MTN Business App of the Year

Share your working set up with

#MTNAppAwards #RecodeTheWorld

Weekly Dose of Inspiration

Luke Draper

MTN Business

Mr Luke Draper

MTN Business App of the Year

Founder / CEO, Hero Entertainment

It's go time everywhere you go

#RecodeTheWorld

Mukondleteri Dumela

Founder at Xitsonga.org

Twitter: @Mukondli
Github: Sneidon

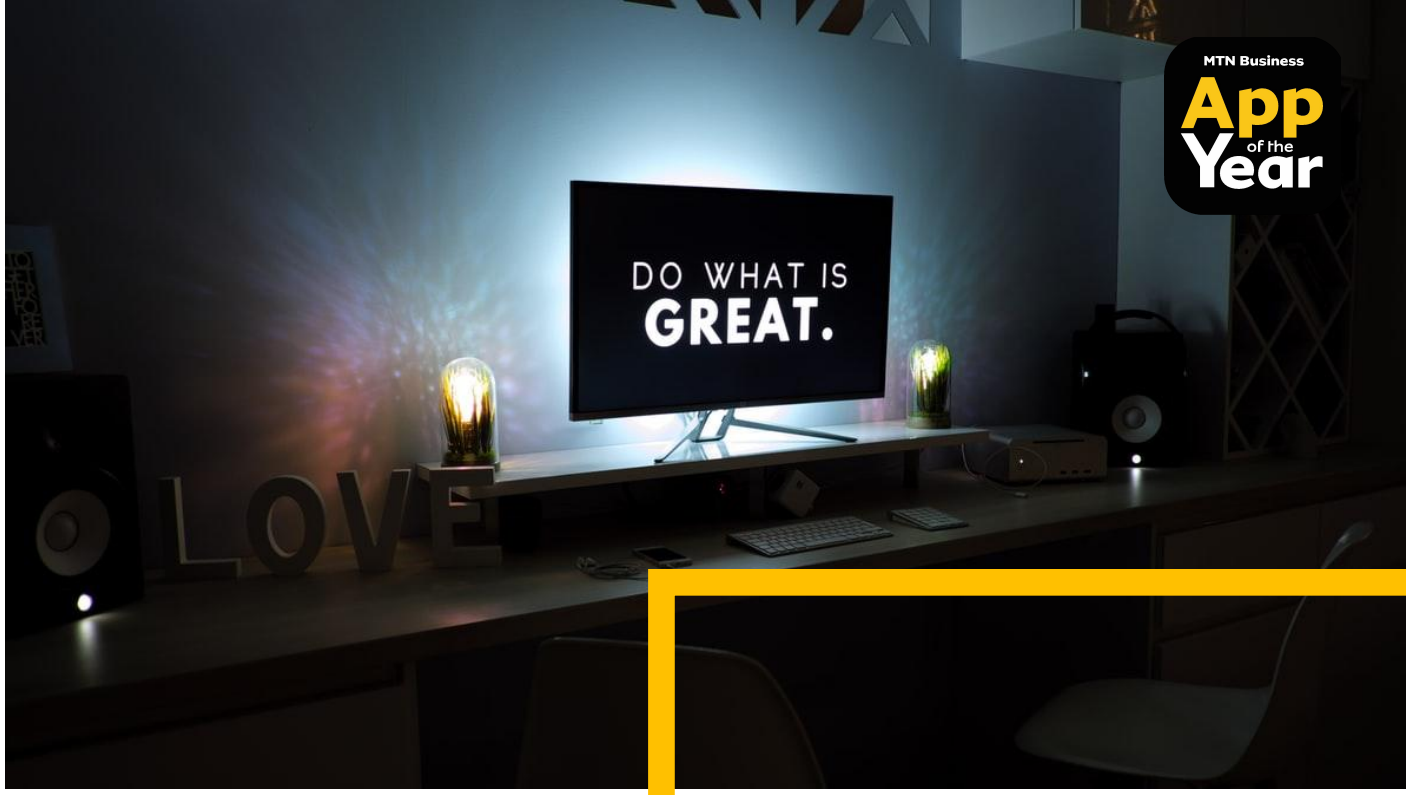
Week 1 Exercise Workout

1. As you develop your app solution, it's important to know exactly who will be using your app solution. Not everyone can be your customer/client.

   When segmenting your customers, you have to be specific... Government has levels - are you targeting a certain Government department; provincial government? Which municipality?

2. Understanding: What is the most value proposition that an app can offer to a user to keep them using or subscribed to the app?

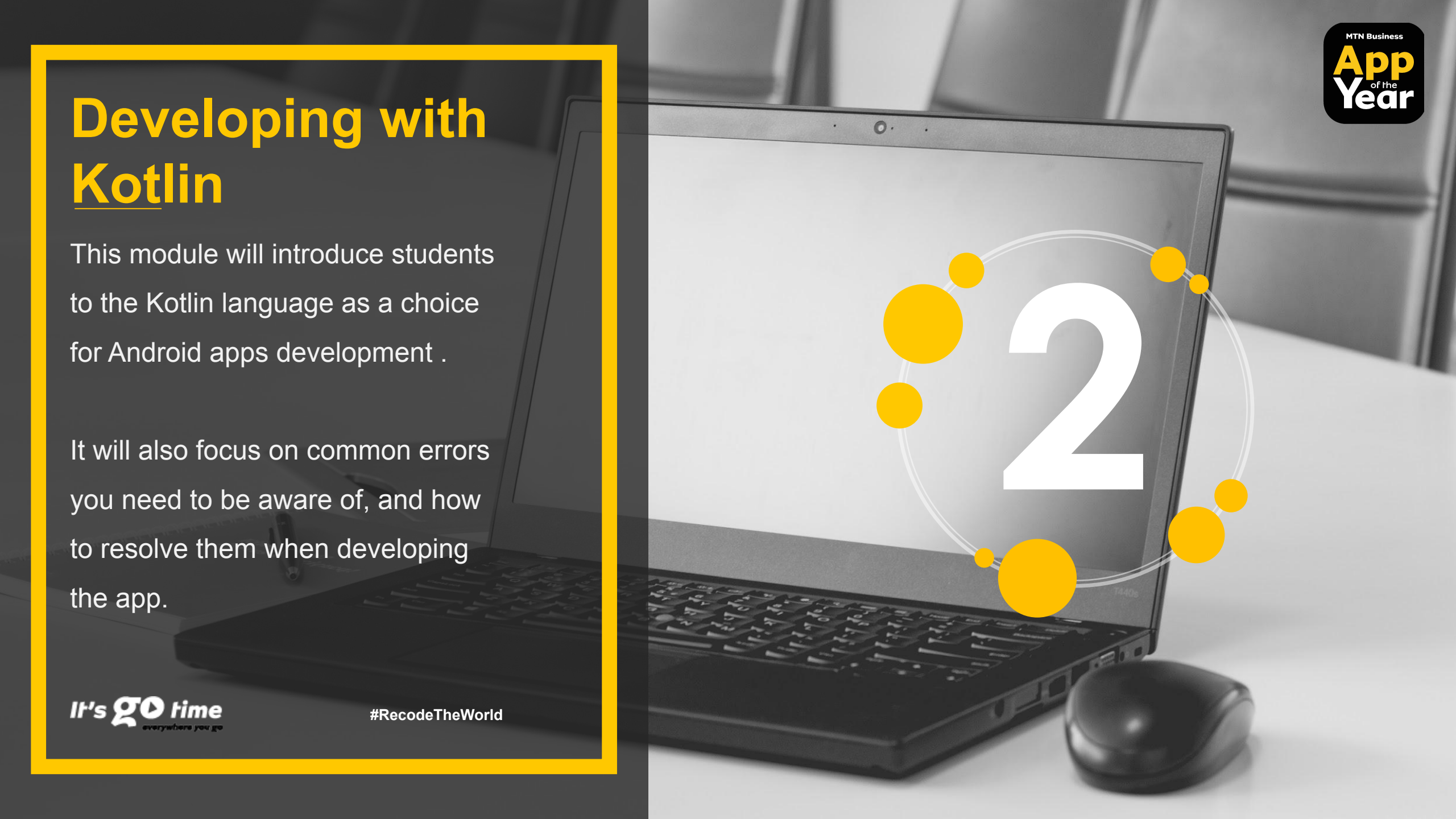
3. Try to estimate and put figures to your costs structure and also, identify key partners that could help your solution succeed.



Developing with Kotlin

This module will introduce students to the Kotlin language as a choice for Android apps development .

It will also focus on common errors you need to be aware of, and how to resolve them when developing the app.



MTN Business App of the Year

It's go time everywhere you go

#RecodeTheWorld

Today's Agenda: Module 1

1. Android overview
2. Development tools
3. Android Studio
4. Your first app
5. Introduction to Kotlin
6. Debugging your app
7. Android app architecture
8. Packaging your first app

MTN Business App of the Year

Android Overview

It's go time everywhere you go

#RecodeTheWorld

Android Overview

What is Android?

Android is an ***open source operating system*** for mobile, embedded and wearable devices.

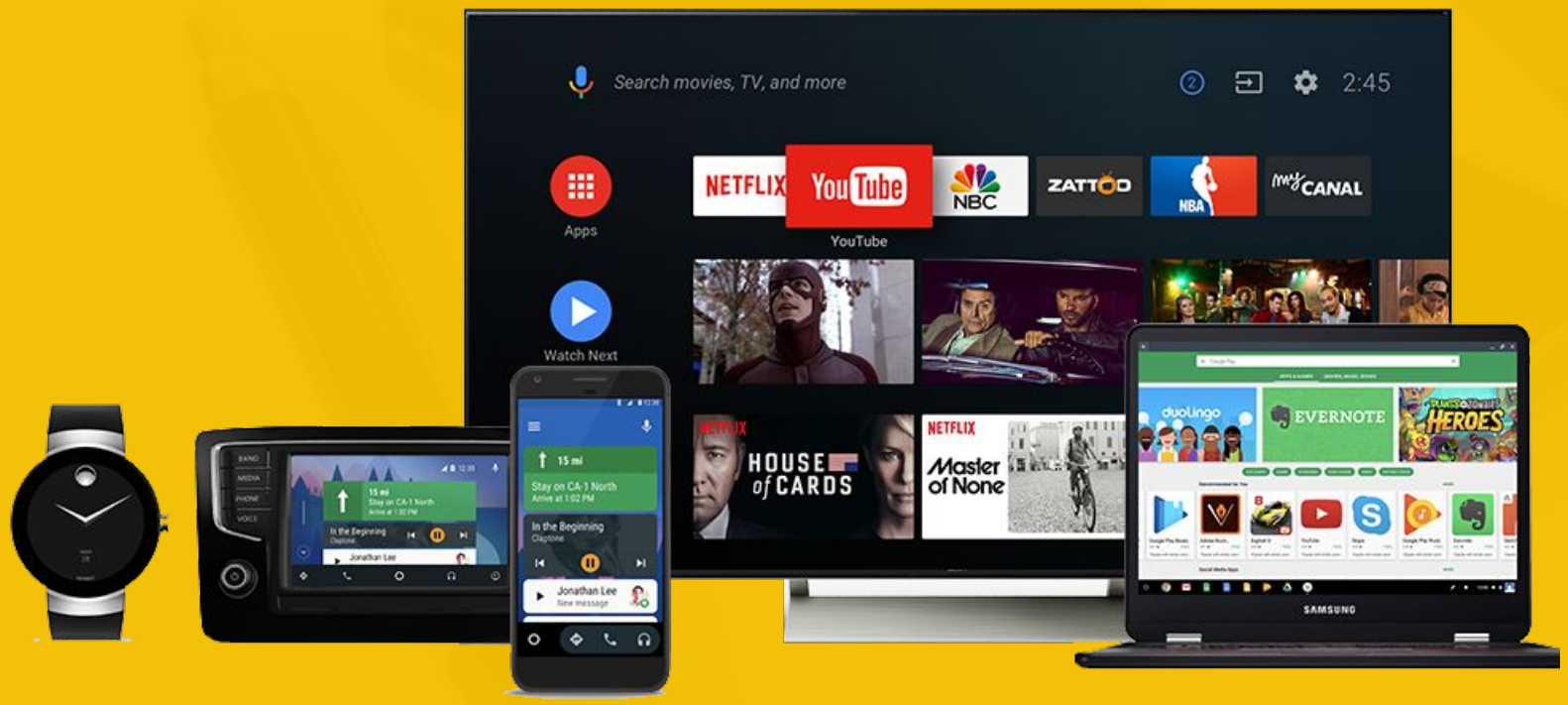
Android was founded in Palo Alto, California in 2003 but first release in September of 2008.

Android maintainers

- Google
- Individuals
- Other companies

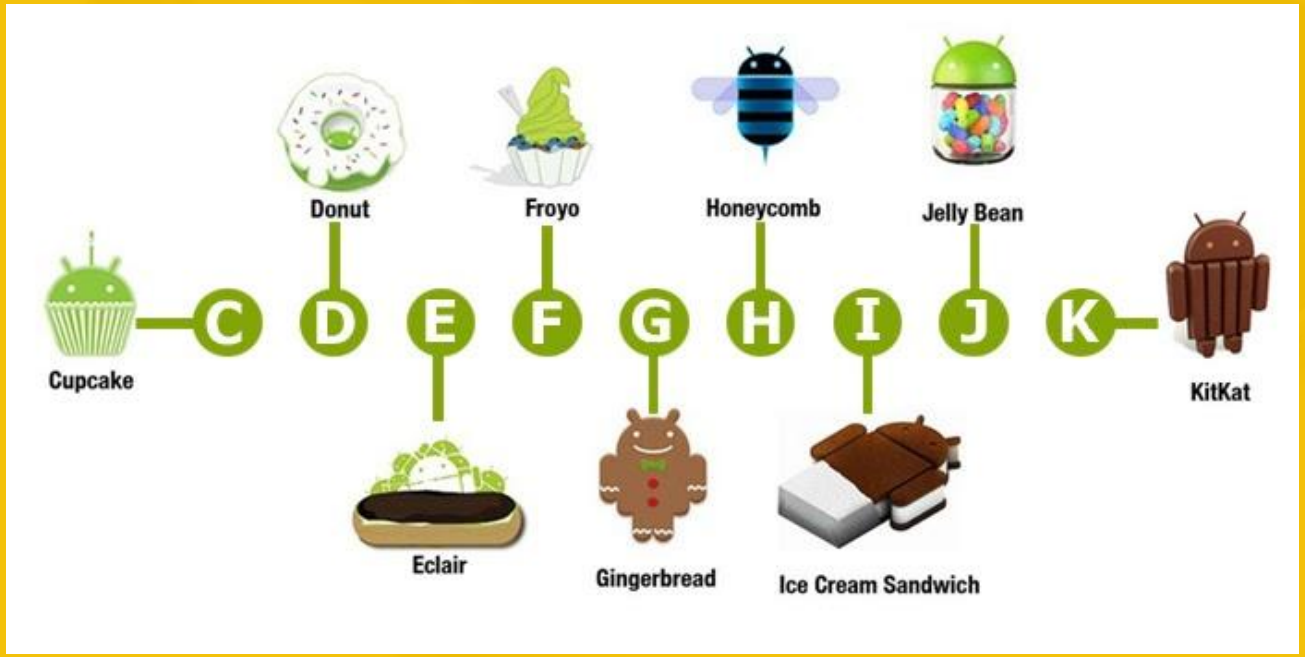
Android Devices

MTN Business App of the Year



Android Versions

- First version was ***Android Alpha*** and most recent version is ***Android 11***.
- Version adaptation: https://tinyurl.com/android-version-distribution.



Manufactures

Market Share

South Africa

Android Overview

Why build for Android?

Android has a majority market share on the mobile phone market.

Android opportunities

- Android Store has a low barrier of entry at ***$25 for a lifetime*** (iOS is $100/y).
- Android is as ***profitable as iOS*** in some categories.
- Android developer community makes it ***easy to get started.***
- Android space has many ***career opportunities*** in SA and abroad.

MTN Business App of the Year

Development tools

Development tools

Native Android apps can be built using ***Java & Kotlin*** on Android Studio.

- ***Native*** Android apps are currently ***portable to HarmonyOS*** (Huawei).
- ***Native*** Android apps are ***not portable to iOS*** which means you will need to build for iOS separately.

Building using cross platform

- Cross platform development enables you to build for iOS, Android and other platforms ***using one source code***.
- Examples are React Native, Flutter, Unity3D, Codova and more.

MTN Business App of the Year

… so **to native** or **not to native?**

It's go time everywhere you go

#RecodeTheWorld

Development tools

MTN Business App of the Year

MTN Business App of the Year

Android Studio

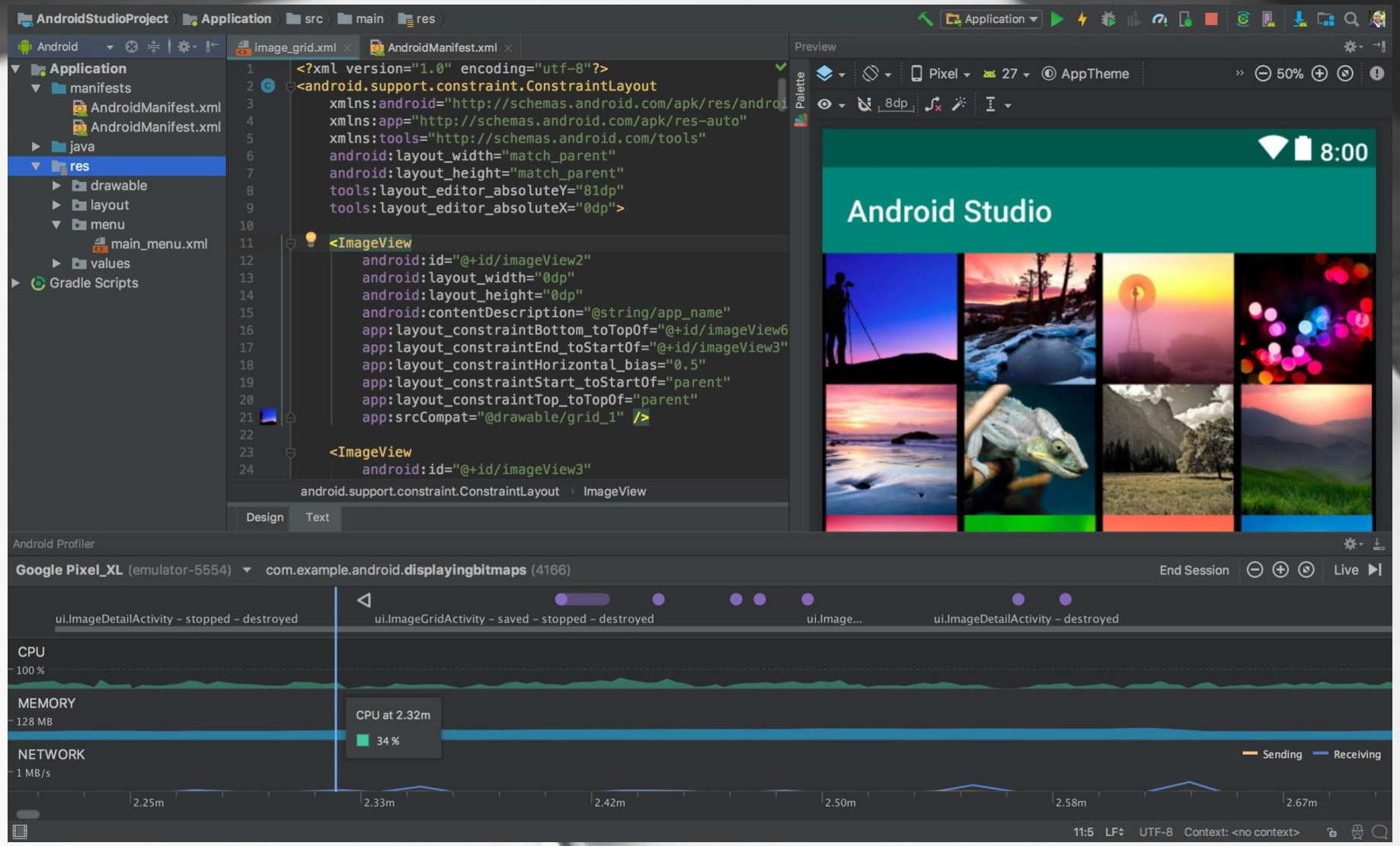
Android Studio

Android Studio provides the fastest tools for building apps on every type of Android device.

Android Studio offers you:

- A flexible Gradle-based build system.
- A fast and feature-rich emulator.
- A unified environment where you can develop for ***all Android devices***.
- Extensive testing tools and frameworks.
- C++ and NDK support.

Android Studio



MTN Business App of the Year

Your first app

It's go time everywhere you go

#RecodeTheWorld

Your first app

We are going to create a "Hello World" app using Android Studio.

Prerequisites

- Java JDK
- Android Studio

Your first app

Android Virtual Device

Android Virtual Device (***AVD***) is a configuration that defines the characteristics of an Android device you want to simulate in the Android Emulator.

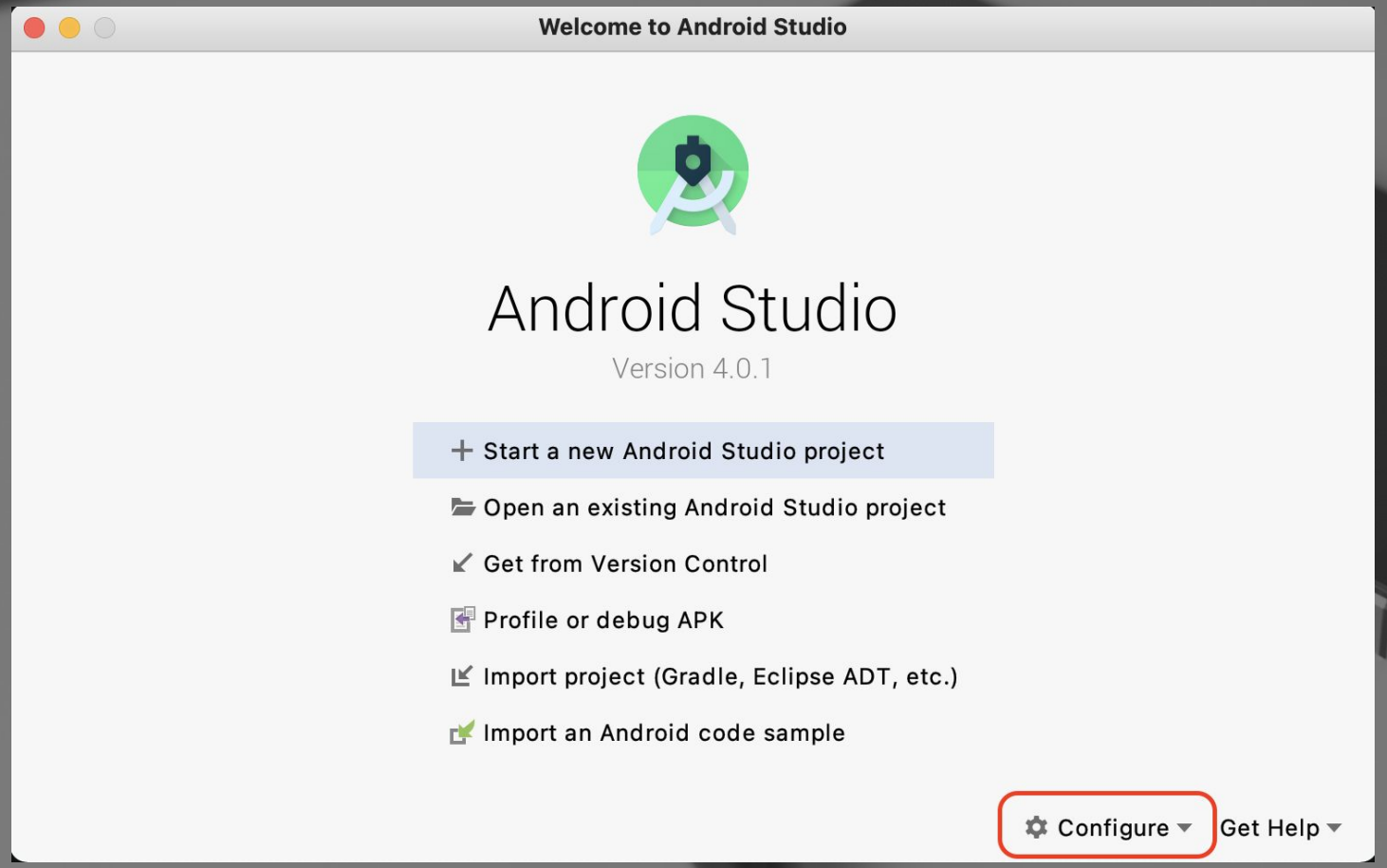
Steps to create a new Android project

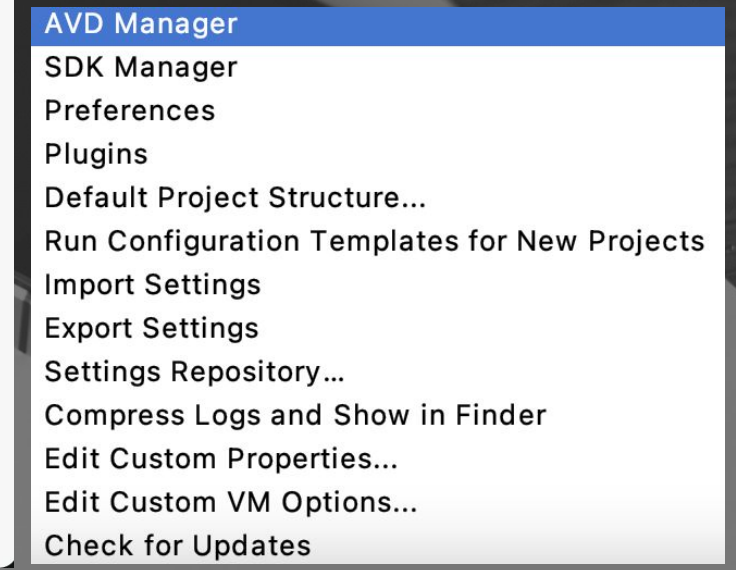
1. Open Android Studio.
2. Click “Configure” and then “AVD Manager”.
3. Click “Create Virtual Device”.
4. Choose a “Phone device from the list”.
5. Choose a “System image from the list”.
6. Click “Next” and then “Finish”.

Let Android Studio work its magic.

Your first app

(2) Android Virtual Device



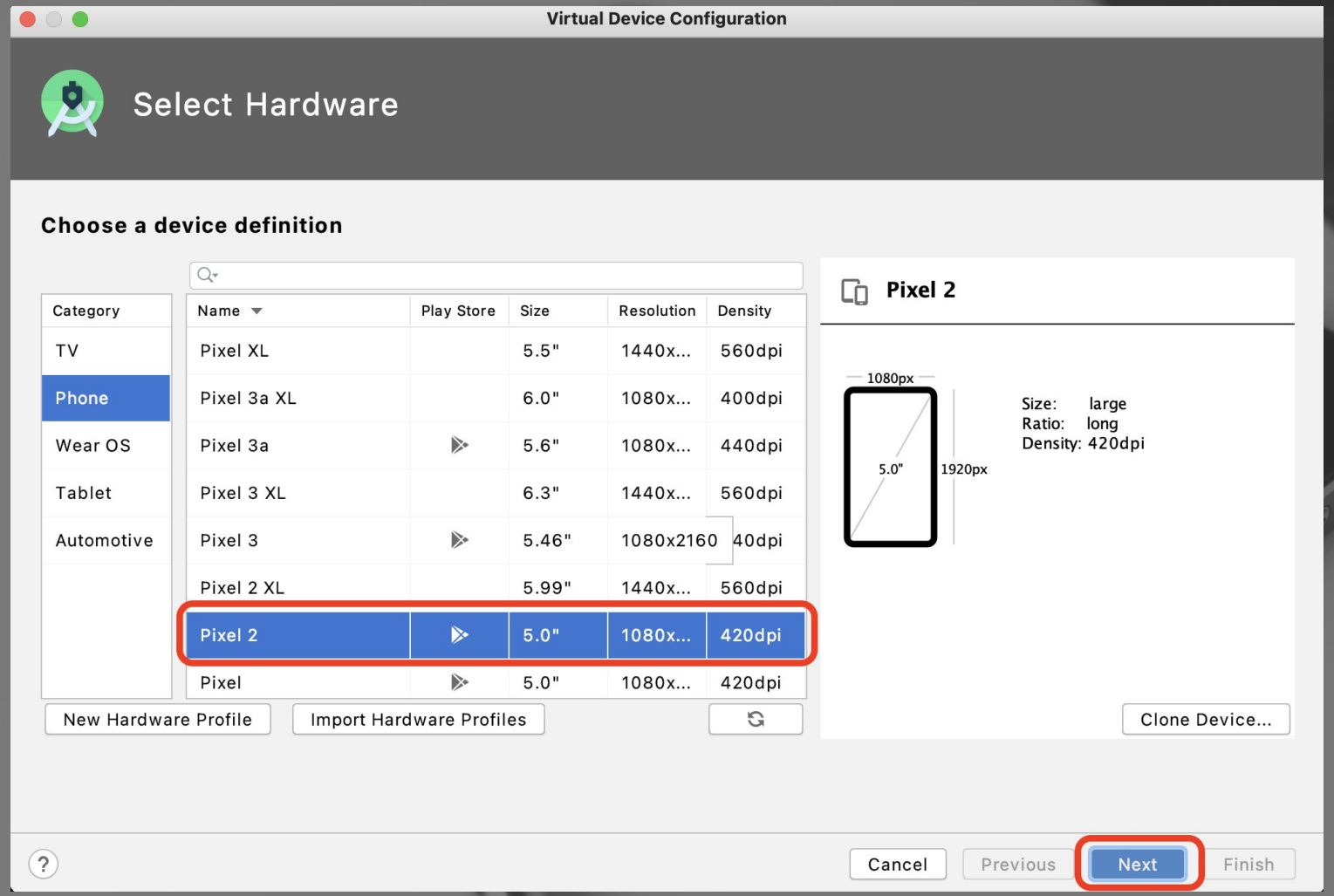


Your first app

(3) Android Virtual Device

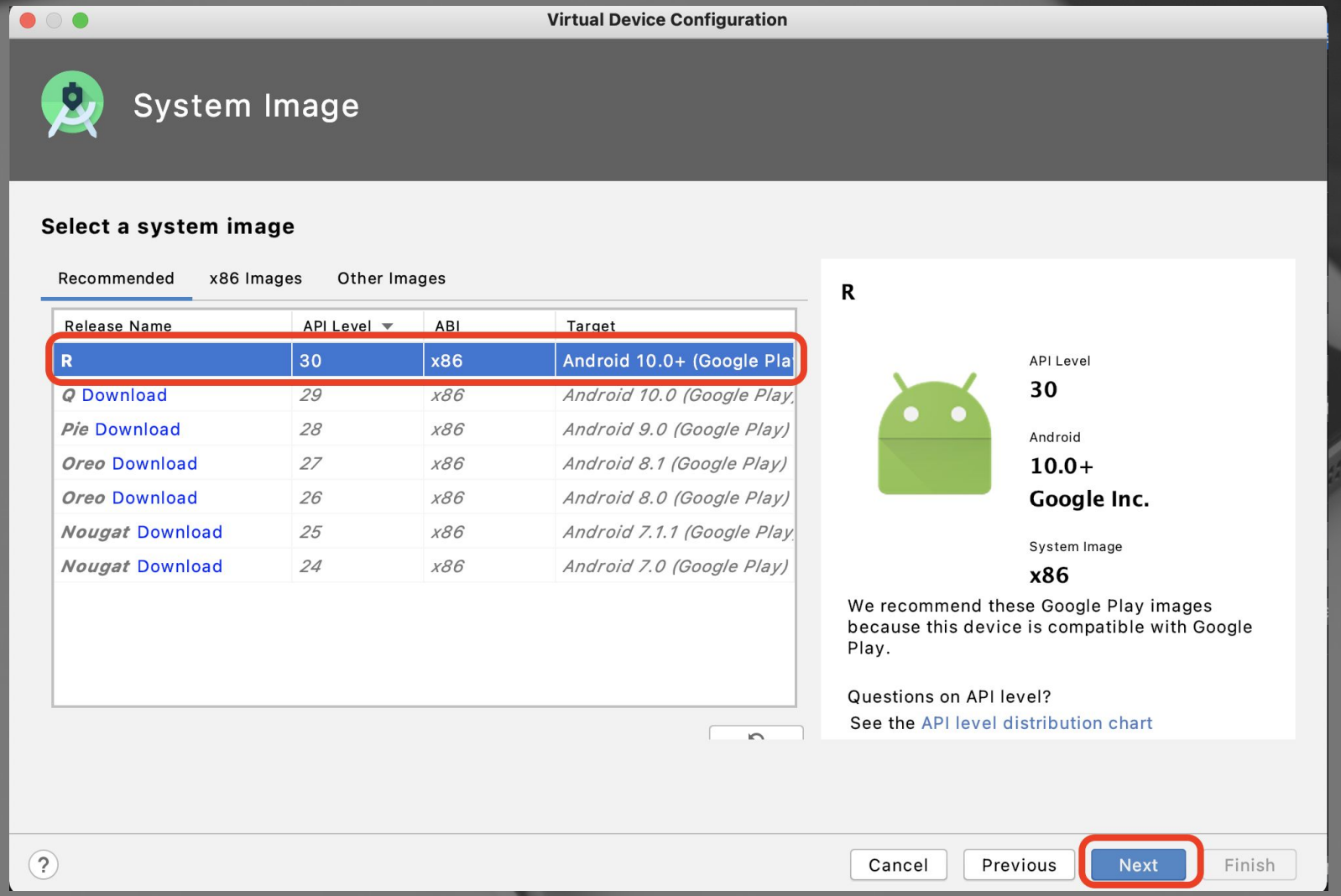
Your first app

(4) Android Virtual Device



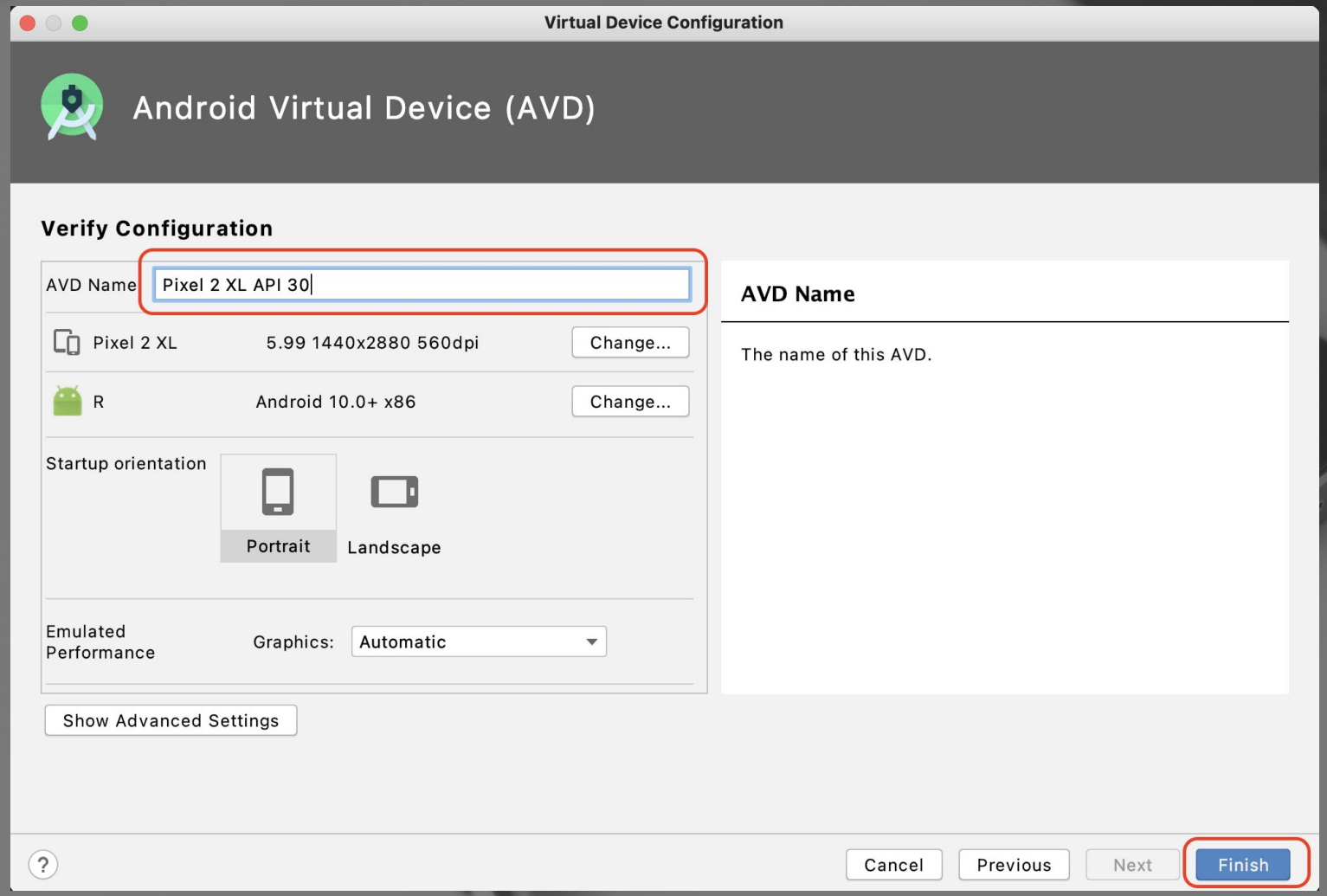
Your first app

(5) Android Virtual Device



Your first app

(6) Android Virtual Device



Your first app

Android Virtual Device

Android Emulator can take a little while to boot.

A few recommendations

1. Keep your cool
2. Test using an actual Android device
3. Choose a resolution lower than your PC.
4. https://tinyurl.com/make-emulator-run-faster

Your first app

Android Studio project

We are going to create a "Hello World" app using Android Studio.

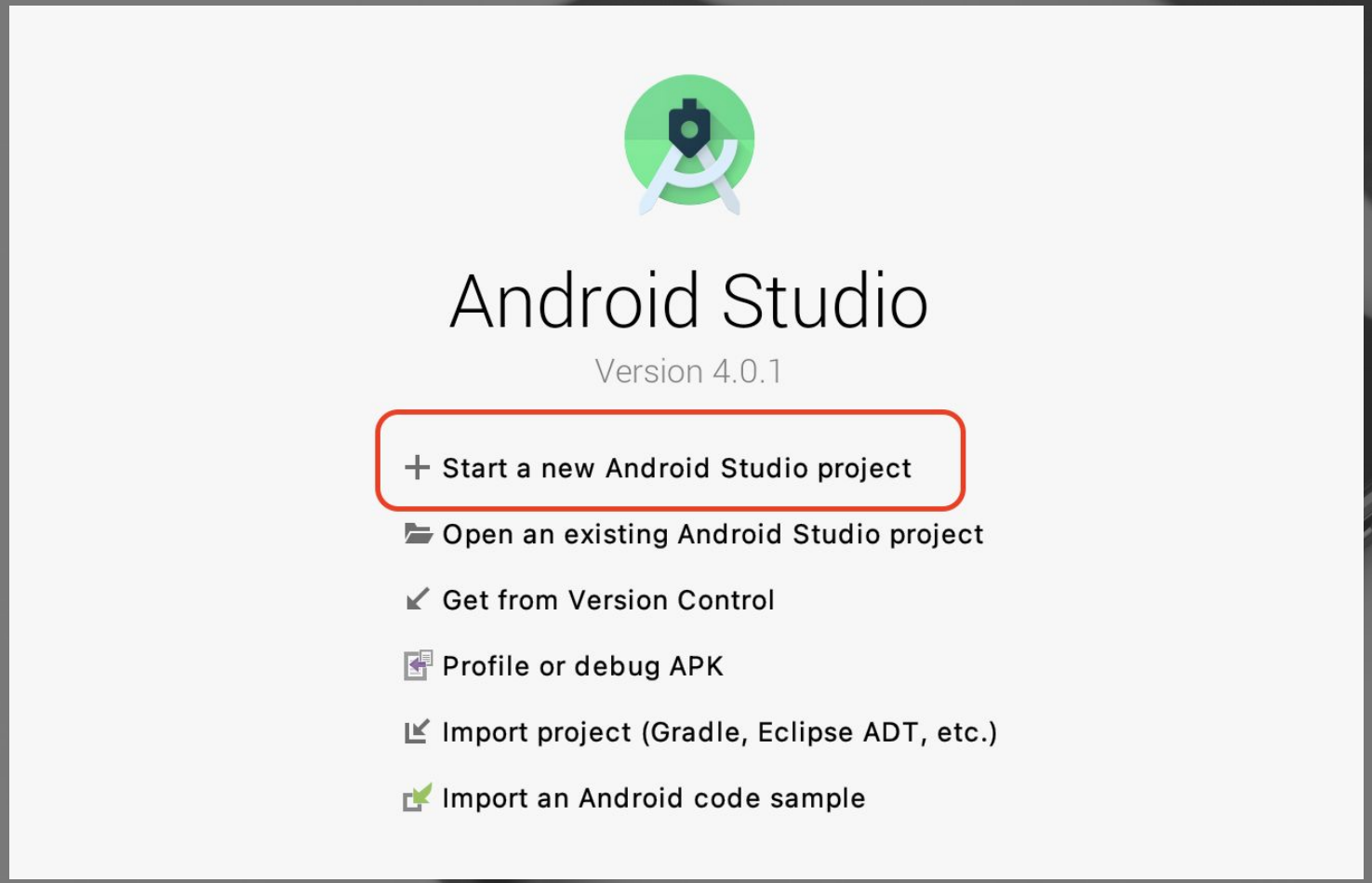
Steps to create a new Android project

1. Open Android Studio.
2. Click “Start a new Android Studio project”.
3. Click “Empty Activity” and then “Next”.
4. Configure your project and click “Next”.
5. Application name and package (Must be unique).
6. Language (Kotlin) and Minimum SDK (23).

Let Android Studio work its magic.

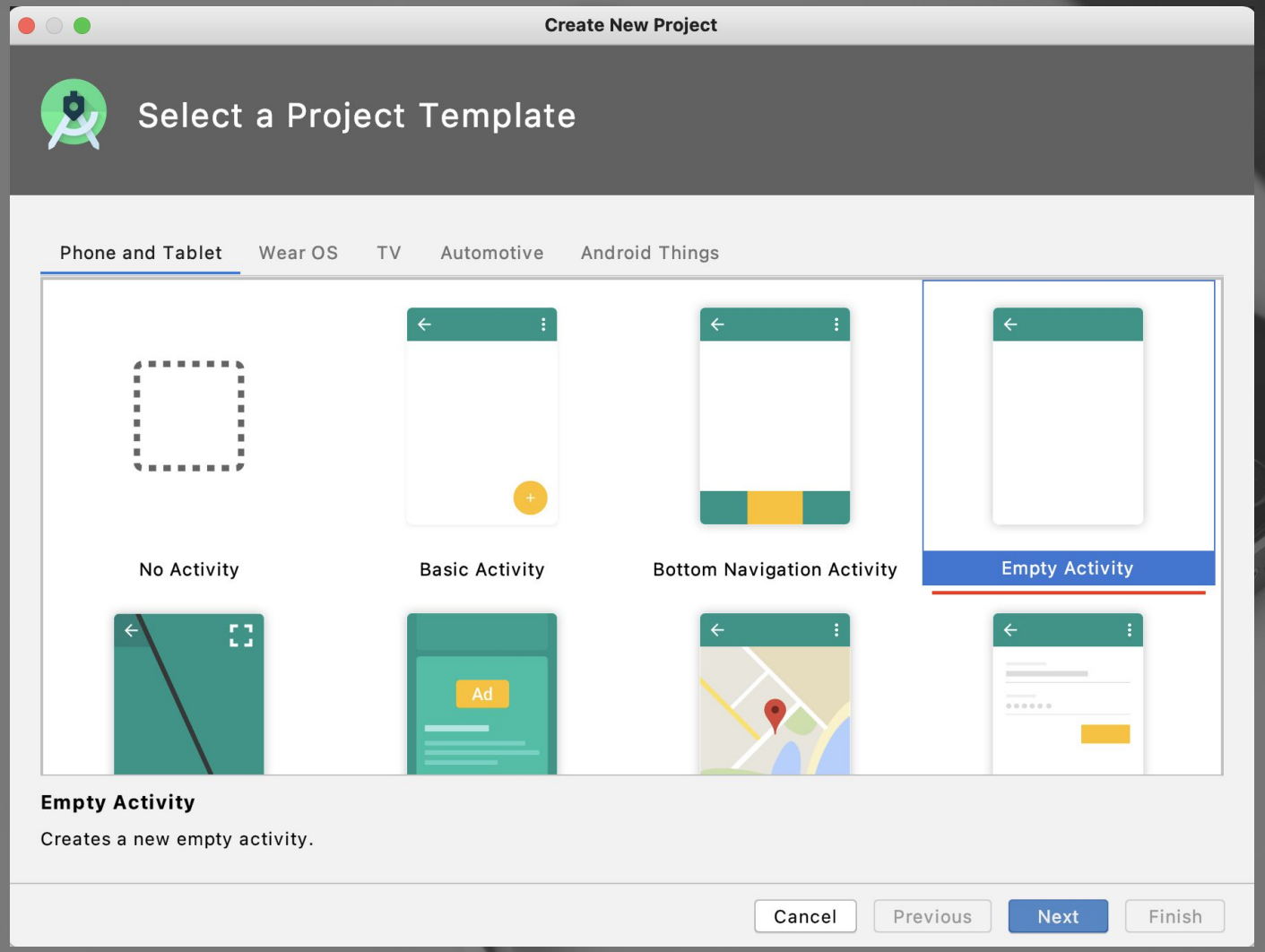
Your first app

(2) Android Studio project



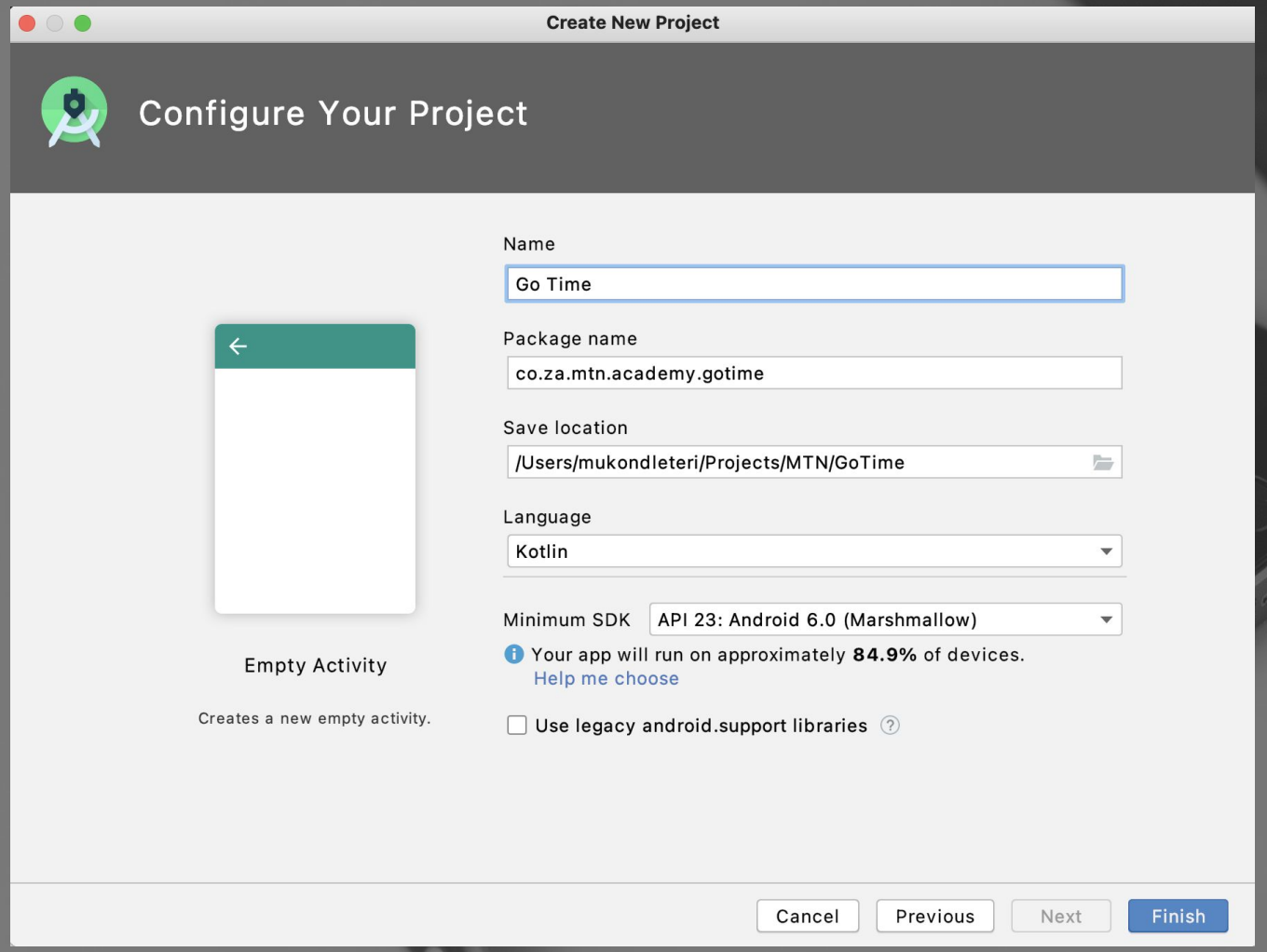
Your first app

(3) Android Studio project



Your first app

(4, 5, 6) Android Studio project



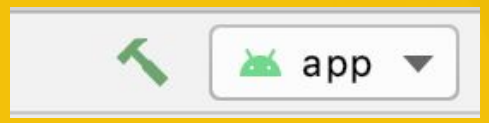
Your first app

Run your app

Now we are going to run the app on an emulated device without any code changes.

Steps to run your app

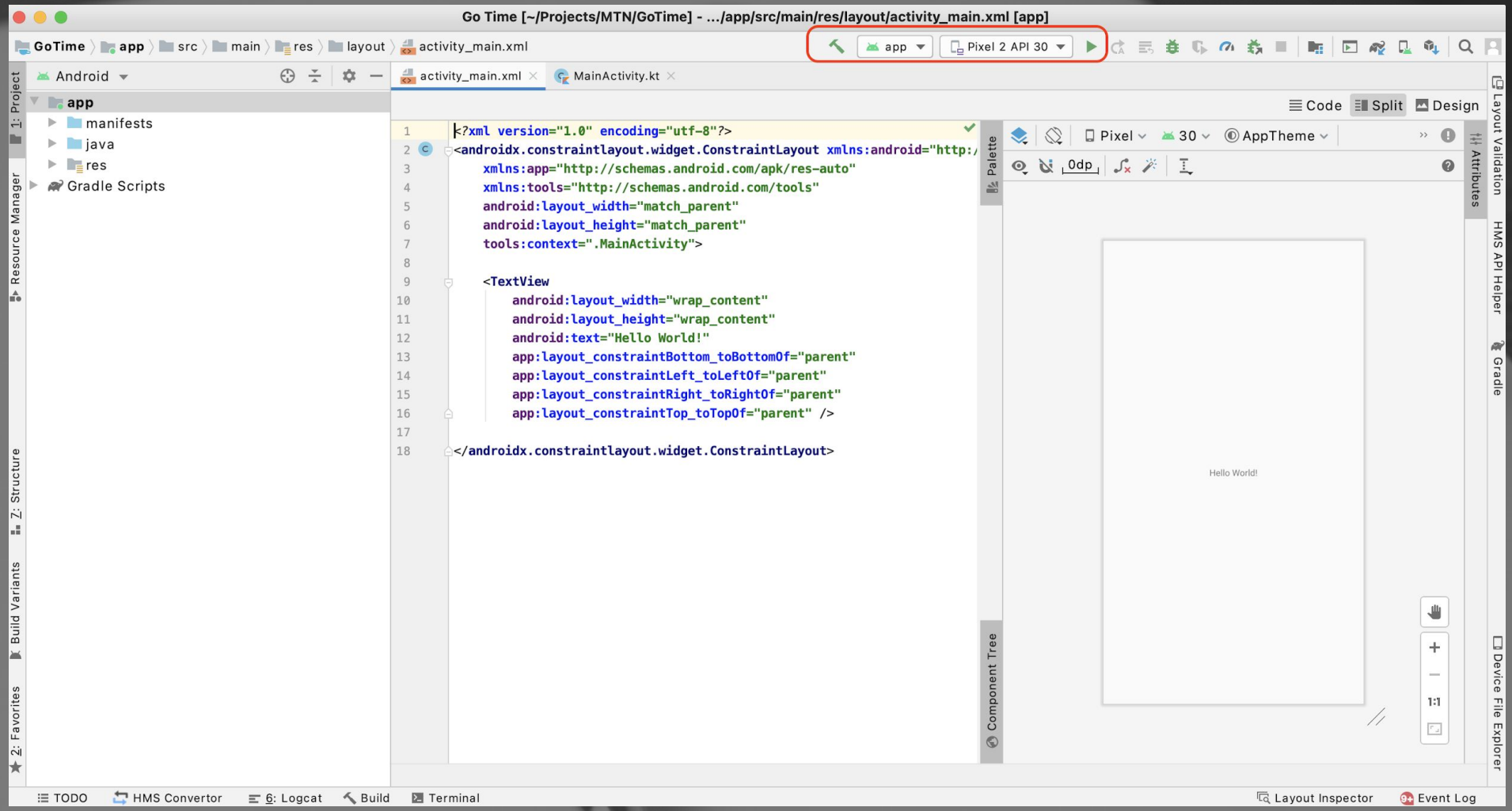
1. Make sure Gradle sync and other processes are done running and were successful.



2. Select your device from list and click “Run” (Green play button)

Your first app

Run your app



Your first app

Run your app

Android Studio has a few things to do before you can run your app.

Things to note

1. Emulator takes a little while to run and often freezes on low spec PCs.
2. Android app can also be run on a live device by connecting a USB cable.

*Demonstration of how to **work with an actual phone.***

COFFEE BREAK

Quick 2 Minutes Break, then...

1. Meet Kotlin
2. Kotlin variables, functions, conditionals and classes.
3. Interoperability

The Journey

Best Campus Cup Solution: This is for students that are enrolled in an institution of higher learning – you can submit an app which was built before App Academy. This can be an existing app.

Best Hackathon Solution: This has to be a solution that was built at the 72 hour hackathon and must meet the criteria of the hackathon (MVP, pitch deck etc).

Campus Cup Challenge

Best Hack Solution

MTN Business App of the Year

Meet Kotlin

It's go time everywhere you go

#RecodeTheWorld

Kotlin

Introduction

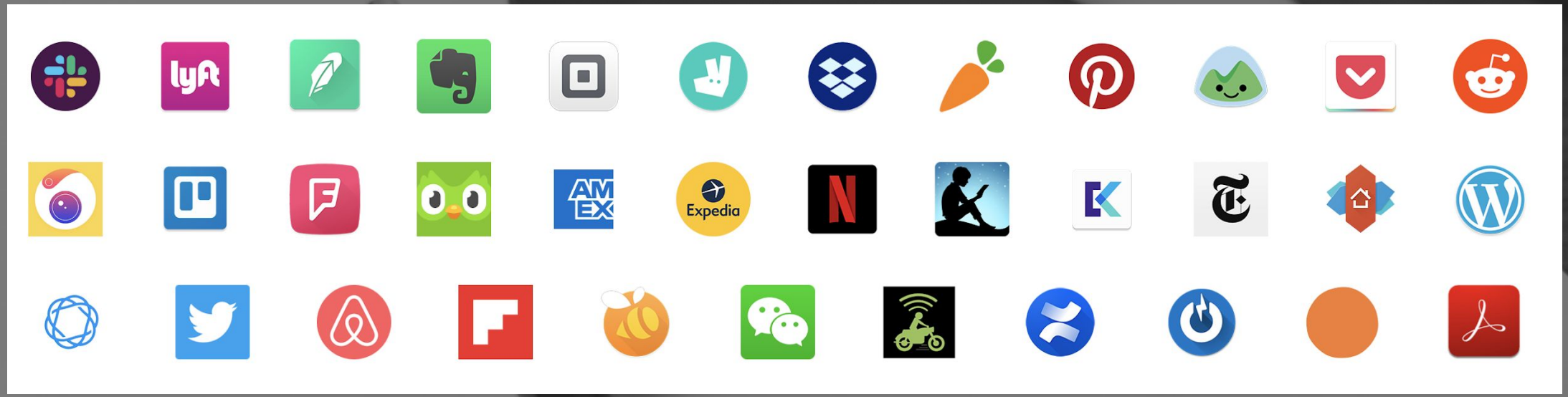
Kotlin is a ***cross-platform***, statically typed, general-purpose programming language with type inference.

A few things to note

1. This module focuses on Kotlin for Android.
2. Kotlin standard library depends on the Java Class Library.
3. Kotlin is used by 60% of professional Android developers.

Kotlin

Introduction



Kotlin

Introduction

Key features

1. **Expressive and concise**
Kotlin's modern language features allow you to focus on expressing your ideas and write less boilerplate code.

2. **Safer code**
Kotlin helps you avoid NullPointerExceptions.

3. **Interoperable**
Kotlin is 100% interoperable with Java.

4. **Structured concurrency**
Kotlin coroutines streamline asynchronous programming.

Kotlin

Basics

1. **Variable declaration**
   Keywords val and var, Type inference, Null safety.

2. **Conditionals**
   If-else statements, when statement, smart casting

3. **Functions**
   Declarations, Anonymous functions, Higher-order functions

4. **Classes**
   Properties, Class functions and encapsulation,

5. **Interoperability**
   Example demo of interoperability with Java

Kotlin

Variable declaring

Kotlin uses two different keywords to declare variables: ***val*** and ***var***.

1. Use ***val*** for a variable whose ***value never change***.

```
val networkProviderFullName = "MTN"
```

```
val networkProviderShortName: String = "MTN"
```

2. Use ***var*** for a variable whose ***value can change***

```
var bundleBalance = 1000

var secondBundleBalance: Int = 2000
```

Kotlin

Variable declaring

Type inference is when the Kotlin compiler infers the type based off of the type of the assigned value.

```
var customerUniqueReference = "Mukondleteri"

// Fails to compile
customerUniqueReference.inc()
```

```
var customerUniqueReference = 1200

// Complies
customerUniqueReference.inc()
```

Kotlin

Variable declaring

MTN Business App of the Year

Null safety - Kotlin variables can't hold null values by default. For a variable to hold a null value, it must be of a *nullable* type.

```
// Fails to compile
val signalStrength: String = null
```

```
// Compiles
val signalStrength: String? = null
```

Kotlin

Conditionals

Conditional expressions - Kotlin offers conditional expressions to simplify stateful logic by removing the need for repletion.

```
if (bundleBalance == 0) {
    println("You have no data remaining.")
} else if (bundleBalance < 1000) {
    println("You have less than 1GB of data remaining.")
} else {
    println("You are set for the next week.")
}
```

Kotlin

Conditionals


```
val messageToShowCustomer: String = if (bundleBalance == 0) {
    "You have no data remaining."
} else if (bundleBalance < 1000) {
    "You have less than 1GB of data remaining."
} else {
    "You are set for the next week."
}

print(messageToShowCustomer)
```

Kotlin

Conditionals

Kotlin uses **when statements** to simplify conditionals.

```
val messageToShowCustomer: String = when {
    bundleBalance == 0 -> "You have no data remaining."
    bundleBalance < 1000 -> "You have less than 1GB of data remaining."
    else -> "You are set for the next week."
}

print(messageToShowCustomer)
```

Kotlin

Functions

You can group one or more expressions into a function to avoid repeating the same series of expressions each time that you need a result.

```
fun getMessageToShowCustomer (): String {

    return "You are an awesome customer."
}
```

```
val message = getMessageToShowCustomer()

print(message)
```

Kotlin

Functions


```
fun getMessageToShowCustomer (bundleBalance: Int): String {
    val message: String = when {
        bundleBalance == 0 -> "You have no data remaining."
        bundleBalance < 1000 -> "You have less than 1GB of data remaining."
        else -> "You are set for the next week."
    }

    return message
}
```

Kotlin

Classes

Classes enable you to define your custom types.

1. How to ***declare*** a class

```
class Customer
```

2. How to add ***class properties***

```
class Customer {
    var firstName: String? = null
    var lastName: String? = null
    var age: Int? = null
    var dateOfBirth: Date? = null
}
```

Kotlin

Classes

1. How to add ***a custom constructor***

```
class Customer (
    var firstName: String? = null,
    var lastName: String? = null,
    var age: Int? = null,
    var dateOfBirth: Date? = null) {
}

var customer  = Customer( firstName: "Mukondleteri", lastName: "Dumela", age: 70, Date())
```

Kotlin

Interoperability

Demonstration

COFFEE BREAK

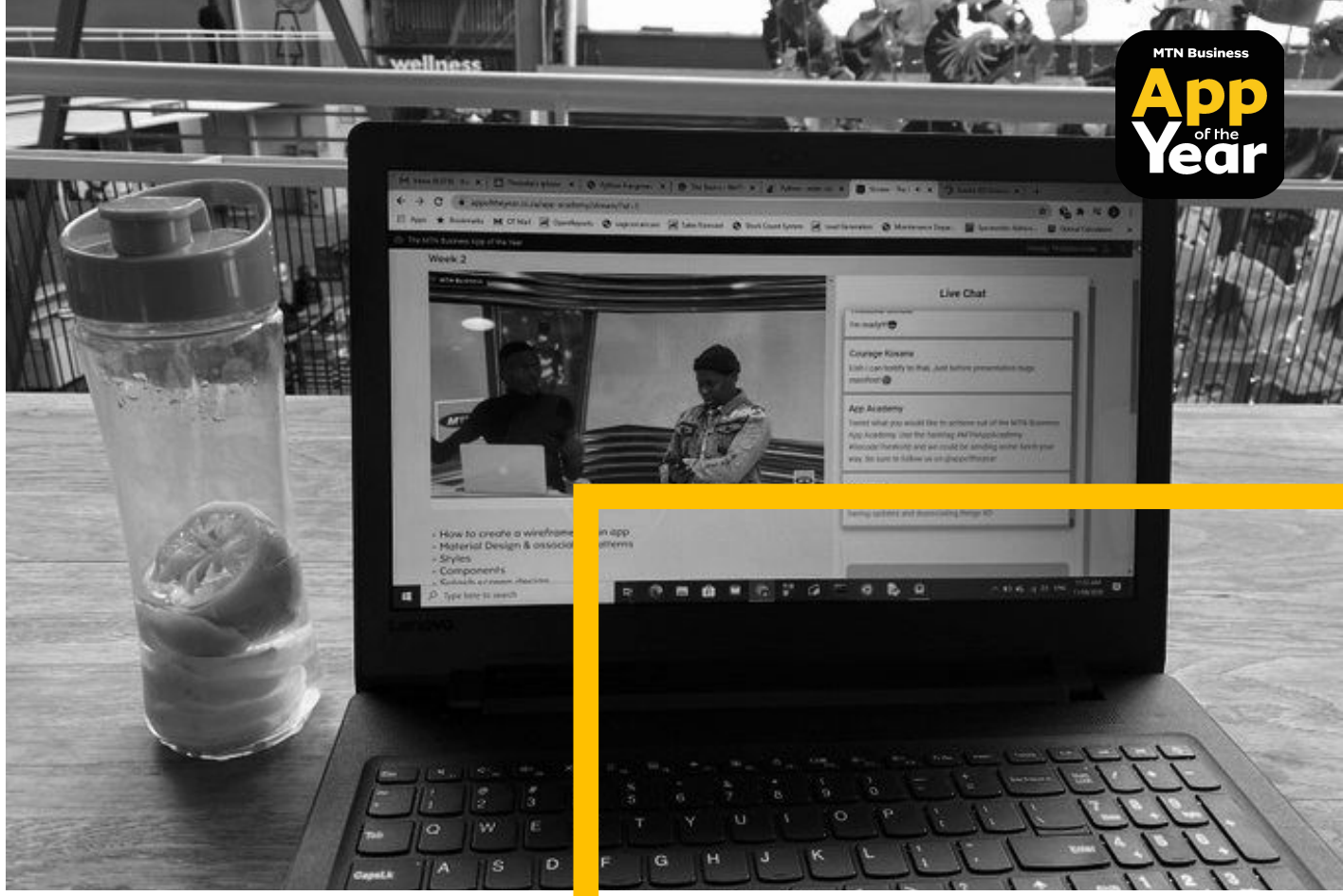
Quick 2 Minutes Break, then...

1. Debugging your app
2. Android architecture
3. Navigating to an activity

Answering Questions

Drop your questions on the

Chat, Slack or the Socials



MTN Business App of the Year

Debugging

It's go time everywhere you go

#RecodeTheWorld

Debugging

Basics

Debugging is the process of identifying and removing errors from computer hardware or software.

Android Studio debugging

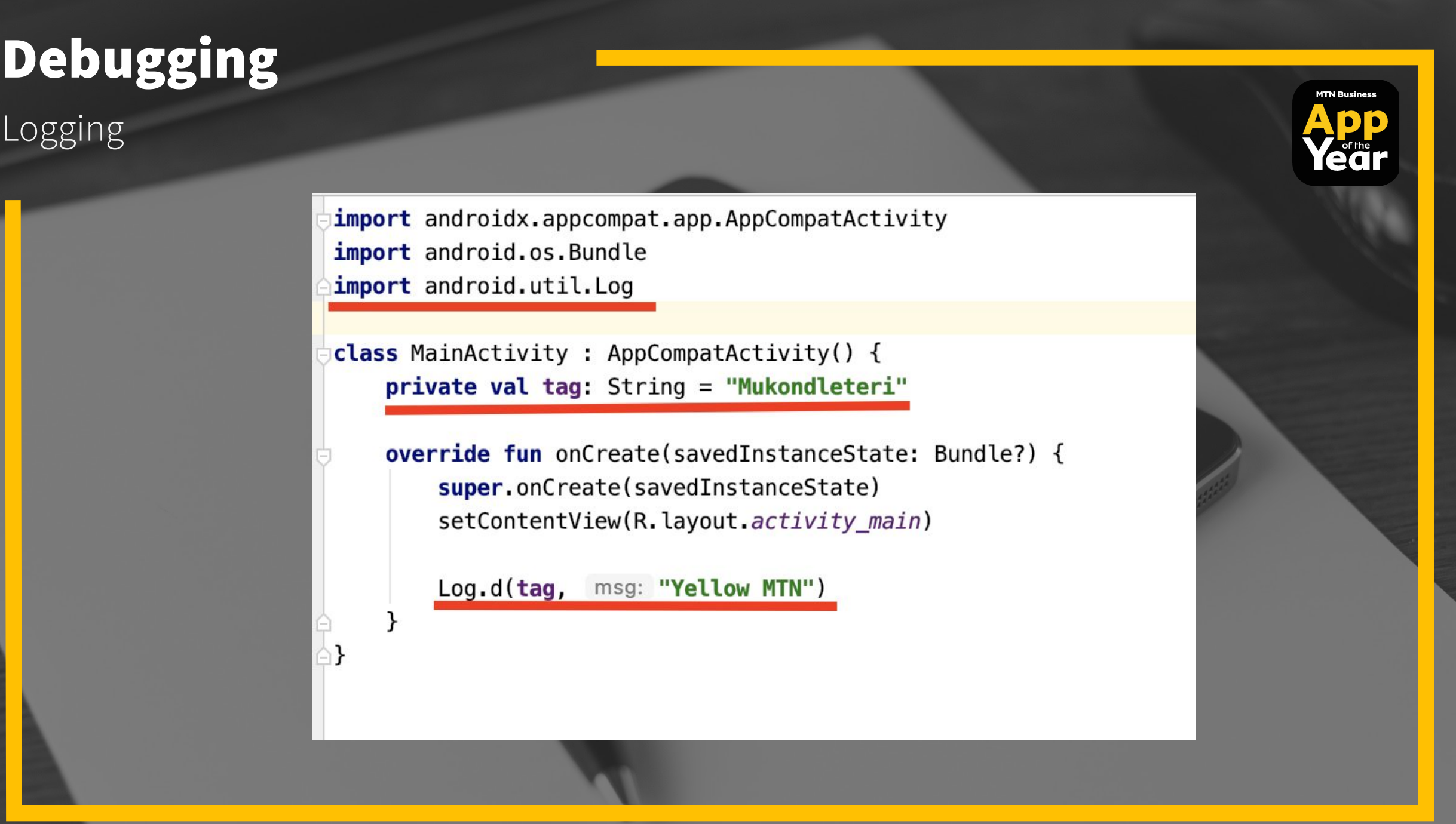
1. **Logging, LogCat**
   - Use android.util.Log to log messages.
   - Logcat displays log/logged messages.
   - Logcat also display a full ***stacktrace*** when an error occurs.

2. **Android Device Monitor**

   Displays useful information such as usage of memory and CPU and network calls.

Debugging

Logging

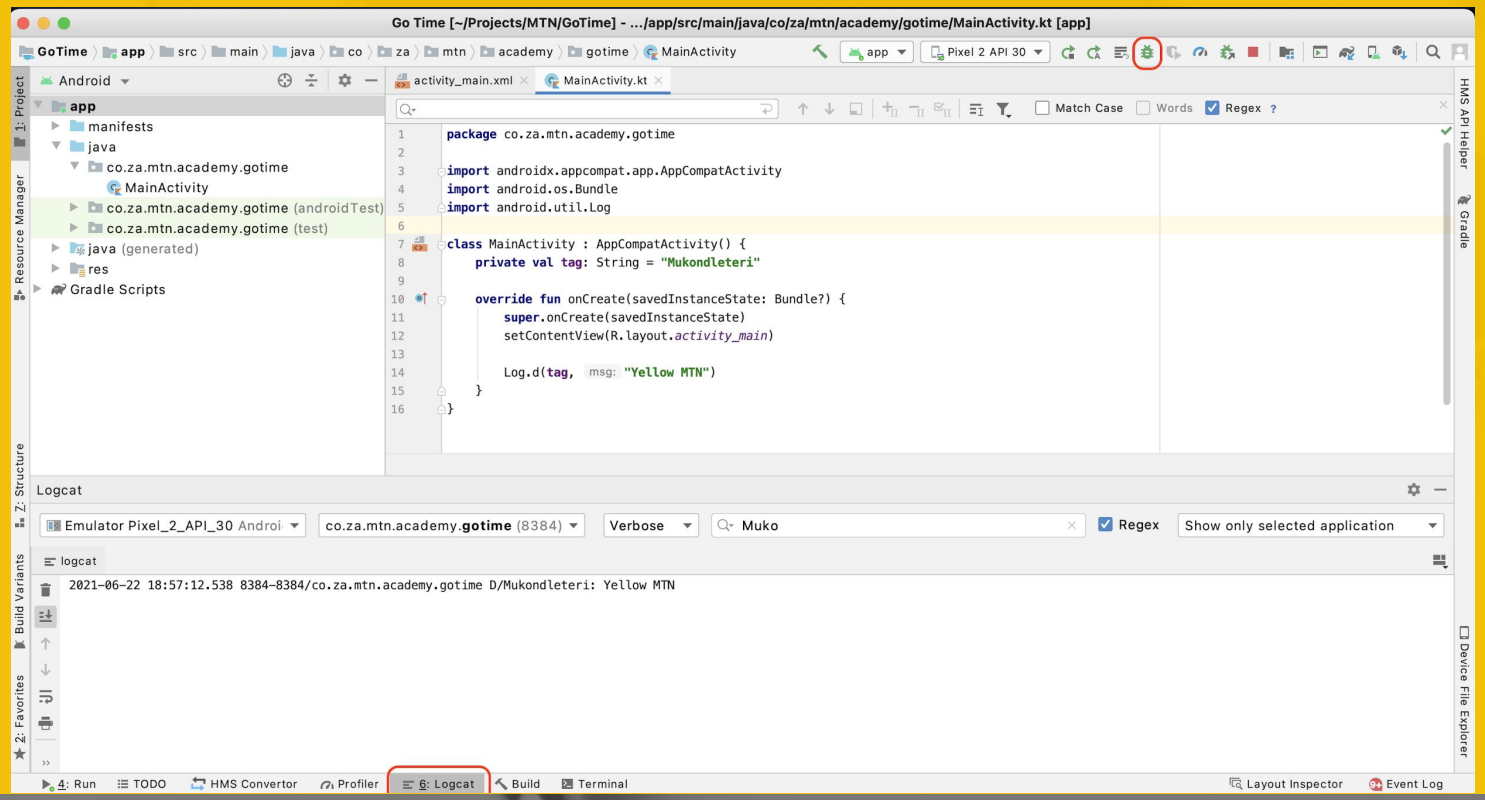



```
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log

class MainActivity : AppCompatActivity() {
    private val tag: String = "Mukondleteri"

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

        Log.d(tag,  msg: "Yellow MTN")
    }
}
```

Debugging

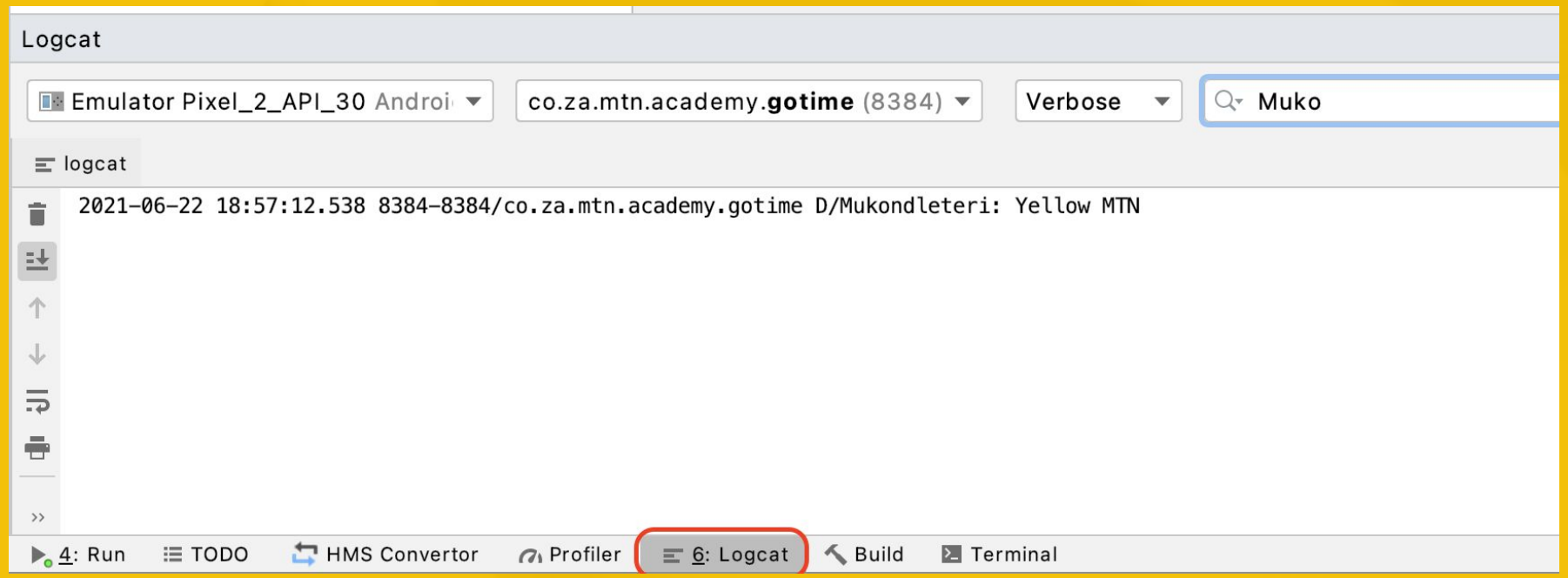
Logcat

Run your app in **debug mode** but even running with play can work.



Debugging

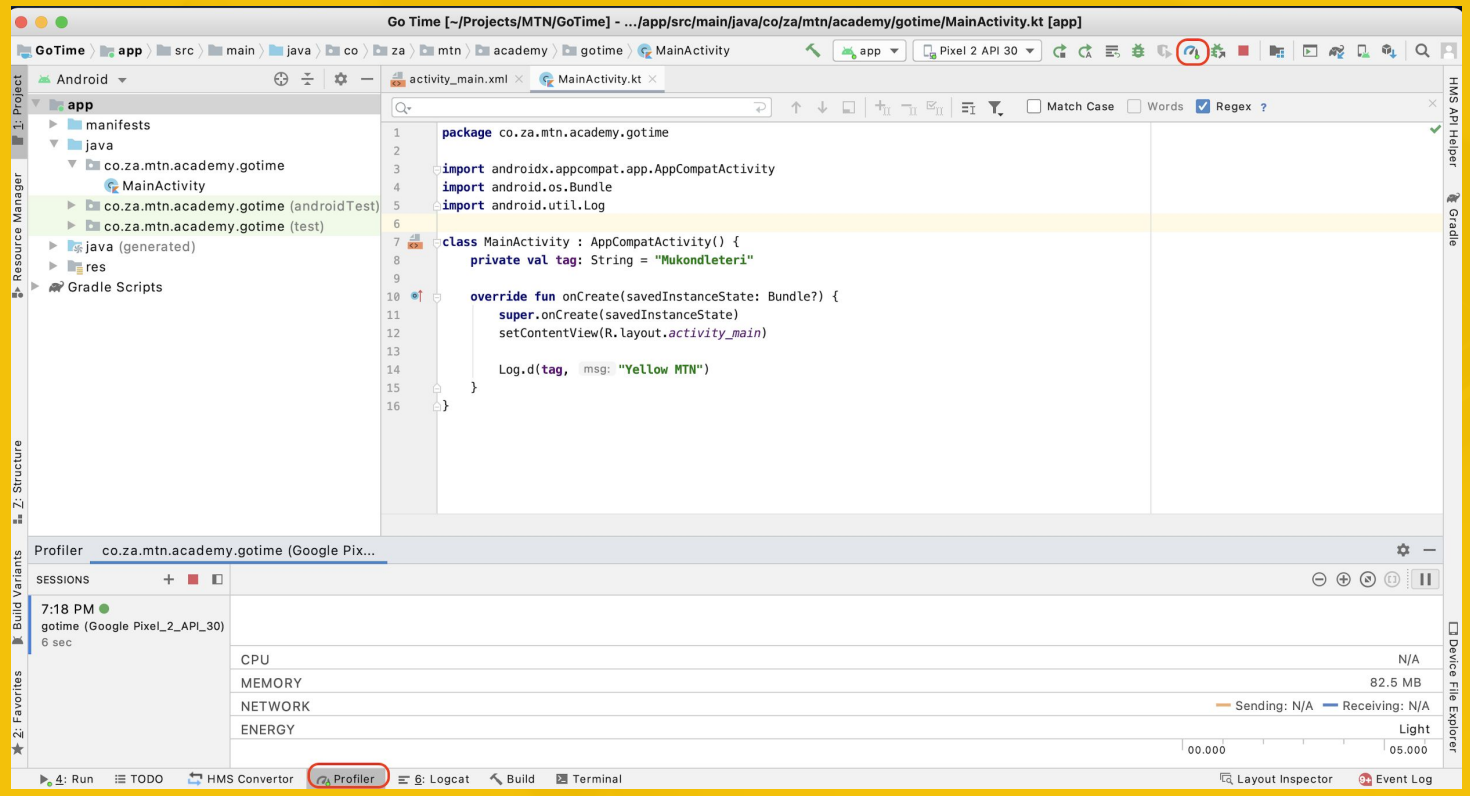
Logcat



Debugging

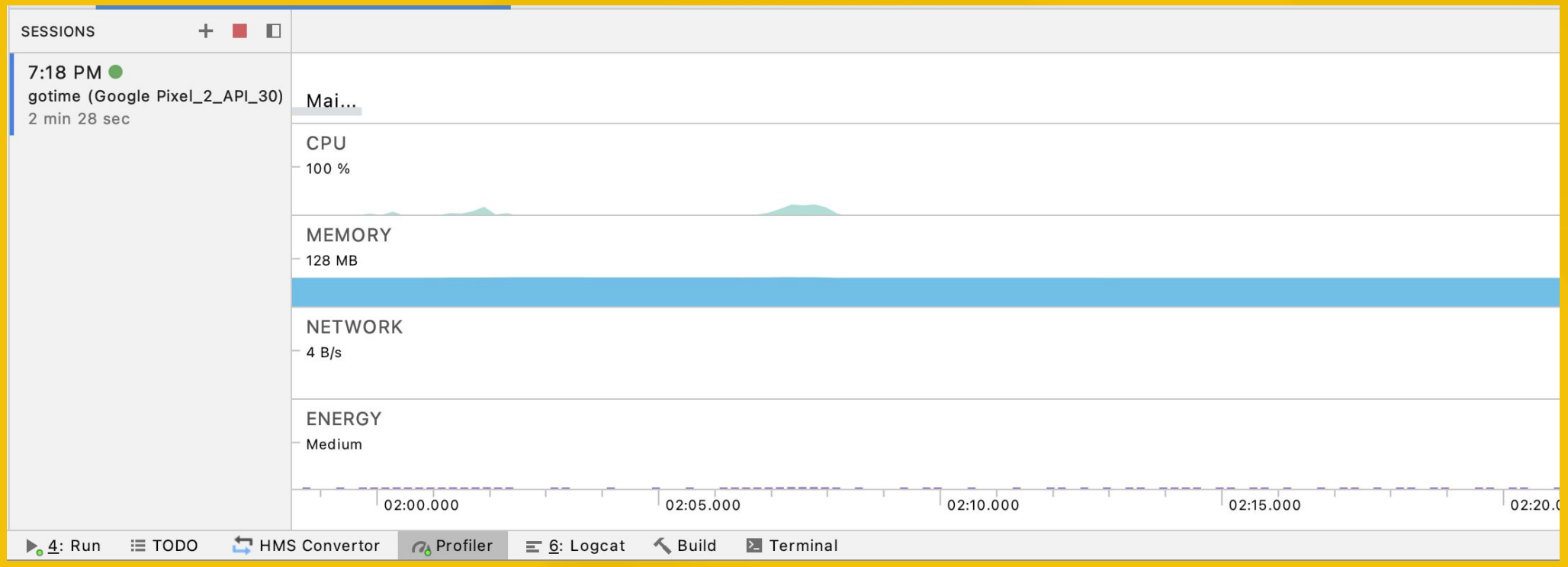
Android Device Monitor

Android Device Monitor - Click on Speedometer icon to run your app in Profiler.



Debugging

Android Device Monitor



MTN Business App of the Year

Android project structure

Android project

Project structure

It is very important to know about the basics of Android Studio's file structure.

Project structure

1. **Manifests**

A manifest file defines the structure and metadata of our app, its components and requirements.

2. **src Folder**

Java or Kotlin source files which control your layouts, etc.

3. **res Folder**

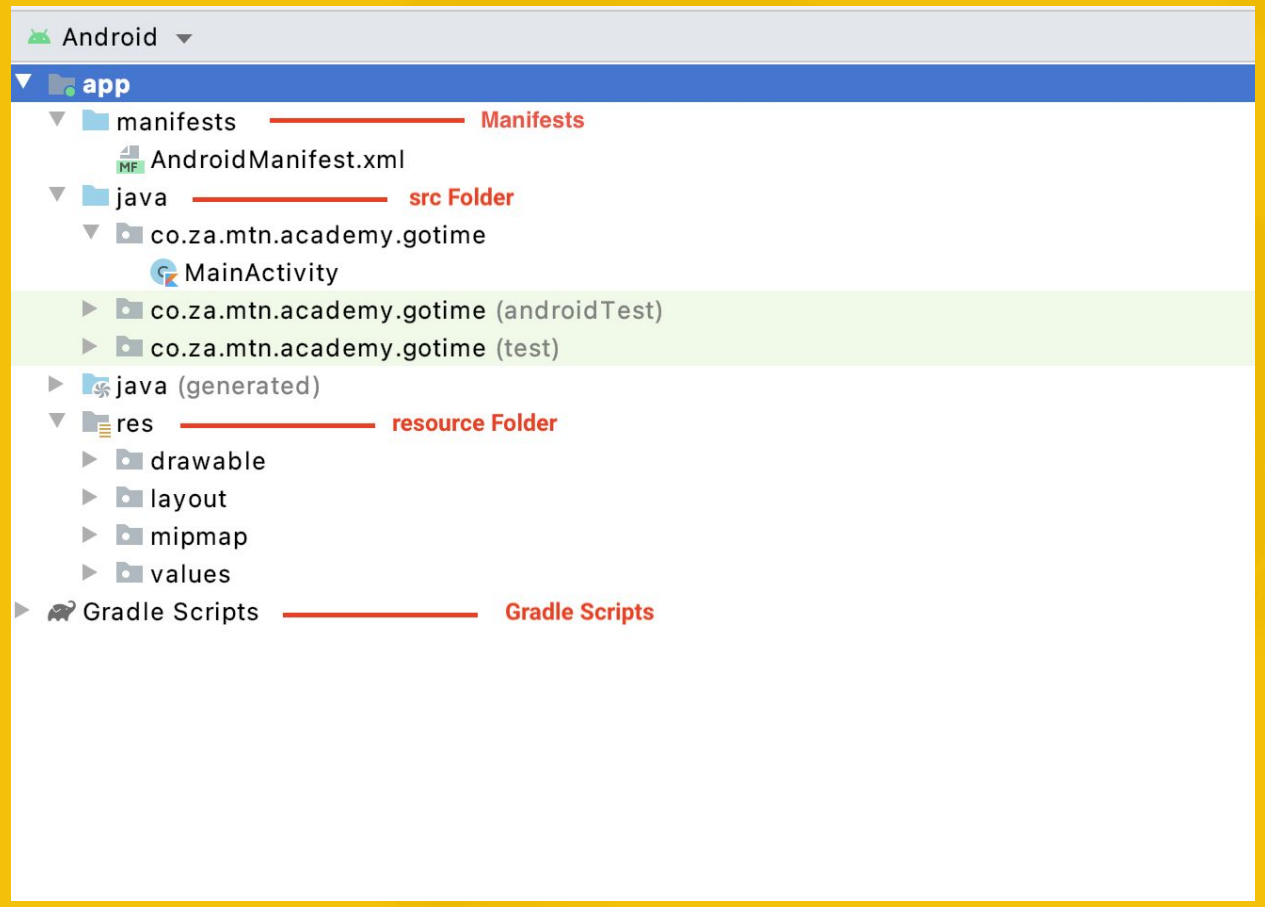
Resources such as Layouts, Shapes, Images, Drawables, etc.

4. **Gradle scripts**

Build configurations and third party plugins.

Android project

Project structure



MTN Business App of the Year

Anatomy of an Android app

Android app

Basics

High-level **concepts** behind the architecture of Android apps.

What makes an Android app

1. Android Manifest File
2. Activities, Services
3. Resources
4. Java code

Android app

Manifest

A manifest file defines the ***structure and metadata*** of our app, its ***components***, and its requirements.

Contents of a manifest file

1. App name, theme, Icon
2. Activities and services
3. Permissions
4. Other components

Android app

Manifest

MTN Business App of the Year

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="co.za.mtn.academy.gotime">

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher" ——— App Icon
        android:label="Go Time" ——— App name
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme"> ——— App theme
        <activity android:name=".MainActivity"> ——— Activity
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
```

Android app

Manifest / Permissions

Android apps must request permission to access ***sensitive user data*** or certain ***system features***. Android has 235 different permissions.

Some examples

1. Internet
2. Camera
3. Contacts

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

Please note some permissions need additional code to take effect.

Android app

Activity

An activity represents a ***single screen*** with a user interface just like window or frame of Java.

"It's similar to a main function in other programs".

Roles of an activity

1. Provides ***a screen*** with which users can interact.

2. Hosts all the ***view components*** of the screen.
   Buttons, Text Views, Image Views, List Views.

3. Hosts all the ***logic*** that manages user interactions.
   Business Logic.

Android app

Activity

How to create a basic activity

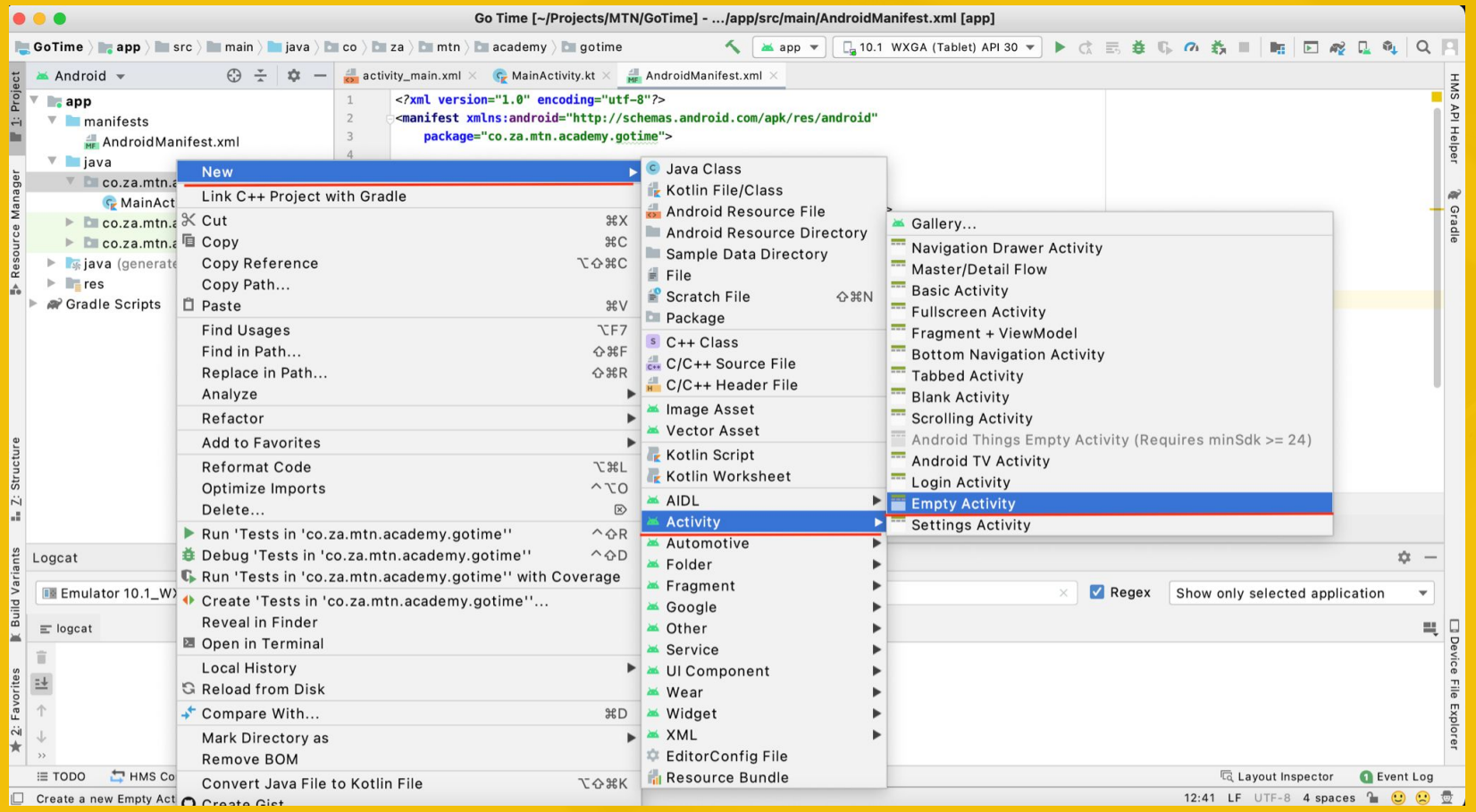
1. Right click on your “root package”.
    - New > Activity > Empty Activity

2. Enter activity name and configurations.

3. Click finish
    - Sit back and allow Android studio to do its work.

Android app

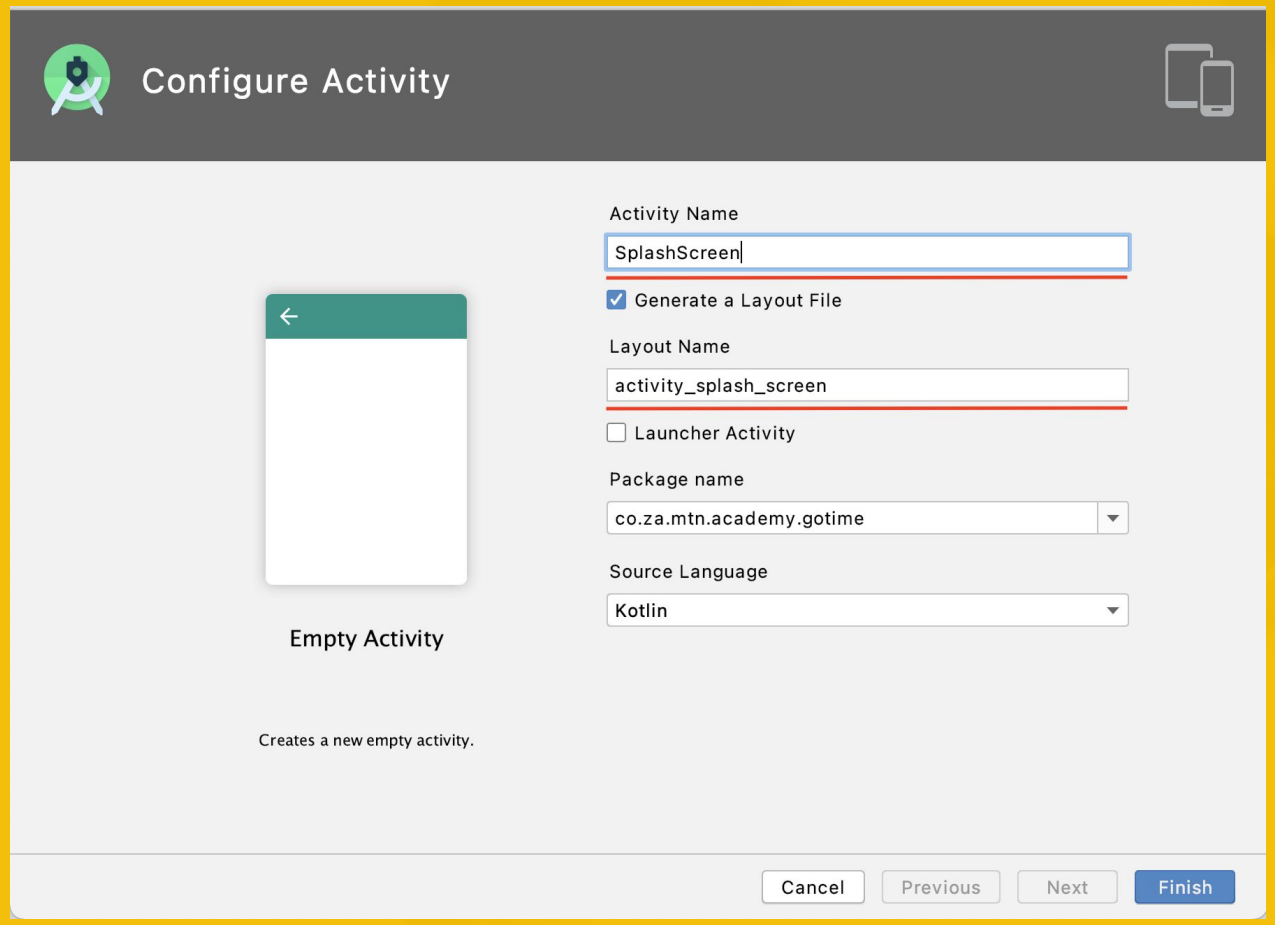
Activity



MTN Business App of the Year

Android app

Activity



Android app

Activity

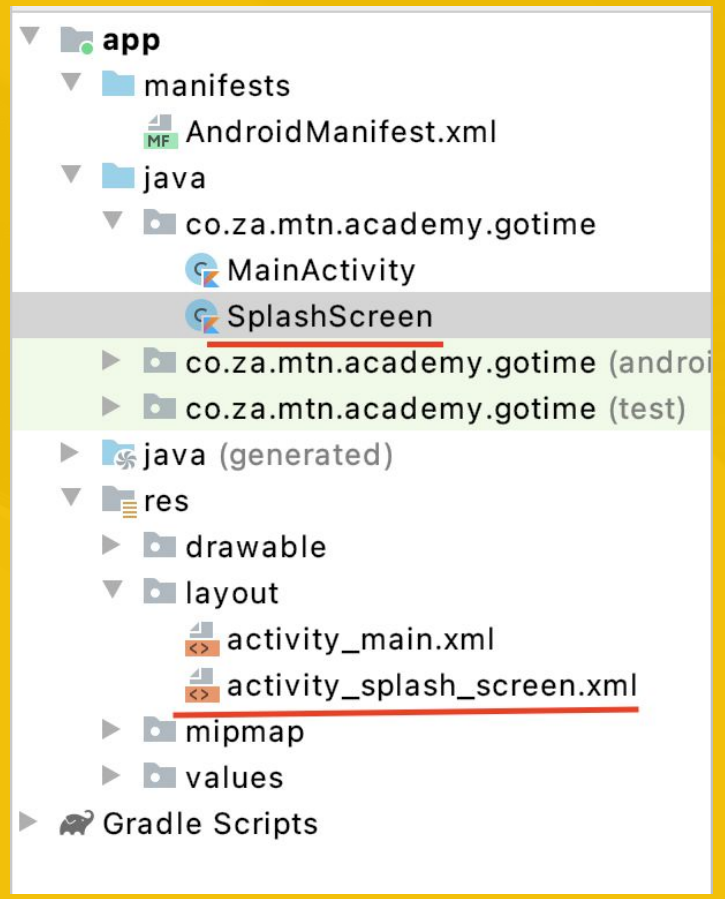
Here is what's created

1. Manifest file amended with new activity configurations.

```
<activity android:name=".SplashScreen"></activity>
```

2. A Kotlin class which extends AppCompatActivity. (SplashScreen.kt)

3. XML file has the activity layout (activity_splash_screen.xml)



Android app

Activity

Activity layouts

1. Defines the structure for an activity user interface.

2. It contains layouts and views of the activity.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Android app

Activity

Demonstration of ***layouts*** and ***views***

Android app

Activity views

Referencing views in Kotlin code

1. XML Layout view. (activity_main.xml).

```
<Button
    android:id="@+id/clickMeButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Click me!"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

2. Kotlin class reference (MainActivity.kt)

```
val clickMeButton = findViewById<Button>(R.id.clickMeButton)
```

Android app

Events

Events are a useful way to collect data about a user's interaction with ***interactive components*** of Applications. Like button presses or screen touch etc.

```
val clickMeButton = findViewById<Button>(R.id.clickMeButton)

clickMeButton.setOnClickListener { it: View!
    Toast.makeText( context: this,  text: "Hello world!", Toast.LENGTH_SHORT).show()
}
```

Android app

Event handlers

Kotlin uses ***lambda*** event listeners or ***class based***.

1. OnClickListener
2. OnLongClickListener
3. OnFocusChangeListener
4. OnMenuItemClickListener

```
clickMeButton.setOnClickListener { it: View!
    Toast.makeText( context: this,  text: "Hello world!", Toast.LENGTH_SHORT).show()
}

clickMeButton.setOnClickListener(View.OnClickListener { it: View!
    Toast.makeText( context: this,  text: "Hello world!", Toast.LENGTH_SHORT).show()
})
```

Android app

Activity life cycle

- **onCreate**
  This is the first callback and called when the activity is first created.
- **onStart**
  This callback is called when the activity becomes visible to the user.
- **onResume**
  This is called when the user starts interacting with the application.
- **onPause**
  The paused activity does not receive user input and cannot execute any code and called when the current activity is being paused and the previous activity is being resumed.

Android app

Activity life cycle

- **onStop**
  This callback is called when the activity is no longer visible.

- **onDestroy**
  This callback is called before the activity is destroyed by the system.

- **onRestart**
  This callback is called when the activity restarts after stopping it.

Android app

Activity life cycle

Demonstration of ***activity life cycle***

Android app

Intents

An Intent is an object that provides ***runtime binding*** between separate components, such as two activities.

Use an intent to start an activity

```
fun goToMain() {
    val intent = Intent( packageContext: this, MainActivity::class.java)
    startActivity(intent)
}
```

Android app

Intents

Sending data to next activity using an intent

```
fun goToMain() {
    val intent = Intent( packageContext: this, MainActivity::class.java)
    intent.putExtra( name: "CustomerName",  value: "Mukondleteri Dumela")
    startActivity(intent)
}
```

Android app

Intents

Receiving data from activity using an intent

```
val customerName = this.intent.getStringExtra( name: "CustomerName")
```

Android app

Activity

Demonstration of ***activity resources***

MTN Business App of the Year

Packaging your app

Packaging

Packaging your app

When you are ready to publish your app, you need to ***sign your app*** and upload it to an app store, such as Google Play.

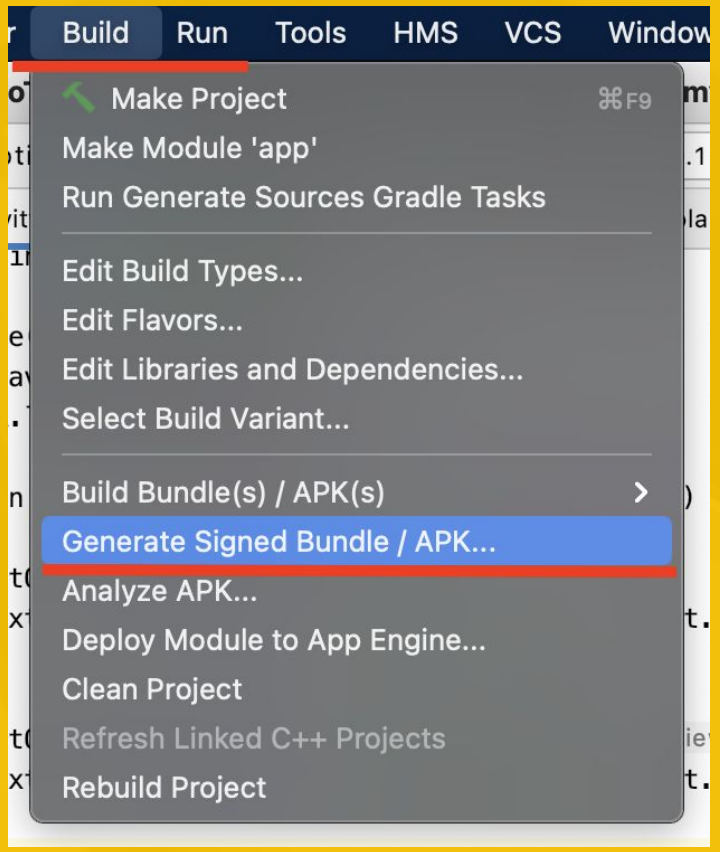
Steps to sign and package your app

1. Click "***Build***" on navigation menu.
2. Click "***Generate Signed Bundle or APK***".
3. Select "***APK***" and click "Next".
4. Click "***New Key Store***" and enter details.
5. Select "***release***" variant and "***v1***" and "***v2***" signatures.
6. Click "***Finish***" and give Android Studio a few seconds.

.

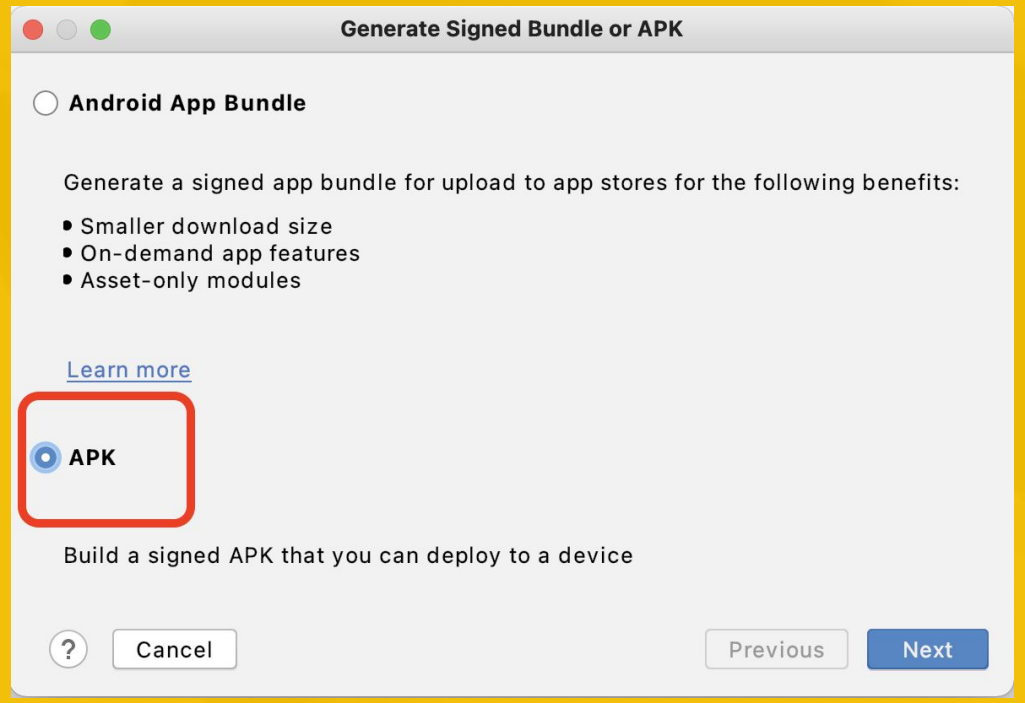
Packaging

Packing your app



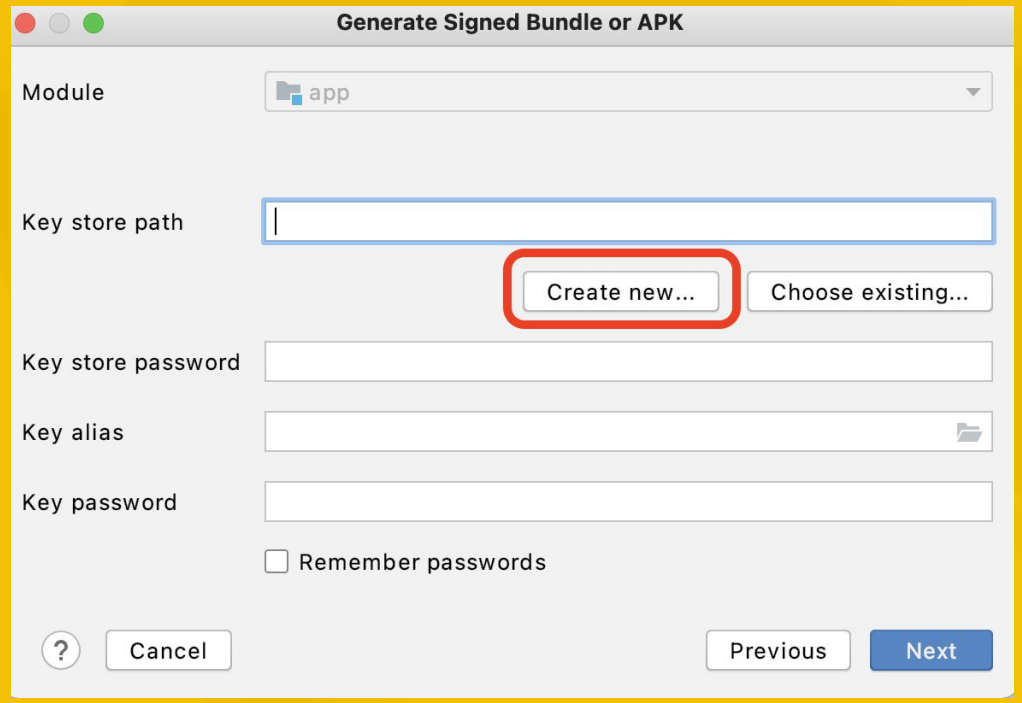
Packaging

Packaging your app



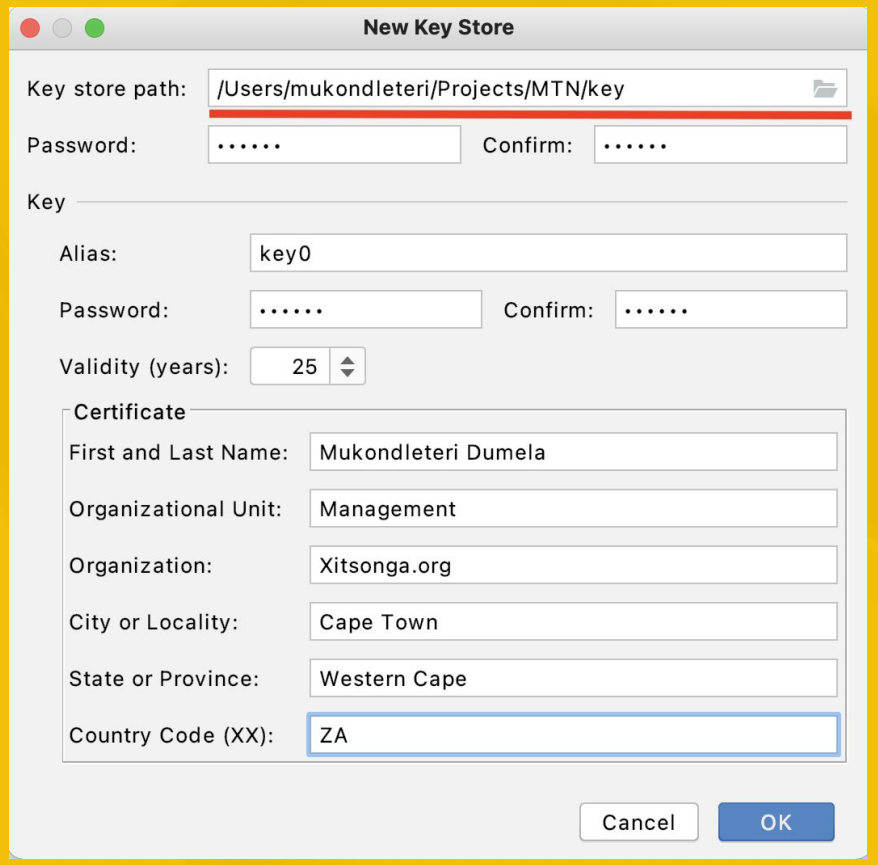
Packaging

Packaging your app



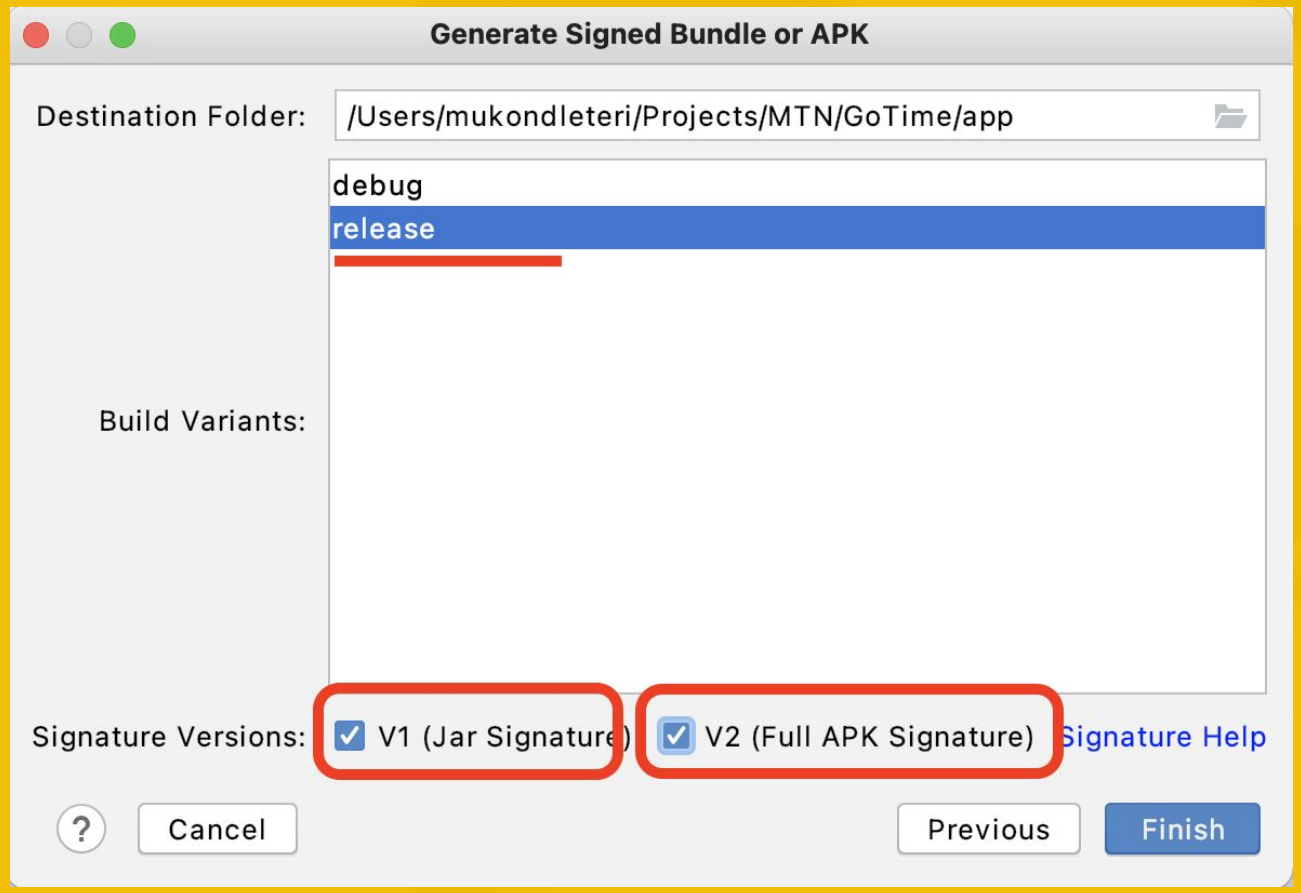
Packaging

Packaging your app



Packaging

Packaging your app



Week 2 Exercise

Create a basic app on Android Studio (Kotlin) with the following:

1. Slash screen with own background colour

2. At least 3 screens

3. Dashboard has buttons that links to other screens on click

SUBMISSION DEADLINE: Sun, 04 June 2021

#RecodeTheWorld

MTN Business App of the Year

bit.ly/appacademy-exercise

It's go time everywhere you go

#RecodeTheWorld

Built it your way

Mukondleteri

MTN Business App of the Year

bit.ly/appacademy-survey

It's go time everywhere you go

#RecodeTheWorld