


# IAT 1 Answer Key

USN <input type="text"/>									
Internal Assessment Test I Set-1 – march 2024									
Sub:	Mobile Application Development				Sub Code:	BIS654C	Branch	ECE	
Date:	27-03-2025	Duration:	90 min's	Max Marks:	50	Sem/Sec:	VI / A, B , C, D	OBE	
<b>Answer any FIVE questions</b>							MARKS	CO	RBT
1	a) What is android? Explain their versions and applications?					5	CO1	L1	
	b) List and Explain the features of Android					5			
2	a) Discuss about softwares and tools used in configuring android environment.					4	CO1	L1	
	b) Explain architecture of Android with the neat diagram					6			
3	a) Explain how you are creating first android application.					5	CO2	L2	
	b) Difference between Table Layout and Frame Layout					5			
4	a) Discuss about the anatomy of an android application					5	CO2	L1	
	b) Explain neatly about Absolute and Relative Layouts					5			
5	a) Briefly discuss about basic views and their contents					6	CO3	L1	
	b) Discuss about Image views to display pictures					4			
6	a) List out common attributes of viewgroups.					5	CO3	L2	
	b) Explain about understanding different components of a Screen and fundamentals of UI Design?					5			

Faculty Signature

CCI Signature

HOD Signature










## Q1 a)What is android? Explain their versions and applications?

Android is a Stack of software for mobile devices that are an Operating System, Middleware and key Applications

Android is a Linux-based operating system which is designed for touch screen mobile devices like smart phones and tablet computers.

It is an open source technology that allows the software to be freely modified and distributed by device manufacturers, wireless carriers and developers.

## Android versions

Versions	APK	Release date and Activity
1.0	1	It was released on September 23, 2008. ❖ You Tube video player ❖ Wi-Fi and Bluetooth support ❖ Camera maintain this version
1.5(Cupcake) 	3	It was released on April 29, 2009. ❖ Recording and watching videos in MPEG-4 and 3GP formats. ❖ It was also populating the home screen with widget and animated screen transition.
1.6 (Donut) Based on Linux Kernel 2.6.29 	4	It was released SDK 1.6 on September 15, during 2009. ❖ Integrated as camera, camcorder, and gallery interface. ❖ Support for WVGA screen resolutions, and an updated search experience.
2.0/2.1 (Eclair) Based on Linux Kernel 2,6.29 	5,6,7	It was released SDK 2.0 on October 26, 2009.. ❖ Changes as the wallpapers, new camera features that are:- Flash support, digital zoom, scene mode, colour effect etc. ❖ Improved typing speed on virtual keyboard, a smarter dictionary that learns from word usage
2.0/2.1 (Eclair) Based on Linux Kernel 2,6.29 	5,6,7	It was released SDK 2.0 on October 26, 2009.. ❖ Changes as the wallpapers, new camera features that are:- Flash support, digital zoom, scene mode, colour effect etc. ❖ Improved typing speed on virtual keyboard, a smarter dictionary that learns from word usage
2.2 (Froyo) Based on 2.6.32 	8	It was released SDK 2.2 on May 20, 2009. ❖ Changes as the integration of chrome's V8 java script engine into the browser app, voice dialling and contact sharing over Bluetooth, adobe flash support, speed implements through JIT compilation, USB tethering and Wi-Fi function.
2.3(Gingerbread) Based on 2.6.35 	9/10	It was released SDK 2.3 on December 6, during 2010. ❖ Support as web M/VP8 video playback and AAC audio encoding, near field communication, and copy/paste functionality that select a world by press-hold, copy and paste.
3.0 (Honeycomb) Based on 2.6.36 	14	It was released as SDK 3.0 on February 22,during 2011. ❖ This version focuses on tables, such as Motorola Xoom, the first tablet to be released. ❖ It improves multitasking. ❖ Supports multicore processor, hardware accelerations. ❖ It provides a 3D desktop with redesigned widget..
4.0 (Icecream Sandwich) Based on 3.0.1 		It was released SDK 4.0.1 on October 19,during 2011 ❖ SDK 4.0.1 as and 4.x successors unify the 2.3.x Smartphone and 3.x tablet SDKs. ❖ Include 1080P recording and a customizable launcher.
4.1(JellyBean) 		It was released SDK 4.1 on June 27, 2012. ❖ Include as triple buffering, automatically resizable app widgets. ❖ Improved voice search and multichannel audio

### Q1 b) List and explain the features of Android?

1. Open Source
2. Storage
3. Media support
4. Streaming media support
5. Multitouch
6. Web browser
7. Video calling
8. Multitasking
9. Accessibility
10. Voice based features
11. External storage

1. **Open Source:** Android is an open-source operating system. This way that the source code for Android is open to the public, dissimilar iOS, which is kept secret by Apple. This way that anyone can work on the operating system, not only one company developers. Therefore app developers for Android devices are able to implement extra features of their apps, due to the access they have to contain the source code. Again, these are only some of the unique features to Android. iOS also have many key abilities that are missing on Android devices. In the end, it is up to you to decide which operating system you would rather have. Anyone of you choose will be the right decision for you.

2. **Storage:** SQLite, a lightweight relational database, is used for data storage purposes.

3. **Media support:** Android supports the following audio/video/still media formats: WebM, H.263, H.264, AAC, HE-AAC (in 3GP or MP4 container), MPEG-4 SP, AMR, AMR-WB (in 3GP container), MP3, MIDI, WAV, JPEG, PNG, GIF, Ogg Vorbis, FLAC, BMP, WebP.

4. **Streaming media support:** RTP/RTSP streaming (3GPP PSS, ISMA), HTML progressive download (HTML5 <video> tag). Adobe Flash Streaming (RTMP) and HTTP Dynamic

Streaming are supported by the Flash plugin. Apple HTTP Live Streaming is support through RealPlayer for Android, and through the operating system for the duration Android 3.0 (Honeycomb).

5. **Multitouch:** Android has native support for multi-touch which was initially made available in handsets such as the HTC Hero. These features was originally disabled at the kernel level (possibly to avoid infringing Apple's patents on touch-screen technology at the time).Google has released an update version for the Nexus One and the Motorola Droid which enables multi-touch natively.

6. **Web browser:** The web browser available in Android is based on the open-source WebKit layout engine, attached with Chrome's V8 JavaScript engine. The browser scores 100/100 on top of the Acid3 test on Android 4.0.

7. **Video calling:** Android does not support local video calling, but several handsets have a customized version of the operating system that supports it, either through the UMTS network or ended IP. Video calling through Google Talk is available within Android 2.3.4 and later. Gingerbread allows Nexus S to place Internet calls with a SIP account. This allows for enhanced VoIP dialling to other SIP accounts and even phone numbers. Skype 2.1 offer video calling in Android 2.3, include front camera support. Users with the Google+ android app can video chat with other google+ users through hangouts.

8. **Multitasking:** Multitasking of application, with unique handling of memory allocation, will be existing.

9. **Accessibility:** Built in text to speech is provided by Talk back for people with low or no vision. Enhancements for people through hearing disabilities are available as is other aids.

10. **Voice based features:** Google search through voice has been available since opening release. Voice activities for navigation, calling, texting, etc. are supported on Android 2.2 forwards. As of Android 4.1, Google has expanded Voice Actions with the ability to talk back and read answers from Google's Knowledge Graph when queried with specific commands. The ability to control hardware have not yet been implemented.
11. **External storage:** Most Android devices include microSD slot and can read microSD cards formatted with FAT32, Ext3 or Ext4 file system. To allow use of high-capacity storage media such as USB flash drives and USB HDDs, many Android tablets also include USB 'A' receptacle. Storage format with FAT32 is handled by Linux Kernel VFAT driver, As 3rd party solutions are necessary to handle other popular file systems such as NTFS, HFS Plus and ex FAT.

## **Q2 a) Discuss about softwares and tools used in configuring android environment**

### **1) Operating System**

To develop an Android program, the necessary supporting operating systems used in a computer can be as follows:

- a. Windows XP (32 bit), Vista (32 or 64 bit), or Windows 7 (32 or 64 bit)
- b. Mac OS X (Intel) 10.5.8 (x86 only)
- c. Linux (i386 : tested in Lucid Lynx, Ubuntu Linux)
  - GNC C Library (glibc) 2.7 or later is required.
  - On Ubuntu Linux, version 8.04 or later is required.
  - 64-bit distributions should be capable of running 32-bit applications

### **2) Java JDK**

As Android programs are developed in the Java programming language we have to install Java Development Kit (JDK) in computer which is a free software where the JDK includes the Java SE Runtime (JRE).

### **3) Android SDK**

The Android SDK (Software Development Kit) is the most important software of android which is installed. The Android SDK provides to test android applications, the API libraries, an emulator, documentation, sample code, developer tools, and tutorials which help you to build, test and debug apps for Android.

Android SDK is made up of two main parts: the tools and the packages. When you first install the SDK, all you obtain are the base tools. These are executables and supporting files that help you develop applications. The packages are the records specific to a particular version of Android (called a platform) or a particular add-on to a platform.

#### **Platform Used**

To develop Android application we use the Eclipse, IntelliJ and AIDE different integrated development environments (IDEs) which supports multiple languages such as Java, C, C++, COBOL, Python, etc. It is such type of environment upon which the extensible plug-in can run.

### **4) Android Development Tools (ADT)**



Android Development Tools (ADT) is a plugin for the Eclipse IDE which provides a suitable environment to develop an android application where creation, compilation and debugging are possible. ADT can able to set up new Android projects, create an application UI, insert packages based on the Android Framework API, debug(or clear up) your applications using the Android SDK tools, and export signed (or unsigned) .apk files in order to distribute in the application.

## **5) Android Virtual Devices (AVDs)**

An Android Virtual Device (AVD) is an emulator configuration that enables to model an actual device by calling hardware and software options to be emulated by the Android Emulator.

Use the AVD Manager to create an AVD. Launch it from eclipse by clicking Window | AVD Manager. You can also start AVD Manager by calling `adoption`, from the `<sdk>/tools/` directory.

An Android Virtual Device (AVD) is used for testing the Android applications. An AVD is an emulator occurrence that enables to form a real device. Each AVD consists of a hardware sketch, a connection to a system image, and emulated storage, such as a secure digital (SD) card.

## **6) Emulators**

The Android SDK comes with a virtual mobile device emulator that runs on your computer. The emulator enables us trial product, to develop and test Android applications without using a physical device. The Android emulator mimics all of the hardware and software features of typical mobile devices, except that it cannot place actual phone calls. It provides a selection of navigation and control keys, which you can “press” using your mouse or keyboard to generate events for our application. It provides a screen in which our application is displayed, together with any other active Android applications.

## **7) Dalvik Virtual Machine**

The key figure in Google’s implementation of JVM is Dan Bornstein, who has written the Dalvik VM-Dalvik is the name of a town in Iceland. Dalvik VM takes the generated Java class files and combines them into one or more Dalvik Executable (.dex) files. It reuses duplicate information from numerous class files, effectively reducing the gap requirement (uncompressed) by half from a traditional .jar file.

Android uses the Dalvik virtual machine with just-during –time compilation to run Dalvik bytecode, which is frequently translated from Java bytecode. Google has also fine-tuned the garbage collection in the Dalvik VM, but it has chosen to remove just-in-time(JIT) compiler, in early releases. Android 2.3 has added JIT.

**Q2 b) Explain architecture of Android with the neat diagram.**



The Android OS can be referred to as a software stack of different layers, where every layer is a group of several programs components. It includes operating systems, middleware and important applications. Every layer in the architecture provides different services.

Android has following layers

- Applications
- Application Framework
- Libraries Android Runtime
- Linux Kernel

### 1) Application

At this top layer, you will find applications that ship with the Android device (such as Phone, Contacts, Browser, etc.), as well as applications that you download and install from the Android Market. Any applications that you write are located at this layer.

### 2) Application Framework:

The Android application framework provides APIs for developers of the application layer, which is actually an application framework. This layer contain following basic components:

- Activity Manager: This layer manages the lifecycle of application and provides a common navigation backstack
- Window Manager: As the name suggest it manages the window surface. Then it organizes the screen layout and locates the drawing surface and also performs other windows related jobs
- View Manager: View the window
- Content Manager: Enables application to access data from other applications or to share their own data
- Notification Manager: we get the notification from the system when the battery is low. If the programmer wants he can also enable all applications to display customer alerts in the status bar
- Package Manager: It manages the packages of the applications.
- Telephony Manager: handle the receiver call or voice calls.
- Resource Manager: Provide access to the non-code resources ( Graphics, localized strings and layout files)
- Location Manager: provides access to the system location services

- XMPP Service Manager: manages services like music application, browser, ringtone etc.

### 3) Libraries

3<sup>rd</sup> layer of the android architecture is the libraries layer. It is written in C and C++ libraries. Following are the major components of this layer:

- Surface manager: Handles all the surfaces rendered by each component of the frame
- Open GL ES: For rendering 2D and 3D graphics on embedded and mobile system
- SQLite database: It is a lightweight RDBMS
- Media Framework: is a set of APIs for developers which enables them to create a multimedia application on an android platform.
- SGL: Scalable graphics libraries responsible for implementing low level graphics by using JNI
- Free type: Support the font quality, the image(bitmap images)
- SSL: Secured socket layer, for establishing secure communication between an app and a server, ensuring data privacy and integrity
- Webkit: Provides browser support. It support browsers like Google Chromes, Apple safari etc.
- Lib C : C libraries. Provides C libraries headers

### 4) Android Runtime

Android Runtime consists of Dalvik Virtual Machine and Core Libraries.

**DVM(Dalvik Virtual Machine)** : Dalvik Virtual Machine (DVM) is the custom program introduced for Android apps. It takes the Java code and creates an optimized version of it in a file with .dex(extension) which is known as Dalvik executable. This format allows the apps to run quickly with fewer resources, i.e. on mobile phones and low-memory, slower devices.

**Core Libraries:** These are different from Java SE and Java ME libraries. But these libraries provide most of the functionalities defined in the Java SE libraries

### 5) Linux Kernel

This is the kernel on which Android is based. This layer contains all the low level device drivers for the various hardware components of an Android device.

Following are the drivers:

- Display driver
- Camera driver
- Bluetooth driver
- Binder driver(IPS)
- USB driver
- WiFi driver
- Keypad driver
- Audio driver
- Power management driver

### Q3 a) Explain how you are creating first android application Creating the first android application

1. Click on Eclipse icon and select your workspace
2. Select File -> New -> Project/Other
3. Expand the android folder and select android project and click Next
4. Give the Project Name and click Next
5. Choose your SDK to target whether Android 2.2 or Android 4.0 Here we select the Android 2.2 SDK and click Next

6. To configure the Android Project we have to give the Package Name as your wish in this format-com.pkg.example and click Finish
7. In Package Explorer a new android project is created named AndroidProject
8. To write the program we have to open the main.xml
9. To save the changes made to your project, press Ctrl+s.
10. You are now ready to test your application on the Android Emulator. Select the project name in Eclipse and press F11. You will be asked to select a way to debug the application. Select required Android Application and click OK.
11. The Android Emulator will now be started (if the emulator is locked, you need to slide the unlock button to unlock it first).
12. Click the Home button (the house icon in the lower-left corner above the keyboard) so that it now shows the Home screen.
13. Click the application Launcher icon to display the list of applications installed on the device.

### Q3b) Difference between Table Layout and Frame Layout

In Android, **TableLayout** and **FrameLayout** serve different purposes for arranging UI elements.

#### TableLayout

- Organizes UI components into rows and columns, similar to a table.
- Each cell can hold one view, and views are aligned in rows and columns.
- Useful for creating structured layouts like forms or grids.
- Example usage: Arranging TextViews and EditTexts in a registration form.

```
<TableLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:stretchColumns="1">
    <TableRow>
        <TextView
            android:text="Label 1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
        <EditText
            android:hint="Input 1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
    </TableRow>
    <TableRow>
        <TextView
            android:text="Label 2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
        <EditText
            android:hint="Input 2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
    </TableRow>
</TableLayout>
```



## FrameLayout

- Displays a single view or stacks multiple views on top of each other (overlapping).
- The child views can overlap and are typically positioned relative to the container.
- Useful for simple layouts or overlaying components like an image with a button.
- Example usage: Creating a layout where a TextView overlays an ImageView.

<FrameLayout

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<ImageView

android:src="@drawable/image"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent" />

<TextView

android:text="Overlay Text"

android:layout\_gravity="center"

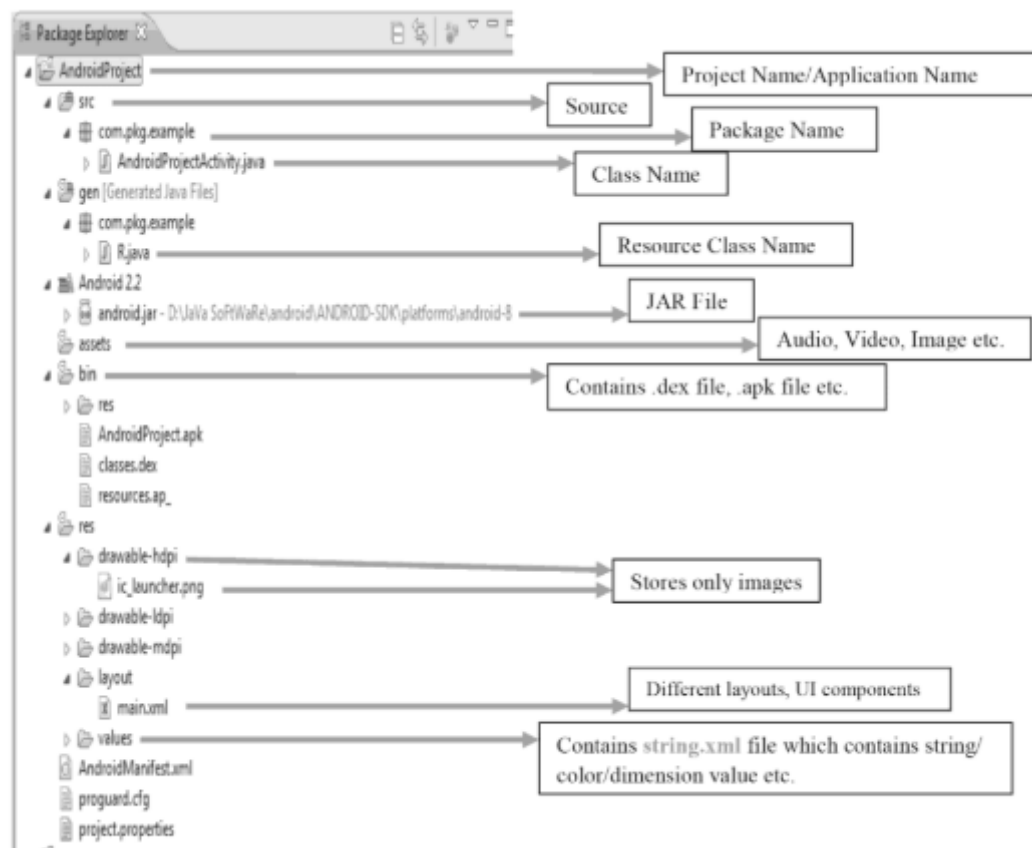
android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content" />

</FrameLayout>

Essentially, **TableLayout** is great for tabular layouts with rows and columns, while **FrameLayout** is better for stacking or overlaying views.

### Q4 a) Discuss about the anatomy of an android application.



There are various files that form an Android project in the Package Explorer in Eclipse.

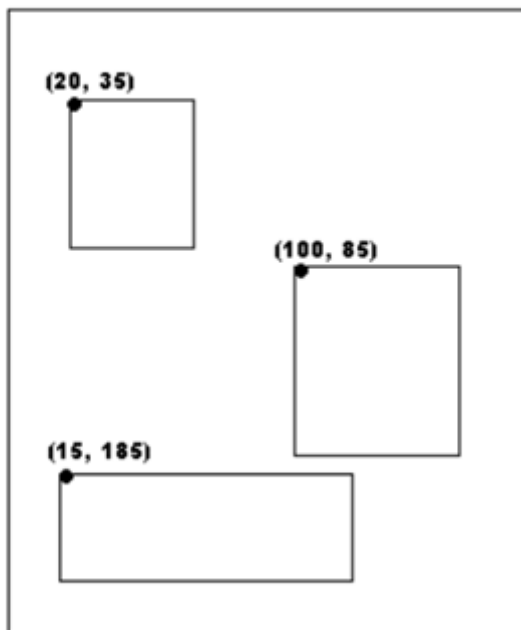
The various folders and their files are as follows:

- ✦ **src:** This folder contains the .java source files for the project. In this example, there is one file, `AndroidProjectActivity.java`. The `AndroidProjectActivity.java` file is the source file for the activity. We write the code for this application in this .java file. The Java file is listed under the package name i.e. `com.pkg.example` for the project.
- ✦ **gen:** This folder contains the `R.java` file, which is generated by the compiler that references all the resources found in this project. We cannot modify this file. All the resources in this project are automatically compiled into this class hence we can transfer to them using the class.
- ✦ **Android 2.2 libraries:** This thing contains one file, `android.jar`, which contains all the class libraries needed for an Android application.
- ✦ **assets:** Folder contains all the resources used by this application, such as HTML, text files, databases, etc.
- ✦ **bin:** This folder contains the files built by the ADT during the compilation. It generates the .apk file (Android Package). An .apk file is the application binary of an Android application. It contains the entire things needed to run an Android application.
- ✦ **res:** This folder contains all the resources files such as pictures, XML files for defining layouts used in this application.  
  
It also contains some other subfolders: `drawable-<resolution>`, `layout`, and `values`. It can also support devices with different screen resolutions and densities.
- ✦ **AndroidManifest.xml:** This is the manifest file for this Android project. It contains information about the Android application such as minimum Android version, permission to access Android device capabilities such as internet access permission , ability to use phone permission, etc. Here we specify the permissions and other features like intent-filters, receivers, etc required by application.
- ✦ **main.xml:** The `main.xml` file defines the user interface for activity.
- ✦ **Layout:** Contains XML layout files. Layout files are XML files which define how various Android objects (such as textboxes, buttons, etc) are organized on the screen.
- ✦ **Values:** XML files which store various string values (titles, labels, etc).\

\*\*\*

#### Q4 b) Explain neatly about Absolute and Relative Layouts

## Absolute Layout



The `AbsoluteLayout` enables you to specify the exact location of its children. Consider the following UI defined in `main.xml`:

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:layout_width="fill_parent" android:layout_height="fill_parent"
    xmlns:android=http://schemas.android.com/apk/res/android>
    <Button
        android:layout_width="188dp" android:layout_height="wrap_content"
        android:text="Button" android:layout_x="126px"
        android:layout_y="361px"/>
    <Button
        android:layout_width="113dp" android:layout_height="wrap_content"
        android:text="Button" android:layout_x="12px"
        android:layout_y="361px"/>
</AbsoluteLayout>
```

There is a problem with absolute layout when the activity is viewed on a high resolution screen. For this reason the `AbsoluteLayout` has been deprecated since Android 1.5. It is not guaranteed to be supported in future version of android.

## RelativeLayout

The `RelativeLayout` enables you to specify how child views are positioned relative to each other. Consider the following `main.xml` file:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout android:id="@+id/RLayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
```

```

xmlns:android="http://schemas.android.com/apk/res/android"
>
<TextView android:id="@+id/lblComments"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Comments" android:layout_alignParentTop="true"
android:layout_alignParentLeft="true"
/>
<EditText android:id="@+id/txtComments"
android:layout_width="fill_parent"

android:layout_height="170px" android:textSize="18sp"
android:layout_alignLeft="@+id/lblComments"
android:layout_below="@+id/lblComments"
android:layout_centerHorizontal="true"
/>
<Button android:id="@+id/btnSave"
android:layout_width="125px"
android:layout_height="wrap_content" android:text="Save"
android:layout_below="@+id/txtComments"
android:layout_alignRight="@+id/txtComments"
/>
<Button android:id="@+id/btnCancel"
android:layout_width="124px"
android:layout_height="wrap_content" android:text="Cancel"
android:layout_below="@+id/txtComments"
android:layout_alignLeft="@+id/txtComments"
/>
</RelativeLayout>

```

The UI of the above code would look like –



- Each view is embedded within the relative layout has attributes that enable it to align with another view.
- The value for each of these attributes is the **ID** for the view that you are **referencing**.
- These **attributes** are as follows:

### Q5 a) Briefly discuss about basic views and their contents

Basic views used to design the UI of your Android applications



These basic views enable you to display text information, as well as perform some basic selection.

- TextView
- EditText
- Button
- ImageButton
- CheckBox
- ToggleButton
- RadioButton
- RadioGroup
- 

```
<Button android:id="@+id/btnSave"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Save" />
```

```
<EditText android:id="@+id/txtName"
android:layout_width="fill_parent"
android:layout_height="wrap_content" />
```

```
<TextView
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/hello"
/>
```

```
<ImageButton android:id="@+id/btnImg1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:src="@drawable/icon" />
```

```
<CheckBox android:id="@+id/chkAutosave"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Autosave" />
```

```
<CheckBox android:id="@+id/star"
style="?android:attr/starStyle"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />
```

```
<ToggleButton android:id="@+id/toggle1"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />
```

```
<RadioGroup android:id="@+id/rdbGp1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:orientation="vertical" >
<RadioButton android:id="@+id/rdb1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Option 1" />
<RadioButton android:id="@+id/rdb2"
```

```
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Option 2" />
</RadioGroup>
```

### Q5 b) Discuss about Image views to display pictures

In Android, **ImageView** is a UI component used to display images. It supports various image formats and can handle scaling, resizing, and other image-related properties.

#### Key Attributes

- **android:src**: Specifies the image to display.
- **android:scaleType**: Defines how the image should be scaled within the ImageView.
- **android:contentDescription**: Provides a description for accessibility purposes.
- **android:adjustViewBounds**: Allows the ImageView to adjust its size based on the image's dimensions.

#### Syntax Example

```
<ImageView
    android:id="@+id/imageView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/example_image"
    android:scaleType="centerCrop"
    android:contentDescription="Example Image" />
```

#### Common scaleType Options

- **center**: Centers the image without scaling.
- **centerCrop**: Scales the image so that it fills the view while keeping its aspect ratio, cropping the sides.
- **fitCenter**: Scales the image to fit within the view, maintaining its aspect ratio.
- **fitXY**: Stretches the image to fit both width and height.

You can use **ImageView** to display graphics, icons, or even dynamically loaded images at runtime. Do you want to see some advanced examples, like loading images from the internet using libraries like Glide or Picasso? Let me know!

### Q6 a) List out common attributes of viewgroups.

One or more views can be grouped together into a ViewGroup. A ViewGroup (which is itself a special type of view) provides the layout in which you can order the appearance and sequence of views. Examples of ViewGroups include LinearLayout and FrameLayout. A ViewGroup derives from the base class android.view.ViewGroup.

Each View and ViewGroup has a set of common attributes as shown in the following table.

ATTRIBUTE	DESCRIPTION
layout_width	Specifies the width of the View or ViewGroup
layout_height	Specifies the height of the View or ViewGroup
layout_marginTop	Specifies extra space on the top side of the View or ViewGroup
layout_marginBottom	Specifies extra space on the bottom side of the View or ViewGroup
layout_marginLeft	Specifies extra space on the left side of the View or ViewGroup
layout_marginRight	Specifies extra space on the right side of the View or ViewGroup
layout_gravity	Specifies how child Views are positioned
layout_weight	Specifies how much of the extra space in the layout should be allocated to the View
layout_x	Specifies the x-coordinate of the View or ViewGroup
layout_y	Specifies the y-coordinate of the View or ViewGroup

Some of these attributes are applicable only when a View is in a specific ViewGroup. For example, the layout\_weight and layout\_gravity attributes are applicable only when a View is in either a LinearLayout or a TableLayout.

Android supports the following ViewGroups:

- LinearLayout
- AbsoluteLayout
- TableLayout
- RelativeLayout
- FrameLayout

### Q6 b) Explain about understanding different components of a Screen and fundamentals of UI Design?

- **Views** : Views are the base class for all visual interface elements (commonly known as *controls* or *widgets*). All user interface UI controls, including the layout classes, are derived from View. A View is an object/widget that draws something on the screen by the help of user interact. Examples of widgets are buttons, text boxes ,labels etc.
- **View Groups** : View Groups are extensions of the View class that can contain multiple child Views, In order to Extend the ViewGroup class to create compound controls made up of interconnected child Views. The ViewGroup class is also extended to provide the Layout Managers that help us to control layout within our Activities. A ViewGroup provides the layout in which we can order the appearance and sequence of views. Examples of View groups are FrameLayout, LinearLayout, etc.
- **Fragments** : Fragments, introduced in Android 3.0 which uses API level 11, are used to encapsulate portions of your UI. This encapsulation makes Fragments particularly useful when optimizing your UI layouts for different screen sizes and creating reusable user interface (UI) elements. Each Fragment includes its own user interface (UI) layout and receives the related input events but is tightly bound to the Activity into which each must be embedded. Fragments are similar to UI View Controllers in iPhone development.
- **Activities** : Activities, represent a single screen that user interact .Activities are the Android equivalent of Forms in traditional Windows desktop development. To display a UI, we assign a View (usually a layout or Fragment) to an Activity.