5. Build An Application

This section describes how to build a simple Android app. First, you learn how to create a "Hello, World!" project with Android Studio and run it. Then, you create a new interface for the app that takes user input and switches to a new screen in the app to display it.

Before you start, there are two fundamental concepts that you need to understand about Android apps: how they provide multiple entry points, and how they adapt to different devices.

Apps provide multiple entry points

Android apps are built as a combination of components that can be invoked individually. For example, an activity is a type of app component that provides a user interface (UI).

The "main" activity starts when the user taps your app's icon. You can also direct the user to an activity from elsewhere, such as from a notification or even from a different app.

Other components, such as WorkManager, allow your app to perform background tasks without a UI.

Apps adapt to different devices

Android allows you to provide different resources for different devices. For example, you can create different layouts for different screen sizes. The system determines which layout to use based on the screen size of the current device.

If any of your app's features need specific hardware, such as a camera, you can query at runtime whether the device has that hardware or not, and then disable the corresponding features if it doesn't. You can specify that your app requires certain hardware so that Google Play won't allow the app to be installed on devices without them.

Some Terminologies That You should Know Before Developing An Android App

Android SDK Version

The minimum required SDK is the lowest version your app will support. Your app will run on devices with this level API or higher. Choosing the right version is a bit of a balance since we want to be able to use newer features from later versions, but still write an application that applies to most users.

App Module

Within a project, there are one or more modules. Typically, an app can be developed with a single module app. Modules provide a container for your app’s source code, resource files, and app level settings, such as the module-level build file and Android manifest file. Each module can be independently built, tested, and debugged.

Android Manifest

Every Android app must include a file called AndroidManifest.xml at its root. The manifest file contains essential information about the app, such as what components it contains, required libraries, and other declarations.

Gradle Build File

Gradle Scripts > build.gradle

There are two files with this name: one for the project and one for the app module. Each module has its own build.gradle file.

Layout Editor

In Android Studio, the app layout can be edited using Android studio Layout Editor. This generates (and modifies) layout XML files in app->scr->res. These files can also be coded directly or dragging GUI components in the design editor.

Android Simulator

The Android emulator allows you to run your app on an Android virtual device (AVD), which behaves just like a physical Android device. You can set up numerous AVDs, each emulating a different type of device.

Android Logcat

```
Log v String tag  String message
// Logs a verbose message.
Log d String tag  String message
// Logs a debug message.
Log i String tag  String message
// Logs a information message.
Log w String tag  String message
// Logs a warning message.
Log e String tag  String message
// Logs a error message.
```

Adding messages to a log can be a useful way of checking that your code works the way you want. Each message is composed of a String tag you use to identify the source of the message, and the message itself.

Top Level Components

The ability to text, email, play games and much more are accomplished in Android applications through four top-level component classes: BroadcastReceiver, ContentProvider, Service, and Activity. These are all represented by Java Objects.

Activity Components

Activities provide the most visible of the application components. They present content to the screen and respond to user interaction. Activity components are the only components that present interactive content to the user. An Activity represents something an application can do, and an application often “does” several things – meaning, most applications provide more than one Activity.

Android Views

In Android, Views are atomic, indivisible elements that draw themselves to the screen. They can display images, text, and more. Like periodic elements combine to form chemicals, we combine Views to form design components that serve a purpose to the user.

Android ViewGroups

In Android, ViewGroups arrange Views (and other ViewGroups) to form meaningful designs. ViewGroups are the special bond that combines Views to form design components that serve a purpose to the user. ViewGroups are Views that may contain other Views within them.

Android Layout Files

```
<ConstratinLayout ...>
    <ImageView id "thumbnail" .../>
  <ImageView id "avatar" .../>
  <ImageView id "title" .../>
  <imageView id "subtitle" .../>
</ConstratinLayout>
```

In Android, each layout is represented by an XML file. These plain text files serve as blueprints for the interface that our application presents to the user.

Mandatory Layout Attributes

In Android, two attributes are mandatory within every layout: layout_width and layout_height, and the most common values for these attributes are match_parent and wrap_content.

Constraint Layout

ConstraintLayout arranges its children relative to itself and other children (leftOf, rightOf, below, etc.). It is more complex than a linear or frame layout, but it's a lot more flexible. It's also much more efficient for complex UIs as it gives you a flatter view hierarchy, which means that Android has less processing to do at runtime. Another advantage of using constraint layouts is that they're specifically designed to work with Android Studio's design editor. Unlike linear and form layouts where you usually make changes to XML, you build constraint layouts visually. You drag and drop GUI components onto the design editor's blueprint tool, and give it instructions for how each view should be displayed.

Linear Layout

In Android, LinearLayout arranges its children in a straight line, either horizontally or vertically. If it is vertically, they're displayed in a single column, and if it's horizontally, they're displayed in a single row.

Material Design

Google introduced Material Design in 2014 to help standardize the appearance of web and mobile applications across the Google ecosystem. Material Design continues to evolve and is available for iOS, Android, and several web frameworks—however, applying its principles is entirely optional.

Layout Inflation Process

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

During project creation, Android Studio associates the activity_main.xml layout file with the MainActivity object. The two are associated not by name, but by the setContentView method found in the Activity class. This complex method accepts a layout file resource identifier, generates the Views designed within, and presents them on screen — a process known as Layout Inflation.

Android FindViewById

```
public class MainActivity extends AppCompatActivity

  setContentView R layout activity_main
  findViewById R id incl_cardview_contact card_main_1
```

setContentView converts all the XML components in our layout into Java View objects, so after this method returns, we can access any View object from our layout programmatically. To find a specific View object, we invoke the findViewById() method. This method returns an object that inherits from the View object, or View itself.