**TRƯỜNG ĐẠI HỌC TÀI NGUYÊN VÀ MÔI TRƯỜNG TPHCM**

**KHOA CÔNG NGHỆ THÔNG TIN**



**TỔNG HỢP LAB**

**LẬP TRÌNH ỨNG DỤNG DI ĐỘNG**

Giảng viên hướng dẫn: **ThS. Phạm Trọng Huynh**

Sinh viên thực hiện: **Ngô Gia Bảo**

Mã số sinh viên: 1150080003

Lớp: **11\_THMT**

Khóa: **K11**

***TP. Hồ Chí Minh, năm học 2025 - 2026***

**TRƯỜNG ĐẠI HỌC TÀI NGUYÊN VÀ MÔI TRƯỜNG TPHCM**

**KHOA CÔNG NGHỆ THÔNG TIN**



**TỔNG HỢP LAB**

**LẬP TRÌNH ỨNG DỤNG DI ĐỘNG**

Giảng viên hướng dẫn: **ThS. Phạm Trọng Huynh**

Sinh viên thực hiện: **Ngô Gia Bảo**

Mã số sinh viên: 1150080003

Lớp: **11\_THMT**

Khóa: **K11**

***TP. Hồ Chí Minh, năm học 2025 - 2026***

**NHẬN XÉT**

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

**MỤC LỤC**

[LAB THỰC HÀNH 1 1](#_Toc215566536)

[1.1 Bài tập 1 1](#_Toc215566537)

[1.1.1 Thiết kế Giao diện (activity\_main.xml) 1](#_Toc215566538)

[1.1.2 Xử lý Logic (MainActivity.java) 5](#_Toc215566539)

[1.2 Bài tập 2: Ứng dụng Random Số / Tung Xúc Xắc 13](#_Toc215566540)

[1.2.1 Thiết kế Giao diện (activity\_main.xml): 13](#_Toc215566541)

[1.2.2 Code Giao diện (activity\_main.xml): 13](#_Toc215566542)

[1.2.3 Xử lý Logic (MainActivity.java): 14](#_Toc215566543)

[1.2.4 Code Xử lý (MainActivity.java): 14](#_Toc215566544)

[1.3 Bài tập Lab02 16](#_Toc215566545)

[1.3.1 Thiết kế Layout màn hình chờ (item\_loading.xml): 16](#_Toc215566546)

[1.3.2 Tích hợp giao diện (activity\_main.xml): 17](#_Toc215566547)

[1.3.3 Ứng dụng Random (Giao diện): 19](#_Toc215566548)

[1.3.4 Xử lý Logic Random (MainActivity.java): 21](#_Toc215566549)

[LAB THỰC HÀNH 2 24](#_Toc215566550)

[2.1 Bài tập thực hành 24](#_Toc215566552)

[2.1.1 Thiết kế Màn hình Chào (Splash Screen): 24](#_Toc215566553)

[2.1.2 Xử lý Logic Màn hình Chào 26](#_Toc215566554)

[2.2 Bài tập 1 27](#_Toc215566555)

[2.2.1 Giao diện điều khiển (M001MainFrg): 28](#_Toc215566556)

[2.2.2 Xử lý Logic Khóa xoay (Orientation Lock): 29](#_Toc215566557)

[2.3 Bài tập 2 33](#_Toc215566558)

[2.3.1 Giao diện Đăng nhập Đa ngôn ngữ 34](#_Toc215566559)

[2.3.2 Xử lý Logic Thay đổi Ngôn ngữ (setLocale): 39](#_Toc215566560)

[LAB THỰC HÀNH 3 44](#_Toc215566561)

[3.1 LAYOUT AND APPLICATION 44](#_Toc215566563)

[3.2 Bài tập 1 49](#_Toc215566564)

[3.3 Bài tập 2: Custom controls 54](#_Toc215566565)

[3.4 Bài tập 3 61](#_Toc215566566)

[LAB THỰC HÀNH 4 63](#_Toc215566567)

[4.1 Khởi tạo và chuẩn bị tài nguyên 63](#_Toc215566569)

[4.2 KẾT QUẢ THỰC HIỆN 66](#_Toc215566570)

[LAB THỰC HÀNH 5 72](#_Toc215566571)

[5.1 LAB 13-14: ỨNG DỤNG ACCOUNT 72](#_Toc215566573)

[5.2 Xây dựng Layout (Giao diện) 75](#_Toc215566574)

[5.3 Xử lý Logic (Controller) 77](#_Toc215566575)

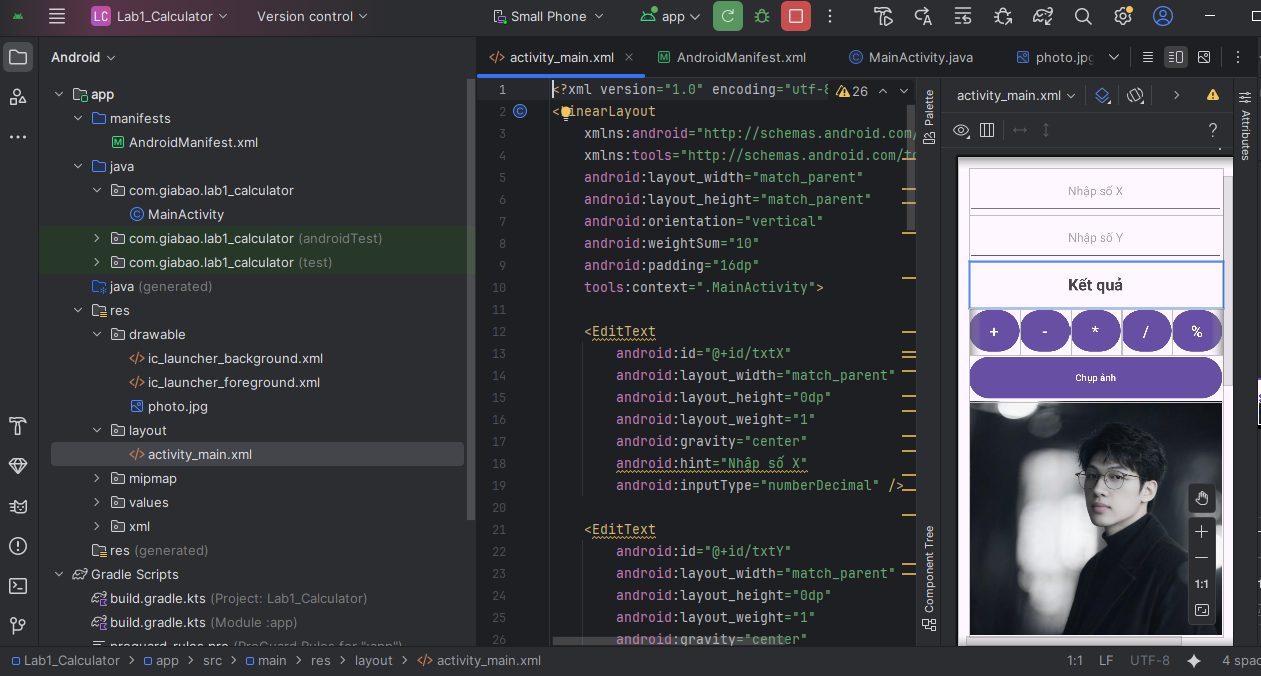
[5.4 Kết quả 80](#_Toc215566576)

[5.5 Tìm hiểu về SharedPreferences 86](#_Toc215566577)

# LAB THỰC HÀNH 1

## Bài tập 1

### Thiết kế Giao diện (activity\_main.xml)



Để thiết kế giao diện máy tính đơn giản, em sử dụng **LinearLayout** với thuộc tính orientation="vertical" để sắp xếp các thành phần theo chiều dọc từ trên xuống dưới:

* **Phân chia tỷ lệ:** Em sử dụng thuộc tính layout\_weight kết hợp với weightSum="10" ở layout cha. Việc này giúp chia màn hình thành các phần cân đối: 2 phần cho ô nhập liệu (EditText), 1 phần hiển thị kết quả (TextView), 1 phần cho hàng nút bấm và phần còn lại dành cho hình ảnh cá nhân.
* **Hàng nút bấm:** Để các nút phép tính (+, -, \*, /, %) nằm ngang hàng nhau, em lồng ghép thêm một LinearLayout con (horizontal) và chia đều tỷ lệ weight cho từng nút, giúp giao diện hiển thị đều đẹp trên mọi kích thước màn hình.
* **Hình ảnh:** Sử dụng ImageView với scaleType="centerCrop" để hiển thị ảnh chân dung cân đối ở phần dưới cùng màn hình.

**Code activity\_main.xml:**

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:weightSum="10"  
 android:padding="16dp"  
 tools:context=".MainActivity">  
  
 <EditText  
 android:id="@+id/txtX"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:layout\_weight="1"  
 android:gravity="center"  
 android:hint="Nhập số X"  
 android:inputType="numberDecimal" />  
  
 <EditText  
 android:id="@+id/txtY"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:layout\_weight="1"  
 android:gravity="center"  
 android:hint="Nhập số Y"  
 android:inputType="numberDecimal" />  
  
 <TextView  
 android:id="@+id/txtResult"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:layout\_weight="1"  
 android:gravity="center"  
 android:text="Kết quả"  
 android:textSize="24sp"  
 android:textStyle="bold" />  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:layout\_weight="1"  
 android:orientation="horizontal"  
 android:weightSum="5">  
  
 <Button  
 android:id="@+id/btnPlus"  
 android:layout\_width="0dp"  
 android:layout\_height="match\_parent"  
 android:layout\_weight="1"  
 android:text="+"  
 android:textSize="24sp" />  
  
 <Button  
 android:id="@+id/btnMinus"  
 android:layout\_width="0dp"  
 android:layout\_height="match\_parent"  
 android:layout\_weight="1"  
 android:text="-"  
 android:textSize="24sp" />  
  
 <Button  
 android:id="@+id/btnMultiply"  
 android:layout\_width="0dp"  
 android:layout\_height="match\_parent"  
 android:layout\_weight="1"  
 android:text="\*"  
 android:textSize="24sp" />  
  
 <Button  
 android:id="@+id/btnDivide"  
 android:layout\_width="0dp"  
 android:layout\_height="match\_parent"  
 android:layout\_weight="1"  
 android:text="/"  
 android:textSize="24sp" />  
  
 <Button  
 android:id="@+id/btnMod"  
 android:layout\_width="0dp"  
 android:layout\_height="match\_parent"  
 android:layout\_weight="1"  
 android:text="%"  
 android:textSize="24sp" />  
  
 </LinearLayout>  
  
 <Button  
 android:id="@+id/btnCamera"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:layout\_weight="1"  
 android:text="Chụp ảnh" />  
  
 <ImageView  
 android:id="@+id/imgPhoto"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:layout\_weight="5"  
 android:src="@drawable/photo"  
 android:scaleType="centerCrop" />  
  
</LinearLayout>

### Xử lý Logic (MainActivity.java)

Trong phần xử lý code, em áp dụng các kỹ thuật mới của Android để code gọn và hiện đại hơn:

* ViewBinding: Em sử dụng ActivityMainBinding để tham chiếu trực tiếp đến các thành phần giao diện, thay thế cho phương thức findViewById cũ, giúp code ngắn gọn và tránh lỗi NullPointer.
* Xử lý Camera & Quyền: Thay vì dùng startActivityForResult (đã deprecated), em sử dụng ActivityResultLauncher để đăng ký callback nhận kết quả chụp ảnh và xử lý xin cấp quyền Camera (Manifest.permission.CAMERA) ngay tại thời điểm chạy (Runtime Permission).

Logic Máy tính: Em xây dựng một hàm xử lý chung là calculate(char operator) nhận vào toán tử (+, -, \*, /, %). Hàm này bao gồm:

* Validation: Kiểm tra ô nhập liệu rỗng và bắt lỗi nhập liệu sai định dạng bằng try-catch.
* Xử lý toán học: Thực hiện phép tính dựa trên toán tử truyền vào, đồng thời xử lý các ngoại lệ logic như lỗi "chia cho 0".
* Định dạng kết quả: Tự động kiểm tra nếu kết quả là số nguyên thì hiển thị dạng nguyên (ví dụ: 5), nếu là số thực thì hiển thị dạng thập phân (ví dụ: 5.5).

**Code xử lý:**

package com.giabao.lab1\_calculator;  
  
import android.Manifest;  
import android.content.Intent;  
import android.content.pm.PackageManager;  
import android.graphics.Bitmap;  
import android.os.Bundle;  
import android.provider.MediaStore;  
import android.view.View;  
import android.widget.Toast;  
  
import androidx.activity.result.ActivityResult;  
import androidx.activity.result.ActivityResultCallback;  
import androidx.activity.result.ActivityResultLauncher;  
import androidx.activity.result.contract.ActivityResultContracts;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.core.content.ContextCompat;  
  
import com.giabao.lab1\_calculator.databinding.ActivityMainBinding;  
  
public class MainActivity extends AppCompatActivity {  
  
 private ActivityMainBinding binding;  
 private ActivityResultLauncher<Intent> cameraLauncher;  
 private ActivityResultLauncher<String> permissionLauncher;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 binding = ActivityMainBinding.*inflate*(getLayoutInflater());  
 setContentView(binding.getRoot());  
  
 registerCameraLauncher();  
 registerPermissionLauncher();  
 setupButtonListeners();  
 }  
  
 private void registerCameraLauncher() {  
 cameraLauncher = registerForActivityResult(  
 new ActivityResultContracts.StartActivityForResult(),  
 new ActivityResultCallback<ActivityResult>() {  
 @Override  
 public void onActivityResult(ActivityResult result) {  
 if (result.getResultCode() == *RESULT\_OK* && result.getData() != null) {  
 Bundle extras = result.getData().getExtras();  
 if (extras != null) {  
 Bitmap imageBitmap = (Bitmap) extras.get("data");  
 binding.imgPhoto.setImageBitmap(imageBitmap);  
 }  
 }  
 }  
 }  
 );  
 }  
  
 private void registerPermissionLauncher() {  
 permissionLauncher = registerForActivityResult(  
 new ActivityResultContracts.RequestPermission(),  
 new ActivityResultCallback<Boolean>() {  
 @Override  
 public void onActivityResult(Boolean isGranted) {  
 if (isGranted) {  
 launchCamera();  
 } else {  
 Toast.*makeText*(MainActivity.this, "Bạn đã từ chối cấp quyền camera", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 }  
 );  
 }  
  
 private void launchCamera() {  
 Intent cameraIntent = new Intent(MediaStore.*ACTION\_IMAGE\_CAPTURE*);  
  
 if (cameraIntent.resolveActivity(getPackageManager()) != null) {  
 cameraLauncher.launch(cameraIntent);  
 } else {  
 Toast.*makeText*(MainActivity.this, "Không tìm thấy ứng dụng camera", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
  
 private void setupButtonListeners() {  
 binding.btnPlus.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 calculate('+');  
 }  
 });  
  
 binding.btnMinus.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 calculate('-');  
 }  
 });  
  
 binding.btnMultiply.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 calculate('\*');  
 }  
 });  
  
 binding.btnDivide.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 calculate('/');  
 }  
 });  
  
 binding.btnMod.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 calculate('%');  
 }  
 });  
  
 binding.btnCamera.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (ContextCompat.*checkSelfPermission*(MainActivity.this, Manifest.permission.*CAMERA*) == PackageManager.*PERMISSION\_GRANTED*) {  
 launchCamera();  
 } else {  
 permissionLauncher.launch(Manifest.permission.*CAMERA*);  
 }  
 }  
 });  
 }  
  
 private void calculate(char operator) {  
 String strX = binding.txtX.getText().toString();  
 String strY = binding.txtY.getText().toString();  
  
 if (strX.isEmpty() || strY.isEmpty()) {  
 Toast.*makeText*(this, "Vui lòng nhập cả hai số X và Y", Toast.*LENGTH\_SHORT*).show();  
 return;  
 }  
  
 try {  
 double x = Double.*parseDouble*(strX);  
 double y = Double.*parseDouble*(strY);  
 double result = 0;  
  
 switch (operator) {  
 case '+':  
 result = x + y;  
 break;  
 case '-':  
 result = x - y;  
 break;  
 case '\*':  
 result = x \* y;  
 break;  
 case '/':  
 if (y == 0) {  
 Toast.*makeText*(this, "Lỗi: Không thể chia cho 0", Toast.*LENGTH\_SHORT*).show();  
 return;  
 }  
 result = x / y;  
 break;  
 case '%':  
 if (y == 0) {  
 Toast.*makeText*(this, "Lỗi: Không thể chia cho 0", Toast.*LENGTH\_SHORT*).show();  
 return;  
 }  
 result = x % y;  
 break;  
 }  
  
 if (result == (long) result) {  
 binding.txtResult.setText(String.*format*("%d", (long) result));  
 } else {  
 binding.txtResult.setText(String.*format*("%s", result));  
 }  
  
 } catch (NumberFormatException e) {  
 Toast.*makeText*(this, "Lỗi: Dữ liệu nhập không hợp lệ", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
}

## Bài tập 2: Ứng dụng Random Số / Tung Xúc Xắc

### Thiết kế Giao diện (activity\_main.xml):

Giao diện của ứng dụng Random số được thiết kế tối giản gồm hai thành phần chính:

* **Hiển thị kết quả (TextView):** Em sử dụng một TextView lớn (textSize="40sp") đặt ở giữa màn hình để hiển thị con số ngẫu nhiên. Thuộc tính android:textStyle="bold" và màu đỏ (holo\_red\_dark) giúp kết quả nổi bật và dễ nhìn.
* **Nút bấm (Button):** Một nút "RANDOM NUMBER" được đặt phía dưới để người dùng tương tác. Khi bấm nút này, sự kiện click sẽ kích hoạt hàm sinh số ngẫu nhiên.

### Code Giao diện (activity\_main.xml):

<TextView

android:id="@+id/txtRandomNumber"

android:layout\_width="match\_parent"

android:layout\_height="0dp"

android:layout\_weight="1"

android:gravity="center"

android:text="00"

android:textColor="@android:color/holo\_red\_dark"

android:textSize="40sp"

android:textStyle="bold" />

<Button

android:id="@+id/btnRandom"

android:layout\_width="match\_parent"

android:layout\_height="0dp"

android:layout\_weight="1"

android:text="RANDOM NUMBER" />

### Xử lý Logic (MainActivity.java):

Để thực hiện chức năng sinh số ngẫu nhiên (mô phỏng việc tung xúc xắc), em sử dụng lớp thư viện java.util.Random:

* **Khởi tạo:** Đối tượng Random được khai báo toàn cục để sử dụng lại mỗi khi cần sinh số mới.
* **Hàm rollDice():** Đây là hàm xử lý chính. Em sử dụng phương thức random.nextInt(6) để tạo ra một số nguyên ngẫu nhiên trong khoảng từ 0 đến 5. Sau đó, cộng thêm 1 để được kết quả từ 1 đến 6 (tương ứng với 6 mặt của xúc xắc).
* **Cập nhật UI:** Kết quả cuối cùng được chuyển sang dạng chuỗi (String.valueOf) và cập nhật lên txtRandomNumber để người dùng nhìn thấy.

### Code Xử lý (MainActivity.java):

binding.btnRandom.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

rollDice();

}

});

private void rollDice() {

int randomNumber = random.nextInt(6) + 1;

binding.txtRandomNumber.setText(String.valueOf(randomNumber));

}

A screenshot of a person

AI-generated content may be incorrect.

## Bài tập Lab02

### Thiết kế Layout màn hình chờ (item\_loading.xml):

Layout này được thiết kế để làm một lớp phủ (overlay) thông báo trạng thái đang tải. Em sử dụng LinearLayout với thuộc tính background="#BF000000" (màu đen bán trong suốt) để làm tối màn hình phía sau, kết hợp với ProgressBar để hiển thị vòng xoay chờ xử lý.

**Code Giao diện (item\_loading.xml):**

<?xml version="1.0" encoding="utf-8"?>

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="#BF000000"

android:gravity="center"

android:orientation="vertical">

<ProgressBar

android:layout\_width="60dp"

android:layout\_height="60dp"

android:indeterminateTint="@color/white" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:fontFamily="sans-serif-light"

android:text="@string/txt\_loading"

android:textColor="@color/white"

android:textSize="24sp" />

</LinearLayout>

### Tích hợp giao diện (activity\_main.xml):

Để hiển thị màn hình chờ đè lên nội dung chính, em sử dụng **FrameLayout** (Layout xếp chồng). Thẻ **<include>** được sử dụng để nhúng layout item\_loading.xml vừa thiết kế vào, giúp tái sử dụng mã nguồn và dễ dàng ẩn/hiện khi cần thiết.

**Code Giao diện (activity\_main.xml):**<?xml version="1.0" encoding="utf-8"?>

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

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="@color/design\_default\_color\_primary\_dark"

tools:context=".MainActivity">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="@color/design\_default\_color\_primary\_dark"

android:gravity="center"

android:orientation="vertical">

<ImageView

android:layout\_width="128dp"

android:layout\_height="128dp"

android:src="@drawable/ic\_penguin"

tools:ignore="ContentDescription" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginTop="10dp"

android:fontFamily="sans-serif-light"

android:text="@string/app\_name"

android:textColor="@color/white"

android:textSize="24sp" />

</LinearLayout>

<include layout="@layout/item\_loading" />

</FrameLayout>

### Ứng dụng Random (Giao diện):

Giao diện chính bao gồm một LinearLayout đóng vai trò là hình nền (đặt ID là main\_background) và một ImageView ở giữa (đặt ID là img\_animal\_icon). Các ID này là cần thiết để code Java có thể tham chiếu và thay đổi thuộc tính của chúng.

**Code Giao diện:**

<?xml version="1.0" encoding="utf-8"?>

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

xmlns:tools="http://schemas.android.com/tools"

android:id="@+id/root\_layout"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="@color/design\_default\_color\_primary\_dark"

tools:context=".MainActivity">

<LinearLayout

android:id="@+id/main\_background"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="@color/design\_default\_color\_primary\_dark"

android:gravity="center"

android:orientation="vertical">

<ImageView

android:id="@+id/img\_animal\_icon"

android:layout\_width="128dp"

android:layout\_height="128dp"

android:src="@drawable/ic\_penguin"

tools:ignore="ContentDescription" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginTop="10dp"

android:fontFamily="sans-serif-light"

android:text="@string/app\_name"

android:textColor="@color/white"

android:textSize="24sp" />

</LinearLayout>

</FrameLayout>

### Xử lý Logic Random (MainActivity.java):

Để thực hiện chức năng thay đổi ngẫu nhiên màu sắc và hình ảnh mỗi khi mở ứng dụng, em sử dụng mảng và lớp Random:

* **Khởi tạo dữ liệu:** Em tạo hai mảng: mảng icons chứa ID của các hình ảnh con vật (trong drawable) và mảng colors chứa các mã màu hexa.
* **Sinh ngẫu nhiên:** Sử dụng đối tượng Random và phương thức nextInt(length) để lấy ra một chỉ số vị trí (index) ngẫu nhiên từ mảng.
* **Cập nhật giao diện:** Dựa vào chỉ số vừa random được, em dùng phương thức setBackgroundColor để đổi màu nền và setImageResource để đổi icon con vật tương ứng.

**Code Xử lý (MainActivity.java):**

package com.giabao.lab2\_ui\_design;

import android.graphics.Color;

import android.os.Bundle;

import android.widget.ImageView;

import android.widget.LinearLayout;

import androidx.appcompat.app.AppCompatActivity;

import java.util.Random;

public class MainActivity extends AppCompatActivity {

private Random random = new Random();

private int[] icons = {

R.drawable.ic\_penguin,

R.drawable.ic\_cat,

R.drawable.ic\_bug

};

private int[] colors = {

Color.parseColor("#FF3700B3"),

Color.parseColor("#FF006400"),

Color.parseColor("#FF8B0000"),

Color.parseColor("#FF00008B")

};

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

LinearLayout mainBackground = findViewById(R.id.main\_background);

ImageView animalIcon = findViewById(R.id.img\_animal\_icon);

int iconIndex = random.nextInt(icons.length);

int colorIndex = random.nextInt(colors.length);

mainBackground.setBackgroundColor(colors[colorIndex]);

animalIcon.setImageResource(icons[iconIndex]);

}

}

A screenshot of a phone

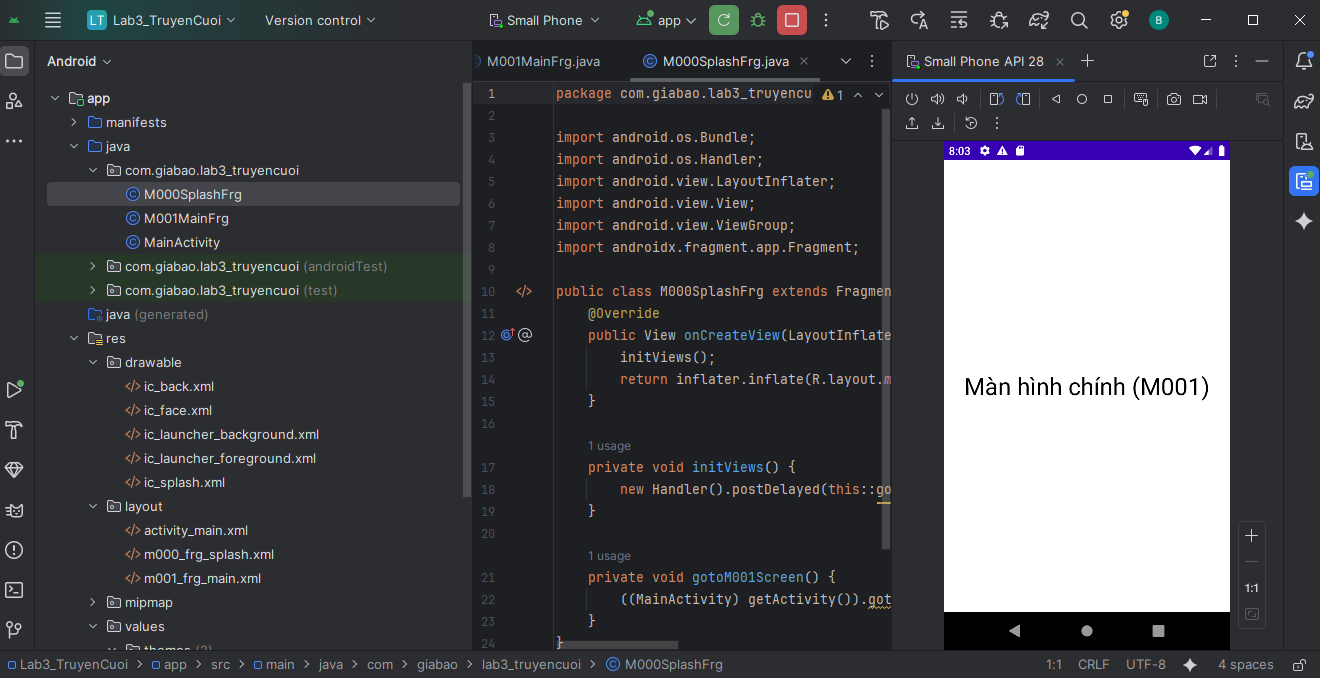
AI-generated content may be incorrect.

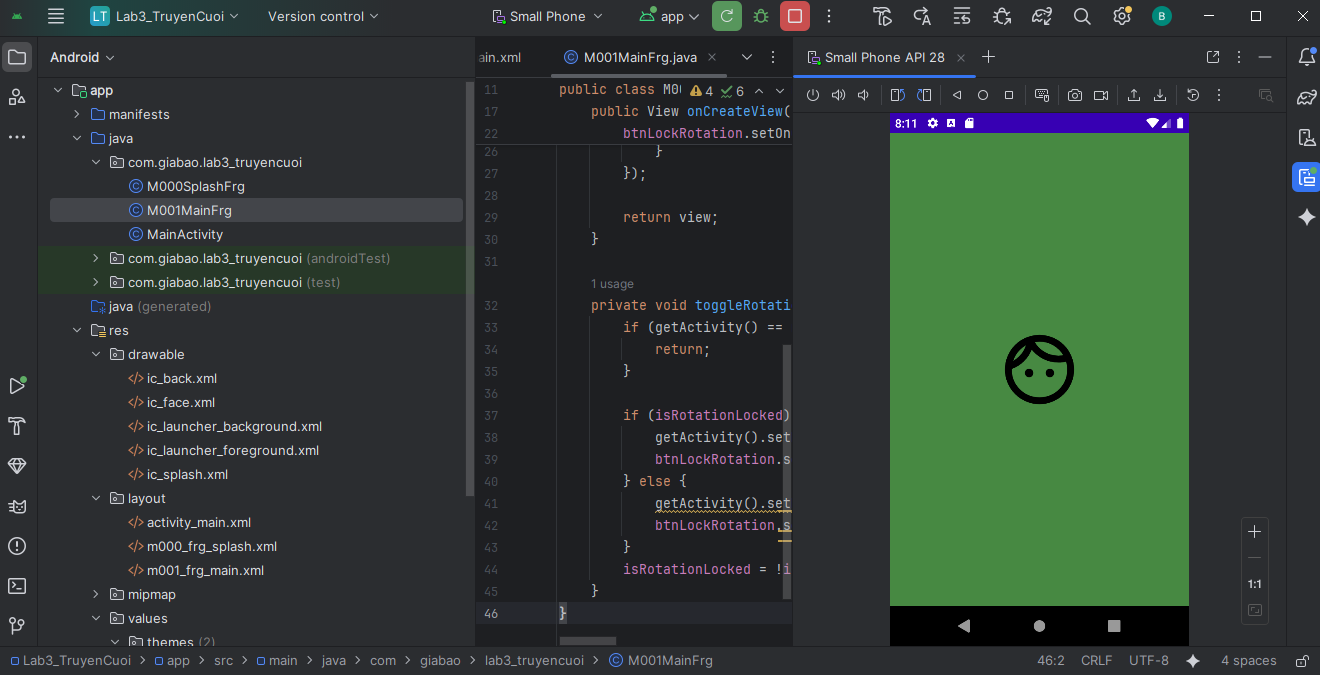
# LAB THỰC HÀNH 2



## Bài tập thực hành

### Thiết kế Màn hình Chào (Splash Screen):





Màn hình chào được thiết kế đơn giản sử dụng FrameLayout để căn giữa Logo (ImageView) trên nền màu xanh (@color/greenMid). Đây là màn hình đầu tiên người dùng nhìn thấy khi mở ứng dụng, giúp tạo ấn tượng thương hiệu và chờ tải dữ liệu.

**Code giao diện (m000\_frg\_splash.xml):**

<?xml version="1.0" encoding="utf-8"?>

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

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="@color/greenMid">

<ImageView

android:layout\_width="200dp"

android:layout\_height="100dp"

android:layout\_gravity="center"

android:src="@drawable/ic\_splash"

tools:ignore="ContentDescription" />

</FrameLayout>

### Xử lý Logic Màn hình Chào

Thay vì để người dùng phải bấm nút, em sử dụng kỹ thuật Handler kết hợp với phương thức postDelayed. Code sẽ đếm ngược 2000ms (2 giây) sau đó tự động gọi lệnh chuyển hướng sang màn hình chính (gotoM001Screen), tạo trải nghiệm mượt mà khi khởi động App.

**Code Logic (M000SplashFrg.java):** (tự động chuyển màn hình sau 2 giây)

package com.giabao.lab3\_truyencuoi;

import android.os.Bundle;

import android.os.Handler;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import androidx.fragment.app.Fragment;

public class M000SplashFrg extends Fragment {

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

initViews();

return inflater.inflate(R.layout.m000\_frg\_splash, container, false);

}

private void initViews() {

new Handler().postDelayed(this::gotoM001Screen, 2000);

}

private void gotoM001Screen() {

((MainActivity) getActivity()).gotoM001Screen();

}

}

## Bài tập 1

A screenshot of a computer

AI-generated content may be incorrect.

### Giao diện điều khiển (M001MainFrg):

**Code Giao diện (m001\_frg\_main.xml):**

<?xml version="1.0" encoding="utf-8"?>

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:gravity="center"

android:orientation="vertical"

android:background="#FFFFFF">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Màn hình chính (M001)"

android:textSize="30sp"

android:textColor="#000000"/>

<Button

android:id="@+id/btn\_lock\_rotation"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginTop="30dp"

android:text="Khóa Xoay (Đang TẮT)" />

</LinearLayout>

### Xử lý Logic Khóa xoay (Orientation Lock):

Để thực hiện chức năng khóa xoay, em sử dụng phương thức setRequestedOrientation của Activity:

* **Khi khóa (Bật):** Thiết lập thuộc tính SCREEN\_ORIENTATION\_PORTRAIT, buộc màn hình luôn ở chế độ dọc dù người dùng có xoay điện thoại.
* **Khi mở (Tắt):** Thiết lập SCREEN\_ORIENTATION\_UNSPECIFIED, cho phép màn hình tự động xoay theo cảm biến của thiết bị.
* Biến isRotationLocked được dùng để lưu trạng thái hiện tại và cập nhật text trên nút bấm cho phù hợp.

**Code Logic (M001MainFrg.java):** (Xử lý logic khi nhấn nút "Khóa Xoay")

package com.giabao.lab3\_truyencuoi;

import android.content.pm.ActivityInfo;

import android.os.Bundle;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.Button;

import androidx.fragment.app.Fragment;

public class M001MainFrg extends Fragment {

private boolean isRotationLocked = false;

private Button btnLockRotation;

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

View view = inflater.inflate(R.layout.m001\_frg\_main, container, false);

btnLockRotation = view.findViewById(R.id.btn\_lock\_rotation);

btnLockRotation.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

toggleRotationLock();

}

});

return view;

}

private void toggleRotationLock() {

if (getActivity() == null) {

return;

}

if (isRotationLocked) {

getActivity().setRequestedOrientation(ActivityInfo.SCREEN\_ORIENTATION\_UNSPECIFIED);

btnLockRotation.setText("Khóa Xoay (Đang TẮT)");

} else {

getActivity().setRequestedOrientation(ActivityInfo.SCREEN\_ORIENTATION\_PORTRAIT);

btnLockRotation.setText("Khóa Xoay (Đang BẬT)");

}

isRotationLocked = !isRotationLocked;

}

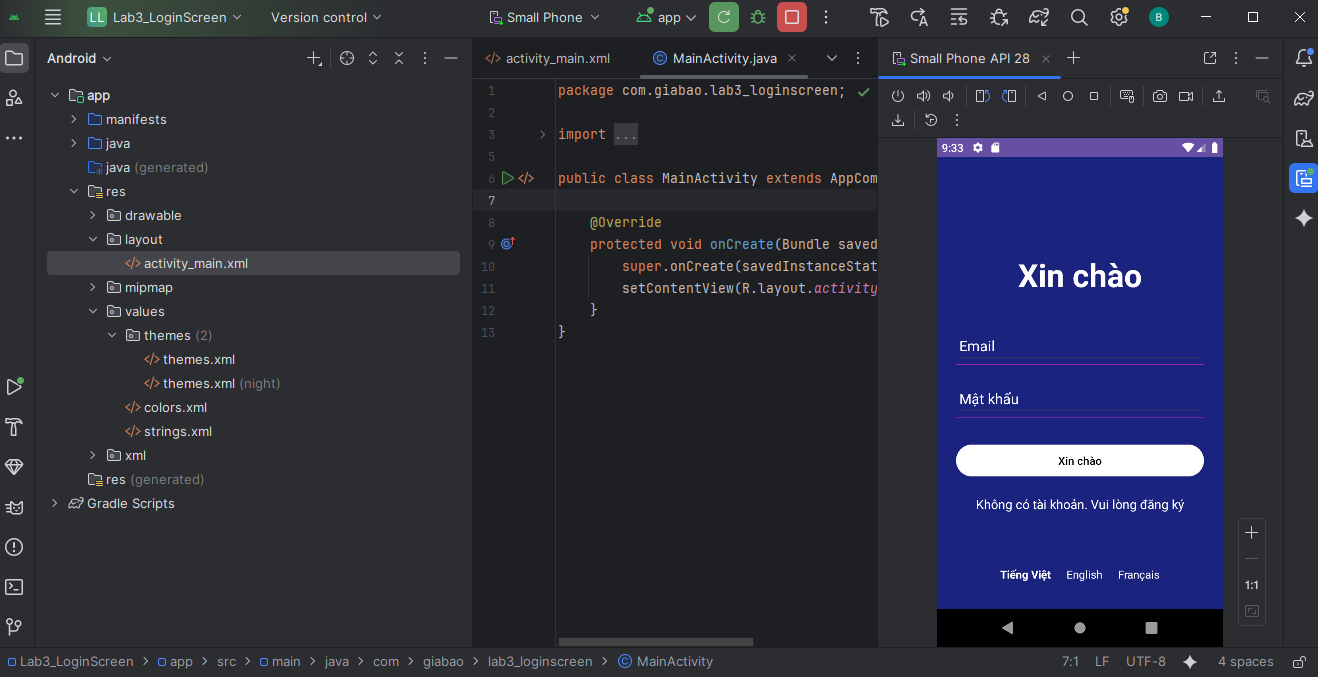
}

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

## Bài tập 2



A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

### Giao diện Đăng nhập Đa ngôn ngữ

Giao diện được xây dựng trên RelativeLayout để linh hoạt vị trí. Khối nhập liệu (Email, Mật khẩu) được căn chính giữa màn hình (centerInParent). Đặc biệt, thanh chọn ngôn ngữ (Việt - Anh - Pháp) được neo cố định ở đáy màn hình (alignParentBottom), giúp người dùng dễ dàng chuyển đổi ngôn ngữ ngay lập tức.

**Code Giao diện (activity\_main.xml):**<?xml version="1.0" encoding="utf-8"?>

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

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="@color/colorPrimaryDark"

android:padding="24dp"

tools:context=".MainActivity">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_centerInParent="true"

android:orientation="vertical">

<TextView

android:id="@+id/tv\_welcome"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

android:text="@string/txt\_welcome"

android:textColor="@color/white"

android:textSize="40sp"

android:textStyle="bold" />

<EditText

android:id="@+id/et\_email"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginTop="40dp"

android:hint="@string/txt\_email"

android:inputType="textEmailAddress"

android:textColor="@color/white"

android:textColorHint="@color/white"

android:theme="@style/Theme.Lab3\_LoginScreen" />

<View

android:layout\_width="match\_parent"

android:layout\_height="1dp"

android:background="@color/purple\_line" />

<EditText

android:id="@+id/et\_password"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginTop="20dp"

android:hint="@string/txt\_password"

android:inputType="textPassword"

android:textColor="@color/white"

android:textColorHint="@color/white"

android:theme="@style/Theme.Lab3\_LoginScreen" />

<View

android:layout\_width="match\_parent"

android:layout\_height="1dp"

android:background="@color/purple\_line" />

<Button

android:id="@+id/btn\_login"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginTop="30dp"

android:backgroundTint="@color/white"

android:text="@string/txt\_welcome"

android:textColor="@color/black" />

<TextView

android:id="@+id/tv\_no\_account"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

android:layout\_marginTop="20dp"

android:text="@string/txt\_no\_account"

android:textColor="@color/white"

android:textSize="16sp" />

</LinearLayout>

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_alignParentBottom="true"

android:gravity="center"

android:orientation="horizontal">

<TextView

android:id="@+id/tv\_lang\_vi"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:padding="10dp"

android:text="@string/txt\_lang\_vi"

android:textColor="@color/white"

android:textStyle="bold" />

<TextView

android:id="@+id/tv\_lang\_en"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:padding="10dp"

android:text="@string/txt\_lang\_en"

android:textColor="@color/white" />

<TextView

android:id="@+id/tv\_lang\_fr"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:padding="10dp"

android:text="@string/txt\_lang\_fr"

android:textColor="@color/white" />

</LinearLayout>

</RelativeLayout>

### Xử lý Logic Thay đổi Ngôn ngữ (setLocale):

Để hiện thực tính năng đa ngôn ngữ, em xây dựng hàm setLocale. Hàm này nhận vào mã ngôn ngữ (vi, en, fr), sau đó cập nhật Locale vào trong Configuration của hệ thống. Cuối cùng, lệnh recreate() được gọi để tải lại Activity, giúp toàn bộ văn bản trên giao diện chuyển sang ngôn ngữ mới ngay lập tức mà không cần khởi động lại ứng dụng.

**Code Logic (MainActivity.java):**

package com.giabao.lab3\_loginscreen;

import android.content.res.Configuration;

import android.content.res.Resources;

import android.os.Bundle;

import android.util.DisplayMetrics;

import android.view.View;

import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import java.util.Locale;

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

TextView tvVi = findViewById(R.id.tv\_lang\_vi);

TextView tvEn = findViewById(R.id.tv\_lang\_en);

TextView tvFr = findViewById(R.id.tv\_lang\_fr);

tvVi.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

setLocale("vi");

}

});

tvEn.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

setLocale("en");

}

});

tvFr.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

setLocale("fr");

}

});

}

// Hàm thay đổi ngôn ngữ

private void setLocale(String langCode) {

Locale locale = new Locale(langCode);

Resources resources = getResources();

DisplayMetrics dm = resources.getDisplayMetrics();

Configuration config = resources.getConfiguration();

config.setLocale(locale);

resources.updateConfiguration(config, dm);

// Tải lại activity để áp dụng ngôn ngữ mới

recreate();

}

}

**File (strings.xml) - Tiếng Việt:**

<resources>

<string name="app\_name">Lab3\_LoginScreen</string>

<string name="txt\_welcome">Xin chào</string>

<string name="txt\_email">Email</string>

<string name="txt\_password">Mật khẩu</string>

<string name="txt\_no\_account">Không có tài khoản. Vui lòng đăng ký</string>

<string name="txt\_lang\_vi">Tiếng Việt</string>

<string name="txt\_lang\_en">English</string>

<string name="txt\_lang\_fr">Français</string>

</resources>

**File (strings.xml) - Tiếng Anh:**

<resources>

<string name="app\_name">Lab3\_LoginScreen</string>

<string name="txt\_welcome">Welcome!</string>

<string name="txt\_email">Email Address</string>

<string name="txt\_password">Password</string>

<string name="txt\_no\_account">Don\'t have account? Sign Up</string>

<string name="txt\_lang\_vi">Tiếng Việt</string>

<string name="txt\_lang\_en">English</string>

<string name="txt\_lang\_fr">Français</string>

</resources>

**File (strings.xml) - Tiếng Pháp:**

<resources>

<string name="app\_name">Lab3\_LoginScreen</string>

<string name="txt\_welcome">accueil</string>

<string name="txt\_email">adresse e-mail</string>

<string name="txt\_password">mot de passe</string>

<string name="txt\_no\_account">Ne pas avoir un compte? signer</string>

<string name="txt\_lang\_vi">Tiếng Việt</string>

<string name="txt\_lang\_en">English</string>

<string name="txt\_lang\_fr">Français</string>

</resources>

# LAB THỰC HÀNH 3



## LAYOUT AND APPLICATION

1. Tạo project:

Em tiến hành tạo một project mới trên Android Studio với tên Lab6\_MoneyConverter.

Ứng dụng sử dụng ngôn ngữ Java và Minimum SDK là API 24 (Android 7.0).

Loại Activity được chọn là Empty Views Activity.

1. Chuẩn bị Tài nguyên Chuỗi (strings.xml):

Em tiến hành mở file strings.xml và khai báo tất cả các tài nguyên chuỗi cần thiết cho ứng dụng theo yêu cầu của tài liệu.

* Các chuỗi này bao gồm tên ứng dụng (app\_name), lời gợi ý (textbox\_hint), và tên đầy đủ (...\_name) cùng ký hiệu (...\_unit) cho 10 loại tiền tệ.

**Source code strings.xml:**

<resources>

<string name="app\_name">Chương trình đổi tiền</string>

<string name="textbox\_hint">Nhập số tiền</string>

<string name="usd\_name">American Dollar</string>  
<string name="eur\_name">European Cash</string>  
<string name="gbp\_name">Greate Britain</string>  
<string name="inr\_name">Indian Rupe</string>  
<string name="aud\_name">Australia Dollar</string>  
<string name="cad\_name">Canadian Dollar</string>  
<string name="zar\_name">South Africa Rand</string>  
<string name="nzd\_name">New Zealand Dollar</string>  
<string name="jpy\_name">Yên Nhật</string>  
<string name="vnd\_name">Việt Nam Đồng</string>

<string name="usd\_unit">USD</string>  
 <string name="eur\_unit">EUR</string>  
 <string name="gbp\_unit">GBP</string>  
 <string name="inr\_unit">INR</string>  
 <string name="aud\_unit">AUD</string>  
 <string name="cad\_unit">CAD</string>  
 <string name="zar\_unit">ZAR</string>  
 <string name="nzd\_unit">NZD</string>  
 <string name="jpy\_unit">JPY</string>  
 <string name="vnd\_unit">VND</string>  
</resources>

1. Chuẩn bị Tài nguyên Theme và Color:

Em tiến hành mở file themes.xml và điều chỉnh parent theme thành Theme.Material3.DayNight để đảm bảo ứng dụng hiển thị Action Bar (thanh tiêu đề) theo đúng mẫu.

Em cũng cập nhật file colors.xml để cung cấp các màu Material Design cơ bản cho theme.

**Source code themes.xml:**

<resources xmlns:tools="http://schemas.android.com/tools">  
 <style name="Base.Theme.Lab6\_MoneyConverter" parent="Theme.Material3.DayNight">  
 </style>  
  
 <style name="Theme.Lab6\_MoneyConverter" parent="Base.Theme.Lab6\_MoneyConverter" />  
</resources>

**Source code colors.xml:**

<?xml version="1.0" encoding="utf-8"?>

<resources>

<color name="purple\_200">#FFBB86FC</color>

<color name="purple\_500">#FF6200EE</color>

<color name="purple\_700">#FF3700B3</color>

<color name="teal\_200">#FF03DAC5</color>

<color name="teal\_700">#FF018786</color>

<color name="black">#FF000000</color>

<color name="white">#FFFFFFFF</color>

</resources>

1. Chuẩn bị Tài nguyên Ảnh (Drawable):

Em tiến hành tải 10 ảnh cờ của 10 đơn vị tiền tệ từ Internet.

Sau đó, em đổi tên chúng thành usd.webp, eur.webp, gbp.gif... và dán tất cả vào thư mục res/drawable của project.

A screenshot of a computer

AI-generated content may be incorrect.

1. Thiết kế Giao diện (activity\_main.xml):

Em tiến hành thiết kế giao diện chính của ứng dụng theo sơ đồ trong tài liệu.

Em sử dụng một LinearLayout (Vertical) làm layout gốc.

Bên trong, em đặt một LinearLayout (Horizontal) chứa một EditText (để nhập số tiền) và một Spinner (để chọn đơn vị tiền).

Bên dưới là một ScrollView chứa TableLayout để hiển thị kết quả.

Mỗi TableRow trong TableLayout chứa 1 ImageView (hiển thị cờ), 1 TextView (hiển thị đơn vị), và 1 TextView (hiển thị kết quả quy đổi).

1. Xử lý Logic (MainActivity.java):

Em tiến hành mở file MainActivity.java và thêm code logic để ứng dụng hoạt động theo hướng dẫn của tài liệu.

Khai báo biến thành viên: Em khai báo 3 biến chính:

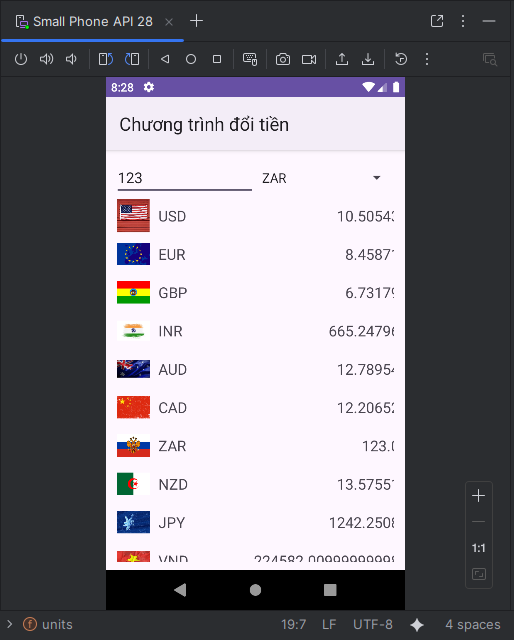
* units: Một mảng String[] 10 phần tử chứa tên các đơn vị tiền (USD, EUR...).
* ratio: Một ma trận double[10][10] chứa tỉ giá quy đổi giữa các đơn vị.
* txtNumber (EditText), spnUnit (Spinner), và lblResults (mảng TextView[] 10 phần tử).

Trong hàm onCreate():

* Em sử dụng findViewById để "ánh xạ" các View từ layout XML vào các biến Java.
* Em khởi tạo một ArrayAdapter để nạp mảng units vào Spinner.
* Em gán 2 hàm lắng nghe sự kiện: setOnItemSelectedListener cho Spinner và addTextChangedListener cho EditText. Cả hai sự kiện này đều sẽ gọi hàm changeMoneyUnit().

Xây dựng hàm changeMoneyUnit():

* Đây là hàm tính toán chính.
* Hàm sẽ lấy vị trí (index) của đơn vị tiền được chọn trong Spinner.
* Lấy số tiền người dùng nhập vào EditText.
* Dùng một vòng lặp for 10 lần, nhân số tiền nhập vào với tỉ giá tương ứng trong ma trận ratio (dựa theo ratio[rowIdx][i]).
* Cuối cùng, hàm cập nhật kết quả vào 10 TextView ở bảng TableLayout.



## Bài tập 1

1. Cấu hình tài nguyên (Strings và Colors)

Em cấu hình các tài nguyên cố định cho ứng dụng:

* Tại colors.xml: Thêm một mã màu (custom\_green) để sử dụng cho thanh tiêu đề "Tương đương với".
* Tại strings.xml: Khai báo tất cả các chuỗi văn bản sẽ sử dụng trong ứng dụng (tên app, tên các đơn vị đo). Điều này giúp việc quản lý và bảo trì code dễ dàng hơn.

1. Thiết kế giao diện - Phần Nhập liệu (EditText và Spinner)

Em mở file activity\_main.xml và tiến hành thiết kế giao diện:

* Sử dụng LinearLayout chính với hướng vertical (chiều dọc) để chứa các thành phần.
* Tạo một LinearLayout con bên trong với hướng horizontal (chiều ngang).
* Trong LinearLayout con, em thêm một EditText (với id là txtNumber) để nhập số và một Spinner (với id là spnUnit) để chọn đơn vị.
* Em sử dụng thuộc tính android:layout\_weight="1" cho cả EditText và Spinner để chúng tự động chia đều không gian theo chiều ngang.

1. Thiết kế giao diện - Tiêu đề và Bảng kết quả

* Em thêm một TextView để làm tiêu đề cho khối kết quả ("Tương đương với"). Em sử dụng thuộc tính android:background để đặt màu nền là màu custom\_green đã định nghĩa và android:textColor là màu trắng cho nổi bật.
* Tiếp theo, em thêm một ScrollView để đảm bảo danh sách kết quả có thể cuộn được nếu nội dung quá dài.
* Bên trong ScrollView, em đặt một TableLayout (với id là tblResults). Đây là thành phần chính để chứa các hàng (các TableRow) hiển thị kết quả chuyển đổi sau này.

1. Thiết kế giao diện - Thêm 4 hàng kết quả đầu tiên

Bên trong TableLayout đã tạo ở bước 4, em tiến hành thêm 4 hàng (TableRow) đầu tiên để hiển thị kết quả cho 4 đơn vị: Hải lý, Dặm, Km, và Lý.

Mỗi TableRow chứa 2 TextView:

* TextView thứ nhất: Dùng để hiển thị giá trị số (ví dụ: 0.0). Em đặt id lần lượt là lblHaily, lblDam, lblKm, lblLy để có thể truy cập và cập nhật giá trị từ code Java. layout\_weight="1" được sử dụng.
* TextView thứ hai: Dùng để hiển thị tên đơn vị, lấy từ strings.xml (ví dụ: @string/unit\_haily). layout\_weight="1" cũng được sử dụng để căn chỉnh.

1. Thiết kế giao diện - Thêm 4 hàng kết quả còn lại

* Các id cho TextView hiển thị giá trị lần lượt là: lblMet, lblYard, lblFoot, và lblInch.
* Các TextView hiển thị tên đơn vị được gán tương ứng: @string/unit\_met, @string/unit\_yard, @string/unit\_foot, @string/unit\_inch.

1. Lập trình Java - Khai báo biến và Ma trận tỉ giá

Em mở file MainActivity.java và tiến hành khai báo các biến thành viên cần thiết cho lớp (class):

* Thêm các thư viện (import) cần thiết cho EditText, Spinner, TextView, ArrayAdapter, TextWatcher...
* Khai báo các biến cho View: txtNumber (EditText), spnUnit (Spinner), và một mảng TextView[] tên là lblResults để lưu 8 ô TextView hiển thị kết quả.
* Khai báo mảng chuỗi units để lưu tên của 8 đơn vị đo.
* Khai báo ma trận 2 chiều (mảng 2D) kiểu double[][] tên là ratio để lưu trữ bảng tỉ giá 8x8 được cung cấp trong đề bài. Mỗi hàng tương ứng với một đơn vị gốc và mỗi cột tương ứng với đơn vị đích.

1. Lập trình Java - Ánh xạ View trong onCreate

Bên trong hàm onCreate, sau dòng setContentView, em tiến hành ánh xạ (kết nối) các biến Java với các thành phần (View) tương ứng trong file activity\_main.xml bằng phương thức findViewById:

* Ánh xạ txtNumber với R.id.txtNumber.
* Ánh xạ spnUnit với R.id.spnUnit.
* Khởi tạo mảng lblResults và lần lượt ánh xạ 8 TextView (từ lblHaily đến lblInch) vào 8 phần tử của mảng. Việc đưa vào mảng giúp dễ dàng truy cập và cập nhật giá trị hàng loạt khi tính toán.

1. Lập trình Java - Cấu hình Spinner

Để Spinner (ô chọn đơn vị) có thể hiển thị danh sách 8 đơn vị, em sử dụng một ArrayAdapter:

* Khởi tạo một ArrayAdapter<String> mới, cung cấp cho nó context (là this), layout chuẩn của Android (android.R.layout.simple\_spinner\_item), và nguồn dữ liệu (mảng units đã khai báo).
* Thiết lập layout khi danh sách xổ xuống (setDropDownViewResource) để hiển thị rõ ràng hơn.
* Cuối cùng, dùng phương thức setAdapter để gán adapter này cho spnUnit.

1. Xây dựng hàm tính toán changeLengthUnit()

Em tạo một hàm riêng tên là changeLengthUnit để xử lý logic tính toán. Hàm này thực hiện các nhiệm vụ sau:

* Lấy vị trí (index) của đơn vị người dùng đã chọn trong Spinner (ví dụ: "Hải lý" là 0, "Dặm" là 1...). Vị trí này chính là chỉ số hàng (rowIdx) trong ma trận tỉ giá ratio.
* Lấy chuỗi (String) từ EditText (txtNumber). Kiểm tra nếu chuỗi rỗng thì gán bằng "0".
* Chuyển đổi chuỗi input thành số thực (double).
* Sử dụng một vòng lặp for chạy 8 lần (từ i = 0 đến 7).
* Bên trong vòng lặp, tính giá trị chuyển đổi bằng công thức: Kết quả = số\_nhập \* ratio[hàng\_đã\_chọn][i].
* Sử dụng String.format("%.4f", ...) để định dạng kết quả luôn hiển thị 4 chữ số thập phân (theo yêu cầu của hình mẫu).
* Cập nhật văn bản cho TextView kết quả tương ứng (lblResults[i]).

1. Lập trình Java - Gắn sự kiện setOnItemSelectedListener cho Spinner

Để ứng dụng tự động tính toán lại khi người dùng thay đổi đơn vị (ví dụ: đổi từ "Dặm" sang "Km"), em gắn một bộ lắng nghe sự kiện setOnItemSelectedListener cho spnUnit.

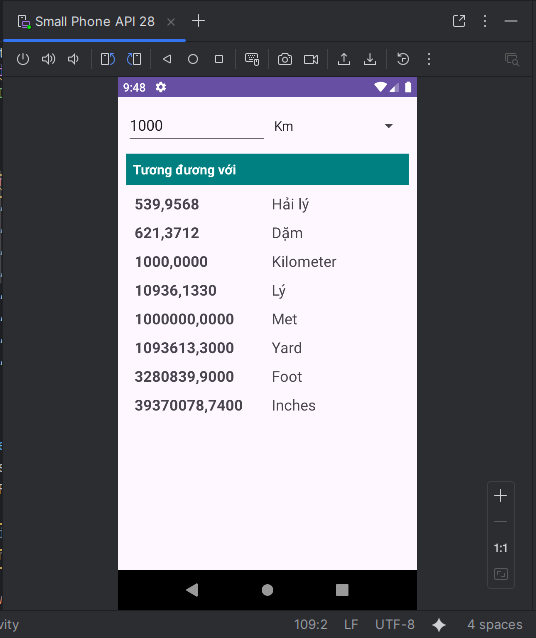
* Bên trong phương thức onItemSelected (được kích hoạt khi người dùng chọn một mục), em gọi đến hàm changeLengthUnit() đã tạo ở bước 10 để cập nhật lại 8 ô kết quả.

1. Lập trình Java - Gắn sự kiện addTextChangedListener cho EditText

Tương tự, để ứng dụng tự động tính toán "real-time" (thời gian thực) ngay khi người dùng gõ số vào EditText, em gắn một bộ lắng nghe sự kiện addTextChangedListener cho txtNumber.

* Bên trong phương thức onTextChanged (được kích hoạt mỗi khi nội dung thay đổi), em cũng gọi đến hàm changeLengthUnit() để cập nhật kết quả.

**Kết thúc:** Ứng dụng đã hoàn thành, đáp ứng yêu cầu tự động chuyển đổi đơn vị đo khi người dùng thay đổi giá trị nhập hoặc thay đổi đơn vị gốc.



## Bài tập 2: Custom controls

1. Tạo Drawable Shape (button\_round\_shape.xml)

Để tạo hình dạng cho nút "ROUND SHAPE", em chuột phải vào thư mục res/drawable và tạo một file tài nguyên mới tên là button\_round\_shape.xml.

* Trong file này, em sử dụng thẻ <shape> để định nghĩa một hình chữ nhật (rectangle).
* sử dụng thẻ <solid> để đặt màu nền là màu trắng.
* sử dụng thẻ <stroke> để tạo đường viền (viền) màu đen với độ dày 2dp.
* sử dụng thẻ <corners> để bo tròn 4 góc với bán kính là 12dp.

1. Tạo Drawable Shape (button\_colored\_normal.xml)

* Để chuẩn bị cho nút "COLORED SELECTOR", em tạo một file button\_colored\_normal.xml trong res/drawable.
* File này định nghĩa hình dạng của nút ở trạng thái bình thường (khi không nhấn).
* Em sử dụng <shape>, <solid> với màu xanh nhạt (#B2EBF2), và <corners> để bo góc 8dp.

1. Tạo Drawable Shape (button\_colored\_pressed.xml)

* Em tạo tiếp file button\_colored\_pressed.xml để định nghĩa hình dạng của nút khi người dùng nhấn giữ.
* Tương tự file normal, em sử dụng <shape>, <solid> nhưng với màu xanh đậm hơn (#00BCD4), và <corners> cũng là 8dp để đảm bảo sự đồng nhất khi chuyển trạng thái.

1. Tạo Drawable Selector (button\_colored\_selector.xml)

Khác với <shape>, file này sử dụng thẻ gốc là <selector>.

Bên trong <selector>, em định nghĩa 2 trạng thái (<item>):

* Trạng thái khi nhấn (android:state\_pressed="true"): Sẽ sử dụng file @drawable/button\_colored\_pressed (màu xanh đậm).
* Trạng thái bình thường (không có thuộc tính state): Sẽ sử dụng file @drawable/button\_colored\_normal (màu xanh nhạt). Đây là trạng thái mặc định.

1. Tạo Drawable Shape (button\_gradient.xml)

* Em tạo file button\_gradient.xml trong res/drawable để định nghĩa hình dạng cho nút "SHAPE WITH GRADIENT".
* Em sử dụng <shape> với màu nền (<solid>) là màu vàng (#FFEB3B).
* Em thêm một đường viền (<stroke>) màu đen, dày 2dp.
* Em bo góc (<corners>) 8dp.

1. Tạo Drawable Shape (button\_reaction.xml)

* Em tạo file button\_reaction.xml cho nút "REACTION".
* sử dụng <shape> với màu nền (<solid>) là màu đỏ (#F44336).
* thêm một đường viền (<stroke>) màu đen, dày 2dp.
* bo góc (<corners>) 8dp.

1. Tạo Drawable Shape (image\_bg\_cyan.xml)

* Để tạo nền cho ImageView (icon mặt cười) đầu tiên, em tạo file image\_bg\_cyan.xml.
* File này định nghĩa một hình chữ nhật (<shape>) với màu nền (<solid>) là màu xanh cyan nhạt (#B2EBF2) và bo góc (<corners>) 8dp.

1. Tạo Drawable Shape (image\_bg\_cyan\_border.xml)

* Em tạo file drawable cuối cùng là image\_bg\_cyan\_border.xml để làm nền cho ImageView thứ hai.
* File này tương tự image\_bg\_cyan.xml (nền cyan, bo góc 8dp) nhưng có thêm thẻ <stroke> để tạo một đường viền màu đen, dày 2dp, giống hệt trong hình minh họa.

1. Thiết kế Layout (activity\_main.xml) - Khung cơ bản

* Em sử dụng RelativeLayout làm layout gốc. Layout này cho phép em định vị các thành phần con một cách linh hoạt (ví dụ: căn giữa, căn lề trên/dưới, hoặc căn theo một thành phần khác).
* thêm TextView tiêu đề ("Custom Button"), đặt id là tvTitle, căn giữa theo chiều ngang (layout\_centerHorizontal="true") và đặt màu chữ đỏ.
* thêm TextView chân trang ("Made by..."), đặt id là tvFooter, căn giữa chiều ngang và neo ở cuối màn hình (layout\_alignParentBottom="true").

1. Thêm nhãn "Selectors" vào Layout

* Em thêm một TextView mới vào RelativeLayout với id là tvLabelSelectors.
* Em sử dụng android:layout\_below="@id/tvTitle" để đặt nó bên dưới tiêu đề "Custom Button".
* Em sử dụng android:layout\_alignParentStart="true" để căn nó về lề trái của layout (với một chút layout\_marginStart để nó không dính sát viền).

1. Thêm nhãn "Shapes+Selectors" vào Layout

* Em thêm một TextView khác với id là tvLabelShapes.
* cũng đặt nó android:layout\_below="@id/tvTitle" để nó thẳng hàng với nhãn "Selectors".
* sử dụng android:layout\_alignParentEnd="true" để căn nó về lề phải của layout (với một chút layout\_marginEnd).
* Cả hai nhãn đều được đặt android:textStyle="bold".

1. Cấu hình colors.xml

* Để tuân thủ nguyên tắc "code sạch" và loại bỏ cảnh báo của Android Studio, em mở file res/values/colors.xml.
* Em định nghĩa một tài nguyên màu mới tên là text\_title\_red với giá trị #D32F2F (màu đỏ cho tiêu đề).

1. Thêm nhãn "Selectors" và "Shapes+Selectors"

* Em thêm hai TextView mới vào RelativeLayout.
* TextView thứ nhất (id: tvLabelSelectors): Đặt bên dưới tvTitle (layout\_below) và căn lề trái (layout\_alignParentStart="true").
* TextView thứ hai (id: tvLabelShapes): Đặt bên dưới tvTitle (layout\_below) và căn lề phải (layout\_alignParentEnd="true").
* Cả hai đều được in đậm và có màu chữ đen.

1. Thêm nút "COLORED SELECTOR"

* Em thêm một Button mới với id là btnColoredSelector.
* đặt nó bên dưới nhãn tvLabelSelectors (layout\_below).
* Thuộc tính quan trọng nhất: android:background="@drawable/button\_colored\_selector". Thao tác này áp dụng file selector (tự động đổi màu khi nhấn) làm nền cho nút.
* đặt android:textColor là màu đen để chữ nổi bật trên nền xanh.

1. Hoàn thiện activity\_main.xml

* Thêm View (Ô vàng): Em thêm một thẻ <View> (với id là yellowBox) ở cột trái, đặt nền là @color/bg\_yellow\_box để tạo ô màu vàng giống hình.
* Thêm Button (Reaction): Em thêm Button btnReaction ở cột phải, bên dưới btnShapeGradient, và gán nền là @drawable/button\_reaction.
* Thêm ImageView (Mặt cười 1): Em thêm ImageView ở cột trái, đặt nền là @drawable/image\_bg\_cyan và dùng icon ic\_menu\_emoticons của Android làm ảnh.
* Thêm ImageView (Mặt cười 2): Em thêm ImageView ở cột phải, đặt nền là @drawable/image\_bg\_cyan\_border (nền cyan có viền đen) và cũng dùng icon ic\_menu\_emoticons.

A screenshot of a chat

AI-generated content may be incorrect.

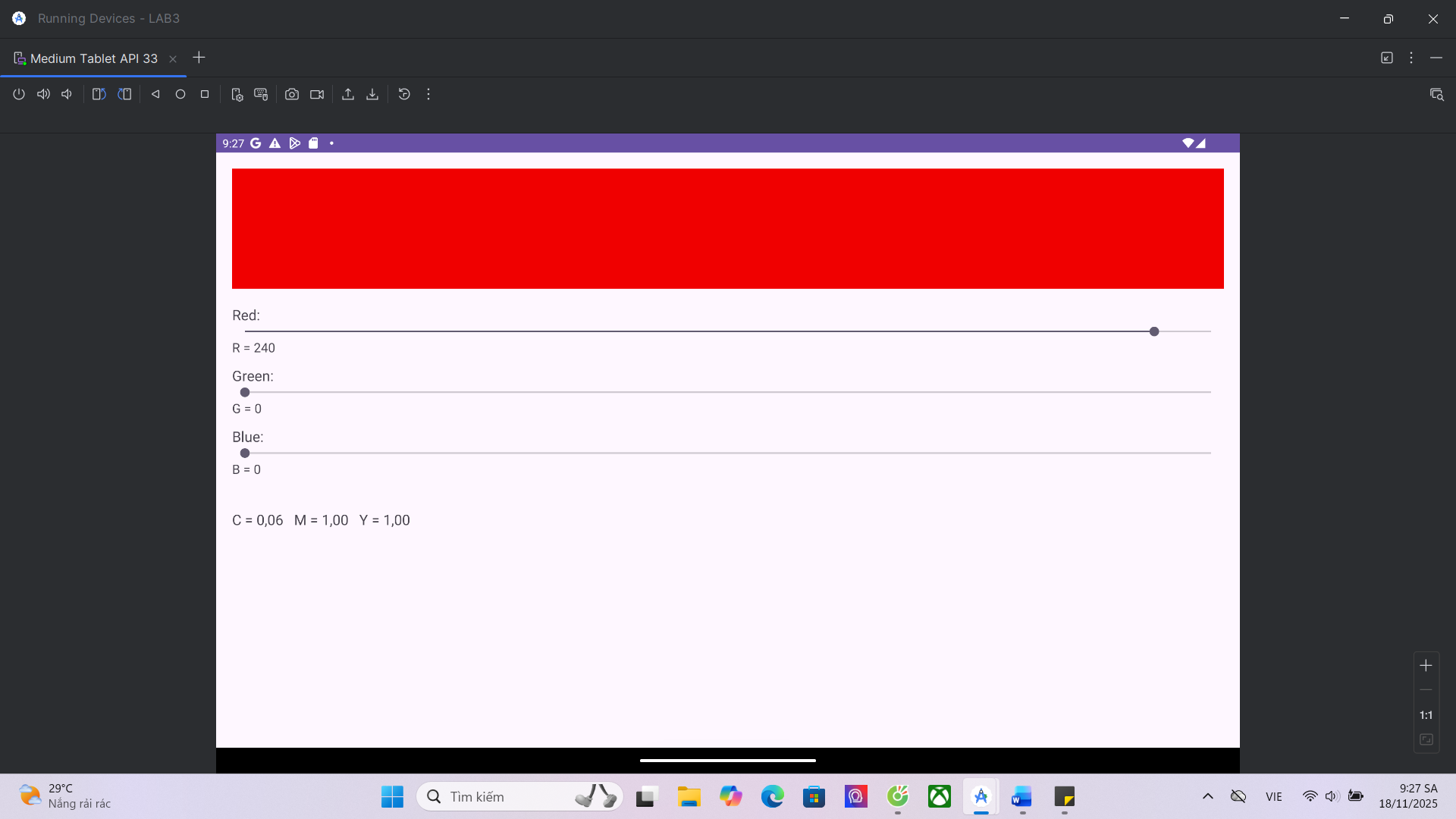
A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## Bài tập 3



A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

# LAB THỰC HÀNH 4



## Khởi tạo và chuẩn bị tài nguyên

1. Cấu hình dự án (Project Setup):

* Tên ứng dụng: **Drama Story**.

1. Cấu trúc dữ liệu (Assets):

Đã tạo thư mục assets trong src/main để lưu trữ dữ liệu tĩnh (không thay đổi).

Trong assets chia làm 2 thư mục con:

assets/photo: Chứa hình ảnh đại diện cho các chủ đề truyện (định dạng .png/.jpg).

assets/story: Chứa các file text (.txt) lưu nội dung truyện.

**Quy tắc đặt tên:** Tên file ảnh và tên file text phải trùng khớp nhau (ví dụ: drama\_giadinh.png và drama\_giadinh.txt) để thuật toán ánh xạ dữ liệu hoạt động chính xác.

1. Cấu trúc file dữ liệu (.txt):

Dữ liệu truyện được định dạng đặc biệt để ứng dụng có thể đọc và phân tách.

Mỗi câu chuyện kết thúc bằng chuỗi ký tự đặc biệt ','0'); để đánh dấu điểm ngắt, giúp ứng dụng nhận biết và tách riêng từng truyện trong cùng một chủ đề.

**Xây dựng Giao diện (Layout) và Model dữ liệu**

1. Xây dựng Model (StoryEntity): - Tạo class StoryEntity để mô hình hóa dữ liệu của một câu chuyện. - Các thuộc tính bao gồm: topicName (Tên chủ đề), name (Tiêu đề truyện), và content (Nội dung chi tiết) . Đây là cơ sở để ánh xạ dữ liệu đọc được từ file assets vào bộ nhớ ứng dụng.

2. Thiết kế hệ thống Layout (XML): Ứng dụng sử dụng cấu trúc phân tầng Layout để tối ưu hóa khả năng tái sử dụng: - activity\_main.xml : Layout chứa chính (Container), sử dụng LinearLayout làm khung sườn để nhúng (replace) các Fragment con vào trong quá trình chạy. - m000\_frg\_splash.xml : Màn hình chào, sử dụng FrameLayout để căn giữa Logo thương hiệu, tạo ấn tượng ban đầu cho người dùng. - item\_topic.xml: Thiết kế giao diện cho từng dòng chủ đề truyện. Sử dụng ImageView hiển thị thumbnail và TextView hiển thị tên chủ đề. - m001\_frg\_topic.xml: Màn hình danh sách chủ đề. Sử dụng thẻ <include> để tái sử dụng thanh tiêu đề (actionbar\_home). Bên dưới là ScrollView chứa LinearLayout để có thể cuộn danh sách khi số lượng chủ đề tăng lên.

**Xử lý Logic**

1. Màn hình M000SplashFrg:

Sử dụng Handler và phương thức postDelayed để tạo bộ đếm thời gian. Sau 2000ms (2 giây), ứng dụng sẽ tự động kích hoạt phương thức gotoM001Screen thông qua MainActivity để chuyển hướng người dùng vào màn hình chính.

2. Màn hình M001TopicFrg (Hiển thị danh sách chủ đề): - Đọc dữ liệu từ Assets: Sử dụng mContext.getAssets().list("photo") để lấy danh sách tên file hình ảnh trong thư mục tài nguyên.

Hiển thị động (Dynamic UI): Với mỗi file ảnh tìm thấy, ứng dụng thực hiện: + Xử lý chuỗi để loại bỏ phần mở rộng (ví dụ .png) lấy tên hiển thị.

* Sử dụng LayoutInflater để nạp layout mẫu item\_topic.xml. + Sử dụng BitmapFactory.decodeStream để đọc dữ liệu ảnh bitmap trực tiếp từ luồng InputStream và hiển thị lên ImageView .
* Add view vừa tạo vào LinearLayout chính để tạo thành danh sách dọc.

3. MainActivity (Điều hướng):

Thiết kế hàm showFrg(Fragment frg) đóng vai trò là trình quản lý chuyển đổi Fragment trung tâm. Sử dụng getSupportFragmentManager().beginTransaction().replace(...) để thay thế nội dung màn hình mà không cần khởi tạo lại Activity, giúp ứng dụng chạy mượt mà hơn.

4. Xử lý Màn hình Danh sách Truyện (M002StoryFrg)

a. Cơ chế đọc dữ liệu từ file Text:

Sử dụng InputStreamReader và BufferedReader để mở luồng đọc file .txt từ thư mục assets/story/.

Tên file cần đọc được xác định động dựa trên topicName được truyền từ màn hình trước (ví dụ: người dùng chọn chủ đề "Gia đình" -> ứng dụng đọc file drama\_giadinh.txt).

Thuật toán tách truyện (Parsing Logic):

* Dòng đầu tiên đọc được coi là Tiêu đề truyện.
* Các dòng tiếp theo được nối vào chuỗi Nội dung. + Vòng lặp đọc nội dung sẽ dừng lại khi gặp chuỗi ký tự đánh dấu kết thúc ','0'); .

b. Hiển thị danh sách:

Dữ liệu sau khi tách được đóng gói vào ArrayList<StoryEntity>.

M002StoryFrg khởi tạo StoryAdapter và truyền danh sách này vào RecyclerView để hiển thị lên màn hình.

Thanh tiêu đề (TextView) được cập nhật theo tên chủ đề hiện tại.

5. Xử lý Màn hình Chi tiết Truyện (M003DetailStoryFrg)

a. Hiển thị nội dung (ViewPager):

- Sử dụng ViewPager kết hợp với DetailStoryAdapter (kế thừa từ PagerAdapter) để tạo trải nghiệm đọc truyện liền mạch. Người dùng có thể vuốt sang trái hoặc phải để chuyển sang truyện kế tiếp mà không cần quay lại danh sách.

b. Truyền dữ liệu giữa các màn hình:

- Tại MainActivity, phương thức gotoM003Screen nhận vào danh sách truyện (listStory) và truyện đang được chọn (story).

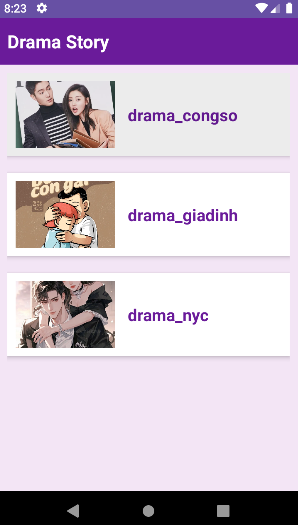
- Dữ liệu này được chuyển ("inject") vào Fragment M003DetailStoryFrg thông qua phương thức setData.

- Fragment sử dụng vp.setCurrentItem(index) để mở đúng trang truyện người dùng vừa click vào, đảm bảo trải nghiệm người dùng chính xác.

## KẾT QUẢ THỰC HIỆN

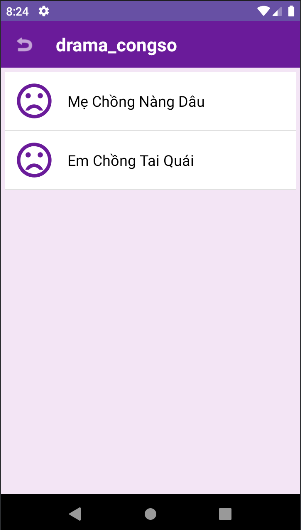
**1**. Giao diện Danh sách Chủ đề (Home Screen)

- Ứng dụng khởi động thành công, load dữ liệu hình ảnh và tên chủ đề từ thư mục assets/photo lên RecyclerView.



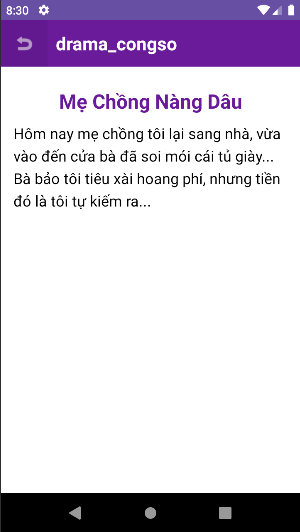
**2. Giao diện Danh sách Truyện (Story List)**

- Khi chọn chủ đề "Drama Công Sở", ứng dụng đọc file drama\_congso.txt, tách chuỗi và hiển thị danh sách tiêu đề các bài drama.



**3. Giao diện Đọc truyện chi tiết (Detail View)**

- Hiển thị toàn bộ nội dung câu chuyện. Đã tích hợp ViewPager cho phép người dùng vuốt sang trái/phải để chuyển sang truyện kế tiếp liền mạch.

A screenshot of a phone

AI-generated content may be incorrect.

**KẾT LUẬN & KIẾN THỨC ĐẠT ĐƯỢC**

Qua bài thực hành Lab 10-11, em đã nắm vững và vận dụng thành công các kiến thức:

1. Kỹ thuật đọc dữ liệu (Assets): Biết cách truy xuất và xử lý file hình ảnh, file văn bản (.txt) từ thư mục tài nguyên Assets.
2. Xử lý chuỗi (String Parsing): Hiểu thuật toán tách chuỗi nội dung dựa trên ký tự đánh dấu đặc biệt để phân loại dữ liệu.
3. Sử dụng Components nâng cao:

* RecyclerView & Adapter: Để hiển thị danh sách dữ liệu động tối ưu hiệu năng.
* ViewPager: Tạo trải nghiệm chuyển trang mượt mà (Swipe) cho người đọc.
* Intent & Bundle: Truyền tải dữ liệu phức tạp (Object) giữa các màn hình Activity/Fragment.

# LAB THỰC HÀNH 5

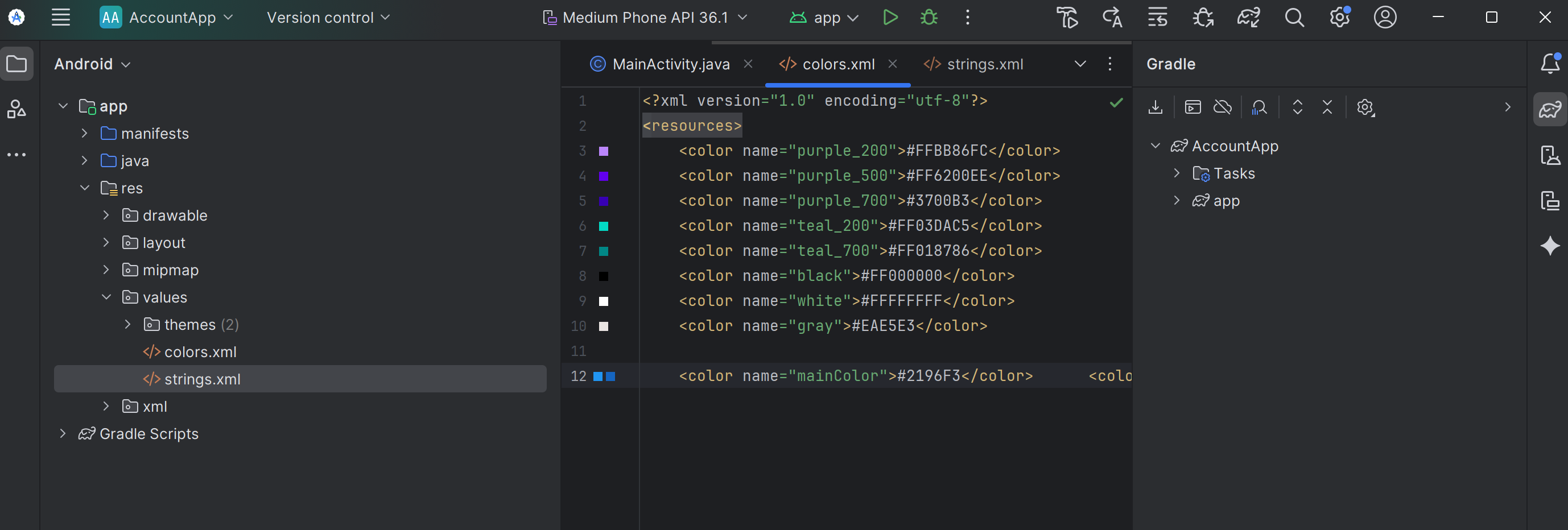


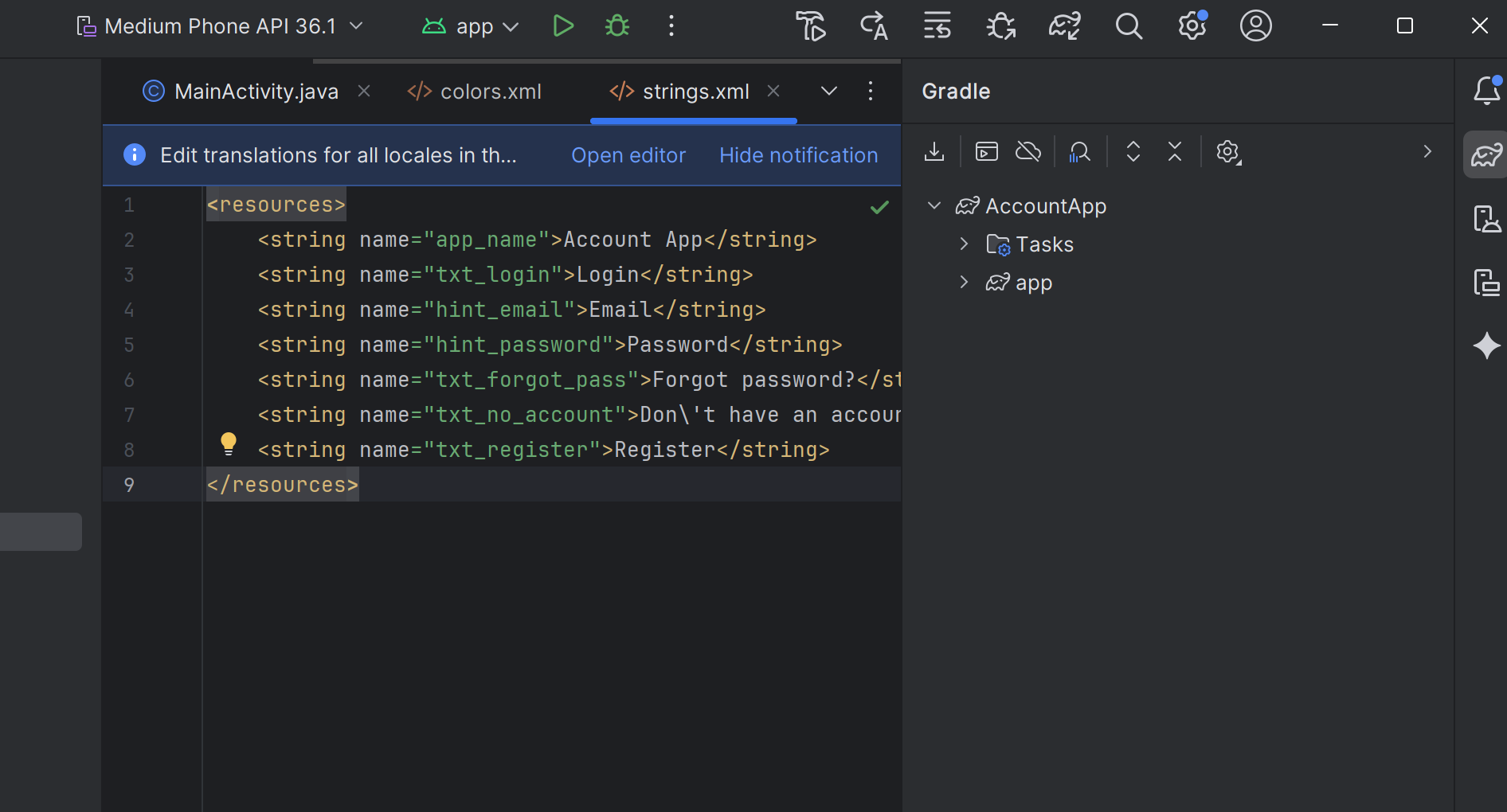
## LAB 13-14: ỨNG DỤNG ACCOUNT

1. Thiết kế giao diện và Cấu hình
2. Khởi tạo dự án và thiết lập tài nguyên

Đầu tiên, em khởi tạo dự án Android mới với tên "AccountApp", sử dụng ngôn ngữ Java. Để đảm bảo tính cá nhân hóa và khác biệt so với bài mẫu, em đã tùy biến lại bảng màu và các tài nguyên cơ bản trong thư mục res/values:

* Màu sắc (**colors.xml**): Em thay đổi tông màu chủ đạo từ Cam sang Xanh Dương (Blue) để giao diện trông hiện đại hơn. Các mã màu chính gồm mainColor (xanh sáng) và mainColorDark (xanh đậm) dùng cho các hiệu ứng chuyển màu.
* Chuỗi ký tự (**strings.xml**): Em khai báo trước các đoạn văn bản sẽ sử dụng như "Login", "Register", "Email"... giúp quản lý nội dung tập trung và dễ dàng chỉnh sửa.

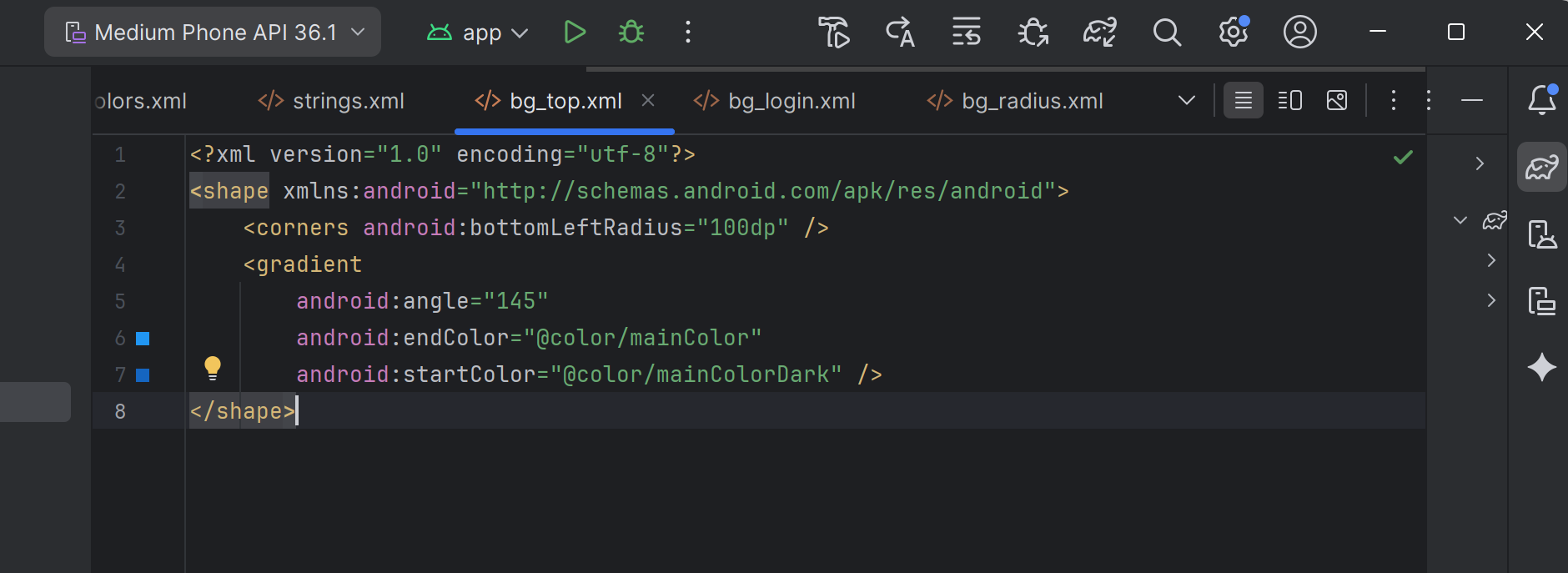


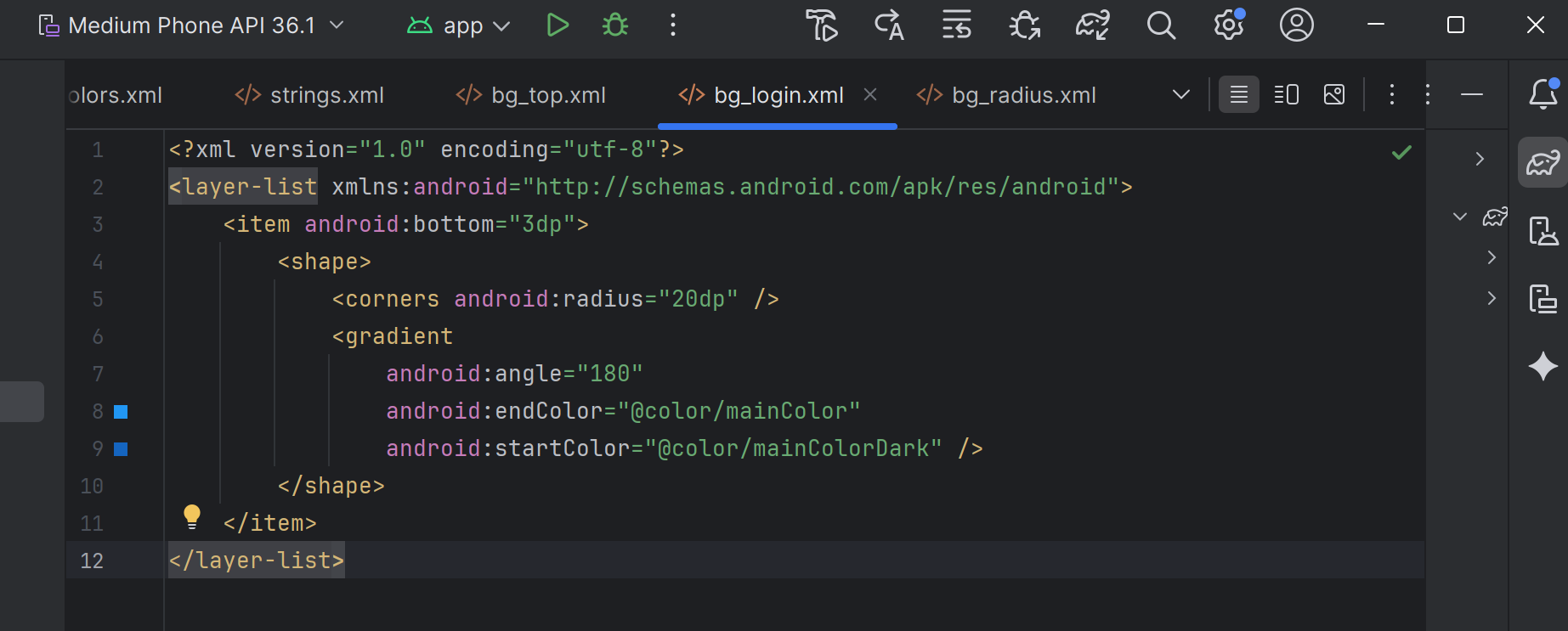


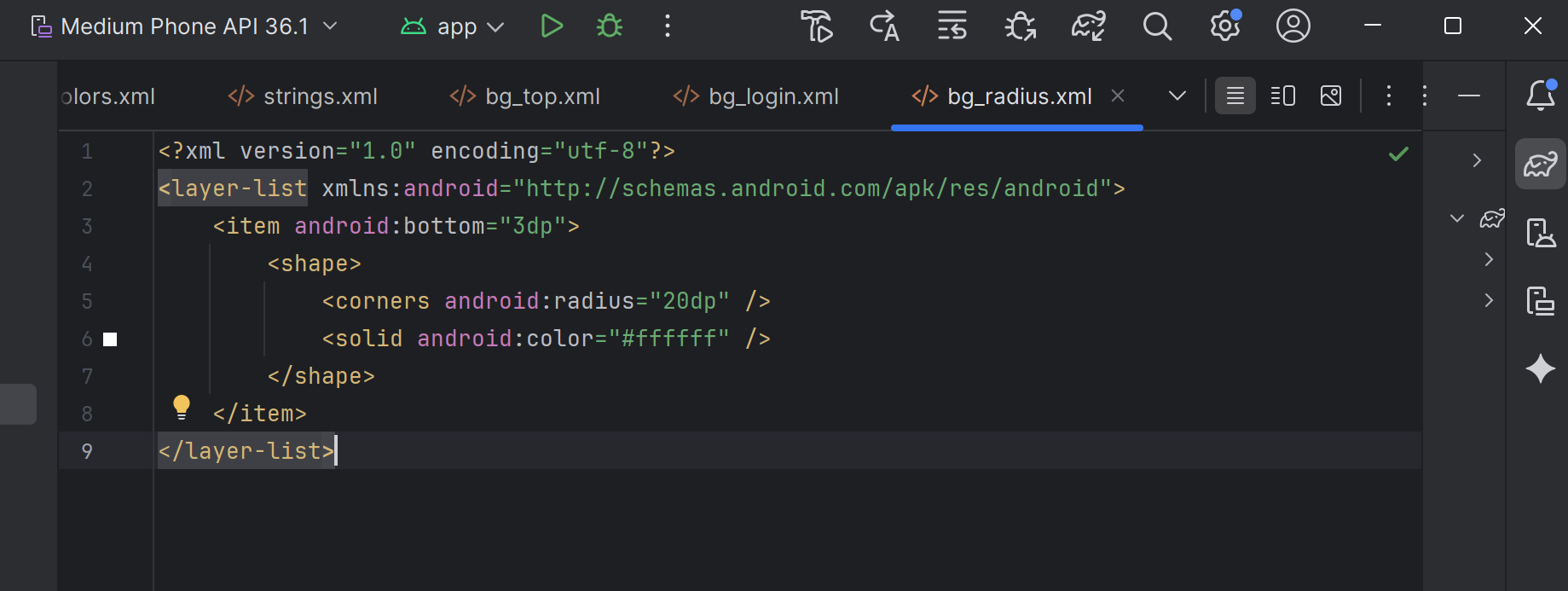
1. Thiết kế tài nguyên hình ảnh (Drawable Resources)

Em tạo các file XML trong thư mục drawable để tùy biến giao diện theo tông màu xanh chủ đạo:

* **bg\_top.xml**: Tạo khối nền tiêu đề với góc bo tròn lệch (bottomLeftRadius 100dp) và hiệu ứng chuyển màu (gradient) xanh dương.
* **bg\_login.xml**: Tạo nền cho các nút bấm (Button) với góc bo tròn 20dp và đổ bóng nhẹ.
* **bg\_radius.xml**: Tạo khung nền màu trắng, bo góc dùng cho các ô nhập liệu (EditText).







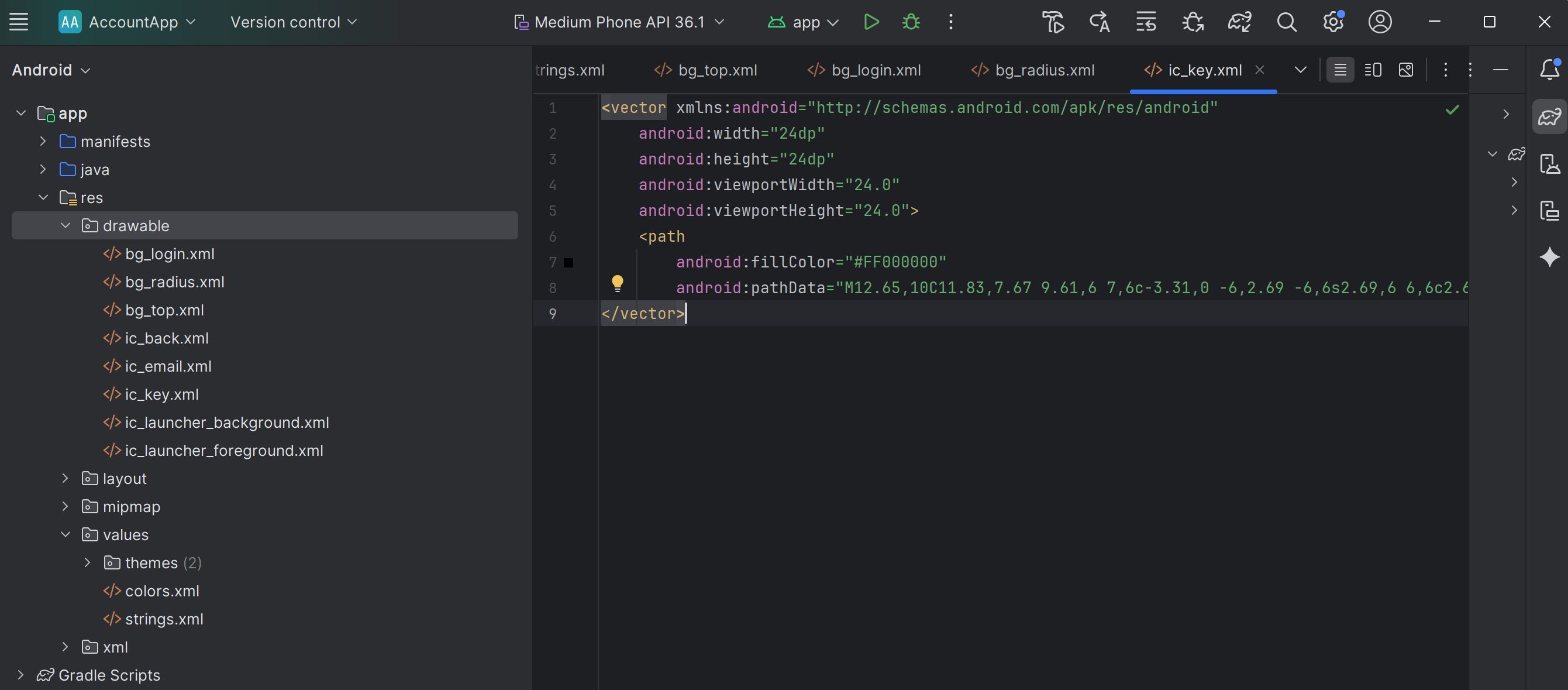
1. Chuẩn bị Icon (Vector Assets)

Thay vì sử dụng hình ảnh PNG thông thường, em sử dụng **Vector Assets** có sẵn trong Android Studio để làm biểu tượng cho ứng dụng. Việc này giúp ứng dụng nhẹ hơn và icon không bị vỡ nét khi hiển thị trên các màn hình có độ phân giải khác nhau.

Các icon đã khởi tạo bao gồm:

* ic\_email.xml: Biểu tượng bì thư dùng cho ô nhập Email.
* ic\_key.xml: Biểu tượng chìa khóa dùng cho ô nhập Mật khẩu.

ic\_back.xml: Biểu tượng mũi tên dùng cho nút quay lại màn hình trước.

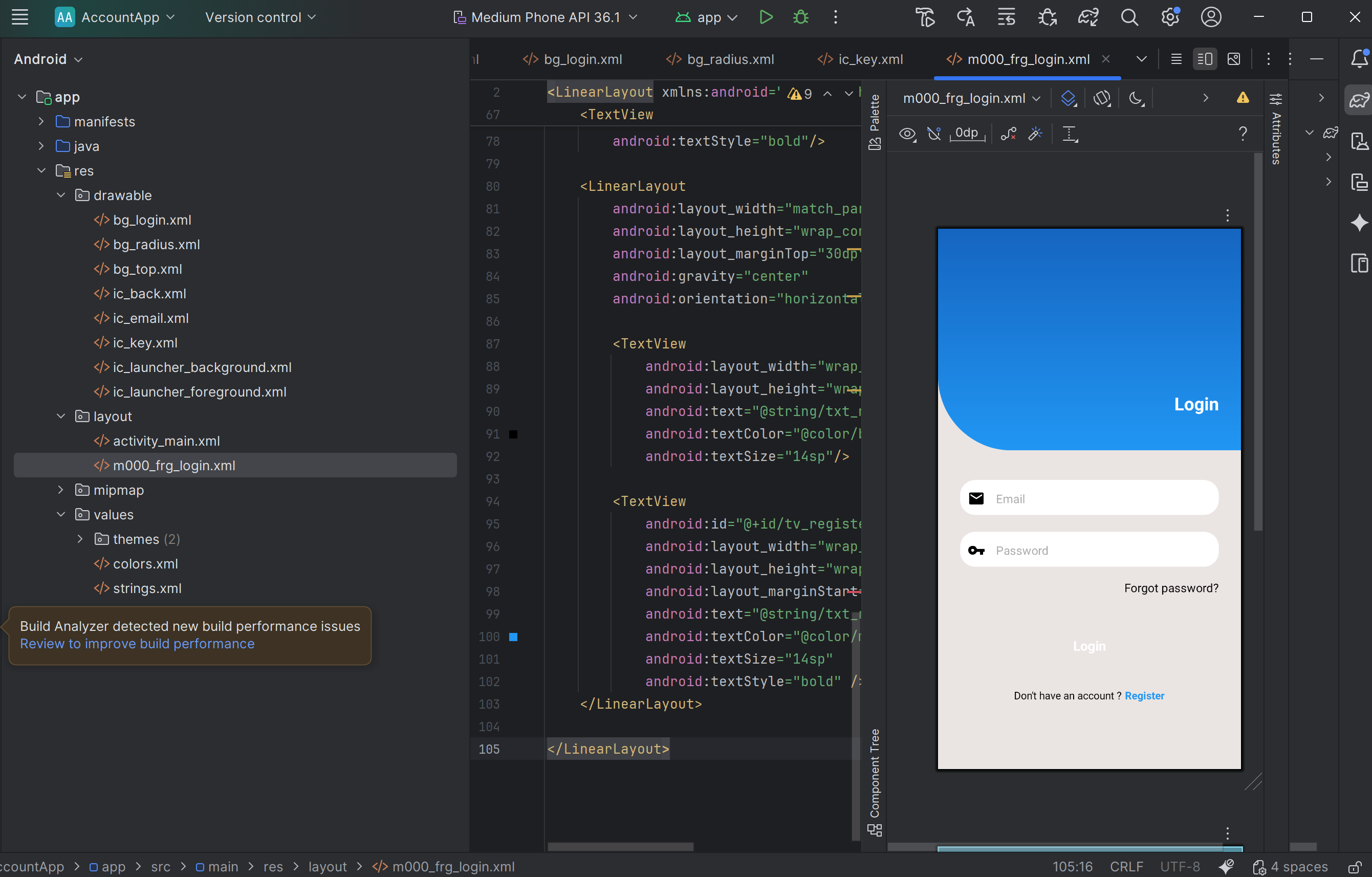


## Xây dựng Layout (Giao diện)

1. Màn hình Đăng nhập (Login Fragment)

Em thiết kế file giao diện m000\_frg\_login.xml dựa trên LinearLayout theo chiều dọc (vertical) để sắp xếp các thành phần từ trên xuống dưới một cách khoa học:

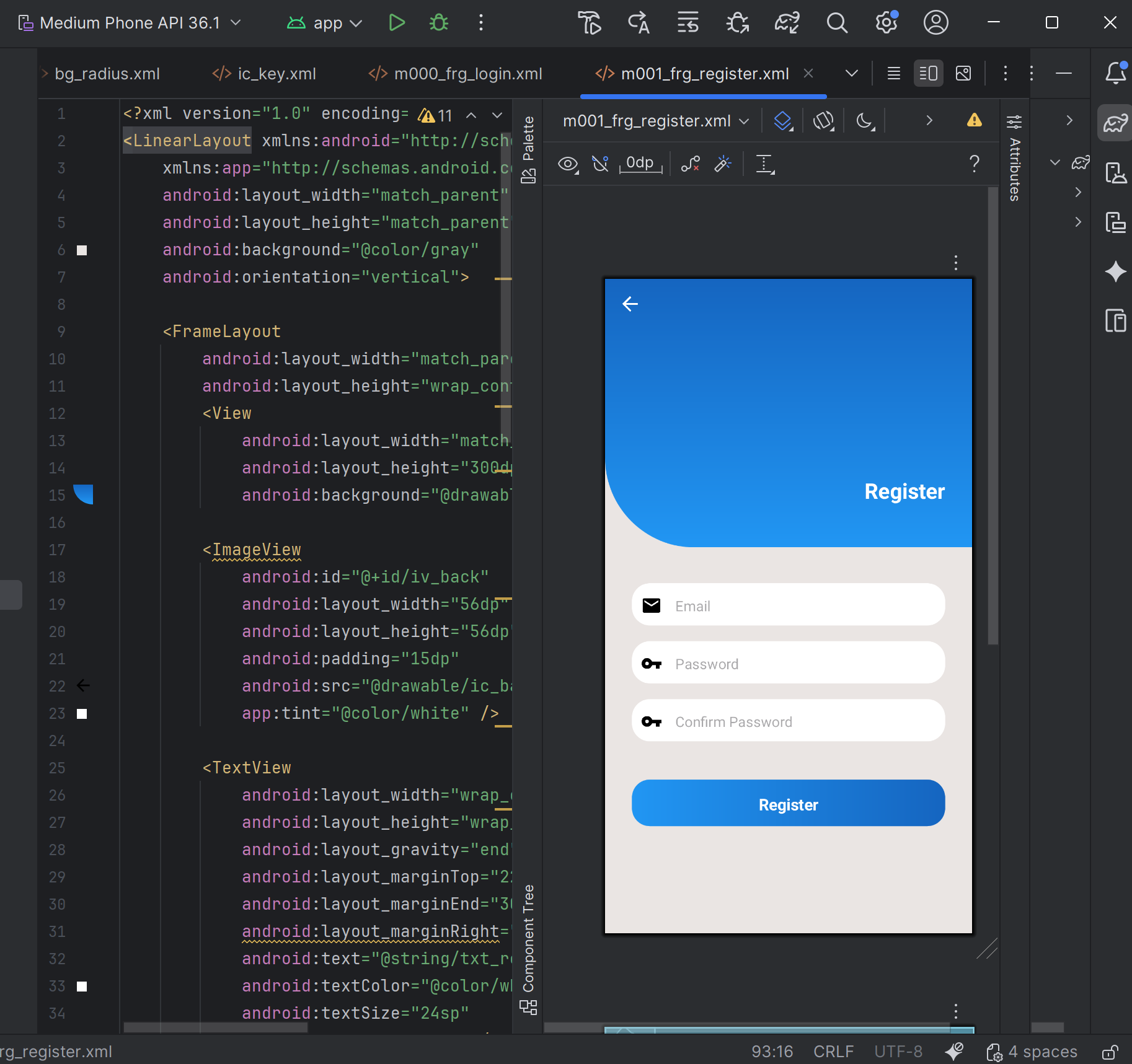
* **Header:** Sử dụng FrameLayout lồng ghép View (chứa background bg\_top.xml) và TextView tiêu đề Login để tạo hiệu ứng lớp chồng lên nhau.
* **Input Fields:** Các ô nhập liệu EditText được gán background bg\_radius (nền trắng bo góc) và tích hợp icon vector (ic\_email, ic\_key) ở đầu dòng thông qua thuộc tính drawableStart.
* **Action Buttons:** Nút Login được tùy biến từ TextView với background bg\_button (gradient xanh) để tạo điểm nhấn thị giác đồng nhất.



1. Màn hình Đăng ký (Register Fragment)

Tương tự như màn hình Đăng nhập, giao diện Đăng ký (m001\_frg\_register.xml) được tái sử dụng các tài nguyên drawable đã thiết kế để đảm bảo tính nhất quán.

* **Cải tiến Header:** Bổ sung nút ImageView (ic\_back) ở góc trên bên trái, cho phép người dùng quay lại màn hình đăng nhập.
* **Bổ sung trường nhập liệu:** Thêm một ô EditText thứ ba (edt\_re\_pass) để người dùng xác nhận lại mật khẩu, đảm bảo tính chính xác khi tạo tài khoản.
* **Nút thao tác:** Đổi nút hành động chính thành "Register" (Đăng ký).



## Xử lý Logic (Controller)

1. Cấu hình MainActivity (Container)

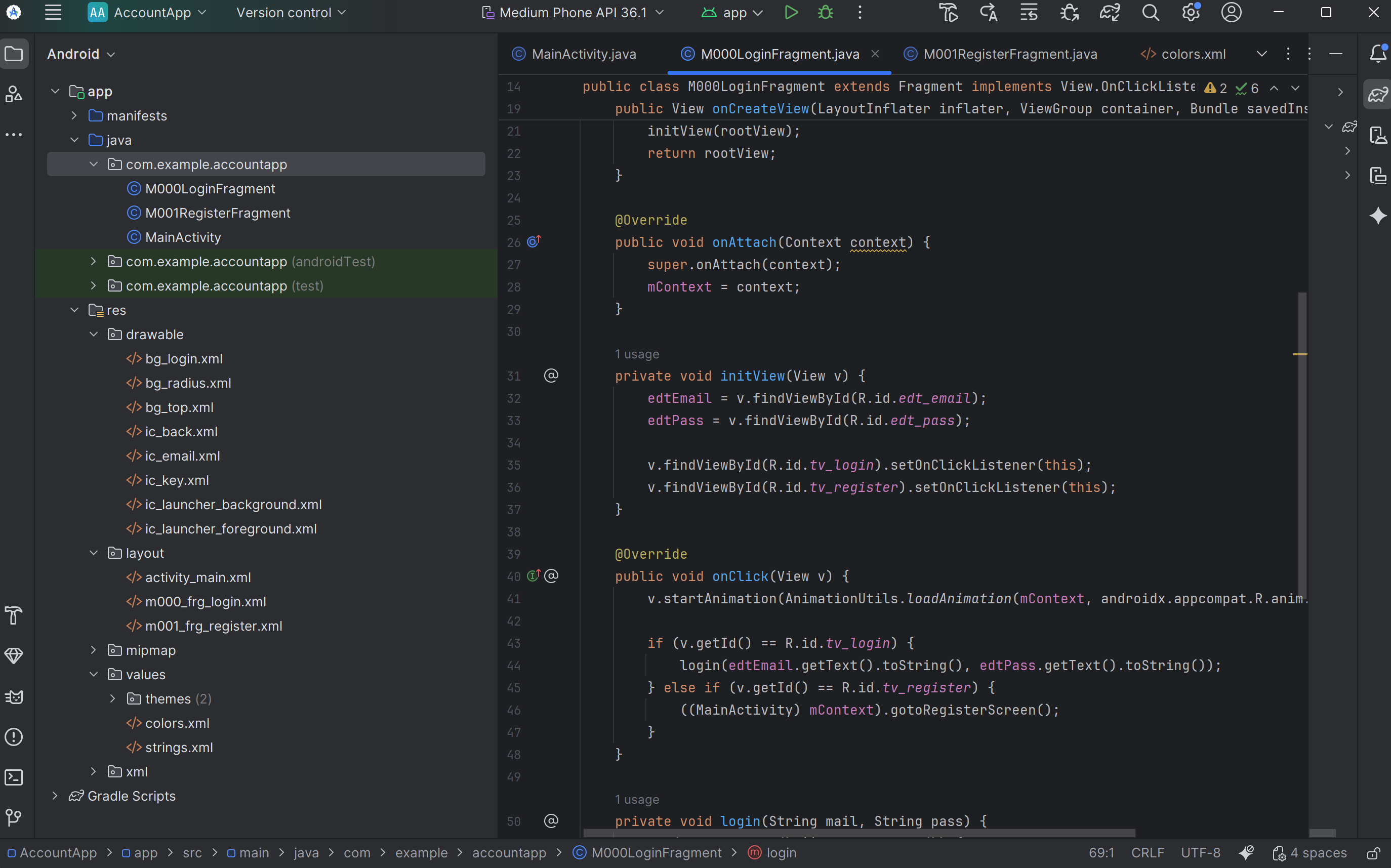
* MainActivity đóng vai trò là container chứa các Fragment. Tại đây, em khai báo hằng số SAVE\_PREF để định danh cho file lưu trữ SharedPreferences.
* Các phương thức gotoLoginScreen() và gotoRegisterScreen() được xây dựng để thực hiện điều hướng (Navigation) giữa các màn hình bằng cách thay thế (replace) Fragment vào ln\_main.

1. Xử lý Đăng nhập (M000LoginFragment)

* Fragment này thực hiện ánh xạ các view (EditText, TextView) từ layout m000\_frg\_login.

Logic đăng nhập:

* Kiểm tra tính hợp lệ của dữ liệu đầu vào (không được để trống).
* Sử dụng SharedPreferences để truy xuất mật khẩu đã lưu dựa trên Key là Email người dùng nhập.
* So sánh mật khẩu nhập vào với mật khẩu trong hệ thống. Nếu trùng khớp sẽ thông báo đăng nhập thành công (Toast), ngược lại sẽ báo lỗi.

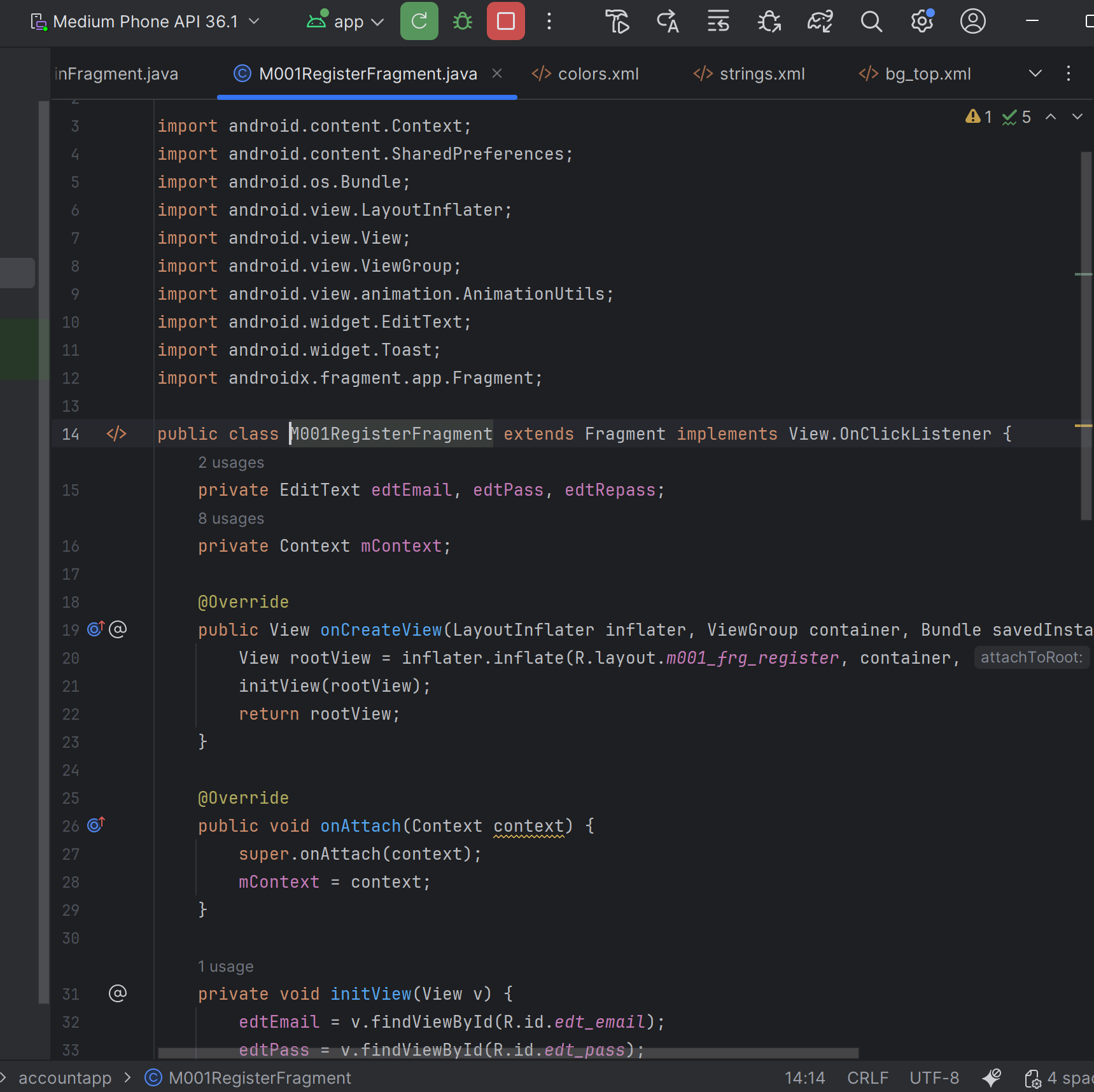


1. Xử lý Đăng ký (M001RegisterFragment)

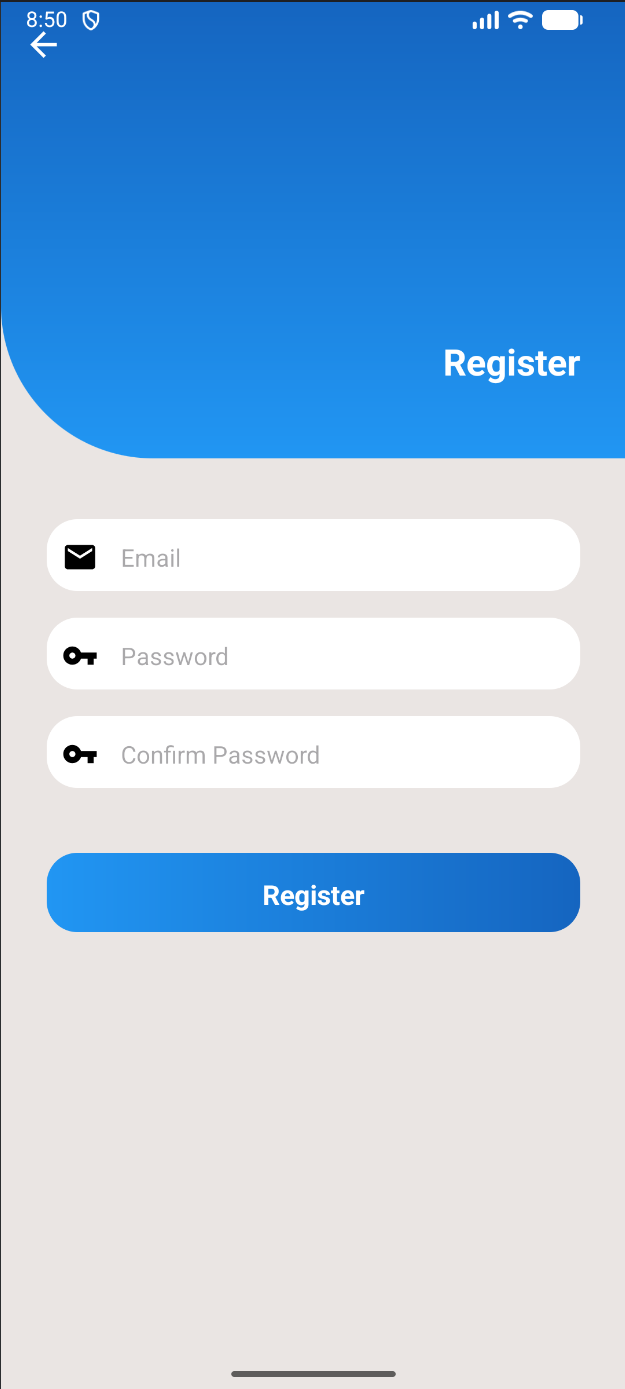
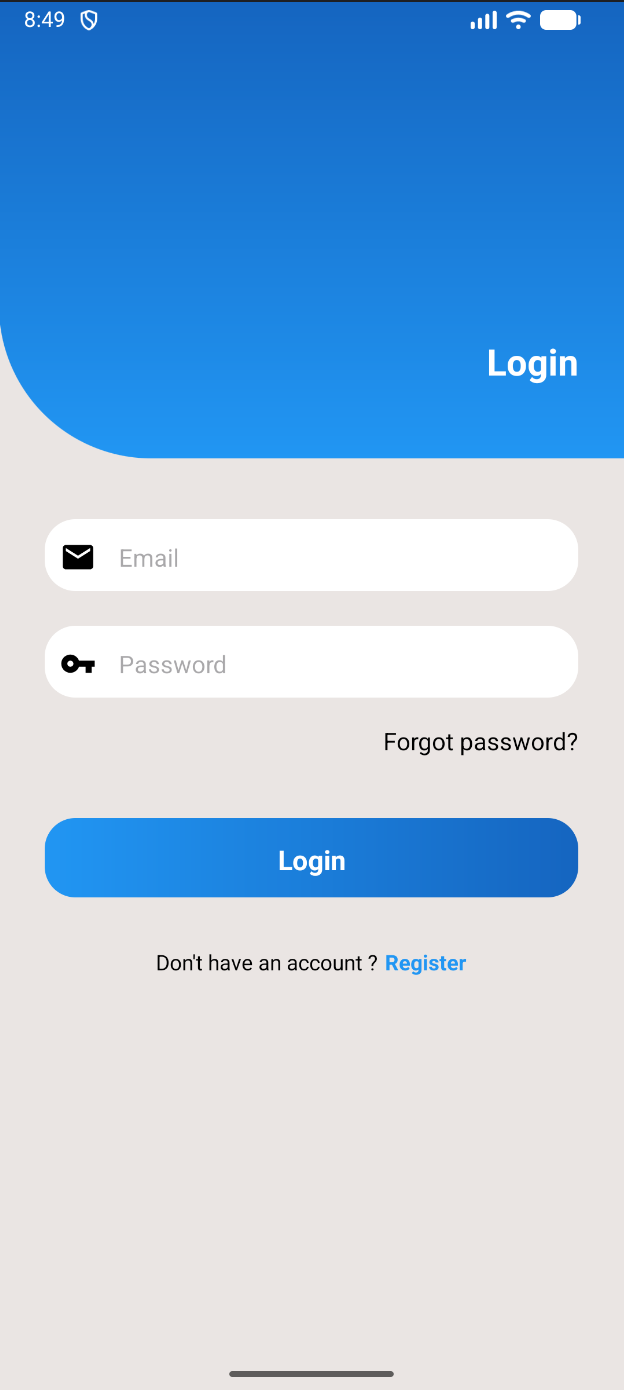
* Tương tự như màn hình đăng nhập, Fragment đăng ký thực hiện ánh xạ các thành phần giao diện gồm 3 ô nhập liệu (Email, Password, Re-password).

Logic đăng ký:

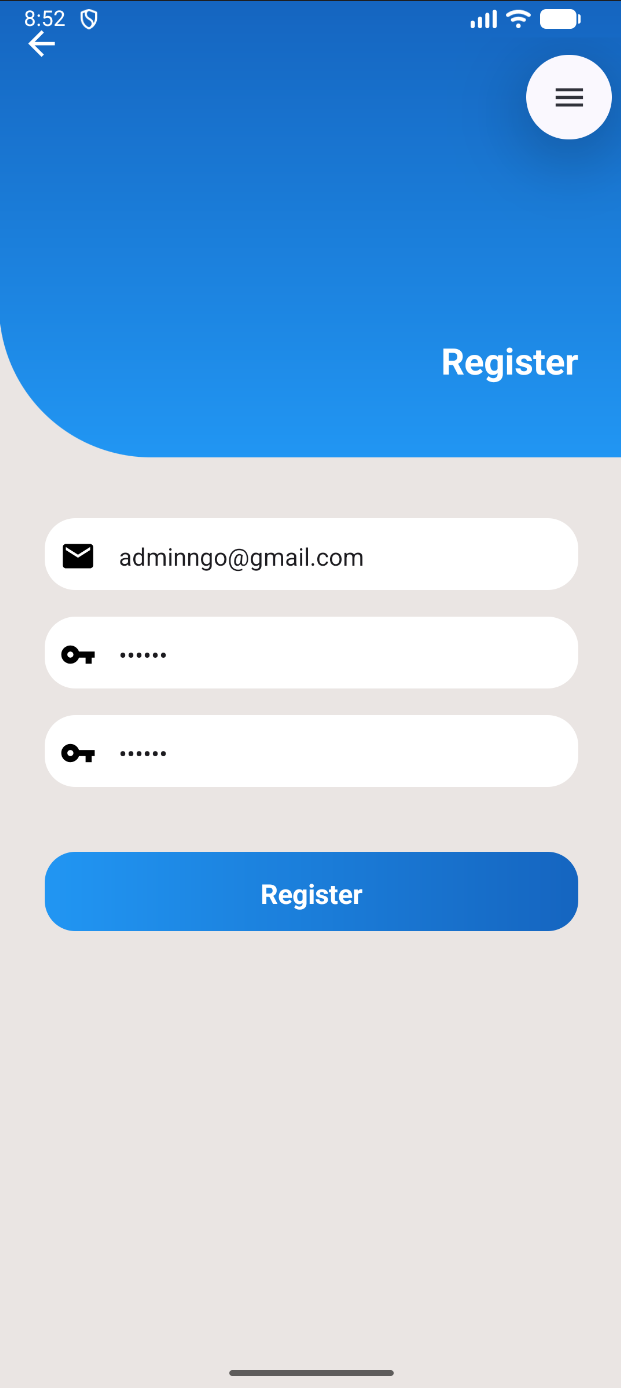
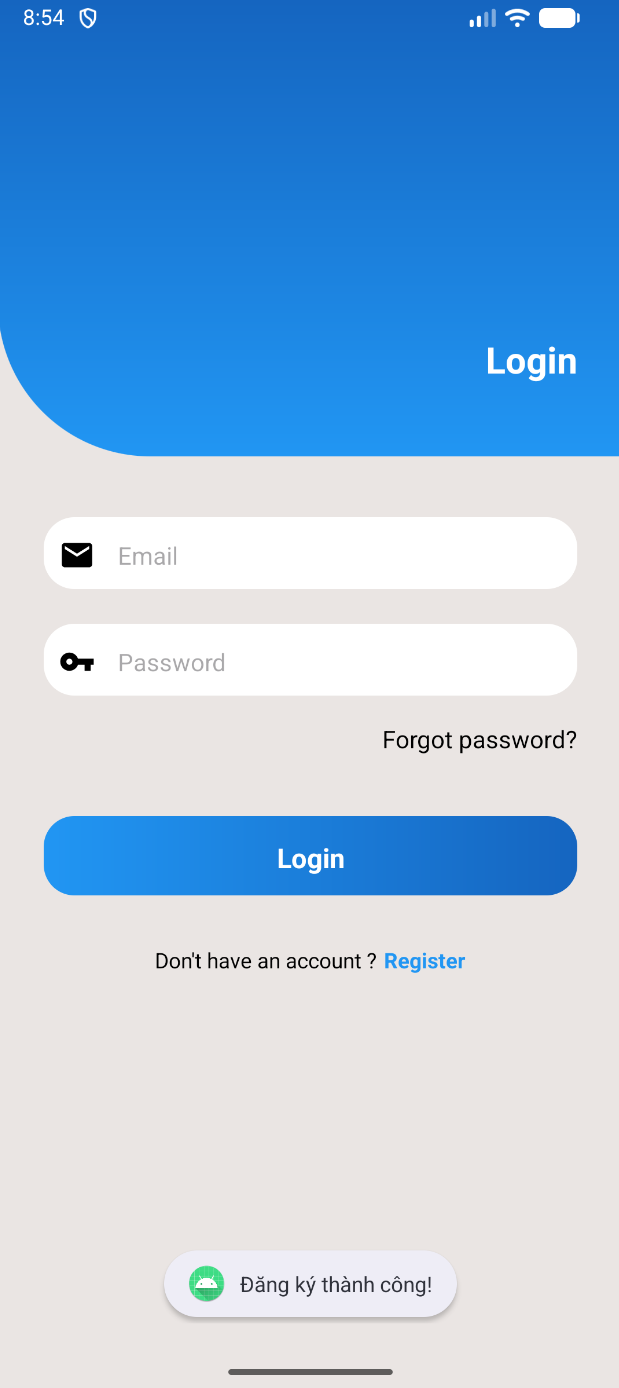
* Kiểm tra các ô nhập liệu có bị trống không.
* Kiểm tra mật khẩu (pass) và xác nhận mật khẩu (repass) có khớp nhau không.
* Sử dụng phương thức pref.getString(mail, null) để kiểm tra xem Email này đã tồn tại trong hệ thống chưa.
* Nếu thỏa mãn tất cả điều kiện, sử dụng pref.edit().putString(mail, pass).apply() để lưu thông tin tài khoản mới vào bộ nhớ máy và chuyển hướng người dùng về màn hình đăng nhập.



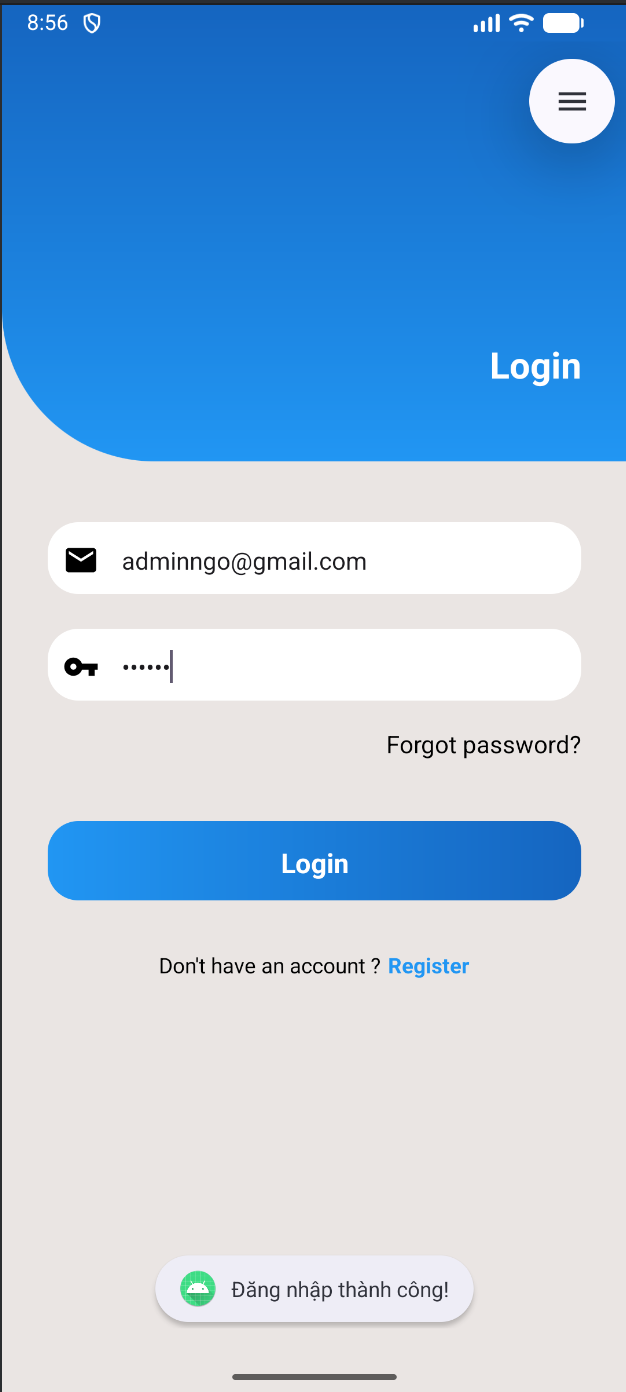
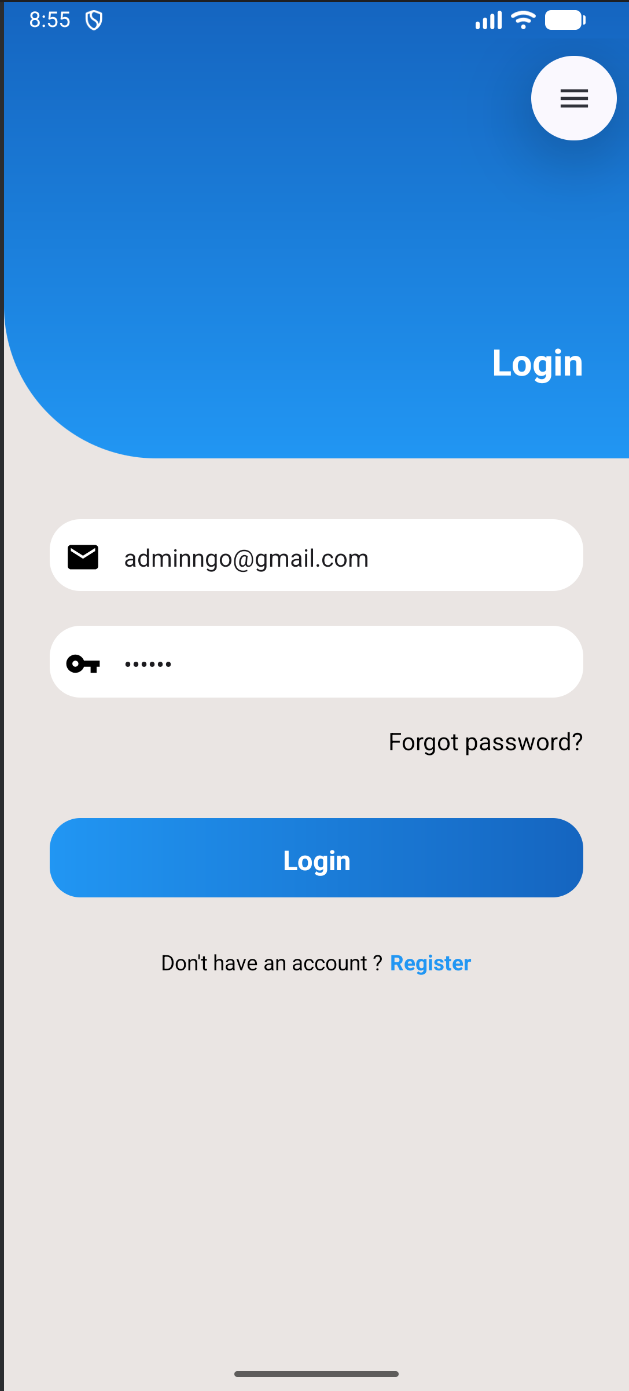
## Kết quả



Tạo thử 1 tài khoản:

Bây giờ sẽ test thử đăng nhập:



## Tìm hiểu về SharedPreferences

1. Khái niệm

SharedPreferences là một API của Android dùng để lưu trữ dữ liệu dưới dạng các cặp khóa - giá trị (Key-Value) đơn giản (như số nguyên, chuỗi, boolean...). Dữ liệu này được lưu trong một file XML bảo mật bên trong thư mục cài đặt của ứng dụng.

1. Ứng dụng trong bài Lab

Trong dự án này, SharedPreferences đóng vai trò thay thế cho SQL Database để lưu trữ thông tin đăng nhập vì dữ liệu đơn giản và cần tốc độ truy xuất nhanh.

* **Key (Khóa):** Là Email của người dùng (duy nhất).
* **Value (Giá trị):** Là Mật khẩu tương ứng.

1. Các phương thức quan trọng đã dùng:

* getSharedPreferences(NAME, MODE): Khởi tạo file lưu trữ.
* edit(): Mở chế độ chỉnh sửa để ghi dữ liệu.
* putString(key, value): Đẩy dữ liệu vào bộ nhớ đệm.

apply(): Lưu dữ liệu xuống file bất đồng bộ (giúp không bị treo giao diện).