Android Studio

Learning Objectives

After studying this module learner should be able to:

- Know about android studio
- List system requirements for android studio
- to download and install android studio
- List features of Android Studio
- Understand App Workflow
- Define Android Virtual Devices (AVD)
- Use Android Studio IDE Components

What is Android Studio?

For developing application for android platform, you will require Integrated Development Environment (IDE). Android Studio is the official IDE for Android application development. Android Studio provides everything you need to start developing apps for Android, including the Android Studio IDE and the Android SDK tools. First, we discuss what the system requirements for android studio are and how to install and configure android studio.

System Requirements for Android Studio

Windows

- 64-bit Microsoft® Windows® 8/10
- x86_64 CPU architecture; 2nd generation Intel Core or newer, or AMD CPU with support for a Windows Hypervisor
- 8 GB RAM or more
- 8 GB of available disk space minimum (IDE + Android SDK + Android Emulator)
- 1280 x 800 minimum screen resolution

Features Android Studio

Android Studio is Android's official IDE. It is purpose-built for Android to accelerate your development and help you build the highest-quality apps for every Android device.

Code and iterate faster than ever

Based on Intellij IDEA, Android Studio provides the fastest possible turnaround on your coding and running workflow.

Apply Changes

Android Studio's Apply Changes feature lets you push code and resource changes to your running app without restarting your app

Intelligent code editor

The code editor helps you write better code, work faster, and be more productive by offering advanced code completion, refactoring, and code analysis. As you type, Android Studio provides suggestions in a dropdown list. Simply press Tab to insert the code.

Fast and feature-rich emulator

The Android Emulator installs and starts your apps faster than a real device and allows you to prototype and test your app on various Android device configurations: phones, tablets, Android Wear, and Android TV devices. You can also simulate a variety of hardware features such as GPS location, network latency, motion sensors, and multi-touch input.

Code with confidence

At every step, Android Studio helps ensure that you're creating the best code possible.

Code templates and sample apps

Android Studio includes project and code templates that make it easy to add well-established patterns such as a navigation drawer and view pager. You can start

with a code template or even right-click an API in the editor and select Find Sample Code to search for examples. Moreover, you can import fully functional apps from GitHub, right from the Create Project screen.

Lintelligence

Android Studio provides a robust static analysis framework and includes over 365 different lint checks across the entirety of your app. Additionally, it provides several quick fixes that help you address issues in various categories, such as performance, security, and correctness, with a single click.

Testing tools and frameworks

Android Studio provides extensive tools to help you test your Android apps with JUnit 4 and functional UI test frameworks. With Espresso Test Recorder, you can generate UI test code by recording your interactions with the app on a device or emulator. You can run your tests on a device, an emulator, a continuous integration environment, or in Firebase Test Lab.

Configure builds without limits

Android Studio's project structure and Gradle-based builds provide the flexibility you need to generate APKs for all device types.

Optimized for all Android devices

Android Studio provides a unified environment where you can build apps for Android phones, tablets, Android Wear, Android TV, and Android Auto.

Create rich and connected apps

Android Studio knows not all code is written in Java and not all code runs on the user's device.

C++ and NDK support

Android Studio fully supports editing C/C++ project files so you can quickly build JNI components in your app. The IDE provides syntax highlighting and refactoring for C/C++, and an LLDB-based debugger that allows you to simultaneously debug

your Java and C/C++ code. The build tools can also execute your CMake and ndk-build scripts without any modification and then add the shared objects to your APK.

Firebase and Cloud integration

The Firebase Assistant helps you connect your app to Firebase and add services such as Analytics, Authentication, Notifications and more with step-by-step procedures right inside Android Studio. Built-in tools for Google Cloud Platform also help you integrate your Android app with services such as Google Cloud Endpoints and project modules specially-designed for Google App Engine.

Eliminate tiresome tasks

Android Studio provides GUI tools that simplify the less interesting parts of app development.

Layout Editor

When working with XML layout files, Android Studio provides a drag-and-drop visual editor that makes it easier than ever to create a new layout. The Layout Editor was built in unison with the ConstraintLayout API, so you can quickly build a layout that adapts to different screen sizes by dragging views into place and then adding layout constraints with just a few clicks.

APK Analyzer

You can use the APK Analyzer to easily inspect the contents of your APK. It reveals the size of each component so you can identify ways to reduce the overall APK size. It also allows you to preview packaged assets, inspect the DEX files to troubleshoot multidex issues, and compare the differences between two APKs.

Vector Asset Studio

Android Studio makes it easy to create a new image asset for every density size. With Vector Asset Studio, you can select from Google-provided material design icons or import an SVG or PSD file. Vector Asset Studio can also generate bitmap files for each screen density to support older versions of Android that don't support the Android vector drawable format.

Translations Editor

The Translations Editor gives you a single view of all of your translated resources, making it easy to change or add translations, and to find missing translations without opening each version of the strings.xml file. It even provides a link to order translation services.

Installing Android Studio

https://developer.android.com/studio/videos/studio-install-windows.mp4

Developer workflow basics

The workflow to develop an app for Android is conceptually the same as other app platforms. However, to efficiently build a well-designed app for Android, you need some specialized tools. The following list provides an overview of the process to build an Android app and includes links to some Android Studio tools you should use during each phase of development.

Set up your workspace

This is the phase you probably already finished: Install Android Studio and create a project.

Write your app

Now you can get to work. Android Studio includes a variety of tools and intelligence to help you work faster, write quality code, design a UI, and create resources for different device types.

Build and run

During this phase, you build your project into a debuggable APK package that you can install and run on the emulator or an Android-powered device.

You can also begin customizing your build. For example, you can create build variants that produce different types of APKs from the same project, and shrink your code and resources to make your APK file smaller.

Debug, profile, and test

This is the iterative phase in which you continue writing your app but with a focus on eliminating bugs and optimizing app performance. Of course, creating tests will help you in those endeavors.

Publish

When you're ready to release your app to users, there are just a few more things to consider, such as versioning your app and signing it with a key.

Android Studio IDE Components

1. The toolbar lets you carry out a wide range of actions, including running your app and launching Android tools.

2. The navigation bar helps you navigate through your project and open files for editing. It provides a more compact view of the structure visible in the Project window.

3. The editor window is where you create and modify code. Depending on the current file type, the editor can change. For example, when viewing a layout file, the editor displays the Layout Editor.

4. The tool window bar runs around the outside of the IDE window and contains the buttons that allow you to expand or collapse individual tool windows.

5. The tool windows give you access to specific tasks like project management, search, version control, and more. You can expand them and collapse them.

6. The status bar displays the status of your project and the IDE itself, as well as any warnings or messages.

You can organize the main window to give yourself more screen space by hiding or moving toolbars and tool windows. You can also use keyboard shortcuts to access most IDE features.

At any time, you can search across your source code, databases, actions, elements of the user interface, and so on, by double-pressing the Shift key, or clicking the magnifying glass in the upper right-hand corner of the Android Studio window. This can be very useful if, for example, you are trying to locate a particular IDE action that you have forgotten how to trigger.

Android Virtual Devices (AVD)

An Android Virtual Device (AVD) is an emulator configuration that lets you model an actual device by defining hardware and software options to be emulated by the Android Emulator. An AVD consists of:

A hardware profile: Defines the hardware features of the virtual device. For example, you can define whether the device has a camera, whether it uses a physical QWERTY keyboard or a dialing pad, how much memory it has, and so on.

A mapping to a system image: You can define what version of the Android platform will run on the virtual device. You can choose a version of the standard Android platform or the system image packaged with an SDK add-on.

Other options: You can specify the emulator skin you want to use with the AVD, which lets you control the screen dimensions, appearance, and so on. You can also specify the emulated SD card to use with the AVD.

A dedicated storage area on your development machine: the device's user data (installed applications, settings, and so on) and emulated SD card are stored in this area. The easiest way to create an AVD is to use the graphical AVD Manager. You can also start the AVD Manager from the command line by calling the android tool with the avd options, from the <sdk>/tools/ directory.

You can also create AVDs on the command line by passing the android tool options.

You can create as many AVDs as you need, based on the types of device you want to model. To thoroughly test your application, you should create an AVD for each

general device configuration (for example, different screen sizes and platform versions) with which your application is compatible and test your application on each one.

Keep these points in mind when you are selecting a system image target for your AVD:

- The API Level of the target is important, because your application will not be able to run on a system image whose API Level is less than that required by your application, as specified in the minSdkVersion attribute of the application's manifest file.
- You should create at least one AVD that uses a target whose API Level is greater than that required by your application, because it allows you to test the forward-compatibility of your application. Forward-compatibility testing ensures that, when users who have downloaded your application receive a system update, your application will continue to function normally.
- If your application declares a uses-library element in its manifest file, the application can only run on a system image in which that external library is present. If you want to run your application on an emulator, create an AVD that includes the required library. Usually, you must create such an AVD using an Add-on component for the AVD's platform.

Using Hardware Device to test Application

When building a mobile application, it's important that you always test your application on a real device before releasing it to users.

You can use any Android-powered device as an environment for running, debugging, and testing your applications. The tools included in the SDK make it easy to install and run your application on the device each time you compile. You can install your application on the device directly from Android Studio or from the command line with ADB.

In this module we have learned in details about android studio, system requirements for android studio, how to download and install android studio, steps for downloading and installing android studio, features of Android Studio, Understand App Workflow and Android Virtual Devices (AVD)

Further Reading

- https://developer.android.com/studio
- https://developer.android.com/studio/install

Acknowledgement: “The content in this module is modifications based on work created and shared by the Android Open-Source Project and used according to terms described in the Creative Commons 2.5 Attribution License.”