

Android - Environment Setup

1. Write two main sophisticated technologies for developing Android application and what are the requirements (hardware and software) to set-up Android studio.

Sophisticated technologies:

There are so many sophisticated Technologies are available to develop android applications, the familiar technologies, which are predominantly using tools as follows:

- Android Studio
- Eclipse IDE (Deprecated)

Requirements software:

We can start our Android application development on either of the following operating systems –

- Microsoft® Windows® 10/8/7/Vista/2003 (32 or 64-bit)
- Mac® OS X® 10.8.5 or higher, up to 10.9 (Mavericks)
- GNOME or KDE desktop

All the required tools to develop Android applications are open source and can be downloaded from the Web. Following is the list of software's we will need before we start our Android application programming.

- Java JDK5 or later version
- Java Runtime Environment (JRE) 6
- Android Studio

Requirements hardware:

The absolute **minimum** requirements for Android were originally a 200 MHz **processor**, 32 MB of **RAM**, and 32 MB of **storage**. Out of the box, Android is incompatible with ARMv4 or lower; ARMv5 or higher is needed to run native code without modifications. **Android 4.4+** requires an ARMv7 **processor**.

2. How to set-up Java Development Kit (JDK) for Android?

We can download the latest version of Java JDK from Oracle's Java site – Java SE Downloads. We will find instructions for installing JDK in downloaded files, follow the given instructions to install and configure the setup. Finally set PATH and

JAVA_HOME environment variables to refer to the directory that contains **java** and **javac**, typically `java_install_dir/bin` and `java_install_dir` respectively.

If we are running Windows and installed the JDK in `C:\jdk1.8.0_102`, we would have to put the following line in our `C:\autoexec.bat` file.

```
set PATH=C:\jdk1.8.0_102\bin;%PATH%
```

```
set JAVA_HOME=C:\jdk1.8.0_102
```

Alternatively, we could also right-click on *My Computer*, select *Properties*, then *Advanced*, then *Environment Variables*. Then, we would update the PATH value and press the OK button.

On Linux, if the SDK is installed in `/usr/local/jdk1.8.0_102` and we use the C shell, we would put the following code into our `.cshrc` file.

```
setenv PATH /usr/local/jdk1.8.0_102/bin:$PATH
```

```
setenv JAVA_HOME /usr/local/jdk1.8.0_102
```

Alternatively, if we use Android studio, then it will know automatically where we have installed our Java.

3. How to set-up Android SDK (Software Development Kit) for Android?

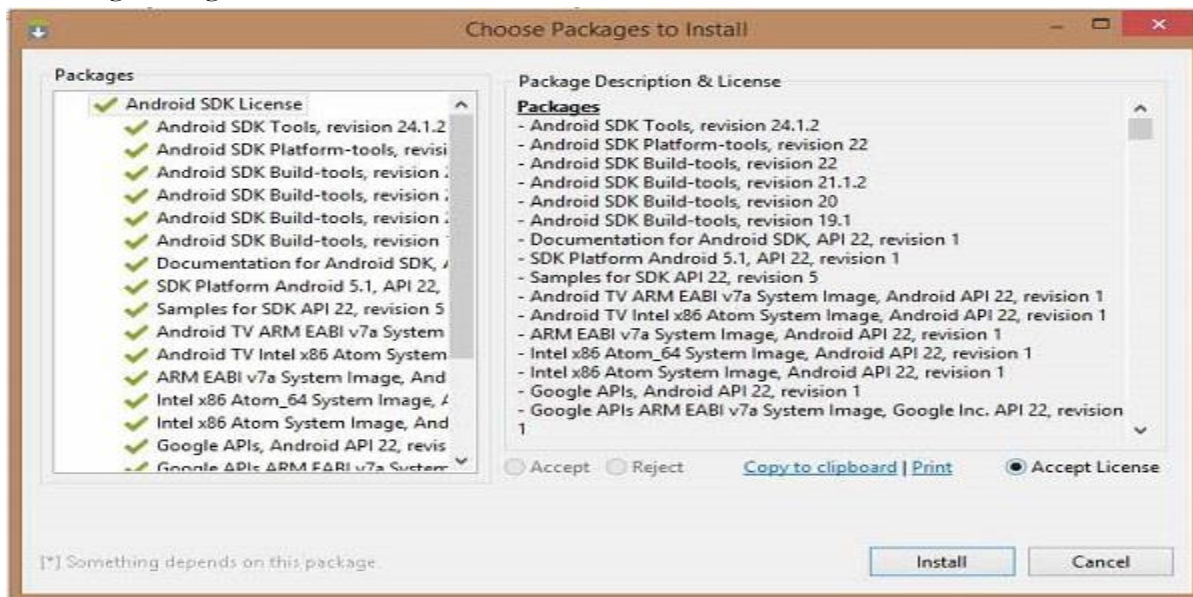
We can download the latest version of Android SDK from Android official website – [Android SDK Downloads](#). If we are installing SDK on Windows machine, then we will find a *installer_rXX-windows.exe*, so just download and run this exe which will launch *Android SDK Tool Set up* wizard to guide we throughout of the installation, so just follow the instructions carefully. Finally we will have *Android SDK Tools* installed on our machine.

If we are installing SDK either on Mac OS or Linux, check the instructions provided along with the downloaded *android-sdk_rXX-macosx.zip* file for Mac OS and *android-sdk_rXX-linux.tgz* file for Linux. Consider that we are going to set up our environment on Windows machine having Windows 7 operating system.

So let's launch *Android SDK Manager* using the option **All Programs > Android SDK Tools > SDK Manager**, this will give we following window –



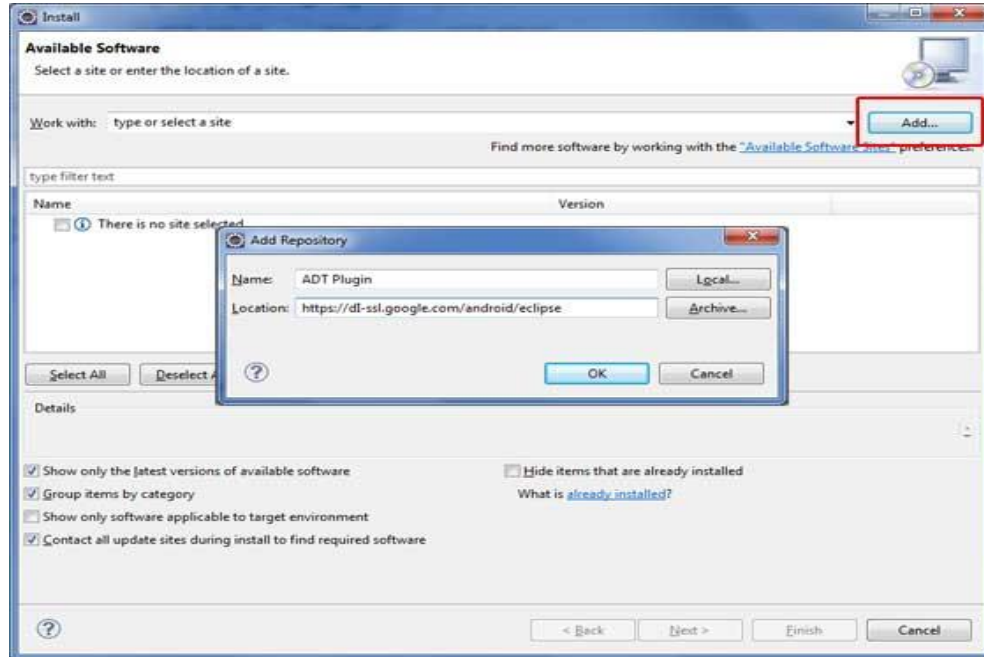
Once we launched SDK manager, its time to install other required packages. By default it will list down total 7 packages to be installed, but suggestion is to de-select *Documentation for Android SDK* and *Samples for SDK* packages to reduce installation time. Next click **Install 7 Packages** button to proceed, which will display following dialogue box –



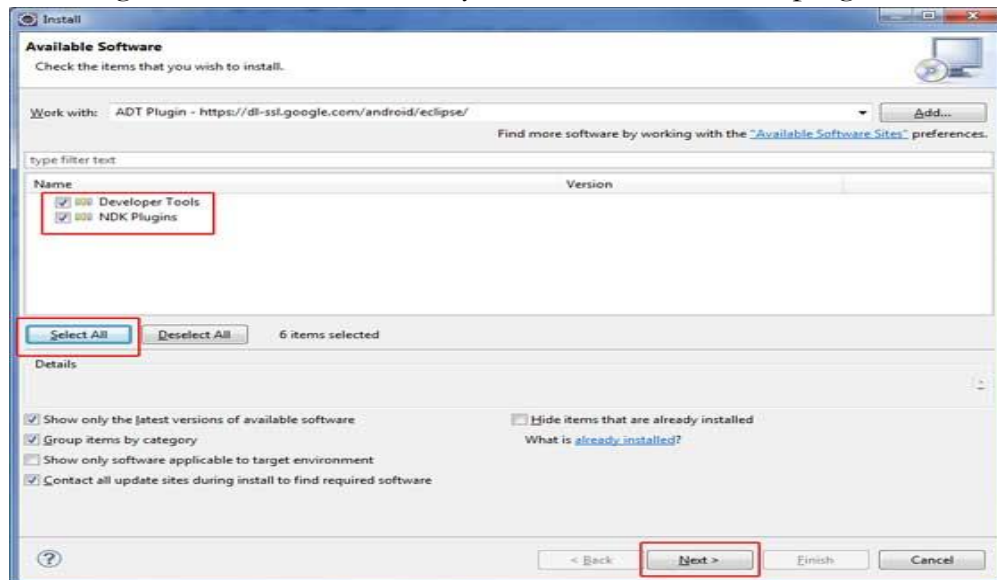
If we agree to install all the packages, select **Accept All** radio button and proceed by clicking **Install** button. Now let SDK manager do its work and we go, pick up a cup of coffee and wait until all the packages are installed. It may take some time depending on our internet connection. Once all the packages are installed, we can close SDK manager using top-right cross button.

4. How to set-up Android Development Tools (ADT) Plug-in for Android?

This step will help us in setting Android Development Tool plug-in for Eclipse. Let's start with launching Eclipse and then, choose **Help > Software Updates > Install New Software**. This will display the following dialogue box.



Now use **Add** button to add *ADT Plug-in* as name and *https://dl-ssl.google.com/android/eclipse/* as the location. Then click OK to add this location, as soon as we will click OK button to add this location, Eclipse starts searching for the plug-in available the given location and finally lists down the found plug-ins.



Now select all the listed plug-ins using **Select All** button and click **Next** button which will guide we ahead to install Android Development Tools and other required plug-ins.

5. How to set-up Eclipse IDE (Integrated Development Environment) for Android?

To install Eclipse IDE, download the latest Eclipse binaries from <https://www.eclipse.org/downloads/>. Once we downloaded the installation, unpack the binary distribution into a convenient location. For example in C:\eclipse on windows, or /usr/local/eclipse on Linux and finally set PATH variable appropriately.

Eclipse can be started by executing the following commands on windows machine, or we can simply double click on eclipse.exe

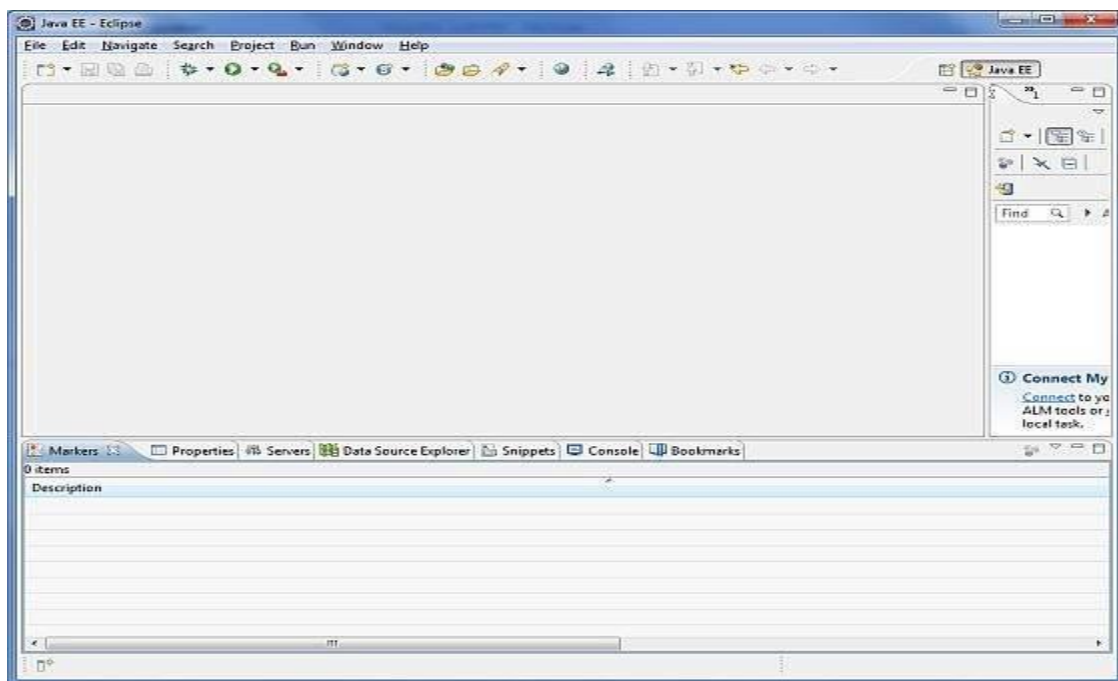
```
%C:\eclipse\eclipse.exe
```

Eclipse can be started by executing the following commands on Linux machine –

```
$/usr/local/eclipse/eclipse
```

After a successful start up, if everything is fine then it should display following result

—



Android –Platform, Feature, Version etc.

6. What is Mobile Platform? Write some examples of good platform.

Platform: Any hardware or software environment in which a program runs, is known as a platform. Since Android has its own runtime environment (JRE) and API, it is called platform.

Mobile platform: The hardware/software environment for laptops, tablets, smartphones and other portable devices. Windows and Mac dominate the laptop world, with Linux a distant third.

9 Best Cross-Platform Mobile App Development Tools:

- PhoneGap. ...
- Appcelerator. ...
- Sencha Touch. ...
- Monocross. ...
- Kony **Mobile Platform**. ...
- NativeScript. ...
- RhoMobile. ...
- Xamarin. Xamarin has made it possible for the **developers** to design native **apps** for multiple **platforms** using only C# code base.

7. What is WAP? Write some features of WAP.

WAP (Wireless Application Protocol) is a specification for a set of communication protocols to standardize the way that wireless devices, such as cellular telephones and radio transceivers, can be used for Internet access, including e-mail, the World Wide Web, newsgroups, and instant messaging.

There are listed some of the key features offered by WAP:

- A programming model similar to the Internet's.
- Reuse the concepts found on the Internet.
- **Wireless** Markup Language (WML)
- WMLScript
- **Wireless** Telephony Application Interface (WTAI)
- Optimized protocol stack.

8. Write some applications of Android.

Android Applications:

Android applications are usually developed in the Java language using the Android Software Development Kit.

Once developed, Android applications can be packaged easily and sold out either through a store such as **Google Play**, **SlideME**, **Opera Mobile Store**, **Mobango**, **F-droid** and the **Amazon Appstore**. Every day more than 1 million new Android devices are activated worldwide (more than 190 countries around the world).

Google Play: Formerly known as the Android Market, is the official app store for Android smartphones and tablets. Google makes software applications, music, movies and books available for purchase and download through the store.

SlideME: A Community & Content Marketplace, uniting developers and users, offers products, services and experience that help promote small Android developers and their creative efforts, without locking them into any closed standards.
















Opera Mobile Store : Offers a large number of applications for Android besides other *mobile* platforms.

MOBANGO: A mobile community enabling mobile users to publish, convert and share user generated content with others.

F-Droid: A software repository (or "app store") for Android applications, similar to the Google Play store.

Amazon Appstore: An app store for the Android operating system operated by **Amazon.com**.

Categories of Android applications

	Music		News		Multimedia
	Sports		Lifestyle		Food & Drink
	Travel		Weather		Books
	Business		Reference		Navigation
	Social Media		Utilities		Finance

9. What's are Android API levels? Make a list of Android version name sequentially with API levels.

What is API level:

API (Application Program Interface) Level is an integer value that uniquely identifies the framework API revision offered by a version of the Android platform.

Android API Levels:

API Levels generally mean that as a programmer, you can communicate with the devices' built in functions and functionality. As the API level increases, functionality adds up (although some of it can get deprecated).

Android Versions and API Levels:

Each Android version is assigned a unique integer identifier, called the *API Level*. Therefore, each Android version corresponds to a single Android API Level. Because users install apps on older as well as the most recent versions of Android, real-world Android apps must be designed to work with multiple Android API levels.

The latest version is Android 8 OREO.

Platform Version	API Level	VERSION_CODE
Android 8.0	26-27	OREO
Android 7.0	24-25	NOUGAT
Android 6.0	23	MARSHMALLOW
Android 5.1	22	LOLLIPOP_MR1
Android 5.0	21	LOLLIPOP
Android 4.4W	20	KITKAT_WATCH
Android 4.4	19	KITKAT

Android 4.3	18	JELLY_BEAN_MR2
Android 4.2, 4.2.2	17	JELLY_BEAN_MR1
Android 4.1, 4.1.1	16	JELLY_BEAN
Android 4.0.3, 4.0.4	15	ICE_CREAM_SANDWICH_MR1
Android 4.0, 4.0.1, 4.0.2	14	ICE_CREAM_SANDWICH
Android 3.2	13	HONEYCOMB_MR2
Android 3.1.x	12	HONEYCOMB_MR1
Android 3.0.x	11	HONEYCOMB
Android 2.3.4 Android 2.3.3	10	GINGERBREAD_MR1
Android 2.3.2 Android 2.3.1 Android 2.3	9	GINGERBREAD
Android 2.2.x	8	FROYO
Android 2.1.x	7	ECLAIR_MR1
Android 2.0.1	6	ECLAIR_0_1
Android 2.0	5	ÉCLAIR
Android 1.6	4	DONUT
Android 1.5	3	CUPCAKE
Android 1.1	2	BASE_1_1
Android 1.0	1	BASE

10. Explain Eclipse IDE and Android Development Tools (ADT) Plug-in with their purpose in Android.

Eclipse IDE:

Eclipse is a free, Java-based development platform known for its plug-ins that allow developers to develop and test code written in other programming languages (multi-language software development platform). It is used to develop applications in Java and, by means of the various plug-ins, in other languages as well - C/C++, Cobol, Python, Perl, PHP.

The Eclipse open source project provides regular releases, currently four releases per year. Every year a larger releases is done, which gets a new release number and a new release name.

As of 2012 the main Eclipse release carried the major version number 4:

<i>Table 1. Eclipse releases</i>		
Release	Rename name	Release year
4.2	Juno	2012
4.3	Kepler	2013
4.4	Luna	2014
4.5	Mars	2015
4.6	Neon	2016
4.7	Oxygen	2017
4.8	Photon	2018

IDE stands for Integrated development environment. It's tool for developing Applications. IDE provides features like:

Syntax Highlighting:

Debugging

Intellisense

source code editor etc.

IDEs are designed to maximize programmer productivity and helps the developer to write less number of codes.

Plug-in:

Plug-in applications are programs that can easily be installed and used as part of our Web browser. Initially, the Netscape browser allowed us to download, install, and define supplementary programs that played sound or motion video or performed other functions. A plug-in application is recognized automatically by the browser and its function is integrated into the main HTML file that is being presented. Among popular plug-ins to download are Adobe's Acrobat, for documentation.

Android Development Tools (ADT) is a plugin for the Eclipse IDE that is designed to give you a powerful, integrated environment in which to build Android applications.

11. What are the possible next Android version names?

Name of next Android 9.0:

Android 9.0 Popcorn (by Google)



Android 9.0 Pastry

Android 9.0 Pasta

Android Pastilla

Android 9.0 Puff

Android 9.0 Pandoro
Android 9.0 Panna Cotta
Android 9.0 Parfait
Android 9.0 Popover
Android 9.0 PanCake
Android 9.0 Peanut Brittle
Android 9.0 Pumpkin Pie
Android 9.0 Popsicle
Android 9.0 Pecan Pie
Android 9.0 Poached Pears
Android 9.0 Praline
Android 9.0 Pastille
Android 9.0 Petit Four
Android 9.0 Pinka
Android Piano
Android Pillow
Android Pilot
Android Pudding
Android Pie

Some of the other suggested Android 9.0 version names are -

- Penuche
- Pignolo
- Pizzelle
- Pecan Pie
- Pavlova
- Profiterole
- Pop Tart

Android P Indian Names

The following popular names have come up from India for naming **Android P** version -

- Android 9.0 Peda
- Android Pista
- Android Pilu

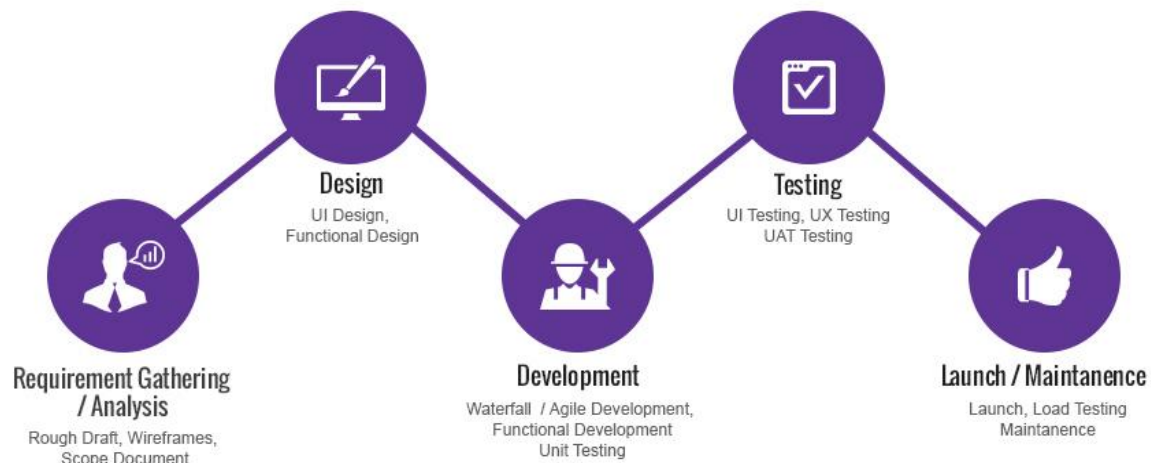
- Android 9.0 Petha
- Android 9.0 Phirni
- Pav or Pav Bhaji (Indian bread pav)

12. Sketch the Block Diagram of Android & IOS Application Development.

Android App Development Process



- ✓ Planning
- ✓ Expert Market Research
- ✓ Critical Analysis
- ✓ App Implementation
- ✓ Testing & Integration
- ✓ Software Release



13. What is android and why it is popular platform? Mention few features of Android operating system.

What is Android:



Android is an open source and Linux-based **Operating System** for mobile devices such as smartphones and tablet computers. Android was developed by the **Open Handset Alliance (OHA)**, led by Google, and other companies(84 firms) include: HTC, Sony, Dell, Intel, Motorola, Qualcomm, Nvidia, Texas Instruments, Samsung, T-Mobile, Electronics, LG Electronics, Sprint Corporation, and Wind River Systems.

The first **beta version** of the Android Software Development Kit (SDK) was released by Google in 2007 where as the first commercial version, Android 1.0, was released in September 2008.

The source code for Android is available under free and open source software licenses. Google publishes most of the code under the **Apache License** (**Apache Software Foundation (ASF)**) version 2.0 and the rest, Linux kernel changes, under the GNU General Public License version 2.

Why Android:



Features of Android

Android is a powerful operating system competing with Apple 4GS and supports great features. Few of them are listed below –

Sr.No.	Feature & Description
1	Beautiful UI Android OS basic screen provides a beautiful and intuitive user interface.
2	Connectivity GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC and Wi-MAX. GSM: Global System for Mobile Communications EDGE: Enhanced Data for GSM Evolution IDEN: Integrated Digital Enhanced Network (Motorola variant of TDMA wireless) TDMA: Time Division Multiple Access CDMS: Code Division Multiple Access EV-DO: Evolution Data Optimized/Only UMTS: Universal Mobile Telecommunications System Bluetooth: Bluetooth is a wireless technology standard for exchanging data over short distances from fixed and mobile devices, and building personal area networks. Wi-Fi: <i>Wireless Fidelity</i> (Fidelity: faithful or loyalty) LTE: Long-Term Evolution is a standard for high-speed wireless communication for mobile phones and data terminals, based on the GSM/EDGE and UMTS/HSPA technologies. HSPA: High Speed Packet Access. NFC: <i>Near Field Communication</i> - is a set of short-range wireless technology. Wi-MAX: Worldwide Interoperability for Microwave Access
3	Storage SQLite, a lightweight relational database, is used for data storage purposes.
4	Media support

	H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, AAC 5.1, MP3, MIDI, OggVorbis, WAV, JPEG, PNG, GIF, and BMP.
5	Messaging SMS and MMS
6	Web browser Based on the open-source WebKit layout engine, coupled with Chrome's V8 JavaScript engine supporting HTML5 and CSS3 (Cascading Style Sheets).
7	Multi-touch Android has native support for multi-touch which was initially made available in handsets such as the HTC (<i>high-tech computer</i>) Hero.
8	Multi-tasking User can jump from one task to another and same time various application can run simultaneously.
9	Resizable widgets Widgets are resizable, so users can expand them to show more content or shrink them to save space.
10	Multi-Language Supports single direction and bi-directional text.
11	GCM Google Cloud Messaging (GCM) is a service that lets developers send short message data to their users on Android devices, without needing a proprietary sync solution.
12	Wi-Fi Direct A technology that lets apps discover and pair directly, over a high-bandwidth peer-to-peer connection.
13	Android Beam A popular NFC (Near Field Communication) -based technology that lets users instantly share, just by touching two NFC-enabled phones together.

RFIR: Radio Frequency Identification

NFC is a branch of High-Frequency (HF) RFID

14. General code meaning

@dimen refers to **dimension** and it's a file where you define dimensions to use them later from in any layout file.

It's located in res/values/dimens , here's what a sample of the file look like:

```
<resources>
  <!-- Default screen margins, per the Android Design guidelines. -->
  <dimen name="activity_horizontal_margin">16dp</dimen>
  <dimen name="activity_vertical_margin">16dp</dimen>
</resources>
```

here activity_veritcal_margin = 16 dp.

and to it's to use it like this:

```
<LinearLayout
  android:id="@+id/activity_main"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:paddingBottom="@dimen/activity_vertical_margin">
```

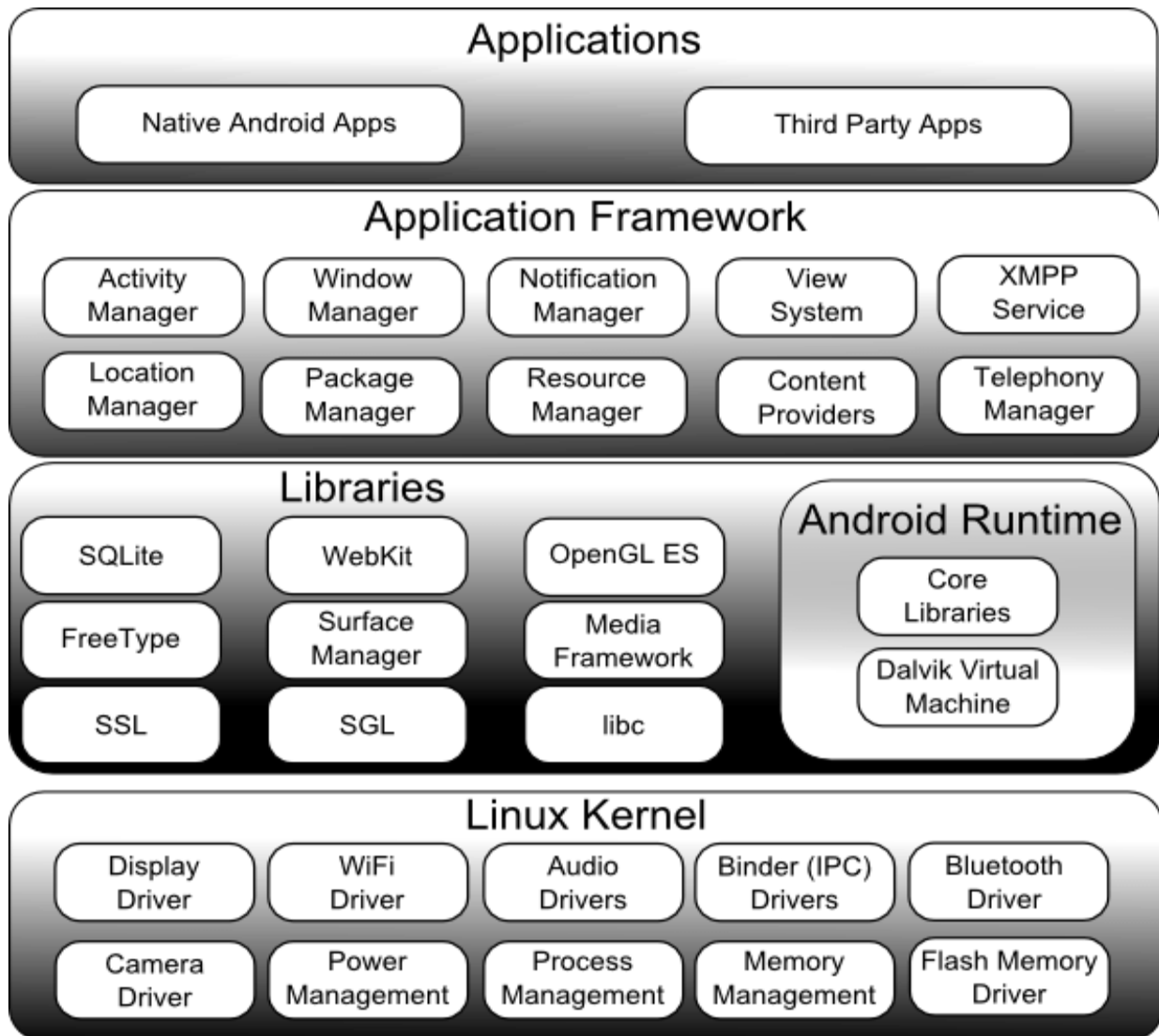
here we give this linear layout a bottom padding with 16dp.

@dimen/activity_vertical_margin or whatever @dimen/whatever_key_name is a reference to a dimension that probably is saved in our projectname/src/main/res/value/dimen.xml file

In android we can save several values for example dimensions, strings, integers, drawables.

Android – Architecture & Components

15. Draw the Architecture of Android with All the Components.



16. How many layers are in Android architecture? Mention the android sections name with few components.

Android operating system is a stack of software components which is roughly divided into:

- Five sections and
- Four main layers

Section names with few components:

1. Applications:

- Home
- Dialer
- Sms
- mms
- web browser
- calculator

2. Application frameworks:

- Unlimited application
- Equality of each apps
- Easy to embed web browser
- Simultaneous running

3. Libraries:

- I. Libc: it is c standard lib.
- II. SSL: Secure Socket Layer for security
- III. SGL: 2D picture engine where SGL is “Scalable Graphics Library”
- IV. OpenGL/ES: 3D image engine
- V. Media Framework: essential part of Android multi-media
- VI. SQLite: Embedded database
- VII. Lib web core: Kernel of web browser
- VIII. Free Type: Bitmap and Vector
- IX. Surface Manager: Manage different windows for different applications

4. Android Run time:

- Core libraries
- Dalvik virtual machine

5. Linux kernel:

- IPC driver

- Display driver
- Camera driver
- Flash memory driver
- Audio driver
- Power management
- Keypad driver
- WIFI driver
- Builder driver

17. Write Down the Definition & Function of:

I. Linux Kernal

At the bottom of the layers is Linux - Linux 3.6 with approximately 115 patches. This provides a level of abstraction between the device hardware and it contains all the essential hardware drivers like camera, keypad, display etc. Also, the kernel handles all the things that Linux is really good at such as networking and a vast array of device drivers, which take the pain out of interfacing to peripheral hardware.

II. Android Runtime

This is the third section of the architecture and available on the second layer from the bottom. This section provides a key component called **Dalvik Virtual Machine** which is a kind of Java Virtual Machine specially designed and optimized for Android.

The Dalvik VM makes use of Linux core features like memory management and multi-threading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine.

The Android runtime also provides a set of **core libraries** which enable Android application developers to write Android applications using standard Java programming language.

III. Java Virtual Machine

The **Java Virtual Machine (JVM)** is the runtime engine of the **Java** Platform, which allows any program written in **Java** or other language compiled into **Java** bytecode to run on any computer that has a native **JVM**.

IV. Dalvik Virtual Machine

Dalvik Virtual Machine (DVM) is a kind of Java Virtual Machine specially designed and optimized for Android.

The Dalvik VM makes use of Linux core features like memory management and multi-threading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine.

V. Android Libraries

On top of Linux kernel there is a set of libraries including open-source Web browser engine WebKit, well known library libc, **SQLite** database which is a useful repository for storage and sharing of application data, libraries to play and record audio and video, SSL libraries responsible for Internet security etc.

SQLite is a [relational database management system](#) contained in a C programming [library](#). In contrast to many other database management systems, SQLite is not a [client-server](#) database engine. Rather, it is embedded into the end program.

Android Libraries encompasses those Java-based libraries that are specific to Android development. Examples of libraries in this category include the application framework libraries in addition to those that facilitate user interface building, graphics drawing and database access.

VI. Application Components

Application components are the essential building blocks of an Android application. These components are loosely coupled by the application manifest file *AndroidManifest.xml* that **describes each component of the application and how they interact.**

There are following four main components that can be used within an Android application –

Sr.No	Components & Description
1	Activities They dictate the UI and handle the user interaction to the smart phone screen.
2	Services They handle background processing associated with an application.
3	Broadcast Receivers They handle communication between Android OS and applications.
4	Content Providers They handle data and database management issues.

VII. Application framework:

The Application Framework layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications.

The Android framework includes the following key services –

- **Activity Manager** – Controls all aspects of the application lifecycle and activity stack.
- **Content Providers** – Allows applications to publish and share data with other applications.
- **Resource Manager** – Provides access to non-code embedded resources such as strings, color settings and user interface layouts.
- **Notifications Manager** – Allows applications to display alerts and notifications to the user.
- **View System** – An extensible set of views used to create application user interfaces.

18. Write Down the Function of Following Android Libraries:

- I. **android.app** – Provides access to the application model and is the cornerstone of all Android applications.
- II. **android.content** – Facilitates content access, publishing and messaging between applications and application components.
- III. **android.database** – Used to access data published by content providers and includes SQLite database management classes.
- IV. **android.opengl** – A Java interface to the OpenGL ES 3D graphics rendering API.
- V. **android.os** – Provides applications with access to standard operating system services including messages, system services and inter-process communication.
- VI. **android.text** – Used to render and manipulate text on a device display.
- VII. **android.view** – The fundamental building blocks of application user interfaces.
- VIII. **android.widget** – A rich collection of pre-built user interface components such as buttons, labels, list views, layout managers, radio buttons etc.
- IX. **android.webkit** – A set of classes intended to allow web-browsing capabilities to be built into applications.

19. Write Short Notes on the Following Application Framework Key Services:

I. Activity Manager

This class gives information about, and interacts with, activities, services, and the containing process.

A number of the methods in this class are for debugging or informational purposes and they should not be used to affect any runtime behavior of your app. These methods are called out as such in the method level documentation.

II. Content Providers

A content provider manages access to a central repository of data. A provider is part of an Android application, which often provides its own UI for working with the data.

III. Resource Manager

If by resource manager you mean applications like Clean master or CCleaner, they help clear out waste Memory and storage which is basically freeing up RAM so that your phone runs better. Some also clean cache which is a part of storing temporary files created by applications.

IV. Notification Manager

Class to notify the user of events that happen. This is how you tell the user that something has happened in the background.

Notifications can take different forms:

- A persistent icon that goes in the status bar and is accessible through the launcher, (when the user selects it, a designated Intent can be launched),
- Turning on or flashing LEDs on the device, or
- Alerting the user by flashing the backlight, playing a sound, or vibrating.

V. View System

This class represents the basic building block for user interface components. A View occupies a rectangular area on the screen and is responsible for drawing and event handling. View is the base class for *widgets*, which are used to create interactive UI components (buttons, text fields, etc.). The [ViewGroup](#) subclass is the base class for *layouts*, which are invisible containers that hold other Views (or other ViewGroups) and define their layout properties.

20. Write Short Notes on Following Application Components:

I. Activities

They dictate the UI and handle the user interaction to the smart phone screen.

An activity represents a single screen with a user interface, in-short Activity performs actions on the screen. For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. If an application has more than one activity, then one of them should be marked as the activity that is presented when the application is launched.

An activity is implemented as a subclass of **Activity** class as follows –

```
public class MainActivity extends Activity {
}
```

II. Services

A service is a component that runs in the background to perform long-running operations. For example, a service might play music in the background while the user is in a different application, or it might fetch data over the network without blocking user interaction with an activity.

A service is implemented as a subclass of **Service** class as follows –

```
public class MyService extends Service {
}
```

III. Broadcast Receivers

Broadcast Receivers simply respond to broadcast messages from other applications or from the system. For example, applications can also initiate broadcasts to let other applications know that some data has been downloaded to the device and is available for them to use, so this is broadcast receiver who will intercept this communication and will initiate appropriate action.

A broadcast receiver is implemented as a subclass of **BroadcastReceiver** class and each message is broadcaster as an **Intent** object.

```
public class MyReceiver extends BroadcastReceiver {
    public void onReceive(context,intent){}
}
```

IV. Content Providers

A content provider component supplies data from one application to others on request. Such requests are handled by the methods of the *ContentResolver* class. The data may be stored in the file system, the database or somewhere else entirely.

A content provider is implemented as a subclass of **ContentProvider** class and must implement a standard set of APIs that enable other applications to perform transactions.

```
public class MyContentProvider extends ContentProvider {
    public void onCreate(){}
}
```

V. Fragments

Represents a portion of user interface in an Activity.

VI. Views

UI elements that are drawn on-screen including buttons, lists forms etc.

VII. Layouts

View hierarchies that control screen format and appearance of the views.

VIII. Intents

Messages wiring components together.

IX. Resources

External elements, such as strings, constants and drawable pictures.

X. Manifest

Configuration file for the application.

Android Frontend (XML UI)

21. Define Android View Group & View.

View:

1. View objects are the basic building blocks of User Interface (UI) elements in Android.
2. View is a simple rectangle box which responds to the user's actions.
3. Examples are EditText, Button, CheckBox etc.
4. View refers to the android.view.View class, which is the base class of all UI classes.

Usually View is used as arguments in methods which act as some kind of listener. For example when we have more than 1 Button in our layout and we set onClick on them, we create a method like this:

```
public void onClick(View view){
```



```
}
```

View Group:

1. **ViewGroup** is the invisible container. It holds **View** and **ViewGroup**
2. For example, **LinearLayout** is the **ViewGroup** that contains **Button** (**View**), and other **Layouts** also.
3. **ViewGroup** is the base class for **Layouts**.
4. The **ViewGroup** is a subclass of **View** and provides invisible container that hold other **Views** or other **ViewGroups** and define their layout properties.

22. Write Short Notes on Different Android Layout Types.

An Android layout is a class that handles arranging the way its children appear on the screen. Anything that is a **View** (or inherits from **View**) can be a child of a layout. All of the layouts inherit from **ViewGroup** (which inherits from **View**). We could also create our own custom layout by making a class that inherits from **ViewGroup**.

The standard **Layouts** are:

- [LinearLayout](#)
- [RelativeLayout](#)
- [FrameLayout](#)
- [AbsoluteLayout](#)
- [TableLayout](#)

1	<u>Linear Layout</u> LinearLayout is a view group that aligns all children in a single direction, vertically or horizontally.
2	<u>Relative Layout</u> RelativeLayout is a view group that displays child views in relative positions.
3	<u>Table Layout</u> TableLayout is a view that groups views into rows and columns.

4	<u>Absolute Layout</u> AbsoluteLayout enables you to specify the exact location of its children.
5	<u>Frame Layout</u> The FrameLayout is a placeholder on screen that you can use to display a single view.
6	<u>List View</u> ListView is a view group that displays a list of scrollable items.
7	<u>Grid View</u> GridView is a ViewGroup that displays items in a two-dimensional, scrollable grid.

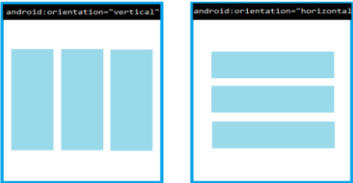
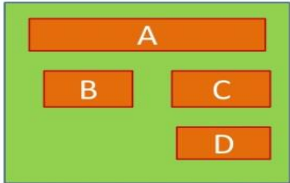
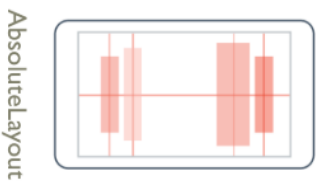
Linear Layout with examples:

Linear layout is a simple layout used in android for layout designing. In the Linear layout all the elements are displayed in linear fashion means all the childs/elements of a linear layout are displayed according to its orientation. The value for orientation property can be either horizontal or vertical.



23. Differentiate Among Linear Layout, Relative Layout & Absolute Layout.

Linear layout	Relative Layout	Absolute Layout
Means you can align views one by one (vertically/horizontally).	Means based on relation of views from its parents and other views.	Is similar to a Relative Layout in that it uses relations to position and size widgets but has additional flexibility and is

		easier to use in the Layout Editor.
Setting all of its paddings to 0.	Enables you to specify the location of child objects relative to each other or to the parent.	If the content doesn't fit, it randomly throws things around.
Arranges its children in a single column or single row.	Displays child views in a relative positions.	specify the exact location of its children.
Can't	Can eliminate nested view groups.	Can't
Can't	Keeps layout hierarchy flat.	Can't
Can't	Several linear nested layout can be replaced by one relative lay out.	Can't
		

See here: <https://mobiforge.com/design-development/understanding-user-interface-android-part-1-layouts>
<https://stackoverflow.com/questions/4905370/what-are-the-differences-between-linearlayout-relativelayout-and-absolutelayout>

24. What are the Differences between Frame Layout & Table Layout?

Frame Layout	Table Layout
The FrameLayout is a placeholder on screen that you can use to display a single view.	TableLayout is a view that groups views into rows and columns.
It is designed to block out an area on the screen to display a single item.	A layout that arranges its children into rows and columns.
	A table Layout consists of a number of table row objects, each define a row.

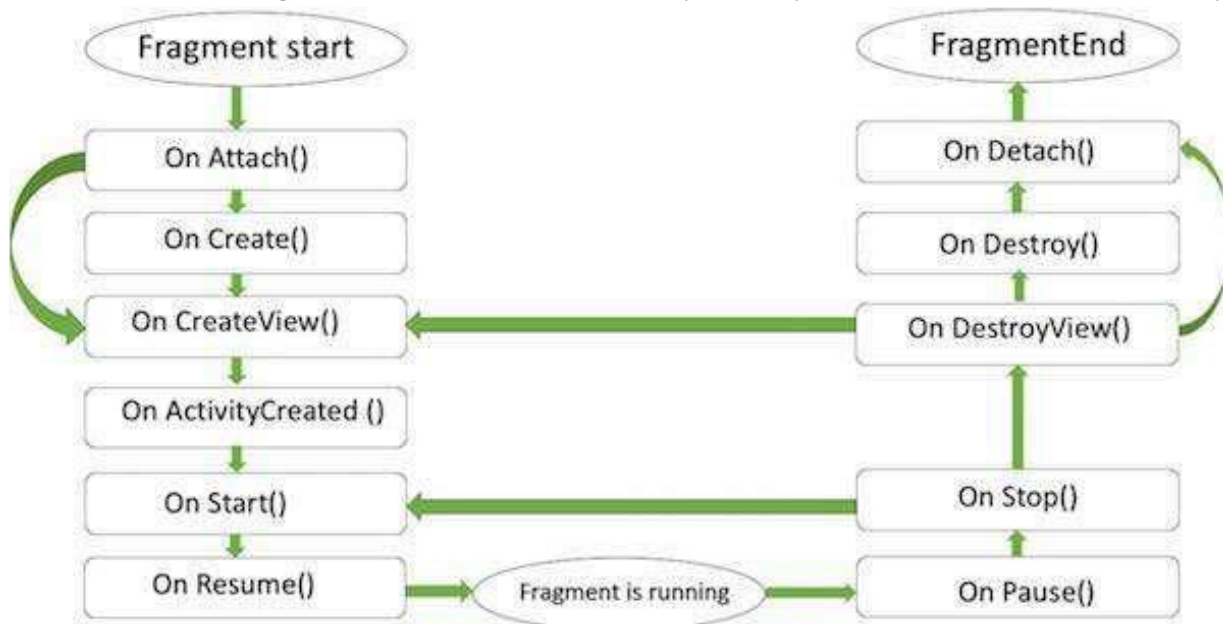
	The width of a column is defined by the row with the widest cell in the column.
Use to hold a single child value	
The size of the frame layout is the size of its largest size.	
Multiple children can be added by assigning gravity to each child.	

25. What is Android Fragment? Write Down the Block Diagram of Fragment Lifecycle.

A **Fragment** is a piece of an activity which enable more modular activity design. It will not be wrong if we say, a fragment is a kind of **sub-activity**.

Fragment Life Cycle:

Android fragments have their own life cycle very similar to an android activity.



26. How to Use Fragments? Describe Different Types of Android Fragments.

There are four types of fragments:

- [ListFragment](#)
- [DialogFragment](#)
- [PreferenceFragment](#)
- [WebViewFragment](#)

ListFragment

This fragment is similar to [ListActivity](#) and contains a [ListView](#) view by default. It is used for displaying a list of items. In our previous sample code, we used [ListFragment](#); see the *Creating and managing fragments* section for [ListFragment](#).

DialogFragment

This fragment displays a dialog on top of its owner activity. In the following sample application, we are going to create a fragment that has a **Delete** button. When the button is clicked, a **DialogFragment** dialog box will be displayed.

27. What is android orientation?

The **screenOrientation** is the attribute of activity element. The orientation of android activity can be portrait, landscape, sensor, unspecified etc. You need to define it in the AndroidManifest.xml file. For example:

1. **<activity**
2. android:name="com.example.screenorientation.MainActivity"
3. android:label="@string/app_name"
4. android:screenOrientation="landscape"
5. **>**

Value	Description
unspecified	It is the default value. In such case, system chooses the orientation.

portrait	taller not wider
landscape	wider not taller
sensor	orientation is determined by the device orientation sensor.

28. Explain AndroidManifest.xml file in detail.



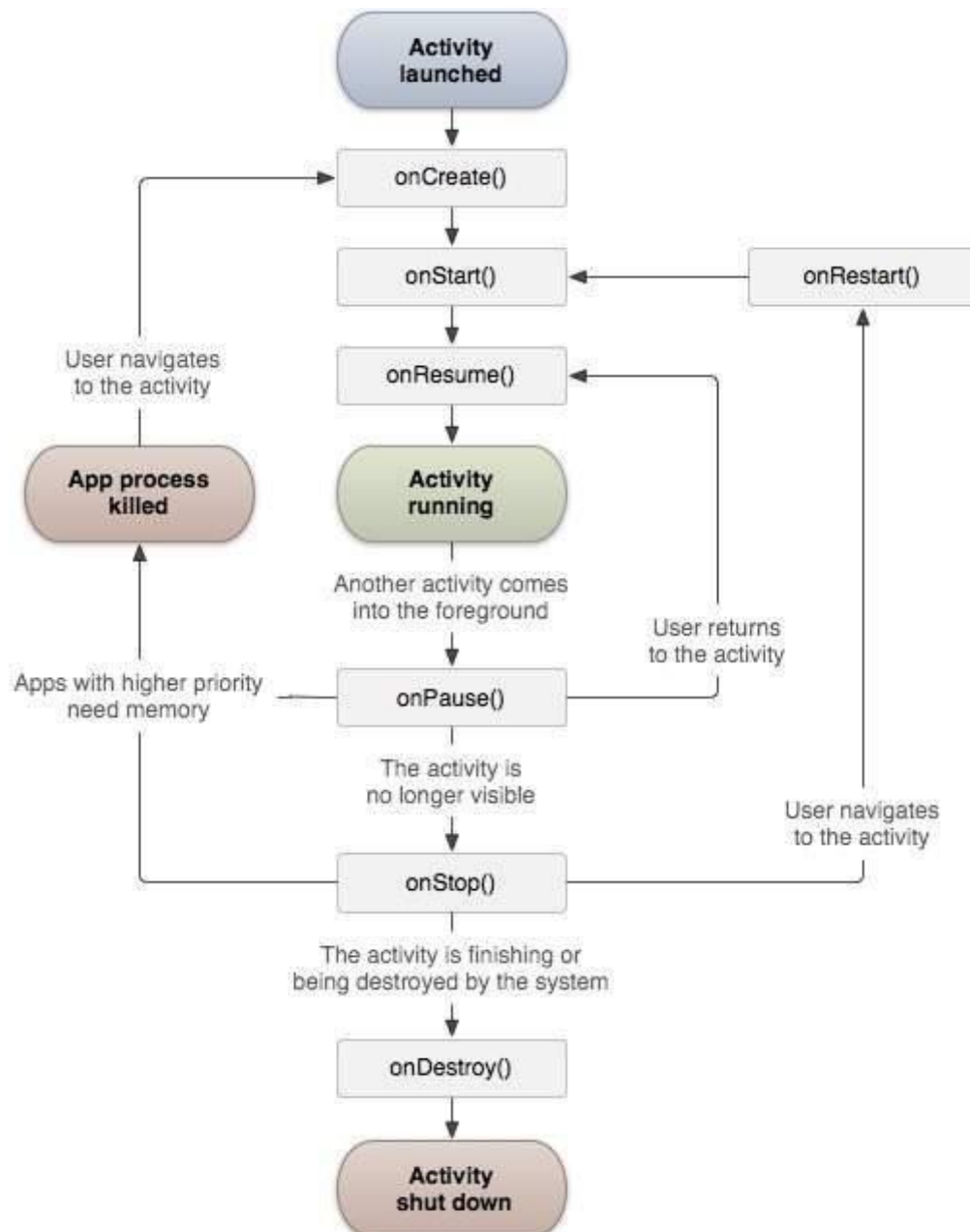
The **AndroidManifest.xml file** *contains information of your package*, including components of the application such as activities, services, broadcast receivers, content providers etc.

Every application must have an AndroidManifest.xml file (with precisely that name) in its root directory. The manifest presents essential information about the application to the Android system, information the system must have before it can run any of the application's code. Among other things, the manifest does the following:

- 1) It is **responsible to protect the application** to access any protected parts by providing the permissions.
- 2) It also **declares the android api** that the application is going to use.
- 3) It **lists the instrumentation classes**. The instrumentation classes provides profiling and other informations. These informations are removed just before the application is published etc.
- 4) This is the **required xml file** for all the android application and located inside the root directory.
- 5) It **names the Java package** for the application. The package name serves as a unique identifier for the application.
- 6) It **describes the components of the application** — the activities, services, broadcast receivers, and content providers that the application is composed of.
- 7) It **names the classes** that implement each of the components and publishes their capabilities (for example, which Intent messages they can handle). These declarations let the Android system know what the components are and under what conditions they can be launched.
- 8) It **determines** which processes will **host application components**.
- 9) It **declares** which **permissions** the application must have in order to **access protected parts** of the API and interact with other applications.
- 10) It also **declares the permissions** that others are required to have in order to **interact** with the application's components.

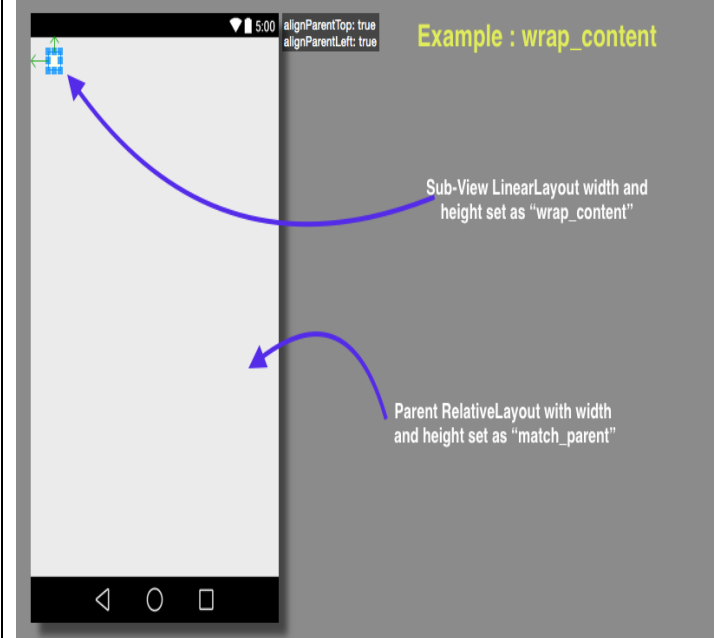
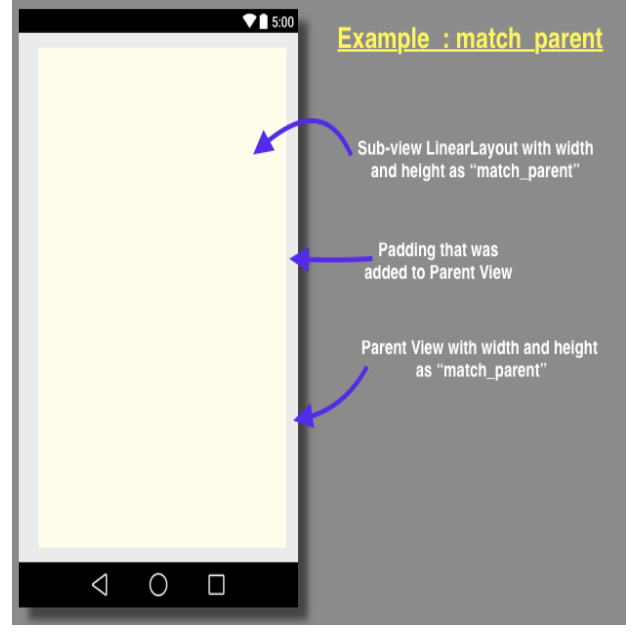
- 11) It **lists the Instrumentation classes** that provide profiling and other information as the application is running. These declarations are present in the manifest only while the application is being developed and tested; they're removed before the application is published.
- 12) It **declares the minimum** level of the Android **API** that the application requires.
- 13) It **lists the libraries** that the application must be linked against.

29. Draw Android activities interface and labeling it.



30. Differentiate the following terms:

i. Wrap content and Match parent

Wrap Content	Match Parent/ Fill Parent
Means that the View wants to be just big enough to enclose its content.	The view should be as big as its parent.
Plus padding.	Minus padding. (If parent is applied a padding then that space would not be included.)
From the beginning named wrap content.	Fill parent is replaced by match parent in API level 8.
Should only big enough to enclose its content.	Setting a top level layout or control to fill parent will force it to take up the whole screen.
 <p>Example : wrap_content</p> <p>Sub-View LinearLayout width and height set as "wrap_content"</p> <p>Parent RelativeLayout with width and height set as "match_parent"</p>	 <p>Example : match_parent</p> <p>Sub-view LinearLayout with width and height as "match_parent"</p> <p>Padding that was added to Parent View</p> <p>Parent View with width and height as "match_parent"</p>

ii. Java and JavaScript.

Java	JavaScript
Java is an OOP programming language.	JavaScript is an OOP scripting language.
Creates applications that run in virtual machine or browser.	Created code contents run on a browser only.
Java codes needs to be compiled.	JavaScript codes are all in text.
Has extension ".java"	Has extension ".js"
Harder than JavaScript.	contains a much smaller and simpler set of commands than does Java .

iii. JDK and IDE


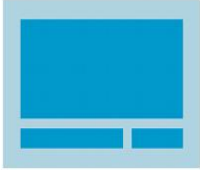


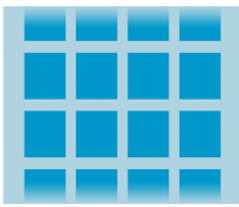
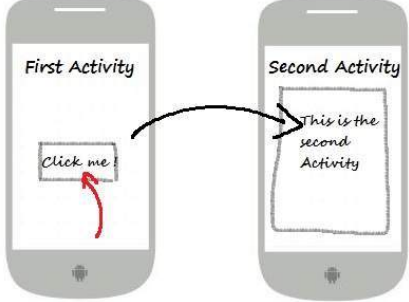
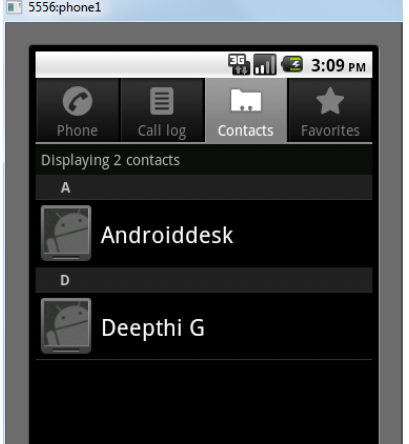
JDK	IDE
Java Development kit.	Integrated Development Environment.
Is a bunch of compiled components that aid you in developing Java programs.	Is a software tool that acts as the communication platform between you, the programmer, and the JDK.
Contains the functions, objects, and other components for you to use in order to develop your own software.	It's the tool through which you can "talk" to the JDK, and use it to write your Java programs.
Includes a complete JRE plus tools for developing, debugging, and monitoring Java applications.	An <i>IDE</i> normally consists of a source code editor, build automation tools, and a debugger. Most modern <i>IDEs</i> have intelligent code completion.
Administrators running applications on a server: Server JRE (Server Java Runtime Environment) For deploying Java applications on servers.	Codes run in IDE.

iv. DVM and JVM

DVM	JVM
DVM is Register-based.	JVM is stack-based.
Dalvik Virtual Machine.	Java Virtual Machine.
Designed to run on low memory	Needs higher memory than DVM.
Uses its own byte code	Uses Java byte code
Runs .dex file	Runs .class file having JIT
More efficient when running multiple instances of the DVM.	Less efficient.
Applications are given their own instance. Hence, multiple active applications require multiple DVM instances.	Many applications use same JVM.

v. Layouts and Intents

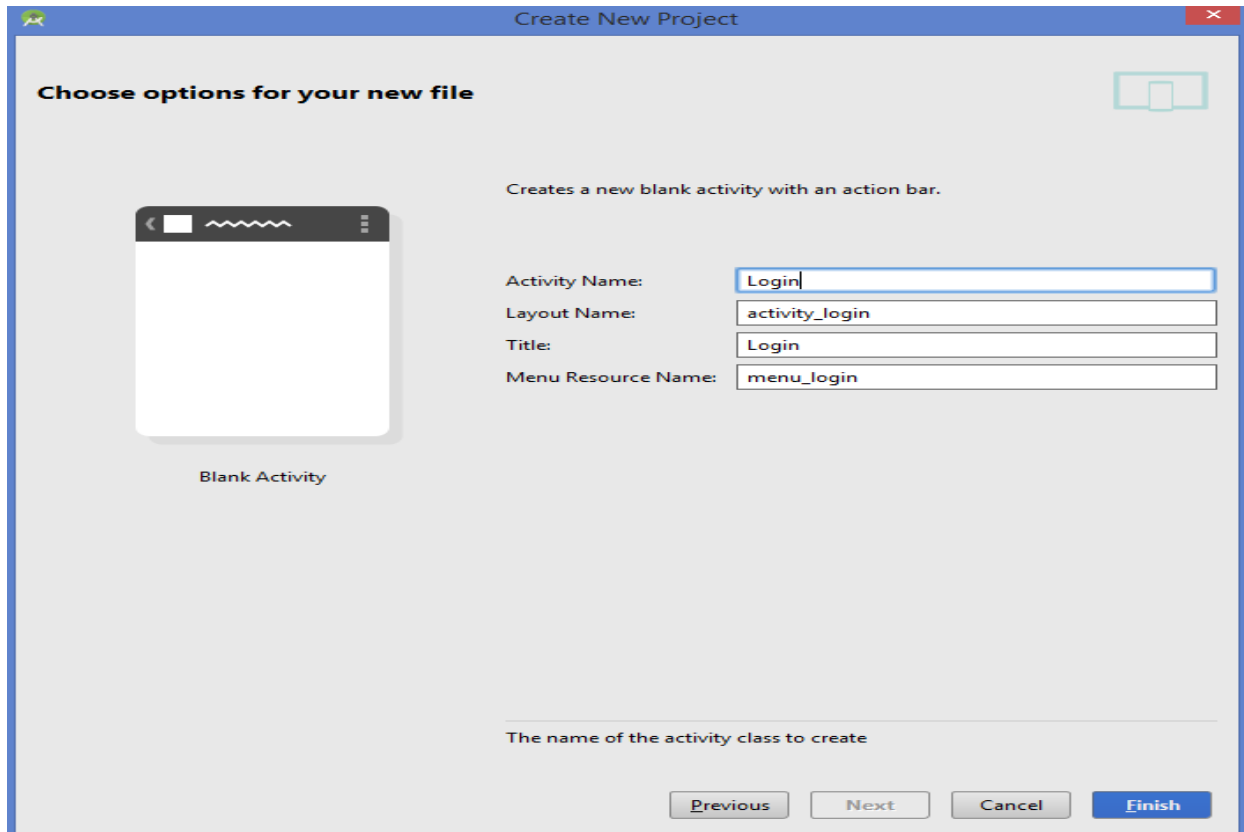
Layouts	Intents
A layout defines the structure for a user interface in your app, such as in an activity.	The intent itself, an Intent object, is a passive data structure holding an abstract description of an operation to be performed.
All elements in the layout are built using a hierarchy of View and ViewGroup objects.	Intents are objects of the android.content.Intent type.

<p>Layouts can be linear, relative, table, absolute, web view, grid view, list view etc.</p>	<p>Intents are 2 types: Explicit and Implicit.</p>
<p>Layouts assembles containers of data.</p>	<p>An intent can contain data via a Bundle.</p>
<div data-bbox="207 331 308 352">Linear Layout</div>  <p>A layout that organizes its children into a single horizontal or vertical row. It creates a scrollbar if the length of the window exceeds the length of the screen.</p> <div data-bbox="425 331 535 352">Relative Layout</div>  <p>Enables you to specify the location of child objects relative to each other (child A to the left of child B) or to the parent (aligned to the top of the parent).</p> <div data-bbox="643 331 714 352">Web View</div>  <p>Displays web pages.</p> <div data-bbox="217 777 295 798">List View</div>  <p>Displays a scrolling single column list.</p> <div data-bbox="477 777 561 798">Grid View</div>  <p>Displays a scrolling grid of columns and rows.</p>	<div data-bbox="873 331 1019 352">Explicit Intents</div> <p>Explicit intent going to be connected internal world of application, suppose if you wants to connect one activity to another activity, we can do this quote by explicit intent, below image is connecting first activity to second activity by clicking button.</p>  <div data-bbox="873 787 1019 808">Implicit Intents</div> <p>These intents do not name a target and the field for the component name is left blank. Implicit intents are often used to activate components in other applications. For example –</p> 

Login

31. How to create a simple Login Screen project Using Android Studio?

Step 1: – Create new Android project. Provide Activity name as *Login*.



Android Login Screen Example Login Activity

Step 2 – Add components in the main activity as shown in the picture below.

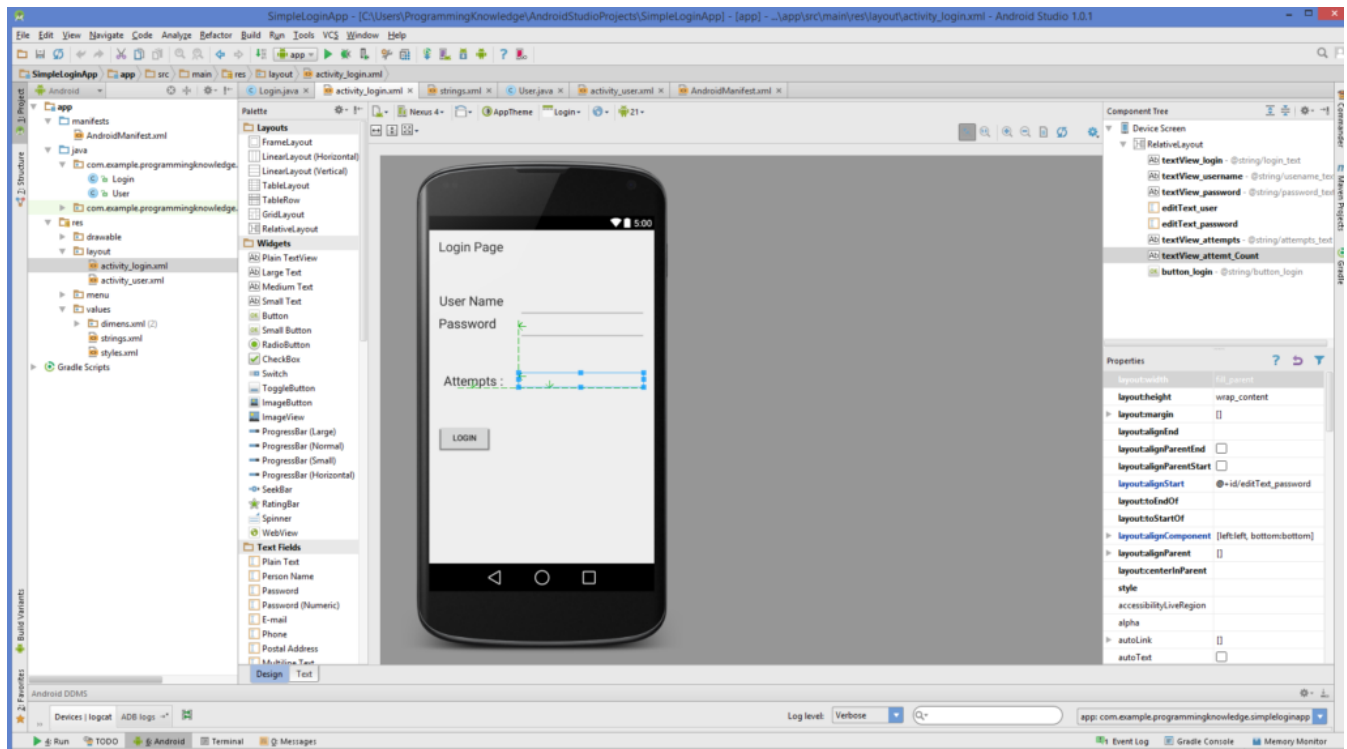


Fig: Android Login Screen Example Design

SimpleLoginApp\app\src\main\res\layout\activity_login.xml

```

1 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2   xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
3   android:layout_height="match_parent"
4   android:paddingLeft="@dimen/activity_horizontal_margin"
5   android:paddingRight="@dimen/activity_horizontal_margin"
6   android:paddingTop="@dimen/activity_vertical_margin"
7   android:paddingBottom="@dimen/activity_vertical_margin" tools:context=".Login">
8
9   <TextView
10    android:layout_width="wrap_content"
11    android:layout_height="wrap_content"
12    android:textAppearance="?android:attr/textAppearanceLarge"
13    android:text="@string/login_text"
14    android:id="@+id/textView_login"
15    android:layout_alignParentTop="true"
16    android:layout_alignParentLeft="true"
17    android:layout_alignParentStart="true" />
18
19   <TextView
20    android:layout_width="wrap_content"
21    android:layout_height="wrap_content"

```

```

22     android:textAppearance="?android:attr/textAppearanceLarge"
23     android:text="@string/username_text"
24     android:id="@+id/textView_username"
25     android:layout_marginTop="66dp"
26     android:layout_below="@+id/textView_login"
27     android:layout_alignRight="@+id/textView_login"
28     android:layout_alignEnd="@+id/textView_login" />
29
30 <TextView
31     android:layout_width="wrap_content"
32     android:layout_height="wrap_content"
33     android:textAppearance="?android:attr/textAppearanceLarge"
34     android:text="@string/password_text"
35     android:id="@+id/textView_password"
36     android:layout_below="@+id/editText_user"
37     android:layout_alignParentLeft="true"
38     android:layout_alignParentStart="true" />
39
40 <EditText
41     android:layout_width="wrap_content"
42     android:layout_height="wrap_content"
43     android:inputType="textPersonName"
44     android:ems="10"
45     android:id="@+id/editText_user"
46     android:layout_alignTop="@+id/textView_username"
47     android:layout_alignParentRight="true"
48     android:layout_alignParentEnd="true" />
49
50 <EditText
51     android:layout_width="wrap_content"
52     android:layout_height="wrap_content"
53     android:inputType="textPassword"
54     android:ems="10"
55     android:id="@+id/editText_password"
56     android:layout_below="@+id/editText_user"
57     android:layout_alignLeft="@+id/editText_user"
58     android:layout_alignStart="@+id/editText_user" />
59
60 <TextView
61     android:layout_width="wrap_content"
62     android:layout_height="wrap_content"
63     android:textAppearance="?android:attr/textAppearanceLarge"
64     android:text="@string/attempts_text"
65     android:id="@+id/textView_attempts"
66     android:layout_below="@+id/editText_password"

```

```

67     android:layout_marginTop="59dp"
68     android:layout_alignRight="@+id/textView_username"
69     android:layout_alignEnd="@+id/textView_username" />
70
71     <TextView
72         android:layout_width="fill_parent"
73         android:layout_height="wrap_content"
74         android:textAppearance="?android:attr/textAppearanceLarge"
75         android:id="@+id/textView_attempt_Count"
76         android:layout_alignBottom="@+id/textView_attempts"
77         android:layout_alignLeft="@+id/editText_password"
78         android:layout_alignStart="@+id/editText_password" />
79
80     <Button
81         android:layout_width="wrap_content"
82         android:layout_height="wrap_content"
83         android:text="@string/button_login"
84         android:id="@+id/button_login"
85         android:layout_below="@+id/textView_attempts"
86         android:layout_alignLeft="@+id/textView_password"
87         android:layout_alignStart="@+id/textView_password"
88         android:layout_marginTop="60dp" />
    </RelativeLayout>

```

Now Change the AndroidManifest.xml file as shown in the code below:

\SimpleLoginApp\app\src\main\AndroidManifest.xml

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="com.example.programmingknowledge.simpleloginapp" >
4
5      <application
6          android:allowBackup="true"
7          android:icon="@drawable/ic_launcher"
8          android:label="@string/app_name"
9          android:theme="@style/AppTheme" >
10         <activity
11             android:name=".Login"
12             android:label="@string/app_name" >
13             <intent-filter>
14                 <action android:name="android.intent.action.MAIN" />
15
16                 <category android:name="android.intent.category.LAUNCHER" />

```



```

17     </intent-filter>
18 </activity>
19 <activity
20     android:name=".User"
21     android:label="@string/title_activity_user" >
22     <intent-filter>
23         <action android:name="com.example.programmingknowledge.simpleloginapp.User" />
24
25         <category android:name="android.intent.category.DEFAULT" />
26     </intent-filter>
27 </activity>
28 </application>
29
30 </manifest>

```

Now go to `values\strings.xml` and change it as shown in the code:

`\SimpleLoginApp\app\src\main\res\values\strings.xml`

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3
4     <string name="app_name">SimpleLoginApp</string>
5     <string name="hello_world">Hello world!</string>
6     <string name="action_settings">Settings</string>
7     <string name="title_activity_user">User</string>
8     <string name="login_text">Login Page</string>
9     <string name="username_text">User Name</string>
10    <string name="password_text">Password</string>
11    <string name="attempts_text">Attempts :</string>
12    <string name="button_login">Login</string>
13    <string name="Secon_page_text">Welocome to the Second Page</string>
14
15 </resources>

```

Step 3 - Add the new activity. Right Click Project -> New -> Activity -> Blank Activity

Give the name to your activity ***User***

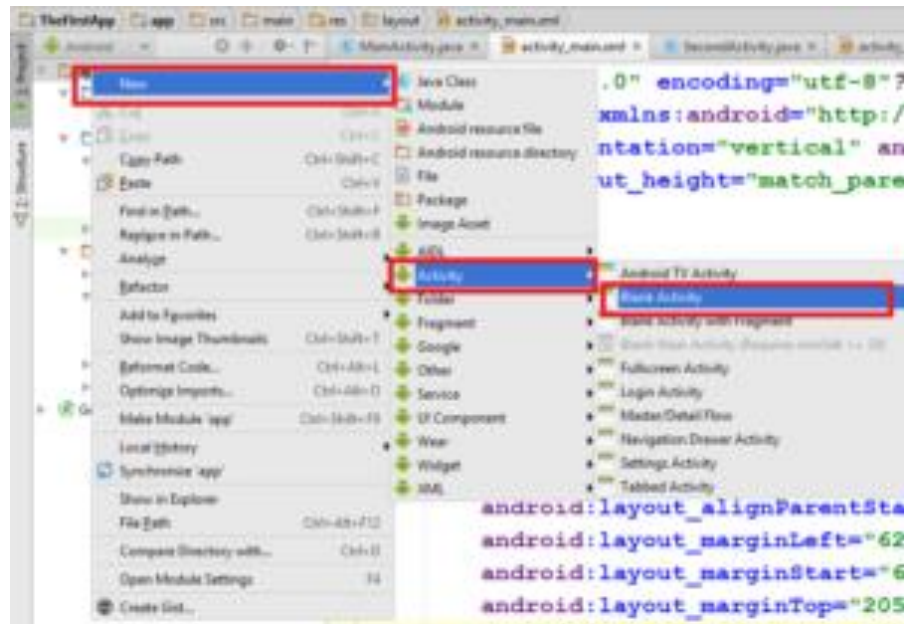


Fig: create blank activity

write the following code:

SimpleLoginApp\app\src\main\res\layout\activity_user.xml

```

1 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2   xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
3   android:layout_height="match_parent"
4   android:paddingLeft="@dimen/activity_horizontal_margin"
5   android:paddingRight="@dimen/activity_horizontal_margin"
6   android:paddingTop="@dimen/activity_vertical_margin"
7   android:paddingBottom="@dimen/activity_vertical_margin"
8   tools:context="com.example.programmingknowledge.simpleloginapp.User"
9   android:id="@+id/user_layout">
10
11 <TextView
12   android:layout_width="wrap_content"
13   android:layout_height="wrap_content"
14   android:textAppearance="?android:attr/textAppearanceLarge"
15   android:text="@string/Secon_page_text"
16   android:id="@+id/textView"
17   android:layout_alignParentTop="true"
18   android:layout_centerHorizontal="true"
19   android:layout_marginTop="137dp" />
</RelativeLayout>

```

com/example/programmingknowledge/simpleloginapp/Login.java

```

1 package com.example.programmingknowledge.simpleloginapp;
2
3 import android.content.Intent;
4 import android.support.v7.app.AppCompatActivity;
5 import android.os.Bundle;
6 import android.view.Menu;
7 import android.view.MenuItem;
8 import android.view.View;
9 import android.widget.Button;
10 import android.widget.EditText;
11 import android.widget.TextView;
12 import android.widget.Toast;
13
14
15 public class Login extends AppCompatActivity {
16     private static EditText username;
17     private static EditText password;
18     private static TextView attempts;
19     private static Button login_btn;
20     int attempt_counter = 5;
21     @Override
22     protected void onCreate(Bundle savedInstanceState) {
23         super.onCreate(savedInstanceState);
24         setContentView(R.layout.activity_login);
25         LoginButton();
26     }
27
28     public void LoginButton() {
29         username = (EditText)findViewById(R.id.editText_user);
30         password = (EditText)findViewById(R.id.editText_password);
31         attempts = (TextView)findViewById(R.id.textView_attemt_Count);
32         login_btn = (Button)findViewById(R.id.button_login);
33
34         attempts.setText(Integer.toString(attempt_counter));
35
36         login_btn.setOnClickListener(
37             new View.OnClickListener() {
38                 @Override
39                 public void onClick(View v) {
40                     if(username.getText().toString().equals("user") &&
41                        password.getText().toString().equals("pass") ) {
42                         Toast.makeText(Login.this,"User and Password is correct",
43                            Toast.LENGTH_SHORT).show();
44                         Intent intent = new
45 Intent("com.example.programmingknowledge.simpleloginapp.User");

```

```

46         startActivity(intent);
47     } else {
48         Toast.makeText(Login.this,"User and Password is not correct",
49             Toast.LENGTH_SHORT).show();
50         attempt_counter--;
51         attempts.setText(Integer.toString(attempt_counter));
52         if(attempt_counter == 0){
53             login_btn.setEnabled(false);
54         }
55     }
56
57 }
58 }
59 );
60 }
61
62 @Override
63 public boolean onCreateOptionsMenu(Menu menu) {
64     // Inflate the menu; this adds items to the action bar if it is present.
65     getMenuInflater().inflate(R.menu.menu_login, menu);
66     return true;
67 }
68
69 @Override
70 public boolean onOptionsItemSelected(MenuItem item) {
71     // Handle action bar item clicks here. The action bar will
72     // automatically handle clicks on the Home/Up button, so long
73     // as you specify a parent activity in AndroidManifest.xml.
74     int id = item.getItemId();
75
76     //noinspection SimplifiableIfStatement
77     if (id == R.id.action_settings) {
78         return true;
79     }
80
81     return super.onOptionsItemSelected(item);
82 }

```

com/example/programmingknowledge/simpleloginapp/User.java

```

1 package com.example.programmingknowledge.simpleloginapp;
2
3 import android.support.v7.app.ActionBarActivity;

```

```

4 import android.os.Bundle;
5 import android.view.Menu;
6 import android.view.MenuItem;
7
8
9 public class User extends ActionBarActivity {
10
11     @Override
12     protected void onCreate(Bundle savedInstanceState) {
13         super.onCreate(savedInstanceState);
14         setContentView(R.layout.activity_user);
15     }
16
17
18     @Override
19     public boolean onCreateOptionsMenu(Menu menu) {
20         // Inflate the menu; this adds items to the action bar if it is present.
21         getMenuInflater().inflate(R.menu.menu_user, menu);
22         return true;
23     }
24
25     @Override
26     public boolean onOptionsItemSelected(MenuItem item) {
27         // Handle action bar item clicks here. The action bar will
28         // automatically handle clicks on the Home/Up button, so long
29         // as you specify a parent activity in AndroidManifest.xml.
30         int id = item.getItemId();
31
32         //noinspection SimplifiableIfStatement
33         if (id == R.id.action_settings) {
34             return true;
35         }
36
37         return super.onOptionsItemSelected(item);
38     }
39 }

```

Now Run your app

Output:

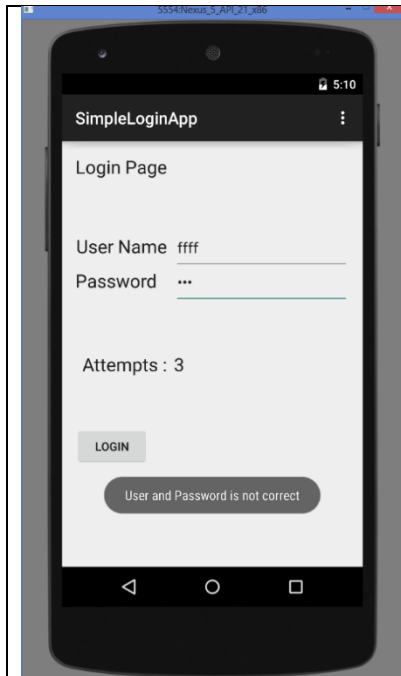


Fig: Android Login Screen
Example Output wrong
user and password

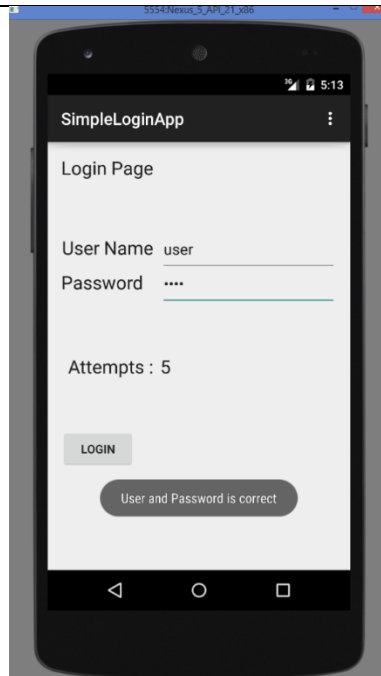


Fig: Android Login Screen
Example Output correct
user and password



Fig: Android Login Screen
Example Output correct
user and password Next
activity