(Uri.parse(videoUri))
54            } else {
55                Toast.makeText(context, text: "Vui lòng nhập URL hợp lệ!", Toast.LENGTH_SHORT).show()
56            }
57        }
58
59    }
60
61    private fun playVideo(uri: Uri) {
62        videoView.setVideoURI(uri)
63        videoView.start()
64    }
65
66}

```



```

1 package com.example.playvideo
2
3 import ...
4
5 class MainActivity : ComponentActivity() {
6     override fun onCreate(savedInstanceState: Bundle?) {
7         startActivityForResult(intent, PICK_VIDEO_REQUEST)
8    }
9
10    // Phát video từ URL
11    btnPlayUrl.setOnClickListener {
12        val videoUrl = edtVideoUrl.text.toString()
13        if (videoUrl.isNotEmpty()) {
14            playVideo(Uri.parse(videoUrl))
15        } else {
16            Toast.makeText(context, text: "Vui lòng nhập URL hợp lệ!", Toast.LENGTH_SHORT).show()
17        }
18    }
19
20    videoView.setOnErrorListener { _, _, _ ->
21        Toast.makeText(context, text: "Lỗi khi phát video!", Toast.LENGTH_SHORT).show()
22        true
23    }
24
25    @Deprecated(message: "This method has been deprecated in favor of using the Activity Result API\nwhich brings increased type safety")
26    override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
27        super.onActivityResult(requestCode, resultCode, data)
28
29        if (requestCode == PICK_VIDEO_REQUEST & resultCode == Activity.RESULT_OK & data != null) {
30            val videoUri = data.data
31            if (videoUri != null) {
32                playVideo(Uri.parse(videoUri))
33            } else {
34                Toast.makeText(context, text: "Vui lòng nhập URL hợp lệ!", Toast.LENGTH_SHORT).show()
35            }
36        }
37
38        videoView.setOnErrorListener { _, _, _ ->
39            Toast.makeText(context, text: "Lỗi khi phát video!", Toast.LENGTH_SHORT).show()
40            true
41        }
42    }
43
44    private fun playVideo(uri: Uri) {
45        videoView.setVideoURI(uri)
46        videoView.start()
47    }
48
49}

```

```

25     class MainActivity : ComponentActivity() {
26         override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
27             if (selectedVideoUri != null) {
28                 playVideo(selectedVideoUri)
29             }
30         }
31     }
32
33     private fun playVideo(uri: Uri) {
34         val mediaController = MediaController(context)
35         mediaController.setAnchorView(videoView)
36         videoView.setMediaController(mediaController)
37         videoView.setVideoURI(uri)
38         videoView.start()
39     }
40
41     override fun onPause() {
42         super.onPause()
43         if (videoView.isPlaying) {
44             videoView.pause()
45         }
46     }
47 }
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90

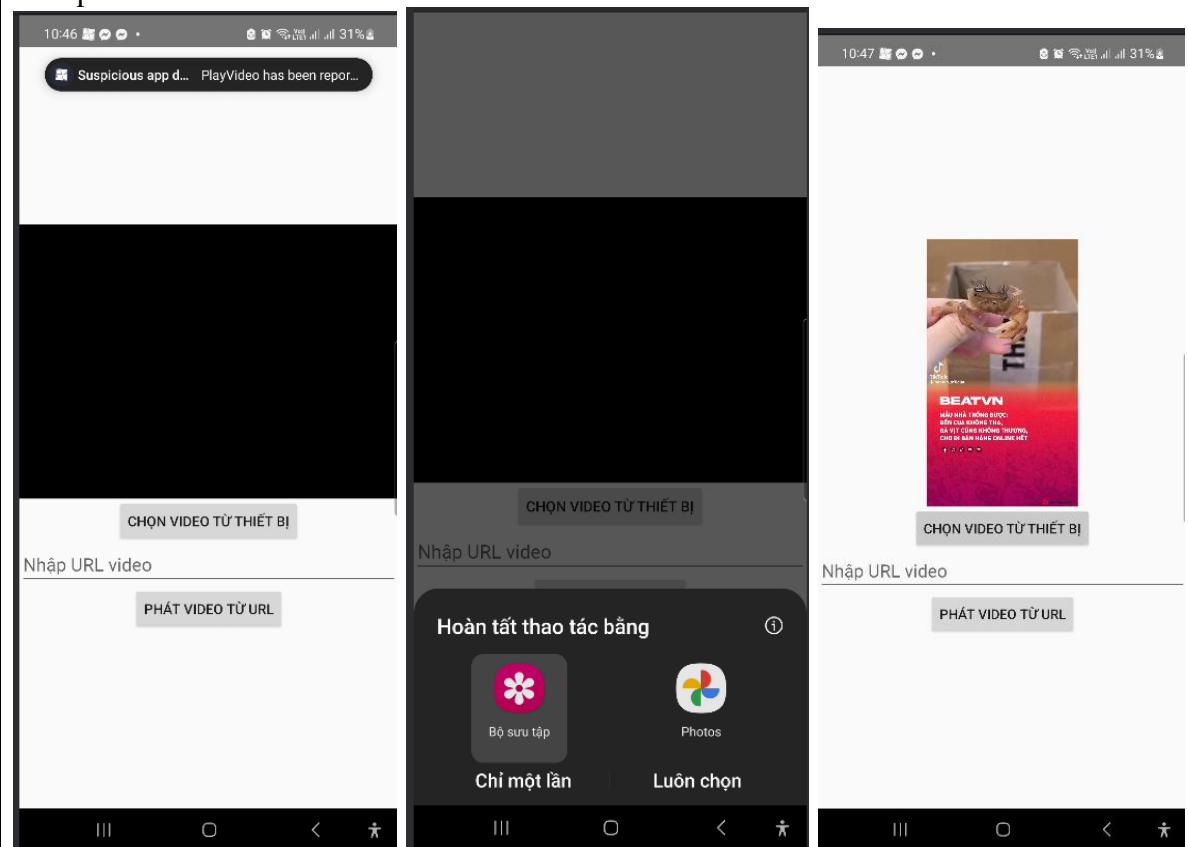
```

PlayVideo > app > src > main > java > com > example > playvideo > MainActivity > onCreate

60:1 LF UTF-8 4 spaces

17°C Nắng ráo ráo 10:40 AM ENG INTL 3/29/2025

Kết quả:



BÀI TẬP 8: Ứng dụng đo gia tốc

Mục tiêu:

- Sử dụng cảm biến gia tốc (TYPE_ACCELEROMETER) để đo gia tốc của thiết bị theo ba trục x, y, z.
- Hiển thị giá trị gia tốc lên TextView.
- Hiển thị một hình ảnh động (ví dụ: một quả bóng) di chuyển theo hướng gia tốc.

Mô tả:

Ứng dụng hiển thị các giá trị gia tốc đo được từ cảm biến gia tốc và mô phỏng sự di chuyển của một vật thể dựa trên các giá trị này.

Các bước thực hiện:

- 1. Lấy SensorManager và Sensor:**
 - Lấy SensorManager từ hệ thống.
 - Sử dụng SensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER) để lấy đối tượng Sensor.
- 2. Đăng ký SensorEventListener:**
 - Đăng ký một SensorEventListener để lắng nghe các sự kiện từ cảm biến gia tốc.
 - Implement hai phương thức của SensorEventListener:
 - onSensorChanged(SensorEvent event): Xử lý các sự kiện cảm biến, lấy giá trị gia tốc từ event.values.
 - onAccuracyChanged(Sensor sensor, int accuracy): Xử lý khi độ chính xác của cảm biến thay đổi.
- 3. Hiển thị giá trị gia tốc:**
 - Cập nhật các TextView để hiển thị giá trị gia tốc theo trục x, y, z.
- 4. Mô phỏng chuyển động:**
 - Sử dụng một ImageView để hiển thị hình ảnh động.
 - Trong phương thức onSensorChanged(), tính toán sự thay đổi vị trí của hình ảnh dựa trên giá trị gia tốc.
 - Cập nhật vị trí của ImageView.

1. Lấy SensorManager và Sensor:

- Lấy SensorManager từ hệ thống.
- Sử dụng SensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER) để lấy đối tượng Sensor.

```
<?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-feature android:name="android.hardware.sensor.accelerometer" 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="Dogiatoc"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Dogiatoc"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="Dogiatoc"
            android:theme="@style/Theme.Dogiatoc">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
            </intent-filter>
            <category android:name="android.intent.category.LAUNCHER" />
        
```

2. Đăng ký SensorEventListener:

- Đăng ký một SensorEventListener để lắng nghe các sự kiện từ cảm biến gia tốc.
- Implement hai phương thức của SensorEventListener:
 - onSensorChanged(SensorEvent event): Xử lý các sự kiện cảm biến, lấy giá trị gia tốc từ event.values.
 - onAccuracyChanged(Sensor sensor, int accuracy): Xử lý khi độ chính xác của cảm biến thay đổi.

3. Hiển thị giá trị gia tốc:

- Cập nhật các TextView để hiển thị giá trị gia tốc theo trục x, y, z.

4. Mô phỏng chuyển động:

- Sử dụng một ImageView để hiển thị hình ảnh động.
- Trong phương thức onSensorChanged(), tính toán sự thay đổi vị trí của hình ảnh dựa trên giá trị gia tốc.
- Cập nhật vị trí của ImageView.

The screenshot shows the Android Studio interface with the project 'Dogiatoc' selected. The left sidebar displays the project structure under 'Android'. The main editor window shows the 'MainActivity.kt' file. The code is as follows:

```
package com.example.dogiatoc
import ...
class MainActivity : ComponentActivity(), SensorEventListener {
    private lateinit var sensorManager: SensorManager
    private var accelerometer: Sensor? = null
    private lateinit var tvX: TextView
    private lateinit var tvY: TextView
    private lateinit var tvZ: TextView
    private lateinit var ball: ImageView
    private var posX = 0f
    private var posY = 0f
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        // Ánh xạ view
        tvX = findViewById(R.id.tv_x)
        tvY = findViewById(R.id.tv_y)
        tvZ = findViewById(R.id.tv_z)
        ball = findViewById(R.id.ball)
    }
}
```

The status bar at the bottom shows the date and time as 3/29/2025, 11:05 AM.

The screenshot shows the same Android Studio interface with the project 'Dogiatoc'. The code in 'MainActivity.kt' has been updated to include sensor registration and unregistration logic:

```
class MainActivity : ComponentActivity(), SensorEventListener {
    override fun onCreate(savedInstanceState: Bundle?) {
        // Lấy SensorManager và cảm biến giá tốc
        sensorManager = getSystemService(SENSOR_SERVICE) as SensorManager
        accelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)
    }
    override fun onResume() {
        super.onResume()
        accelerometer?.let {
            sensorManager.registerListener(listener, it, SensorManager.SENSOR_DELAY_UI)
        }
    }
    override fun onPause() {
        super.onPause()
        sensorManager.unregisterListener(listener, this)
    }
    @SuppressLint("SetTextI18n")
    override fun onSensorChanged(event: SensorEvent) {
        posX = event.values[0]
        posY = event.values[1]
        posZ = event.values[2]
        ball.x = posX
        ball.y = posY
    }
}
```

The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The left sidebar shows the project structure under "Android". The "app" module contains "manifests", "kotlin+java", "res", and "layout" directories.
- MainActivity.kt:** The code editor displays the `MainActivity` class. It includes methods for handling sensor changes and moving a ball. The ball's position is updated every 2 units, and it is bounded within a 1000x1000 area.
- AndroidManifest.xml:** The manifest file is shown at the bottom of the editor.
- activity_main.xml:** The XML layout file is shown at the bottom of the editor.
- Bottom Bar:** Shows the device (Samsung SM-A715F), orientation (Portrait), battery level (17%), weather (Nắng rải rác), and system status (ENG INTL, 11:05 AM, 3/29/2025).

```
class MainActivity : ComponentActivity(), SensorEventListener {
    override fun onSensorChanged(event: SensorEvent?) {
        // Hiển thị giá trị trên TextView
        tvX.text = "X: $x"
        tvY.text = "Y: $y"
        tvZ.text = "Z: $z"

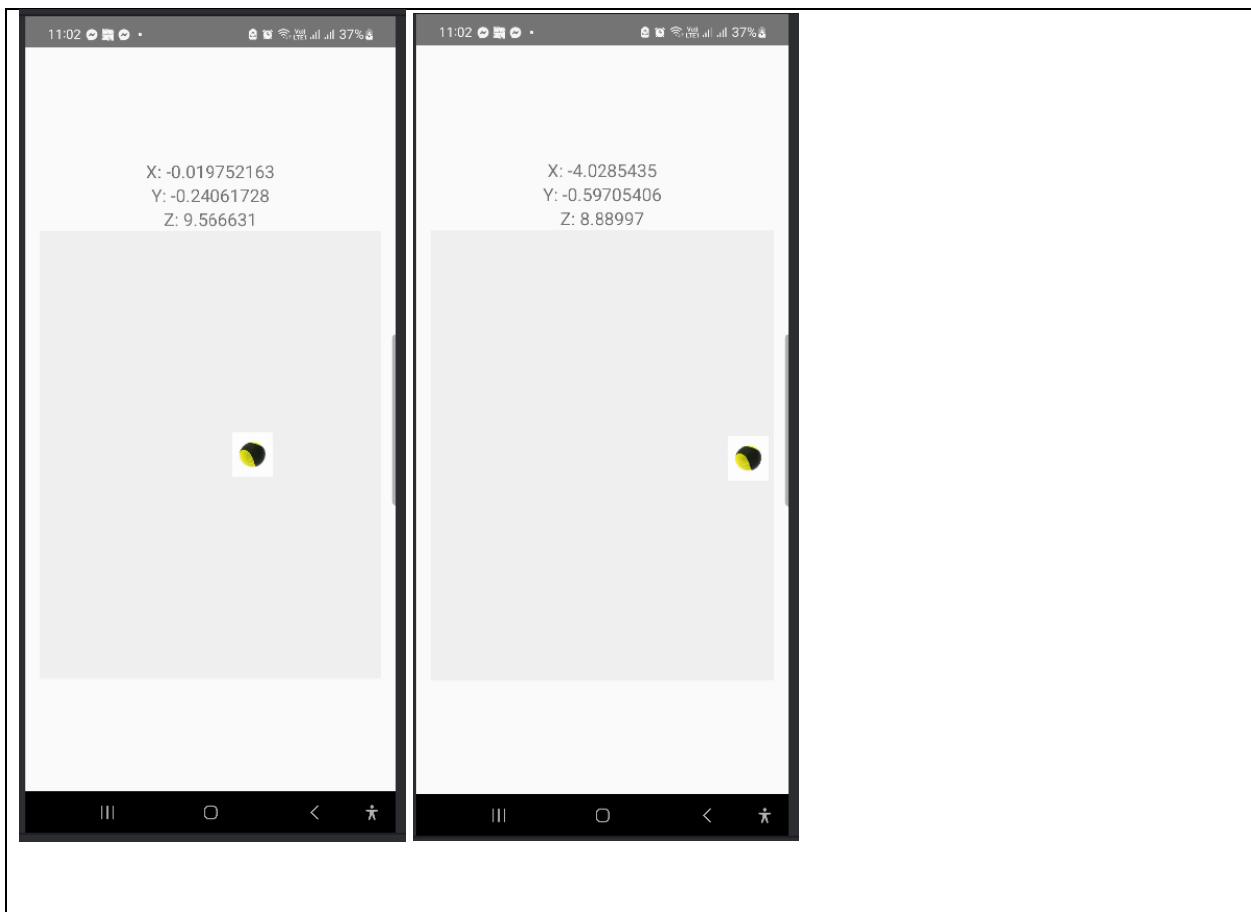
        // Cập nhật vị trí hình ảnh
        moveBall(x, y)
    }

    private fun moveBall(x: Float, y: Float) {
        posX -= x * 2
        posY += y * 2

        // Giới hạn biên
        if (posX < 0) posX = 0f
        if (posY < 0) posY = 0f
        if (posX > 1000) posX = 1000f
        if (posY > 1000) posY = 1000f

        ball.translationX = posX
        ball.translationY = posY
    }
}
```

Kết quả



BÀI TẬP 9: Ứng dụng la bàn

Mục tiêu:

- Sử dụng cảm biến từ trường (TYPE_MAGNETIC_FIELD) và cảm biến gia tốc (TYPE_ACCELEROMETER) để xác định hướng bắc.
- Hiển thị hướng bắc trên một ImageView (ví dụ: kim la bàn).
- Hiển thị góc lệch so với hướng bắc.

Mô tả:

Ứng dụng hiển thị la bàn và hướng bắc dựa trên dữ liệu từ cảm biến từ trường và cảm biến gia tốc.

Các bước thực hiện:

1. Lấy SensorManager và Sensor:

- Lấy SensorManager từ hệ thống.
- Sử dụng SensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD) để lấy đối tượng Sensor từ trường.
- Sử dụng SensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER) để lấy đối tượng Sensor gia tốc.

2. Đăng ký SensorEventListener:

- Đăng ký một SensorEventListener để lắng nghe các sự kiện từ cảm biến từ trường và gia tốc.
- Trong phương thức onSensorChanged():
 - Lấy giá trị từ cả hai cảm biến.
 - Sử dụng các hàm SensorManager.getRotationMatrix() và SensorManager.getOrientation() để tính toán hướng bắc và góc lệch.

3. Hiển thị la bàn:

- Sử dụng một ImageView để hiển thị hình ảnh la bàn.
- Sử dụng phương thức ImageView.setRotation() để xoay hình ảnh la bàn theo hướng bắc.

4. Hiển thị góc lệch:

- Hiển thị giá trị góc lệch so với hướng bắc lên một TextView.

Lưu ý:

- **Quyền (Permissions):** Khai báo các quyền cần thiết trong AndroidManifest.xml, bao gồm quyền sử dụng cảm biến.
- **Độ chính xác:** Độ chính xác của cảm biến có thể bị ảnh hưởng bởi nhiều từ môi trường. Cần có các biện pháp lọc dữ liệu hoặc hiệu chỉnh để cải thiện độ chính xác.
- **Tiết kiệm pin:** Hủy đăng ký SensorEventListener khi không sử dụng cảm biến để tiết kiệm pin.
- **Kiểm tra cảm biến:** Kiểm tra xem thiết bị có hỗ trợ các cảm biến cần thiết hay không trước khi sử dụng.

1. Lấy SensorManager và Sensor:

- Lấy SensorManager từ hệ thống.

- o Sử dụng `SensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD)` để lấy đối tượng Sensor từ trường.
- o Sử dụng `SensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)` để lấy đối tượng Sensor gia tốc.

```

<?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.USE_FULL_SCREEN_INTENT"/>
    <uses-permission android:name="android.permission.POST_NOTIFICATIONS"/>

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="LABAN"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.LABAN"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="LABAN"
            android:theme="@style/Theme.LABAN">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
            </intent-filter>
        </activity>
    </application>
</manifest>

```

2. Đăng ký SensorEventListener:

3. Đăng ký một SensorEventListener để lắng nghe các sự kiện từ cảm biến từ trường và gia tốc.

4. Trong phương thức `onSensorChanged()`:

- Lấy giá trị từ cả hai cảm biến.
- Sử dụng các hàm `SensorManager.getRotationMatrix()` và `SensorManager.getOrientation()` để tính toán hướng bắc và góc lệch.

- **Hiển thị la bàn:**

1. Sử dụng một `ImageView` để hiển thị hình ảnh la bàn.
2. Sử dụng phương thức `ImageView.setRotation()` để xoay hình ảnh la bàn theo hướng bắc.

- **Hiển thị góc lệch:**

1. Hiển thị giá trị góc lệch so với hướng bắc lên một `TextView`.

```
LA LABAN main samsung SM-A715F app activity_main.xml
1 package com.example.laban
2
3 > import ...
4
5
6 <> class MainActivity : ComponentActivity() , SensorEventListener {
7     private lateinit var sensorManager: SensorManager
8     private var accelerometer: Sensor? = null
9     private var magnetometer: Sensor? = null
10
11     private val gravity = FloatArray( size: 3 )
12     private val geomagnetic = FloatArray( size: 3 )
13     private var azimuth = 0f // Góc lệch
14
15     private lateinit var compassNeedle: ImageView
16     private lateinit var angleText: TextView
17
18     override fun onCreate(savedInstanceState: Bundle?) {
19         super.onCreate(savedInstanceState)
20         setContentView(R.layout.activity_main)
21
22         compassNeedle = findViewById(R.id.compassNeedle)
23         angleText = findViewById(R.id.angleText)
24
25         // Lấy cảm biến
26         sensorManager = getSystemService(Context.SENSOR_SERVICE) as SensorManager
27         accelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)
28         magnetometer = sensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD)
29
30     }
31
32     override fun onResume() {
33         super.onResume()
34         accelerometer?.also { sensor ->
35             sensorManager.registerListener( listener: this, sensor, SensorManager.SENSOR_DELAY_UI )
36         }
37         magnetometer?.also { sensor ->
38             sensorManager.registerListener( listener: this, sensor, SensorManager.SENSOR_DELAY_UI )
39         }
40
41     }
42
43     override fun onPause() {
44         super.onPause()
45         sensorManager.unregisterListener( listener: this )
46     }
47
48     override fun onSensorChanged(event: SensorEvent?) {
49         if (event == null) return
50
51     }
52
53     override fun onAccuracyChanged(sensor: Sensor, accuracy: Int) {
54
55     }
56
57     override fun onConfigurationChanged(newConfig: Configuration) {
58
59     }
60
61     override fun onSensorChanged(event: SensorEvent?) {
62
63     }
64 }
```

LABAN main samsung SM-A715F app activity_main.xml

```
LA LABAN main samsung SM-A715F app activity_main.xml
1 package com.example.laban
2
3 > import ...
4
5
6 <> class MainActivity : ComponentActivity() , SensorEventListener {
7     private lateinit var sensorManager: SensorManager
8     private var accelerometer: Sensor? = null
9     private var magnetometer: Sensor? = null
10
11     private val gravity = FloatArray( size: 3 )
12     private val geomagnetic = FloatArray( size: 3 )
13     private var azimuth = 0f // Góc lệch
14
15     private lateinit var compassNeedle: ImageView
16     private lateinit var angleText: TextView
17
18     override fun onCreate(savedInstanceState: Bundle?) {
19         super.onCreate(savedInstanceState)
20         setContentView(R.layout.activity_main)
21
22         compassNeedle = findViewById(R.id.compassNeedle)
23         angleText = findViewById(R.id.angleText)
24
25         // Lấy cảm biến
26         sensorManager = getSystemService(Context.SENSOR_SERVICE) as SensorManager
27         accelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)
28         magnetometer = sensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD)
29
30     }
31
32     override fun onResume() {
33         super.onResume()
34         accelerometer?.also { sensor ->
35             sensorManager.registerListener( listener: this, sensor, SensorManager.SENSOR_DELAY_UI )
36         }
37         magnetometer?.also { sensor ->
38             sensorManager.registerListener( listener: this, sensor, SensorManager.SENSOR_DELAY_UI )
39         }
40
41     }
42
43     override fun onPause() {
44         super.onPause()
45         sensorManager.unregisterListener( listener: this )
46     }
47
48     override fun onSensorChanged(event: SensorEvent?) {
49         if (event == null) return
50
51     }
52
53     override fun onAccuracyChanged(sensor: Sensor, accuracy: Int) {
54
55     }
56
57     override fun onConfigurationChanged(newConfig: Configuration) {
58
59     }
60
61     override fun onSensorChanged(event: SensorEvent?) {
62
63     }
64 }
```

LABAN main samsung SM-A715F app activity_main.xml

The screenshot shows the Android Studio interface with the following details:

- Title Bar:** LA LABAN, main, samsung SM-A715F, app, activity_main.xml
- Main Area:** Code editor for `MainActivity.kt`. The code is as follows:

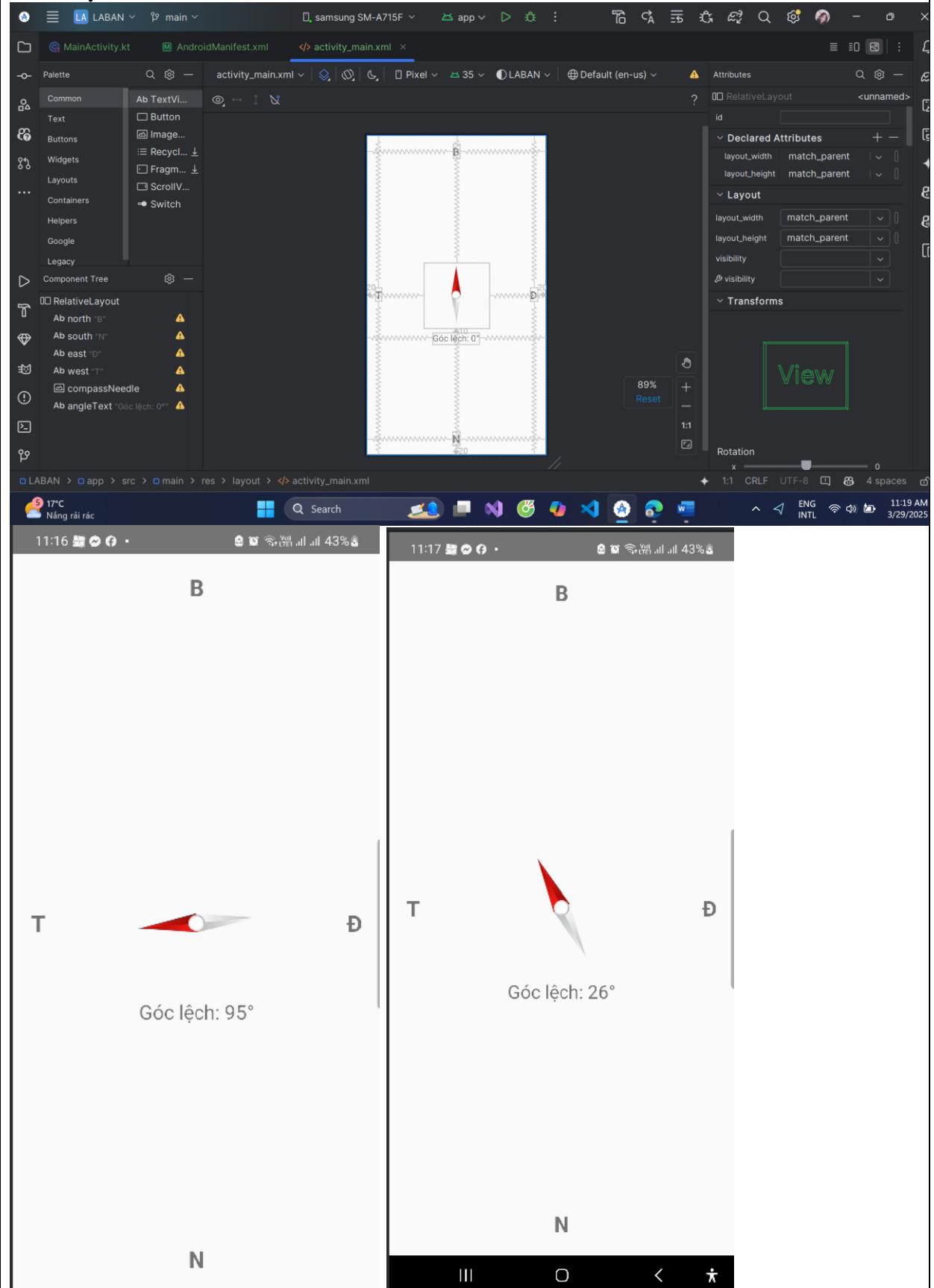
```
24     class MainActivity : ComponentActivity() , SensorEventListener {
62     }
63
64     override fun onSensorChanged(event: SensorEvent?) {
65         if (event == null) return
66
67         when (event.sensor.type) {
68             Sensor.TYPE_ACCELEROMETER -> {
69                 System.arraycopy(event.values, srcPos: 0, gravity, destPos: 0, event.values.size)
70             }
71             Sensor.TYPE_MAGNETIC_FIELD -> {
72                 System.arraycopy(event.values, srcPos: 0, geomagnetic, destPos: 0, event.values.size)
73             }
74         }
75
76         val rotationMatrix = FloatArray( size: 9)
77         val orientation = FloatArray( size: 3)
78
79         if (SensorManager.getRotationMatrix(rotationMatrix, null, gravity, geomagnetic)) {
80             SensorManager.getOrientation(rotationMatrix, orientation)
81             azimuth = Math.toDegrees(orientation[0].toDouble()).toFloat()
82             azimuth = (azimuth + 360) % 360
83
84             // Xoay kim la bàn
85             compassNeedle.rotation = -azimuth
86
87             // Cập nhật góc lệch
88             angleText.text = "Góc lệch: ${azimuth.toInt()}"
89         }
90     }
91
92     override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) {}
93 }
```

The code implements a `SensorEventListener` to handle sensor events. It uses `System.arraycopy` to copy sensor values into `gravity` and `geomagnetic` arrays. It then uses `SensorManager.getRotationMatrix` and `SensorManager.getOrientation` to calculate the rotation matrix and orientation. The azimuth angle is calculated from the orientation array. Finally, the azimuth value is used to rotate a compass needle and update a text view with the current angle.

Bottom Bar: LABAN > app > src > main > java > com > example > laban > MainActivity > accelerometer

System Tray: 6 17°C Nắng rải rác, Search, various icons, ENG INTL, 11:18 AM, 3/29/2025

Giao diện:



BÀI TẬP 10: ỦNG DỤNG CLIENT-SERVER ĐƠN GIẢN SỬ DỤNG TCP

Mục tiêu:

- Xây dựng một ứng dụng Android (Client) có thể gửi và nhận dữ liệu từ một ứng dụng Server (có thể chạy trên PC hoặc thiết bị Android khác) sử dụng giao thức TCP.
- Ứng dụng Client cho phép người dùng nhập tin nhắn và gửi đến Server.
- Ứng dụng Server nhận tin nhắn và hiển thị chúng.

Mô tả:

Ứng dụng này minh họa giao tiếp cơ bản giữa Client và Server sử dụng TCP, tập trung vào việc thiết lập kết nối, gửi và nhận dữ liệu.

BÀI TẬP 11: Ứng dụng gửi và nhận tin nhắn sử dụng UDP

Mục tiêu:

- Xây dựng một ứng dụng Android có thể gửi và nhận tin nhắn sử dụng giao thức UDP.
- Ứng dụng cho phép người dùng gửi tin nhắn đến một thiết bị khác trên mạng.
- Ứng dụng có thể nhận tin nhắn từ các thiết bị khác.

Mô tả:

Ứng dụng này minh họa giao tiếp sử dụng UDP, tập trung vào việc gửi và nhận các gói tin độc lập.

Các bước thực hiện:

Gửi tin nhắn:

1. **Lấy dữ liệu:** Lấy dữ liệu từ người dùng (ví dụ: một EditText).
2. **Tạo DatagramPacket:** Tạo một DatagramPacket chứa dữ liệu cần gửi, địa chỉ IP và cổng của người nhận.
3. **Tạo DatagramSocket:** Tạo một DatagramSocket để gửi gói tin.
4. **Gửi gói tin:** Sử dụng DatagramSocket.send() để gửi DatagramPacket.
5. **Đóng DatagramSocket:** Đóng DatagramSocket sau khi gửi.

Nhận tin nhắn:

1. **Tạo DatagramSocket:** Tạo một DatagramSocket và lắng nghe trên một cổng cụ thể.
2. **Tạo DatagramPacket:** Tạo một DatagramPacket để chứa dữ liệu nhận được.
3. **Nhận gói tin:** Sử dụng DatagramSocket.receive() để nhận DatagramPacket.
4. **Xử lý dữ liệu:** Trích xuất dữ liệu từ DatagramPacket và hiển thị.
5. **Đóng DatagramSocket:** Đóng DatagramSocket khi không còn cần nhận tin nhắn.

Lưu ý:

- **Quyền (Permissions):** Đảm bảo khai báo các quyền cần thiết trong AndroidManifest.xml, bao gồm android.permission.INTERNET.
- **Luồng (Threads):** Thực hiện các hoạt động mạng trong các luồng riêng để tránh làm treo UI thread.
- **Xử lý lỗi:** Xử lý các trường hợp lỗi có thể xảy ra, chẳng hạn như kết nối không thành công, mất kết nối, v.v.
- **Giao thức:** Xác định rõ giao thức giao tiếp giữa Client và Server (ví dụ: định dạng tin nhắn, các lệnh).