Mobile Application Development
(CSE3709)

(using Android Programming)

Dr. Nikhil Kumar

Genesis of Android

- In October 2003, Andy Rubin, Rich Miner, Nick Sears, and Chris White has started Android Inc. with the intention to create an advanced OS for digital cameras. They have used the Linux kernel to create an embedded OS called the Android operating system.
- Google acquired Android Inc. in August 2005.
- In November 2007, several tech companies including Google, Sony,

HTC, Samsung, and others created a consortium called the Open Handset Alliance (OHA) to develop open standards for mobile devices.

- The OHA released the first Android OS open-source software in 2007.
- One year later, in October 2008, the first Android phone, the HTC Dream, was launched.

More about Android

- **Android is Open Source:** Android is a mobile operating system based on the Linux kernel. It is an open-source operating system for mobile devices that has been built upon other open-source projects. As a developer, this means you have access to the source code of the platform, which in turn helps you to better understand how the interface and the various other pieces of the platform work. If you happen to find a bug, you can also submit your solution for the issue and get rewarded by being a part of future Android improvement.

- **Android Libraries:** In addition to phones and tablets, Android powers watches, TVs, and cars and has a library to develop apps for the IoT (internet of things). Android provides users with the interface for touchscreens. Users can interact with the Android devices by swiping, tapping, pinching, or using the virtual keyboard and voice. The voice access app for Android lets you control your device with spoken commands, i.e., use your voice to open apps, navigate, and edit text hands-free. Android has built-in support for Bluetooth, USB, and peripherals such as printers. Android has sensors to discover actions such as user moving, phone rotation, tilting steer, etc.
- Android is the most popular operating system in the world, with over 2.5 billion active users spanning over 190 countries (holding more than 71.1% of mobile market – Statista Report, 2022).

Android Development Environment

Android Development Environment

- **Java** (by Sun Microsystems, now Oracle Inc.) and **Kotlin** (JetBrains) •

Android SDK (Software Development Kit: libraries, debugger, emulator, etc.)

- **XML** (eXtensible Markup Language: to store and transport data in a text-based document)
- **Gradle** (a general-purpose dependency management & build automation tool)
- **IDEs** (e.g. Eclipse, **Android Studio** (JetBrains' IntelliJ IDEA))
- **AVD** (Android Virtual Device) or Android Emulator
- **DVM** (Dalvik Virtual Machine) – now replaced by Android Runtime (ART)
- **APK** (Android Package) – archive file format that Android uses to distribute and install apps
- **DEX** – (or **d8**) – a tool to convert java bytecode to DEX byte code

View Controller and App Development

- Android programming involves not only a clear separation between the visual components and the app's computational logic, or business objects, but also using XML elements to represent the visual objects. You can create all the GUI (graphical user interface) components of your app using XML files. At runtime, the XML files are converted to Java code and linked to the main source code to be executed. The separation between views, logic, and data is essential in the Android app development philosophy.

Android's Main Program

Android's Main Program

- For any program to start, there is a need for an entry point into the program. For Java programs, the main method in a class is the entry point. For Web applications, you type the address of the website in the browser which looks for the index.html file to load the page and display it. To run Android programs, you need to create the AndroidManifest.XML file. All Android application programs start with the manifest file. When you use Android Studio to create a project, the manifest file is automatically created for you. Every time you add an Android Activity class, i.e., screen, to your app, the Activity is automatically added to the manifest file. Inside the manifest file,

you declare one of the Activity classes to be the main class, i.e., the starting point for your app.

Android Versions

Name	Internal codename[9]	Version number(s)	API level	Initial stable release date
Android 1.0	—	1.0	1	September 23, 2008
Android 1.1	Petit Four	1.1	2	February 9, 2009
Android Cupcake	Cupcake	1.5	3	April 27, 2009
Android Donut	Donut	1.6	4	September 15, 2009
Android Eclair	Eclair	2.0	5	October 27, 2009
		2.0.1	6	December 3, 2009
		2.1	7	January 11, 2010[16]
Android Froyo	Froyo	2.2 – 2.2.3	8	May 20, 2010
Android Gingerbread	Gingerbread	2.3 – 2.3.2	9	December 6, 2010
		2.3.3 – 2.3.7	10	February 9, 2011
Android Honeycomb	Honeycomb	3.0	11	February 22, 2011
		3.1	12	May 10, 2011
		3.2 – 3.2.6	13	July 15, 2011

Android Versions

Android Ice Cream Sandwich	Ice Cream Sandwich	4.0 – 4.0.2	14	October 18, 2011
		4.0.3 – 4.0.4	15	December 16, 2011
Android Jelly Bean	Jelly Bean	4.1 – 4.1.2	16	July 9, 2012
		4.2 – 4.2.2	17	November 13, 2012
		4.3 – 4.3.1	18	July 24, 2013
Android KitKat	Key Lime Pie	4.4 – 4.4.4	19	October 31, 2013
		4.4W – 4.4W.2	20	June 25, 2014
Android Lollipop	Lemon Meringue Pie	5.0 – 5.0.2	21	November 4, 2014[17]
		5.1 – 5.1.1	22	March 2, 2015[18]
Android Marshmallow	Macadamia Nut Cookie	6.0 – 6.0.1	23	October 2, 2015[19]
Android Nougat	New York Cheesecake	7.0	24	August 22, 2016
		7.1 – 7.1.2	25	October 4, 2016

Android Versions

Android Oreo	Oatmeal Cookie	8.0	26	August 21, 2017	January 2021
		8.1	27	December 5, 2017	October 2021
Android Pie	Pistachio Ice Cream[23]	9	28	August 6, 2018	January 2022
Android 10	Quince Tart[24]	10	29	September 3, 2019	February 2023
Android 11	Red Velvet Cake[24]	11	30	September 8, 2020	January 2024
Android 12	Snow Cone	12	31	October 4, 2021	
Android 12L	Snow Cone v2	12.1[a]	32	March 7, 2022	
Android 13	Tiramisu	13	33	August 15, 2022	
Android 14	Upside Down Cake[27]	**14**	34	October 4, 2023	
Android 15	Vanilla Ice Cream[28]	15	TBA	Q3 2024	—

Installing Android

Installing Android

- Install JDK
https://download.oracle.com/java/18/latest/jdk-18_windows-x64_bin.exe

Download Android Studio & App Tools - Android Developers

- Install Android Studio
- Creating an Application (or project) • Tools:
  - SDK Manager
  - Project Window
  - Device Manager
  - Design Window
  - Code Window
  - Output Window

- Toolbar

Other Operations and tools

- Turning .class into .dex Bytecode

Source code (*.java) → using javac → Java bytecode (*.class) → using ProGuard or R8 → optimized Java bytecode (*.class) → using D8 or DEX → Dalvik optimized bytecode (*.dex)

- APK (Android Package Kit): The APK Packager combines the DEX files and compiled resources into a single APK file. The APK is a file format designed for distributing Android applications. It is simply just a ZIP archive file similar to the Java JAR package designed for the distribution of Java applications. In addition to the .dex file, APK contains other files, for example, AndroidManifest.xml. In addition to the APK, Android supports another build format, the Android App Bundle (AAB) build format.
- Android App Bundle: The Android App Bundle (AAB) is Android's new format to build and release apps. The generated bundle includes all your app's compiled code and resources. However, Google Play defers generating the APK to optimize downloading for each device

configuration. That means only the code and resources that are needed for a specific device are downloaded. To use or switch to the new publishing format, you don't need to refactor your code. When you develop apps using Android Studio 3.2 or higher versions, you can generate AAB builds from Android Studio.

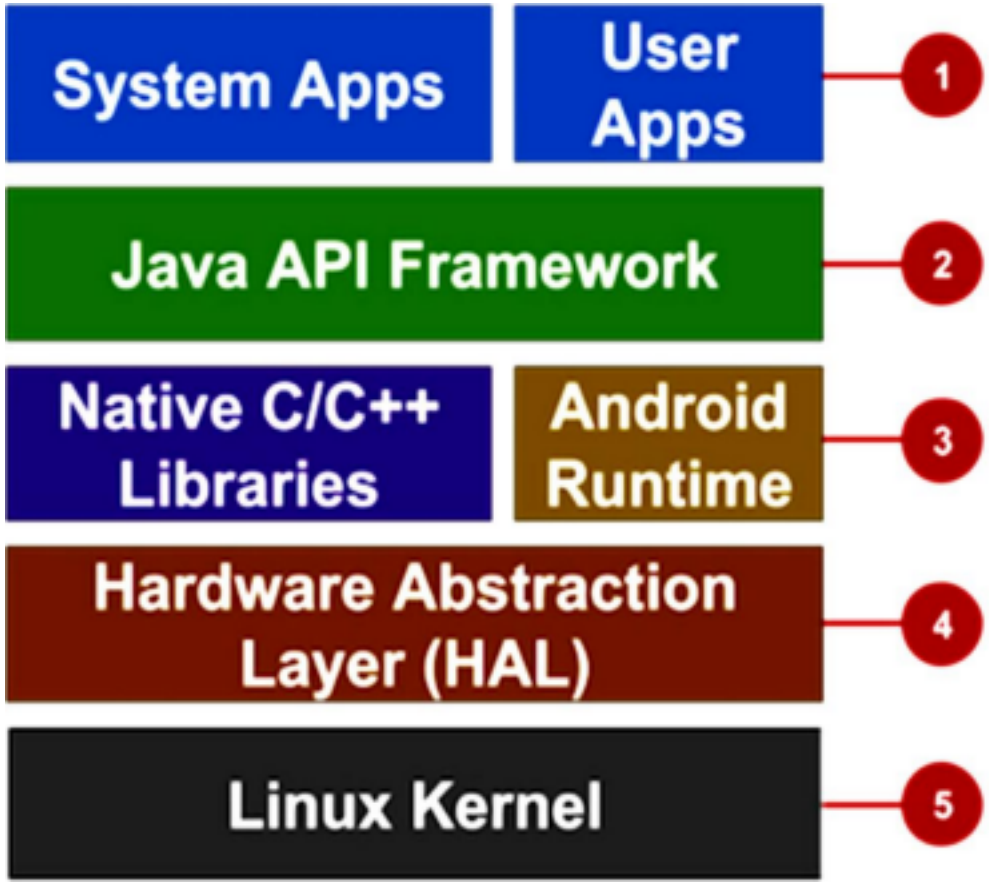
Other Operations and tools

Other Operations and tools

- Android Architecture
- Android Training Material:

  https://google-developer training.github.io/android-developer fundamentals-course-concepts-v2/

- Android Developers' Guide



https://developer.android.com/guide •

Creating XML Layouts for Android

Create XML layouts for Android

Dr. Nikhil Kumar

Creating First Android Application

Creating First Android Application

1. **Create new project:** File -> New -> New Project
2. **In new project window:** Choose ***"Empty Views Activity"*** under Phone and Tablet.
   - **i. Name:** Name of your application
   - **ii. Package name:** Every Android app has a unique application ID, which identifies your app on the device and in the Google Play Store.
   - **iii. Save location:** Location where your project files will reside in your computer.

iv. Language: Java or Kotlin

v. Minimum SDK: It is the minimum version of Android on which your app can run. It specifies the lowest Android API level that your app supports.

3. Following files will be generated in the project

i. AndroidManifest.xml

ii. MainActivity.java

iii. actuvity_main.xml

iv. strings.xml

v. colors.xml

MainActivity.java

```
package com.example.cse1;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
public class MainActivity extends AppCompatActivity {
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
```

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

This file contains the logic part of the main Activity of the application. The first line represents the package of your application. The following two lines import the minimum necessary classes. The MainActivity class extends the AppCompatActivity class. In the next line, an overridden method onCreate() is used to execute the things on the creation of the Activity. Next, the method setContentView() is called to render the graphical components (a View or a Layout) on the screen.

activity_main.xml

The activity_main.xml is a layout file available in the res/layout directory, referenced by your application when building its interface. You will modify this file frequently to change your application's layout.

AndroidManifest.xml

AndroidManifest.xml file plays a vital role in Android development. This is the heart of any android application. It has all details about the application, including the package of the app, version of the

app, minimum and maximum Android SDK versions supported by the app, themes, permissions, activities, intents, broadcast receivers, content providers and much more.

strings.xml

Android does not recommend any hard-coded string in the layout file. Strings should be saved separately as a resource in a resource file strings.xml. This file is found in the “values” folder under the “res” folder.

colors.xml

This file contains the colors as resources. A color value is defined in XML. The color is specified with an RGB value and alpha channel. You can use a color resource in any place that accepts a hexadecimal color value. You can also use a color resource when a drawable resource is expected in XML.

- **R.java**
  - Android recommends that the logic part should be written in Java and the design part should be written in XML, as using these different technologies, makes it easier to manage layouts, views, and other resources separately. The graphical components coded in XML are going to be used in Java code. But how can it be done? The answer is – by using R.java. R.java is a Java file created automatically by the Android development environment during the programming. It works as a bridge between

logic part and design part. It interprets the references of UI components of XML files to the Java file and to other XML files. It is recommended that you do not make any change in the R.java file because you can alter the reference of any graphical component by mistake.

Dr. Nikhil Kumar

Building blocks of Android application

Building blocks of Android application

- Building blocks or application components of an Android application define the

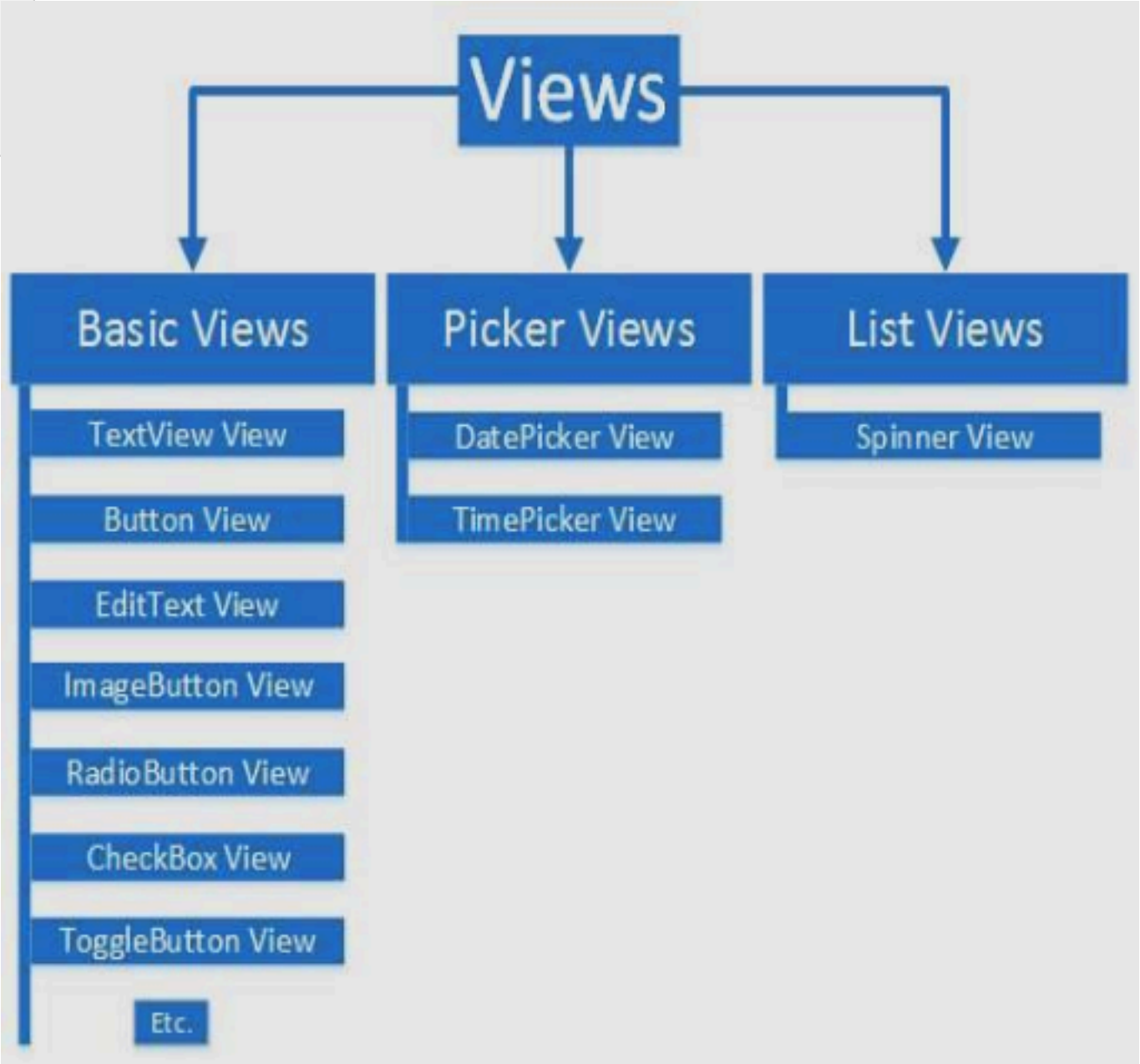
application's behavior. Each Android application has four components; each component plays a specific role and has its lifecycle.

- **Activity:** An Activity is a user interface provided as a screen with which a user interacts.
- **Services:** A service is an application component that runs in the background and does not need any user interface.
- **Content Providers:** A content provider is a component that manages the application data stored in the SQLite database, file system, on the web or any other storage location accessed by your application. Other applications can access or modify your application's data using the content provider.
- **Broadcast Receivers:** A broadcast receiver is used to broadcast the announcements to the whole system, whether it is a notification to the status bar to alert the user or a message to another application or Activity.

- All these components need to be registered in the AndroidManifest.xml file.

Using Views and Layouts

- The graphical design of an Activity is a collection of different graphical components (such as Buttons, Text Fields, etc.) called **views**.
- Usually, Activity uses the **layout managers** (such as Linear Layout, Relative Layout, Constraint Layout, etc.) to contain these views. Layout managers help the Activity to arrange the views on



the screen. These layout managers are also called **view groups**.

Measurement Units

- To specify the size of different components (such as Views, GroupViews or text) on the Android user interface, there are some measurement units. These are XML-defined dimension values.

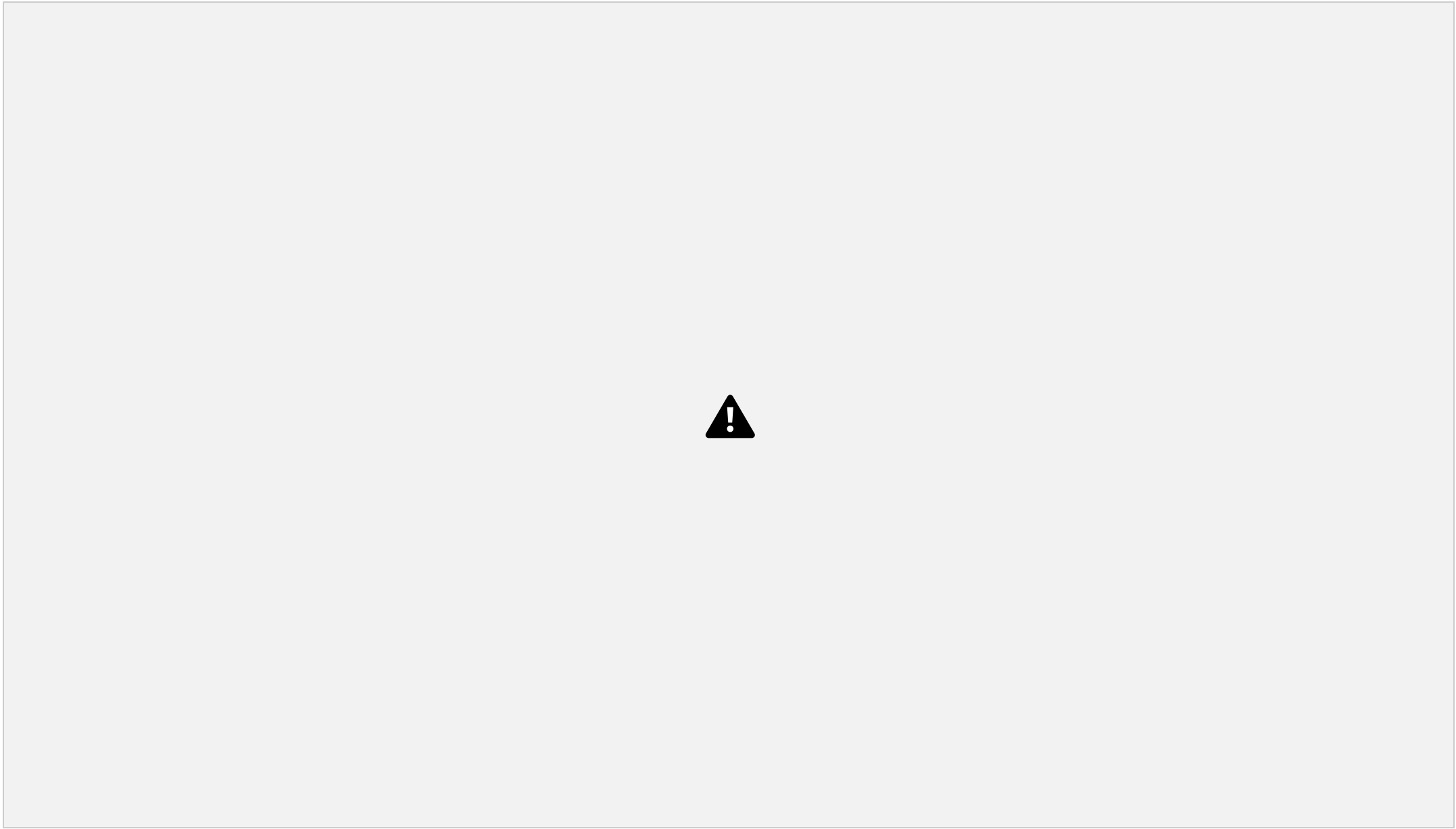
1. **dp (Density Independent Pixel)**: One inch of the device's physical screen is equivalent to 160 dp. This measurement unit is recommended for specifying the size of Views in the layout. It is also represented as a **dip**. It adjusts the size of the Views accordingly on screens of different densities. (The screen's density equals the total number of pixels per square inch of the screen.)
2. **sp (Scale Independent Pixel):** It is similar to dp, i.e. one inch of the device's physical screen equals 160 sp. It is recommended to specify the font size. It is different from the dp as it remembers the font size preferences of the user while the dp does not. It can also adjust the font size of the text according to the different screens of different densities.
3. **pt (Point):** One inch of the physical screen equals 72 pt. It is not independent of the density of the

screen.

4. **px (Pixel):** One px represents the one pixel of the screen. It is not recommended because the Views cannot be rendered accurately on screens of different sizes.
5. **mm (Millimeters):** Based on the physical size of the screen.
6. **In (Inches):** Based on the physical size of the screen.

Dr. Nikhil Kumar

Measurement Units



Measurement Units continue…

- There are two predefined constants which are used to assign the size to graphical components:
  1. **match_parent:** This is a predefined constant used to define the size of a View or ViewGroup as big as its parent. The match_parent is formerly known as **fill_parent**. It is renamed as match_parent after **API Level 8** or higher. Although fill_parent is still in use but not recommended. Using a match_parent is called **minus padding**.
  2. **wrap_content:** This is also a predefined constant used to define the size of a View or a ViewGroup to be just big enough to surround its content. Using a wrap_content is called **plus padding**.

Dr. Nikhil Kumar