

**TRƯỜNG ĐẠI HỌC THỦY LỢI**  
**KHOA CÔNG NGHỆ THÔNG TIN**



# **GIÁO TRÌNH**

## **THỰC HÀNH PHÁT TRIỂN ỨNG DỤNG CHO THIẾT BỊ DI ĐỘNG**

Hà Nội, 2.2025

# MỤC LỤC

CHƯƠNG 1. Làm quen .....	4
Bài 1) Tạo ứng dụng đầu tiên .....	4
1.1) Android Studio và Hello World.....	4
1.2) Giao diện người dùng tương tác đầu tiên .....	21
1.3) Trình chỉnh sửa bố cục .....	33
1.4) Văn bản và các chế độ cuộn .....	33
1.5) Tài nguyên có sẵn .....	33
Bài 2) Activities.....	33
2.1) Activity và Intent.....	33
2.2) Vòng đời của Activity và trạng thái .....	33
2.3) Intent ngầm định .....	33
Bài 3) Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ .....	33
3.1) Trình gỡ lỗi.....	33
3.2) Kiểm thử đơn vị .....	33
3.3) Thư viện hỗ trợ .....	33
CHƯƠNG 2. Trải nghiệm người dùng.....	34
Bài 1) Tương tác người dùng .....	34
1.1) Hình ảnh có thể chọn.....	34
1.2) Các điều khiển nhập liệu .....	34
1.3) Menu và bộ chọn .....	34
1.4) Điều hướng người dùng .....	34
1.5) RecyclerView .....	34
Bài 2) Trải nghiệm người dùng thú vị.....	34
2.1) Hình vẽ, định kiểu và chủ đề .....	34
2.2) Thẻ và màu sắc .....	34
2.3) Bố cục thích ứng.....	34
Bài 3) Kiểm thử giao diện người dùng .....	34

3.1) Espresso cho việc kiểm tra UI .....	34
CHƯƠNG 3. Làm việc trong nền .....	34
Bài 1) Các tác vụ nền.....	34
1.1) AsyncTask.....	34
1.2) AsyncTask và AsyncTaskLoader .....	34
1.3) Broadcast receivers .....	34
Bài 2) Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền.....	34
2.1) Thông báo.....	34
2.2) Trình quản lý cảnh báo .....	34
2.3) JobScheduler .....	34
CHƯƠNG 4. Lưu dữ liệu người dùng .....	35
Bài 1) Tùy chọn và cài đặt.....	35
1.1) Shared preferences .....	35
1.2) Cài đặt ứng dụng.....	35
Bài 2) Lưu trữ dữ liệu với Room .....	35
2.1) Room, LiveData và ViewModel.....	35
2.2) Room, LiveData và ViewModel.....	35

# CHƯƠNG 1. LÀM QUEN

## Bài 1) Tạo ứng dụng đầu tiên

### 1.1) Android Studio và Hello World

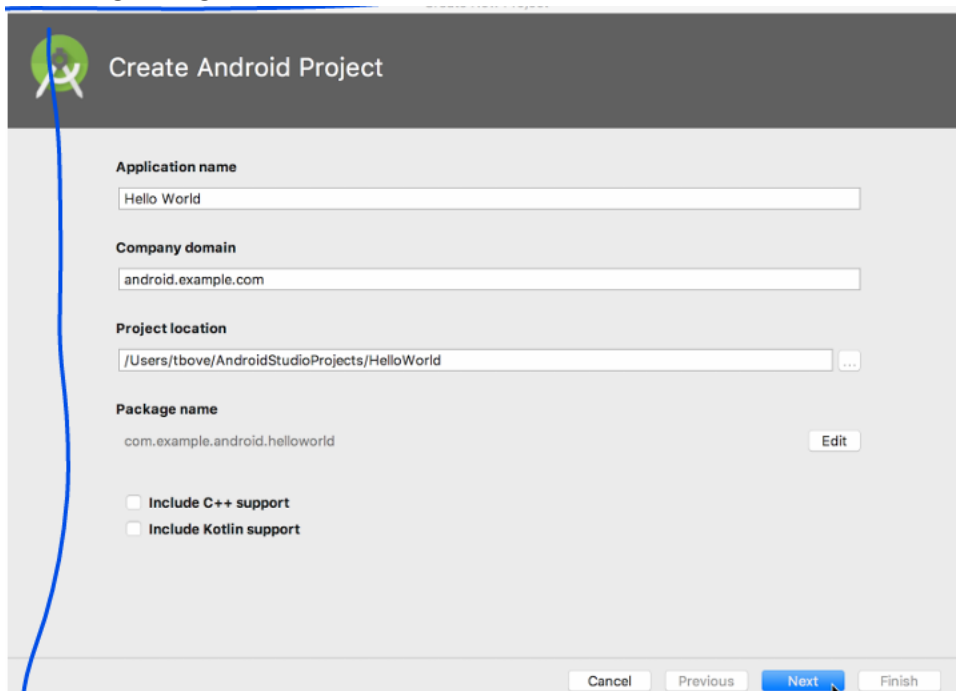
#### Giới thiệu

Trong bài thực hành này, bạn sẽ tìm hiểu cách cài đặt Android Studio, môi trường phát triển Android. Bạn cũng sẽ tạo và chạy ứng dụng Android đầu tiên của mình, Hello World, trên một trình giả lập và trên một thiết bị vật lý.

#### Những gì Bạn nên biết

Bạn nên có khả năng:

- Hiểu quy trình phát triển phần mềm tổng quát cho các ứng dụng lập trình hướng đối tượng sử dụng một IDE (môi trường phát triển tích hợp) như Android Studio.
- Chứng minh rằng bạn có ít nhất 1-3 năm kinh nghiệm trong lập trình hướng đối tượng, với một phần trong số đó tập trung vào ngôn ngữ lập trình Java. (Các bài thực hành này sẽ không giải thích về lập trình hướng đối tượng hoặc ngôn ngữ Java.)



## Những gì Bạn sẽ cần:

- Một máy tính chạy Windows hoặc Linux, hoặc một Mac chạy macOS. Xem trang tải xuống Android Studio để biết yêu cầu hệ thống cập nhật.
- Truy cập Internet hoặc một phương pháp thay thế để tải các cài đặt mới nhất của Android Studio và Java lên máy tính của bạn.

## Những gì bạn sẽ học

- Cách cài đặt và sử dụng IDE Android Studio.
- Cách sử dụng quy trình phát triển để xây dựng ứng dụng Android.
- Cách tạo một dự án Android từ một mẫu.
- Cách thêm thông điệp ghi lại vào ứng dụng của bạn để phục vụ mục đích gỡ lỗi.

## Những gì bạn sẽ làm

- Cài đặt môi trường phát triển **Android Studio**.
- Tạo một trình giả lập (thiết bị ảo) để chạy ứng dụng của bạn trên máy tính.
- Tạo và chạy ứng dụng **Hello World** trên các thiết bị ảo và vật lý.
- Khám phá cấu trúc dự án.
- Tạo và xem các thông điệp ghi lại từ ứng dụng của bạn.
- Khám phá tệp **AndroidManifest.xml**

## Tổng quan về ứng dụng

Sau khi cài đặt thành công Android Studio, bạn sẽ tạo một dự án mới cho ứng dụng Hello World từ một mẫu. Ứng dụng đơn giản này hiển thị chuỗi “Hello World” trên màn hình của thiết bị ảo hoặc vật lý.

Ứng dụng hoàn thành sẽ trông như thế này:



## Nhiệm vụ 1: Cài đặt Android Studio

Android Studio cung cấp một môi trường phát triển tích hợp (IDE) hoàn chỉnh bao gồm trình chỉnh sửa mã nâng cao và một bộ mẫu ứng dụng. Ngoài ra, nó còn chứa các công cụ để phát triển, gỡ lỗi thử nghiệm và hiệu suất giúp phát triển ứng dụng nhanh hơn và dễ dàng hơn. Bạn có thể kiểm tra ứng dụng của mình bằng nhiều trình mô phỏng được cấu hình sẵn hoặc trên thiết bị di động của riêng mình, tạo ứng dụng chính thức trên cửa hàng Google Play.

**Lưu ý:** Android Studio liên tục được cải tiến. Để biết thông tin mới nhất về yêu cầu hệ thống và hướng dẫn cài đặt, hãy xem **Android Studio**

Android Studio có sẵn cho máy tính chạy Windows hoặc Linux hoặc máy Mac chạy macOS. OpenJDK (Java Development Kit) mới nhất được đi kèm với Android Studio.

Để thiết lập và chạy Android Studio, trước tiên hãy kiểm tra **system requirements** để đảm bảo hệ thống của bạn đáp ứng các yêu cầu đó. Việc cài đặt tương tự cho tất cả các nền tảng. Bất kỳ sự khác biệt được lưu ý bên dưới

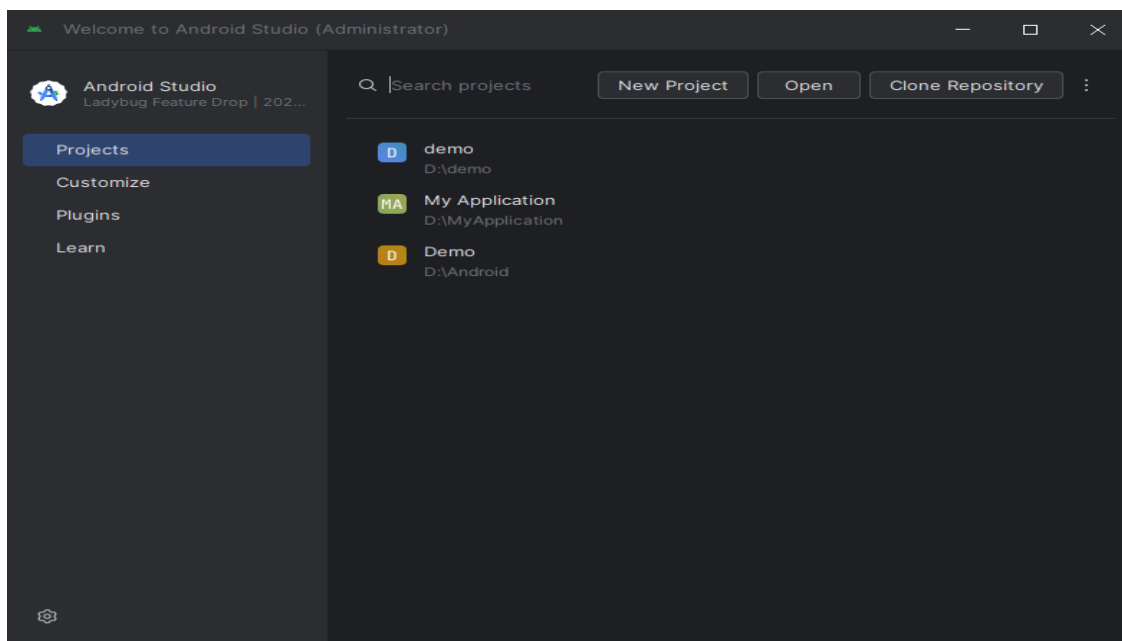
1. Truy cập đến **Android developers site** và làm theo hướng dẫn để tải xuống và cài đặt **Android Studio**
2. Chấp nhận cấu hình mặc định cho tất cả các bước và đảm bảo rằng tất cả các thành phần được chọn để cài đặt
3. Sau khi hoàn tất cài đặt, trình hướng dẫn cài đặt sẽ tải xuống và cài đặt một số thành phần bổ sung bao gồm SDK Android. Hãy kiên nhẫn, quá trình này có thể mất một chút thời gian tùy thuộc vào tốc độ Internet của bạn và một số bước có vẻ dư thừa
4. Khi quá trình tải xuống hoàn tất, Android Studio sẽ khởi động và bạn đã sẵn sàng tạo dự án đầu tiên của mình

**Xử lý sự cố:** Nếu bạn gặp sự cố khi cài đặt, hãy kiểm tra **Android Studio release notes** hoặc nhờ người hướng dẫn trợ giúp

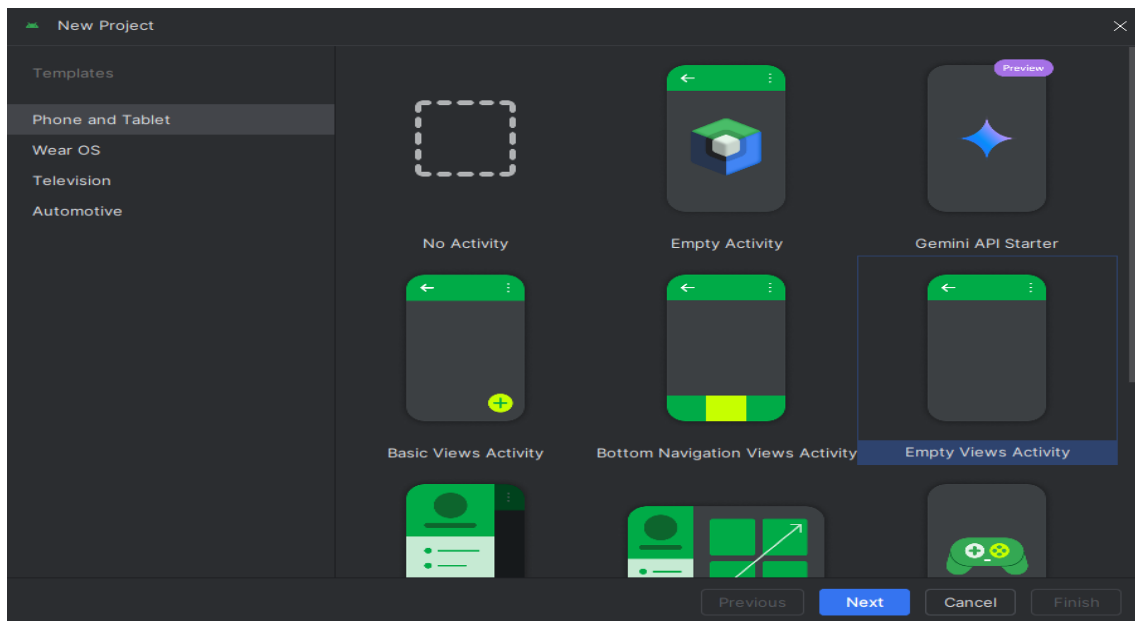
## Nhiệm vụ 2: Tạo ứng dụng Hello World

### 2.1. Tạo dự án ứng dụng

1. Mở Android Studio nếu nó chưa được mở
2. Trong cửa sổ chính **Welcome to Android Studio**, nhấp vào **New Project**

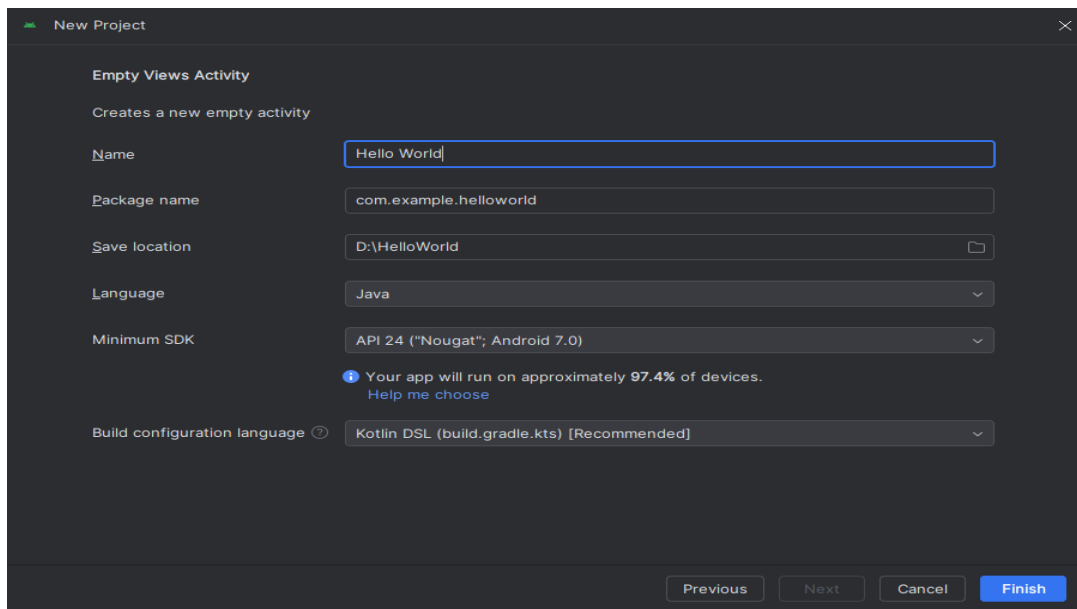


- Trong cửa sổ **New Project**, chọn Activity cho dự án. Activity là một việc người dùng có thể làm. Nó là một thành phần quan trọng của bất kỳ ứng dụng Android nào. Activity thường có bố cục được liên kết với nó xác định cách các thành phần UI xuất hiện trên màn hình. Android Studio cung cấp các mẫu Activity để giúp bạn bắt đầu. Đối với dự án Hello World thì chọn **Empty Views Activity** và sau đó ấn **Next**



- Trong cửa sổ **New Project**, nhập **Hello World** cho **Name**.
- Chấp nhận tên miền mặc định của công ty **com.example.helloworld** hoặc tạo một tên miền công ty độc quyền  
Nếu bạn không có kế hoạch xuất bản ứng dụng của mình, bạn có thể để mặc định. Lưu ý rằng việc thay đổi **Package name** của bạn sau này sẽ tốn thêm công sức
- Xác minh vị trí dự án mặc định **Save location** là nơi bạn muốn lưu trữ ứng dụng Hello World và các dự án Android Studio khác, hoặc thay đổi vị trí đó thành thư mục ưa thích của bạn
- Chọn **Language** là **java**
- Trong **Minimum SDK** được để mặc định là **API 24 ("Nougat", Android 7.0)**, thiết lập này làm cho ứng dụng Hello World của bạn tương thích với 97.4% thiết bị Android đang hoạt động trên Google Play Store. Nếu bạn không muốn hãy bật danh sách Minimum SDK lên để chọn
- Trong **Build configuration language** chọn **Kotlin DSL (build.gradle.kts)** và cuối cùng ấn Finish

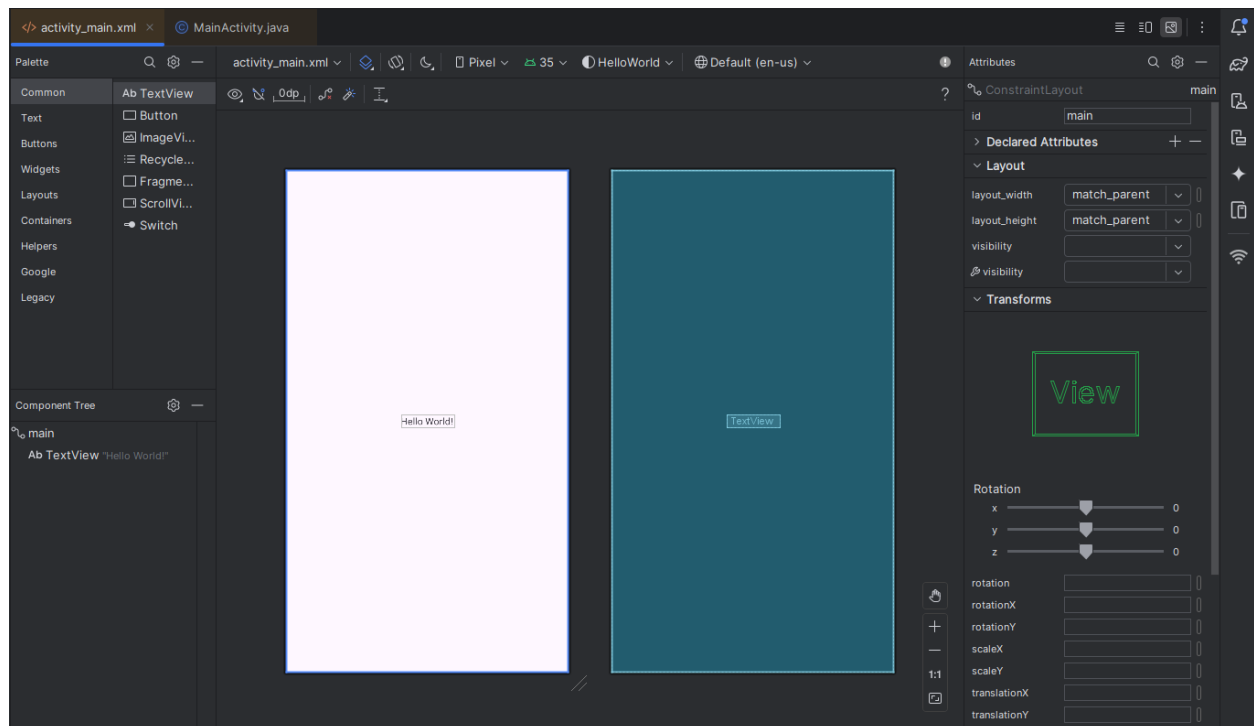




Android Studio tạo một thư mục cho các dự án của bạn và xây dựng dự án bằng Gradle (có thể mất vài phút)

Trình chỉnh sửa Android Studio sẽ xuất hiện. Làm theo các bước sau:

1. Nhấp vào tab **activity\_main.xml** để xem trình chỉnh sửa bố cục
2. Nhấp vào tab **Design**, nếu chưa chọn, biểu tượng đồ họa của bố cục hiển thị như hình dưới



3. Nhấp vào tab **MainActivity.java** để xem trình sửa code như hình dưới

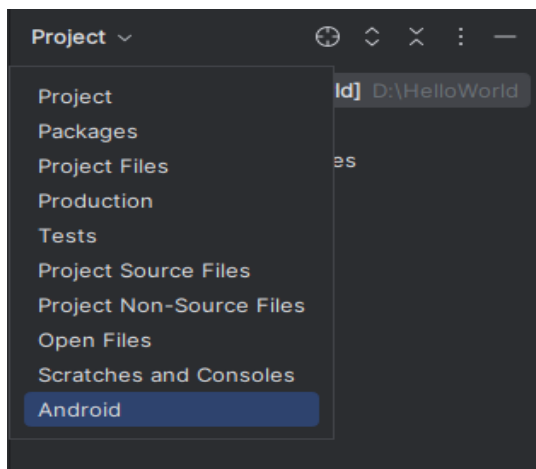


```
1 package com.example.helloworld;
2
3 > import
10
11 public class MainActivity extends AppCompatActivity {
12
13     @Override
14     protected void onCreate(Bundle savedInstanceState) {
15         super.onCreate(savedInstanceState);
16         EdgeToEdge.enable(this);
17         setContentView(R.layout.activity_main);
18         ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
19             Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
20             v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
21             return insets;
22         });
23     }
24 }
```

## 2.2. Khám phá dự án > Bảng điều khiển Android

Trong thực tế này, bạn sẽ khám phá cách tổ chức dự án trong Android Studio

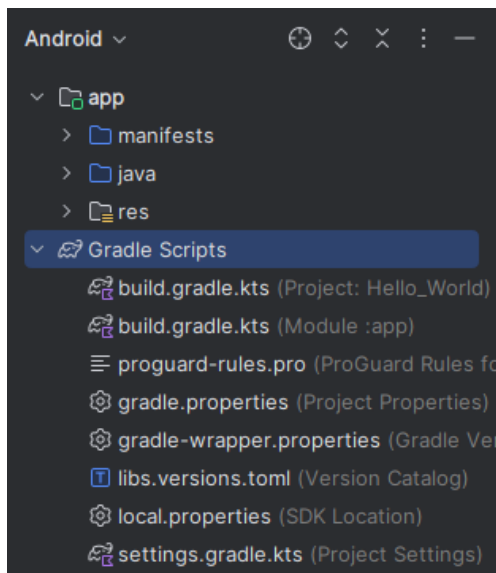
1. Nếu chưa chọn, hãy nhấp vào tab **Project** trong cột tab dọc ở phía bên trái của cửa sổ Android Studio. Ngăn Project xuất hiện
2. Để xem dự án trong hệ thống phân cấp dự án Android tiêu chuẩn, hãy chọn Android từ menu bật lên ở trong ngăn Project, như hình bên dưới



## 2.3. Khám phá thư mục Gradle Scripts

Hệ thống bản dựng Gradle trong Android Studio giúp bạn dễ dàng đưa các tệp nhị phân bên ngoài hoặc các mô-đun thư viện khác vào bản dựng dưới dạng phần phụ thuộc

Khi bạn tạo dự án ứng dụng lần đầu tiên, ngăn **Project > Android** sẽ xuất hiện với thư mục **Gradle Scripts** được mở rộng như hình minh họa bên dưới



Làm theo các bước sau để khám phá hệ thống Gradle:

1. Nếu thư mục **Gradle Scripts** chưa được mở rộng, hãy nhấp vào hình tam giác để mở rộng. Thư mục này chứa tất cả các tệp cần thiết cho hệ thống xây dựng

2. Tìm tệp **build.gradle.kts(Project:HelloWorld)**

Đây là nơi bạn sẽ tìm thấy các tùy chọn cấu hình chung cho tất cả các mô-đun tạo nên dự án của bạn. Mỗi dự án Android Studio đều chứa một tệp bản dựng Gradle cấp cao nhất. Hầu hết thời gian, bạn sẽ không cần thực hiện bất kỳ thay đổi nào đối với tệp này, nhưng nó vẫn hữu ích để hiểu nội dung của nó

Theo mặc định, tệp bản dựng cấp cao nhất sử dụng khối buildscript để xác định kho lưu trữ Gradle và các thành phần phụ thuộc chung cho tất cả các mô-đun trong dự án. Khi phần phụ thuộc của bạn không phải là thư viện cục bộ hoặc cây tệp, Gradle sẽ tìm kiếm các tệp trong bất kỳ kho lưu trữ trực tuyến nào được chỉ định trong khối kho lưu trữ của tệp này. Theo mặc định, các dự án Android Studio mới khai báo Jcenter và Google (bao gồm Google Maven repository) là các kho vị trí lưu trữ

```
plugins {  
    alias(libs.plugins.android.application) apply false  
}
```

3. Tìm tệp **build.gradle.kts(Module:app)**

Ngoài tệp build.gradle.kts cấp độ dự án, mỗi mô-đun cũng có một build.gradle.kts riêng, cho phép bạn cấu hình các cài đặt build cho từng mô-

đơn cụ thể (trong ứng dụng HelloWorld, chỉ có một mô-đun duy nhất). Việc cấu hình các cài đặt build này giúp bạn tùy chỉnh các tùy chọn đóng gói, chẳng hạn như bổ sung kiểu build và biến thể sản phẩm. Ngoài ra bạn cũng có thể ghi đè các thiết lập trong tệp `AndroidManifest.xml` hoặc tệp `build.gradle` ở cấp dự án

Tệp này thường là tệp cần chỉnh sửa khi thay đổi cấu hình cấp ứng dụng, chẳng hạn như khai báo các phần phụ thuộc trong phần phụ thuộc. Bạn có thể khai báo phần phụ thuộc thư viện bằng cách sử dụng một trong một số cấu hình phần phụ thuộc khác nhau. Mỗi cấu hình phần phụ thuộc cung cấp cho Gradle các hướng dẫn khác nhau về cách sử dụng thư viện. Ví dụ: câu lệnh thực hiện `fileTree` (`dir: 'libs', include: ['*.jar']`) thêm phần phụ thuộc của tất cả các tệp `“.jar”` bên trong thư mục `libs`

Sau đây là tệp **`build.gradle.kts(Module:app)`**:

```

plugins { alias(libs.plugins.android.application) }
android {
    namespace = "com.example.helloworld"
    compileSdk = 35
    defaultConfig {
        applicationId = "com.example.helloworld"
        minSdk = 24
        targetSdk = 35
        versionCode = 1
        versionName = "1.0"
        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner" }
    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("name: "proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_11
        targetCompatibility = JavaVersion.VERSION_11
    }
}
dependencies {
    implementation(libs.appcompat)
    implementation(libs.material)
    implementation(libs.activity)
    implementation(libs.constraintlayout)
    testImplementation(libs.junit)
    androidTestImplementation(libs.ext.junit)
    androidTestImplementation(libs.espresso.core)
}

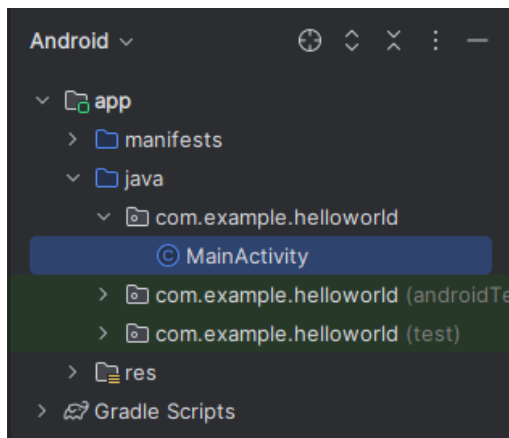
```

4. Nhấp vào hình tam giác để đóng **Gradle Scripts**

## 2.4. Khám phá ứng dụng và thư mục res

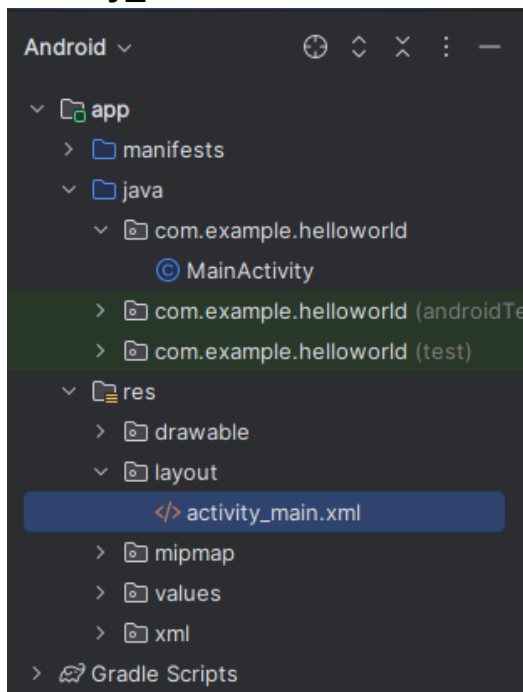
Tất cả mã nguồn và tài nguyên cho ứng dụng đều nằm trong thư mục app và res

1. Mở rộng thư mục **app**, thư mục **java**, và thư mục **com.example.android.helloworld** để xem tệp java **MainActivity**. Nhấp đúp vào tệp để mở nó trong trình chỉnh sửa mã nguồn



Thư mục **java** bao gồm các tệp lớp java trong ba thư mục con, như hình trên. Thư mục **com.example.android.helloworld** (hoặc tên miền bạn chỉ định) chứa tất cả các tệp cho gói ứng dụng. Hai thư mục còn lại được sử dụng để kiểm tra và được đề cập trong một bài học khác. Đối với ứng dụng Hello World, chỉ có một gói duy nhất, trong đó chứa MainActivity.java. Tên của Activity đầu tiên (màn hình đầu tiên mà người dùng nhìn thấy), đồng thời khởi tạo các tài nguyên dùng chung cho toàn bộ ứng dụng, theo thông lệ thường được đặt là **MainActivity** (phần mở rộng tệp được ẩn trong phần **Project > Android**)

2. Mở rộng thư mục **res** và thư mục **layout** đồng thời nhấp đúp vào tệp **activity\_main.xml** để mở nó trong trình chỉnh sửa bố cục

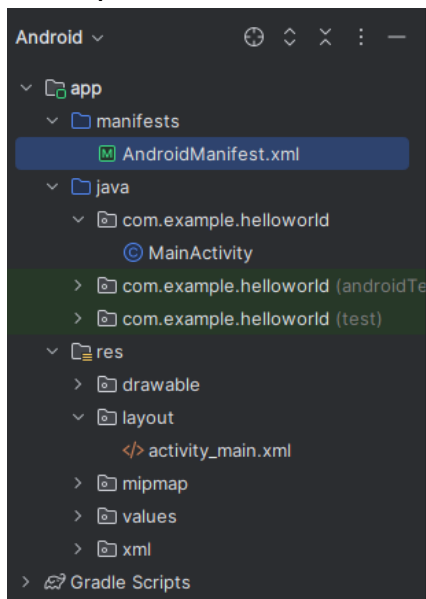


Thư mục **res** chứa các tài nguyên, chẳng hạn như bố cục, chuỗi kí tự và hình ảnh. Một Activity thường được liên kết với bố cục giao diện người dùng, được định nghĩa trong một tệp XML. Tệp này thường được đặt tên theo Activity tương ứng

## 2.5. Khám phá thư mục manifests

Thư mục manifests chứa các tệp cung cấp thông tin cần thiết về ứng dụng của bạn cho hệ thống Android, hệ thống phải có thông tin này trước khi có thể chạy bất kỳ mã nguồn nào của ứng dụng

1. Mở rộng thư mục **manifests**
2. Mở tệp **AndroidManifest.xml**



Tệp AndroidManifest.xml mô tả tất cả các thành phần của ứng dụng Android của bạn. Tất cả các thành phần của một ứng dụng, chẳng hạn như mỗi Activity phải được khai báo trong tệp XML này. Trong các bài học khác của khóa học, bạn sẽ chỉnh sửa tệp này để thêm tính năng và quyền truy cập tính năng. Để tìm hiểu tổng quan, hãy xem **App Manifest Overview**

### Nhiệm vụ 3: Sử dụng thiết bị ảo (trình giả lập)


Trong nhiệm vụ này, bạn sẽ sử dụng **Android Virtual Device (AVD) manager** để tạo một thiết bị ảo (còn được gọi là trình mô phỏng) mô phỏng cấu hình cho một loại thiết bị Android cụ thể và sử dụng thiết bị ảo đó để chạy ứng dụng. Xin lưu ý rằng trình mô phỏng Android có các yêu cầu bổ sung ngoài các yêu cầu hệ thống cơ bản đối với Android Studio

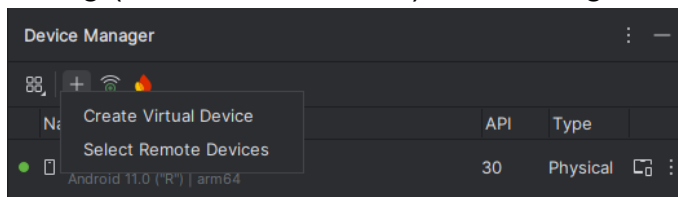
Khi sử dụng Trình quản lý AVD, bạn xác định các đặc điểm phần cứng của thiết bị, cấp độ API, bộ nhớ, giao diện và các thuộc tính khác, lưu lại dưới dạng một thiết bị ảo. Với thiết bị ảo, bạn có thể kiểm thử ứng dụng trên nhiều cấu hình thiết bị khác nhau (chẳng hạn như máy tính bảng và điện thoại) với các cấp độ API khác nhau mà không cần phải sử dụng thiết bị vật lý.

### 3.1. Tạo thiết bị ảo Android (AVD)

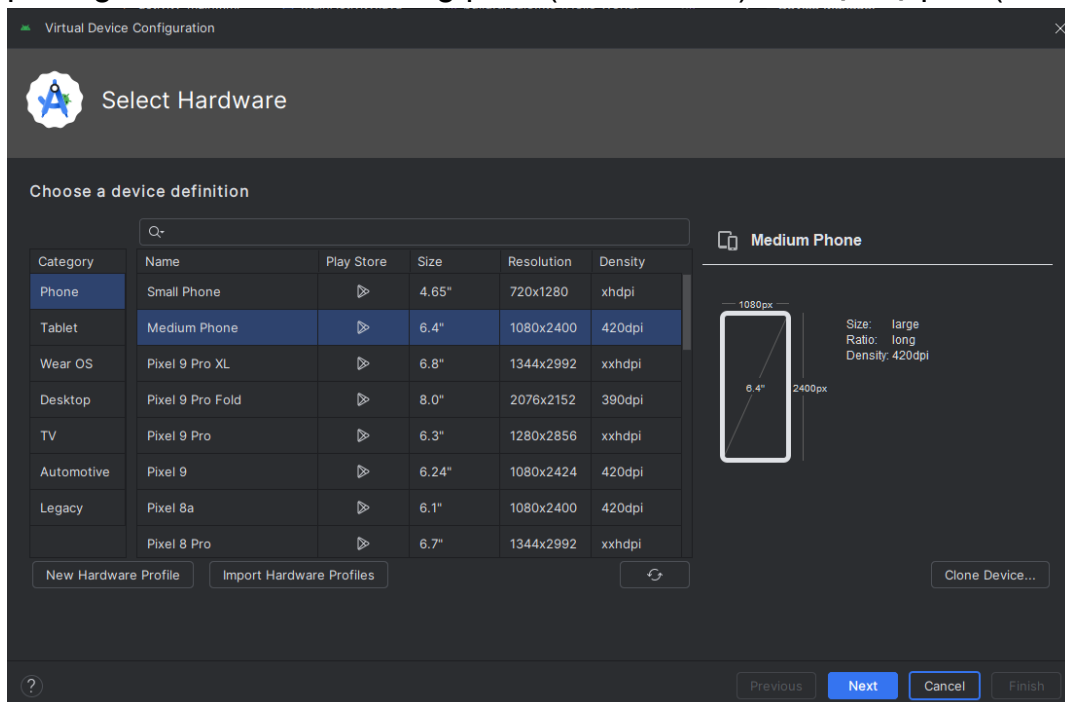
Để chạy trình mô phỏng trên máy tính, bạn phải tạo một cấu hình mô tả thiết bị ảo.

1. Trong Android Studio, chọn **Tools > Android > AVD Manager**, hoặc nhấp

vào biểu tượng Trình quản lý AVD  trên thanh công cụ. Màn hình Your Virtual Devices xuất hiện. Nếu bạn đã tạo thiết bị ảo, màn hình sẽ hiển thị chúng (như hình bên dưới), nếu không bạn sẽ thấy một danh sách trống.

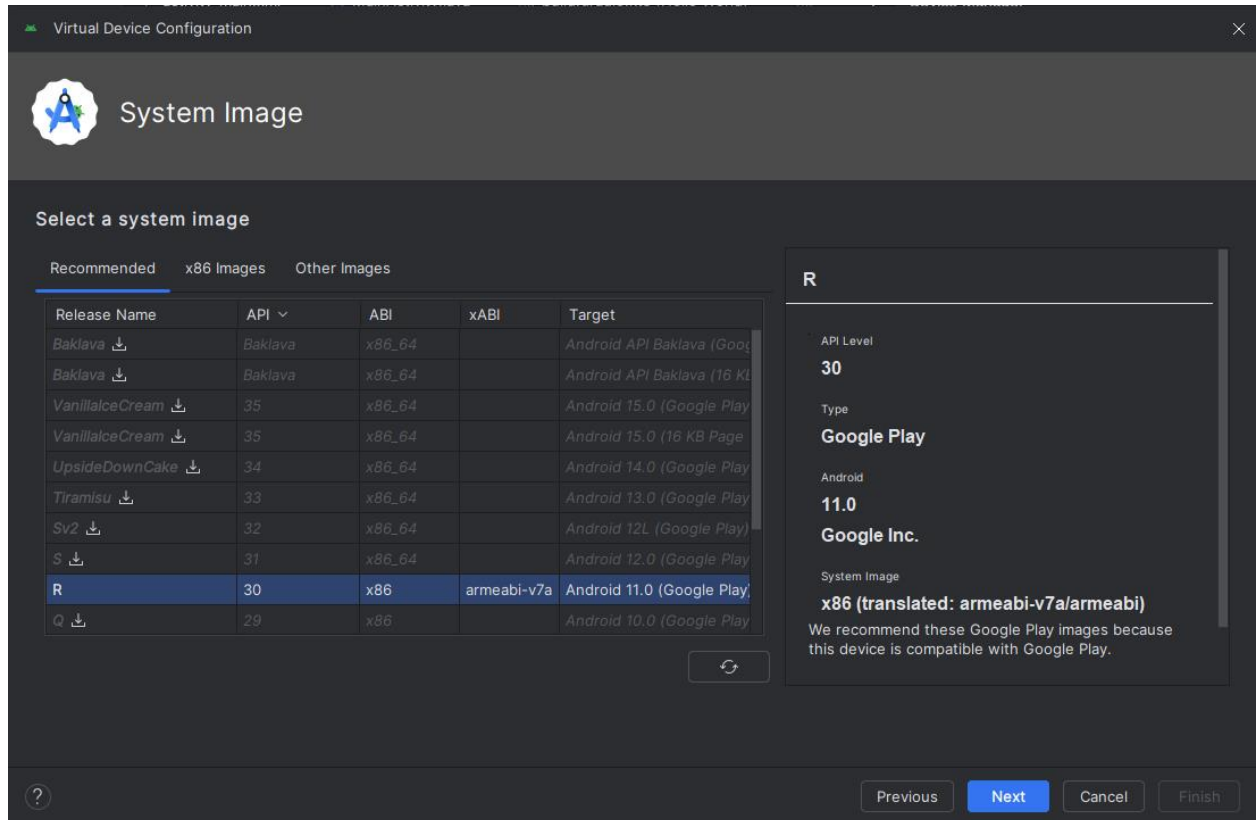


2. Nhấp vào **+Create Virtual Device**. Cửa sổ **Select Hardware** xuất hiện hiển thị danh sách các thiết bị phần cứng được cấu hình sẵn. Đối với mỗi thiết bị, bảng cung cấp một cột cho kích thước màn hình chéo (**Size**), độ phân giải màn hình tính bằng pixel (**Resolution**) và mật độ pixel (**Density**).





3. Chọn một thiết bị như **Medium Phone** rồi nhấp vào Next. Màn hình **System Image** sẽ xuất hiện
4. Nhấp vào tab **Recommended** nếu chưa được chọn và chọn phiên bản hệ thống Android để chạy thiết bị ảo (**R**)



Có nhiều phiên bản hơn được hiển thị trong tab **Recommended**. Nhấp vào tab **x86 Images** và **Other Images** để xem thêm

Nếu có liên kết **Download** hiển thị bên cạnh một hình ảnh hệ thống mà bạn muốn sử dụng, thì liên kết đó chưa được cài đặt. Nhấp vào liên kết để bắt đầu tải xuống và nhấp **Finish** khi hoàn tất

5. Sau khi chọn hình ảnh hệ thống, nhấp **Next**. Cửa sổ thiết bị ảo **Android (AVD)** sẽ xuất hiện. Bạn cũng có thể thay đổi tên của AVD. Kiểm tra cấu hình của bạn và nhấp vào **Finish**

## 3.2. Chạy ứng dụng trên thiết bị ảo

Trong nhiệm vụ này, bạn sẽ chạy ứng dụng Hello World của mình

1. Trong Android Studio, chọn **Run > Run app** hoặc nhấp vào biểu tượng **Run**



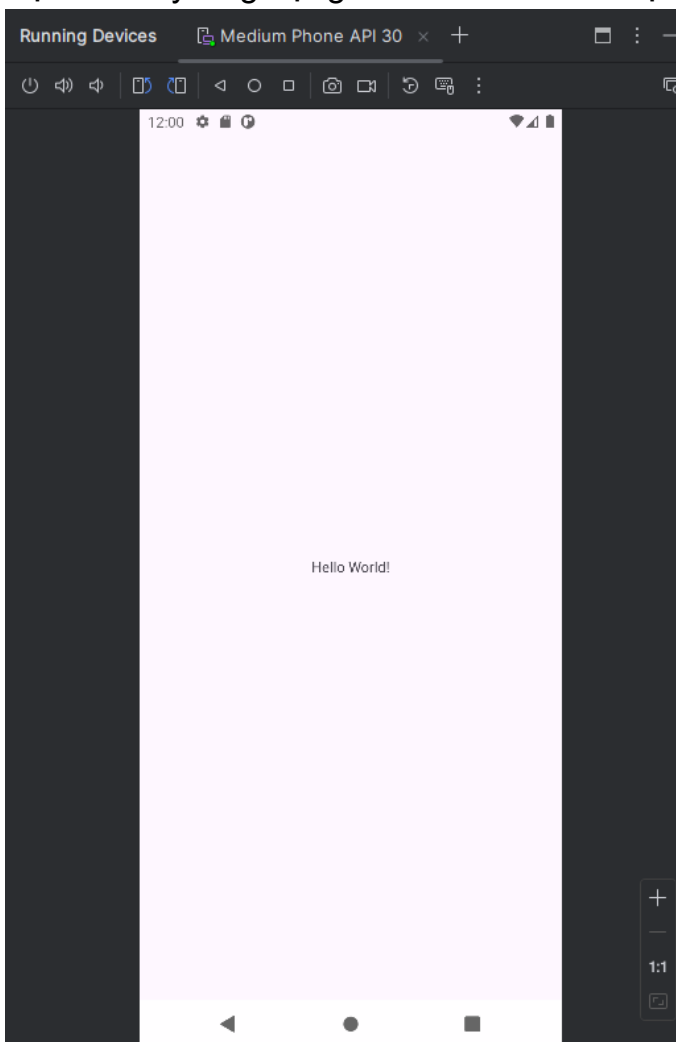
trên thanh công cụ

2. Trong cửa sổ **Select Deployment Target**, dưới mục **Available Virtual Devices**, chọn thiết bị ảo mà bạn vừa tạo



Trình giả lập sẽ khởi động như một thiết bị vật lý Tùy vào tốc độ của máy tính, quá trình này có thể mất một chút thời gian. Ứng dụng của bạn sẽ được biên dịch và khi trình giả lập sẵn sàng, Android Studio sẽ tải ứng dụng lên trình giả lập và chạy nó.

Bạn sẽ thấy ứng dụng Hello World hiển thị như trong hình sau



## Nhiệm vụ 4: (Tùy chọn) Sử dụng thiết bị vật lý

Trong nhiệm vụ cuối cùng này, bạn sẽ chạy ứng dụng của mình trên thiết bị vật lý như điện thoại hoặc máy tính bảng. Bạn phải luôn kiểm tra ứng dụng của mình trên cả thiết bị ảo và thiết bị vật lý

Những gì bạn cần:

- Thiết bị android như điện thoại hoặc máy tính bảng
- Cáp dữ liệu để kết nối thiết bị android với máy tính qua cổng USB
- Nếu bạn đang sử dụng hệ thống Linux hoặc Windows, bạn có thể cần thực hiện các bước bổ sung để chạy trên thiết bị phần cứng. Kiểm tra tài liệu **Using Hardware Devices**
- Bạn cũng có thể cần cài đặt trình điều khiển USB phù hợp cho thiết bị của mình. Đối với trình điều khiển USB trên Windows, xem **OEM USB Drivers**

### 4.1. Bật gỡ lỗi USB


Để cho phép Android Studio giao tiếp với thiết bị của bạn, bạn phải bật tính năng USB Debugging trên thiết bị Android của mình. Tính năng này được bật trong phần **Developer options** trên thiết bị của bạn.

Trên Android 4.2 trở lên, màn hình **Developer options** bị ẩn theo mặc định. Để hiển thị tùy chọn này và bật USB Debugging:

1. Trên thiết bị của bạn, mở **Settings**, tìm kiếm **About phone**, nhấn vào **About phone** và nhấn **Build number** bảy lần liên tiếp
2. Quay lại màn hình trước đó (**Settings / System**). **Developer options** xuất hiện trong danh sách. Nhấn vào **Developer options**
3. Chọn **USB Debugging**

### 4.2. Chạy ứng dụng của bạn trên thiết bị

Giờ bạn có thể kết nối thiết bị và chạy ứng dụng từ Android Studio

1. Kết nối thiết bị của bạn với máy tính phát triển bằng cáp USB
2. Nhấp vào nút Run  trên thanh công cụ. Cửa sổ **Select Deployment Target** sẽ mở ra với danh sách các trình giả lập có sẵn và thiết bị được kết nối
3. Chọn thiết bị của bạn, nhấp **OK**

Android Studio sẽ cài đặt và chạy ứng dụng trên thiết bị của bạn

## Xử lý sự cố

Nếu Android Studio không nhận diện được thiết bị của bạn, hãy thử các cách sau:

1. Rút cáp và cắm lại thiết bị
2. Khởi động lại Android Studio

Nếu máy tính vẫn không tìm thấy thiết bị hoặc hiển thị trạng thái “unauthorized”, hãy làm theo các bước sau:

1. Rút cáp khỏi thiết bị
2. Trên thiết bị, mở **Developer options in Settings app**
3. Nhấn vào thu hồi quyền **USB Debugging**
4. Kết nối lại thiết bị với máy tính
5. Khi được nhắc cấp ủy quyền

Bạn có thể cần cài đặt trình điều khiển USB thích hợp cho thiết bị của mình. Xem **Using Hardware Devices documentation**

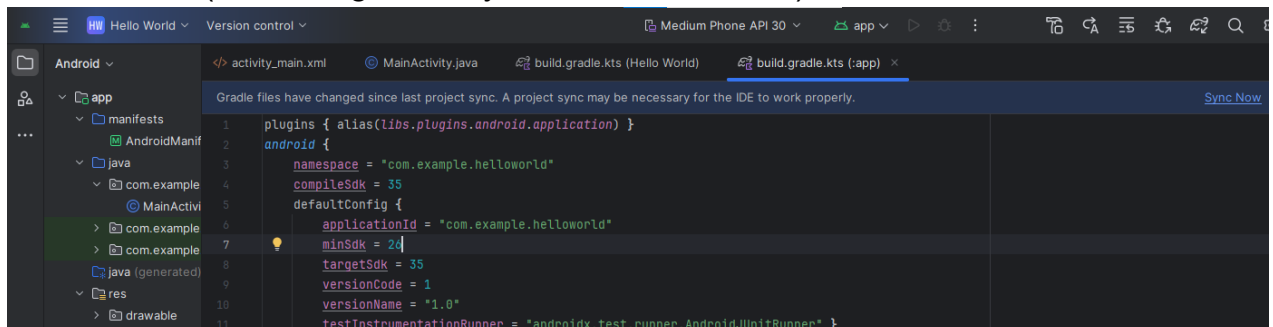
## Nhiệm vụ 5: Thay đổi cấu hình Gradle của ứng dụng

Trong nhiệm vụ này, bạn sẽ thay đổi một số điều về cấu hình ứng dụng trong tệp `build.gradle.kts(Module:app)` để tìm hiểu cách thực hiện các thay đổi và đồng bộ hóa chúng với các dự án Android Studio của bạn

### 5.1. Thay đổi phiên bản minimum SDK cho ứng dụng

Làm theo các bước sau:

1. Mở thư mục **Gradle Scripts** nếu thư mục chưa mở và nhấp vào tệp **build.gradle.kts(Module:app)**  
Nội dung của tệp tin xuất hiện trong trình soạn thảo mã
2. Trong khối `defaultConfig`, hãy thay đổi giá trị của `minSdk` thành 26 như hiển thị bên dưới (ban đầu giá trị này được đặt thành 24)



Trình chỉnh sửa mã hiển thị thanh thông báo ở trên cùng với liên kết **Sync Now**

## 5.2. Đồng bộ cấu hình Gradle mới

Khi bạn thực hiện thay đổi đối với các tệp cấu hình xây dựng trong một dự án, Android Studio yêu cầu bạn *đồng bộ hóa* các tệp dự án để có thể nhập các thay đổi cấu hình bản dựng và chạy một số kiểm tra để đảm bảo cấu hình sẽ không tạo ra lỗi bản dựng

Để đồng bộ các tệp dự án, hãy nhấp vào **Sync Now** trên thanh thông báo xuất hiện khi thực hiện thay đổi (như thể hiện trong hình trước) hoặc nhấn vào biểu

tượng **Sync Project with Gradle Files**  trong thanh công cụ


Khi quá trình đồng bộ hóa Gradle hoàn tất, thông báo Gradle build finished sẽ xuất hiện ở góc dưới bên trái của cửa sổ Android Studio

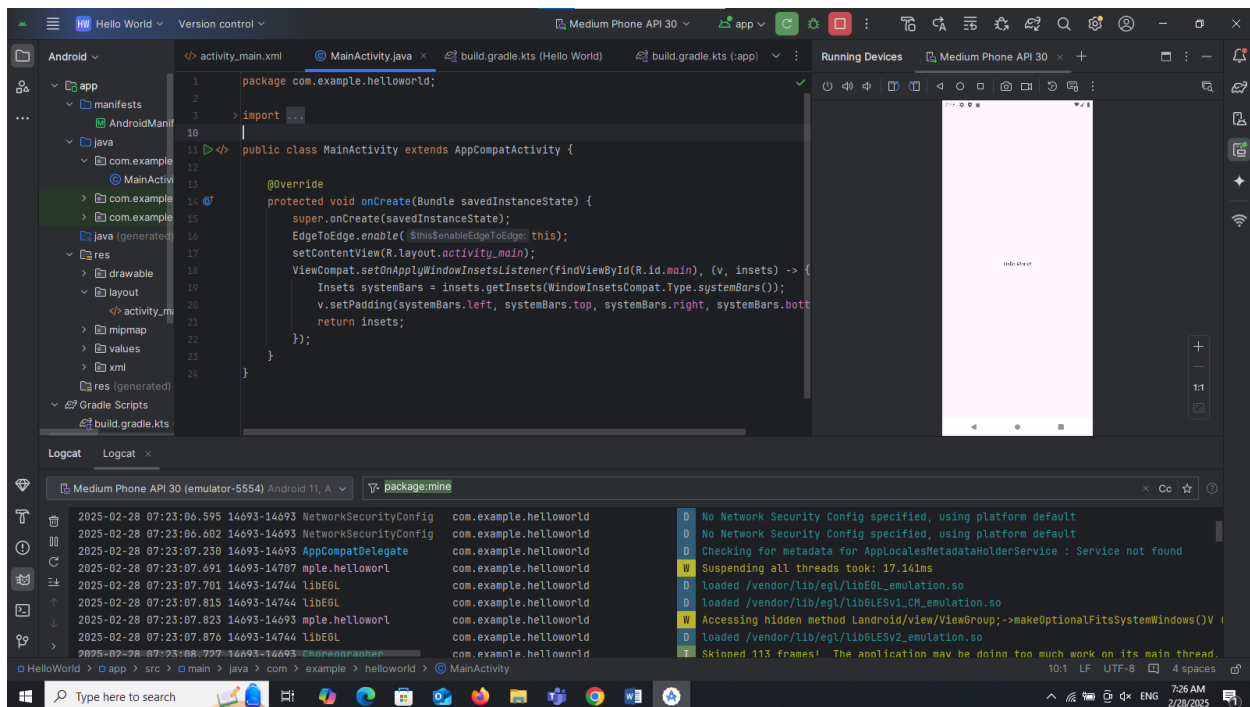
Để hiểu sâu hơn về Gradle, hãy tham khảo tài liệu **Build System Overview** và **Configuring Gradle Builds**

## Nhiệm vụ 6: Thêm log statement vào ứng dụng của bạn

Trong tác vụ này, bạn sẽ thêm các câu lệnh **Log** vào ứng dụng của mình, hiển thị các thông báo trong ngăn **Logcat**. Thông báo Log là một công cụ gỡ lỗi mạnh mẽ mà bạn có thể sử dụng để kiểm tra các giá trị, đường dẫn thực thi và báo cáo các ngoại lệ

### 6.1. Xem ngăn Logcat

Để xem ngăn **Logcat**, hãy nhấp vào biểu tượng **Logcat**  ở thanh công cụ bên trái màn hình Android Studio như minh họa trong hình bên dưới



Trong hình trên:

1. Tab **Logcat** để mở và đóng ngăn **Logcat**, hiển thị thông tin về ứng dụng của bạn khi ứng dụng đang chạy. Nếu bạn thêm câu lệnh Log vào ứng dụng, thông báo Log sẽ xuất hiện ở đây
2. Menu cấp độ của Log được để mặc định, hiển thị tất cả các thông báo Log. Các thiết lập bao gồm **Debug**, **Error**, **Info** và **Warn**

## 6.2 Thêm câu lệnh Log vào ứng dụng của bạn

Các câu lệnh log trong mã ứng dụng của bạn hiển thị thông báo trong ngăn Logcat. Ví dụ:

```
Log.d( tag: "MainActivity", msg: "Hello World");
```

Các phần của tin nhắn bao gồm:

- Log: Lớp **Log** để gửi tin nhắn log đến ngăn Logcat
- d: Cài đặt mức **Debug** Log để lọc hiển thị thông báo log trong ngăn Logcat. Các mức log khác là e cho **Error**, w cho **Warn** và i cho **Info**
- "MainActivity": Đối số đầu tiên là một thẻ có thể được sử dụng để lọc tin nhắn trong ngăn Logcat. Đây thường là tên của Activity mà tin nhắn bắt nguồn. Tuy nhiên, bạn có thể biến nó thành bất kỳ thứ gì hữu ích cho bạn để gỡ lỗi

Theo quy ước, thẻ log được định nghĩa là hằng số cho Activity:

```
private static final String LOG_TAG = MainActivity.class.getSimpleName();|
```

- "Hello World": Đối số thứ hai là thông điệp thực tế

Làm theo các bước sau:

1. Mở ứng dụng Hello World của bạn trong Android studio và mở MainActivity.
2. Để tự động thêm các lệnh nhập rõ ràng vào dự án của bạn (chẳng hạn như android.util.Log cần thiết để sử dụng Log), hãy chọn **File > Settings** trong Windows hoặc **Android Studio > Preferences** trong macOS
3. Chọn **Editor > General > Auto Import**. Chọn tất cả các hộp kiểm và thiết lập **Insert imports on paste to All**
4. Chọn **Apply** và sau đó nhấn **OK**
5. Trong phương thức onCreate() của MainActivity, thêm câu lệnh sau:

```
Log.d( tag: "MainActivity", msg: "Hello World");
```

Phương thức onCreate() bây giờ sẽ trông giống như đoạn mã sau:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable( $this$enableEdgeToEdge: this);
    setContentView(R.layout.activity_main);
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
        Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
        v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
        return insets;
    });
    Log.d( tag: "MainActivity", msg: "Hello World");
}
```

6. Nếu ngăn Logcat chưa mở, hãy nhấp vào biểu tượng Logcat ở thanh công cụ bên trái Android Studio để mở
7. Kiểm tra xem tên mục tiêu và tên gói của ứng dụng có đúng không
8. Thay đổi mức Log trong ngăn **Logcat** thành **Debug** (hoặc giữ nguyên vì có rất ít thông báo log)
9. Chạy ứng dụng của bạn

Thông báo sau sẽ xuất hiện trong ngăn Logcat:

```
2025-02-28 07:43:40.972 15720-15720 MainActivity com.example.helloworld D Hello World
```

## Câu hỏi 1

Tên của tệp bố cục cho main activity là gì?

- MainActivity.java
- AndroidManifest.xml
- **activity\_main.xml**
- build.gradle

## Câu hỏi 2

Tên của chuỗi tài nguyên chỉ định tên ứng dụng là gì?

- **app\_name**
- xmlns:app
- android:name
- applicationId

## Câu hỏi 3

Bạn sử dụng công cụ nào để tạo trình giả lập mới?

- Android Device Monitor
- **AVD Manager**
- SDK Manager
- Theme Editor

## Câu hỏi 4

Giả sử ứng dụng của bạn bao gồm câu lệnh log này:

```
Log.i("MainActivity", "MainActivity layout is complete");
```

Bạn thấy câu lệnh "MainActivity layout is complete" trong ngăn **Logcat** nếu menu cấp độ Log được đặt thành tùy chọn nào sau đây? (Gợi ý: trả lời nhiều câu hỏi là được.)

- **Verbose**
- **Debug**



- Info
- Warn
- Error
- Assert

## 1.2) Giao diện người dùng tương tác đầu tiên

### Giới thiệu

Giao diện người dùng (UI) xuất hiện trên màn hình của thiết bị Android bao gồm một hệ thống phân cấp các đối tượng được gọi là *ché độ xem* - mọi thành phần của màn hình là một **View**. Lớp View biểu thị khối xây dựng cơ bản cho tất cả các thành phần UI và là lớp cơ sở cho các lớp cung cấp các thành phần UI tương tác như buttons, checkboxes và text entry fields. Các lớp con View thường được sử dụng được mô tả trong nhiều bài học bao gồm:

- **TextView** để hiển thị văn bản
- **EditText** để cho phép người dùng nhập và chỉnh sửa văn bản
- **Button** và các thành phần có thể nhấp khác (như **RadioButton**, **CheckBox** và **Spinner**) để cung cấp hành vi tương tác
- **ScrollView** và **RecyclerView** để hiển thị các mục có thể cuộn
- **ImageView** để hiển thị hình ảnh
- **ConstraintLayout** và **LinearLayout** để chứa các thành phần View khác và định vị chúng

Đoạn mã Java hiển thị và điều khiển giao diện người dùng (UI) được chứa trong một lớp mở rộng từ **Activity**. Một **Activity** thường được liên kết với một bố cục của các thành phần giao diện người dùng được định nghĩa trong một tệp XML (eXtended Markup Language). Tệp XML này thường được đặt tên theo tên của **Activity** và định nghĩa bố cục của các thành phần **View** trên màn hình

Ví dụ, mã MainActivity trong ứng dụng Hello World hiển thị một bố cục được định nghĩa trong tệp bố cục activity\_main.xml, trong đó bao gồm một TextView với nội dung "Hello World".

Trong các ứng dụng phức tạp hơn, một Activity có thể triển khai các hành động để phản hồi thao tác chạm của người dùng, vẽ nội dung đồ họa, hoặc yêu cầu dữ liệu từ cơ sở dữ liệu hoặc internet. Bạn sẽ tìm hiểu thêm về lớp Activity trong một bài học khác

Trong bài thực hành này, bạn sẽ học cách tạo ứng dụng tương tác đầu tiên của mình - một ứng dụng cho phép tương tác với người dùng. Bạn sẽ tạo ứng dụng bằng mẫu Empty Activity. Đồng thời, bạn cũng học cách sử dụng trình chỉnh sửa bố cục (layout editor) để thiết kế bố cục và chỉnh sửa bố cục trong XML. Việc phát triển những kỹ năng này là cần thiết để bạn hoàn thành các bài thực hành khác trong khóa học này

## Những điều bạn nên biết

Bạn cần làm quen với:

- Cách cài đặt và mở Android Studio
- Cách tạo ứng dụng HelloWorld
- Cách chạy ứng dụng HelloWorld

## Những gì bạn sẽ học

- Cách tạo một ứng dụng với hành vi tương tác
- Cách sử dụng layout editor để thiết kế bố cục
- Cách chỉnh sửa bố cục trong XML
- Rất nhiều thuật ngữ mới. Hãy tham khảo **Vocabulary words and concepts glossary** để có các định nghĩa dễ hiểu

## Những gì bạn sẽ làm

- Tạo một ứng dụng và thêm hai phần tử Button cùng một TextView vào bố cục
- Điều chỉnh từng phần tử trong **ConstraintLayout** để ràng buộc chúng vào lề (margins) và các phần tử khác
- Thay đổi thuộc tính của các phần tử giao diện người dùng (UI)
- Chỉnh sửa bố cục của ứng dụng trong XML
- Trích xuất các chuỗi được mã hóa thành tài nguyên chuỗi (string resources)
- Triển khai các phương thức xử lý sự kiện nhấn để hiển thị thông báo trên màn hình khi người dùng nhấn vào từng Button

## Tổng quan về ứng dụng

Ứng dụng **HelloToast** bao gồm hai phần tử **Button** và một **TextView**. Khi người dùng nhấn vào **Button** đầu tiên, một thông báo ngắn (**Toast**) sẽ hiển thị trên màn hình. Nhấn vào **Button** thứ hai sẽ tăng giá trị bộ đếm "click" được hiển thị trong **TextView**, bắt đầu từ số không.

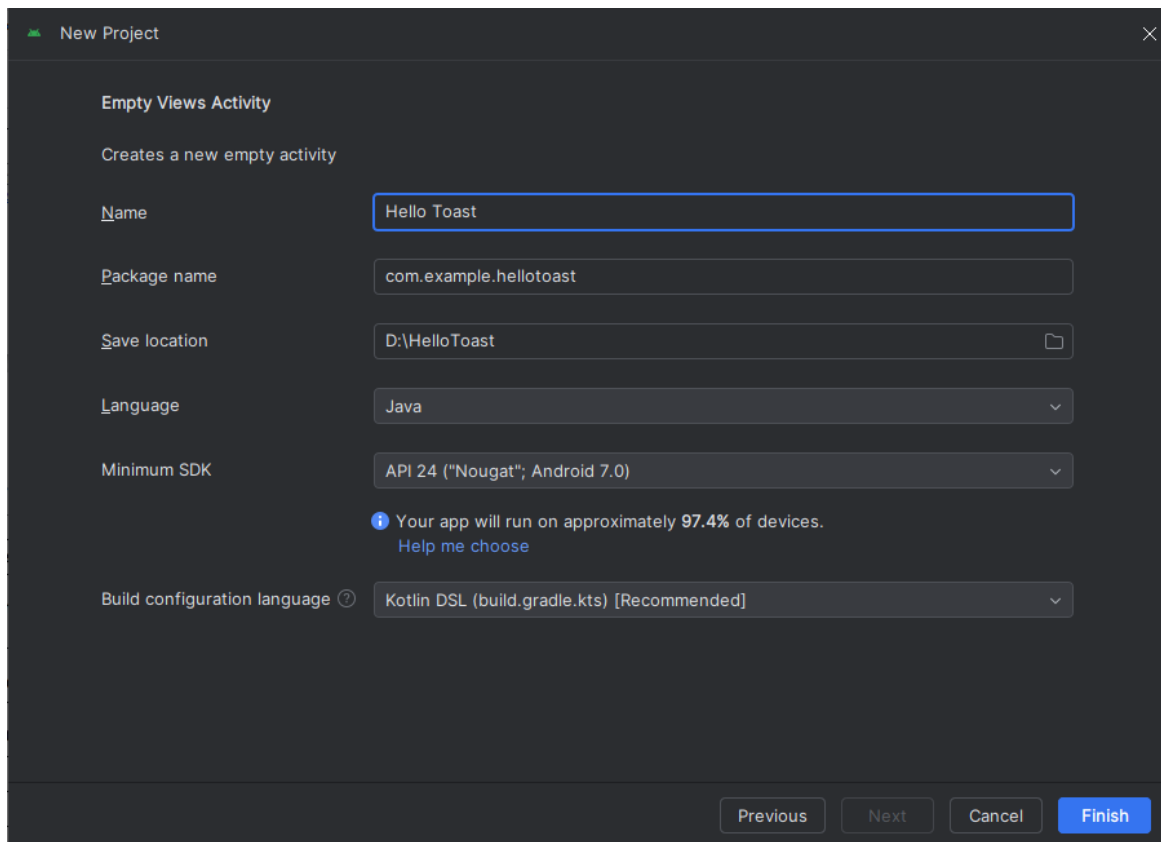
Sau đây là hình ảnh ứng dụng đã hoàn thành:


## Nhiệm vụ 1: Tạo và khám phá một dự án mới

Trong bài thực hành này, bạn thiết kế và triển khai một dự án cho ứng dụng HelloToast. Một liên kết đến mã giải pháp được cung cấp ở cuối

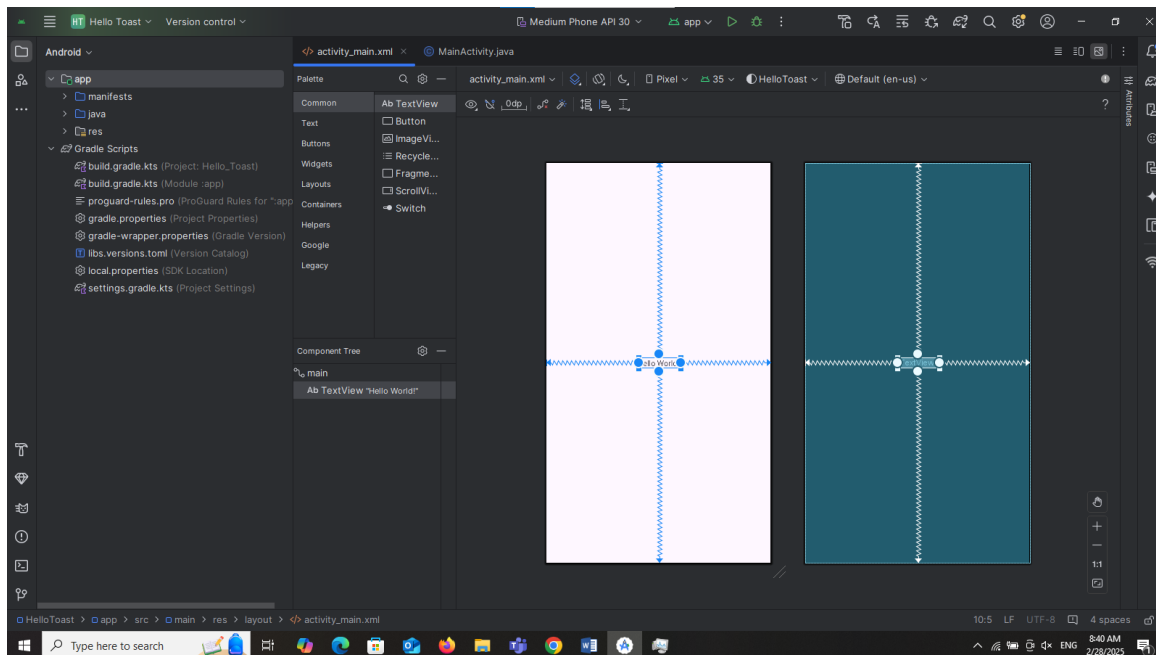
### 1.1 Tạo dự án Android Studio

14. Khởi động Android Studio và tạo một dự án mới với các tham số sau:



15. Chọn **Run > Run app** hoặc nhấp vào biểu tượng **Run**  trên thanh công cụ để biên dịch và chạy ứng dụng trên trình giả lập hoặc thiết bị của bạn

### 1.2 Khám phá trình chỉnh sửa bố cục



1. Trong thư mục **app > res > layout** của **Project > Android** pane, nhấn đúp vào tệp **activity\_main.xml** để mở nó nếu tệp chưa được mở
2. Nhấp vào tab **Design** nếu tab này chưa được chọn. Bạn sử dụng tab **Design** để thao tác với các phần tử và bố cục, và sử dụng tab **Text** để chỉnh sửa mã XML cho bố cục
3. Ngăn **Palettes** hiển thị các phần tử UI mà bạn có thể sử dụng trong bố cục của ứng dụng
4. Ngăn **Component tree** hiển thị cấu trúc phân cấp của các phần tử UI. Các phần tử View được tổ chức thành một cây phân cấp gồm các phần tử cha và con, trong đó phần tử con kế thừa các thuộc tính của phần tử cha. Trong hình minh họa, TextView là một phần tử con của **ConstraintLayout**. Bạn sẽ học thêm về các phần tử này trong bài học sau
5. Các ngăn design và blueprint của layout editor hiển thị các phần tử UI trong bố cục. Trong hình minh họa, bố cục chỉ hiển thị một phần tử: một TextView hiển thị "Hello World"
6. Tab **Attributes** hiển thị ngăn **Attributes**, nơi bạn có thể thiết lập các thuộc tính cho một phần tử UI

## Nhiệm vụ 2: Thêm các thành phần View vào trình chỉnh sửa bố cục


Trong nhiệm vụ này, bạn sẽ tạo bố cục giao diện người dùng cho ứng dụng HelloToast trong layout editor bằng cách sử dụng các tính năng của **ConstraintLayout**. Bạn có thể tạo các ràng buộc thủ công, như được minh họa sau đó, hoặc tự động bằng công cụ **Autoconnect**.


## 2.1 Kiểm tra các ràng buộc của phần tử

Thực hiện các bước sau:

1. Mở tệp **activity\_main.xml** từ **Project > Android** pane nếu tệp này chưa được mở. Nếu tab **Design** chưa được chọn, hãy nhấp vào đó

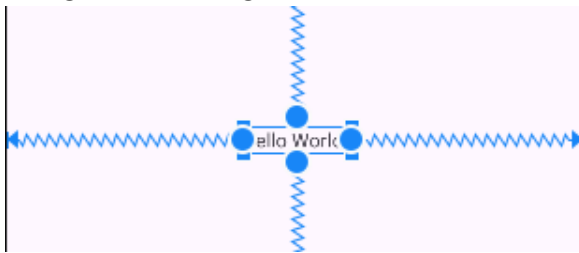
Nếu không có bản thiết kế (**blueprint**), nhấp vào nút **Select Design Surface** trên thanh công cụ và chọn **Design + Blueprint**

2. Công cụ **Autoconnect**  cũng nằm trên thanh công cụ và nó được bật theo mặc định. Đối với bước này, đảm bảo rằng công cụ này không tắt

3. Nhấp vào nút Zoom  in để phóng to các ngăn thiết kế và bản thiết kế để quan sát kỹ hơn

4. Chọn **TextView** trong ngăn Component Tree. Phần tử TextView "Hello World" sẽ được làm nổi bật trong cả hai ngăn design và blueprint, và các ràng buộc của phần tử sẽ hiển thị

5. Làm theo hình minh họa động dưới đây: Nhấp vào biểu tượng hình tròn ở phía bên phải của TextView để xóa ràng buộc ngang kết nối phần tử này với cạnh phải của bố cục. TextView sẽ chuyển sang phía bên trái vì nó không còn bị ràng buộc vào cạnh phải. Để thêm lại ràng buộc ngang, nhấp vào cùng biểu tượng hình tròn và kéo một đường tới cạnh phải của bố cục



Trong các ngăn **blueprint** hoặc **design**, các tay nắm (**handles**) sau sẽ xuất hiện trên phần tử **TextView**:

- **Constraint handle:** Để tạo một ràng buộc như minh họa trong hình động ở trên, nhấp vào tay nắm ràng buộc, được hiển thị dưới dạng một hình tròn ở cạnh của một phần tử. Sau đó, kéo tay nắm đó đến một tay nắm ràng buộc khác hoặc đến đường biên của phần tử cha. Một đường gấp khúc sẽ biểu thị ràng buộc được tạo ra.



- **Resizing handle:** Để thay đổi kích thước phần tử, kéo các tay nắm chỉnh kích thước hình vuông. Khi bạn kéo, tay nắm sẽ chuyển thành một góc xiên.

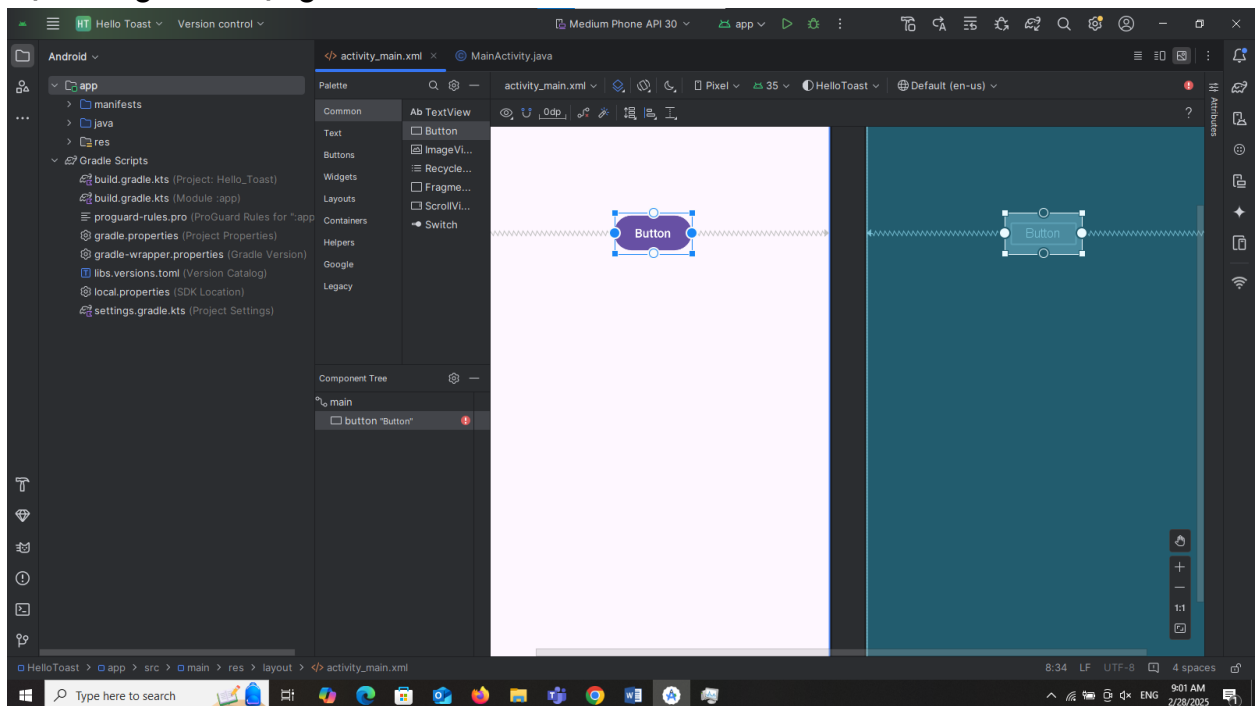


## 2.2 Thêm một Button vào bố cục

Khi được bật, công cụ **Autoconnect** tự động tạo hai hoặc nhiều ràng buộc cho một phần tử giao diện người dùng với bố cục cha. Sau khi bạn kéo phần tử vào bố cục, công cụ này sẽ tạo ràng buộc dựa trên vị trí của phần tử

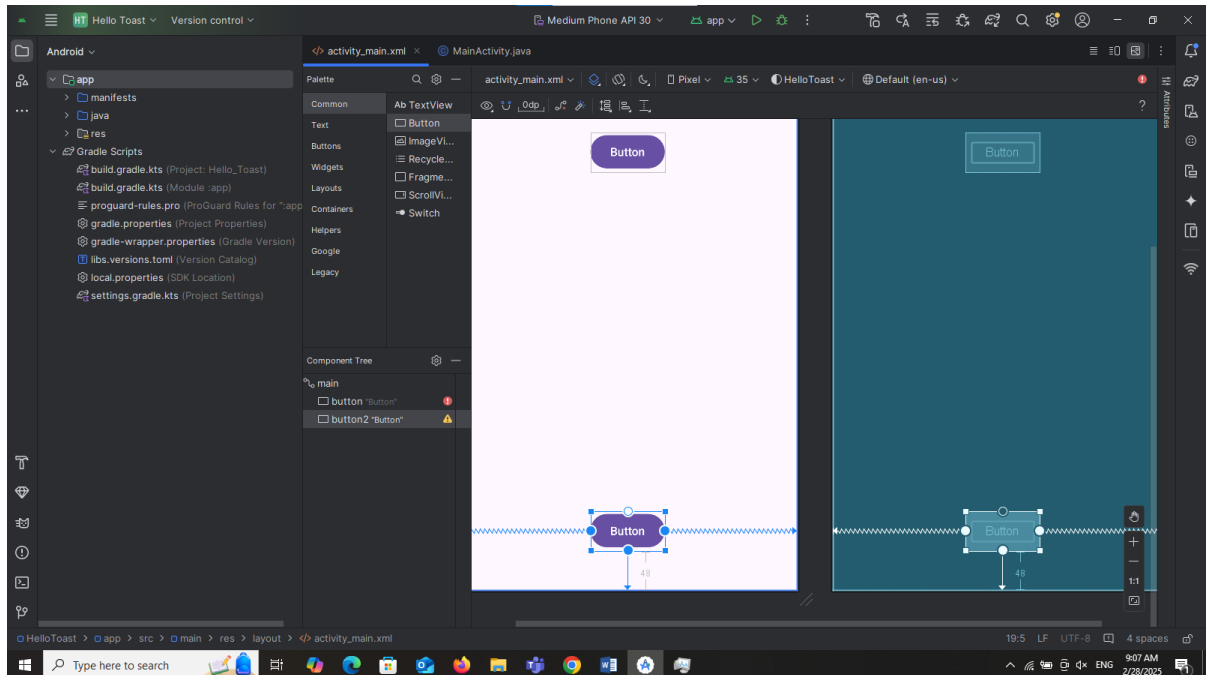
Thực hiện các bước sau để thêm một Button:


1. Bắt đầu với một bố cục trống. Phần tử **TextView** không cần thiết, vì vậy khi nó đang được chọn, hãy nhấn phím **Delete** hoặc chọn **Edit > Delete**. Lúc này, bạn sẽ có một bố cục hoàn toàn trống.
2. Kéo một **Button** từ ngăn **Palette** vào bất kỳ vị trí nào trong bố cục. Nếu bạn thả **Button** vào khu vực chính giữa phía trên của bố cục, các ràng buộc có thể tự động xuất hiện. Nếu không, bạn có thể kéo các ràng buộc để kết nối **Button** với cạnh trên, cạnh trái, và cạnh phải của bố cục như được minh họa trong hình động bên dưới



## 2.3 Thêm Nút thứ hai vào bố cục

1. Kéo một **Button** khác từ ngăn **Palette** vào giữa bố cục, như được minh họa trong hình động bên dưới. Công cụ **Autoconnect** có thể tự động tạo các ràng buộc ngang cho bạn (nếu không, bạn có thể tự kéo các ràng buộc này)
2. Kéo một ràng buộc dọc từ **Button** xuống cuối của bố cục (như minh họa trong hình bên dưới)



Bạn có thể xóa các ràng buộc khỏi một phần tử bằng cách chọn phần tử đó và di chuột qua nó để hiển thị nút **Clear Constraints** . Nhấp vào nút này để xóa tất cả các ràng buộc trên phần tử đã chọn. Để xóa một ràng buộc cụ thể, hãy nhấp vào tay cầm đặt ràng buộc đó. Để xóa tất cả các ràng buộc trong toàn bộ bố cục, nhấp vào công cụ **Clear All Constraints** trên thanh công cụ. Công cụ này rất hữu ích nếu bạn muốn thiết lập lại tất cả các ràng buộc trong bố cục của mình

### Nhiệm vụ 3: Thay đổi thuộc tính của phần tử UI

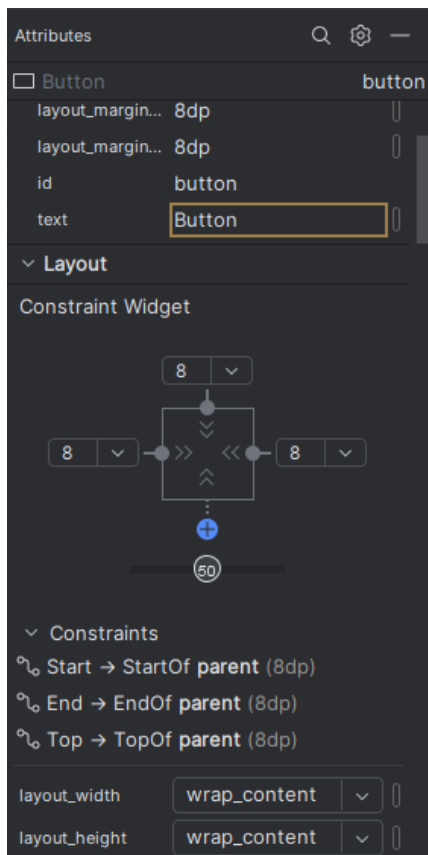
Ngăn **Attributes** cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một phần tử UI. Bạn có thể tìm các thuộc tính (được gọi là thuộc tính *properties*) chung cho tất cả các View trong **View class documentation**

Trong nhiệm vụ này, bạn sẽ nhập các giá trị mới và thay đổi các giá trị cho các thuộc tính quan trọng của Button, có thể áp dụng cho hầu hết các loại View.

#### 3.1 Thay đổi kích thước Button

Layout Editor cung cấp các tay cầm thay đổi kích thước ở cả bốn góc của một View, giúp bạn có thể nhanh chóng thay đổi kích thước View. Bạn có thể kéo các tay cầm ở mỗi góc của View để thay đổi kích thước, nhưng việc này sẽ mã hóa cứng các kích thước chiều rộng và chiều cao. Hạn chế mã hóa cứng kích thước cho hầu hết các phần tử View, vì các kích thước được mã hóa cứng không thể thích ứng với nội dung và kích thước màn hình khác nhau.

Thay vào đó, hãy sử dụng bảng **Attributes** ở phía bên phải của trình chỉnh sửa bố cục để chọn chế độ kích thước không sử dụng các kích thước được mã hóa cứng. Bảng **Attributes** bao gồm một bảng kích thước hình vuông được gọi là **view inspector** ở phía trên. Các biểu tượng bên trong hình vuông đại diện cho các cài đặt chiều cao và chiều rộng như sau:



Trong hình trên:

1. **Height control.** Điều khiển này xác định thuộc tính **layout\_height** và xuất hiện ở hai đoạn trên và dưới của hình vuông. Các góc xiên cho biết rằng điều khiển này được đặt thành **wrap\_content**, nghĩa là View sẽ mở rộng theo chiều dọc khi cần để phù hợp với nội dung của nó. Số "8" chỉ ra một lề chuẩn được đặt là 8dp



2. **Width control.** Điều khiển này xác định thuộc tính **layout\_width** và xuất hiện ở hai đoạn bên trái và phải của hình vuông. Các góc xiên cho biết rằng điều khiển này được đặt thành **wrap\_content**, nghĩa là View sẽ mở rộng theo chiều ngang khi cần để phù hợp với nội dung của nó, tối đa đến một lần là 8dp
3. **Nút đóng Attributes pane.** Nhấp để đóng bảng điều khiển

1.3) Trình chỉnh sửa bố cục

1.4) Văn bản và các chế độ cuộn

1.5) Tài nguyên có sẵn

## Bài 2) Activities

2.1) Activity và Intent

2.2) Vòng đời của Activity và trạng thái

2.3) Intent ngầm định

## Bài 3) Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ

3.1) Trình gỡ lỗi

3.2) Kiểm thử đơn vị

3.3) Thư viện hỗ trợ

## **CHƯƠNG 2. TRẢI NGHIỆM NGƯỜI DÙNG**

### **Bài 1) Tương tác người dùng**

- 1.1) Hình ảnh có thể chọn
- 1.2) Các điều khiển nhập liệu
- 1.3) Menu và bộ chọn
- 1.4) Điều hướng người dùng
- 1.5) RecyclerView

### **Bài 2) Trải nghiệm người dùng thú vị**

- 2.1) Hình vẽ, định kiểu và chủ đề
- 2.2) Thẻ và màu sắc
- 2.3) Bố cục thích ứng

### **Bài 3) Kiểm thử giao diện người dùng**

- 3.1) Espresso cho việc kiểm tra UI

## **CHƯƠNG 3. LÀM VIỆC TRONG NỀN**

### **Bài 1) Các tác vụ nền**

- 1.1) AsyncTask
- 1.2) AsyncTask và AsyncTaskLoader
- 1.3) Broadcast receivers

### **Bài 2) Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền**

- 2.1) Thông báo
- 2.2) Trình quản lý cảnh báo
- 2.3) JobScheduler

## **CHƯƠNG 4. LƯU DỮ LIỆU NGƯỜI DÙNG**

### **Bài 1) Tùy chọn và cài đặt**

**1.1) Shared preferences**

**1.2) Cài đặt ứng dụng**

### **Bài 2) Lưu trữ dữ liệu với Room**

**2.1) Room, LiveData và ViewModel**

**2.2) Room, LiveData và ViewModel**