Unit – 2 Android Fundamentals

A) Android Development Tools

Android development tools are essential for creating Android applications. Google provides a comprehensive set of tools and resources for developers to build, test, and debug Android apps. Here are some of the key tools commonly used in Android development:

Eclipse:

- **Description:** Eclipse was historically a popular integrated development environment (IDE) for Java-based applications, including Android development. However, Google officially moved away from supporting Eclipse in favor of Android Studio, which is now the recommended IDE.

Kony:

- **Description:** Kony is a cross-platform mobile app development platform that allows developers to create applications for various platforms, including Android. It provides tools for designing, developing, and deploying mobile apps across different devices and operating systems.

Xamarin:

- **Description:** Xamarin is a cross-platform app development framework that allows developers to build native Android, iOS, and Windows apps using a single codebase written in C#. Xamarin uses a single, shared codebase for the core logic, and platform-specific code can be added as needed.

Android Studio:

- **Description:** Android Studio is the official IDE for Android app development, provided by Google. It is based on IntelliJ IDEA and offers a feature-rich environment for designing, coding, testing, and debugging Android applications. Android Studio is widely adopted and supported by the Android development community.

- Android Studio was announced on May 16, 2013, at the Google I/O conference. It was released in June 2014. The first stable build was released in December 2014, starting from version 1.0.

While Eclipse, Kony, and Xamarin were once popular in Android development, Android Studio has become the standard and recommended IDE by Google. Developers often choose Android Studio for its integration with official Android development tools, continuous updates, and strong community support. It's important to note that technology landscapes can change, so it's always a good idea to check for the latest information and recommendations from official sources.

B) Setting up Android Development Environment o Configuration of Android Environment.

Installation Steps:

1. **Download Android Studio:**

Visit the official Android Studio download page.

Choose the appropriate version for your operating system (Windows, macOS, Linux).

Click the "Download" button to download the installer.

2. **Run the Installer:**

Execute the installer file you downloaded.

Follow the on-screen instructions to complete the installation.

On Windows, you might need to confirm the installation of additional

components and accept the license agreements.

3. **Configure Android Studio:**

Upon the first launch, Android Studio will prompt you to install the

Android SDK components and set up a virtual device.

Follow the setup wizard to install the required components.

4. **Install the Android SDK Components:**

Android Studio will download and install the necessary SDK

components. This may take some time depending on your internet connection.

5. **Set Up a Virtual Device (Emulator):**

If you plan to test your apps on an emulator, the setup wizard will guide you through creating a virtual device.

Choose a hardware profile for the virtual device, and download the

system image for the selected Android version.

6. **Complete the Installation:**

Once the setup is complete, Android Studio will open, and you will be

presented with the welcome screen.

7. **Configure SDK and Emulator:**

Go to "Configure" > "SDK Manager" to manage Android SDK versions and install additional components.

Use the "AVD Manager" to create, manage, and launch virtual devices.

8. **Update Android Studio (if necessary):**

Periodically check for updates within Android Studio by going to "Help" > "Check for Updates."

C) Exploring the Android SDK

The Android SDK (Software Development Kit) is a software development kit that includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator etc.

Android SDK Components play a major role in the Development of Android applications. Below are the important components:

1) Android SDK Tools

Android SDK tool is an important component of Android SDK. It consists of a complete set of development and debugging tools. Below are the SDK developer tools:

- Android SDK Build tool.
- Android Emulator.
- Android SDK Platform-tools.

2) Android SDK Build-Tools

Android SDK build tools are used for building actual binaries of Android App. The main functions of Android SDK Build tools are built, debug, run and test Android applications. The latest version of the Android SDK Build tool is 30.0.3. While downloading or updating Android in our System, one must ensure that its latest version is download in SDK Components.

3) Android Emulator

An Android Emulator is a device that simulates an Android device on your system. Suppose we want to run our android application that we code. One option is that we will run this on our Android Mobile by Enabling USB Debugging on our mobile. Another option is using Android Emulator. In Android Emulator the virtual android device is shown on our system on which we run the Android application that we code.

Thus, it simply means that without needing any physical device Android SDK component “Android Emulator” provides a virtual device on the System where we run our Application. The emulator’s come with the configuration for Various android phones, tablets, Wear OS, and Android TV devices.

In Android Virtual Emulator all functions that are feasible on real Android mobile is works on virtual Device like:

- phone calls, text messages.
- stimulate different network speeds.
- specify the location of a device

- access on google play store and lot’s more.

But there is one disadvantage of this emulator is that. It is very slow when System’s PC has less RAM. It works fine when a maximum GB of RAM is present on our device.

4) Android SDK Platform-tools

Android SDK Platform-tools is helpful when we are working on Project and they will show the error messages at the same time. It is specifically used for testing. It includes:

- Android Debug Bridge (ADB), is a command-line tool that helps to communicate with the device. It allows us to perform an action such as Installing App and Debugging App etc.
- Fast boot allows you to flash a device with a new system image.
- Systrace tools help to collect and inspect timing information. It is very crucial for App Debugging.

D) First Android Application.

1. Creating an Android Virtual Device (AVD).

An Android Virtual Device (AVD) is a configuration that defines the characteristics of an Android phone, tablet, Wear OS, Android TV, or Automotive OS device that you want to simulate in the Android Emulator. It contains a hardware profile, system image, storage area, skin, and other properties.The Device Manager is a tool you can launch from Android Studio that helps you create and manage AVDs.

To create virtual device got to the Android Studio Welcome screen, select **More Actions > Virtual Device Manager**. **OR** After opening a project, select **View > Tool Windows > Device Manager** from the main menu bar, and then click **Create device**.

Hardware profile

The hardware profile defines the characteristics of a device as shipped from the factory. The Device Manager comes pre-loaded with certain hardware profiles, such as Pixel devices, and you can define or customize the hardware profiles as needed.

System images

A system image labeled with **Google APIs** includes access to Google Play services.

Storage area

The AVD has a dedicated storage area on your development machine. It stores the device user data, such as installed apps and settings, as well as an emulated SD card. If needed, you can use the Device Manager to wipe user data so the device has the same data as if it were new.

Skin

An emulator skin specifies the appearance of a device. The Device Manager provides some predefined skins. You can also define your own or use skins provided by third parties.

AVD and app features

Make sure your AVD definition includes the device features your app depends on. See the sections about hardware profile properties and AVD properties for lists of features you can define in your AVDs.

2. Creating and Configuring a New Android Project

Step 1: Create a new project

✧ In Android Studio create your first project with basic activity and java language and name it My First Application.

Step 2: Explore the project structure and layout

Based on you selecting the **Basic Activity** template for your project, Android Studio has set up a number of files for you. You can look at the hierarchy of the files for your app in multiple ways, one is in **Project** view. **Project** view shows your files and folders structured in a way that is convenient for working with an Android project. (This does not always match the file hierarchy! To see the file hierarchy, choose the **Project files** view by clicking.

- ✧ Double-click the **app** folder to expand the hierarchy of app files. If you click **Project**, you can hide or show the **Project** view. You might need to select **View > Tool Windows** to see this option.
- ✧ The current **Project** view selection is **Project > Android**.

In the **Project > Android** view you see three or four top-level folders below your **app** folder: **manifests**, **java**, **java (generated)** and **res**. You may not see **java (generated)** right away.

Expand the manifests folder.

This folder contains AndroidManifest.xml**.** This file describes all the components of your Android app and is read by the Android runtime system when your app is executed.

Expand the java folder.

All your Java language files are organized here. The **java** folder contains three subfolders.

com.example.myfirstapp: This folder contains the Java source code files for your app.

com.example.myfirstapp (androidTest): This folder is where you would put your instrumented tests, which are tests that run on an Android device. It starts out with a skeleton test file.

com.example.myfirstapp (test): This folder is where you would put your unit tests. Unit tests don't need an Android device to run. It starts out with a skeleton unit test file. 3. Expand the **res** folder. This folder contains all the resources for your app, including images, layout files, strings, icons, and styling. It includes these subfolders:

layout: This folder contains the UI layout files for your activities. Currently, your app has one activity that has a layout file called `activity_main.xml`. It also contains `content_main.xml`, `fragment_first.xml`, and `fragment_second.xml`.

Step 4: Create a virtual device (emulator)

In this task, you will use the Android Virtual Device (AVD) manager to create a virtual device (or emulator) that simulates the configuration for a particular type of Android device.

The first step is to create a configuration that describes the virtual device.

✧ In Android Studio, select **Tools** > **AVD Manager**, or click the AVD Manager icon in the toolbar.

✧ Click **+Create Virtual Device**. (If you have created a virtual device before, the window shows all of your existing devices and the **+Create Virtual Device** button is at the bottom.) The **Select Hardware** window shows a list of pre-configured hardware device definitions.

✧ Choose a device definition, such as **Pixel 2**, and click **Next**. (For this codelab, it really doesn't matter which device definition you pick).

✧ In the **System Image** dialog, from the **Recommended** tab, choose the latest release. (This does matter.)

✧ If a **Download** link is visible next to a latest release, it is not installed yet, and you need to download it first. If necessary, click the link to start the download, and click **Next** when it's done. This may take a while depending on your connection speed.

✧ In the next dialog box, accept the defaults, and click **Finish**. The AVD Manager now shows the virtual device you added.

✧ If the **Your Virtual Devices** AVD Manager window is still open, go ahead and close it.

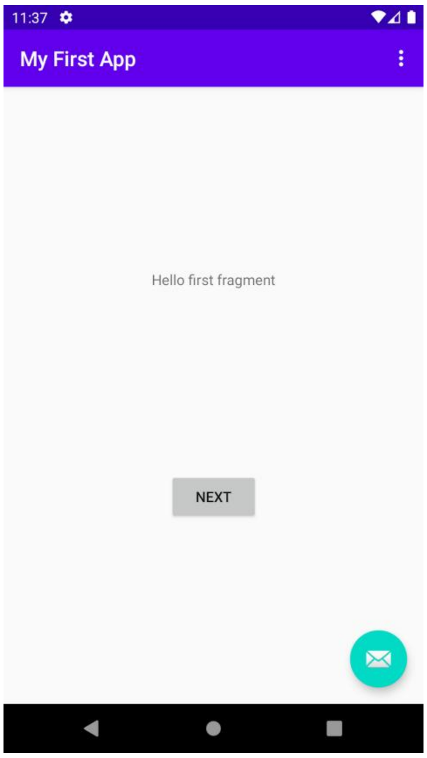
Step 5: Run your app on your new emulator

✧ In Android Studio, select **Run > Run 'app'** or click the **Run** icon in the toolbar. ▶ The icon will change when your app is already running.

✧ In **Run > Select Device**, under **Available devices**, select the virtual device that you just configured. This menu also appears in the toolbar.

The emulator starts and boots just like a physical device. Depending on the speed of your computer, this may take a while. You can look in the small horizontal status bar at the very bottom of Android Studio for messages to see the progress.

Once your app builds and the emulator is ready, Android Studio uploads the app to the emulator and runs it. You should see your app as shown in the following screenshot.



3. Core Files and Directories of the Android Application.

It is very important to know about the basics of Android Studio’s file structure. In this topic, some important files/folders, and their significance is explained for the easy understanding of the Android studio work environment.

AndroidManifest.xml: Every project in Android includes a manifest file, which is AndroidManifest.xml, stored in the root directory of its project hierarchy. The manifest file is an important part of our app because it defines the structure and metadata of our application, its components, and its requirements.

This file includes nodes for each of the Activities, Services, Content Providers and Broadcast Receiver that make the application and using Intent Filters and Permissions, determines how they co-ordinate with each other and other applications.

Java: The Java folder contains the Java source code files. These files are used as a controller for controlled UI (Layout file). It gets the data from the Layout file and after processing that data output will be shown in the UI layout. It works on the backend of an Android application.

drawable: A Drawable folder contains resource type file (something that can be drawn). Drawables may take a variety of file like Bitmap (PNG, JPEG), Nine Patch, Vector (XML), Shape, Layers, States, Levels, and Scale.

layout: A layout defines the visual structure for a user interface, such as the UI for an Android application. This folder stores Layout files that are written in XML language. You can add additional layout objects or widgets as child elements to gradually build a View hierarchy that defines your layout file.
Below is a sample layout file:

mipmap: Mipmap folder contains the Image Asset file that can be used in Android Studio application. You can generate the following icon types like Launcher icons, Action bar and tab icons, and Notification icons.

colors.xml: colors.xml file contains color resources of the Android application. Different color values are identified by a unique name that can be used in the Android application program.

strings.xml: The strings.xml file contains string resources of the Android application. The different string value is identified by a unique name that can be used in the Android application program. This file also stores string array by using XML language.

styles.xml: The styles.xml file contains resources of the theme style in the Android application. This file is written in XML language.

Below is a sample styles.xml file:

build.gradle(Module: app): This defines the module-specific build configurations. Here you can add dependencies what you need in your Android application. Located in the *project/module* directory of the project this Gradle script is where all the dependencies are defined and where the SDK versions are declared. This script has many functions in the project which include additional build types and override settings in the main/app manifest or *top-level build.gradle* file.

4. Crating a Launch Configuration for new Project