



# Android Fundamentals

**Dr. Abdelkarim Erradi**  
**CSE@QU**

# Outline

1. Introduction to Android
2. Android Programming Model

# Introduction to Android



android

# Why learn app development?

- Smart devices are ubiquitous
  - Estimated 3.5 billion smartphones + tablets, smart watches, IoT devices...
  - Apps **interwoven** into daily life – work, play, study
  - Mobile = **dominant** end-user device. It represents and intimately “knows” the user: much more than just a small computer, **it represents the user**
  - Brings in outside world: **sensing, location, communication**
- Apps less expensive and more portable
- Large market opportunity for businesses and developers

# Types of mobile development: Web vs Hybrid vs Native



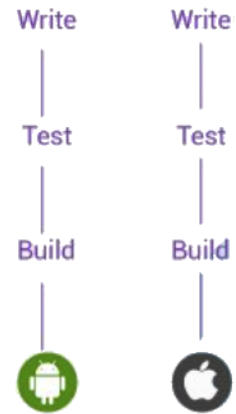
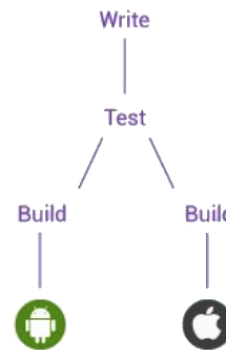
## Web Apps

- Multi-platform
- Leverage existing web dev skillset and code
- Web UI: Run in browser or WebView (can be offline)
- Least access to hardware, sensors, OS
- Slower performance




## Hybrid Apps

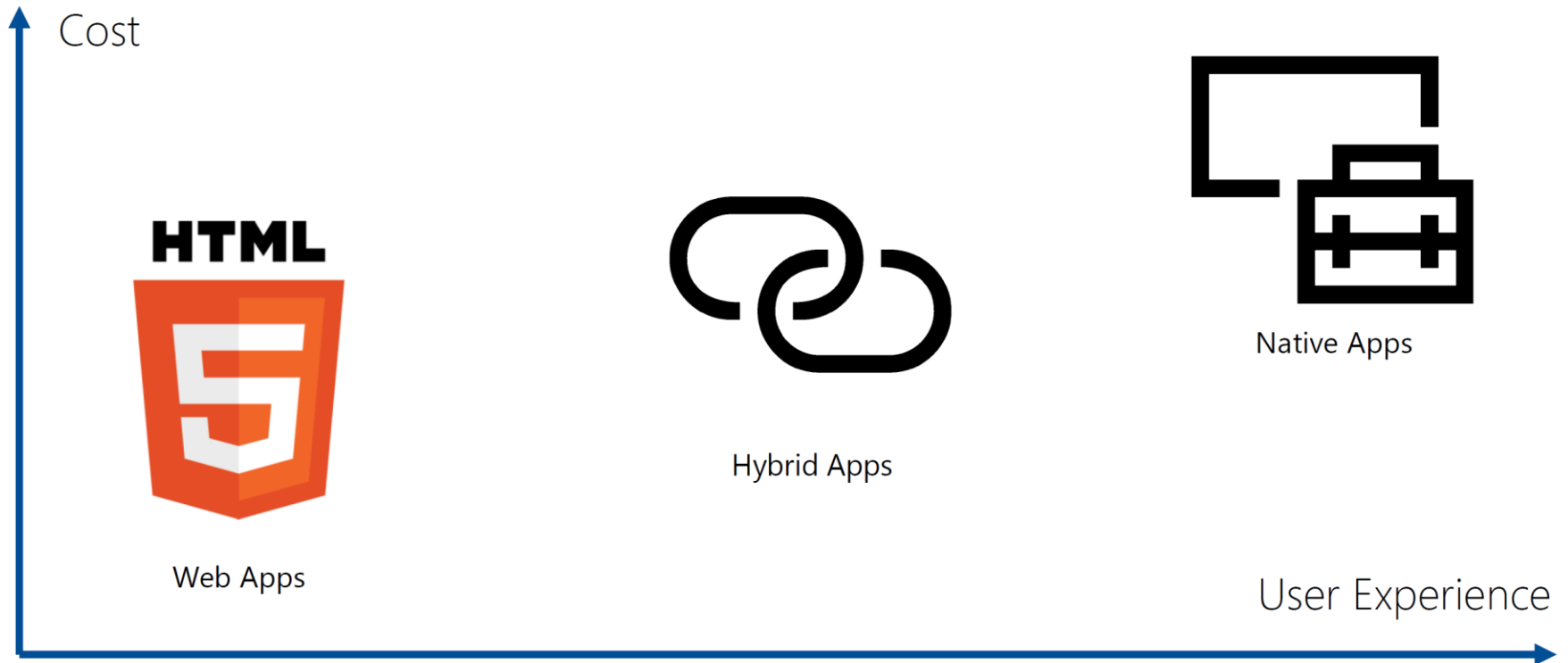
- Shared codebase for iOS & Android
- Leverage web dev skillset and code
- Web app hosted in a **native app shell** to mediate access to hardware, sensors, OS



## Native Apps

- Single platform **iOS** or 
- Native UI and best user experience
- Run directly on OS: Fast performance
- Best system Integration: Full access to hardware, sensors, OS
- More expensive: requires multiple code bases and teams

# Web vs Hybrid vs Native

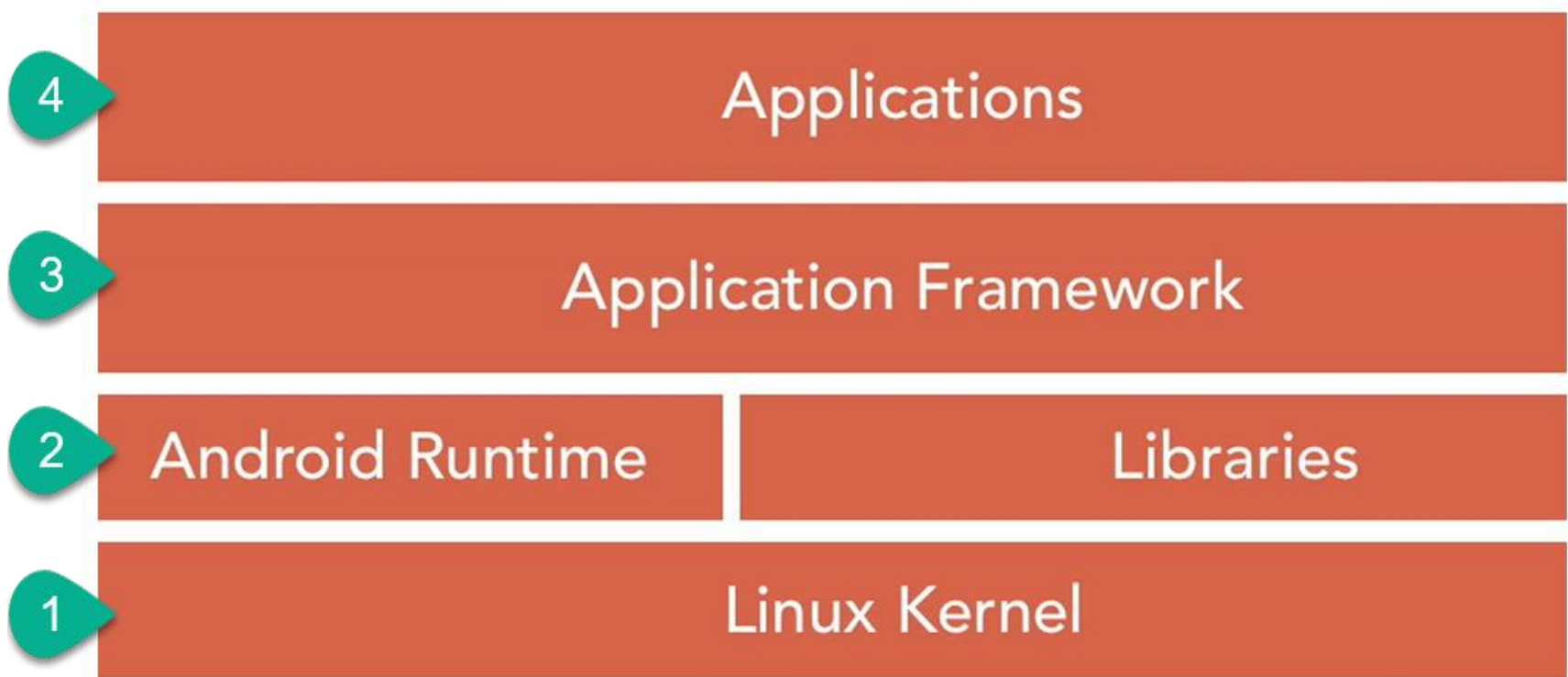


# What is Android?

- Open source mobile operating system (OS) based on [Linux kernel](#) for phones, tablets, wearable
  - originally purchased by Google from Android, Inc. in 2005
- The #1 OS worldwide
  - Used on [over 80%](#) of smartphones
  - As of 2019, over 2.5 billion Android devices worldwide
  - Over 2 Million Android apps in Google Play store
- Highly customizable for devices by vendors



# Android Software Stack



1. Interacts and manages hardware
2. Expose native APIs & run apps
3. Java API exposing Android OS features
4. System and user apps (e.g., contacts, camera)



# Android Software Stack

1. Optimized **Linux Kernel** for interacting with the device's processor, memory and hardware drivers (e.g., WiFi Driver)
  - Acts as an abstraction layer between the hardware and the rest of the software stack
2. **Android runtime (ART)** = Virtual Machine to run Apps
  - Each app runs in its own process and with its own instance of the Android Runtime that controls the app execution (e.g., permission checks) in isolation from other apps
  - Expose native APIs and OS Core Libraries including 2D/3D graphics, Audi Manager, SQLite database, encryption ...
3. **Application Framework**: Java APIs (Application Programming Interfaces) make Android OS features available to Apps (e.g., Activity Manager that manages the lifecycle of apps)

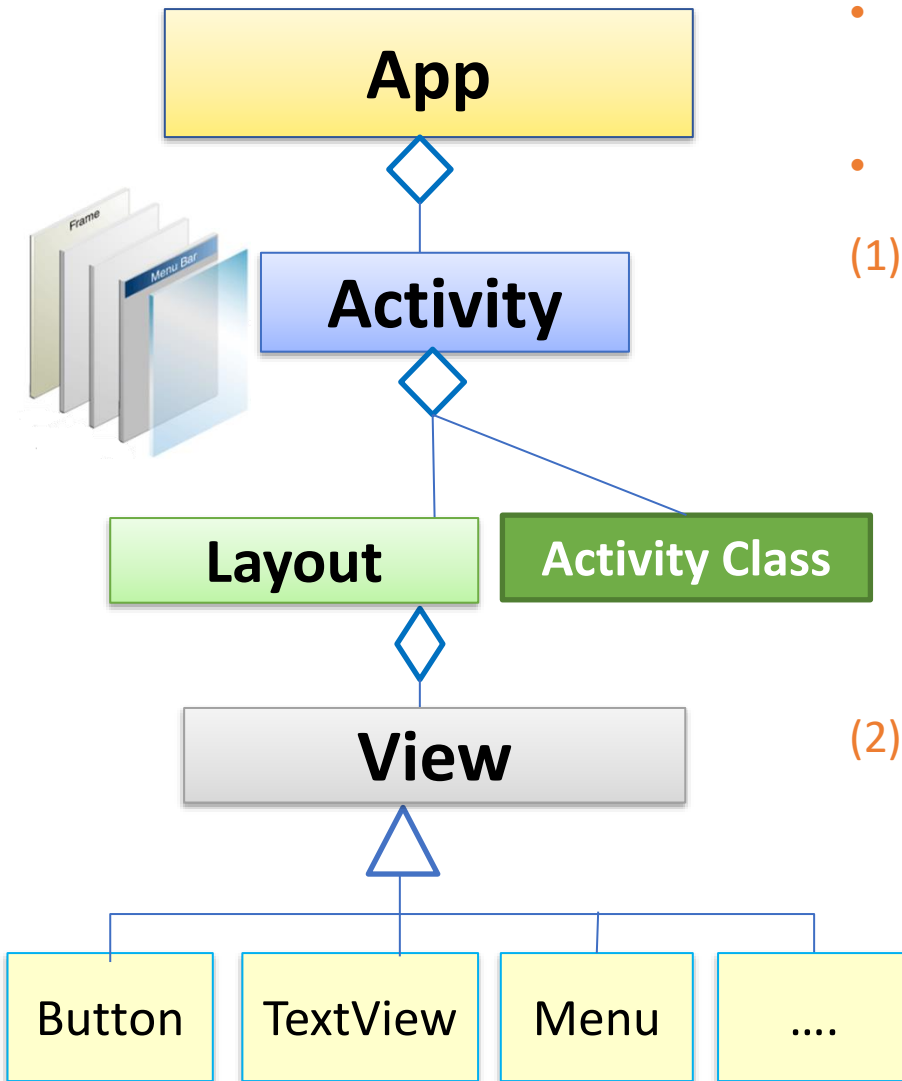
<https://developer.android.com/guide/platform>



# Android Programming Model

# Android Programming Model

**IMPORTANT**



- App is composed of one or more **screens** (called **Activity**)
- An **activity** has:
  - (1) a **Layout** that define its appearance (how it **looks like**)
    - Layout acts as a **container** for UI Components (called **View**)
    - It decides the size and position of views placed in it
  - (2) Activity Kotlin class that provides the data to the UI and handles events
    - UI Components **raise Events** when the user interacts with them (such as a Clicked event is raised when a button is pressed).
    - In the activity class we define **Event Handlers** to respond to the UI events

# Activity

- **Activity** provides the UI that the user interacts with
  - Allow the user to do something such as order groceries, send email
  - Has **layout** (.xml) file & **Activity class**
  - This allows a **clear separation** between the UI and the app logic

- Connecting activity with the layout is done in the **onCreate** method

```
setContentView(R.layout.activity_main)
```

- Activity class defines listeners to handle events:
  - User interaction events such press a button or enters text in a text view
  - External events such as receiving a notification or screen rotation

# Example

```
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
        changeColorBtn.setOnClickListener {  
            greetingTv.setTextColor(getRandomColor())  
        }  
    }  
}
```

Connects  
activity  
with layout



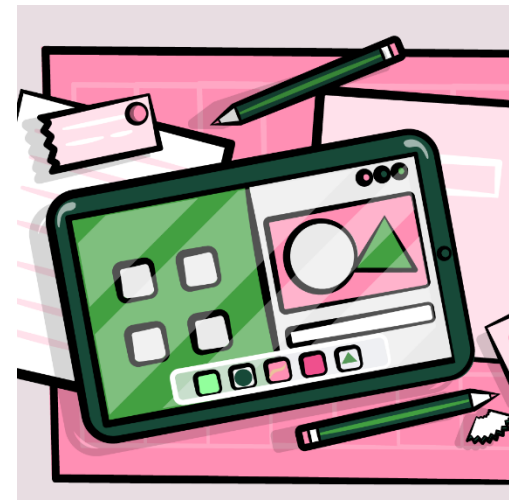
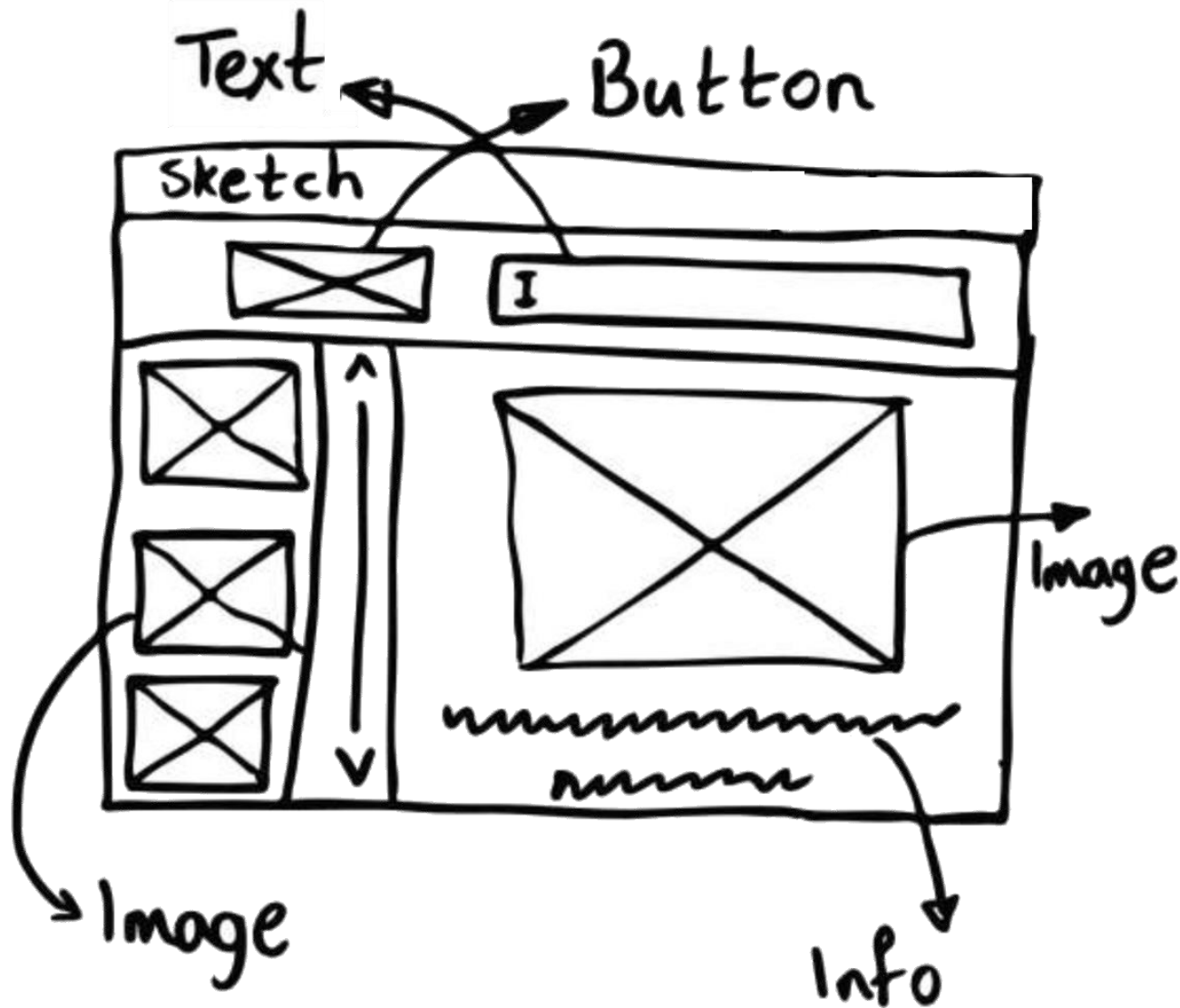
# Event Driven Programming

- GUI programming model is based on **event driven programming**
  - Code is executed upon activation of events
- An **event** is a signal from Android system that something of interest to the app has occurred
  - UI Events (click, tap, swipe, drag)
  - Input focus (gained, lost)
  - Keyboard (key press, key release)
  - Activity events (e.g., onCreate, onRestart)
  - Device: [DetectedActivity](#) such as walking, driving, tilting
- When an event is triggered, an event handler can run to respond to the event. e.g.,
  - When the button is clicked -> load the data from a file into a list

# Steps to creating a GUI Interface

1. Design it on paper (sketch)
  - Decide what information to present to the user and what input they should supply
  - Decide the UI components and the layout on paper
2. Create a layout and add UI components to it using the Layout Editor
  - Use the Layout Editor to group and arrange components
3. Add event handlers to respond to the user actions
  - Do something when the user presses a button, selects an item from list, change text of input field, etc.

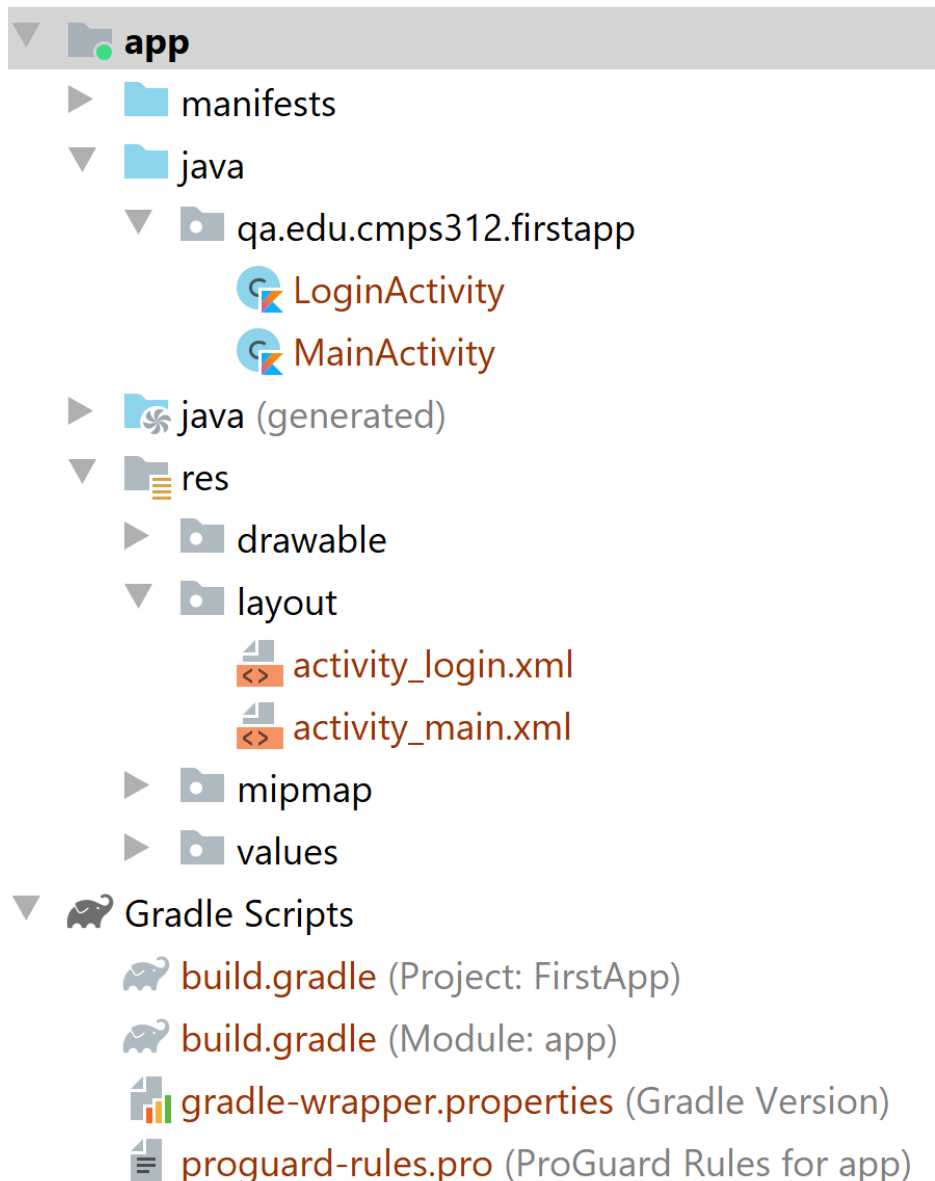
# UI Sketch - Example



You may design different layouts per screen size



# Project structure



## AndroidManifest.xml

- app config and settings (e.g., list app activities and required permissions)

## java/...

- Kotlin source code

## res/... = resource files (*many are XML*)

- drawable/ = images
- layout/ = GUI layouts
- menu/ = app menu options
- values/ = **Externalize** constant values
- strings/ = user-visible strings
- styles/ = appearance styling

## Gradle

- a build/compile management system
- **build.gradle** = define config and dependencies (one for entire project & other for app module)

# Resources

- **Separate** static data needed by the UI from the code that does computations and handles events
  - **drawable** – image files
  - **layout** - layout files for Activities
  - **strings.xml**: Defines any user-visible strings as string resources + Supports localization
  - **dimens.xml**: Defines any view/text dimensions
  - **colors.xml** & **styles.xml**: Define reusable colors and styles to customize the overall design aesthetic of the app
- Resource files are stored in **res** folder

# Refer to resources in code

- Layout:

```
setContentView(R.layout.activity_main)
```

- View:

```
greetingTv.text = "Salam"
```

- String:

In Kotlin: `R.string.title`

In XML: `android:text="@string/title"`

# Externalize Constants

- Edit res/layout/activity\_main.xml
  - Replace string “Start” of the start button with “@string/start”.
  - Benefit = Localization, e.g., es/values-es/strings.xml

Start

Comienzo

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="start">Comienzo</string>
    <string name="stop">Detener</string>
</resources>
```

res/layout/activity\_main.xml

1. In the XML editor, right-click the `android:text="Start"` attribute of the `<Button>` tag. A context menu appears with the option "Extract string resource" highlighted.

2. The "Extract Resource" dialog is shown. The "Resource name" field contains "start", the "Resource value" is "Start", the "Source set" is "main", and the "File name" is "strings.xml". The "Create the resource in directories:" section has "values" checked.

3. The resulting `strings.xml` file is shown, containing the new string resource: `<string name="start">Start</string>`.

res/values/strings.xml

# Resources

- Android Kotlin Fundamentals Course
  - <https://codelabs.developers.google.com/android-kotlin-fundamentals/>
  - <https://developer.android.com/courses/android-basics-kotlin/course>
- Android Dev Guide
  - <https://developer.android.com/guide/>