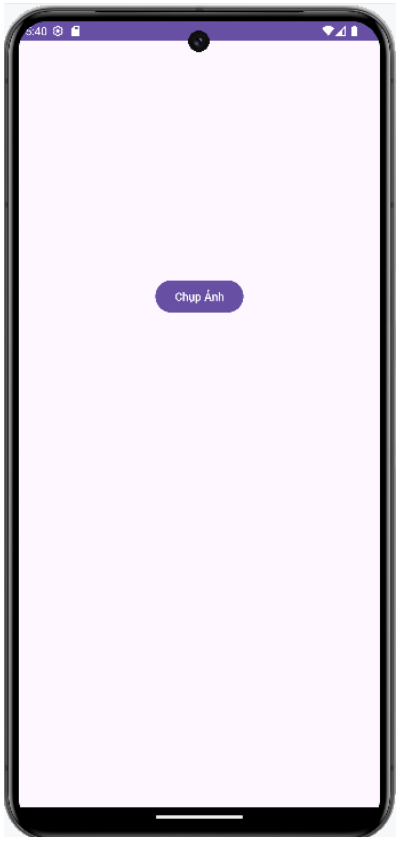
TRUY XUẤT PHẦN CỨNG

Lưu ý đối với các bài tập truy xuất phần cứng, nên dùng thiết bị thật (thay vì emulator) để có thử nghiệm tốt nhất.

BÀI TẬP 1: Chụp ảnh bằng Camera và lưu vào bộ nhớ



Hướng dẫn:

Thêm quyền vào AndroidManifest.xml

```
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-feature android:name="android.hardware.camera" android:required="true"/>
```

Tạo giao diện trong activity_main.xml và thêm một nút để mở Camera và một ImageView để hiển thị ảnh:

```xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center"
    android:padding="16dp">
    <Button
        android:id="@+id/btnCapture"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Chụp Ảnh"/>
    <ImageView
        android:id="@+id/imageView"
        android:layout_width="300dp"
        android:layout_height="300dp"
        android:scaleType="centerCrop"
        android:layout_marginTop="16dp"/>
</LinearLayout>
```

Trong MainActivity.java, thêm code để mở camera và lưu ảnh:

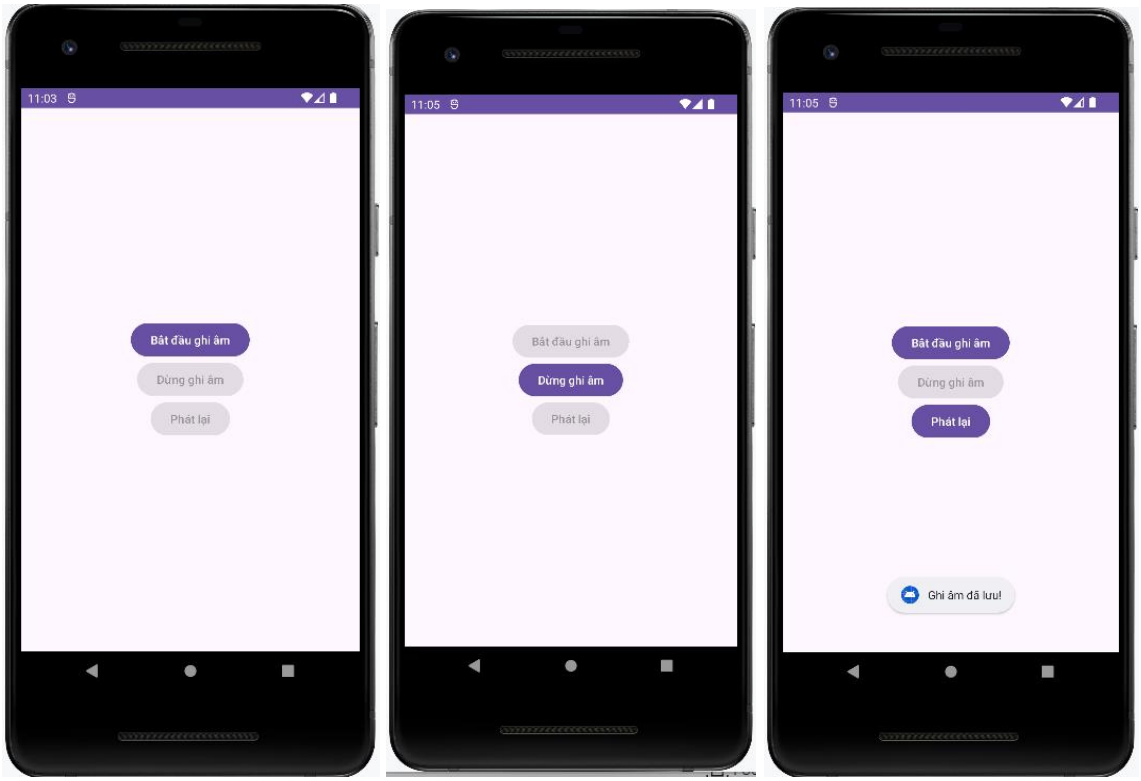
```java
public class MainActivity extends AppCompatActivity {

    2 usages
    private static final int CAMERA_PERMISSION_CODE = 100;
    2 usages
    private ImageView imageView;
    1 usage
    private Uri photoURI;
    no usages
    private File photoFile;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button btnCapture = findViewById(R.id.btnCapture);
        imageView = findViewById(R.id.imageView);

        btnCapture.setOnClickListener(v -> checkCameraPermission());
    }
```

```java
// Kiểm tra quyền CAMERA
1 usage
private void checkCameraPermission() {
    if (ContextCompat.checkSelfPermission( context: this, Manifest.permission.CAMERA)
            != PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions( activity: this,
                new String[]{Manifest.permission.CAMERA}, CAMERA_PERMISSION_CODE);
    } else {
        openCamera();
    }
}

// Xử lý kết quả cấp quyền
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
                                       @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    if (requestCode == CAMERA_PERMISSION_CODE) {
        if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            openCamera();
        } else {
            Toast.makeText( context: this, text: "Quyền Camera bị từ chối!",
                    Toast.LENGTH_SHORT).show();
        }
    }
}

// Mở Camera
2 usages
private void openCamera() {
    Intent intent = new Intent();
    intent.setAction("android.media.action.STILL_IMAGE_CAMERA");

    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    } else {
        Toast.makeText( context: this, text: "Không tìm thấy ứng dụng Camera!",
                Toast.LENGTH_SHORT).show();
    }
}
```

```
// Nhận kết quả từ Camera
no usages
private final ActivityResultLauncher<Intent> cameraLauncher = registerForActivityResult(
        new ActivityResultContracts.StartActivityForResult(),
        new ActivityResultCallback<ActivityResult>() {
            @Override
            public void onActivityResult(ActivityResult result) {
                if (result.getResultCode() == RESULT_OK) {
                    imageView.setImageURI(photoURI);
                    Toast.makeText( context: MainActivity.this,
                            text: "Ảnh đã lưu!", Toast.LENGTH_SHORT).show();
                } else {
                    Toast.makeText( context: MainActivity.this,
                            text: "Chụp ảnh thất bại!", Toast.LENGTH_SHORT).show();
                }
            }
        }
);
```

```
    // Tạo file ảnh và lưu vào bộ nhớ
    no usages
    private File createImageFile() throws IOException {
        String timeStamp = new SimpleDateFormat( pattern: "yyyyMMdd_HHmmss",
                Locale.getDefault()).format(new Date());
        String imageFileName = "IMG_" + timeStamp;
        File storageDir = getExternalFilesDir(Environment.DIRECTORY_PICTURES);
        return File.createTempFile(imageFileName,  suffix: ".jpg", storageDir);
    }
}
```

BÀI TẬP 2: Ghi âm giọng nói và phát lại



Hướng dẫn:

Thêm quyền vào AndroidManifest.xml

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

Thiết kế giao diện

```xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center"
    android:padding="16dp">
    <Button
        android:id="@+id/btnRecord"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bắt đầu ghi âm" />
    <Button
        android:id="@+id/btnStop"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Dừng ghi âm" />
    <Button
        android:id="@+id/btnPlay"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Phát lại" />
</LinearLayout>
```

Xử lý Ghi âm và phát lại

```java
import android.Manifest;
import android.content.pm.PackageManager;
import android.media.MediaPlayer;
import android.media.MediaRecorder;
import android.os.Bundle;
import android.os.Environment;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
import androidx.activity.result.contract.ActivityResultContracts;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import java.io.IOException;
```

```java
public class MainActivity extends AppCompatActivity {

    2 usages
    private static final int AUDIO_PERMISSION_CODE = 200;
    11 usages
    private MediaRecorder mediaRecorder;
    7 usages
    private MediaPlayer mediaPlayer;
    3 usages
    private String audioFilePath;

    4 usages
    private Button btnRecord, btnStop, btnPlay;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    btnRecord = findViewById(R.id.btnRecord);
    btnStop = findViewById(R.id.btnStop);
    btnPlay = findViewById(R.id.btnPlay);

    btnStop.setEnabled(false);
    btnPlay.setEnabled(false);

    btnRecord.setOnClickListener(v -> checkAudioPermission());
    btnStop.setOnClickListener(v -> stopRecording());
    btnPlay.setOnClickListener(v -> playRecording());
}

//Kiểm tra quyền Ghi âm
1 usage
private void checkAudioPermission() {
    if (ContextCompat.checkSelfPermission( context: this, Manifest.permission.RECORD_AUDIO)
            != PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions( activity: this,
                new String[]{Manifest.permission.RECORD_AUDIO}, AUDIO_PERMISSION_CODE);
    } else {
        startRecording();
    }
}
```

```java
//Xử lý kết quả cấp quyền
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
                                       @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    if (requestCode == AUDIO_PERMISSION_CODE) {
        if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            startRecording();
        } else {
            Toast.makeText( context: this, text: "Quyền ghi âm bị từ chối!",
                    Toast.LENGTH_SHORT).show();
        }
    }
}

// Bắt đầu ghi âm
2 usages
private void startRecording() {
    try {
        audioFilePath = getExternalFilesDir(Environment.DIRECTORY_MUSIC)
                .getAbsolutePath() + "/recorded_audio.3gp";
        mediaRecorder = new MediaRecorder();
        mediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
        mediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
        mediaRecorder.setOutputFile(audioFilePath);
        mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
        mediaRecorder.prepare();
        mediaRecorder.start();

        btnRecord.setEnabled(false);
        btnStop.setEnabled(true);
        btnPlay.setEnabled(false);

        Toast.makeText( context: this, text: "Đang ghi âm...", Toast.LENGTH_SHORT).show();
    } catch (IOException e) {
        e.printStackTrace();
        Toast.makeText( context: this, text: "Lỗi khi bắt đầu ghi âm!",
                Toast.LENGTH_SHORT).show();
    }
}
```

```java
// Dừng ghi âm
1 usage
private void stopRecording() {
    if (mediaRecorder != null) {
        mediaRecorder.stop();
        mediaRecorder.release();
        mediaRecorder = null;

        btnRecord.setEnabled(true);
        btnStop.setEnabled(false);
        btnPlay.setEnabled(true);

        Toast.makeText( context: this,  text: "Ghi âm đã lưu!", Toast.LENGTH_SHORT).show();
    }
}
```

```java
    // Phát lại ghi âm
    1 usage
    private void playRecording() {
        mediaPlayer = new MediaPlayer();
        try {
            mediaPlayer.setDataSource(audioFilePath);
            mediaPlayer.prepare();
            mediaPlayer.start();
            Toast.makeText( context: this,  text: "Đang phát lại...", Toast.LENGTH_SHORT).show();
        } catch (IOException e) {
            e.printStackTrace();
            Toast.makeText( context: this,  text: "Lỗi khi phát lại!", Toast.LENGTH_SHORT).show();
        }
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (mediaPlayer != null) {
            mediaPlayer.release();
            mediaPlayer = null;
        }
    }
}
```

Chạy ứng dụng.

BÀI TẬP 3: Hiển thị tin tức từ RSS Feed lên ứng dụng

Hướng dẫn:

Thêm quyền vào AndroidManifest.xml

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

Giao diện danh sách tin tức

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

Đọc RSS Feed

```
import android.os.Bundle;
import android.os.Handler;
import android.os.Looper;
import android.util.Xml;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.util.Log;
import androidx.appcompat.app.AppCompatActivity;

import org.xmlpull.v1.XmlPullParser;

import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.ArrayList;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
```

```java
public class MainActivity extends AppCompatActivity {
    2 usages
    private ListView listView;
    3 usages
    private ArrayAdapter<String> adapter;
    3 usages
    private ArrayList<String> titles = new ArrayList<>();
    3 usages
    private ExecutorService executorService;
    2 usages
    private Handler handler;

@Override                                                    ⚠ 3  ✓ 8
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    listView = findViewById(R.id.listView);
    adapter = new ArrayAdapter<>( context: this, android.R.layout.simple_list_item_1, titles);
    listView.setAdapter(adapter);

    // Khởi tạo ExecutorService và Handler
    executorService = Executors.newSingleThreadExecutor();
    handler = new Handler(Looper.getMainLooper());

    // Gọi hàm lấy dữ liệu RSS
    fetchRSS( urlString: "https://vnexpress.net/rss/tin-moi-nhat.rss");
}

private void fetchRSS(String urlString) {
    executorService.execute(() -> {
        ArrayList<String> fetchedTitles = new ArrayList<>();
        try {
            Log.d( tag: "RSS",  msg: "Fetching from URL: " + urlString); // Kiểm tra URL
            URL url = new URL(urlString);
            HttpURLConnection connection = (HttpURLConnection) url.openConnection();
            connection.setRequestMethod("GET");
            connection.setConnectTimeout(5000);
            connection.setReadTimeout(5000);
            connection.setDoInput(true);

            int responseCode = connection.getResponseCode();
            // Kiểm tra phản hồi HTTP
            Log.d( tag: "RSS",  msg: "Response Code: " + responseCode);
```

```
if (responseCode == HttpURLConnection.HTTP_OK) {                    ⚠ 3  ✓ 6
    try (InputStream inputStream = connection.getInputStream()) {
        XmlPullParser parser = Xml.newPullParser();
        parser.setInput(inputStream,  inputEncoding: "UTF-8");

        boolean insideItem = false;
        String title = "";
        int eventType = parser.getEventType();

        while (eventType != XmlPullParser.END_DOCUMENT) {
            if (eventType == XmlPullParser.START_TAG) {
                if ("item".equalsIgnoreCase(parser.getName())) {
                    insideItem = true;
                } else if (insideItem && "title".equalsIgnoreCase(parser.getName())) {
                    title = parser.nextText();
                    Log.d( tag: "RSS",  msg: "Fetched Title: " + title);
                }
            } else if (eventType == XmlPullParser.END_TAG && "item".
                    equalsIgnoreCase(parser.getName())) {
                fetchedTitles.add(title);
                insideItem = false;
            }
            eventType = parser.next();
        }
    }
```

```
                }
            } else {
                Log.e( tag: "RSS",  msg: "HTTP Error: " + responseCode);
            }
        } catch (Exception e) {
            // Log lỗi chi tiết
            Log.e( tag: "RSS",  msg: "Error fetching RSS: " + e.getMessage());
        }

        handler.post(() -> {
            if (!fetchedTitles.isEmpty()) {
                titles.clear();
                titles.addAll(fetchedTitles);
                adapter.notifyDataSetChanged();
            } else {
                // Nếu không có dữ liệu
                Log.e( tag: "RSS",  msg: "No data fetched! Check network & RSS format.");
            }
        });
    });
}

    @Override
    protected void onDestroy() {
        super.onDestroy();
        executorService.shutdown(); // Đóng ExecutorService để giải phóng tài nguyên
    }
}
```