Yashwantrao
Chavan
Maharashtra
Open University

CMP516

Android Programming

Android Programming

Yashwantrao Chavan Maharashtra Open University

Dnyangangotri, Near Gangapur Dam

Nashik-422222

Yashwantrao Chavan Maharashtra Open University

Vice-Chancellor: Prof. E. Vayunandan

SCHOOL OF COMPUTER SCIENCE

Dr. Pramod Khandare Director School of Computer Science Y.C.M.Open University Nashik	**Shri. Madhav Palshikar** Associate Professor School of Computer Science Y.C.M.Open University Nashik	**Dr. P.V. Suresh** Director School of Computer and Information Sciences I.G.N.O.U. New Delhi
Dr. Pundlik Ghodke General Manager R&D, Force Motors Ltd. Pune.	**Dr. Sahebrao Bagal** Principal, Sapkal Engineering College Nashik	**Dr. Madhavi Dharankar** Associate Professor Department of Educational Technology S.N.D.T. Women's University, Mumbai
Dr. Urmila Shrawankar Associate Professor, Department of Computer Science and Engineering G.H. Raisoni College of Engineering Hingana Road, Nagpur	**Dr. Hemant Rajguru** Associate Professor, Academic Service Division Y.C.M.Open University Nashik	**Shri. Ram Thakar** Assistant Professor School Of Continuing Education Y.C.M.Open University Nashik
Mrs. Chetna Kamalskar Assistant Professor School of Science and Technology Y.C.M.Open University, Nashik	**Smt. Shubhangi Desle** Assistant Professor Student Service Division Y.C.M.Open University Nashik	

Writer	Editor	Co-ordinator	Director
Prof.A.R.Bramhecha Assistant Professor, SNJB's Late Sau. K.B. Jain College of Engineering Chandwad, Nashik.	Mr. Vipin Wani Assistant Professor, Department of Computer Science & Engineering, Sandip University, Nashik,	Ms. Monali R. Borade Academic Co-ordinator School of Computer Science, Y.C.M. Open University, Nashik	**Dr. Pramod Khandare** **Director** School of Computer Science, Y.C.M. Open University, Nashik

Production

Android Programming (CMP516)

Course Objective

- To understand the fundamentals involved in technologies of Mobile computing
- To introduce Android & understand the basic of Android.
- Design the home screen using UI screen elements.
- Describe the platforms upon which the Android operating system will run.
- To understand android terminologies & resources
- Create an application that uses user interface elements under the Android operating system
- Access and work with databases under the Android operating system
- To share data with another application.

Course Outcomes:

- Students will be able to understand fundamentals of mobility computing.
- Students will be able to understand working of Android architectures and their applications.
- Students will be able understand the user interface elements and learn the database tools for developing applications on mobile platforms like Android.
- Student will be able to gain the knowledge of deployment of application in actual android device.

Unit No and Name	**Title**	**Counseling Sessions**	**Weightage**
Unit 1: Introduction to Mobile Development	❖Mobile Computing ❖Historical of Mobile Environments ❖Early Mobile Phones to Smartphone's ❖Tablets ❖Mobile Computing Architecture ❖Mobile Generation o Devices for 1G, 2G, 2,5G,3G o Applications for 1G, 2G, 2,5G, 3G ❖Handoff ❖Roaming ❖GSM & GSM Architecture ❖Network Signalling ❖GSM INTERFACES ❖GSM Channels ❖Mobility Management in GSM	3	10
Unit 2: Introduction to Android	❖ Android o 2.1.1 What is Android o 2.1.2 History and Version o 2.1.3 Android Architecture o 2.1.4 Hello Android example ❖ Dalvik VM ❖ Software Stack ❖ R.java file	3	10

	❖ Screen Orientation ❖ Android Operating System o Introduction o Android Versions with Features ❖ Android Development Elements ❖ Installing the Java Development Kit ❖ Installing Android Studio ❖ Set up Android Studio ❖ Start a new Android Studio project ❖ Update your Android Studio software often		
Unit 3: User Interface Screen Elements	❖Toast & Snack Bar ❖Custom Toast ❖Button o Toggle Button o Switch Button o Image Button o Radio Button ❖Text View and EditText, CheckBox ❖Alert Dialog and Button Sheets ❖Spinner ❖Date Picker and Time Picker ❖Rating Bar and Progress Bar ❖File Download	4	10
Unit 4: Android Development Elements	❖Terminologies o Context o Activity o Intent o Linking Activity using Intent o Calling Build-In Application using Intent ❖Notifications Service ❖ Broadcast ❖Adapter Resources o Working with different types of Resources	4	10
Unit 5: Android Terminologies and Resource Handling	❖Layouts o Linear Layout o Absolute Layout o Frame Layout o Relative Layout o Table Layout o Creation of Layout Programmatically ❖View o ListView o GridView o RecyclerView o ScrollView o WebView	4	10

Unit 6: Android User Interface Elements	❖File system in android ❖Internal and external storage ❖Creating SQLite database ❖Editing Tasks with SQLite ❖Cursors and content values ❖ Working with Android database ❖Publish Android Application in Android Market	4	10
Unit 7: Providers	❖Content Provider ❖Content Provider Fundamental o Uri o Content Resolver ❖How to Create a Content Provider? ❖Contact Content Provider ❖Other Built-in Content Providers ❖Creating Custom Content Provider	4	10
Unit 8: Receivers	❖Broadcast Receivers ❖Basics of Broadcast Receiver o Register Broadcast o Receive Broadcasts ❖Implementing a broadcast receiver ❖Case Study	4	10

Text Books:

1.**Wireless and Mobile Network Architectures**by Yi Bang Lin, Wiley Publications

2.**Hello Android: Introducing Google's Mobile Development**by Ed Burnette, 3rd Ed., 2010, The Pragmatic Programmers

3.**Mobile and Personal Communication System and Services**by Raj Pandya, Prentice Hall, Eastren Economy Edition

Reference Books

1.**Android Wireless Application Development**byLauren Darcey and Shane Conder, Pearson Education,2nd Edition

2.**Professional Android 4 Application Development**by Reto Meier, John Wiley & Sons

3.**Android User Interface Design: Turning Ideas and Sketches into Beautifully Designed Apps**by Ian G. Clifton

Unit 1

Introduction to Mobile Development

Learning Objectives:

After going through this unit , you will be able to understand:

- Mobile Generations
- Mobile Computing Architecture
- Devices 1G,2G,2.5G,3G
- How GSM Works and Its Architecture

1.1 Mobile Computing

Mobile computing can be defined as a human-computer interaction that enables transmission of voice, video and data. It comprises of mobile communication, mobile hardware, and mobile software.

Concept of Mobile Computing

Mobile computing works supported these three concepts-

- Mobile Communication.
- Mobile Hardware
- Mobile Software.

Three concepts of Mobile Computing.

1. **Mobile Communication**: Mobile communication represents the infrastructure covered within the wireless device that supports seamless and reliable communication. this can be inclusive of services, protocols, bandwidth and portals required for rendering services. It also defines the information format and prevents collision among other systems that deliver similar services. Mobile communication is created possible with radio-wave oriented infrastructure where signals are transmitted across the air to the recipient devices adept at receiving and sending similar signals.

2. **Mobile Hardware**: Mobile Hardware refers to device components or mobile devices that employ and deploys the service of mobility. It includes smartphones, tablet Pc's, portable laptops and private Digital Assistants. A receptor medium designed to sense and receive signals are installed in these devices. Configured to work fully duplex, signals are often sent and received simultaneously. Mobile Hardware operates on the wireless network.

3. **Mobile Software**: Mobile software is to blame for the operation of the device. It will be understood because the engine of the device. It handles the features and necessities of the mobile device. It may also be called because the software system of the gadget. thanks to its emphasis on portability, users aren't entitled to 1 geographical location but can operate the device from anyplace. Mobile Software encompasses all facilities of wireless communications enabling mobile computing.

1.2 Historical of Mobile Environments

Table 1.1 History of Mobile Generations

Year	Description
1982	First Generation (1G — analog voice only) systems with large heavy phones and poor network quality were introduced.
1992	Second Generation (2G) was deployed with improvements in signaling and hardware that were primarily aimed toward the voice market but, contrasting the first-generation systems, this generation used digital modulation to enhance call quality and enable new applications such as Short Messaging Services (SMS) and other low-data-rate (9.6 to 237 kbps) wireless applications.
2001	Third Generation (3G) was introduced, providing a significant jump over 2G, with much higher data rates (typically ranging from 400 Kbps to 16 Mbps), substantial increases in network voice capacity, along with better channel quality and — most important — support for advanced services and applications, including multimedia.
2012	Fourth Generation (4G) was deployed (some call it 4G-LTE). This was an all IP-network with increased speeds ranging from 21 Mbps to 1 gigabit speeds with wireless network latencies of 5 milliseconds. The wireless service providers were able to decrease network-per-megabyte costs with this new technology, while responding to increasing bandwidth demands from subscribers. One of the goals of LTE was to make the mobile Internet experience as good as or better than that offered by the wired broadband access systems deployed today.
2020	5G is on schedule to be deployed in this timeframe.

1.3 Early Mobile Phones to Smartphone's

In 1983, Motorola released its first commercial portable phone, called the Motorola DynaTAC 8000X. The handset offered half-hour of talk-time, six hours standby, and able to store 30 phone numbers. It also cost £2639 ($3995). Earlier the mobile space handsets weren't designed with consumers in mind. At the beginning of the 1990s this was still the case despite Nokia and NEC entering the fray. Nokia's first 'handheld' mobile, the Mobira Cityman 900, launched in 1989 and weighed just 800g – a large improvement over 1982's 9.8kg Mobira Senator model. 1990 to 1995 represented an upward swerve in design and portability, with mobile devices gradually commencing to appear within the hands of average consumers for the very first time. By the late-1990s, mobile devices were fast becoming the norm.

1989 – MOTOROLA MICROTAC 9800X

The Motorola MicroTAC had a reasonably long shelf life; it had been first introduced in 1989, then again went through some changes that moved it from an analogue phone to a GDM-compatible handset in 1994. It was eventually succeeded by Motorola's StarTAC in 1996, which was one in every of the primary "true" mobile-capable phones ever released	

1992 – MOTOROLA INTERNATIONAL 3200

The Motorola International 3200 was the world's first digital portable. just like the StarTAC, it had been GSM compatible. However, it had been never certified, so it may well be officially linked to a mobile network. The phone itself, which continues to be available in some places, as a collector's item, will actually work on 900MHz network.	

992 – NOKIA 1011

The Nokia 1011 was the world's first mass-produced mobile. It could hold 99 numbers and was originally released in November 1992. Due to technology inside the phone, the value of the Nokia 1011 was remarkably high – around €1796 in today's money! The Nokia 1011 was able to send and receive SMS messages, making it the world's first SMS-capable phone.	

1993 – BELLSOUTH/IBM SIMON PERSONAL COMMUNICATOR

The IBM Simon was built by IBM and was the first examples of a hand held, touchscreen device. Originally released in 1994, the IBM Simon went on to sell around 50,000 units total, between 1994 and 1995.
The battery on this communicator, however, only lasted an hour.

1996 – Nokia 8810 (AKA: "The Matrix Phone")

The Nokia 8810 was designed to be the foremost luxurious phone of all time. Nokia pulled out all the stops with this one, adding within the world's first internal antenna and support for 250 contacts. The 8810 would also hookup with early 2G networks, and it had been also the phone utilized by Neo's crew within the blockbuster film, The Matrix, making it a real classic in every sense of the word.

1996 – NOKIA 9000 COMMUNICATOR

The Nokia Communicator 9000 is essentially the world's first smartphone, because it ran on an Intel 24 MHz 1386 CPU and also had 8MB of RAM. it absolutely was heavy too, and it had a full QWERTY keyboard. It could do the following things
1.email, 2.texts and 3.make calls
three things that hardly any of the competition could manage within the mid-90s. It cost £1000 when it was first released within the UK.

1997 – Motorola StarTAC

Inspired by the communicator from Star Trek, this was the world's first clamshell handset. Another first for Motorola.

1997 – Nokia 6110

The Nokia 6110 was the first to use an ARM processor, and was only 1 of an extended line of 6xxx series phones from the Finnish company. The handset was targeted at the patron market, and featured things like improved call-quality and long battery life. it was the first mobile phone to support advanced graphical user interface.

Nokia 6110 Features:

- Three games: Memory, Snake, Logic
- Calculator, clock and calendar
- Currency converter
- Works as a pager
- Profile settings
- 4 colours

1998 – Nokia 5110

Outstanding battery, slim by 1998's standards, and it also included Snake.

Nokia 5110 Features:

- **Dimensions: 48 x 132 x 31 mm**
- **Battery: 900 mAh NiMH**
- **Display: 47 x 84 B/W**

1998 – NOKIA 9110I

The Nokia 9210i was the next in line model to the Nokia 9000 Communicator and it featured a TFT color display, a 32-bit ARM processor and the first instance of Symbian OS on a mobile phone.

1999 – BlackBerry 850

The BlackBerry 850 was the first handset released under the BlackBerry brand. Ten years later, RIM would be crowned the fastest growing company on the planet. And we all know what happened post-2010.

2000 – Nokia 3310

The phone that all of your mates had at school – if you went to school in the mid-to-late-90s, that is. Even in 2013, many regard the 3310 as one of the best mobile devices ever created. Some even say it's indestructible.	

2002 – Samsung SGH-T100

Before Samsung took over the world it made handsets like this, which was the first phone ever to use a thin-film transistor active matrix LCD display.	

2003 – NOKIA 1100

One of the most successful phone releases of all time. The Nokia 1100 was designed to be basic; it didn't feature a lot of the fancy stuff other Nokia phones had, and this made it very eye-catching to more basic users. The Nokia 1100 was officially developed for emerging countries like Nigeria and India. Nokia's one billionth phone was sold in 2005.	

2003 – NOKIA N-GAGE

The Nokia N-GAGE was the first "gaming phone". It ran on Symbian and featured a rather exceptional design, whereby the keys were set out on the side of the display. The phone was designed in an attempt to appeal away gamers from Nintendo's Game Boy Advanced. The handset also included support for apps and also MP3, making it a true smartphone. On sale between 2003 and 2005, Nokia sold about three million units of this device.	

2004 – Motorola Razr V3

Motorola shifted over a 130 million of its 'fashion' phone between the years 2004 and 2006, making it the best-selling clamshell handset in history.	

2006 – Nokia N95

A true smartphone, one that ran on Symbian, packed in a 332MHz Texas Instruments CPU, and feature 160MB of RAM. It also featured a decent 5-megapixel camera, Bluetooth, and Wi-Fi.	

2007 – LG Shine

LG Shine Features: • Dimensions: 99.8 x 50.6 x 13.8mm • Weight: 118g • Operating system: Java MIDP 2.0 • CPU: ARM9 115 MHz • Memory: 50 MB Internal, microSD (TransFlash) external memory card slot • Battery: 800mAh Li-Ion • Display: 240 x 320, 2.2-inch Display 262K-color TFT LCD • Camera: 2.0 megapixels + Autofocus	

2008 – Apple iPhone 3G

Apple's iPhone popularized applications with millions of consumers, helped make touchscreen interfaces the norm, and broke new ground for overall design and finish. Iphone is mainly responsible for changing the face of the mobile space forever.	

1.4 Tablets

A tablet, or tablet PC, may be a PC that uses a touchscreen as its primary data input device. Most tablets are slightly smaller and weigh less than the typical laptop. While some tablets include fold out keyboards, others, like the Apple iPad and Motorola Xoom, only offer touchscreen input.

Early tablet touchscreens were designed to operate with light pens, but latest tablets support human touch input. Many tablets now support multitouch input. It allows you to perform gestures with multiple fingers, like pinching a picture to zoom out, or spreading your fingers apart to centre.
Tablets without physical keyboards allow you to enter text employing a pop-up keyboard that appears on the screen.

Since tablets don't use a standard keyboard and mouse as their primary kinds of input, the interface of a tablet is different than a typical laptop. as an example, rather than double-clicking to open a program, most applications open with one tap. rather than clicking on a scroll bar to scroll through a window, most tablet applications allow you to swipe up and down anywhere within a window to scroll through the content.

Since tablet PCs provide a unique interface than traditional computers, they provide unique possibilities for graphics applications, games, and other programs. thanks to their small form factor, they're extremely portable and might be easily stored in a very backpack or a briefcase. Still, because tablets lack a keyboard and mouse, some tasks like typing documents and writing email messages, are harder on tablets than traditional computers. Therefore, tablets are generally seen as accessories to laptops and desktop computers instead of replacements.

Figure 1.2 Tablets

1.5 Mobile Computing Architecture

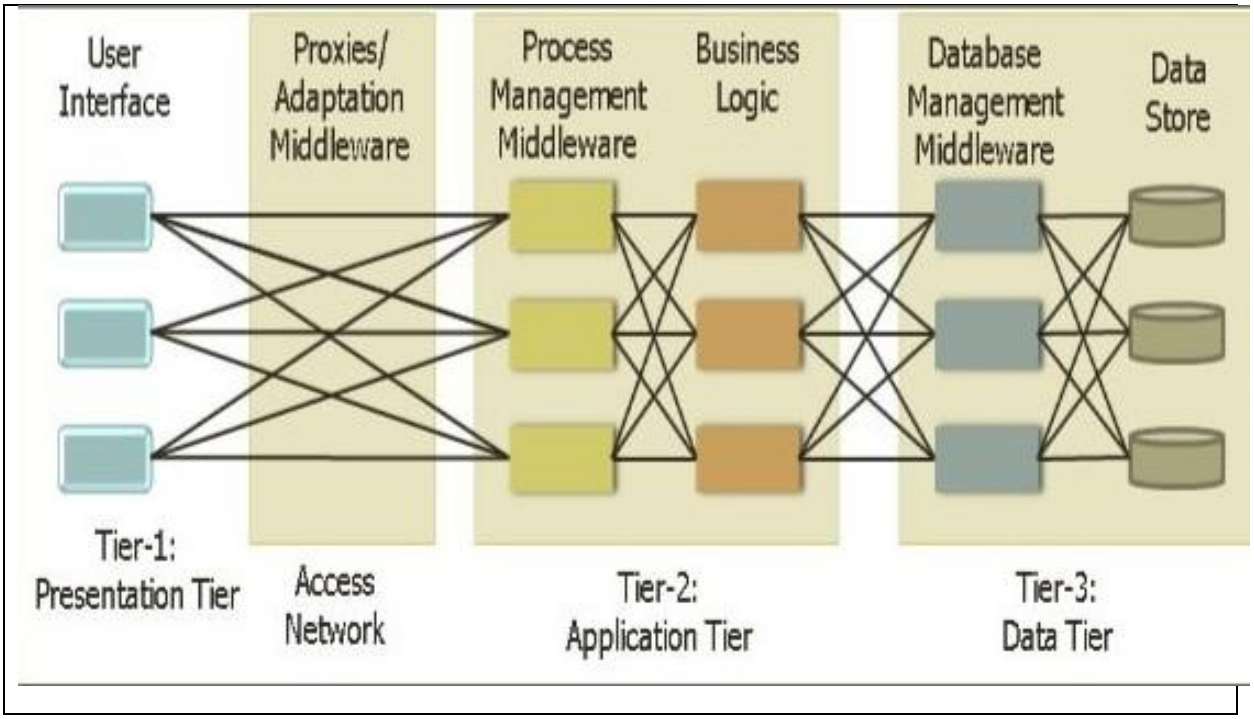


Figure 1.3 Mobile Computing Architecture

A 3-tier architecture is an application program that is organized into three major parts, comprising of:

- Bottom:- The data access layer,
- Middle :- The application tier (business logic)
- Top:- The client tier (presentation).

Each tier is distributed to a different place or places in a network. These tiers do not necessarily correspond to physical locations on various computers on a network, but slightly to logical layers of the application.

1.5.1. Presentation Layer (UI):

- This layer presents data to the user and optionally permits data manipulation and data entry, also this layer requests the data form Business layer.
- This layer accomplished through use of programming languages like Dynamic HTML and client-side data sources and data cursors.

1.5.2. Business Logic Layer:

- The business logic acts as the server for client requests from workstations. It acts according Business rules fetch or inserts data through the Data Layer.

- In turn, it determines what data is needed (and where it is located) and acts as a client in relation to a third tier of programming that might be located on a local or mainframe computer.
- Because these middle-tier components are not tied to a specific client, they can be used by all applications and can be moved to different locations, as response time and other rules require.

1.5.3. Data Access Layer:

- The third tier of the 3-tier system is made up of the Database Management System that provides all the data for the above two layers.
- This is the actual DBMS access layer.
- Minimizing dependencies on the storage mechanisms allows for updates or changes without the application tier clients being affected by or even aware of the change.

1.6 Mobile Generation

1.6.1 First Generation (1G)

First generation mobile networks were dependent upon analog radio systems. It meant that users could only make phone calls, they couldn't send or receive text messages. The 1G network was first introduced in Japan in 1979 before it absolutely was extended in other countries like the USA in 1980. so as to form it work, cell towers were built round the country which meant that signal coverage may be obtained from greater distances. However, the network was unreliable and had some security problems. for example, cell coverage would often drop, it might experience interference by other radio signals and because of a scarcity of encryption, it could easily be hacked. this suggests that with some tools, conversations might be heard and recorded.

1.6.2 Second Generation (2G)

The 1G network wasn't perfect, but it remained until 1991 when it had been replaced with 2G. This new mobile network worked on digital signal, not analog, which greatly improved its security & capacity. On 2G, users could send SMS and MMS messages and when GPRS was introduced in 1997, users could receive and send emails on the move.

1.6.3 Third Generation (3G)

Third generation mobile networks are still in use today, but normally when the superior 4G signal fails. 3G revolutionized mobile connectivity and therefore the capabilities of cell-phones. as compared to 2G, 3G was much faster and will transmit greater amounts of knowledge. With 3G users could do video call, share files, surf the net, watch TV online and play online games on their mobiles. Under 3G, cell-phones were not nearly only for calling and texting, they were the hub of social connectivity.

1.6.4 Fourth Generation (4G)

The introduction of 4G went one step further than the revolutionary 3G. It's five times faster than the 3G network – and can in theory provide speeds of up to 100Mbps. All mobile models released from 2013 onwards should support this network, which can offer connectivity for tablets and laptops as well as smartphones. Under 4G, users can experience better latency, higher voice quality, easy access to instant messaging services and social media, quality streaming and make faster downloads.

1.6.5 Fifth Generation (5G)

The 5G network is yet to be released but is widely anticipated by the mobile industry. Many experts claim that the network will change not just how we use our mobiles, but how we connect our devices to the internet. The better-quality speed and capacity of the network will signal new IoT trends, such as connected cars, smart cities and IoT in the home and office.

Mobile network operators claim that 5G will be available by 2020 but nothing is certain just yet.

1.7 Handoff

When a mobile user is engaged in conversation, the Mobile Station (MS) is linked to Base Station (BS) via radio link (communication system). If the user moves to the coverage area of another BS, the link to old BS is disconnected and link to new BS is established to continue conversation. This process is termed automatic link transfer or handoff. Depending on the mobility of MS, the handoff is split into two categories:

1.7.1Inter-Base Station Handoff /Inter Cell Handoff

Here Mobile Station (MS) usually moves from one Base Station (BS) to another Base Station (BS) under one Mobile Switching Center (MSC).

Following are the series of Action taken for Communication;

1. The MS momentarily suspends conversation & starts the hand-off procedure by picking a channel in new BS. Then it resumes the conversation in old BS.

2. When MSC receives that signal, it transfers the information to the new BS & sets up new conversation path to MS through that channel.

3. After MS has been transferred to new BS, it starts the conversation channel with new BS & then MSC disconnect the link with old BS.

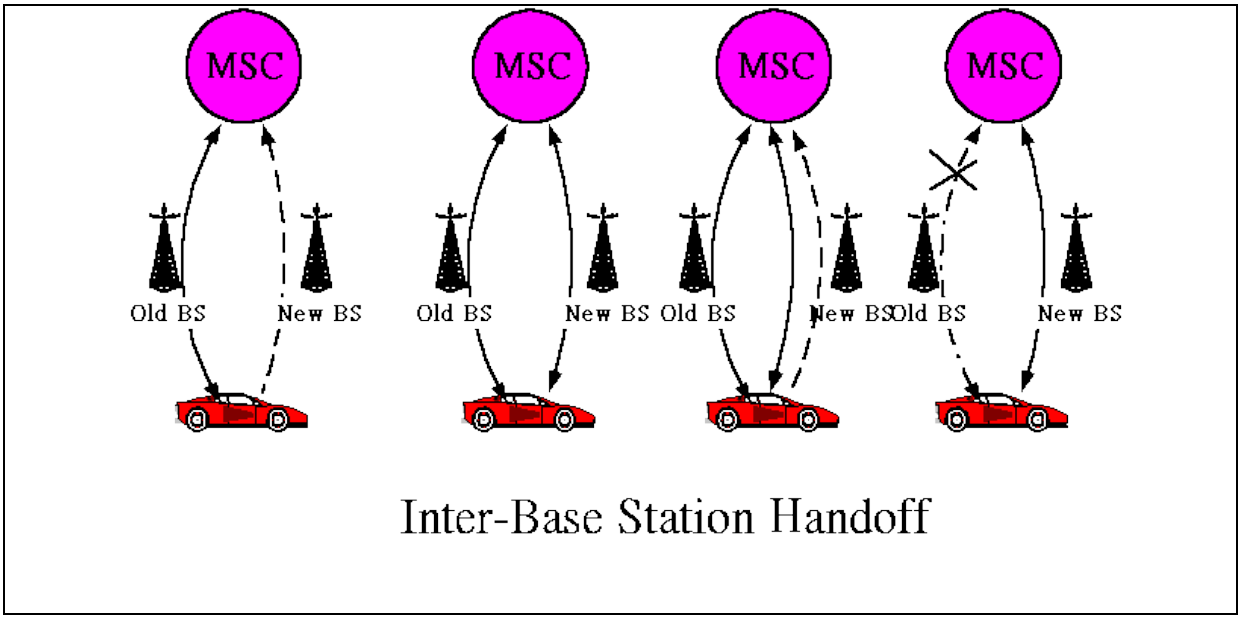


Figure 1.4 Inter-BS link Transfer

1.7.2Inter-System Handoff/Inter-MSC Handoff

MS moves from one BS to another connected to two different MSCs. Following are the series of Action taken for Communication;

1.MSC1 requests MSC2 to perform handoff measurement on the call in progress.

2.MSC2 then selects a BS by interrogating the signal quality and sends the information to MSC1.

3.Then MSC1 asks MSC2 to setup a voice channel.

4.Assuming that a voice channel is available in BS2.MSC2 instructs MSC1 to start radio link transfer.

5.MSC1 sends the MS a handoff order. Now MS can access BS2 of MSC2.MSC2 informs MSC1 that handoff is successful.MSC1 then connects call path to MSC2.

6.In the intersystem handoff process, anchor MSC is always in call path before & after handoff.

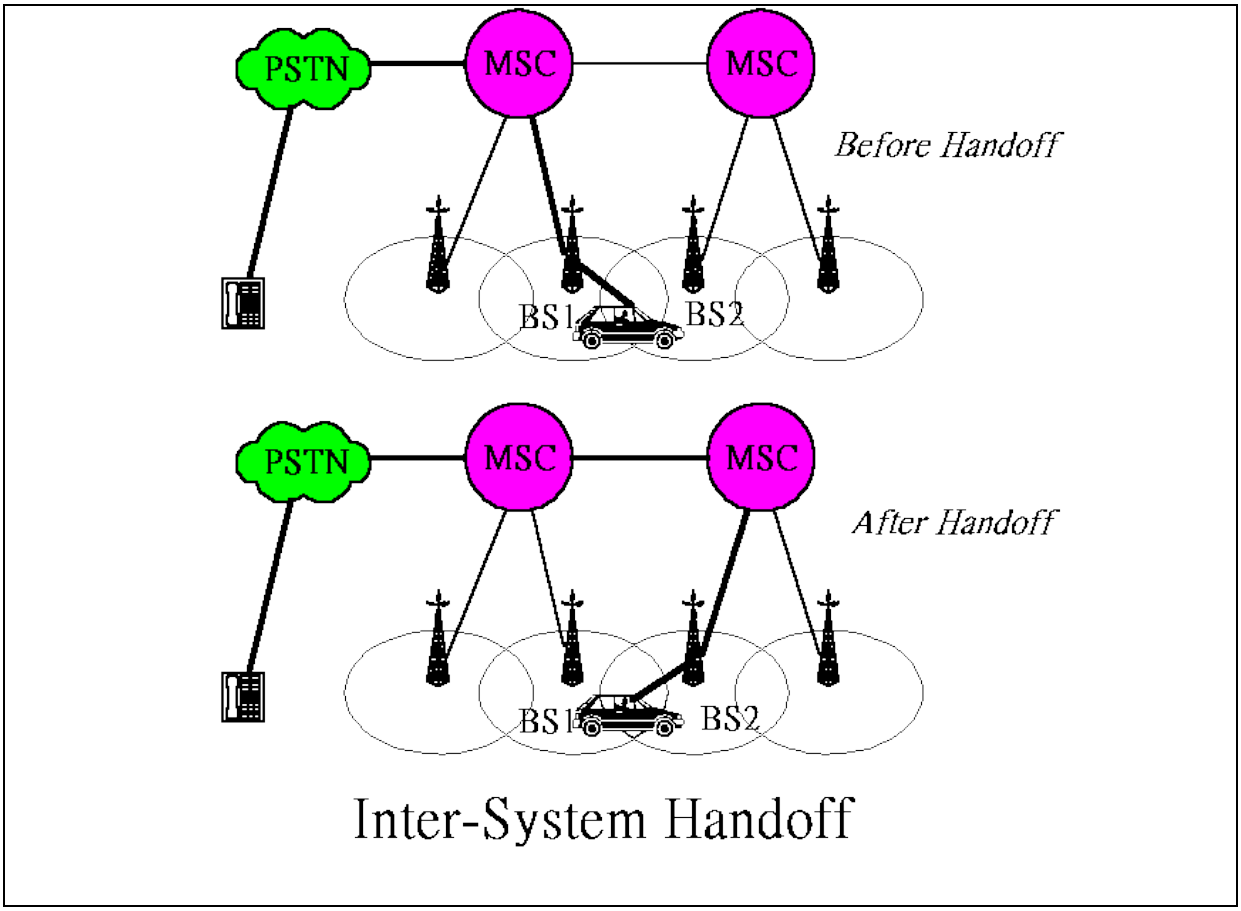


Figure 1.5 Inter System Handof

1.8 Roaming

ROAMING: When a mobile user moves from one PCS system to another, then the system should be informed of the current location of the user. Otherwise it is impossible to deliver services. Two basic operations are performed under roaming management

1. **Registration (location update):** Where MS informs the system its currentlocation.
2. **Location tracking**: Process during which a system locates MS. Locationtracking is required when n/w attempts to deliver call to a mobile user.

The roaming management follows a two level strategy where two tier systems of home and visited databases are used. When a user subscribes to the services of a network, a record is created in the system's database called HLR. This is referred to as home
system of the mobile user. HLR is a n/w database, where MS's identity, profile, current location & validation period is stored.

When the mobile user visits a new network other than home system, a temporary record for the mobile user is created in the VLR of visited system. VLR momentarily stores information for visiting subscribers so that corresponding MSC can provide service.

Registration Procedure includes following steps:

1. When mobile user enters into new PCS n/w, it must register in VLR of new system.
2. The new VLR informs mobile user's HLR regarding the current location & address of user. The HLR sends an acknowledgement which includes MS's profile, to the new VLR.
3. New VLR informs MS about successful registration.
4. HLR sends a deregistration message to cancel the location record of MS in old VLR. The old VLR acknowledges the deregistration.

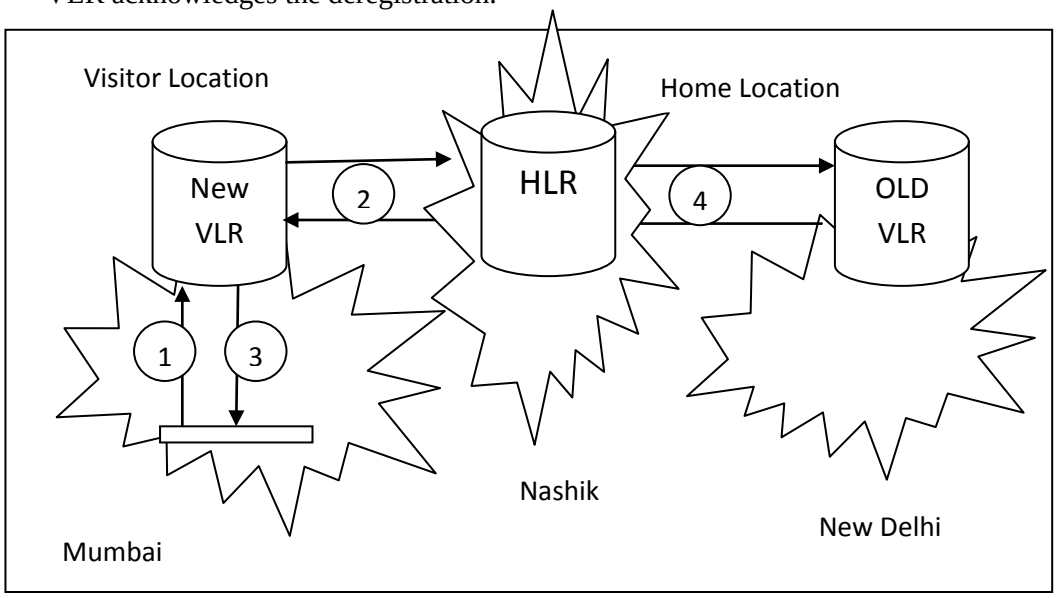


Figure 1.6 MS registration Process

To originate a call, MS first contacts with MSC in the new PCS n/w .The call request is forwarded to VLR for approval. If it is approved, MSC sets up the call to the user following the standard PSTN procedures.

1. If a wire line phone attempts to call a mobile subscriber, the call is forwarded to switch called the originating switch in PSTN. The switch passes a query to HLR to find current VLR of MS. The HLR queries the VLR in which MS resides to get a communicable address.
2. The VLR returns the address to change through HLR.
3. Based on address, a communication link is established between MS through visited MSC.

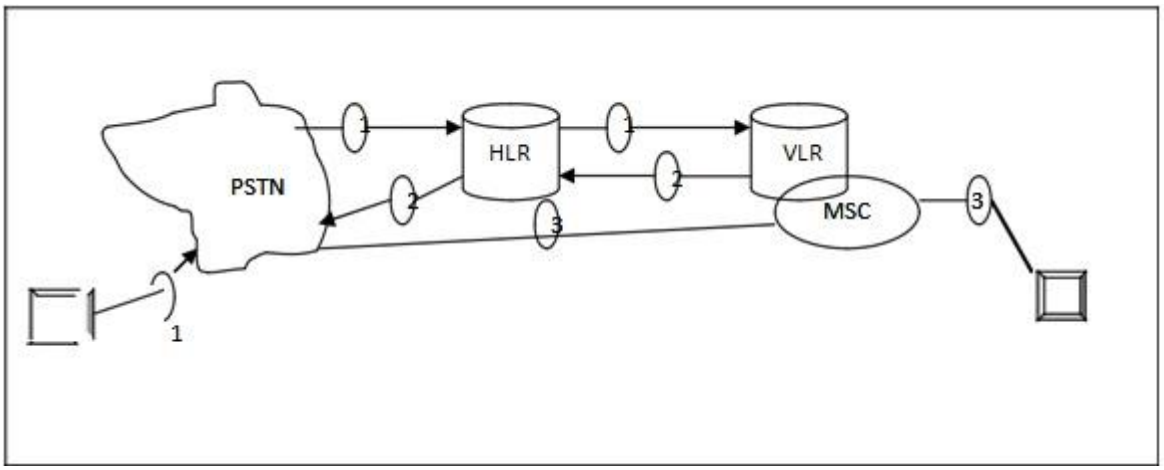
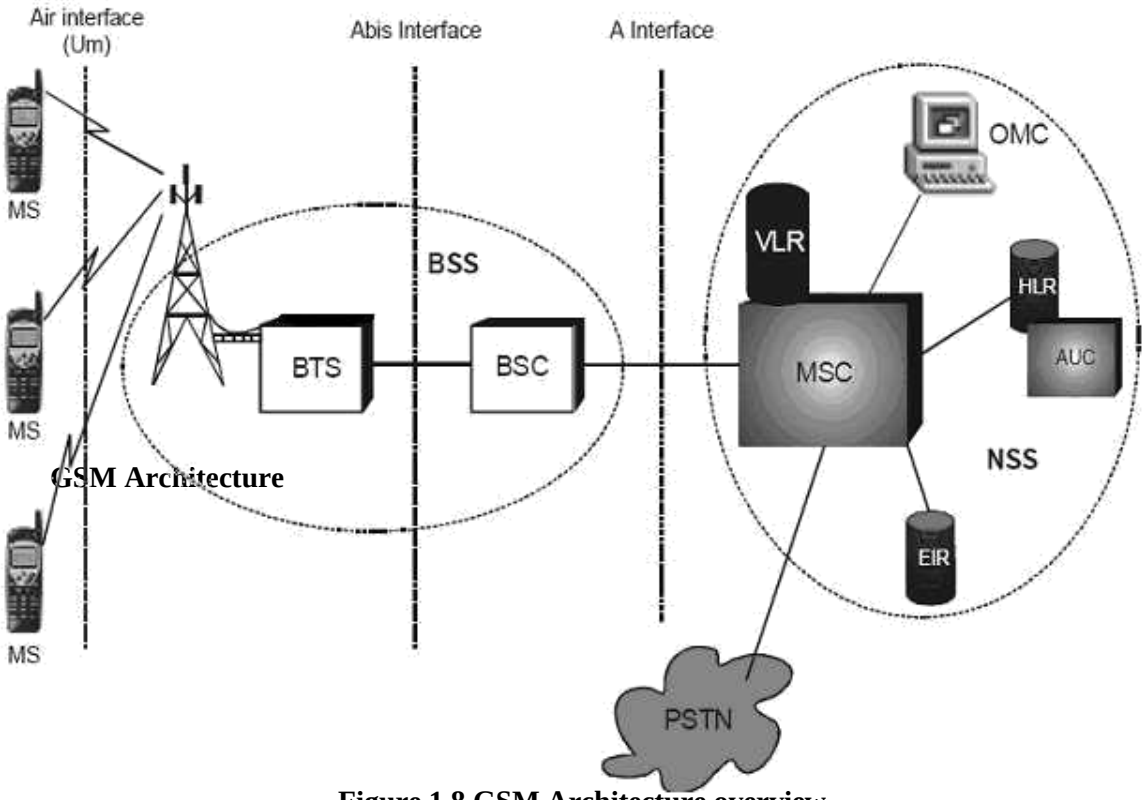


Figure 1.7 Call Delivery Procedure

1.9 GSM (Global System for Mobile Communication)

GSM is the most popular standard for mobile phones in the world. It is a digital mobile telephony system that is widely used in Europe and other parts of the world.
In 1982, the European Conference of Postal and Telecommunications Administrations(CEPT) created the Group Special Mobile (GSM) to develop a standard for a mobile telephone system that could be used across Europe. In 1989, GSM responsibility was given to the European Telecommunications Standards Institute (ETSI) and phase I of the GSM specifications were published in 1990.



GSM Architecture

Figure 1.8 GSM Architecture overview

Table 1.2 Abbreviations Associated with GSM

Abbreviations	
MSC :	Mobile switching center
BSC :	Base station controller
BTS :	Base transceiver station
TRX :	Transceiver.
MS :	Mobile station
OMC :	Operations and Maintenance centre.
PSTN	Public switched telephone network.
BSS :	Base station sub-system.
HLR :	Home location register
VLR :	Visitor locations register

AUC :	Authentication centre
EIR:	Equipment Identity Register

The first GSM network was launched in 1991 by Radiolinja in Finland with joint technical infrastructure maintenance from Ericsson. The proposed GSM system had to meet certain business objectives:

- Support for International Roaming
- Good Speech Quality
- Ability to support handheld terminals
- Low terminal and service cost.
- Spectral Efficiency

GSM uses a combination of FDMA and TDMA. The GSM system has an allocation of 50 MHz bandwidth in the 900 MHz frequency band. Using FMA, this band is divided into 124 channels each with a carrier bandwidth of 200 KHz. Using TDMA, each of these channels is further divided into 8 time slots. Therefore with combination of FDMA and TDMA we can realize a maximum of 992 channels for transmit and receive.

Cell: Cell is the basic service area: one BTS covers one cell. Each cell is given aCell Global Identity (CGI), a number that uniquely identifies the cell.

Location Area: A group of cells form a Location Area. This is the area that is pagedwhen a subscriber gets an incoming call. Each Location Area is assigned a Location Area Identity (LAI). Each Location Area is functioned by one or more BSCs.
GSM network can be divided into 4 groups.

1.9.1 MS (Mobile Station)

An MS is used by a mobile subscriber to communicate with the mobile network. Several types of MSs exist, each allowing the subscriber to make and receive calls. Manufacturers of MS offer a variety of design and features to meet the need of different market.

The mobile station consists of:

- Mobile Equipment (ME)
- Subscriber identity module (SIM)

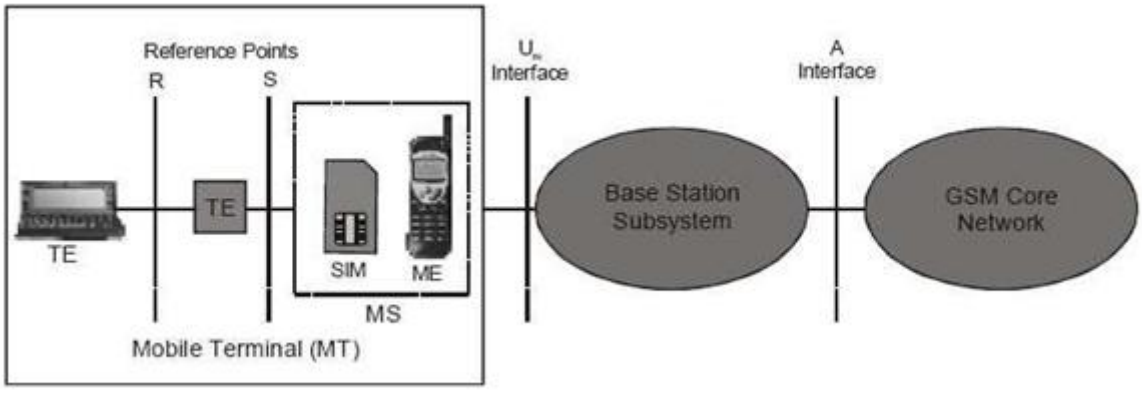


Figure 1.9. GSM Mobile Terminal

1.9.2 ME (Mobile Equipment):

ME Stands for “Cellular phone without SIM card” The mobile equipment has a unique international mobile equipment identity (IMEI) which is used by EIR. The numbers of GSM terminal types are defined within the GSM specification. They are distinguished primarily by their power output rating. The range or coverage area of an MS is dependent on the output power capabilities and accordingly different ranges. For example, hand held MSs have a lower output power and shorter range than car-installed MSs with a roof mounted antenna

1.9.3 SIM (Subscriber Identity Module)

1.9.3 SIM (Subscriber Identity Module) SIM card employed in phones are smart processor cards. It possesses a processor and a little memory. The SIM stores permanent and temporary data about the mobile, the subscriber and also the network. It contains a serial no, PIN, PUK (Pin Unblocking Key), an authentication key (Ki), IMSI (International Mobile Subscriber Identity).
The SIM may be plugged into any GSM mobile terminal. This brings the benefits of security and portability for subscriber. Example: Subscriber A’s mobile terminal may be stolen. However, A’s own SIM is utilized in another person’s mobile terminal and also the calls are charged to subscriber A. Functions of MS Function of MS is transmission of signal from MS to BTS (using uplink) and reception of signal from BTS to MS (using down link).

1.9.4 BSS (Base Station Subsystem)

BSS contains two components:

- BTS
- BSC

1.9.5 BTS (Base Transceiver Station):

Itcomprises all radio equipment’s (e.g.: antenna, signal processing & amplifier required for transmission).It is placed in the center of a cell. Its transmitting power defines the size of a cell. It is connected to MS via Um interface and connected to BSC via Abis Interface. It manages the radio resources for BTSs. It handles & handover the radio frequency, radio channel set up from one BTS to other.

1.9.6 BSC (Base Station Controller)

It connects the BTS and MSC of NSS. It manages radio resources for one or more BTS. It handles and Handover the radio frequency, radio channel setup from one BTS to another.

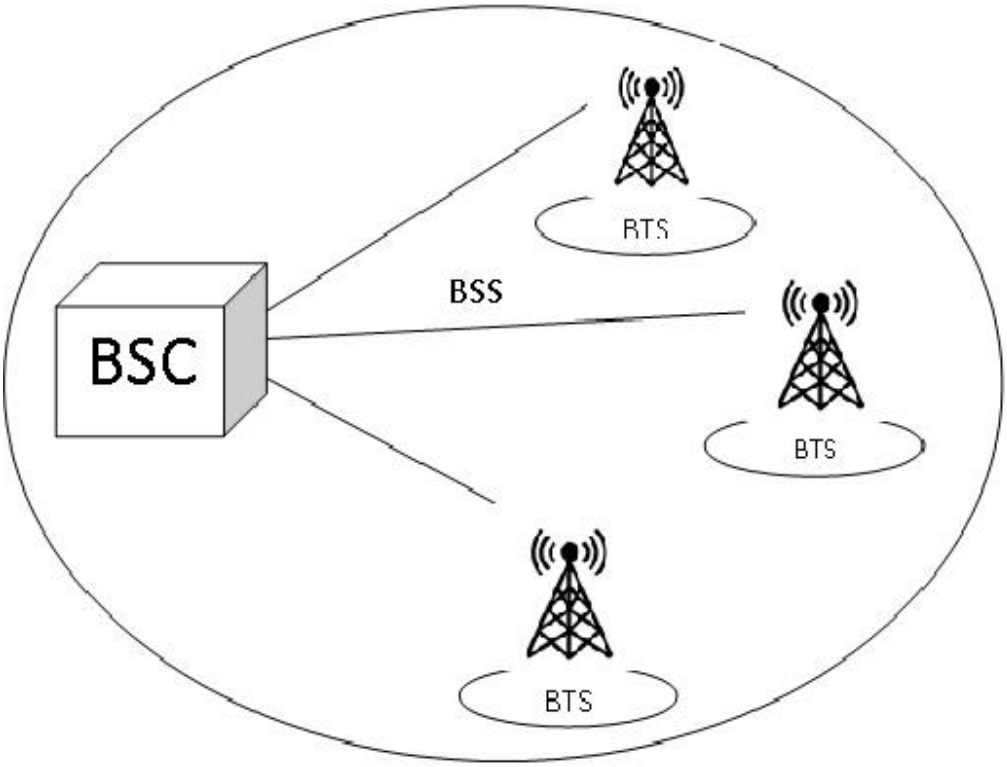


Figure 1.10 NSS (Network Switching Subsystem)

The NSS combines the call rotating switches (MSC and GMSC) with data base registered required to keep track of subscriber's movements and use of the system. Key elements of NSS are:

- MSC
- VLR
- HLR

1.9.7 MSC (Mobile Switching Centre)

The mobile-services switching center is an exchange which performs all the switching and signaling functions for mobile stations located in a geographical area, designated as the MSC area. These are high performance digital ISDN switches. It is used for connection between mobile phone to mobile phone within same network. It is used for connection between mobile phone to fixed phone within a network. It manages BSC within a geographical area.

1.9.8 GMSC (Gateway MSC)

Connection for another network MSC handles all the signaling needed for connection set up and connection release.

1.9.9 HLR (Home Location Register)

The HLR is a centralized network data base that stores and manages all mobile services belonging to a specific operator. It acts as a permanent store for a person's subscription information. It provides call routing and roaming facility by combining with MSC and VLR. It is considered as a Database which

stores the information about the subscriber within covering area of MSC. Information includes current location of the mobile & all the service providing information, when a phone is powered off this information is stored in HLR. It is also a database but contains a temporary copy of some of important information stored in HLR. If a new MS user comes into location area, then VLR will provide relevant information by bringing it from HLR.

1.9.10 VLR (Visitor Location Resister)

It is a temporary storage device of GSM network. It stores subscribers' subscription information for MS which are within the particular MSC service Area. There is one VLR for each MSC service area

1.9.11 OSS (Operation and Support Subsystem)

It contains necessary function for network operation and maintenance. Key Elements are as follow

- OMC
- EIR
- AUC

1.9.12 OMC (Operation andmaintenance center)

It is connected to different components of NSS & to BSC. It controls the traffic load of BSS.

1.9.13 EIR (Equipment Identity Register)

A database that contains an list of all valid mobile equipment within the network where each MS is identified by IMEI (International Mobile Equipment Identity).EIR contains an inventory of IMEI of all valid terminals. An IMEI is marked invalid if it's stolen. EIR allows the MSC to forbid calls from this stolen terminal. The equipment identification procedure uses the identity of the equipment itself (IMEI) to confirm that the MS terminal equipment is valid.

1.9.14 AUC (Authentication Center)

It is defined to protect user identity & transmission. It is a protected database and stores a copy of secret information stored in SIM card .These data help to verify user's identity.

1.10 Network Signaling

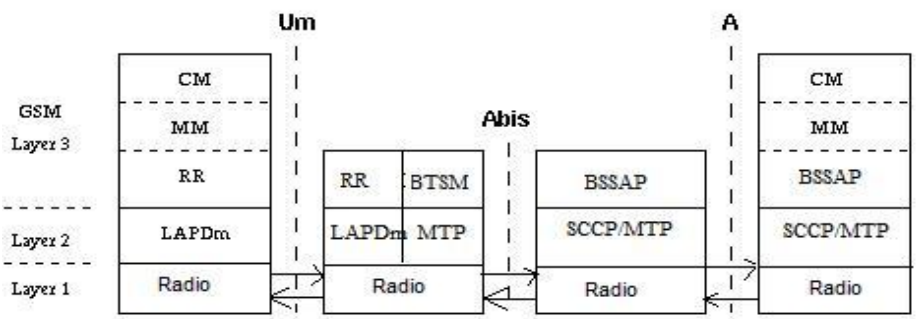


Figure 1.11: Network Signaling

Table 1.3 Abbreviations Associated with Network Signaling

Abbreviation	
LAPD	Link Access Procedure D-Channel Managed
RR	Radio Resource
MM	Mobility Management
CM	Call Management
BTSM	BTS Management
BSSMAP	BSS Application Protocol
SCCP	Signaling Connection Control Part

The signaling protocol in GSM is structured into 3 layers.

- Layer 1 Physical Layer
- Layer 2 Data Link Layer
- Layer 3 Network Layer

1.10.1 MS & BTS

The physical layer between MS & BTS is called Um interface. It performs following functions

- Full or half duplex access.
- Provides TDMA, FDMA, and CDMA.
- Framing of data.

The data link layer controls the flow of packets to and from network layer and provides access to various services like:

Connection: Provides connection between two terminals.

Teleservices -Services offered by a mobile network to users like: MMS, SMS, etc

The data link layer present between MS & BTS is LAPDm (Link Access Protocol managed). LAPDm protocol describes the standard procedure in GSM for accessing D-channel Link. Its functions are:

Its functions are:

- Dataflow control.
- Acknowledged / unacknowledged data Transmission.
- Address and sequence no. check.
- Segmentation.

The network layer has 3-sublayers

1.10.2 CM (Call Management)

Supports call establishment, maintenance, termination.
It supports SMS.
Support DTMF (Dual Tone multiple frequency) signaling.

1.10.3 MM (Mobility Management)

Control the issue regarding mobility Management, location updating & registration.

1.10.4 RRM (Radio Resource Management.)

It manages radio resources such as: frequency assignment, signal measurement.

1.10.5 BTS & BSC signaling protocols

The physical layer between BTS & BSC is called Abis interface, where voice is coded by using 64kbps PCM. The connection between BTS and BSC is through a wired network. The data link layer is LAPDm. Network Layer protocol is called BTS Management which interact with BSSAP.

1.10.6 BSC & MSC signaling protocol

Physical layer between BSC & MSC is called U interface. Data link layer protocol between BSC & MSC is MTP (Message Transfer Protocol) & SCCP (Signaling Connection Control Protocol). MTP and SCCP are part of the SS7 (Signaling System No7) used by interface A. NETWORK layer protocols at the MSC are CM, MM and BSSAP (Base Subsystem Application Part).

1.11 GSM INTERFACES

1.11.1 Um Interface (MS to BTS)

The Um radio interface (between MS and base transceiver stations [BTS]) is the most vital in any mobile radio system. It addresses the demanding characteristics of the radio environment. The

physical layer interfaces to the information link layer and radio resource management sublayer within the MS and BS and to other functional units within the MS and network subsystem for supporting traffic channels. The physical interface comprises a collection of physical channels accessible through FDMA and TDMA.

1.11.2 Abis Interface (BTS to BSC)

The interconnection between the BTS and the BSC is through a standard interface, Abis. The first functions carried over this interface are traffic channel transmission, terrestrial channel management, and radio channel management. This interface supports two styles of communications links: traffic channels at 64 kbps carrying speech or user data for a full- or half-rate radio traffic channel and signaling channels at 16 kbps carrying information for BSC-BTS and BSC-MSC signaling. The BSC handles the LAPD channel signaling for each BTS carrier. There are two varieties of messages handled by the traffic management procedure a part of the signaling interface—**transparent** and **nontransparent**. Transparent messages are between the MS and BSC-MSC and don't require analysis by the BTS. Nontransparent messages do require BTS analysis

1.11.3 A Interface (BSC to MSC)

The A interface allows interconnection between the BSS radio base subsystem and the MSC.The physical layer of the A interface is a 2-Mbps standard Consultative Committee on Telephone and Telegraph (CCITT) digital connection. The signaling transport uses Message Transfer Part (MTP) and Signaling Connection Control Part (SCCP) of SS7. Error-free transport is handled by a subset of the MTP, and logical connection is handled by a subset of the SCCP. The application parts are divided between the BSS application part (BSSAP) and BSS operation and maintenance application part (BSSOMAP). The BSSAP is further divided into Direct Transfer Application Part (DTAP) and BSS management application part (BSSMAP). The DTAP is used to transfer layer 3 messages between the MS and the MSC without BSC involvement. The BSSMAP is responsible for all aspects of radio resource handling at the BSS. The BSSOMAP supports all the operation and maintenance communications of BSS.

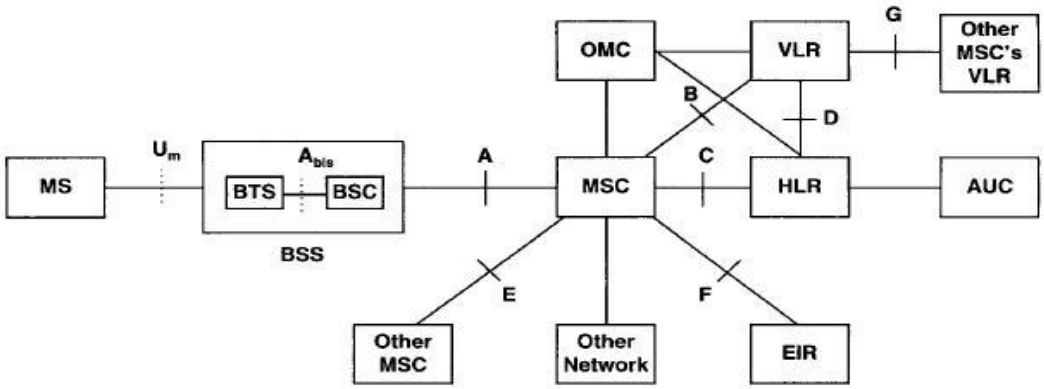


Figure 1.12 **I**nterfaces between the GSM entities.

1.12 GSM Channels

GSM has been allocated an operational frequency from 890 MHz to 960 MHz. GSM uses the frequency band 890 MHz-915 MHz for uplink (reverse) transmission, and for downlink (forward)

transmission, it uses be frequency band 935 MHz-960 MHz. The available 25-MHz spectrum for each direction is divided into 124 Frequency Division Multiplexing (FDM) channels, each occupying 200 kHz with 100 kHz guard band at two edges of the spectrum as shown in fig.

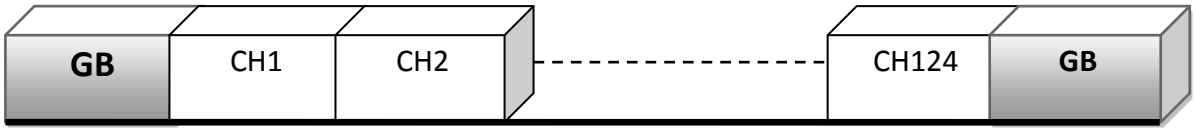


Figure 1.13 GSM Channels

Table 1.4 Logical Channels in GSM

Channel type	***Channel group***	***Channel***	***Direction***
Control Channel (CCH)	Broadcast Channel (BCH)	Broadcast Control Channel (BCCH)	Downlink
		Frequency Correction Channel (FCCH)	Downlink
		Synchronization Channel (SCH)	Downlink
	Common control Channel (CCCH)	Paging Channel (PCH)	Downlink
		Random Access Channel (RACH)	Uplink
		Access Grant Channel (AGCH)	Downlink
	Dedicated control Channel (DCCH)	Standalone Dedicated Control Channel (SDCCH)	Uplink and Downlink
		Slow Associated Control Channel (SACCH)	Uplink and Downlink
		Fast Associated Control Channel (FACCH)	Uplink and Downlink
Traffic Channel (TCH)	Traffic Channel (TCH)	Half-rate Traffic Channel (TCH/H)	Uplink and Downlink
		Full-rate Traffic Channel (TCH/F)	Uplink and Downlink

The logical channels in the GSM network are divided into two principal categories: Control Channels (CCHs) and Traffic Channels (TCHs). Control channels carry signaling and synchronizing commands between the base station and the mobile station. Traffic channels carry digitally encoded user speech or user data and have identical functions and formats on both the forward and reverse link. GSM system uses a variety of logical control channels to ensure uninterrupted communication between MSs and the BS.

1.12.1 GSM Control Channels

There are three classes of control channels defined in GSM: Broadcast Channels (BCH), Common Control Channels (CCCH) and Dedicated Control Channels (DCCH). Each control channel consists of several logical channels that arc distributed in time to provide the necessary GSM control functions

I. Broadcast Channel (BCH)

The BCH channels are broadcast from the BTS to MSs in the coverage area of the BTS and are one way channels. The broadcast channel operates on the forward link of a specific ARFCN within each

cell and transmits data only in the first time slot of certain GSM frames. The BCH provides synchronization for all mobiles within the cell and is occasionally monitored by mobiles in neighboring cells. There are three separate broadcast channels:

1. Broadcast Control Channel (BCCH):

This channel is used by BTS to broadcast system parameters such as frequency of operation in the cell, operator identifiers, cell ID and available services to all the MSs. Once the carrier, bit, and frame synchronization between the BTS and MS are established, the BCCH informs MS about the environment parameters associated with the BTS covering that area such as current a channel structure, channel availability, and congestion. The BCCH also broadcasts a list of channels are currently in use within the cell.

2. Frequency Correction Control Channel (FCCH):

This is used by the BTS tobroadcast frequency references and frequency correction burst of 148 bits length. An MS in the coverage area of a BTS uses broadcast FCCH signal to synchronize its carrier frequency and bit timing.

3. Synchronization Channel (SCH):

This channel is used by the BTS to broadcast frame synchronization signals containing the synchronization training sequences burst of 64 bits length to all MSs. Using SCH, MSs will synchronize their counters to specify the location of arriving packets in the TDMA hierarchy. SCH is broadcast in Time Slot 0 of the frame immediately following the FCCH frame and is used to identify the serving base station while allowing each mobile to frame-synchronize with the base station.

II. Common Control Channels (CCCH)

The Common Control Channels (CCCH) are one-way channels used for establishing links between the and the BS for any ongoing call management. CCCHs are the most commonly used control channel and are used to page specific subscribers, assign signaling channels to specific users, and receive mobile requests for service. There are three CCCH logical channels:

1.12.2Paging Channel (PCH):

This is a forward link channel and is used by the BTS topage or notify a specific individual MS for an incoming call in the cell. The PCH transmits the IMSI of the target subscriber, along with a request for acknowledgment from the mobile unit on the RACH.

1.12.3 Random Access Channel (RACH):

This is a reverse link channel and is usedby the MS either to access the BTS requesting the dedicated channel for call establishment or to acknowledge a page from the PCH. The RACH is used with implementation of a slotted-ALOHA protocol, which is used by MSs to contend for one of the available slots in the GSM traffic frames. The RACH is implemented on the short Random Access Burst (RAB).

I. Dedicated Control Channels (DCCH)

Dedicated Control Channels (DCCH) are two-way channels having the same format and function on both the forward and reverse links, supporting signaling and control for individual mobile subscribers. These are used along with voice channels to serve for any control information transmission during actual voice communication. There are three DCCH logical channels:

1.Stand-alone Dedicated Control Channel (SDCCH):

This is a two-way channel allocated with SACCH to mobile terminal to transfer network control and signaling information for call establishment and mobility management. The SDCCH ensures that the mobile station and the base station remain connected while the base station and MSC verify the subscriber unit and allocate resources for the mobile. The SDCCH is used to send authentication and alert messages as the mobile synchronizes itself with the frame structure and waits for a TCH.

2. Slow Associated Control Channel (SACCH):

This is a two-way channelassociated with a TCH or a SDCCH and maps onto the same physical channel. The SACCH is used to exchange the necessary parameters between the BTS and the MS during the actual transmission to maintain the communication link. Each ARFCN systematically carries SACCH data for all of its current users. The gross data rate of the SACCH channel is half of that of the SDCCH. On the forward link, the SACCH is used to send slow but regularly changing control information to the mobile subscriber. The reverse SACCH carries information about the received signal strength and quality of the TCH.

3.Fast Associated Control Channel (FACCH):

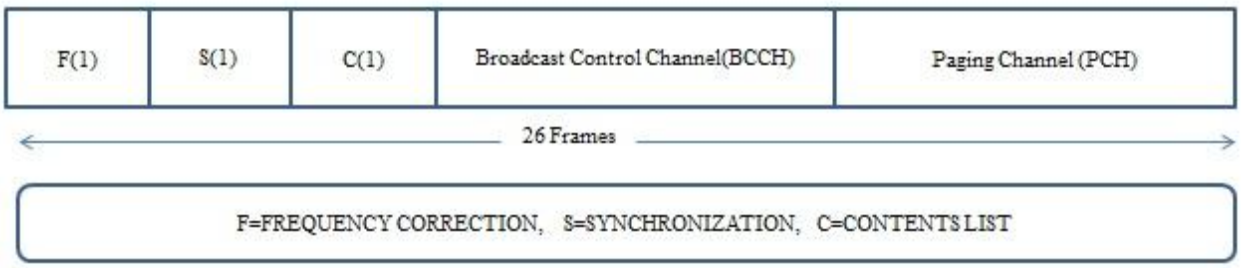
This is a two-way channel used tosupport fast transitions such as a hand-off request in the channel when SACCH is not adequate. The FACCH is physically multiplexed with the TCH or SDCCH to provide additional support to the SACCH. FACCH is not a dedicated control channel but carries the same information as SDCCH. FACCH is a part of the traffic channel, while SDCCH is a part of the control channel.

Figure 1.13 FACCH Channel

Control information in GSM is mainly on two logical channels—the Broadcast Channel (BCCH) and the Paging Channel (PCH). The broadcast information is transmitted first, followed by paging information. Following Figure shows the structure of a GSM logical control channel.

1.13 Mobility Management in GSM

Mobility Management function handles the function that arises due to mobility of the subscriber.



Main objective of MM is location tracking & call set up. The current location of an MS is maintained

by a 2-level hierarchical strategy with HLR & VLR. When an MS visits a new location it must register in the VLR of visited location. The HLR must be informed about the registration. The registration process of MS moving from one VLR to another VLR follows following steps.

STEP-1.MS periodically listens to the BCCH (Broadcast Control Channel) broadcastfrom BSS. If the MS detects that it has entered into a new location area, it sends a registration message to the new VLR by using SDCCH (Standalone Dedicated Control Channel) channel.
SDCCH: Used only for signaling & short message.

BCCH: Provides system information.
STEP-2.The new VLR communicates with old VLR to find HLR of MS. The new VLRthen performs authentication process.
STEP-3.After MS is authenticated, new VLR sends a registration message to HLR. Ifthe registration request is accepted, the HLR provides new VLR with all relevant subscriber information.
STEP-4.The new VLR informs the MS of successful registration.
STEP-5. Then the HLR sends a deregistration (Cancellation) message to old VLR. Theold VLR cancels the record for MS & sends an acknowledgement to the HLR regarding cancellation.

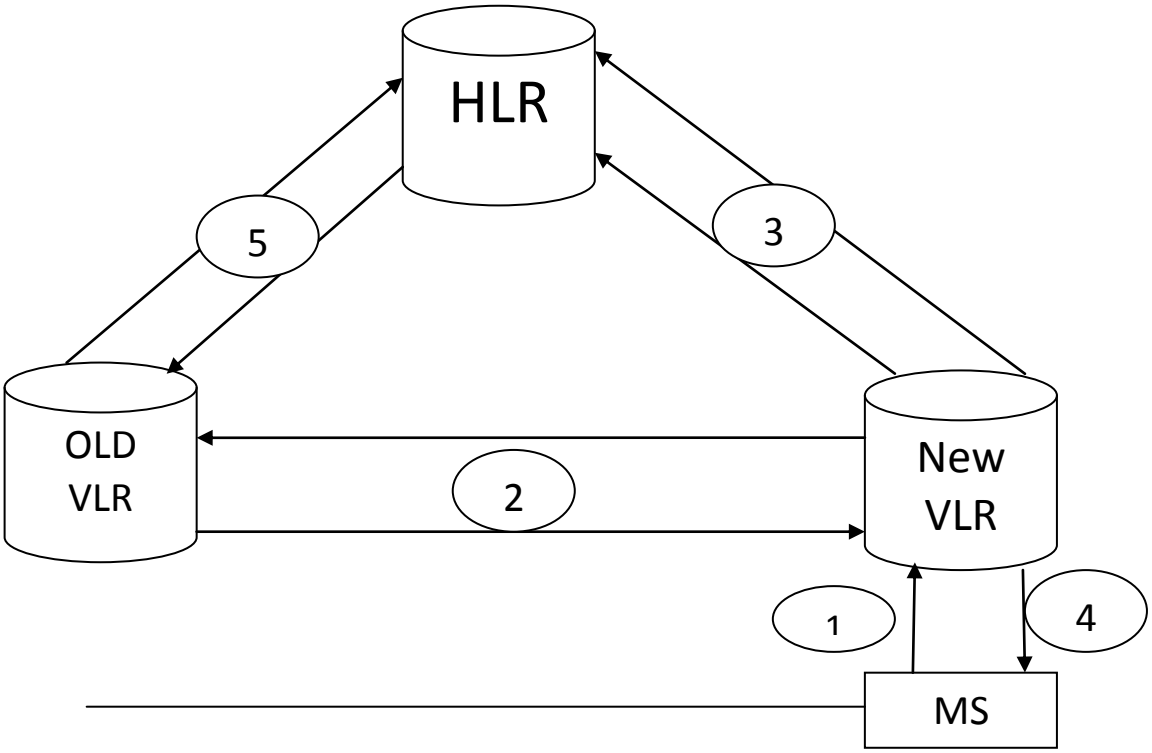


Figure 1.14: GSM Mobility Management

1.13.1 GSM Call Origination

1. MS sends the call origination request to MSC.
2. MSC forwards the request to VLR by sending **MAP_SEND_INFO_**
3. **FOR_OUGOING_CALL.**
4. VLR checks MS's profile & sends an ACK to MSC to grant call request.
5. MSC sets up communication link according to standard PSTN call set up procedure.

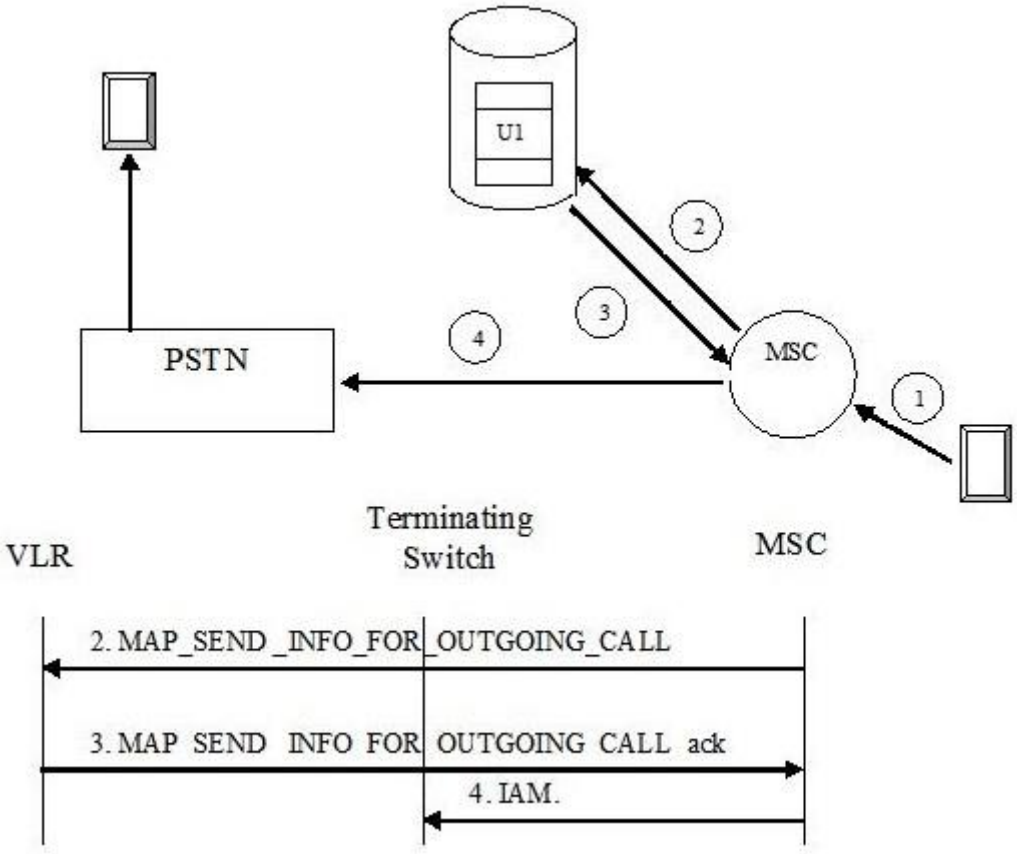


Figure 1.15 Call Origination Operation

1.13.2 Call Termination

When mobile station number is dialed by PSTN user, call is routed to GMSC by IAM (Initial Addressing Message) message.

1. To obtain routing information, GMSC interrogates HLR by sending M**AP_SEND_ROUTING_INFORMATION** to HLR.
2. HLR sends a **MAP_PROVIDE_ROAMING_NUMBER** message to VLR to obtain MSRN (MS Roaming Number). The message consists of IMSI, MSC number etc.
3. The VLR creates the MSRN by using MSC number stored in VLR record of MS. The MSRN no is sent back to GMSC through HLR.
4. MSRN provides address of target MSC where the MS resides. Then a message is directed from GMSC to target MSC to set communication link.

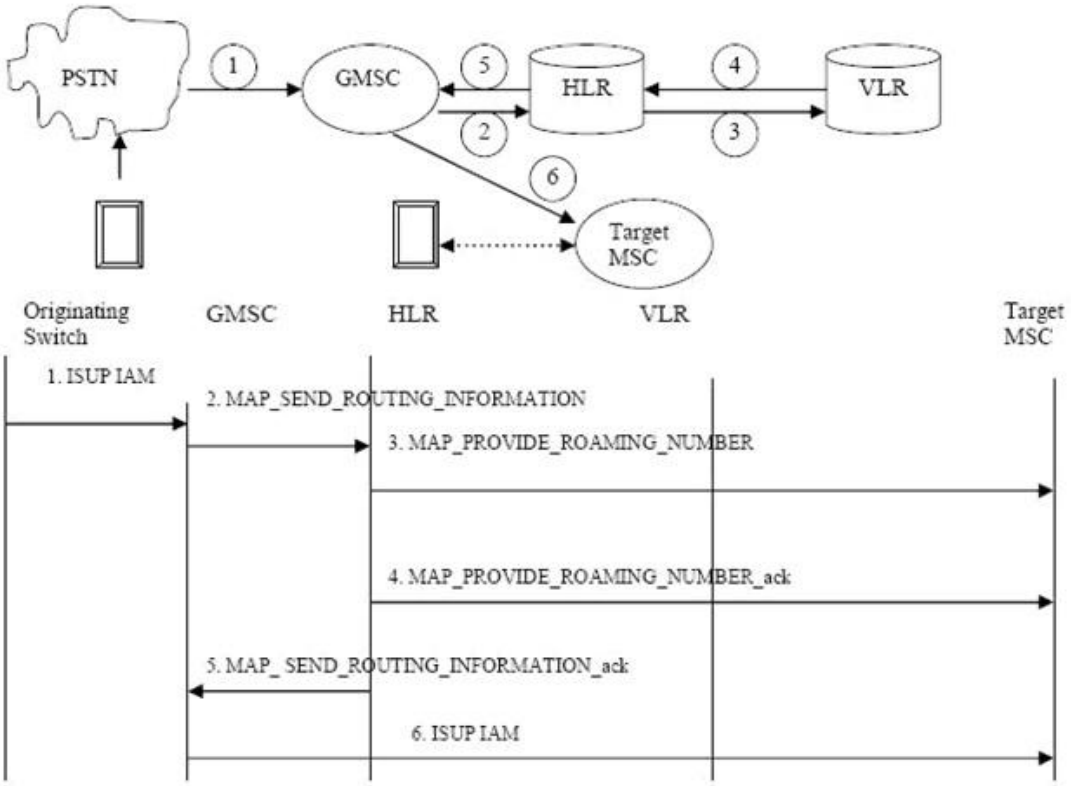


Figure 1.16: GSM Call Termination Operation

Above figure shows the GSM Call termination operation with various steps. As shown in figure the operation begins with obtaining routing information and end with establishing the communication link

Summary:

In this Chapter we studied and covered the following Topics.

- What is Mobile Computing?
- Mobile Computing Architecture
- Various Mobile Generations
- Devices 1G,2G,2.5G,3G
- How GSM Works and Its Architecture
- GSM Channels

Exercise:

1. Define Mobile Computing &Ubiquitous Computing?
2. What is Handoff? Explain with example?
3. Define Roaming? List out types of roaming?
4. What is Mobile Station?
5. What is Mobile Computing?
6. Discuss in Brief Mobile phones & Smart phones?
7. What is Tablets? Explain any one type of Tablet?
8. Draw and Explain Mobile Computing Architecture?
9. Explain Mobile Generation in Brief?
10. Compare 2nd Generation & 3 Generation?
11. What is Handoff? Explain Soft Handoff & Hard Handoff?
12. Draw GSM Architecture?
13. List out Popular Mobile Operating Systems
14. What are the main features of a mobile phone?

Unit 2

Introduction to Android

Learning Objectives:

After going through this unit , you will be able to learn:

- Android and its versions
- Android Architecture
- Simple Android App.

2.1 Android

No of mobile phones user are increased day by day and facilities are also increased. In Today they are not used just for making the calls but they have innumerable users and can be used as camera, Music Player, web browser and many more

2.1.1 What is Android

- **Its** an open source operating system based on Linux with a Java programming interface for mobile devices such as Smartphone. Smart phone is a Touch Screen Devices which supports Android OS as well for Tablets too.
- It was developed by the **Open Handset Alliance** (OHA), which is led by Google. The **Open Handset Alliance** (OHA) is consortium of multiple companies like Samsung, Sony, Intel and many more to provide services and deploy handsets using android platform.
- Unlike IOS Android is Open Source. It gives the Opportunity to the Developers to introduce and incorporate any technological advancement.

2.1.2 History and Version

- Android was first created in 2003 by Andy Rubin in Palo Alto, California, United States. He first started developing the OS for digital cameras. Soon, he realized that the marketplace for digital camera operating systems perhaps wasn't all that big, and Android, Inc. diverted its attention toward smartphones.
- On 17th August 2005, Google acquired android Incorporation. Since that time, it is in the subsidiary of Google Incorporation.
- In 2007 Google announces the development of android Operating Ssystem.
- In 2008, HTC launched the first Android Application.

Following are the versions of Android With their code Names and API Level;

Version	Code Name	API	Release Date
1.0	Apple Pie	1	September 23, 2008
1.1	Banana Bread (Petit Four)		February 9, 2009
1.5	Cupcake	3	April 27, 2009
1.6	Donut	4	September 15, 2009
2.1	Éclair	7	October 26, 2009
2.2	Froyo	8	May 20, 2010
2.3	Gingerbread	9 and 10	December 6, 2010
3.1 and 3.3	Honeycomb	12 and 13	May 10, 2011, July 15, 2011
4.0	Ice Cream Sandwich	15	October 18,2011
4.1,4.2 and 4.3	Jelly Bean	16 , 17 and 18	July 9, 2012, November 13, 2012, July 24, 2013
4.4	Kitkat	19	October 31, 2013
5.0	Lollipop	21	October 17, 2014
6.0	Marshmallow	23	October 5, 2015
7.0	Nougat	24-25	August 22, 2016
8.0	Oreo	26-27	August 21, 2017
9.0	Pie	28	August 6, 2018
10.0	Q	29	Yet to Come

2.1.3 Android Architecture

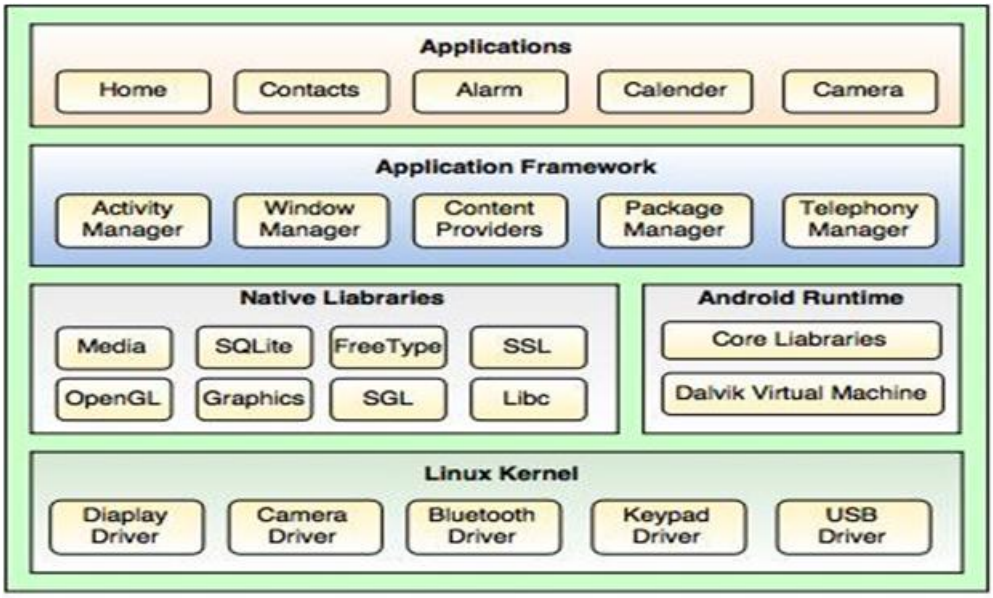


Figure: 2.1 Android Architecture

- Android Operating System is roughly divided into five sections and four main layers as shown in above architecture diagram.

Linux kernel: At the very bottom of the layers is Linux. This layer provides a level of abstraction between the device hardware and it contains all the essential hardware drivers like camera, keypad, display etc. The kernel takes care of all the things that Linux is really good at such as networking and a vast array of device drivers. It take the pain out of interfacing to peripheral hardware.

Libraries: Top of the linux kernel has set of libraries including open-source Web browser engine WebKit, well known library libc, SQLite is a database which is useful repository for storing and sharing of application data, libraries to play and record audio and video, SSL libraries responsible for Internet security etc.

Android Libraries: This category includes 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. A summary of some key core Android libraries available to the Android developer is as given below −

- **android.app** – this provides access to the application model and is the cornerstone of all Android applications.
- **android.content** –this Facilitates content access, publishing and messaging between applications and application components.
- **android.database** – this access data published by content providers and includes SQLite database management classes.
- **android.opengl** – Its Java interface to the OpenGL ES 3D graphics rendering API.
- **android.os** – this provides applications with access to standard operating system services including messages, system services and inter-process communication.
- **android.text** − Used to render and manipulate text on a device display.
- **android.view** − The basic building blocks of application user interfaces.

- **android.widget** − A rich collection of pre-built user interface components such as buttons, labels, list views, layout managers, radio buttons etc.
- **android.webkit** − A set of classes intended to allow web-browsing capabilities to be built into applications.

Following are the C/C++ based libraries contained in this layer of the Android software stack.

Android Runtime: This is the third section of the architecture and available on the second layer from the bottom. This section has a very important component called **Dalvik Virtual Machine** which is a kind of Java Virtual Machine specifically designed and optimized for Android.

The Dalvik VM makes use of Linux core features like memory management and multi-threading, which is fundamental 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 developers to write Android applications using standard Java programming language.

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

Applications

This is the top layer of application framework. The application which you create goes on this layer.

Examples. Facebook,google,internet browser, contacts etc.

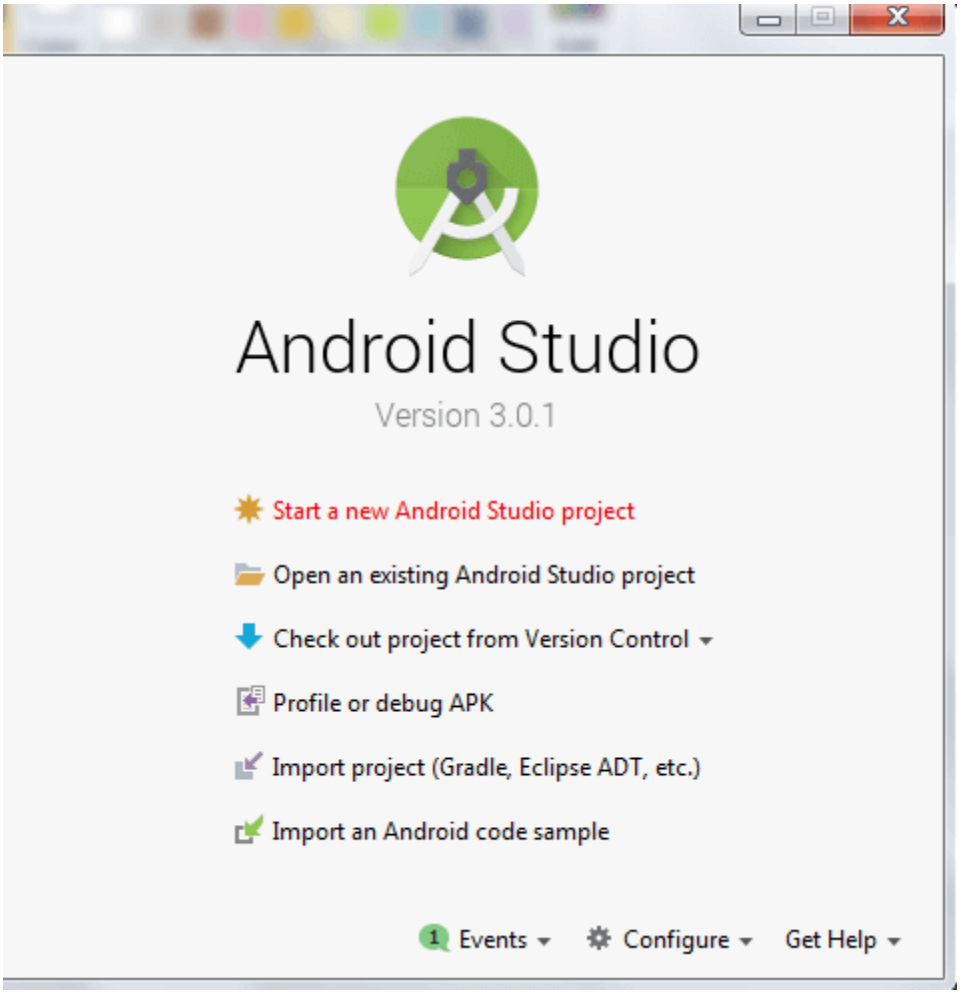
2.1.4 Hello Android example:

Here we write a Android application to print the Message Hello Android. You need to follow the 3 steps for creating the Hello android application.

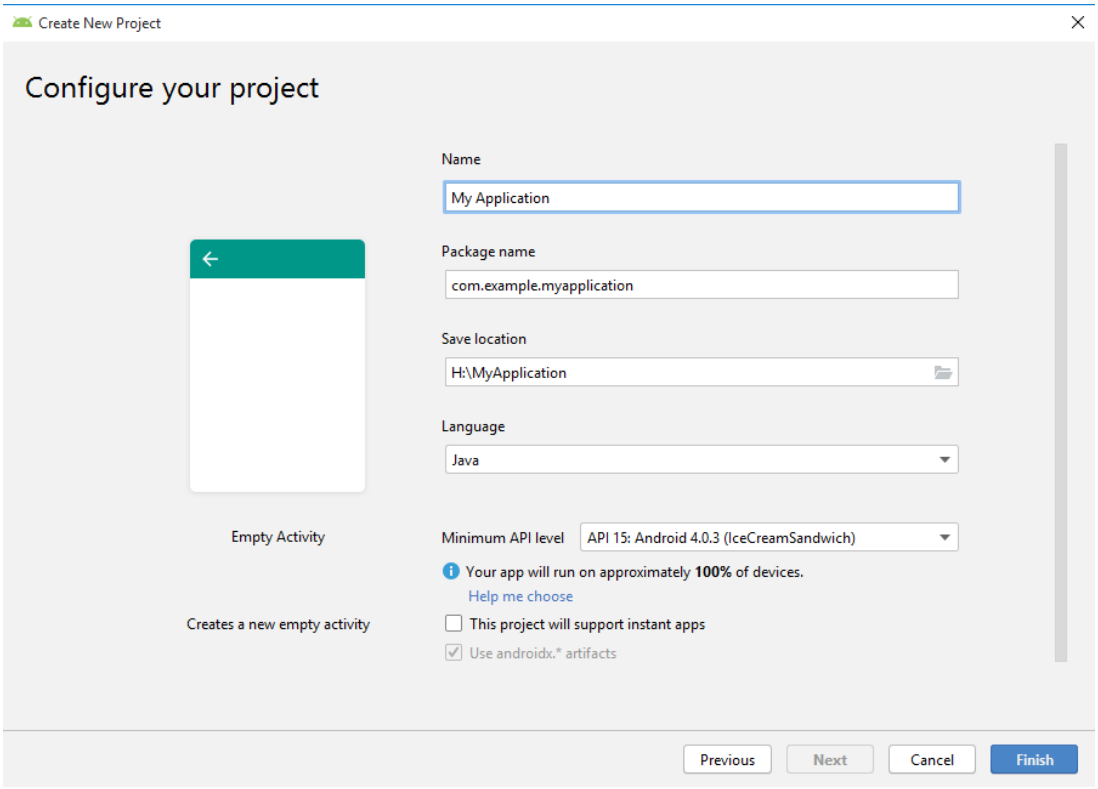
i) Create new Project in Android Studio
ii) Write the Message Hello Android
iii) Run the Android Application on Emulator / Mobile Device

1) Create the New Android project: For creating the new android studio project:

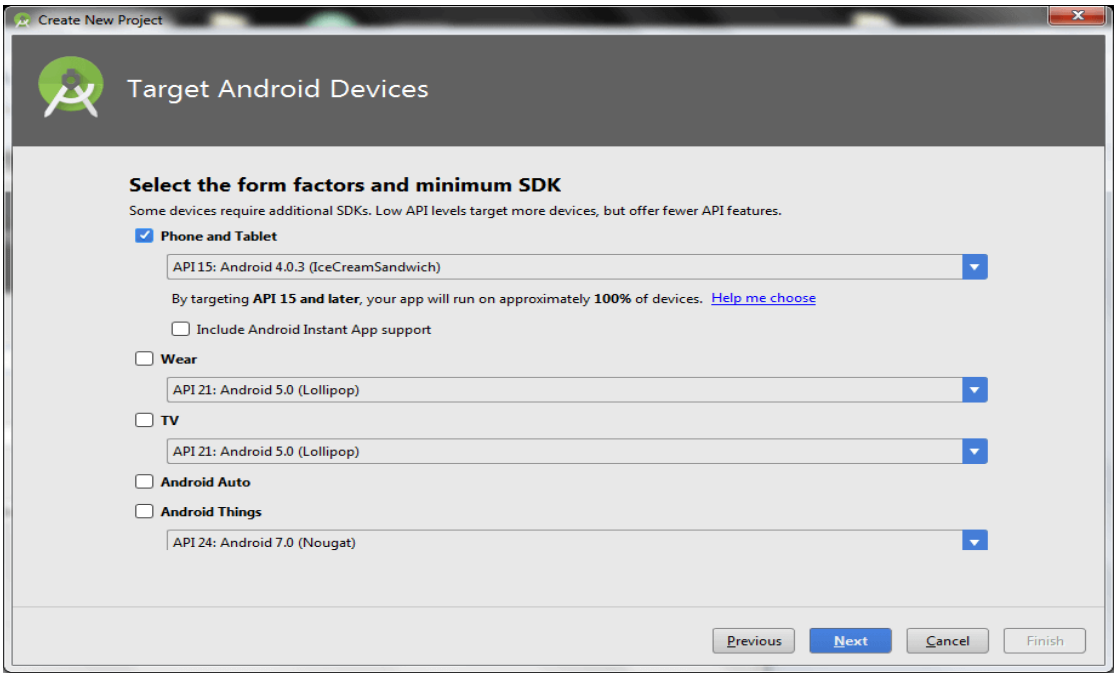
1. Select *Start a new Android Studio project*



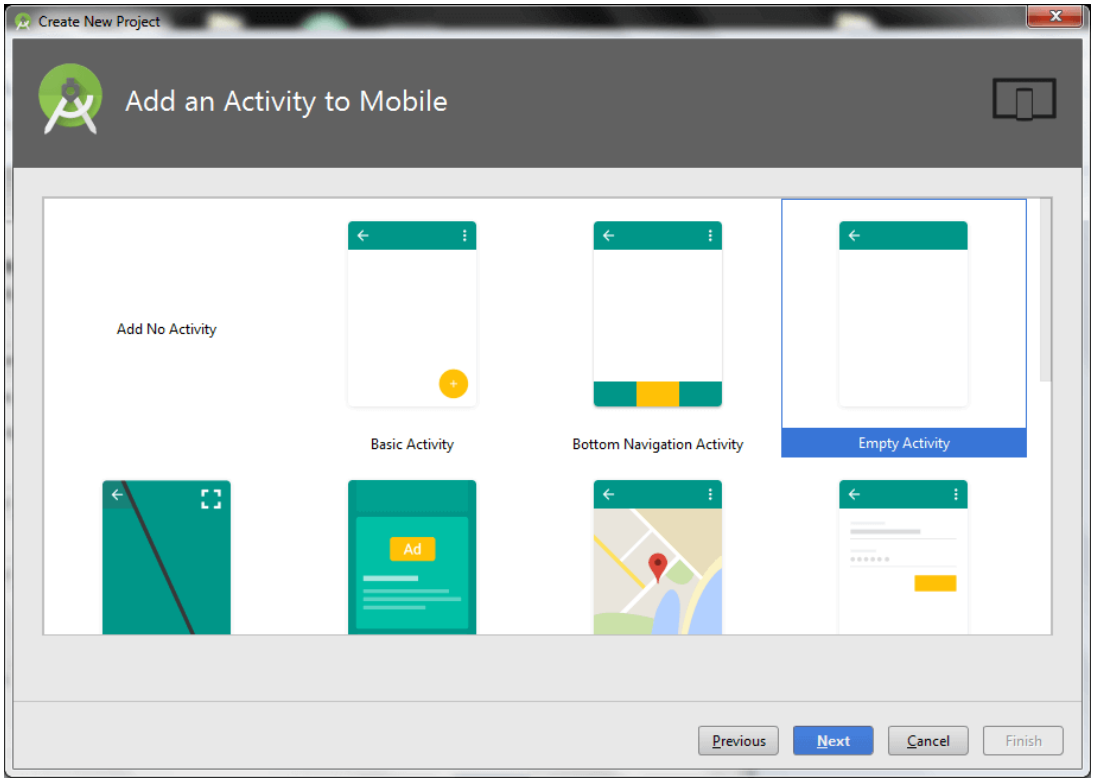
1) Provide the following information: Application name, Project location and Package name of application and click next.



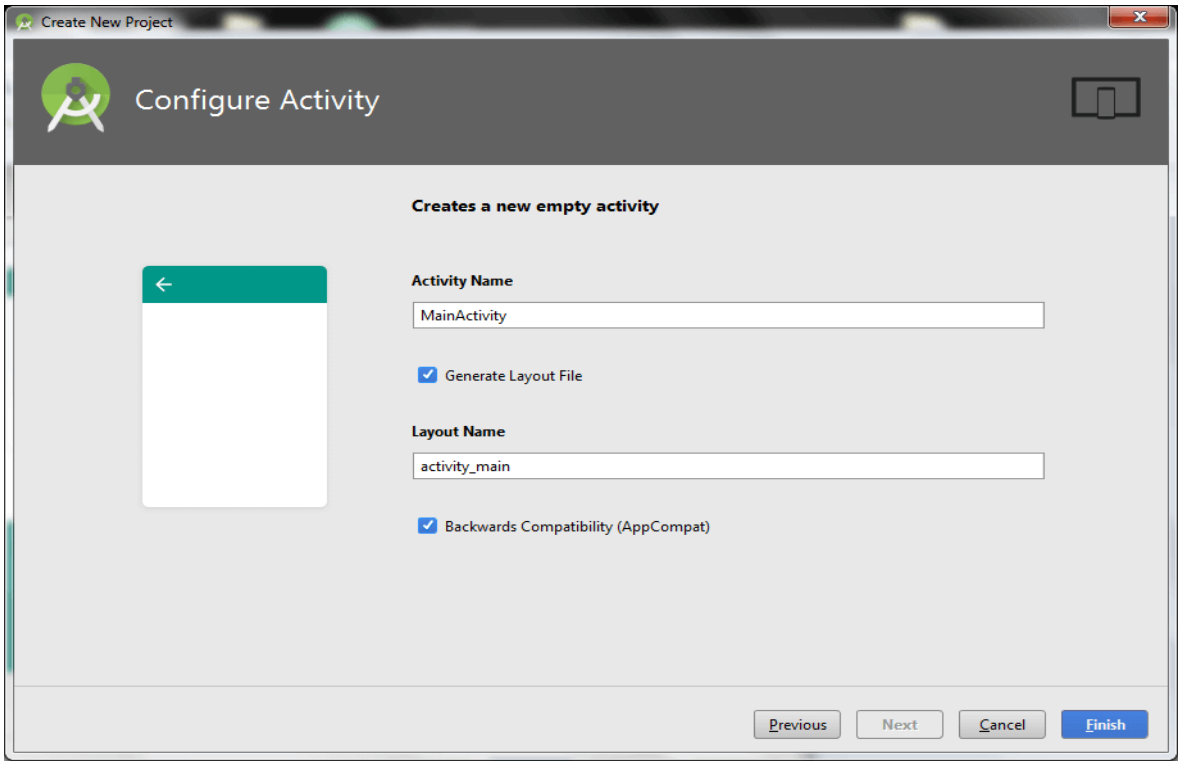
2) Select the API level of application and click next.



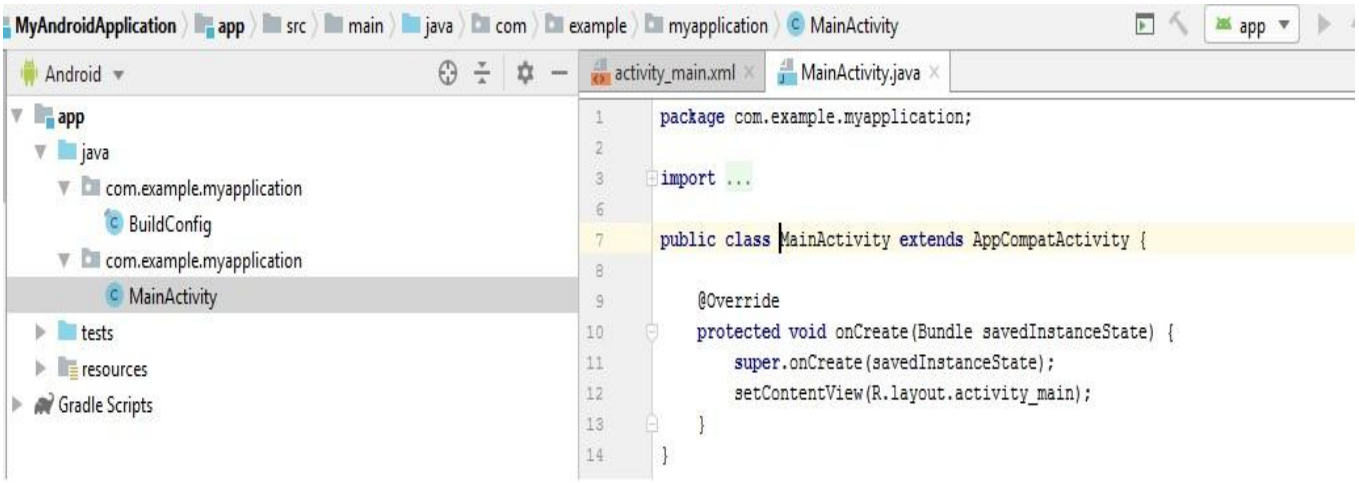
3) **Select the Activity type (Empty Activity).**



4) **Provide the Activity Name and click finish.**



After finishing the Activity configuration, Android Studio auto generates the activity class and other required configuration files. Now an android project has been created. You can explore the android project and see the simple program, it looks like this:



2) Writing the Message in activity_main.xml

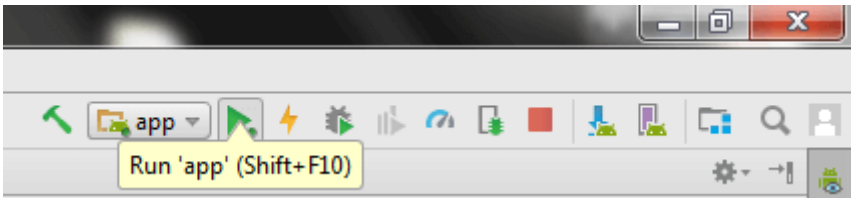
Android studio auto generates code for activity_main.xml file in res->Layout folder. You may edit this file according to your requirement.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/Text"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

<TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
android:text="Hello Android!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

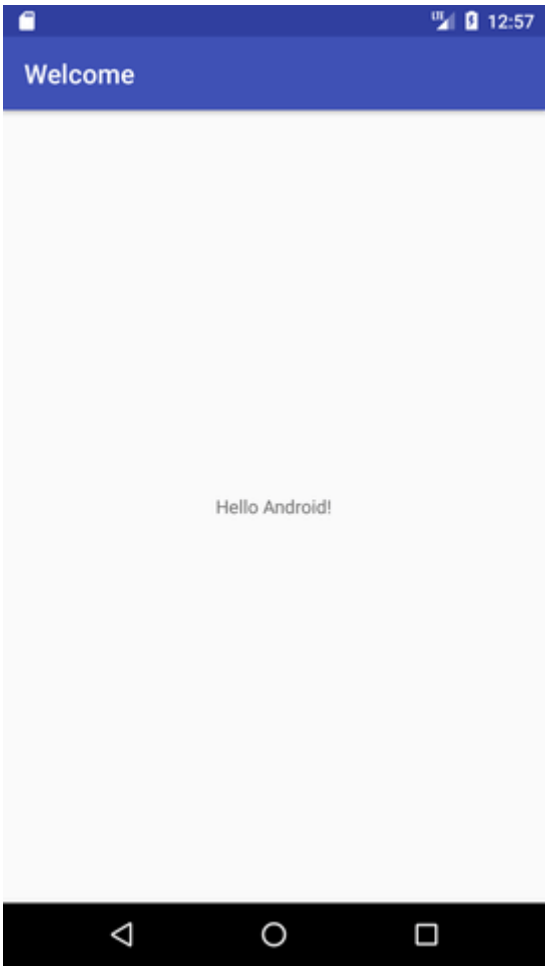
Printing Message
Hello Android

3) Run the android application

To run the android application, click the run icon on the toolbar or simply press Shift + F10.



The android emulator might take 2 or 3 minutes to boot. After booting the emulator, the android studio installs the application and launches the activity. You will see something like this:



2.2 Dalvik VM

A virtual machine is same like a software implementation of a physical computer which works like real physical computer. It means a virtual machine can compile and run any program just like a physical computer does for us. Its just like a emulator compared to the real physical machine. But there is a dark-side of virtual machine as it is less efficient than real physical computer and provides unstable performance when multiple virtual machines runs at the same time on same machine.

2.2.1 Dalvik Virtual Machine

Dalvik is a purposely built virtual machine designed specially for android which was developed by Dan Bornstein and his team. It was only developed for mobile devices. It uses register based architecture. Due to this dalvik virtual machine has few advantages over JAVA virtual machine such as:

1. Dalvik uses its own 16 bit instruction set while java uses 8 bit stack instructions, which reduce the dalvik instruction count and raised its interpreter speed.

2. Dalvik use less space, which means an uncompressed **.dex** file is smaller in size(few bytes) than compressed java archive file(.jar file).

2.2.2 Role of Dalvik Virtual Machine

In java programming we write and compile java program using java compiler and run that bytecode on the java virtual machine. In android we still write and compile java source file(bytecode) on java compiler, and it is once again recompiled using dalvik compiler to dalvik bytecode(*dx tool converts java .class file into .dex format*) and this dalvik bytecode is then executed on the dalvik virtual machine.

Note: Dalvik team have added Just In Time (JIT) compiler to the Dalvik Virtual Machine. The JIT is a software component which takes application code, analyzes it, and actively translates it into a form that runs faster, doing so while the application continues to run.

2.3 Software Stack

Android software stack is classified into five parts:

1. Linux based kernel
2. native libraries (middleware),
3. Android Runtime
4. Application Framework
5. Applications

Note:- The Description is already written section 2.1.3

Terminology

- **Android Software Development Kit (Android SDK)** contains the necessary tools to create, compile and package the Android applications
- **Android debug bridge (adb)**, is a tool that allows you to connect to a virtual or real Android device
- Google provides two integrated development environments (IDEs) to develop new applications.
    - **Android Developer Tools (ADT)** are based on the Eclipse IDE
    - **Android Studio** based on the IntelliJ IDE
- **Android RunTime (ART)** uses Ahead of Time compilation, and optional runtime for Android 4.4
- **Android Virtual Device (AVD)** - The Android SDK contains an Android device emulator. This emulator can be used to run an Android Virtual Device (AVD), which emulates a real Android phone where you can deploy and test your application.
- **Dalvik Virtual Machine** (Dalvik)-
    - The Android system uses a special virtual machine, Dalvik, to run Java-based applications. Dalvik uses a custom bytecode format which is different from Java bytecode.
    - Therefore you cannot run Java class files on Android directly; they need to be converted into the Dalvik bytecode format.

2.4 R.java file

Android R.java is *an auto-generated file by aapt* (Android Asset Packaging Tool) that contains resource IDs for all the resources of res/ directory.

If you create any component in the activity_main.xml file ex. If u drag a button widget on activity_main.xml file , id for the corresponding component is automatically created in this file. This id can be used in the activity source file (MainActivity.java) to perform any action on the component.

Note: If you delete R.jar file, android creates it automatically when it is rebuild.

Let's see the android R.java file. It includes a lot of static nested classes such as menu, id, layout, attr, drawable, string etc.

```
/* AUTO-GENERATED FILE.  DO NOT MODIFY.

 * This class was automatically generated by the

 * aapt tool from the resource data it found.  It

 * should not be modified by hand.

package com.example.helloandroid;


public final class R {

   public static final class attr {

   }

   public static final class drawable {

      public static final int ic_launcher=0x7f020000;

   }

   public static final class id {

      public static final int menu_settings=0x7f070000;

   }

   public static final class layout {

      public static final int activity_main=0x7f030000;
```

```
    }
    public static final class menu {
        public static final int activity_main=0x7f060000;
    }
    public static final class string {
        public static final int app_name=0x7f040000;
        public static final int hello_world=0x7f040001;
        public static final int menu_settings=0x7f040002;
    }
    public static final class style {
        /**
Base application theme, dependent on API level. This theme is replaced by AppBaseTheme from res/
values-vXX/styles.xml on newer devices.

 Theme customizations available in newer API levels can go in res/valuesvXX/styles.xml, while custo
mizations related to backward-compatibility can go here.

 Base application theme for API 11+. This theme completely replaces AppBaseTheme from res/values
/styles.xml on API 11+ devices.

 API 11 theme customizations can go here.

 Base application theme for API 14+. This theme completely replaces AppBaseTheme from BOTH
res/values/styles.xml and  res/values-v11/styles.xml on API 14+ devices.

 API 14 theme customizations can go here.          */
        public static final int AppBaseTheme=0x7f050000;
        /** Application theme.
```

```
All customizations that are NOT specific to a particular API-level can go here.        */

    public static final int AppTheme=0x7f050001;

  } }
```

2.5 Screen 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. Here we will see the xml code how the mobile screen is set to either landscape or portrait position.

Syntax:

```
<activity android:name="package_name.Your_ActivityName"

    android:screenOrientation="orirntation_type">

</activity>
```

Example:

```
<activity android:name=" com.example.myapplication"
```

```
    android:screenOrientation="portrait">
</activity>
activity android:name=".SecondActivity"
android:screenOrientation="landscape">
/activity>
```

The common values for screen Orientation attribute are as follows:

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.

Android Portrait and Landscape mode screen orientation example.

In this example, we will create two activities of different screen orientation. The first activity (MainActivity) will be as "portrait" orientation and second activity (SecondActivity) as "landscape" orientation type.

Activity File: File Name: activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.MainActivity">

     <Button
```

```
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginTop="112dp"
    android:onClick="onClick"
    android:text="Launch next activity"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.612"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/editText1"
    app:layout_constraintVertical_bias="0.613" />

<TextView
    android:id="@+id/editText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="124dp"
    android:ems="10"
    android:textSize="22dp"
    android:text="This activity is portrait orientation"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.502"
```

```
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>
```

Activity class: File Name: MainActivity.java

```
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;


public class MainActivity extends AppCompatActivity {
     Button button1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
          button1=(Button)findViewById(R.id.button1);
    }
    public void onClick(View v) {
        Intent intent = new Intent(MainActivity.this,SecondActivity.class);
        startActivity(intent);
    } }
```

Second Activity File Name: activity_second.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/re
```

```
s/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=" com.example.SecondActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="180dp"
        android:text="this is landscape orientation"
        android:textSize="22dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.502"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</android.support.constraint.ConstraintLayout>
```

Second Activity class File Name: SecondActivity.java

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;


public class SecondActivity extends AppCompatActivity {
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_second);
  } }
```

Android Manifest File Name: AndroidManifest.xml

In AndroidManifest.xml file add the screenOrientation attribute in activity and provides its orientation. In this example, we provide "portrait" orientation for MainActivity and "landscape" for SecondActivity.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="example.com.screenorientation">

   <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:roundIcon="@mipmap/ic_launcher_round"
      android:supportsRtl="true"
      android:theme="@style/AppTheme">
      <activity
         android:name="example.javatpoint.com.screenorientation.MainActivity"
         android:screenOrientation="portrait">
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
```

```
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity android:name=".SecondActivity"
        android:screenOrientation="landscape">
    </activity>
  </application>

</manifest>
```

Output:-

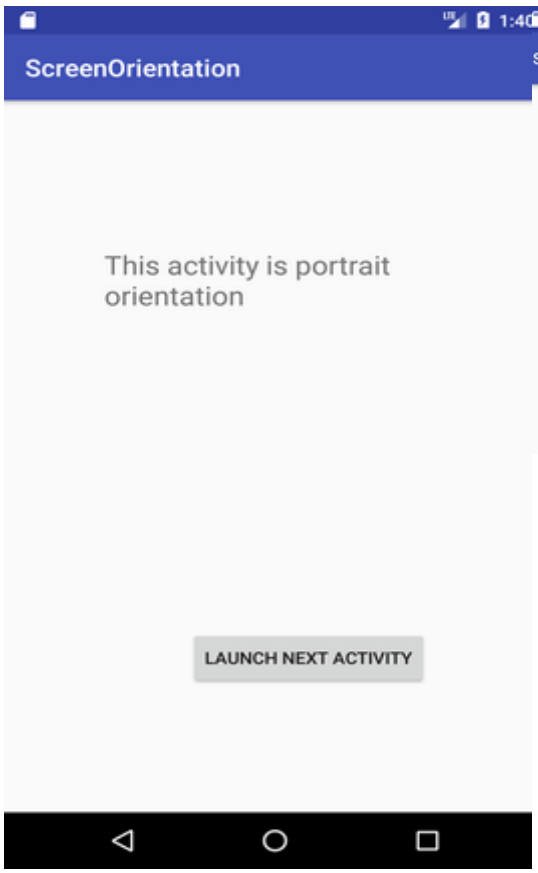


Fig. Portrait Orientation

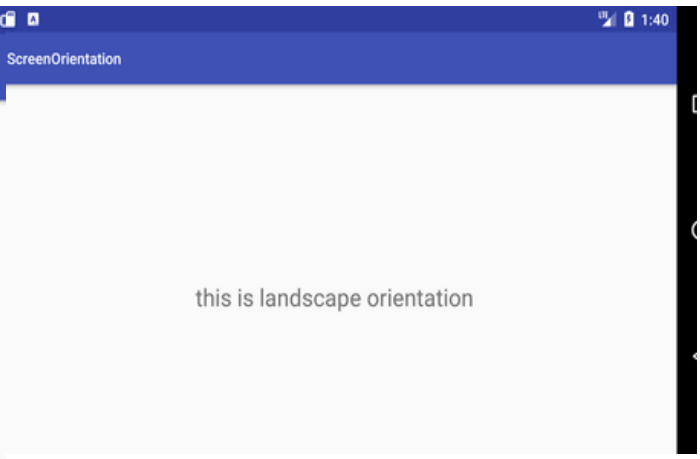


Fig. Landscape Orientation

2.6 Android Operating System

2.6.1 Introduction

What Is the Android Operating System?

Android operating system is a mobile operating system which was developed by Google to be primarily used for touchscreen devices, cell phones, and tablets. Its design lets users operate the mobile devices intuitively, with finger movements that emulate common motions, such as pinching, swiping, and tapping.

In addition to mobile devices, Google also employs Android software in televisions, cars, and wristwatches—each of which is fitted with a unique user interface.

Understanding the Android Operating System

The Android operating system was first developed by Android, Inc., a software company located in Silicon Valley before Google acquired it in 2005. Investors and electronics industry analysts have questioned Google's true intentions for entering the mobile market space, since that acquisition. But in any case, soon thereafter, Google announced the forthcoming rollout of its first commercially available Android-powered device in 2007, although that product actually hit the marketplace in 2008.

Since then, software and application developers have been able to use Android technology to develop mobile applications, which are sold on app stores, such as Google Play store. And because it is developed as a Google product, Android users are given the opportunity to link their mobile device to other Google products, such as cloud storage, email platforms, and video services.

Android OS Has the Following Features;

- Integrated browser, based on the open source WebKit engine
- Optimized 2D and 3D graphics, multimedia and GSM connectivity
- Bluetooth
- EDGE
- 3G
- WiFi
- SQLite
- Camera
- GPS
- Compass
- Accelerometer

Software developers who want to create applications for the Android OS can download the Android Software Development Kit (SDK) for a particular version. The SDK includes a debugger, libraries, an emulator, some documentation, sample code and tutorials. For faster development, interested parties can use graphical integrated development environments (IDEs) such as Eclipse to write applications in Java.

2.6.2 Android Versions with Features

Version	Features
1.0	✓ Download and updates via Android Market ✓ Web Browser ✓ Camera support ✓ Gmail, Contacts and Google Agenda synchronization ✓ Google Maps ✓ YouTube application
1.1	✓ "Show" & "Hide" numeric keyboard, in caller application ✓ Ability to save MMS attachments
1.5	✓ Bluetooth A2DP, AVRCP support ✓ Soft-keyboard with text-prediction ✓ Record/watch videos
1.6	✓ Gesture framework ✓ Turn-by-turn navigation
2.1	✓ HTML ✓ Digital zoom ✓ Microsoft Exchange support ✓ Bluetooth 2.1 ✓ Live Wallpapers ✓ Updated UI
2.2	✓ Speed improvements ✓ JIT implementation ✓ USB Tethering ✓ Applications installation to the expandable memory ✓ Upload file support in the browser ✓ Animated GIFs
2.3	✓ Updated UI ✓ Improved keyboard ease of use ✓ Improved copy/paste ✓ Improved power management ✓ Social networking features ✓ Near Field Communication support ✓ Native VoIP/SIP support ✓ Video call support
3.1 and 3.3	✓ UI improvements

	✓ Open Accessory API ✓ USB host API ✓ Mice, joysticks, gamepads... support ✓ Resizable Home screen widgets ✓ MTP notifications ✓ RTP API for audio
4.0	✓ New lock screen actions ✓ Improved text input and spell-checking ✓ Control over network data ✓ Email app supports EAS v14 ✓ WI-FI direct ✓ Bluetooth Health Device Profile
4.1,4.2 and 4.3	✓ Dial pad auto-complete ✓ Photo Sphere enhancements ✓ Camera app UI updated ✓ 4K resolution support ✓ Ability to create restricted profiles for tablets ✓ Hebrew and Arabic right-to-left (RTL) support ✓ Bluetooth Low Energy (BLE) support ✓ Bluetooth Audio/Video Remote Control Profile (AVRCP) 1.3 support ✓ Security and performance enhancements
4.4	✓ Screen recording ✓ New Translucent system UI ✓ Enhanced notification access ✓ System-wide settings for closed captioning ✓ Performance improvements
5.0	✓ Multiple SIM cards support ✓ Quick settings shortcuts to join Wi-Fi networks or control Bluetooth devices ✓ Lock protection if lost or stolen ✓ High Definition voice call ✓ Stability and performance enhancements
6.0	✓ USB Type-C support ✓ Fingerprint Authentication support ✓ Better battery life with "deep sleep" ✓ Permissions dashboard ✓ Android Pay ✓ MIDI support ✓ Google Now improvements
7.0	✓ Unicode 9.0 emoji ✓ Better multitasking ✓ Multi-window mode (PIP, Freeform window) ✓ Seamless system updates (with dual system partition)

	✓ Better performance and code size thanks to new JIT Compiler
8.0	✓ PIP: Picture-in-Picture with resizable windows ✓ Android Instant apps ✓ Improved notifications system ✓ Improved system settings ✓ Lock screen redesign
9.0	✓ User interface updates: ✓ Rounded corners across the UI ✓ Quick settings menu change. ✓ Notification bar, the clock has moved to the left. ✓ The "dock" now has a semi-transparent background. ✓ New transitions when switching between apps, or within apps ✓ Volume slider updated ✓ Richer messaging notifications: with full conversation, large images, smart replies ✓ The power options now has a "screenshot" button ✓ Biometric authentication can now be disabled only once
10.0	✓ Smart Reply in all messaging apps ✓ Enhanced location and privacy tools ✓ Edge-to-edge gestures ✓ compatible with 5G
11	✓ Built-in screen recording ✓ Smart home and media controls ✓ Improved accessibility

The above table shows the different version of android available so far and their respective features. The last version launched by android was 11.0 Android 11.0 has introduced with so many exciting features like Built-in screen recording and Smart home and media controls.

2.7 Android Development Elements

4.1 Installing the Java Development Kit

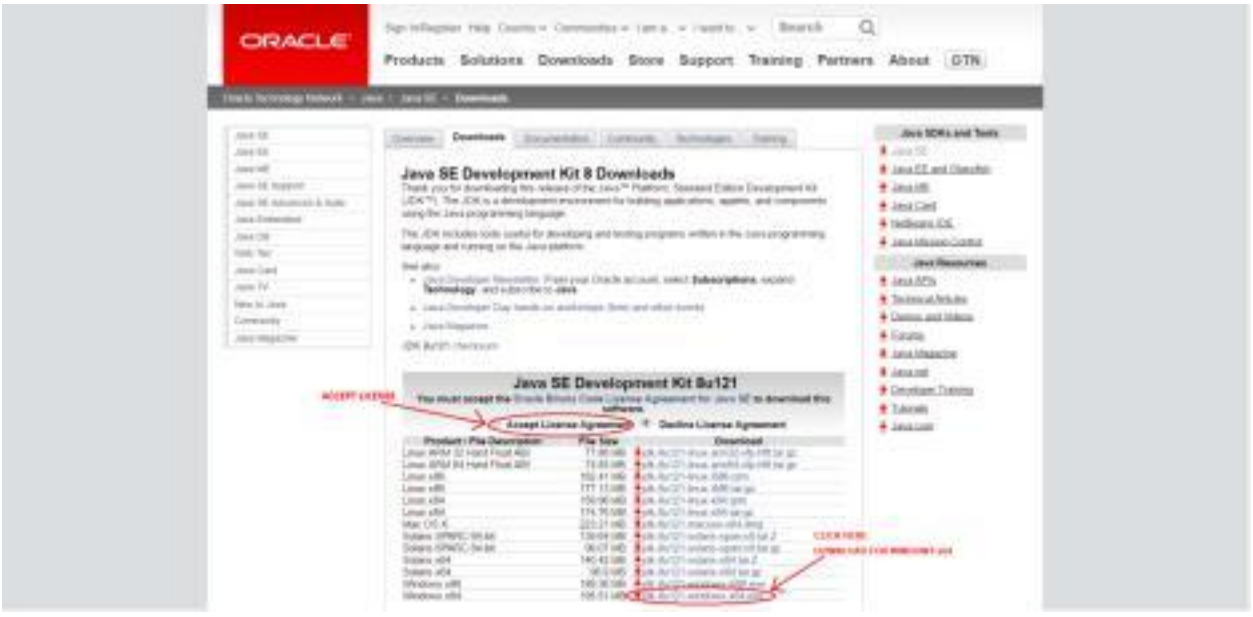
• Follow the Below Given Steps :-

1. **Navigate - http://www.oracle.com/technetwork/java/javase/downloads/index.html.** Youcan download a simple Java Software Development Kit (JDK) installer for Windows, macOS, or Linux directly from Oracle.

2. **Click the "Download" button beneath "JDK."** This will open a new page containingseveral download options.

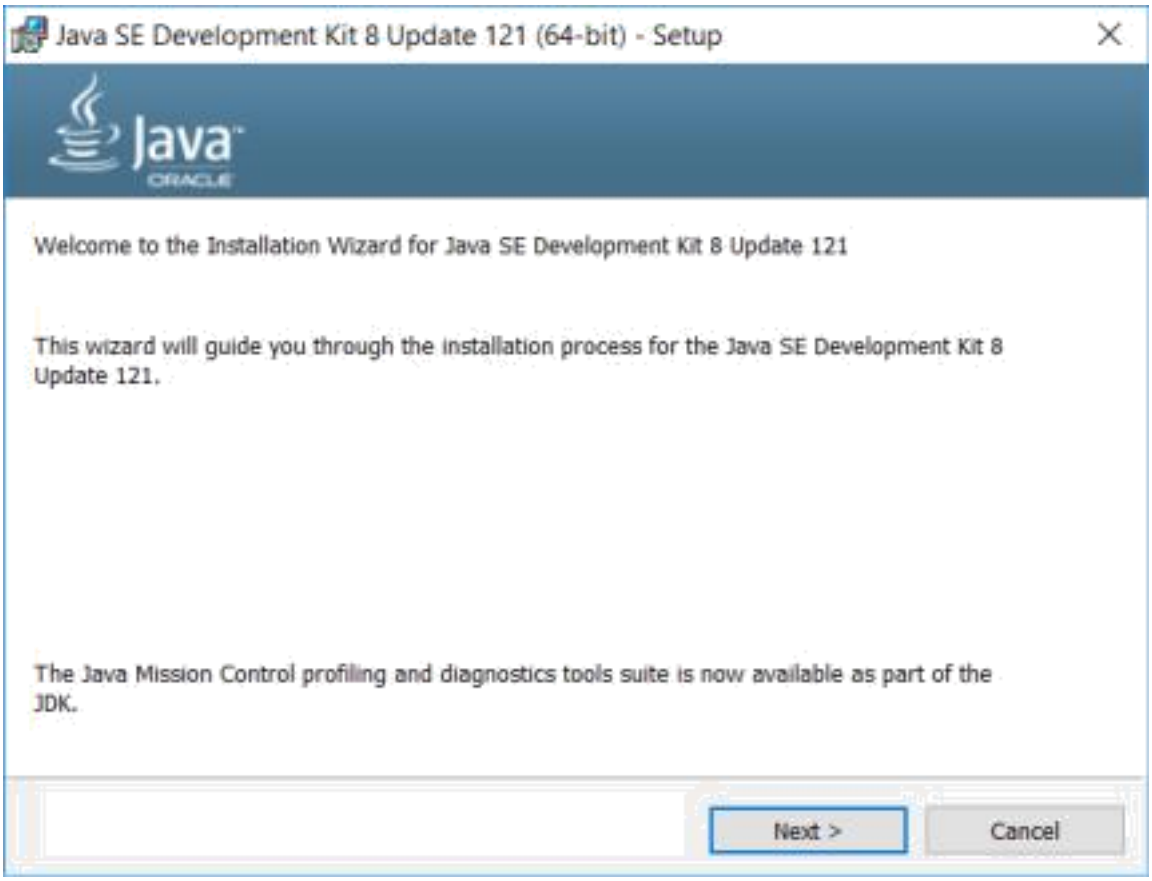
3. **Scroll to the latest version of Java SE Development Kit.** You should always use the lateststable version of the toolkit.

4. **Click "Accept License Agreement."** Before you can click the download link, you mustaccept the license agreement. The option is just beneath the JDK version number.

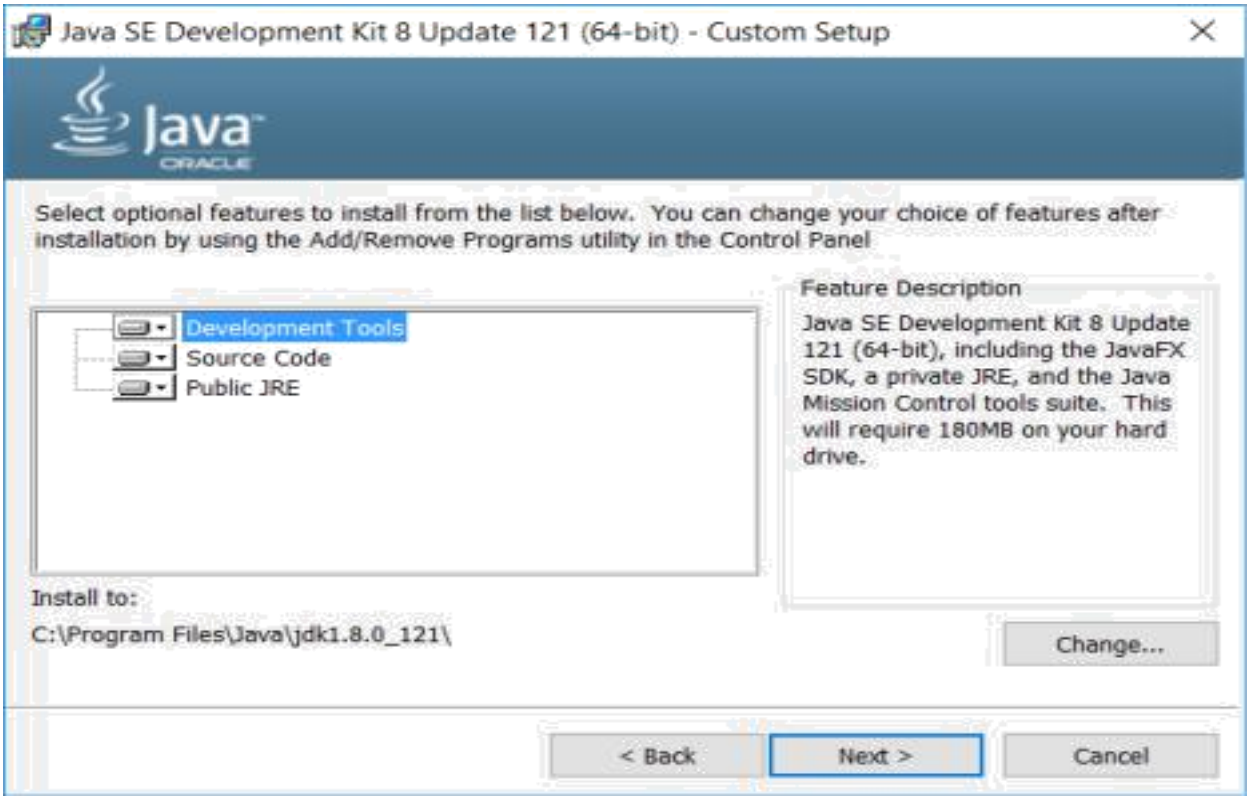


5. Click the download link next to your operating system. Once you click the link, follow the prompts to select a save location on your computer and start the download

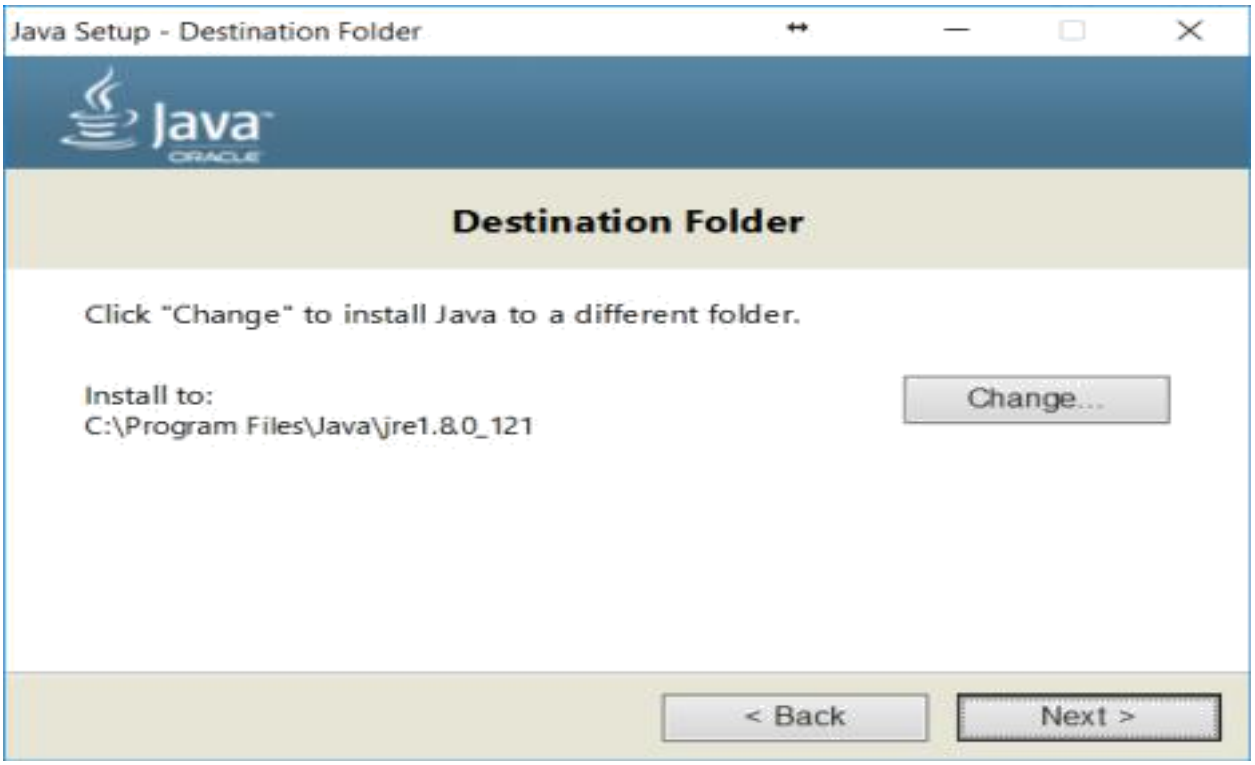
6. Once your file is downloaded, navigate to the directory where you saved the file and double click the file to begin the installation. After installation window appears, click next:



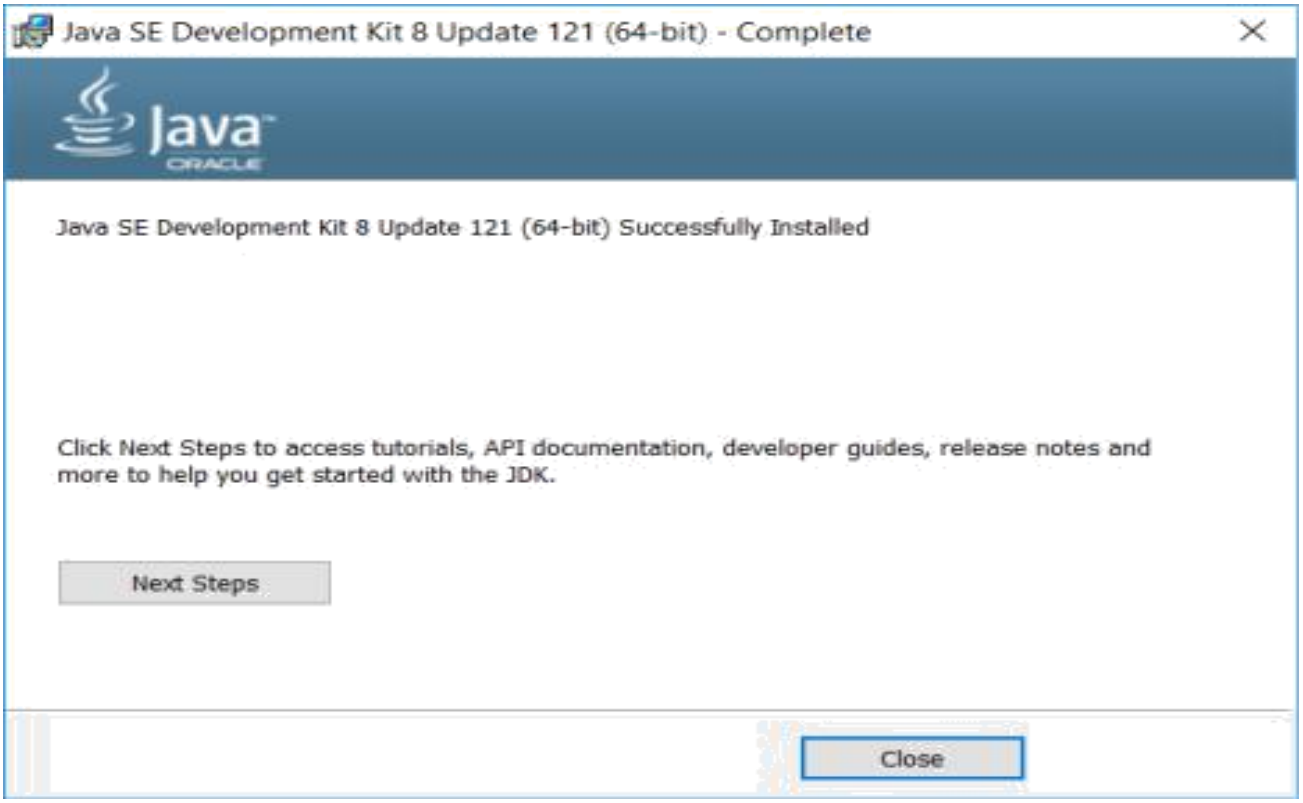
7. Select installation directory. It is recommended to proceed with the default installation directory. Click next.



8. Click Next to install JRE:



9. Click Close to complete setup.



4.2 Installing Android Studio

Follow the Below Given Steps :-

1. Download the Android Studio Bundle from Below given Url
Note:- Always use Latest Stable Version https://developer.android.com/sdk/index.html

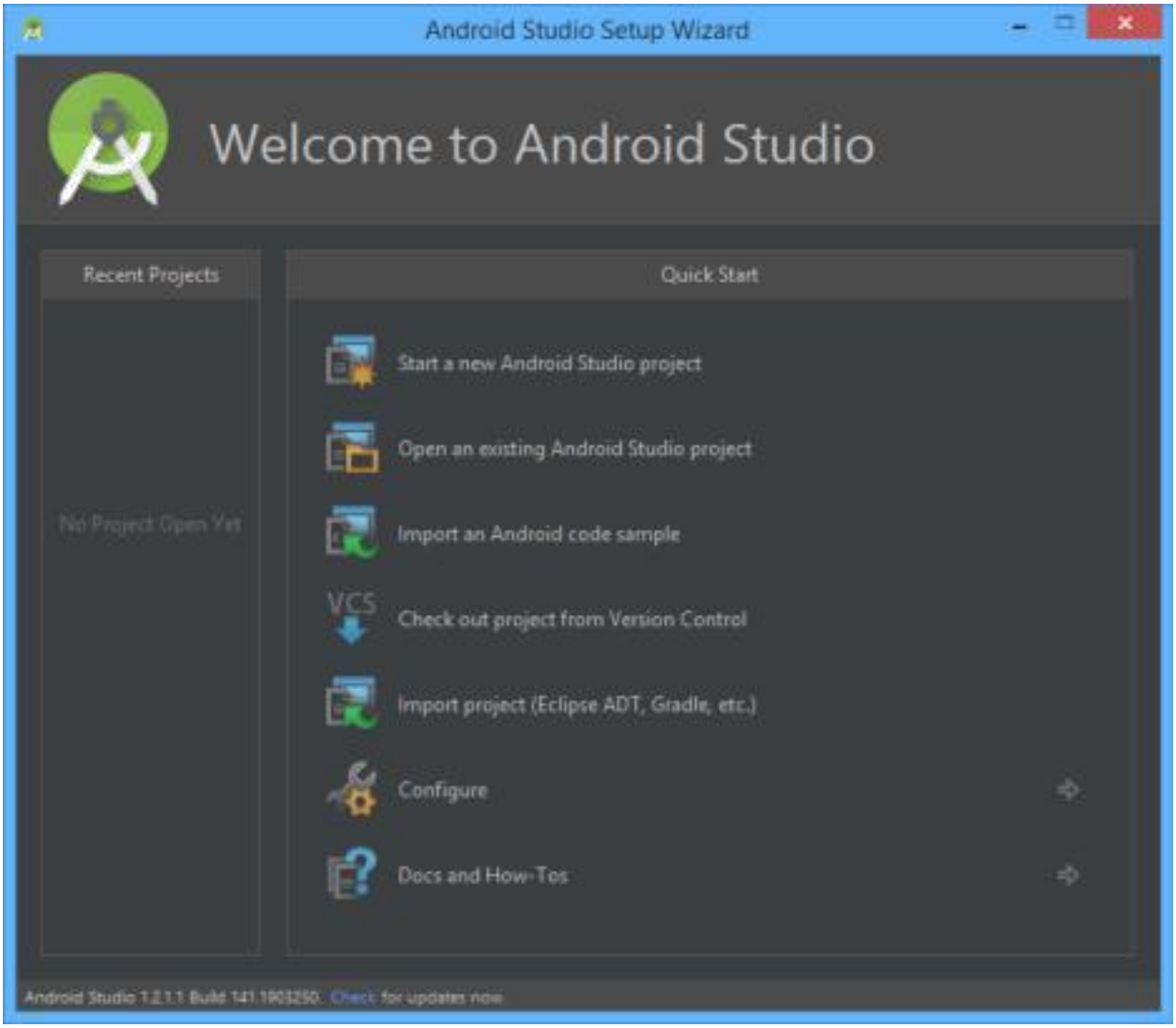
2. Once Download is Finished Start Installing Android Studio

Follow the prompts to complete the installation. I used the default settings.

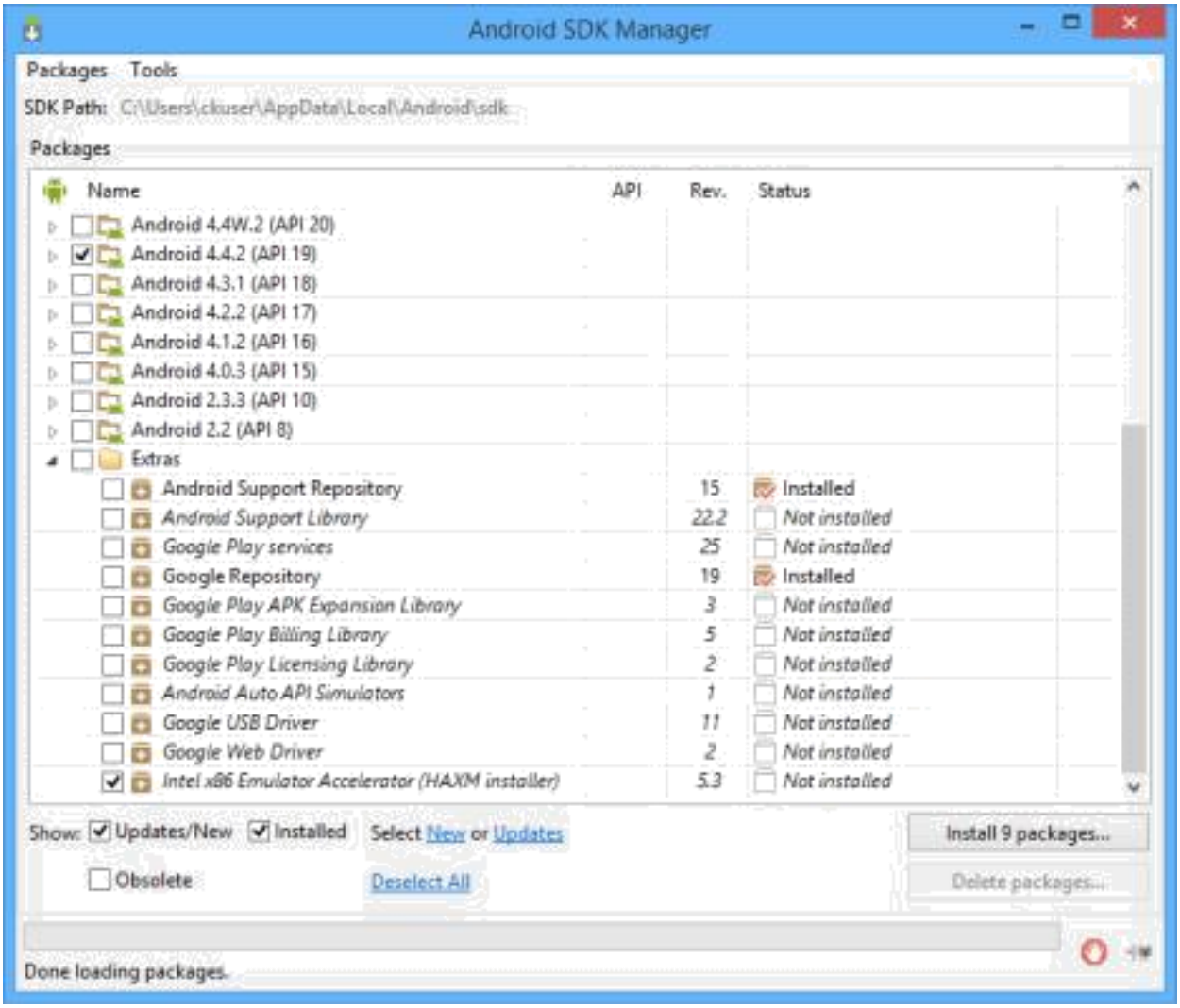
4. Allow Android Studio access to the network.

5. Select your desired UI theme.

6. Android Studio will download additional components. This will take several minutes.


Android Studio Setup Wizard
Welcome to Android Studio
Recent Projects
No Project Open Yet
Quick Start
Start a new Android Studio project
Open an existing Android Studio project
Import an Android code sample
VCS
Check out project from Version Control
Import project (Eclipse ADT, Gradle, etc.)
Configure
Docs and How-Tos
Android Studio 1.2.1.1 Build 141.1903250. Check for updates now

6. Select “Configure/SDK Manager”

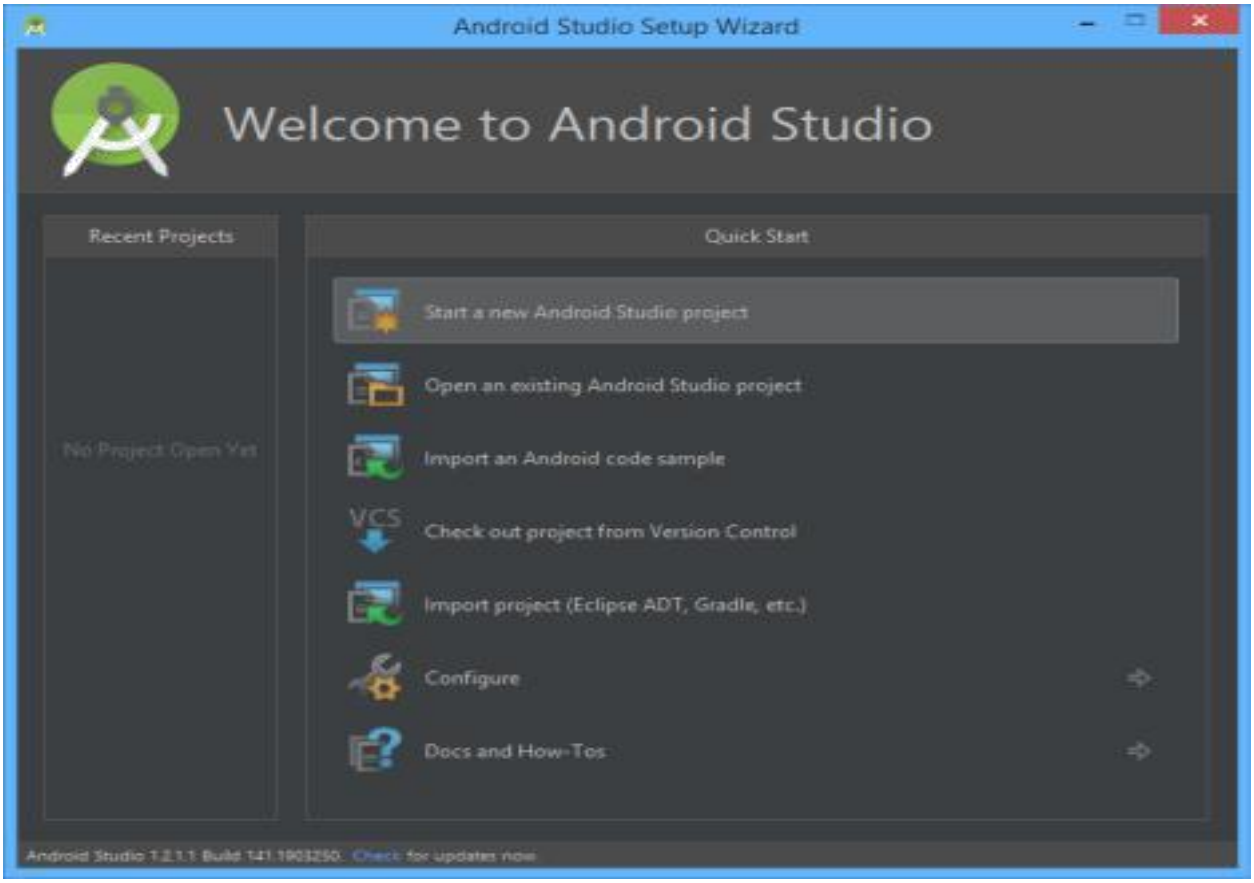


7. Deselect All

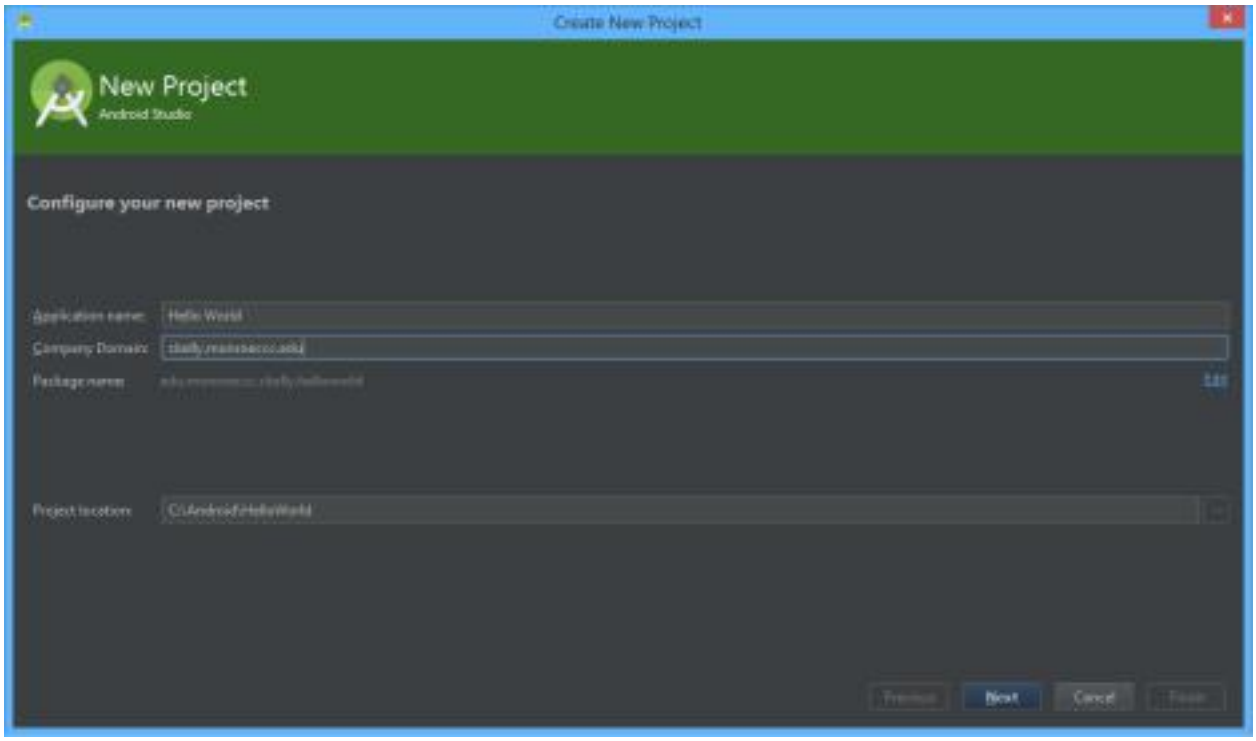
8. Scroll down and select “Android 4.4.2 (API 19)” and “Intel x86 Emulator Accelerator (HAXM installer)”

9. Install the packages. You may need to repeat the process of installing packages until all of them are installed. When the “Install packages…” button is no longer active you may close the

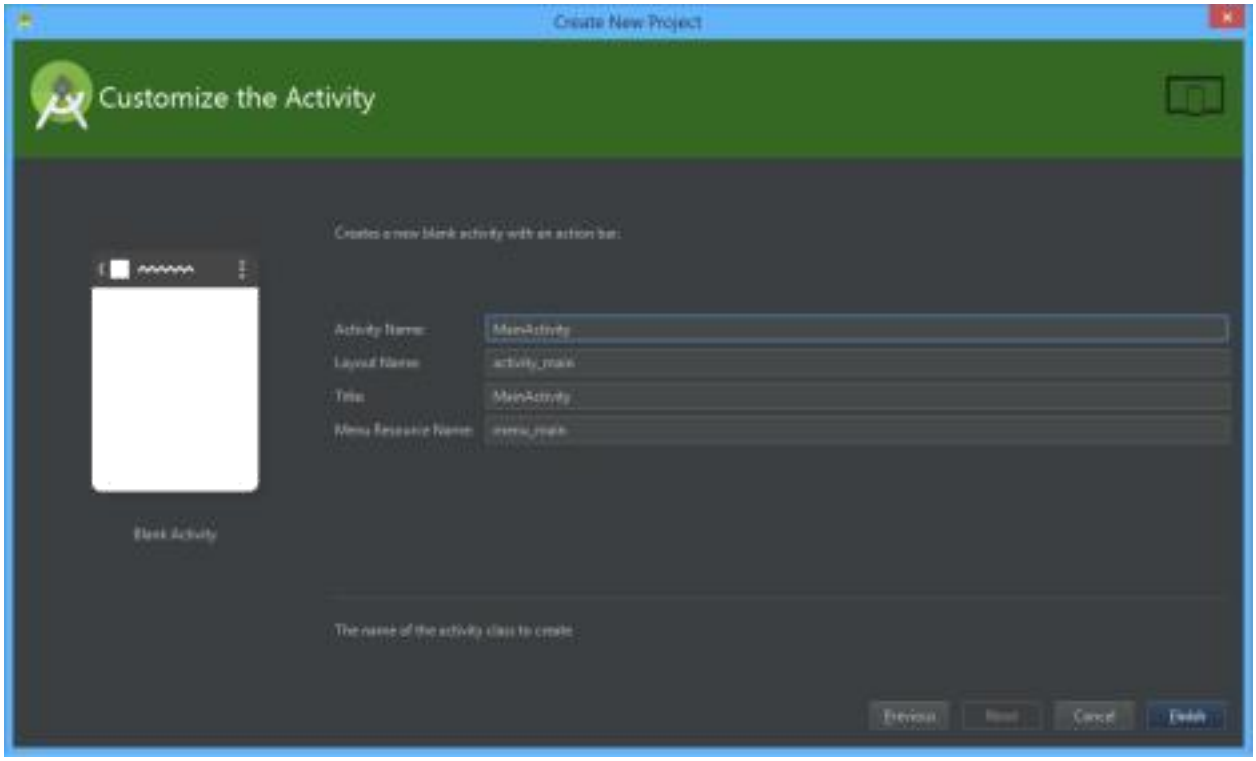
Android SDK Manager.



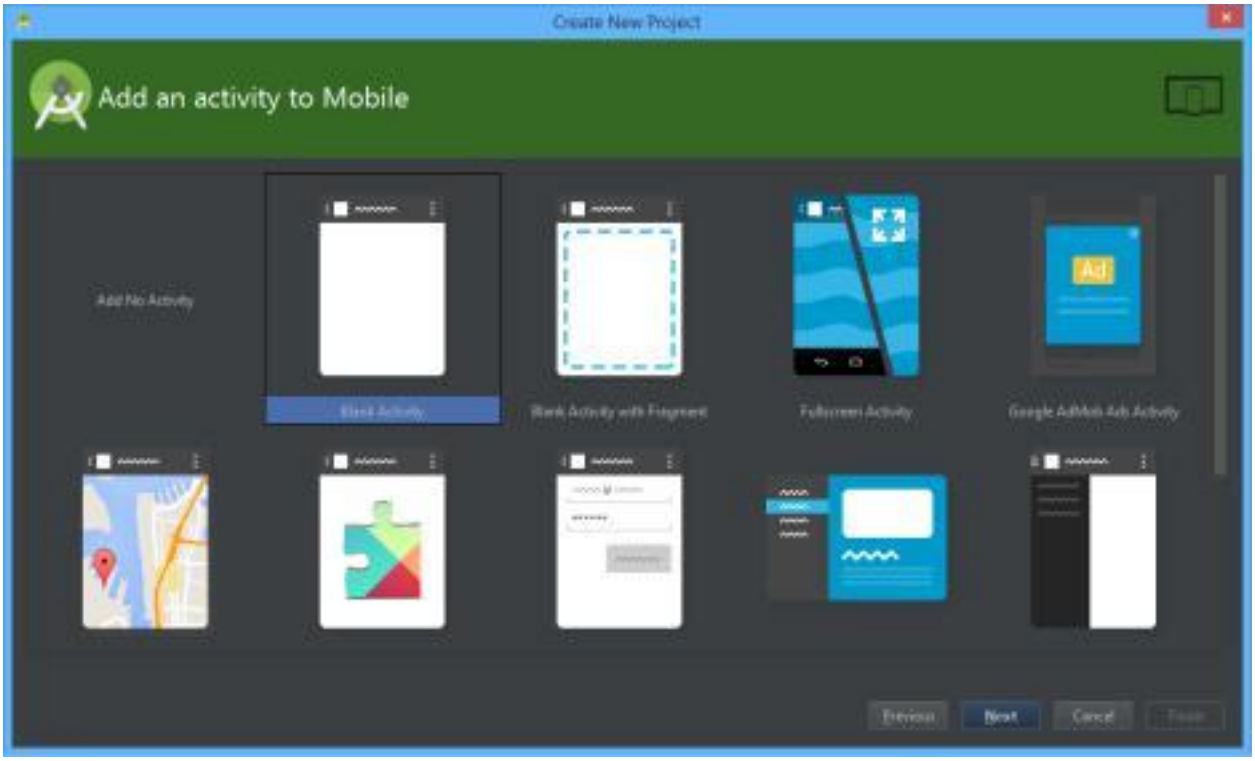
10. Start a new Android Studio project



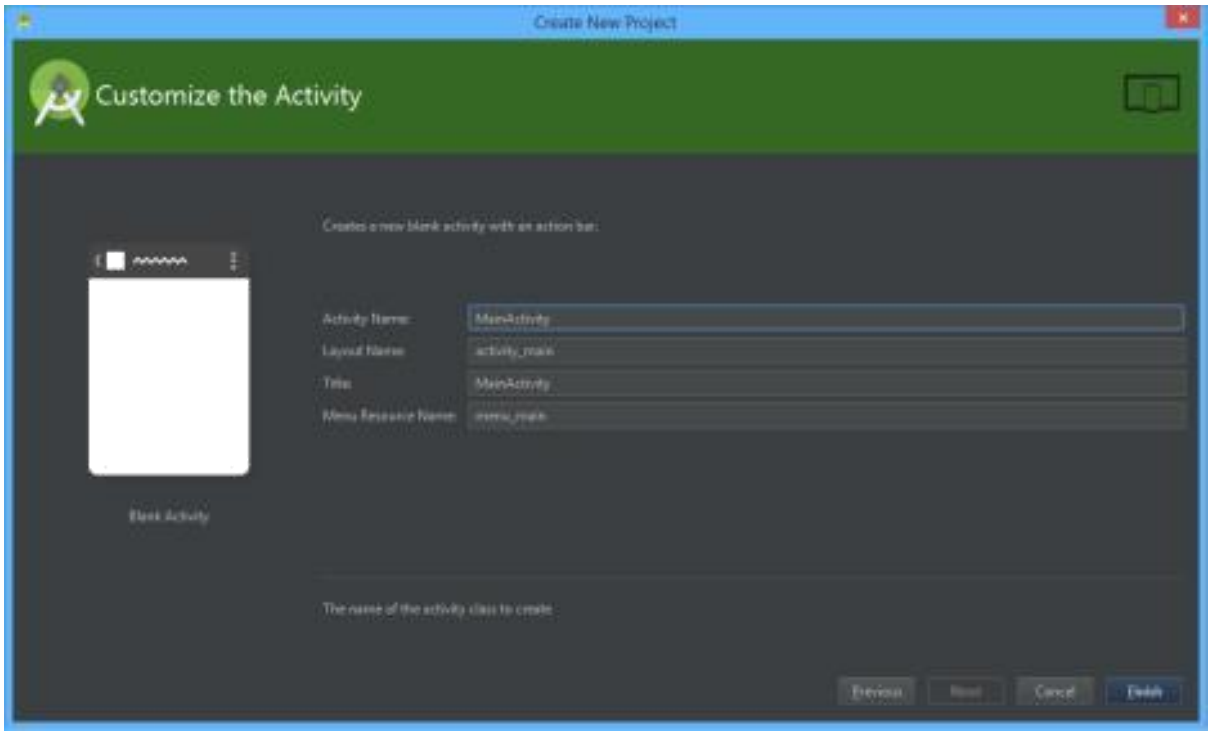
a. You may change the Application name and Company Domain according to your application



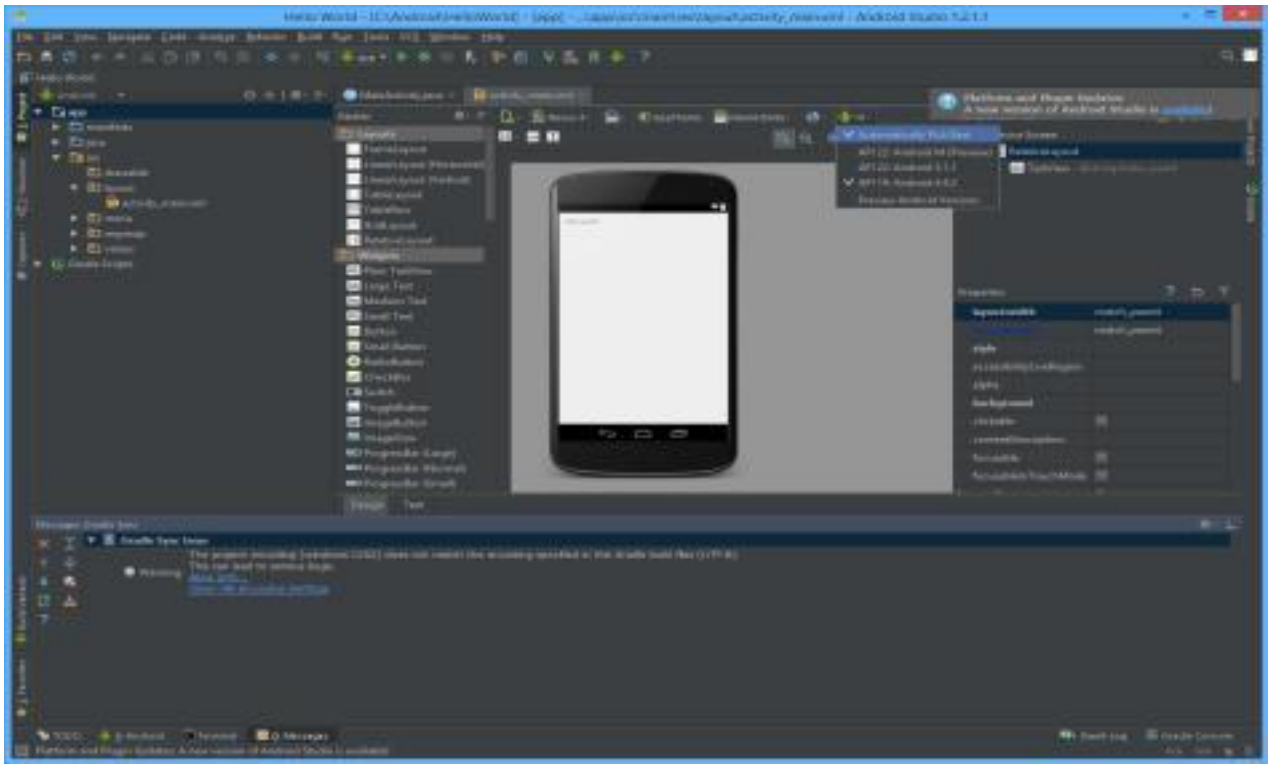
11. Select API 19 as the Minimum SDK.



12. Select Blank Activity



13. Click on Finish



13. Start Working on GUI

Android Studio is now installed. It automatically checks for updates. If you update it you may need to repeat some of the steps above to get it working again.

4.3 Update your tools with the SDK Manager

The Android SDK Manager helps you download the SDK tools, platforms, and other components which you will need to develop the application. Once downloaded, you can find each package in the directory indicated as the **Android SDK Location**, shown in figure 2.

To open the SDK Manager from Android Studio, click **Tools > SDK Manager** or click **SDKManager** in the toolbar. If you're not using Android Studio, you can download tools using thesdkmanager command-line tool.

When an update is available for a package you already have, a dash appears in the check box next to the package.

- To update an item or install a new one, click the check box so it shows a checkmark.
- To uninstall a package, click to clear the check box.

Pending updates are indicated in the left column with a download icon . Pending removals are indicated with a red cross .

To update the selected packages, click **Apply** or **OK**, then agree to any license agreements.

Summary:

In this chapter we have seen and discuss the following topics:

- Introduction to Android
- Various Android versions and Features
- Android Architecture
- We have written a Simple Android App to print the message Hello Android.
- We have written a Simple Android App to Adjust Screen orientation to Landscape or portrait.
- Installation steps of Android Studio.

Exercise:

Q. 1 What Is the Android Operating System? List out the name of Android operating system?

Q.2 List out any five Android versions and their names?

Q.3 Draw Android Architecture?

Q.4 Write step to create a simple Android Application using Android studio?

Q.5 what is Dalvik Virtual Machine in Android?

Q.6 What is the use of R Java file in Android Studio?

Q.7 what is screen orientation? How many types of orientation are there?

Q.8 what is the latest version of Android? Which operating system is used in Android?

Q.9.What is Android? Introduction of Android OS &it's Applications

Unit 3
User Interface Screen Elements

Learning Objectives:

After going through this unit , you will be able to:

- Fundamentals of User Interface
- Android UI Components
- Working of Toast
- Working of Buttons
- Working of TextView, EditText and Checkboxes
- Working of Spinners & use of Adapters
- Working of Rating bar and Progress bar and Its implementations

3.1 Toast & Snack Bar

Toast:

- Android Toast is used to display information for the short period of time.
- A Toast contains message to be displayed quickly and disappears after sometime.
- android.widget.Toast class is the subclass of java.lang.Object class
- User can create custom toast to display the images with the text.

How to Create the Toast?

```
Toast toast=Toast.makeText(getApplicationContext(),"Hello  Javatpoint ",Toast.LENGTH_SHORT);

                    //Displaying Toast with Hello  Javatpointmessage
```

Constants of Toast class

There are only 2 constants of Toast class which are given below.

Table 3.1 Constant Description

Constant	Description
public static final int LENGTH_LONG	Displays view for the long duration of time.
public static final int LENGTH_SHORT	Displays view for the short duration of time.

Methods of Toast class: The widely used methods of Toast class are given below.

Table 3.2 Method Description

Method	Description
public static Toast makeText(Context context, CharSequence text, int duration)	makes the toast containing text and duration.
public void show()	displays toast.
public void setMargin (float horizontalMargin, float verticalMargin)	changes the horizontal and vertical margin difference.

Output:-

Program:-

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.Toast;

public class MainActivity extends AppCompatActivit
y {

    @Override
    protected void onCreate(Bundle savedInstanceStat
e) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //Displaying Toast with Hello Android message
        Toast.makeText(getApplicationContext(),"Hello
Android",Toast.LENGTH_SHORT).show();
    }
}
```

Snackbar:-

- Snack bar in android is a new widget introduced with the Material Design library as a replacement of a Toast.
- It is used to show the messages in the bottom of the application with swiping enabled.
- It contain option action button.

Table 3.3. Comparison of Snakbar and Toast

Snakbar	Toast
Snackbar can be only showed in the bottom of the screen	A Toast messages can be customized and printed anywhere on the screen,
Snackbar may have action button optionally	A Toast message don't have action button,
Snackbar can be swiped off before the time limit	Toast message cannot be off until the time limit finish.

Note: Toast message and Snackbar have display length property in common.

How to Create the Snakbar?

```
Snackbar snackbar = Snackbar.make(coordinatorLayout, "www.example.com", Snackbar.LENGTH_LONG);
snackbar.show();
```

In the above snippet make() method accepts three parameters:

1. **coordinatorLayout** : It is the root layout of the activity
2. **www.example.com**: This is the message to be appear on snackbar, and we can customise it with our own message
3. **Snackbar.LENGH_LONG**: This is last parameter which is the time limit how long snackbar to be displayed

show() method is used to display the Snackbar on the screen.

1. Simple Snakbar

MainActivity.java

```
final CoordinatorLayout coordinatorLayout = (CoordinatorLayout) findViewById(R.id.cordinatorLayout);

findViewById(R.id.btnSimple).setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {
Snackbar snackbar = Snackbar.make(coordinatorLayout, "Simple Snackbar", Snackbar.LENGTH_LONG);
snackbar.show();
}
});
```

2. Snakbar with Action Button

MainActivity,java

```
findViewById(R.id.btnCallback).setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {
Snackbar snackbar = Snackbar
.make(coordinatorLayout, "Snackbar with Callback", Snackbar.LENGTH_LONG)
.setAction("OK", new View.OnClickListener() {
@Override
public void onClick(View view) {
Snackbar snackbar1 = Snackbar.make(coordinatorLayout, "Snackbar with Callback called.", Snackbar.LENGTH_SHORT);
snackbar1.show();
}
});
```

3.2 Custom Toast

- Sometimes Displaying the text on Toast may not be satisfactory.
- So we can extend the functionalities of toast by creating the custom toast.

Steps for Implementation of Custom Toast In Android:

Step 1: In first step Retrieve the Layout Inflater with getLayoutInflater() (or getSystemService()) and then inflate the layout from XML using inflate(int, ViewGroup). In inflate method first parameter is the layout resource ID and the second is the root View.

Step 2 : Create a new Toast with Toast(Context) and set some properties of the Toast, such as the duration and gravity.

Step 3: Call setView(View) and pass the inflated layout in this method.

Step 4: Display the Toast on the screen using show() method of Toast.

Code :-

activity_main.xml	create a xml layouts by right clicking on res/layout -> New -> Layout Resource File and name it custom_toast_layout.xml
<RelativeLayoutxmlns:android="http://schemas.android.com/apk/res/android" xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent" android:layout_height="match_parent" android:paddingBottom="@dimen/activity_vertical_margin" android:paddingLeft="@dimen/activity_horizontal_margin" android:paddingRight="@dimen/activity_horizontal_margin" android:paddingTop="@dimen/activity_vertical_margin" tools:context=".MainActivity"> <!-- Button's for simple and custom Toast --> <Button android:id="@+id/simpleToast" android:layout_width="200dp" android:layout_height="wrap_content" android:layout_centerHorizontal="true" android:layout_marginTop="150dp" android:background="#f00" android:text="Simple Toast" android:textColor="#fff" android:textSize="20sp"/> <Button android:id="@+id/customToast"	<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/android" android:id="@+id/toast_layout_root" android:layout_width="fill_parent" android:layout_height="fill_parent" android:background="#DAAA" android:orientation="horizontal" android:padding="8dp"> <!-- ImageVView and TextView for custom Toast --> <ImageView android:id="@+id/toastImageView" android:layout_width="wrap_content" android:layout_height="wrap_content" android:layout_marginRight="8dp"/> <TextView android:id="@+id/toastTextView" android:layout_width="wrap_content" android:layout_height="wrap_content" android:textColor="#FFF"/> </LinearLayout>

```
android:layout_width="200dp"
android:layout_height="wrap_content"
android:layout_below="@+id/simpleToast"
android:layout_centerHorizontal="true"
android:layout_margin="50dp"
android:background="#0f0"
android:text="Custom Toast"
android:textColor="#fff"
android:textSize="20sp"/>

</RelativeLayout>
```

MainActivity.java

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.Button;
import android.view.ViewGroup;

publicclassMainActivityextendsAppCompatActivity{

Button simpleToast, customToast;

@Override
protectedvoid onCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
// get the reference of Button's
      simpleToast =(Button) findViewById(R.id.simpleToast);
      customToast =(Button) findViewById(R.id.customToast);
// perform setOnClickListener event on simple Toast Button
      simpleToast.setOnClickListener(newView.OnClickListener(){
@Override
publicvoid onClick(View v){
// initiate a Toast with message and duration
Toast toast =Toast.makeText(getApplicationContext(),"Simple Toast In Android",Toast.LENGTH_LONG);// initiate the Toast with context, message and duration for the Toast
toast.setGravity(Gravity.BOTTOM |Gravity.CENTER_HORIZONTAL,0,0);// set gravity for the Toast.
            toast.show();// display the Toast

}
});
// perform setOnClickListener event on custom Toast Button
      customToast.setOnClickListener(newView.OnClickListener(){
@Override
publicvoid onClick(View v){
```

```
// Retrieve the Layout Inflater and inflate the layout from xml
LayoutInflater inflater = getLayoutInflater();
View layout = inflater.inflate(R.layout.custom_toast_layout,
(ViewGroup) findViewById(R.id.toast_layout_root));
// get the reference of TextView and ImageVIew from inflated layout
TextView toastTextView =(TextView) layout.findViewById(R.id.toastTextView);
ImageView toastImageView =(ImageView) layout.findViewById(R.id.toastImageView);
// set the text in the TextView
        toastTextView.setText("Custom Toast In Android");
// set the Image in the ImageView
        toastImageView.setImageResource(R.drawable.ic_launcher);
// create a new Toast using context
Toast toast =newToast(getApplicationContext());
        toast.setDuration(Toast.LENGTH_LONG);// set the duration for the Toast
        toast.setView(layout);// set the inflated layout
        toast.show();// display the custom Toast

}
});}}
```

Output: -

Custom Toast With Image

3.3 Button

- A Button is a a widget which can be pressed, or clicked, by the user to perform an some action.
- Android buttons are GUI components which are sensible to taps (clicks) by the user.
- When the user clicks on button in an Android app, the app respond to the clicks by executing suitable code written in the function onClick(View v)
- There are different types of buttons used in android such as CompoundButton, ToggleButton, RadioButton.

Button Code in XML	Output
<Button android:id="@+id/simpleButton" android:layout_width="wrap_content" android:layout_height="wrap_content" android:text="Abhi Android"/>	

Android Button Example with Listener

- We can Apply the listener on button
- Whenever user click on button some action can be performed.
- We can perform action on button using different ways such as calling listener on button or adding onClick property of button in activity's xml file.

Method 1: Calling Listner	**Method 2**: Using Activity XML

button.setOnClickListener(new
View.OnClickListener() {
@Override
public void onClick(View view) {
//code
}
});
	<Button
	android:onClick="methodName"
	/>
	Ex.
	<Button
	android:onClick="Addition"
	/>

- **Button Attributes**

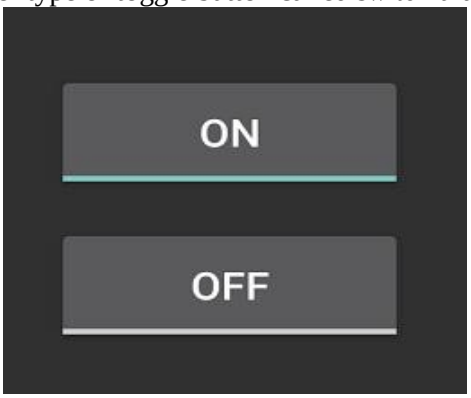
  Following are some of the XML attributes associated with Button

Table 3.4 XML attributes associated with Button

Sr.No	Name	Description
1	**android:background**	This is a drawable to use as the background.
2	**android:contentDescription**	This defines text that briefly describes content of the view.
3	**android:id**	This supplies an identifier name for this view.
4	**android:onClick**	This is the name of the method in this View's context to invoke when the view is clicked.
5	**android:visibility**	This controls the initial visibility of the view.

3.3.1 Toggle Button

- **Android Toggle Button** can be used to display checked/unchecked (On/Off) state on the button.
- It can be used to On/Off Sound, Wifi, Bluetooth etc.
- Android 4.0, there is another type of toggle button called *switch* that provides slider control.



- **Toggle Button Attributes**
  Following are some of the XML attributes associated with Button

Table 3.5 XML attributes associated with Toggle Button

Sr.No	Name	Description
1	**android:disabledAlpha**	The alpha to apply to the indicator when disabled.
2	android:textOff	The text for the button when it is not checked.
3	android:textOn	The text for the button when it is checked.

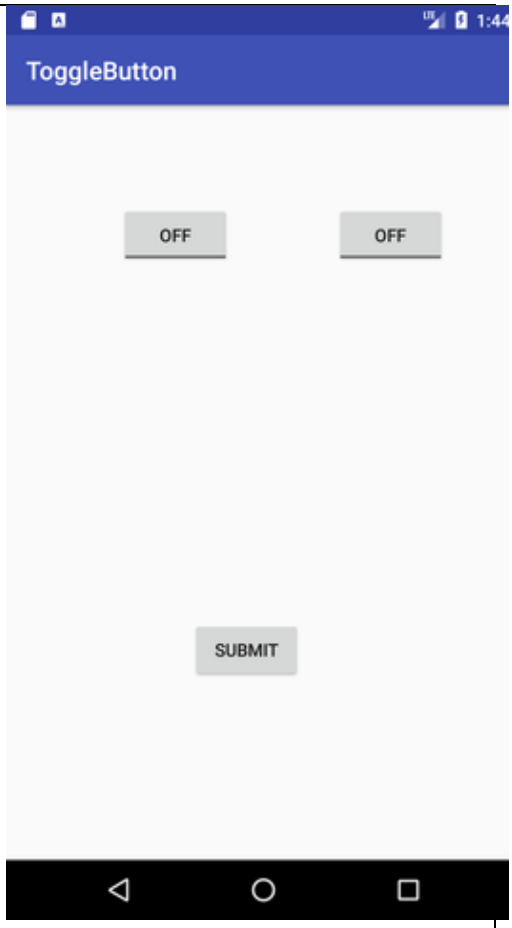
Android Toggle Button Example

activity_main.xml

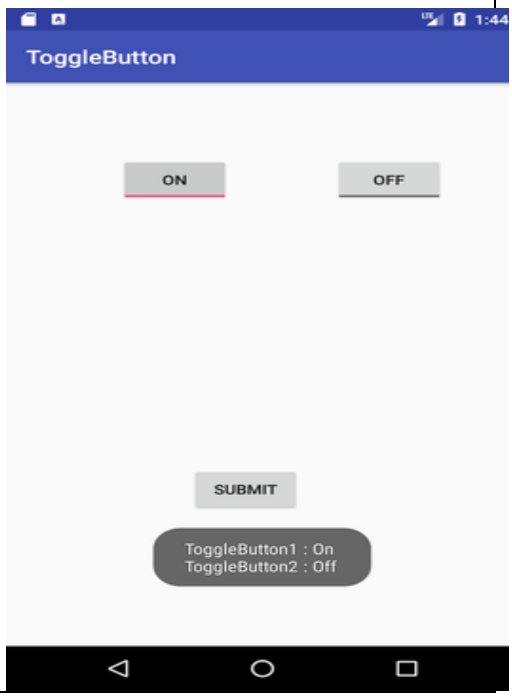
```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context="example.javatpoint.com.togglebutton.MainActivity
">

<ToggleButton
android:id="@+id/toggleButton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginLeft="8dp"
android:layout_marginTop="80dp"
android:text="ToggleButton"
android:textOff="Off"
android:textOn="On"
```

Toggle Button Before Click

```
app:layout_constraintEnd_toStartOf="@+id/toggleButton2"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent" />

<ToggleButton
android:id="@+id/toggleButton2"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginRight="60dp"
android:layout_marginTop="80dp"
android:text="ToggleButton"
android:textOff="Off"
android:textOn="On"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintTop_toTopOf="parent" />

<Button
android:id="@+id/button"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginBottom="144dp"
android:layout_marginLeft="148dp"
android:text="Submit"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintStart_toStartOf="parent" />
</android.support.constraint.ConstraintLayout>
```



Toggle Button After Click

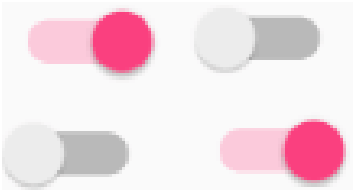


MainActivity.java

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
import android.widget.ToggleButton;

public class MainActivity extends AppCompatActivity {
private ToggleButton toggleButton1, toggleButton2;
private Button buttonSubmit;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);

addListenerOnButtonClick();
}

public void addListenerOnButtonClick(){
//Getting the ToggleButton and Button instance from the layout xml file
toggleButton1=(ToggleButton)findViewById(R.id.toggleButton);
toggleButton2=(ToggleButton)findViewById(R.id.toggleButton2);
buttonSubmit=(Button)findViewById(R.id.button);

//Performing action on button click
buttonSubmit.setOnClickListener(new View.OnClickListener(){

@Override
public void onClick(View view) {
StringBuilder result = new StringBuilder();
result.append("ToggleButton1 : ").append(toggleButton1.getText());
result.append("\nToggleButton2 : ").append(toggleButton2.getText());
//Displaying the message in toast
Toast.makeText(getApplicationContext(), result.toString(),Toast.LENGTH_LONG).show();
}
});
}
}
```

3.3.2 Switch Button

- Switch is a two-state user interface element which is used to display ON (Checked) or OFF (Unchecked) states as a button with thumb slider.
- By using thumb, the user may drag back and forth to choose an option either ON or OFF.
- It is used to change the setting between two states either **ON** or **OFF**.



- By default, the android Switch will be in OFF (Unchecked) state. We can change the default state of Switch by using android:checked attribute.
- Ex. **android:checked = "true"**
- **There are two ways to create the switch component in Android**
  1. XML 2. Activity file programmatically.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
android:layout_height="match_parent">
<Switch
   android:id="@+id/switch1"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:switchMinWidth="56dp"
   android:layout_marginLeft="100dp"
   android:layout_marginTop="120dp"
   android:text="Switch1:"
   android:checked="true"
   android:textOff="OFF"
   android:textOn="ON"/>
</RelativeLayout>
```

```
RelativeLayout layout =
(RelativeLayout)findViewById(R.id.r_layout);
Switch sb = new Switch(this);
sb.setTextOff("OFF");
sb.setTextOn("ON");
sb.setChecked(true);
layout.addView(sb);
```

Handle Switch Click Events:-

```
Switch sw = (Switch) findViewById(R.id.switch1);
sw.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
   public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
      if (isChecked) {
         // The toggle is enabled
      } else {
         // The toggle is disabled
      }
   }
});
```

- **Switch Button Attributes**

  Following are some of the XML attributes associated with Switch Button

Table 3.6 XML attributes associated with Switch Button

Sr.No	Name	Description
1	android:id	It is used to uniquely identify the control
2	android:checked	It is used to specify the current state of switch control

3	android:gravity	It is used to specify how to align the text like left, right, center, top, etc.

3.3.3 Image Button

- In Android, you can display a normal "Button", with a customized background image.
- Image Button is a button with an image that can be pressed or clicked by the users.
- It looks like a normal button with the standard button background that changes the color during different button states.

- An image on the surface of a button is defined within a xml (i.e. layout) by using src attribute (android:src="@drawable/img") or within java class by using setImageResource() method.
- ImageButton has all the properties of a normal button so you can easily perform any event like click or any other event which you can perform on a normal button.
- ImageButton code in XML:

```
<!--Make Sure you have Image Name home in Drawable Folder-->
<ImageButton
android:id="@+id/simpleImageButton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:src="@drawable/home"/>
```

3.3.4 Radio Button

- **RadioButton** is a two states button which is either checked or unchecked.
- Clicking an unchecked button changes its state to "checked" state and "unchecked" for the previously selected radio button.
- If RadioButtons are in group, when one RadioButton within a group is selected, all others are automatically deselected.

Steps for Implementing Radio Button
Step 1:
Custom String
Open "res/values/strings.xml" file, add some custom string for radio button.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
```

```
<string name="hello">Hello World, MyAndroidAppActivity!</string>
<string name="app_name">MyAndroidApp</string>
<string name="radio_male">Male</string>
<string name="radio_female">Female</string>
<string name="btn_display">Display</string>
</resources>
```

Step 2: RadioButton

Open "res/layout/main.xml" file, add "RadioGroup", "RadioButton" and a button, inside the LinearLayout.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >

<RadioGroup
      android:id="@+id/radioSex"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" >

<RadioButton
         android:id="@+id/radioMale"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="@string/radio_male"
         android:checked="true" />

<RadioButton
         android:id="@+id/radioFemale"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="@string/radio_female" />

</RadioGroup>

<Button
      android:id="@+id/btnDisplay"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/btn_display" />
</LinearLayout>
```

Step 3. MainAcitivty.java

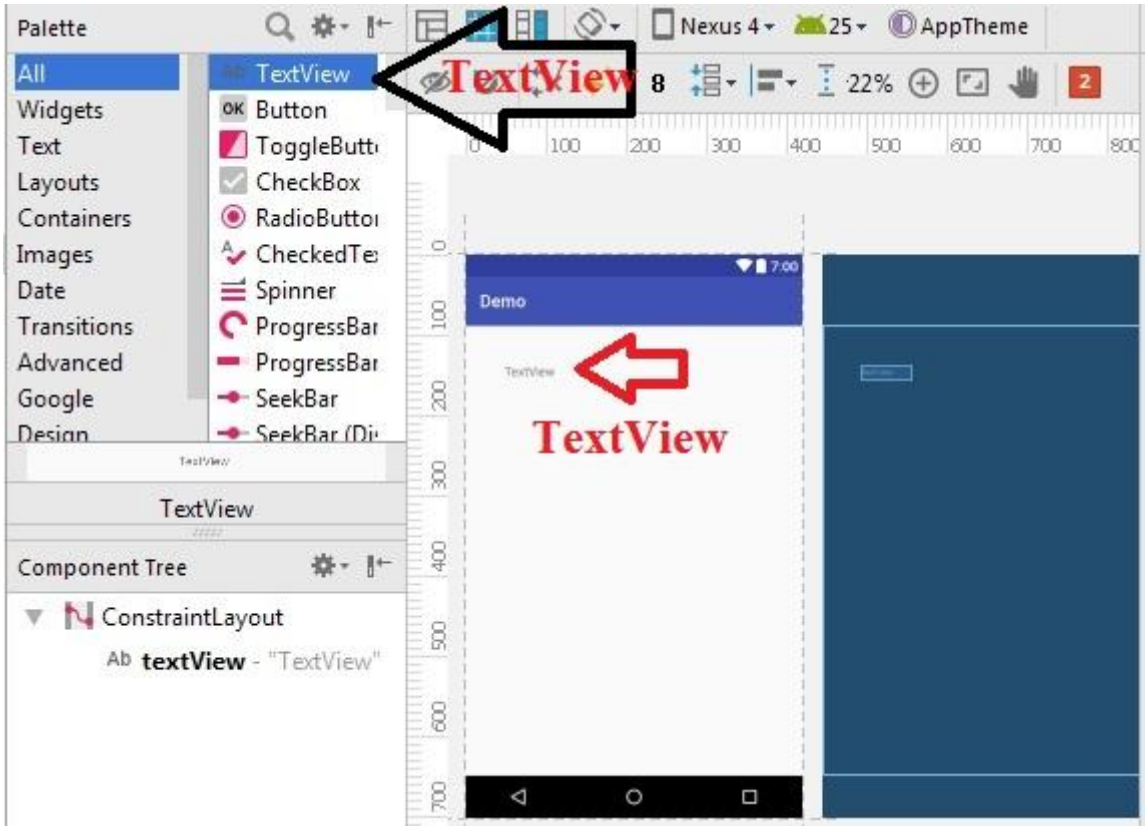
```
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
```

```
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.Toast;

publicclassMainActivityextendsActivity{
private RadioGroup radioSexGroup;
private RadioButton radioSexButton;
private Button btnDisplay;

@Override
publicvoidonCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
addListenerOnButton();
}

publicvoidaddListenerOnButton(){

radioSexGroup =(RadioGroup)findViewById(R.id.radioSex);
btnDisplay =(Button)findViewById(R.id.btnDisplay);

btnDisplay.setOnClickListener(newOnClickListener(){

@Override
publicvoidonClick(View v){

// get selected radio button from radioGroup
int selectedId = radioSexGroup.getCheckedRadioButtonId();

// find the radiobutton by returned id
radioSexButton =(RadioButton)findViewById(selectedId);
Toast.makeText(MyAndroidAppActivity.this,
radioSexButton.getText(), Toast.LENGTH_SHORT).show();

} }); } }
```

3.4 TextView, EditText And CheckBox

A TextView Is a Component used todisplays text to the user

- TextView is a complete text editor,



- **Text View Attributes**

Following are some of the Important XML attributes associated with TextView

Table 3.7 XML attributes associated with Text View

Sr.No	Name	Description
1	android:id	It is used to uniquely identify the control
2	**android:capitalize**	If set, specifies that this TextView has a textual input method and should automatically capitalize what the user types. • Don't automatically capitalize anything - 0 • Capitalize the first word of each sentence - 1 • Capitalize the first letter of every word - 2 • Capitalize every character - 3
3	**android:cursorVisible**	Makes the cursor visible (the default) or invisible. Default is false..

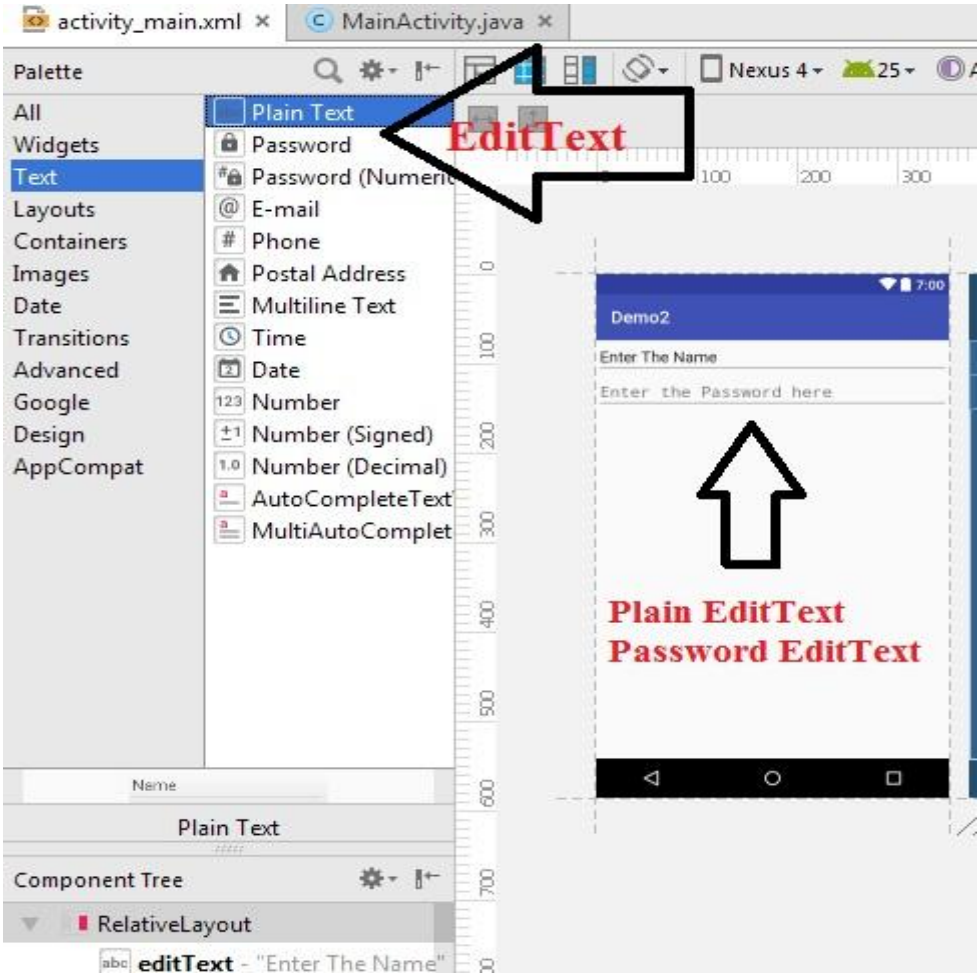
4	**android:fontFamily**	Font family (named by string) for the text.
5	**android:gravity**	Specifies how to align the text by the view's x- and/or y-axis when the text is smaller than the view.
6	**android:hint**	Hint text to display when the text is empty.
7	**android:password**	Whether the characters of the field are displayed as password dots instead of themselves. Possible value either "true" or "false".
8	**android:text**	Text to display.

Code:-

TextView Code in XML	TextView Code in Activity
<TextViewandroid:id="@+id/simpleTextView" android:layout_width="wrap_content" android:layout_height="wrap_content" android:text="Hello World"/>	TextView textView =(TextView) findViewById(R.id.textView); textView.setText("Hello World");//set text for text view

EditText:-

- **EditText in used in applications in order to provide an input or text field, especially in forms.**
- It is an overlay over TextView that configures itself to be editable.
- Android **EditText** is a subclass of *TextView*.



- **EditText Attributes**
  Following are some of the Important XML attributes associated with EditText

Table 3.8 XML attributes associated with Edit Text

Sr.No	Name	Description
1	android:id	It is used to uniquely identify the control
2	**android:background**	This is a Drawable to use as the background.
3	**android:contentDescription**	This defines text that briefly describes content of the view.
4	**android:onClick**	This is the name of the method in this View's context to invoke when the view is clicked.
5	**android:visibility**	This controls the initial visibility of the view.

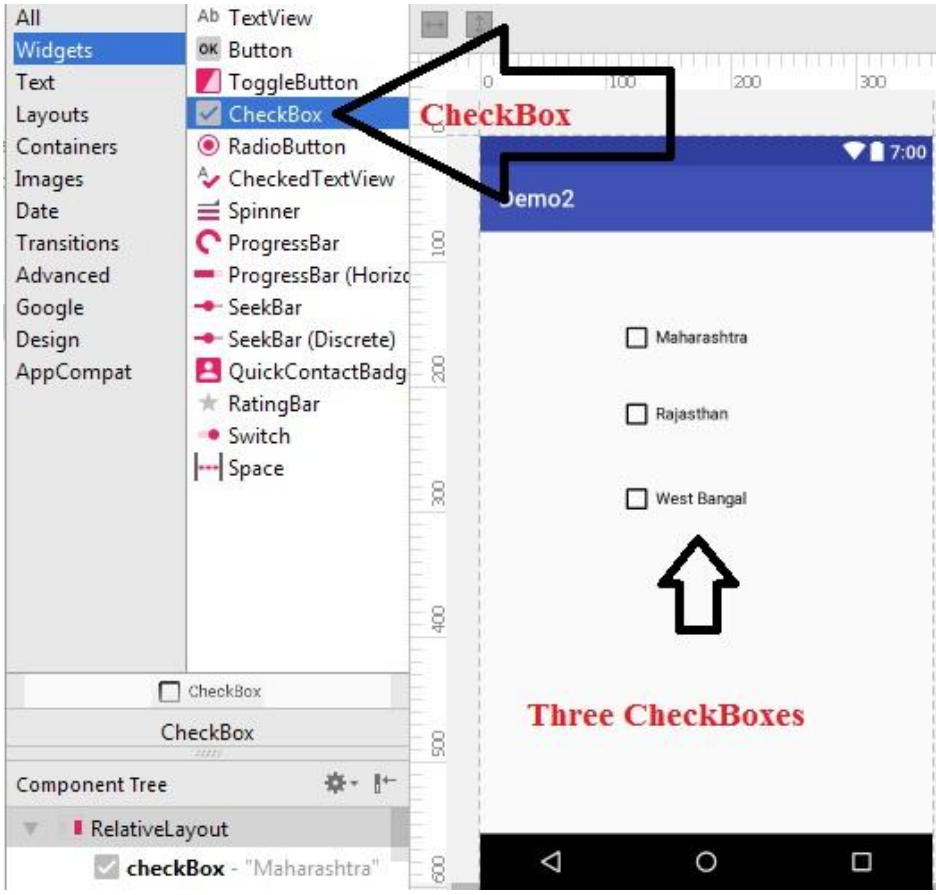
Code:-

EditText Code in XML	TextView Code in Activity

<EditText
android:id="@+id/edittext"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_alignLeft="@+id/button"
android:layout_below="@+id/textView1"
android:layout_marginTop="61dp"
android:ems="10"
android:text="@string/enter_text"
android:inputType="text" />
/>
	TextView textView =(TextView) public class MainActivity extends Activity {
	EditText eText;
	Button btn;
	
	@Override
	protected void onCreate(Bundle savedInstanceState) {
	super.onCreate(savedInstanceState);
	setContentView(R.layout.activity_main);
	eText = (EditText) findViewById(R.id.edittext);
	btn = (Button) findViewById(R.id.button);
	btn.setOnClickListener(new OnClickListener() {
	public void onClick(View v) {
	String str = eText.getText().toString();
	Toast msg = Toast.makeText(getBaseContext(),str,Toast.LENGTH_LONG);
	msg.show();
	} }); }

CheckBox:-

- **Android CheckBox** is a type of two state button either checked or unchecked.
- You should use check-boxes when presenting users with a group of selectable options that are not mutually exclusive.
- For example, it can be used to know the hobby of the user, activate/deactivate the specific action etc.



- You can check the current state of a check box programmatically by using isChecked() method. This method returns a Boolean value either true or false, if a check box is checked then it returns true otherwise it returns false. Below is an example code in which we checked the current state of a check box.
  Code Snippet

```
//initiate a check box
CheckBox simpleCheckBox =(CheckBox) findViewById(R.id.simpleCheckBox);

//check current state of a check box (true or false)
Boolean checkBoxState = simpleCheckBox.isChecked();
```

- **CheckBox Attributes**
  Following are some of the Important XML attributes associated with CheckBox

Table 3.9 XML attributes associated with CheckBox

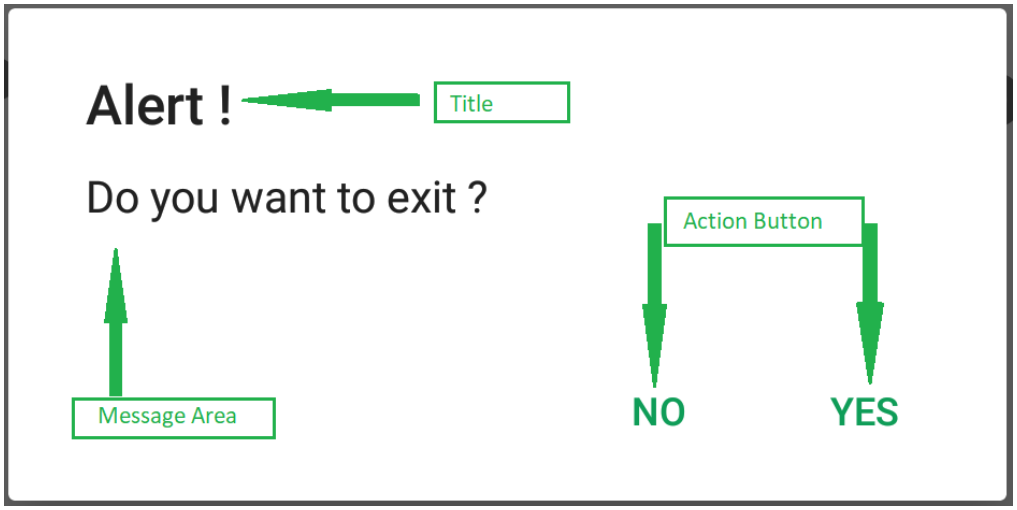
Sr.No	Name	Description
1	android:id	It is used to uniquely identify the control
2	**android:checked="true"**	checked is an attribute of check box used to set the current state of a check box. The value should be true or false where true shows the checked state and false shows unchecked state of a check box.
3	android:gravity="right|center_vertical"	The gravity attribute is an optional attribute which is used to control the alignment of the text in CheckBox like left, right, center, top, bottom,

		center_vertical, center_horizontal etc.
4	android:text="Text Attribute Of Check Box"	text attribute is used to set the text in a check box. We can set the text in xml as well as in the java class.
5	android:textColor="#f00"	textColor attribute is used to set the text color of a check box. Color value is in form of “#argb”, “#rgb”, “#rrggbb”, or “#aarrggbb”.

3.5 Alert Dialog and Button Sheets

Alert Dialog:-

- **Android AlertDialog** is used to display the dialog message with OK and Cancel buttons.
- **It Is** used to interrupt and ask the user about his/her choice to continue or discontinue.
- Android AlertDialog is composed of three regions: title, content area and action buttons.
- It is a subclass of Dialog class.



Android Alert Dialog is built with the use of three **fields**: **Title, Message area, Action Button.** Alert Dialog code has three **methods**:

- **setTitle()** method for displaying the Alert Dialog box Title
- **setMessage()** method for displaying the message
- **setIcon()** method is use to set the icon on Alert dialog box.

Following are the steps to create a AlertDialog :-

Step 1: Create a new project. After that, you have java and XML file.

Step 2: Open your XML file and then add TextView for message as shown below (you can change it accordingly).

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Press The Back Button of Your Phone."
    android:textStyle="bold"
    android:textSize="30dp"
    android:gravity="center_horizontal"
    android:layout_marginTop="180dp"
    />

</RelativeLayout>
```

Step 3: Now, open up the activity java file. After, on create method declaration, the onbackpressed method is called when you click the back button of your device.

Step 4: Create the object of Builder class Using AlertDialog.Builder. Now, set the Title, message.

Step 5: In a builder object set the positive Button now gives the button name and add the OnClickListener of DialogInterface. Same as with the negative Button, at last, create the Alert dialog Box with builder object and then show the Alert Dialog.

Step 6: Now if positive button press finish the app goto outside from the app if negative then finish the dialog box

Step 7: Now run it and then press the back button. After that click Yes or No Button.

The complete code of MainActivity for Alert Dialog is given below:

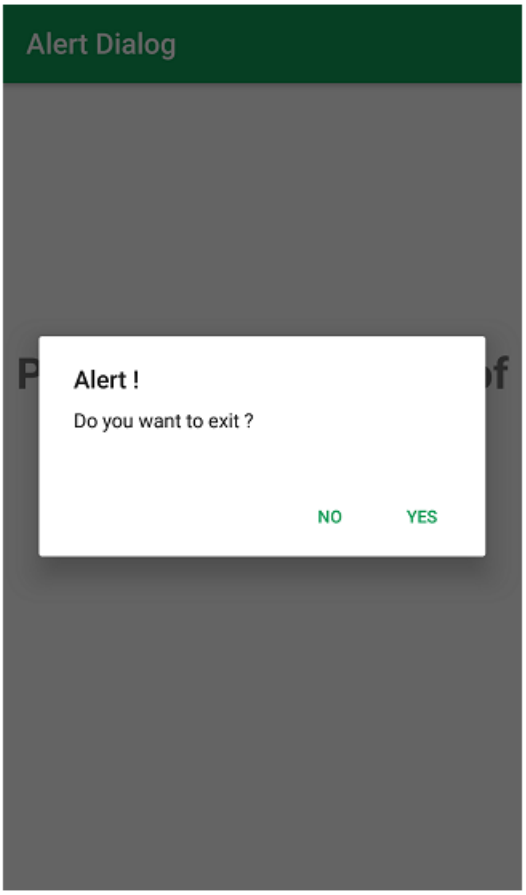
```
import android.content.DialogInterface;
import android.support.v7.app.AlertDialog;
import
android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends
AppCompatActivity {

@Override
protected void onCreate(Bundle
```

```
@Override
public void onClick(DialogInterface dialog,
int which)
{

// When the user click yes button
// then app will close
finish();
}
});
```

```
savedInstanceState)
{
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
}

// Declare the onBackPressed method
// when the back button is pressed
// this method will call
@Override
public void onBackPressed()
{

// Create the object of
// AlertDialog Builder class
AlertDialog.Builder builder
= new AlertDialog
.Builder(MainActivity.this);

// Set the message show for the Alert time
builder.setMessage("Do you want to exit ?");

// Set Alert Title
builder.setTitle("Alert !");

// Set Cancelable false
// for when the user clicks on the outside
// the Dialog Box then it will remain show
builder.setCancelable(false);

// Set the positive button with yes name
// OnClickListener method is use of
// DialogInterface interface.

builder
.setPositiveButton(
"Yes",
new DialogInterface
.OnClickListener() {
```

```
// Set the Negative button with No name
// OnClickListener method is use
// of DialogInterface interface.
builder
.setNegativeButton(
"No",
new DialogInterface
.OnClickListener() {

@Override
public void onClick(DialogInterface dialog,
int which)
{

// If user click no
// then dialog box is canceled.
dialog.cancel();
}
});

// Create the Alert dialog
AlertDialog alertDialog = builder.create();

// Show the Alert Dialog box
alertDialog.show();
}
}
```

Output:-

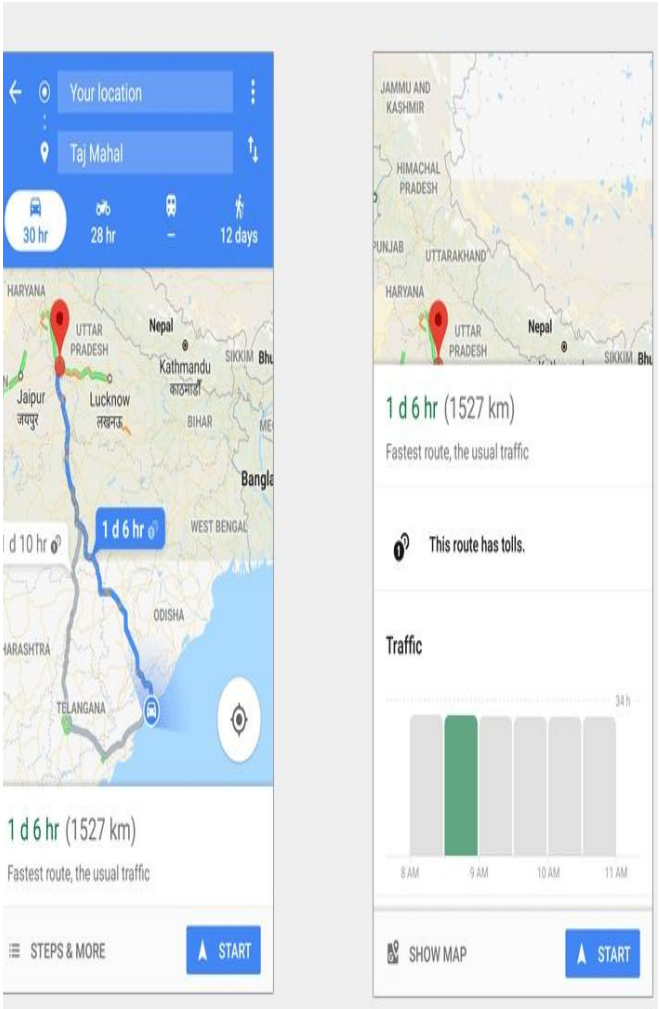


Bottom Sheets:-

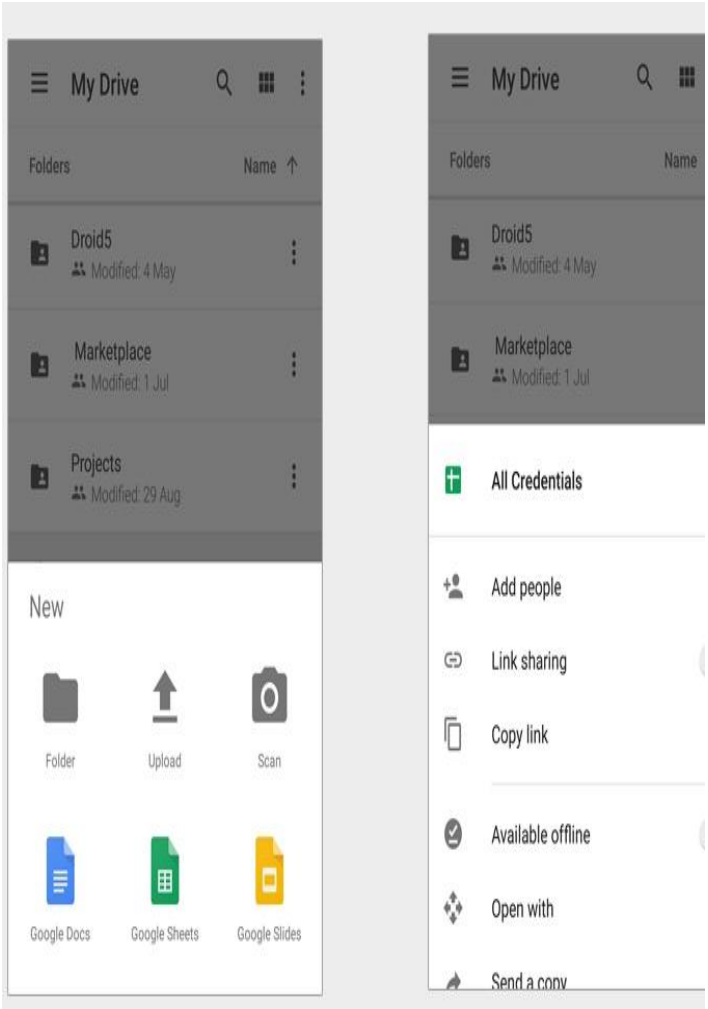
- Android Bottom Sheet component slides up from the bottom showing more relevant content.
- Bottom Sheet is a really nice way to create improved UI and UX designs.
- They are used by almost all big applications like Uber, Google Maps, Google Docs and almost any other Google application.
- Android comes with built in support for making bottom sheets.

There are two major types of bottom sheets:

- **Persistent bottom sheet:** Normally used to display additional information to that shown in the main view. These panels have the same elevation as the main content and remain visible even when not being actively used.
- **Modal bottom sheet:** Commonly used as an alternative to simple menus or dialogs. They have a higher elevation than the main content, darken the display, and it is necessary to hide them continue to interact with the rest of the application.



a) Persistent bottom sheet



b) Modal bottom sheet

3.6 Spinner

- It is used display the multiple options to the user in which only one item can be selected by the user.
- Android spinner is like the drop down menu with multiple values from which the user can select only one value.
- Android spinner is associated with AdapterView. So you need to use one of the adapter classes with spinner.

Code Snippet :-Activity.XML

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context="example.javatpoint.com.spinner.MainActivity">

<Spinner
android:id="@+id/spinner"
android:layout_width="149dp"
android:layout_height="40dp"
android:layout_marginBottom="8dp"
android:layout_marginEnd="8dp"
android:layout_marginStart="8dp"
android:layout_marginTop="8dp"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.502"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintVertical_bias="0.498" />

</android.support.constraint.ConstraintLayout>
```

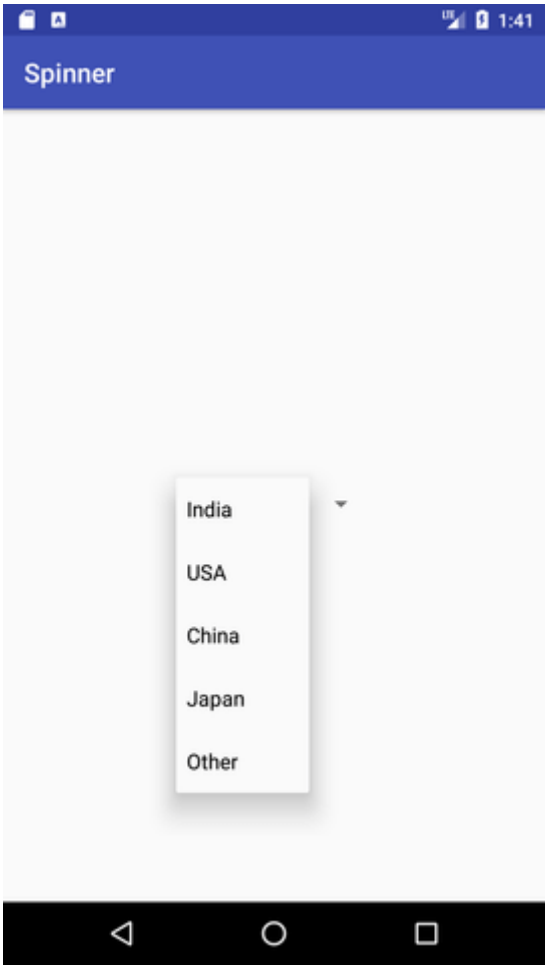
MainActivity.java

```
import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.AdapterView;

import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.Toast;
```

```
public class MainActivity extends AppCompatActivity implements
AdapterView.OnItemSelectedListener {
String[] country = { "India", "USA", "China", "Japan", "Other"};
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
//Getting the instance of Spinner and applying OnItemSelectedListener on it
Spinner spin = (Spinner) findViewById(R.id.spinner);
spin.setOnItemSelectedListener(this);
//Creating the ArrayAdapter instance having the country list
ArrayAdapter aa = new ArrayAdapter(this,android.R.layout.simple_spinner_item,country);
aa.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
//Setting the ArrayAdapter data on the Spinner
spin.setAdapter(aa);  }
//Performing action onItemSelected and onNothing selected
@Override
public void onItemSelected(AdapterView<?> arg0, View arg1, int position, long id) {
Toast.makeText(getApplicationContext(),country[position] , Toast.LENGTH_LONG).show();
} @Override
public void onNothingSelected(AdapterView<?> arg0) {
// TODO Auto-generated method stub
}  }
```

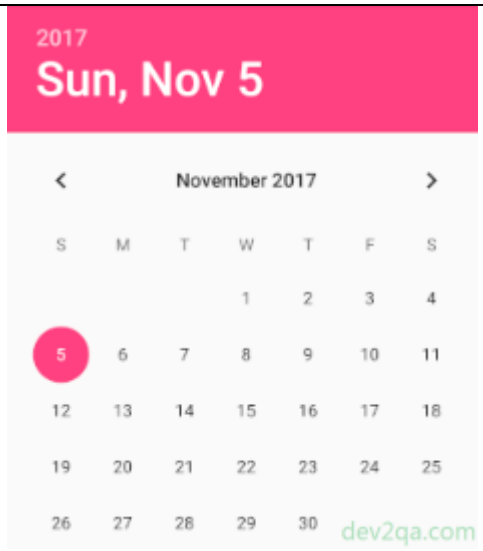
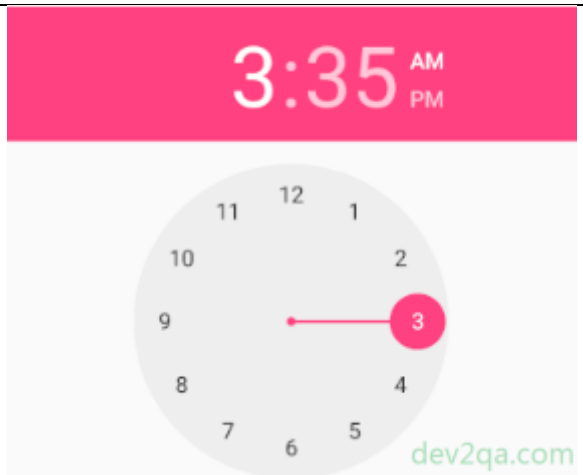
Output:-


1:41
Spinner
India


1:41
Spinner
India
USA
China
Japan
Other

3.7 Date Picker and Time Picker

- They are used to display date and time selection widget, in android application.
- They can be used in either spinner mode or calendar mode (date picker), clock mode (time picker).
- User can control their appearance with their properties.

Date Picker Calender Mode	Date Picker Spinner Mode
	
Time Picker Clock Mode	**Time Picker Spinner Mode**
	

- **Some Properties of Date Picker**
  Following are some of the Important Properties of Date Picker

Table 3.10 Important Properties of Date Picker

Sr.No	Name	Description
1	**datePickerMode**	Value can be spinner or calendar. If set to calendar, it will display a calendar which let you choose date. If set to spinner, it will display a spinner to let you choose date.
2	**calendarViewShown**	This is a boolean value, only take effect when **datePickerMode** is spinner. If set to true, it will display a claendar.
3	**spinnersShown**	This is a boolean value also, only take effect when **datePickerMode** is spinner. If set to true, it will display a spinner.
4	**minDate**	Set the optional minimum date in mm/dd/yyy format.
5	**maxDate**	Set the maximum date to select, format is mm/dd/yyyy.
6	**startYear**	Set optional start year.
7	**endYear**	Set optional end year

- **Some Properties of Time Picker**
  Following are some of the Important Properties of Time Picker

Table 3.11 Important Properties of Time Picker

Sr.No	Name	Description
1	**timePickerMode()**	Value can be spinner or clock. When set it' value to clock, it will display a clock. When set is's value to spinner will display a spinner.
2	**headerBackground**	This is a Color or Drawable resource id which can be set to TimePicker's header background.
3	**is24HourView()**	Check whether it is 24 hour time system.
4	**setIs24HourView(boolean 24HourView)** :	Set if use 24 hour time system.
5	**getHour()**	Get current hour integer value.
6	**getMinute()**	Get current minute integer value.
7	**setOnTimeChangedListener()** :	Set call back listener when time is changed.

3.8 Rating Bar and Progress Bar

Rating Bar:-

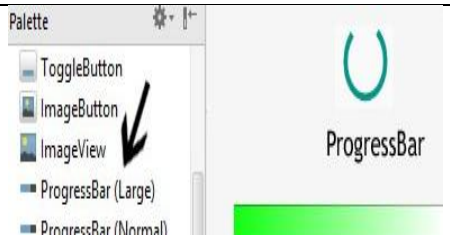
- RatingBar is used to get the rating from the app user.
- A user can simply touch, drag or click on the stars to set the rating value. The value of rating always returns a floating point number which may be 1.0, 2.5, 4.5 etc.

Code Snippet:-

<RatingBar android:id="@+id/simpleRatingBar" android:layout_width="wrap_content" android:layout_height="wrap_content"/>	

Progress Bar:-

- In Android, ProgressBar is used to display the status of work being done like analyzing status of work or downloading a file etc.
- In Android, by default a progress bar will be displayed as a spinning wheel but If user want it to be displayed as a horizontal bar then you can to use style attribute as horizontal.

<ProgressBar android:id="@+id/simpleProgressBar" android:layout_width="fill_parent" android:layout_height="wrap_content" style="@style/Widget.AppCompat.ProgressBar.Horizontal"/>	

3.9 File Download

- There are many ways to download files. Below we will see most common ways;
- Use AsyncTask and show the download progress in a dialog
- This method will allow you to execute some background processes and update the UI at the same time (in this case, we'll update a progress bar).

```
/ declare the dialog as a member field of your activity
ProgressDialog mProgressDialog;

// instantiate it within the onCreate method
mProgressDialog =newProgressDialog(YourActivity.this);
mProgressDialog.setMessage("A message");
mProgressDialog.setIndeterminate(true);
mProgressDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
mProgressDialog.setCancelable(true);

// execute this when the downloader must be fired
finalDownloadTask downloadTask =newDownloadTask(YourActivity.this);
downloadTask.execute("the url to the file you want to download");

mProgressDialog.setOnCancelListener(newDialogInterface.OnCancelListener(){

@Override
publicvoid onCancel(DialogInterface dialog){
      downloadTask.cancel(true);//cancel the task
}
});
```

The AsyncTask will look like this:

```
// usually, subclasses of AsyncTask are declared inside the activity class.
// that way, you can easily modify the UI thread from here
privateclassDownloadTaskextendsAsyncTask<String,Integer,String>{

privateContext context;
privatePowerManager.WakeLock mWakeLock;

publicDownloadTask(Context context){
this.context = context;
}

@Override
protectedString doInBackground(String... sUrl){
InputStream input =null;
OutputStream output =null;
HttpURLConnection connection =null;
try{
URL url =new URL(sUrl[0]);
connection =(HttpURLConnection) url.openConnection();
connection.connect();

// expect HTTP 200 OK, so we don't mistakenly save error report
```

```
// instead of the file
if(connection.getResponseCode()!=HttpURLConnection.HTTP_OK){
return"Server returned HTTP "+ connection.getResponseCode()
+" "+ connection.getResponseMessage();
}

// this will be useful to display download percentage
// might be -1: server did not report the length
int fileLength = connection.getContentLength();

// download the file
input = connection.getInputStream();
output =newFileOutputStream("/sdcard/file_name.extension");

byte data[]=newbyte[4096];
long total =0;
int count;
while((count = input.read(data))!=-1){
// allow canceling with back button
if(isCancelled()){
input.close();
returnnull;
}
total += count;
// publishing the progress....
if(fileLength >0)// only if total length is known
publishProgress((int)(total *100/ fileLength));
output.write(data,0, count);
}
}catch(Exception e){
return e.toString();
}finally{
try{
if(output !=null)
output.close();
if(input !=null)
input.close();
}catch(IOException ignored){
}

if(connection !=null)
connection.disconnect();
}
returnnull;
}
```

The method above (doInBackground) runs always on a background thread. You shouldn't do any UI tasks there. On the other hand, the onProgressUpdate and onPreExecute run on the UI thread, so there you can change the progress bar:

```
@Override
protectedvoid onPreExecute(){
super.onPreExecute();
// take CPU lock to prevent CPU from going off if the user
// presses the power button during download
PowerManager                                                    pm
```

```
=(PowerManager)context.getSystemService(Context.POWER_SERVICE);
mWakeLock = pm.newWakeLock(PowerManager.PARTIAL_WAKE_LOCK,
getClass().getName());
mWakeLock.acquire();
mProgressDialog.show();
}

@Override
protectedvoid onProgressUpdate(Integer... progress){
super.onProgressUpdate(progress);
// if we get here, length is known, now set indeterminate to false
mProgressDialog.setIndeterminate(false);
mProgressDialog.setMax(100);
mProgressDialog.setProgress(progress[0]);
}

@Override
protectedvoid onPostExecute(String result){
mWakeLock.release();
mProgressDialog.dismiss();
if(result !=null)
Toast.makeText(context,"Download error: "+result,Toast.LENGTH_LONG).show();
else
Toast.makeText(context,"File downloaded",Toast.LENGTH_SHORT).show(); }
```

For this to run, you need the WAKE_LOCK permission.

```
<uses-permission android:name="android.permission.WAKE_LOCK"/>
```

Summary:

After studying this chapter students will be able to :

- Understand Fundamentals of User Interface
- Understand Android UI Components
- Work of Toast
- Work of Buttons
- Work of TextView, EditText and Checkboxes
- Work of Spinners & use of Adapters
- Work of Rating bar and Progress bar and Its implementations

Exercise:

1 What is Toast in Android?
2 Define Custom Toast with Example.
3 Write a short note on Snakbar.
4 Define Buttons in Android in Details.
5 Write the difference between toggle button and switch button?
6 Write a short note on TextView,EditText and Checkboxes.
7 Write a program to Print “Welcome to Android” using Textview and set the color,size and style of the text.
8 Explain with Example use of Alert Dialog.
9 Write a program to create a drop down list of Countries in the World using Spinner.
10 Explaint the Date and Time Picker with their types
11 Write a program (XML,Java Class) to show the Ratings of Movie using Rating Bar
12 Differentiate Between Rating bar and Progress Bar
13 Describe File Download in Android.

Unit 4.
Android Terminologies and Resource Handling

Learning Objectives:

After going through this unit , you will be able to learn:

- Activities and Activity Life Cycle
- Context
- Intent and types of intent.
- Notification service
- Adapter recourse

4.1 Terminologies

Following is the list of android terminologies;

Table 4.1 the list of android terminologies

XML	In Android, XML file is used for designing the application's user interface like creating layouts, views, buttons, text fields etc. and it is used in parsing data feeds from the internet.
View	A view is an UI which occupies rectangular area on the screen to draw and handle user events.
Layout	Layout is the parent of view. It organizes all the views in a proper manner on the screen.
Activity	An activity is a device's screen which users see and interacts. User can place UI elements in any order in the created window of user's choice.
Emulator	An emulator is an Android virtual device through which you can select the target Android version or platform to run and test your own developed application.
Manifest file	It is a metadata file for every application. This file contains all the essential information about the application like app icon, app name, launcher activity. User can set all the permission like Send SMS, Bluetooth, Camera in this file.
Service	Services are the process which runs in the Background. It is not associated with any activity as there is no UI. Any other component of the application can start a service and this service will continue to run even when the user switches from one application to another.
Broadcast Receiver	Broadcast Receiver is another building block of Android application development which allows you to register for system and application events. It works in such a way that, when the event triggers for the first time all the registered receivers through this broadcast receiver will get notified for all the events by Android Runtime.

Content Providers	Content Providers are used to share data between two applications. This can be implemented in two ways: 1. When you want to implement the existing content provider in another application. 2. When you want to create a new content provider that can share its data with other applications
Intent	Intent is a message passing mechanism which is used to communicate between two or more components like services, activities , broadcast receiver etc. Intent can also be used to start an activity or service or to deliver broadcast messages.

Let's have a look on the Terminologies in Details

4.1.1Context

- The Context class gives you access to the global information about an application's environment.
- It lets you to Access the apps resources and classes and Communicate with other app components
- The context also gives you access to the *System Services*. This enables you to interact with the various managers, some examples being the:
  1. AlarmManager
  2. AudioManager
  3. Location Manager
- Context also helps the current activity to interact with outside android environment like local files, databases, class loaders associated to the environment.

Methods to get the Context: Following are the methods to get the context

1. getApplicationContext(),
2. getContext(),
3. getBaseContext()
4. or this (when in the activity class).

Example:-

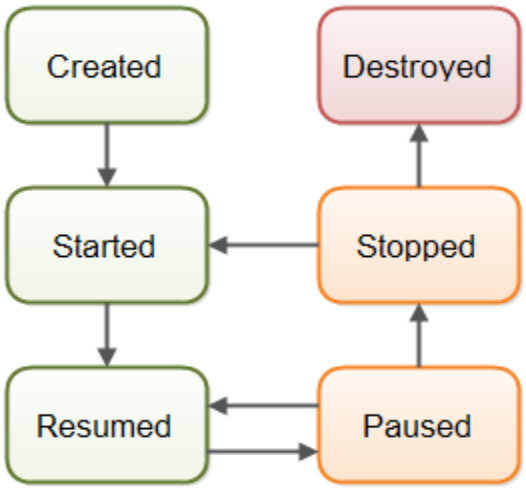
```
//Creating ui instance

ImageButton button = new ImageButton(getContext());

//creating adapter

ListAdapter adapter = new SimpleCursorAdapter(getApplicationContext(), ...);

//querying content provider
```

```
getApplicationContext().getContentResolver().query(uri, ...);

//start activity. Here this means activity context

Intent intent = new Intent(this, SecondActivity.class);
```

4.1.2 Activity

- An Android activity is one screen of the Android app's user interface, where user interacts.
- Android activity is very similar to windows in a desktop application.
- Android App may have one or more activities (Screens)
- The Android app starts by showing the main activity, and from there the app may make it possible to open additional activities.

Activity Life Cycle



Android Activity Lifecycle methods:-

Table 4.2 Android Activity Lifecycle methods descriptions.

Methods	Description
onCreate	called when activity is first created. When user opens an app
onStart	called when activity is becoming visible to the user.
onResume	called when activity will start interacting with the user.
onPause	called when activity is not visible to the user.
onStop	called when activity is no longer visible to the user.
onRestart	called after your activity is stopped, prior to start.
onDestroy	called before the activity is destroyed.

Android Activity Lifecycle Example

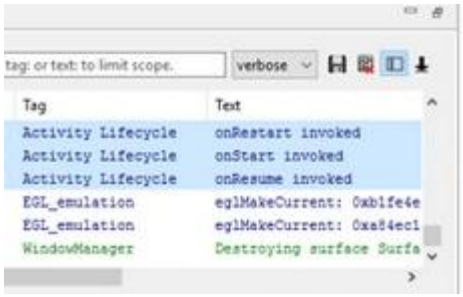
```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      Log.d("Activity Lifecycle","onCreate invoked");
   }
   @Override
   protected void onStart() {
      super.onStart();
      Log.d("Activity Lifecycle","onStart invoked");
   }
   @Override
   protected void onResume() {
      super.onResume();
      Log.d("Activity Lifecycle","onResume invoked");
   }
   @Override
   protected void onPause() {
      super.onPause();
      Log.d("Activity Lifecycle","onPause invoked");
   }
   @Override
   protected void onStop() {
      super.onStop();
      Log.d("Activity Lifecycle","onStop invoked");
   }
   @Override
```

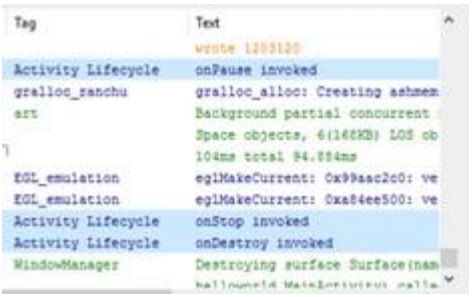
```
    protected void onRestart() {
        super.onRestart();
        Log.d("Activity Lifecycle","onRestart invoked");
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        Log.d("Activity Lifecycle","onDestroy invoked");
    }
}
```

The code of **AndroidManifest.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.tutlane.helloworld" >

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme" >
        <activity android:name=".MainActivity" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Now open Android Device Monitor (Android Virtual Device Monitor) to see our log messages in LogCat window in android studio like as shown below





4.1.3 Intent

- Intent is a message passing mechanism which is used to communicate between two or more components like services, activities , broadcast receiver etc. Intent can also be used to start an activity or service or to deliver broadcast messages.
- Intents are also used to transfer data between activities.
- Following are the uses of Intent
  1. For Launching an Activity
  2. To start a New Service
  3. For Broadcasting Messages
  4. To Display a list of contacts in List View

4.1.3.1 Types of Intent

Intent is of two types:

1. Implicit Intent
2. Explicit Intent

1. Implicit Intent:-

An Implicit intent specifies an action that can invoke any app on the device to be able to perform an action. Using an Implicit Intent is useful when your app cannot perform the certaub action but other apps probably can and you'd like the user to pick which app to use.

Some of the examples of implicit intents are as follows:

- Call
- Dialpad
- Contact
- Browser
- Call Log
- Gallery
- Camera

Syntax:

Intent i=new Intent();

i.setAction(Intent.ACTION_SEND);

2. Explicit Intent:

An explicit intent is an Intent where you explicitly define the component that needs to be called by the Android System. An explicit intent is one that you can use to launch a specific app component, such as a particular activity or service in your app.

Syntax:

Intent I = new Intent(getApplicationContext(),NextActivity.class);

I.putExtra("value1" , "This value for Next Activity");

I.putExtra("value2" , "This value for Next Activity");

4.1.4 Linking Activity using Intent

An Android application can contain zero or more activities. When your application has more than one activity, you may need to navigate from one activity to another.

Steps to Link Activities

Step 1.create new android project.

Step 2.add new Activity by right click on package name under src folder then choose New –> Other –> AndroidActivity, And give it a name (Ex. "Activity2") and press finish button.

Step 3. After that add newly created activity to AndroidManifest.xml

Step 4: Then go to res –> layout –> right click –> New –> Other –> Android XML File …

(Ex. activity_activity2.xml)

then add this code to it.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

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

</LinearLayout>
```

Return back again to strings.xml to add txtactivity by adding this line

```
    <string name="txtactivity">This is Activity 2</string>
```

Step5: go to first activity xml (activity_main.xml file) and create a button by adding this code …

```
    <Button
        android:id="@+id/btnGo"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/Go" />
```

And in strings.xml add a text for button like this

```
    <string name="Go">Go To Activity 2</string>
```

Step6: Then go to MainActivity.java class to link your button in .xml with your button in .java class, and after that when button pressed we write the code that enable us to navigate to another Activity.

```
public class MainActivity extends Activity {

    Button next;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        next = (Button) findViewById(R.id.btnGo);
        next.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                Intent secondScreen = new Intent(getApplicationContext(),
                        Activity2.class);
                startActivity(secondScreen);
            }
        });
    }
}
```

4.1.5 Calling Build-In Application using Intent

- You can call in built application using Intents like web browser, Android Caller, Map, Contact
- Following is the example which demonstrate the working of intents.

Activity_main.XML

```
<?xml version=”1.0” encoding=”utf-8”?>
   <LinearLayout xmlns:android=”http://schemas.android.com/apk/res/android”
        android:orientation=”vertical”
         android:layout_width=”fill_parent”
         android:layout_height=”fill_parent” >
       <Button
       android:id=”@+id/btn_webbrowser”
        android:layout_width=”fill_parent”
        android:layout_height=”wrap_content”
        android:text=”Web Browser” />
      <Button
        android:id=”@+id/btn_makecalls”
        android:layout_width=”fill_parent”
        android:layout_height=”wrap_content”
        android:text=”Make Calls” />
     <Button
        android:id=”@+id/btn_showMap”
       android:layout_width=”fill_parent”
       android:layout_height=”wrap_content”
       android:text=”Show Map” />
    <Button
      android:id=”@+id/btn_chooseContact”
      android:layout_width=”fill_parent”
     android:layout_height=”wrap_content”
      android:text=”Choose Contact” />
</LinearLayout>
```

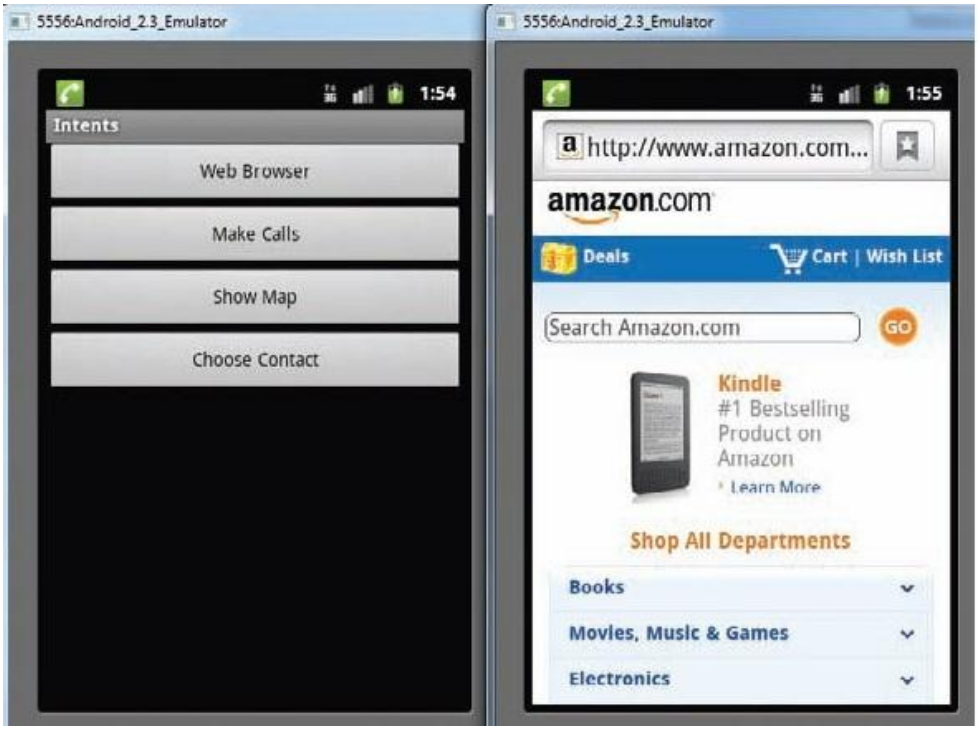
MainActivity.java

```
import android.app.Activity;
     import android.os.Bundle;
     import android.content.Intent;
     import android.net.Uri;
     import android.provider.ContactsContract;
     import android.view.View;
     import android.view.View.OnClickListener;
     import android.widget.Button;
     import android.widget.Toast;
```

```
    public class MainActivity extends Activity
   {
    Button b1, b2, b3, b4;
    int request_Code = 1;

 // Called when the activity is first created.
    public void onCreate(Bundle savedInstanceState)
   {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.main);

       b1 = (Button) findViewById(R.id.btn_webbrowser);
       b1.setOnClickListener(new OnClickListener()
      {
         public void onClick(View arg0){
        Intent i = new
        Intent(android.content.Intent.ACTION_VIEW,
        Uri.parse("http://www.amazon.com"));
        startActivity(i);
       }
    });


//Make calls button---
   b2 = (Button) findViewById(R.id.btn_makecalls);
   b2.setOnClickListener(new OnClickListener()
  {
    public void onClick(View arg0){
    Intent i = new
   Intent(android.content.Intent.ACTION_DIAL,
   Uri.parse("tel:+919767637772"));
    startActivity(i);
    }
 });

//Show Map button
  b3 = (Button) findViewById(R.id.btn_showMap);
   b3.setOnClickListener(new OnClickListener()
   {
    public void onClick(View arg0)
  {
      Intent i = new
     Intent(android.content.Intent.ACTION_VIEW,
    Uri.parse("geo:37.827500,-122.481670"));
    startActivity(i);
```

```
    }
 });

//Choose Contact button

   b4 = (Button) findViewById(R.id.btn_chooseContact);
   b4.setOnClickListener(new OnClickListener()
  {
      public void onClick(View arg0){
     Intent i = new
     Intent(android.content.Intent.ACTION_PICK);
     i.setType(ContactsContract.Contacts.CONTENT_TYPE);
     startActivityForResult(i,request_Code);
    }
 });
 }
 public void onActivityResult(int requestCode, int resultCode, Intent data)
 {
    if (requestCode == request_Code)
    {
    if (resultCode == RESULT_OK)
   {
     Toast.makeText(this,data.getData().toString(),
    Toast.LENGTH_SHORT).show();
    Intent i = new Intent(
    android.content.Intent.ACTION_VIEW,
    Uri.parse(data.getData().toString()));
    startActivity(i);
    }
  }
 }
}
```

Output:-



4.2 Notifications Service

- **Services:** are part of the application and run on a different thread in the background and supports some long-running operation, such as, handling location updates from the Location Manager. services operate outside of the user interface.
- **Notification** allows apps or services associated with an app to inform the user of an event, Notification on Android can be done in any of the following ways:
  1. Status Bar Notification
  2. Vibrate
  3. Flash lights
  4. Play a sound

4.3 Broadcast

- A *broadcast receiver* is an Android component which allows you to register for system or application events. All registered receivers for an event are notified by the Android runtime once this event happens / triggers.
- For example, applications can register for the ACTION_BOOT_COMPLETED system event which is fired once the Android system has completed the boot process.
- When any of those events occur it brings the application into action by either creating a status bar notification or performing a particular task.

Following are the some of the important system wide generated intents.

android.intent.action.BATTERY_LOW:	Indicates low battery condition on the device.
android.intent.action.BOOT_COMPLETED:	This is broadcast once, after the system has finished booting
android.intent.action.CALL:	To perform a call to someone specified by the data
android.intent.action.DATE_CHANGED :	The date has changed
android.intent.action.REBOOT :	Have the device reboot
android.net.conn.CONNECTIVITY_CHANGE:	The mobile network or wifi connection is changed(or reset)

To set up a Broadcast Receiver in android application we need to do the following two things.

1. Creating a BroadcastReceiver
2. Registering a BroadcastReceiver

1. Creating a BroadcastReceiver

```
publicclassMyReceiverextendsBroadcastReceiver{
publicMyReceiver(){
   }
@Override
publicvoidonReceive(Context context, Intent intent){
      Toast.makeText(context, "Action: " + intent.getAction(),
Toast.LENGTH_SHORT).show();
   }
}
```

- BroadcastReceiver is an abstract class with the onReceiver() method being abstract.
- The `onReceiver()` method is first called on the registered Broadcast Receivers when any event occurs.
- The intent object is passed with all the additional data. A Context object is also available and is used to start an activity or service using

  context.startActivity(myIntent);

orcontext.startService(myService); respectively.

2. Registering the BroadcastReceiver in android app

- A Broadcast Receiver can be registered in two ways.

By defining it in the AndroidManifest.xml file as shown below.

```
<receiver android:name=".ConnectionReceiver" >
<intent-filter>
<action android:name="android.net.conn.CONNECTIVITY_CHANGE" />
</intent-filter>
</receiver>
```

Using intent filters we tell the system any intent that matches our sub elements should get delivered to that specific broadcast receiver.

By defining it programmatically

Following code snippet shows a sample example to register broadcast receiver programmatically.

```
IntentFilter filter = new IntentFilter();
intentFilter.addAction(getPackageName() +
"android.net.conn.CONNECTIVITY_CHANGE");

MyReceiver myReceiver = new MyReceiver();
registerReceiver(myReceiver, filter);
```

To unregister a broadcast receiver in onStop() or onPause() of the activity the following snippet can be used.

```
@Override
protected void onPause() {
   unregisterReceiver(myReceiver);
   super.onPause();
}
```

Sending Broadcast intents from the Activity

The following snippet is used to send an intent to all the related BroadcastReceivers.

```
Intent intent = new Intent();
     intent.setAction("com.demo.CUSTOM_INTENT");
     sendBroadcast(intent);
```

4.4 Adapter Resources

- Adapter is a bridge between an AdapterView and the underlying data for that view.
- It is a View object. This means, you can add it to your activities the same way you add any other user interface widget. However, it is incapable of displaying any data on its own.
- Its contents are always determined by another object, an adapter.

4.4.1 What is Adapter?

An adapter is an object of a class that implements the Adapter interface. It acts as a link between a data set and an adapter view, an object of a class that extends the abstract AdapterView class. The data set can be anything that presents data in a structured manner like Arrays, List objects, and Cursor objects.

An adapter is responsible for retrieving data from the data set and for generating View objects based on that data. The generated View objects are then used to populate any adapter view that is bound to the adapter.

You can create your own adapter classes from scratch, but most developers choose to use or extend adapter classes provided by the Android SDK, such as Array Adapter and SimpleCursorAdapter.

4.4.2 How Do Adapter Views Work?

Adapter views can display large data sets very efficiently. For instance, the ListView and GridView widgets can display millions of items without any noticeable lag while keeping memory and CPU usage very low.

How do they do that?

Different adapter views follow different ways. However, here's what most of them usually do.

They render only those View objects that are either already on-screen or that are about to move on-screen. This way, the memory consumed by an adapter view can be constant and independent of the size of the data set.

They also allow developers to reduce expensive layout inflate operations and recycle existing View objects that have move off-screen. This keeps CPU consumption low.

4.4.3 Array Adapter Example.

Lets create an ArrayAdapter class.

To create an adapter three things are required

1. Data Set
2. A resource file containing the layout of the generated View objects
3. A TextView Widget

Step 1: Create the Data set

Create an Array as Data Set

The ArrayAdapter class can use both arrays and List objects as data sets.

```
String[ ] Country = {
        "Afghanistan",
```

```
    "Bangladesh",
    "Chaina",
    "Ethiopia",
    "India"
  };
```

Step 2: Create the Resource File

Create a new layout XML file whose root element is a LinearLayout(vertical) and name it country_list.xml. Drag and drop a Large text widget in it and set the value of its id attribute to country_name. The layout XML file should look like this:

```
<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:padding="@dimen/activity_horizontal_margin">
   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textAppearance="?android:attr/textAppearanceLarge"
      android:text="Large Text"
      android:id="@+id/country_name "/>
</LinearLayout>
```

Step 3: Create the Adapter

In your activity, create a new instance of the ArrayAdapter class using its constructor. As its arguments, pass the name of the resource file, the identifier of the TextView, and a reference to the array. The adapter is now ready.

```
ArrayAdapter<String>countryAdapter =
   newArrayAdapter<String>(this,
      R.layout.country_list,
      R.id.country_name,
      Country
   );
```

Step 4: Creating a List

To display a vertically scrollable list of items, you can use the ListView widget. To add the widget to your activity, you can either drag and drop it inside the activity's layout XML file or create it using its constructor in your Java code

```
ListView countrylist = new ListView(this);
```

Usually, no other user interface widgets are placed inside a layout that contains a ListView. Therefore, pass the ListView to the setContentView() method of your activity so that it takes up the entire screen.

setContentView(countrylist);
To bind the ListView to the adapter we created in the previous step, call the setAdapter() method as shown below.
countrylist.setAdapter(countryAdapter);
then run your app, you should be able to see the contents of the array in the form of a list.

Summary:

After studying this chapter students will be able to :

- Understand Activities and Activity Life Cycle
- Understand Context
- Understand Intent and to classify types of intent.
- Use Notification service
- Use Adapter recourse

Exercise:

1 List out the Terminologies of Android in Details
2 Write a short note on Android Context.
3 Write a short note on Activity.
4 Draw and Explain Activity Life Cycle.
5 Explain Intent in Android.
6 Explain Implicit and Explicit Intent in Android
7 Demonstrate the Linking of Two Activities in Android using Intent.

UNIT 5

Android User Interface Elements

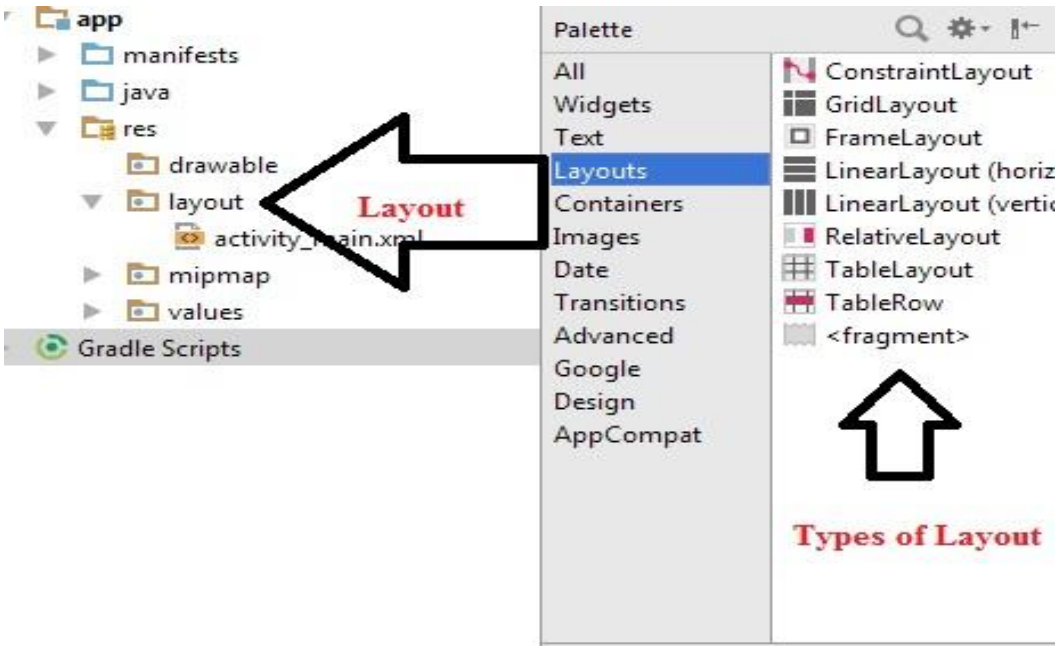
Learning Objectives:

After going through this unit , you will be able to learn:

- What is Layout?
- Types of Layout.
- Creation of layout.
- Views and types of views
- Implementation of Layout and Views.

5.1 Layouts

- An Android layout is a class that is responsible for arranging the way its children appear on the screen.
- Anything that is a View or which is inherit from view can be a child of a layout. All of the layouts in android are inherited from **ViewGroup** (which inherits from **View**)
- Android allows you to create view layouts using simple XML file (user can also create a layout using java code).



5.1.1 Linear Layout

- It is the simplest layout used in android for layout designing.
- In the Linear layoutdisplays all the elements 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.

Vertical Layout	**Horizontal Layout**
	
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android" **xmlns:app="http://schemas.android.com/apk/res-auto"** **xmlns:tools="http://schemas.android.com/tools"** **android:layout_width="match_parent"** **android:layout_height="match_parent"** **android:orientation="vertical"** **android:background="#662233"** **tools:context="com.example.snjbetc.demo2.MainActivity">** **<Button** **android:id="@+id/button"** **android:layout_width="match_parent"** **android:background="#006622"** **android:layout_height="wrap_content"** **android:layout_marginBottom="5dp"** **android:text="Button 1" />** **<Button** **android:id="@+id/button2"**	**<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"** **xmlns:app="http://schemas.android.com/apk/res-auto"** **xmlns:tools="http://schemas.android.com/tools"** **android:layout_width="match_parent"** **android:layout_height="match_parent"** **android:orientation="horizontal"** **android:background="#662233"** **tools:context="com.example.snjbetc.demo2.MainActivity">** **<Button** **android:id="@+id/button"** **android:layout_width="match_parent"** **android:background="#006622"** **android:layout_height="wrap_content"** **android:layout_marginBottom="5dp"** **android:text="Button 1" />** **<Button** **android:id="@+id/button2"**

android:layout_width="match_parent" **android:layout_height="wrap_content"** **android:background="#006622"** **android:text="Button 2"** /> </**LinearLayout**>	**android:layout_width="match_parent"** **android:layout_height="wrap_content"** **android:background="#006622"** **android:text="Button 2"** /> </**LinearLayout**>

Linear Layout Attributes: Following are some of the XML attributes associated with Linear Layout

Table 5.1 XML Attributes associated with Linear Layout

Sr.No	Name	Description
1	**android:orientation**	The orientation attribute used to set the childs/views horizontally or vertically. In Linear layout default orientation is vertical.
2	**android:gravity**	The gravity attribute is an optional attribute which is used to control the alignment of the layout like left, right, center, top, bottom etc.
3	**android:layout_weight**	The layout weight attribute specify each child control's relative importance within the parent linear layout.
4	**android:weightSum**	weightSum is the sum up of all the child attributes weight. This attribute is required if we define weight property of the childs.

5.1.2Absolute Layout

- An Absolute Layout is a layout used to design the custom layouts.
- In this layout the exact location of its children by using x and y coordinates can be specify.

Absolute Layout GUI	Absolute Layout XML
	<?xml version="1.0" encoding="utf-8"?> <AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android" android:layout_width="fill_parent" android:layout_height="fill_parent"> <Button android:id="@+id/button" android:layout_width="wrap_content" android:layout_height="wrap_content" android:layout_x="200px" android:text="X Coordinate" /> <Button android:id="@+id/button2" android:layout_width="wrap_content" android:layout_height="wrap_content" android:layout_y="800px" android:text="Y Coordinate" /> <TextView android:id="@+id/textView2" android:layout_width="wrap_content" android:layout_height="wrap_content" android:layout_x="195dp" android:layout_y="25dp" android:text="X Coordinate" android:textSize="20dp" /> <TextView android:id="@+id/textView3" android:layout_width="wrap_content" android:layout_height="wrap_content" android:layout_x="41dp" android:layout_y="300dp" android:textSize="20dp" android:text="Y Coordinate" /> <TextView android:id="@+id/textView4" android:layout_width="wrap_content" android:layout_height="wrap_content" android:layout_x="15dp"

	android:layout_y="364dp" android:text="Absolute Layout Demo" android:textSize="35dp" android:textStyle="bold" /> </AbsoluteLayout>

Important Note 1: Absolute layout are harder to preserve for different mobile screen sizes than other types of layouts because we set the exact location of a child view or called component. The positioning is based on x(top) and y(left) coordinates and that positioning is not as useful in world of various screen resolutions(sizes) and aspect ratios.

Important Note 2: Absolute layout is depreciated in Android because of the same reason as discussed in above note.

Absolute Layout Attributes: Following are some of the XML attributes associated with absolute Layout

Table 5.2 XML Attributes associated with absoluteLayout

Sr.No	Name	Description
1	**android:layout_x:**	In Absolute Layout layout_x attribute is used to specify the x- coordinate of the view(TextView or any other view). The possible value for this is in dp or px.
2	**android:layout_y**	In AbsoluteLayout layout_y attribute is used to specify the y- coordinate of the view(TextView or any other view). The possible value for this is in dp or px.

5.1.3Frame Layout

- FrameLayout is a layout whis is designed to block out an area on the screen to display a single item.
- The frame layout is often used as a container layout, as it generally only has a single child view (often another layout, used to organize more than one view).
- Multiple children to a FrameLayout can be added and control their position by assigning gravity to each child, using the **android:layout_gravity** attribute.

Frame Layout GUI	**Frame Layout XML**
	<?xml version="1.0" encoding="utf-8"?> <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android" android:layout_width="fill_parent" android:layout_height="fill_parent" android:id="@+id/framelayout" > <ImageView android:id="@+id/frameImage" android:layout_width="200dp" android:layout_height="300dp" android:src="@drawable/a" android:layout_gravity="center" android:clickable="true" /> <TextView android:id="@+id/frameText" android:layout_width="wrap_content" android:layout_height="wrap_content" android:text="India" android:textSize="50sp" android:textStyle="bold" android:visibility="visible" android:layout_gravity="bottom" /> </FrameLayout>

Frame Layout Attributes: Following are some of the XML attributes associated with Frame Layout

Table 5.3 XML Attributes associated with FrameLayout

Sr.No	Name	Description
1	**android:id**	It is the unique id which identifies the layout in the R.java file.
2	**android:foreground**	it defines the drawable to draw over the content and this may be a color value. The Possible color values can be in the form of "#rgb", "#argb", "#rrggbb", or "#aarrggbb". This all are different color code model used.
3	**android:foregroundGravity**	This defines the gravity to apply to the foreground drawable. Default value of gravity is fill. The values can be set in the form of "top", "center_vertical" , "fill_vertical", "center_horizontal", "fill_horizontal", "center", "fill", "clip_vertical", "clip_horizontal", "bottom", "left" or "right" . Which is used to set the gravity of foreground. We can also set multiple values by using "|". Ex: fill_horizontal|top .Both the fill_horizontal and top gravity are set to framelayout.
4	**android:visibility**	This determine whether to make the view visible, invisible or gone. **visible –** the view is present and also visible **invisible –** The view is present but not visible **gone –** The view is neither present nor visible
5	**android:measureAllChildren**	This determines whether to measure all children including gone state visibility or just those which are in the visible or invisible state of measuring visibility. The default value of measureallchildren is false. We can set values in the form of Boolean i.e. "true" OR "false". This may also be a reference to a resource (in the form "@[*package*:]*type*:*name*") or theme attribute (in the form "?[*package*:][*type*:]*name*") containing a value of this type.

5.1.4Relative Layout

- RelativeLayout is layout which let you position your component base on the nearby (relative or sibling) component's position.
- It's the most flexible layout, that allow you to position your component to display in anywhere you want (if you know how to "relative" it).
- In Relative Layout, one can use "above, below, left and right" to arrange the component's position in relation to other component.

Relative Layout GUI	**Relative Layout XML**
	<?xml version="1.0" encoding="utf-8"?> <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android" android:layout_width="fill_parent" android:layout_height="fill_parent" android:id="@+id/relative" > <TextView android:id="@+id/frameText" android:layout_width="wrap_content" android:layout_height="wrap_content" android:text="Android" android:textSize="50dp" android:textStyle="bold" android:layout_marginTop="63dp" android:layout_alignParentTop="true" android:layout_alignEnd="@+id/textView5" android:layout_marginEnd="92dp" /> <TextView android:id="@+id/textView5" android:layout_width="wrap_content" android:layout_height="wrap_content" android:layout_marginTop="78dp" android:textAlignment="center" android:textSize="20dp" android:text="This is the content which is placed below the Android (Heading)" android:layout_below="@+id/frameText" android:layout_alignParentEnd="true" android:layout_marginEnd="42dp" /> </RelativeLayout>

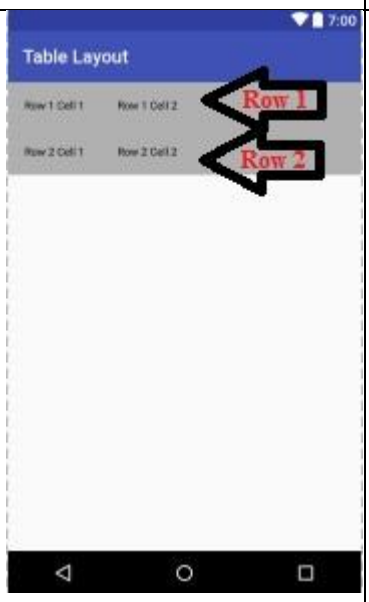
Relative Layout Attributes: Following are some of the XML attributes associated with Relative Layout

Table 5.4 XML Attributes associated with RelativeLayout

Sr. No	Name	Description
1	**android:above**	Position the bottom edge of the view above the given anchor view ID and must be a reference of the another resource in the form of id.
2	**android:alignBottom:**	alignBottom is used to makes the bottom edge of the view match the bottom edge of the given anchor view ID and it must be a reference to another resource, in the form of id.
3	**android:alignLeft:**	alignLeft is used to make the left edge of the view match the left edge of the given anchor view ID and must be a reference to another resource, in the form of Example: android:layout_ alignLeft =”@+id/button1″.
4	**android:alignRight:**	alignRight property is used to make the right edge of this view match the right edge of the given anchor view ID and must be a reference to another resource, in the form like this example: android:layout_alignRight=”@+id/button1″
5	**android:alignStart**	alignStart property is used to makes the start edge of this view match the start edge of the given anchor view ID and must be a reference to another resource, in the form of like this example: android:layout_alignStart=”@+id/button1″

5.1.5 Table Layout

- Table Layout is a layout which is used to arrange the group of views into rows and columns.
- The containers of table layout do not display a border line for their columns, rows or cells.
- A Table will have many columns and the row with the most cells.
- A table can also leave the cells empty but cells can't span the columns as they can in HTML(Hypertext markup language).

Table Layout GUI	Table Layout XML Code
	<?xml version="1.0" encoding="utf-8"?> <TableLayout xmlns:android="http://schemas.android.com/apk/res/android" android:id="@+id/simpleTableLayout" android:layout_width="match_parent" android:layout_height="match_parent" android:stretchColumns="1"> <!-- stretch the second column of the layout--> <!-- first row of the table layout--> <TableRow android:id="@+id/firstRow" android:layout_width="fill_parent" android:layout_height="wrap_content"> <!-- first element of the row--> <TextView android:id="@+id/st" android:layout_width="wrap_content" android:layout_height="wrap_content" android:background="#b0b0b0" android:padding="18dip" android:text="Row 1 Cell 1" android:textColor="#000" android:textSize="12dp" /> <!-- first element of the row--> <TextView android:id="@+id/st2" android:layout_width="wrap_content" android:layout_height="wrap_content" android:background="#b0b0b0" android:padding="18dip"

```
android:text="Row 1 Cell 2"
android:textColor="#000"
android:textSize="12dp" />


</TableRow>
<!-- first row of the table layout-->
<TableRow
android:id="@+id/secondrow"
android:layout_width="fill_parent"
android:layout_height="wrap_content">

<!-- first element of the row-->
<TextView
android:id="@+id/st21"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:background="#b0b0b0"
android:padding="18dip"
android:text="Row 2 Cell 1"
android:textColor="#000"
android:textSize="12dp" />

<!-- first element of the row-->
<TextView
android:id="@+id/st22"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:background="#b0b0b0"
android:padding="18dip"
android:text="Row 2 Cell 2"
android:textColor="#000"
android:textSize="12dp" />


</TableRow>
</TableLayout>
```

Table Layout Attributes: Following are some of the XML attributes associated with Table Layout

Table 5.5 XML Attributes associated with TableLayout

Sr. No	Name	Description
1	**android:stretchColumns**	Stretch column attribute is used in Table Layout to change the default width of a column which is set equal to the width of the widest column. The columns can be stretch to take up available free space by using this attribute. The value which is assigned to this attribute can be a single column number or a comma delimited list of column numbers (1, 2, 3…n).
2	**android:shrinkColumns**	Shrink column attribute is used to shrink or reduce the width of the column's. We can specify either a single column or a comma delimited list of column numbers for this attribute. The content in the specified columns word-wraps to reduce their width. (Values are 0,1,*)
3	**android:collapseColumns**	collapse columns attribute is used to collapse or invisible the column's of a table layout. These columns are the part of the table information and they are invisible.If the values is 0 then the first column appears collapsed, i.e it is the part of table but it is invisible.

5.2 Creation of Layout Programmatically

- In some cases, you have to create and style a LinearLayout,RelativeLayout or any other layout programmatically.
- Attributes that you can programmatically apply to the layout can be background color, layout width, height, margins, orientation, layout gravity, paddings, and so son.

You need to use Linear Layout class To create a LinearLayout, You pass the current activity object to its constructor.

```
LinearLayout layout = new LinearLayout(MainActivity.this);
```

For setting background color of a layout, setBackGroundColor(int color) method is used. Here integer color value represents a color to the method. parseColor(String color) method of Color class is used to convert a string hex color to its integer equivalence.

```
layout.setBackgroundColor(Color.parseColor("#135517"));
```

To set width and height of the layout, you have to create a LayoutParams object and set it to the layout using setLayoutParams(LayoutParam params) method. With the LayoutParams object, you also can set the left, top, right, and bottom margins of the layout.

```
LinearLayout.LayoutParams params = new LinearLayout.LayoutParams
            (LinearLayout.LayoutParams.MATCH_PARENT,
LinearLayout.LayoutParams.WRAP_CONTENT);

params.setMargins(15, 5, 5, 5);

layout.setLayoutParams(params);
```

The orientation, layout gravity, paddings of the layout can be set using the setOrientation(int orientation) and setHorizontalGravity(int horizontalGravity) or setVerticalGravity(int verticalGravity), and setPaddings(int left, int top, int right, int bottom) methods.

```
layout.setOrientation(LinearLayout.HORIZONTAL);

layout.setHorizontalGravity(Gravity.CENTER_HORIZONTAL);

layout.setPadding(10, 10, 5, 5);
```

5.3 View

- The View is a base class (Super Class) for all UI components in android.
- For example, the **EditText** class is used to accept the input from users in android apps, which is a sub class of View.

Following are the some of common `View` subclasses which will be used in android applications.

- TextView
- EditText
- Button
- CheckBox
- ListView
- RecyclerView
- WebView
- ImageButton
- Progress Bar
- Spinner
- ImageButton
- GridView
- ScrollView

Like this there are so many `View` subclasses available in android.

5.3.1 Android ViewGroup

The `ViewGroup` is a subclass of `View` and it acts as a base class for **layouts** and **layouts parameters**. An invisible container are provided by `ViewGroup` to hold other **Views** or **ViewGroups** and to define the layout properties.

Let us Consider an example, Linear Layout is the `ViewGroup` that contains a UI controls like button, textview, etc. and other layouts also.

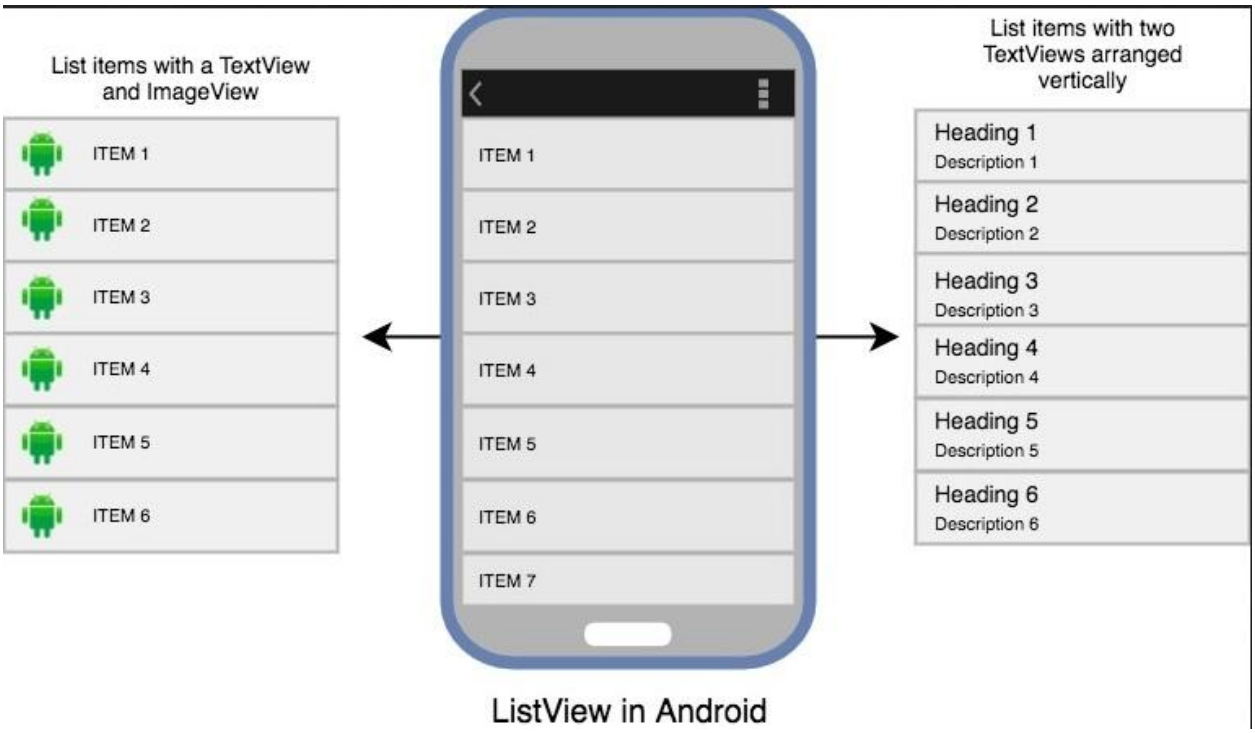
Following are the commonly used `ViewGroup` subclasses in android applications.

- Linear Layout
- Relative Layout
- Table Layout
- Frame Layout
- Web View
- List View
- Grid View

Both `View` and `View Group` subclasses together will play a key role to create layouts in android applications.

5.3.2ListView

- Android **ListView** is a view which contains the group of items and displays in a scrollable list.
- ListView is implemented by importing *android.widget.ListView* class. It uses Adapter classes which add the content from data source (such as string array, array, database etc) to ListView.
- Adapter links data between an *AdapterViews* and other Views (ListView, ScrollView etc).



ListView in Android

Below we have shown how you can add a ListView to your android app using the layout XML.

```
<ListView
android:id="@+id/listView"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:divider="@android:color/black"
android:dividerHeight="1dp"/>
```

Using Adapter with ListView

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
```

```
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

publicclassMainActivityextendsAppCompatActivity{
ListView listView;
TextView textView;
String[] festivals ={
"Diwali",
"Holi",
"Christmas",
"Eid",
"Baisakhi",
"Halloween"
};

@Override
protectedvoidonCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
listView =(ListView)findViewById(R.id.listView);
textView =(TextView)findViewById(R.id.textView);
final ArrayAdapter adapter =newArrayAdapter(this,
R.layout.list_item, android.R.id.textView, festivals);
listView.setAdapter(adapter);
listView.setOnItemClickListener(newAdapterView.OnItemClickListener(){
@Override
publicvoidonItemClick(AdapterView<?> adapterView, View view,int position,long l){
// TODO Auto-generated method stub
```

```
/* appending Happy with festival name */
String value ="Happy "+ adapter.getItem(position);
/* Display the Toast */
Toast.makeText(getApplicationContext(), value, Toast.LENGTH_SHORT).show();
}
});}}
```

5.3.3GridView

- GridView is a view group that display items in two dimensional scrolling grid (rows and columns), the grid items are automatically inserted to the layout using a ListAdapter.
- Users select any grid item by clicking on it. This view is by default scrollable so we don't need to use ScrollView or anything else with GridView.
- GridView is widely used in android applications. An example of GridView is your default Gallery, where you have number of images displayed using grid.
- **Adapter Is Used To Fill Data In Gridview:** To fill the data in a GridView ,adapter can be used and grid items are automatically inserted to a GridView using an Adapter which pulls the content from a source such as an arraylist, array or database.

Grid View UI	Basic GridView code in XML:
	<GridView android:id="@+id/simpleGridView" android:layout_width="fill_parent" android:layout_height="wrap_content" android:numColumns="3"/>

Grid View Attributes: Following are some of the XML attributes associated with GridView

Table 5.6 XML Attributes associated with TableLayout

Sr.No	Name	Description
1	**android:numColumns:**	numColumn define how many columns to show. It may be a integer value, such as "5" or auto_fit.**auto_fit** is used to display as many columns as possible to fill the available space on the screen.
2	**android:horizontalSpacing:**	horizontalSpacing property is used to define the default horizontal spacing between columns. This could be in pixel(px),density pixel(dp) or scale independent pixel(sp).
3	**android:verticalSpacing:**	verticalSpacing property used to define the default vertical spacing between rows. This should be in px, dp or sp..
4	**android :columnWidth:**	columnWidth property specifies the fixed width of each column. This could be in px, dp or sp.

GridView Example Using Different Adapters in Android Studio:

An adapter act as a bridge between UI component and data source that helps us to fill data in UI component. It holds the data which is then sends to adapter view, then view can takes the data from the adapter view and shows the data on different views like as list view, grid view, spinner etc.

GridView and ListView both are subclasses of AdapterView and which then can be populated by binding to an Adapter, which retrieves the data from an external source and creates a View that represents each data entry.

In android following adapters can be used to fill data in GridView :

1. Array Adapter
2. Base Adapter
3. Custom Array Adapter

Now we explain these adapters in detail:

1. Avoid Array Adapter To Fill Data In GridView:

Whenever you have a list of single items which is then backed by an array, you can use ArrayAdapter. For example, list of phone contacts, countries or names.

ArrayAdapter expects a Layout with a single TextView bydefault, If you want to use more complex views means more customization in grid items, please avoid ArrayAdapter and use custom adapters.

```
ArrayAdapter adapter
=newArrayAdapter<String>(this,R.layout.ListView,R.id.textView,StringArray);
```

2. GridView Using Base Adapter In Android:

Base Adapter is a common base class or super class of a general implementation of an Adapter that can be used in GridView.

Whenever you need a customized grid view you can create your own adapter which can be then extend base adapter in that.

Base Adapter can be extended to create a custom Adapter for displaying custom grid items. ArrayAdapter is also an implementation of BaseAdapter.

5.3.4RecyclerView

- It is advanced and flexible version of ListView and GridView.
- It is a container used for displaying large amount of data sets that can be scrolled very efficiently by maintaining a limited number of views.
- RecyclerView was introduced in Material Design in API level 21 (Android 5.0 i.e Lollipop).

Need of RecyclerView In Android?

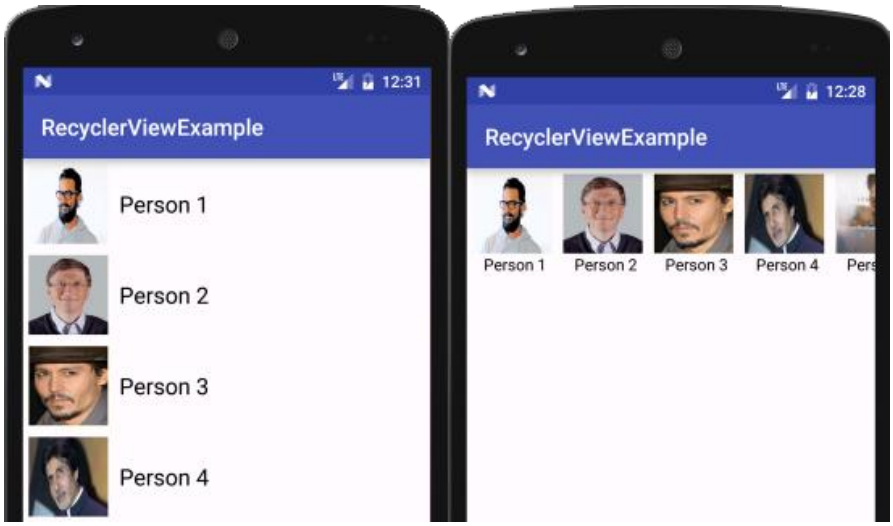
For storing the reference of the view for one entry in the RecyclerView it uses a ViewHolder. At the time of using ListView or GridView for displaying custom items then we need to create a custom xml file and then use it inside our Adapter.

We create a CustomAdapter class and then need toextends our Base or any other Adapter in it. In getView() method of our Adapter we inflate the item layout xml file and then give the reference of every view by using the unique id's we provide in our xml file .

Once it is done we pass that view to the ListView, ready to be drawn, but the truth is that ListView and GridView do only half the job of achieving true memory efficiency.

ListView/GridView recycle the item layout but don't keep the reference to the layout children, forcing us to call findViewById() for every child of our item layout for every time we call getView(). Such issues causes the scrolling or non-responsive problem as it frantically tries to grab references to the view's we needed.

With the arrival of RecyclerView everything is changed. RecyclerView uses Adapter to act as Data source but in this we need to create a ViewHolder to keep the reference of View in memory, so when we need a new view it either creates a new ViewHolder object to inflate the layout and hold those references or it recycles one from existing stack.



It is used for displaying the data items in different scrolling list such as a horizontal or vertical scrolling List. If we need a list(vertical or horizontal) then we need to use LinearLayoutManager with

require orientation. In other words we can say that we use the LinearLayoutManager for displaying RecyclerView as a ListView.

Components of a RecyclerView

LayoutManagers	A `RecyclerView`requiredto have a layout manager and an adapter to be instantiated. A layout manager positions item views inside a `RecyclerView` and which also determines when to reuse item views that are no longer visible to the user. RecyclerView provides these built-in layout managers: LinearLayoutManager shows all items in ahorizontal or vertical scrolling list. GridLayoutManager shows items in a grid. StaggeredGridLayoutManager shows items in a staggered grid. Extend the RecyclerView.LayoutManager class to create a custom layout manager,
RecyclerView.Adapter	RecyclerView includes a new kind of adapter. It's a similar approach to the ones you already used, but with some peculiarities, such as a required ViewHolder. You need to override two main methods: one to inflate the view and its view holder, and another one to bind data to the view. The good thing about this is that first method is called only when we really need to create a new view. No need to check if it's being recycled.
ItemAnimator	RecyclerView.ItemAnimator will animate ViewGroup modifications like as add/delete/select which are then notified to adapter. DefaultItemAnimator can be used for basic default animations and works quite well.

Using the RecyclerView

Using a `RecyclerView` has the following key steps:

1. Add RecyclerView support library to the gradle build file
2. To use as the data source Define a model class
3. To display the items Add a RecyclerView to your activity
4. To visualize the item Create a custom row layout XML file
5. To render the item Create a RecyclerView.Adapter and ViewHolder
6. To populate the RecyclerView Bind the adapter to the data source

Creating the RecyleView

Step 1. Update build.gradle file

Before you can use RecyclerView in your projects you need to add the following compile line to your Gradle dependencies block in your build.gradle file and rebuilt the project .

```
dependencies {
    ...
    compile 'com.android.support:appcompat-v7:23.1.1'
    compile 'com.android.support:design:23.1.1'
    compile 'com.android.support:recyclerview-v7:23.1.1'
}
```

Step 2. Modify activity_main.xml

```
<?xmlversion="1.0"encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context=".MainActivity">

    <android.support.v7.widget.RecyclerView
        android:id="@+id/recycler_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:scrollbars="vertical" />

</RelativeLayout>
```

Step 3. Write Book model class

Create a class **Book.java** and declare the variables title and author. Also add the getter/setter methods to each variable.

```
public class Book {
```

```
    private String title;
    private String author;
    public Book(String title, String author) {
        this.title = title;
        this.author = author;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }
}
```

Step 4. List Row layout

Create an layout xml named **book_list_row.xml** with the below code. This layout file renders a single row in recycler view by displaying book title and author .

```
<?xmlversion="1.0"encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:focusable="true"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp"
    android:clickable="true"
    android:background="?android:attr/selectableItemBackground"
    android:orientation="vertical">
```

```
    <TextView
        android:id="@+id/title"
        android:textColor="@android:color/black"
        android:textSize="16dp"
        android:textStyle="bold"
        android:layout_alignParentTop="true"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
    <TextView
        android:id="@+id/author"
        android:layout_below="@id/title"
        android:layout_width="match_parent"
        android:textColor="@android:color/black"
        android:layout_height="wrap_content"/>
</RelativeLayout>
```

Step 5. Writing the Adapter Class

Create a class named **BookAdapter.java** and add the below code. Here onCreateViewHolder() method inflates book_list_row.xml. In onBindViewHolder() method the appropriate book data (title and author) set to each row.

```
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import com.androidtutorialshub.recyclerviewtutorial.Model.Book;
import com.androidtutorialshub.recyclerviewtutorial.R;
import java.util.List;
public class BookAdapter extends RecyclerView.Adapter<BookAdapter.BookViewHolder>{
   private List<Book> bookList;
   public BookAdapter(List<Book> bookList) {
      this.bookList = bookList;
   }
```

```
    @Override
    public BookViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View itemView = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.book_list_row, parent, false);
        return new BookViewHolder(itemView);
    }

    @Override
    public void onBindViewHolder(BookViewHolder holder, int position) {
            holder.title.setText(bookList.get(position).getTitle());
            holder.author.setText(bookList.get(position).getAuthor());
    }
    @Override
    public int getItemCount() {
        return bookList.size();
    }
    public class BookViewHolder extends RecyclerView.ViewHolder {
        public TextView title;
        public TextView author;
        public BookViewHolder(View view) {
            super(view);
            title = (TextView) view.findViewById(R.id.title);
            author = (TextView) view.findViewById(R.id.author);
        }
    }
}
```

Step 6. Binding Adapter and RecyclerView

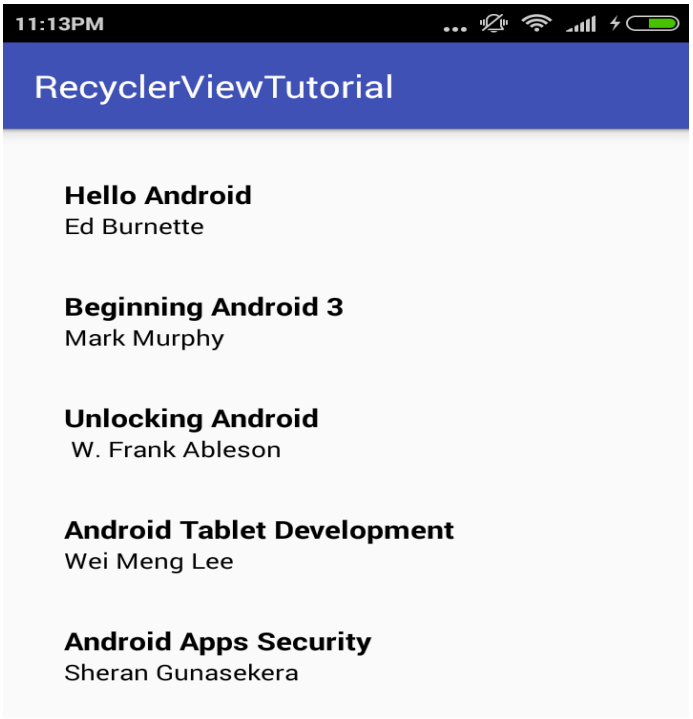
Open **MainActivity.java** and update the below changes. Here initBookData() method sets the sample data to recycler view.

```
 import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.DefaultItemAnimator;
```

```
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import com.androidtutorialshub.recyclerviewtutorial.Adapter.BookAdapter;
import com.androidtutorialshub.recyclerviewtutorial.Model.Book;

import java.util.ArrayList;
import java.util.List;
public class MainActivity extends AppCompatActivity {
    private List<Book> bookList = new ArrayList<>();
   private RecyclerView recyclerView;
   private BookAdapter mAdapter;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      recyclerView = (RecyclerView) findViewById(R.id.recycler_view);
      mAdapter = new BookAdapter(bookList);
      RecyclerView.LayoutManager mLayoutManager = new
LinearLayoutManager(getApplicationContext());
      recyclerView.setLayoutManager(mLayoutManager);
      recyclerView.setItemAnimator(new DefaultItemAnimator());
      recyclerView.setAdapter(mAdapter);
       initBookData();
   }
    private void initBookData() {
      Book book = new Book("Hello Android", "Ed Burnette");
      bookList.add(book);
       book = new Book("Beginning Android 3", "Mark Murphy");
      bookList.add(book);
       book = new Book("Unlocking Android", " W. Frank Ableson");
      bookList.add(book);
      book = new Book("Android Tablet Development", "Wei Meng Lee");
```

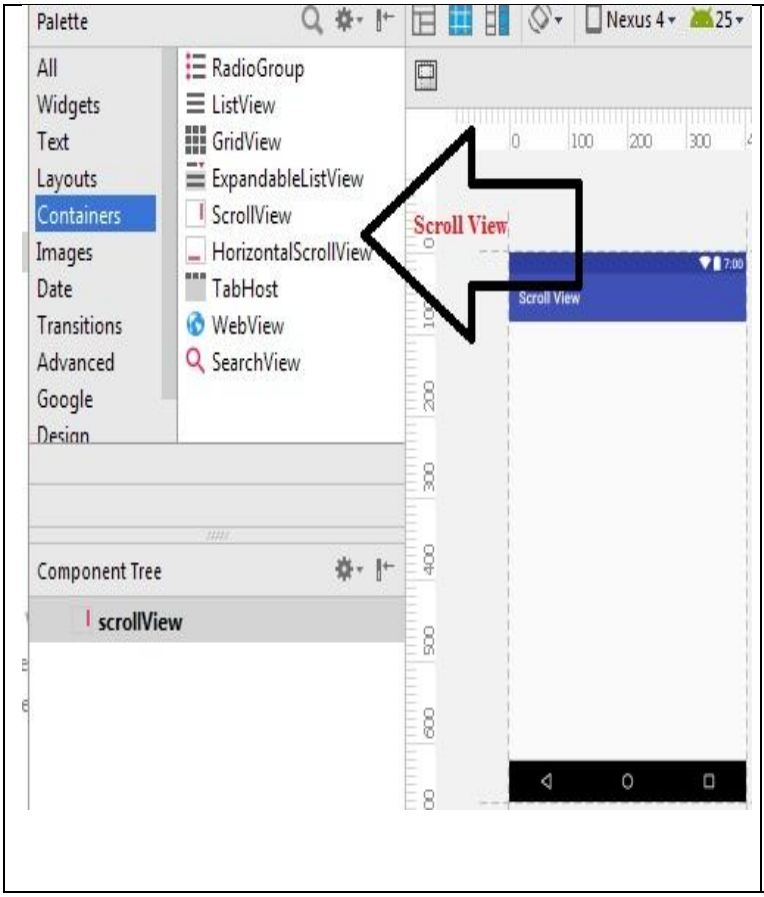
```
        bookList.add(book);
        book = new Book("Android Apps Security", "Sheran Gunasekera");
        bookList.add(book);
         mAdapter.notifyDataSetChanged();
    }
}
```

Output: -



5.3.5ScrollView

- ScrollView is used When an app has layout content that might be longer than the height of the device and that content should be vertically scrollable.
- You can specify layout_width and layout_height to adjust width and height of screen. One can specify height and width in dp(density pixel) or px(pixel). Then after enclosing them in a standard layout, enclose the whole layout in ScrollViewwhich will make all the element or views scrollable.
- It is present inside Containers




```
<?xml version="1.0" encoding="utf-8"?>

<ScrollView
android:id="@+id/scrollView"
android:layout_width="fill_parent
"
android:layout_height="fill_parent
"
xmlns:android="http://schemas.an
droid.com/apk/res/android">


<!-- add child view's here -->


</ScrollView>
```

5.3.6 Horizontal ScrollView:

- In android, one can scroll the elements or views in both vertical and horizontal directions.
- To scroll in Vertical, we simply use ScrollView as we shown in the above diagram and to scroll in horizontal direction, we need to use HorizontalScrollview.

```
<HorizontalScrollView
android:id="@+id/horizontalscrollView"
android:layout_width="fill_parent"
android:layout_height="fill_parent">


<-- add child view's here -->
</HorizontalScrollView>
```

Scroll View Attributes

Following are some of the XML attributes associated with Scroll View

Table 5.7 XML Attributes associated with Scroll View

Sr.No	Name	Description
1	**android:scrollbars:**	In android, scrollbars attribute is used to show the scrollbars in horizontal or vertical direction. The possible Value of scrollbars is vertical, horizontal or none. By default, scrollbars are shown in vertical direction in scrollView and in horizontal direction in HorizontalScrollView.

5.3.6WebView

- Android WebView is used to display online content in android activity
- It displays the HTML Pages in Android App.
- Android WebView component is a full-fledged browser implemented as a View subclass to embed it into our android application.

Integrating a WebView in your app: -

XML Code

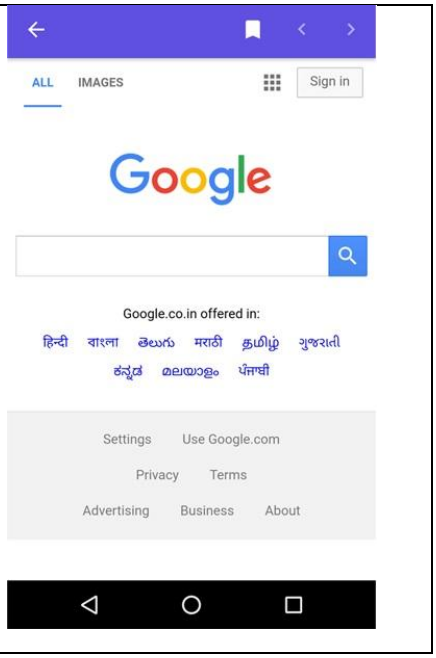
```
<WebView
    android:id="@+id/webView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"/>
```

//To be written in xml

Java Code

Step 2:

```
WebView webView = (WebView) findViewById(R.id.webView);
webView.loadUrl("http://google.com");
```



Note: - The loadUrl() and loadData() methods of Android WebView class are used to load and display web page.

Summary:

After going through this unit, you will be able to:

- Understand Layout
- Types of Layout.
- Create and implement layout.
- Understand Views and types of views
- Implement Layout and Views.

Exercise:

1 Write a short note on Layouts in Android.
2 Explain Linear Layout with its orientation and attributes.
3 Explain Absolute Layout with its Attributes in Details
4 Explain Frame Layout with its Attributes in Details
5 Explain Relative Layout with its Attributes in Details
6 Explain Table Layout with its Attributes in Details
7 Write a short note creating Layouts Programmatically
8 Write a short note on Views in Android.
9 Explain Listview with Example (XML and Java Class)
10 Explain GridView with Example (XML and Java Class)
11 Explain GridView with Example (XML and Java Class)
12 Write a short note on Recyle View.
13 Explain the ScrollView with its Attributes. 14 Explain Web View With its XML and Java Code.

UNIT 6

Data Storage and Introduction to SQLITE

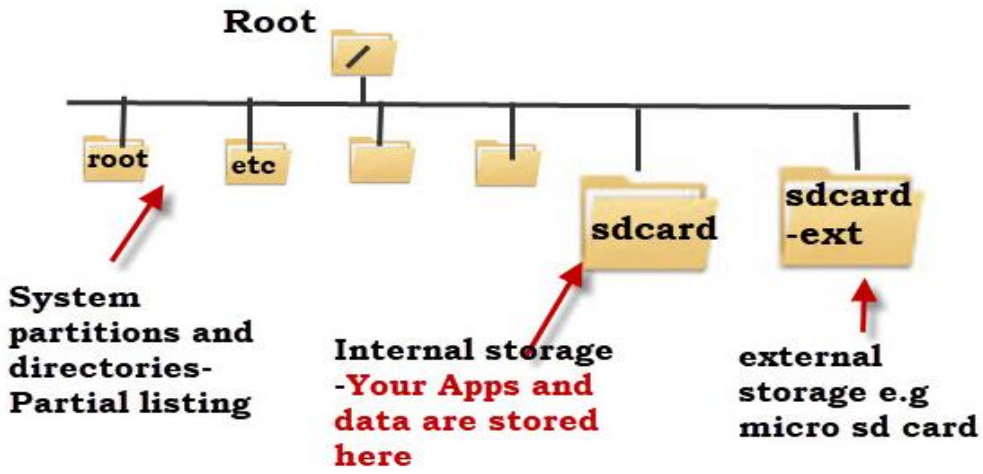
Learning Objectives:

After going through this unit, you will be able to learn:

- File system in android
- Internal and external Storage.
- Creation of SQL Database.
- Editing Task with SQL Lite
- Cursor and content values.
- Working with android database.

6.1 File system in android

- In Linux / Android / Unix, the file hierarchy is a single tree, with the top of the tree being "/" - the root of the tree. Under "/" are files and directories.
- The Linux file hierarchy doesn't support the concept of drives which is of windows.
- Instead of it files systems are mounted on single directory to create a single integrated tree.
- Android Linux file system structure which has a single root



- The system partitions and directories are protected and unless your device is rooted you don't normally have access to these although some file managers will display them.
- Android doesn't normally come with a default file manager, and so you will need to install a file manager App like ES File Explorer, to locate and manage files and folders.

6.2 Internal and external storage

Android provides many kinds of storage for applications to store their data. These storage places are shared preferences, internal and external storage, SQLite storage, and storage via network connection etc.

6.2.1 Internal Storage:

- By Default all the stored files in the internal storage are private and are accessed only by your application and will be deleted only when user delete the application.
- In Order to use internal storage to write some data on the files use the following functions;

openFileOutput():- this method has 2 parameters 1. File name 2. File Mode(Private,Pubic etc)

Syntax:-

```
FileOutputStream fOut = openFileOutput("file_name",MODE_WORLD_READABLE);
Returns an instance of FileOutputStream. So you receive it in the object of FileInputStream
```

6.2.3 Writing on a File

1. write(String):- this method has a parameter string which user wants to write

Syntax:-

```
String str = "data";
fOut.write(str.getBytes());
fOut.close();
```

6.2.4 Reading From a File

In Order to read from a file use the following function

2. FileInputStream fin = openFileInput(file);

After that, you can call read method to read one character at a time from the file and then you can print it.

Syntax:-

```
int c;
String temp="";
while( (c = fin.read()) != -1){
   temp = temp + Character.toString((char)c);
}
//temp contains all the data of the file.
fin.close();
```

Apart from just Reading and Writing on a file following are some other methods which are provided by the FileOutputStream Class & FileInputStreamClass;

6.2.5 FileOutputStream Class

Table 6.1 Method Description of FileOutputStream Class

Sr.No	Method	Description
1	**FileOutputStream(File file, boolean append)**	This method constructs a new FileOutputStream that writes to file.
2	**getChannel()**	This method returns a write-only FileChannel that shares its position with this stream
3	**getFD()**	This method returns the underlying file descriptor
4	**write(byte[] buffer, int byteOffset, int byteCount)**	This method Writes count bytes from the byte array buffer starting at position offset to this stream

6.2.6 FileInputStream Class

Table 6.2 Method Description of FileInputStream Class

Sr. No	Method	Description
1	**available()**	This method returns an estimated number of bytes that can be read or skipped without blocking for more input
2	**getChannel()**	getChannel() method returns a read-only FileChannel that shares its position with this stream
3	**getFD()**	getFD() method returns the underlying file descriptor
4	**read(byte[] buffer, int byteOffset, int byteCount)**	This method reads at most length bytes from this stream and stores them in the byte array b starting at offset

6.3 External Storage

- An android devices supports another type of storage called external storage where apps can save files. It can be either removable like an SD card or non-removable in which case it is internal. Files in this storage are world readable which means other applications have access to them.

- Before writing to this volume we must check that it is available as it can become unavailable if the SD card is removed or mounted to the user's computer.

- Using getExternalStorageState() we can get the current state of the primary external storage device. If it's equal to Environment.MEDIA_MOUNTED then we'll have read/write access and if equal to Environment.MEDIA_MOUNTED_READ_ONLY then we have only read access.

```
Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED)
```

- Although external storage is accessible and modifiable by the user and other apps, user can save the files in two ways – public and private

- In order to Read and Write files to the External Storage we will need the Following two permissions;

i. READ_EXTERNAL_STORAGE

ii. WRITE_EXTERNAL_STORAGE

6.3.1 Public File:

These files remain on the storage even after the application is uninstalled by the user like media (photos, videos, etc.) or other downloaded files. There are 2 methods that we can use to get the public external storage directory for placing files:

getExternalStorageDirectory()	This returns the primary (top-level or root) external storage directory.
getExternalStoragePublicDirectorty()	This returns a top level public external storage directory for shoving files of a particular type based on the argument passed. So basically the external storage has directories like Music, Podcasts, Pictures, etc. whose paths can be determined and returned via this function by passing the appropriate environment constants.

Example to Save a file on External Storage

```
String content = "hello lets save external file";
File file;
FileOutputStream outputStream;
```

```
try{
    file = newFile(Environment.getExternalStorageDirectory(), "MyFile");
    outputStream = newFileOutputStream(file);
    outputStream.write(content.getBytes());
    outputStream.close();
} catch(IOException e) {
    e.printStackTrace();
}
```

Note:-

You can useEnvironment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNLOADS) instead of Environment.getExternalStorageDirectory() to save the file in the Downloads directory of your external storage.

6.3.2 Private files:

- Private files belonging to your app will get deleted once the user uninstalls the app.
- These files will be accessible to other apps but are such that don't provide any value to other apps or even to the user outside the context of the app like certain media (audio, images, etc.) files downloaded by a game.
- Saving files to the directories appropriate for holding your private files is very easy. You can try the previous examples but instead use getExternalFilesDir() to get the appropriate directory path. Again Environment constants like DIRECTORY_MUSIC or DIRECTORY_PICTURES can be passed or you can also pass null to return the root directory for your app's private directory on the volume.
- The path should be Android/data/[package name]/files/ on the external storage. You can check the new files created in Android File Transfer, adb shell or ES file explorer.
- Similar to getCacheDir() in terms of internal storage, we also have getExternalCacheDir() to save cache files in our private external store.

6.4 Creating SQLite database

- **SQLite** is an **open-source relational database** which is used to perform database operations on android devices some of the operations are storing, manipulating or retrieving persistent data from the database.
- It is embedded in android bydefault. Henceyou don't need to perform any database setup or administration task.
- It is very lightweight database that comes with Android OS.

6.4.1 What is CRUD in SQLite Database?

CRUD is an abbreviation for the basic operations that we perform in any database. And the operations are

- Create
- Read
- Update
- Delete

6.5 Editing Tasks with SQLite

Lets see an Example of Creating the SQLite Database for Storing Student Information.

This example allows a user to add, delete, modify and view student details. The application accepts a student's roll number, name and marks and adds these details to a student table. For simplicity, all fields of VARCHAR data type, which is a variable length character string.

The SQLiteDatabase class from the android.database.sqlite package and the Cursor class from the android.database package provide all the functionality required for performing Data Manipulation Language (DML) and query operations on an SQLite table.

Step 1.create an SQLite database and a table in the database.

db=openOrCreateDatabase("StudentDB", Context.MODE_PRIVATE, null);

db.execSQL("CREATE TABLE IF NOT EXISTS student(rollno VARCHAR,name VARCHAR,marks VARCHAR);");

In the above mentioned code, the `openOrCreateDatabase()` function is used to open the `StudentDB` database if it exists or create a new one if it does not exist. Out of different parameters the first parameter of this function specifies the name of the database to be opened or created. The second parameter, `Context.MODE_PRIVATE` which indicates that the database file can only be accessed by the calling application or all applications sharing the same user ID. The third parameter is a Cursor factory object,it can be left `null` if not required.

The db.execSQL() function executes any SQL command. Here it is used to create the student table in case it is not exist in the database.

Step 2: full code of the onCreate() method of the main activity.

```
public void onCreate(Bundle savedInstanceState)
{
<code>super.onCreate(savedInstanceState);
   setContentView(R.layout.main);
// Initializing controls
   editRollno=(EditText)findViewById(R.id.editRollno);
   editName=(EditText)findViewById(R.id.editName);
   editMarks=(EditText)findViewById(R.id.editMarks);
   btnAdd=(Button)findViewById(R.id.btnAdd);
   btnDelete=(Button)findViewById(R.id.btnDelete);
   btnModify=(Button)findViewById(R.id.btnModify);
   btnView=(Button)findViewById(R.id.btnView);
```

```
    btnViewAll=(Button)findViewById(R.id.btnViewAll);
    btnShowInfo=(Button)findViewById(R.id.btnShowInfo);
// Registering event handlers
    btnAdd.setOnClickListener(this);
    btnDelete.setOnClickListener(this);
    btnModify.setOnClickListener(this);
    btnView.setOnClickListener(this);
    btnViewAll.setOnClickListener(this);
    btnShowInfo.setOnClickListener(this);

// Creating database and table
    db=openOrCreateDatabase("StudentDB", Context.MODE_PRIVATE, null);
db.execSQL("CREATE TABLE IF NOT EXISTS student(rollno VARCHAR,name
VARCHAR,marks VARCHAR);");
}
```

In the onClick() event handler, we can write the code required to add, delete, modify and view records.

The following code uses the db.execSQL() function to insert a student record in the student table.

```
db.execSQL("INSERT INTO student VALUES('"+editRollno.getText()+"','"+
editName.getText()+"','"+editMarks.getText()+"');");
```

The above code generates an INSERT statement by appending the contents of the editable fields into a string and executes the INSERT statement.

In the same way, the DELETE command can be executed as follows:

```
db.execSQL("DELETE FROM student WHERE rollno='"+editRollno.getText()+"'");
```

The above mentioned code deletes the record of the student whose roll number is entered in the editable field.

The UPDATE command can be executed as follows:

```
db.execSQL("UPDATE student SET name='"+editName.getText()+"',marks='"+
editMarks.getText()+"' WHERE rollno='"+editRollno.getText()+"'");
```

The above mentioned code updates the record of the student whose roll number is entered in the editable field.

6.6 Cursors and content values

ContentValues are the objects which are used to insert new rows into database tables (and Content Providers). Each Content Values objectrepresents a single row which act as a map of column names to values.
Cursors contain the result set of a query made against a database in Android. The Cursorclass has an API which allows an app to read (in a type-safe manner) the columns that were returned from the query as well as iterate over the rows of the result set.

6.6.1 Reading Cursor Data

An app needs to iterate over the result set and read the column data from the cursor Once a cursor has been returned from a database query. Internally, the cursor stores the rows of data which is returned by the query along with a position that points to the current row of data in the result set. When a cursor is returned from a query() method, its position points to the spot *before* the first row of data. Which means that *before* any rows of data can be read from the cursor, the position must be moved to point to a valid row of data.

The Cursor class provides the following methods to manipulate its internal position:

Table 6.3 Method Description of Cursor Class

Method	Description
boolean Cursor.move(int offset):	It will Moves the position by the given offset
boolean Cursor.moveToFirst():	It Moves the position to the first row
boolean Cursor.moveToLast():	It Moves the position to the last row
boolean Cursor.moveToNext():	It will Moves the cursor to the next row relative to the current position
boolean Cursor.moveToPosition(int position)	It will Moves the cursor to the specified position
Cursor.moveToPrevious():	Moves the cursor to the previous row relative to the current position

Each move() method returns a boolean to indicate whether the operation was successful or not. This flag is useful for iterating over the rows in a cursor.

Lets Continue with above Student Information Application and Display the values of Student using Content Values.

To view a student record need to execute a query using the rawQuery() method of the SQLiteDatabase class as follows:

```
Cursor c=db.rawQuery("SELECT * FROM student WHERE
```

```
rollno="'+editRollno.getText()+"'", null);
if(c.moveToFirst())
{
   editName.setText(c.getString(1));
   editMarks.setText(c.getString(2));
}
```

The above code uses the rawQuery() method of the SQLiteDatabase class to execute the SELECTstatement to select the record of the student, whose roll number is specified. It then checks if the record is found using the moveToFirst() method of the Cursor class and displays the name and marks in the respective editable fields.

The following code can be used To view all records:

```
Cursor c=db.rawQuery("SELECT * FROM student", null);
if(c.getCount()==0)
{
   showMessage("Error", "No records found");
   return;
}
StringBuffer buffer=new StringBuffer();
while(c.moveToNext())
{
   buffer.append("Rollno: "+c.getString(0)+"\n");
   buffer.append("Name: "+c.getString(1)+"\n");
   buffer.append("Marks: "+c.getString(2)+"\n\n");
}
showMessage("Student Details", buffer.toString());
```

The above code executes the SELECT command to retrieve records of all students which is then get appended into a string buffer. Finally, it displays the student details using the user-defined showMessage() function.

Following is the full code of the onClick() event handler:

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

```
{
// Adding a record
   if(view==btnAdd)
   {
   // Checking empty fields
      if(editRollno.getText().toString().trim().length()==0||
        editName.getText().toString().trim().length()==0||
        editMarks.getText().toString().trim().length()==0)
      {
         showMessage("Error", "Please enter all values");
         return;
      }
   // Inserting record
      db.execSQL("INSERT INTO student
VALUES('"+editRollno.getText()+"','"+editName.getText()+
              "','"+editMarks.getText()+"');");
      showMessage("Success", "Record added");
      clearText();
   }
// Deleting a record
   if(view==btnDelete)
   {
   // Checking empty roll number
      if(editRollno.getText().toString().trim().length()==0)
      {
         showMessage("Error", "Please enter Rollno");
         return;
      }
   // Searching roll number
      Cursor c=db.rawQuery("SELECT * FROM student WHERE
rollno='"+editRollno.getText()+"'", null);
      if(c.moveToFirst())
      {
```

```
        // Deleting record if found
            db.execSQL("DELETE FROM student WHERE rollno='"+editRollno.getText()+"'");
            showMessage("Success", "Record Deleted");
        }
        else
        {
            showMessage("Error", "Invalid Rollno");
        }
        clearText();
    }
// Modifying a record
    if(view==btnModify)
    {
    // Checking empty roll number
        if(editRollno.getText().toString().trim().length()==0)
        {
            showMessage("Error", "Please enter Rollno");
            return;
        }
    // Searching roll number
        Cursor c=db.rawQuery("SELECT * FROM student WHERE
rollno='"+editRollno.getText()+"'", null);
        if(c.moveToFirst())
        {
        // Modifying record if found
            db.execSQL("UPDATE student SET
name='"+editName.getText()+"',marks='"+editMarks.getText()+
                    "' WHERE rollno='"+editRollno.getText()+"'");
            showMessage("Success", "Record Modified");
        }
        else
        {
            showMessage("Error", "Invalid Rollno");
```

```
        }
        clearText();
    }
// Viewing a record
    if(view==btnView)
    {
    // Checking empty roll number
        if(editRollno.getText().toString().trim().length()==0)
        {
            showMessage("Error", "Please enter Rollno");
            return;
        }
    // Searching roll number
        Cursor c=db.rawQuery("SELECT * FROM student WHERE
rollno="'+editRollno.getText()+"'", null);
        if(c.moveToFirst())
        {
        // Displaying record if found
            editName.setText(c.getString(1));
            editMarks.setText(c.getString(2));
        }
        else
        {
            showMessage("Error", "Invalid Rollno");
            clearText();
        }
    }
// Viewing all records
    if(view==btnViewAll)
    {
    // Retrieving all records
        Cursor c=db.rawQuery("SELECT * FROM student", null);
    // Checking if no records found
```

```
        if(c.getCount()==0)
        {
            showMessage("Error", "No records found");
            return;
        }
    // Appending records to a string buffer
        StringBuffer buffer=new StringBuffer();
        while(c.moveToNext())
        {
            buffer.append("Rollno: "+c.getString(0)+"\n");
            buffer.append("Name: "+c.getString(1)+"\n");
            buffer.append("Marks: "+c.getString(2)+"\n\n");
        }
    // Displaying all records
        showMessage("Student Details", buffer.toString());
    }

// Displaying info
    if(view==btnShowInfo)

    {
    showMessage("Student Management Application", "Developed By Azim");
    }
}
```

The following user-defined function is used to display message to the user:

```
public void showMessage(String title,String message)
{
    Builder builder=new Builder(this);
    builder.setCancelable(true);
    builder.setTitle(title);
    builder.setMessage(message);
```

```
    builder.show();
}
```

The following user-defined function is used to clear edit fields:

```
public void clearText()
{
    editRollno.setText("");
    editName.setText("");
    editMarks.setText("");
    editRollno.requestFocus();
}
```

Screen Shots of Student Information Application

Sqlite Database Advantages and Disadvantages are as follows;

Table 6.4 Method Description of Cursor Class

Advantages	Disadvantages
Toolchain, e.g. DB browser	Using SQLite means a lot of boilerplate code and thus inefficiencies (also in the long run with the app maintenance)
No dependencies, is included with Android and iOS	1 MB BLOB Limitation on Android
Developers can define exactly the data schema they want	No compile time checks (e.g. SQL queries)
Developers have full control, e.g. handwritten SQL queries	The performance of SQLite is unreliable
SQL is a powerful and established query language, and SQLite supports most of it	SQL is another language to master
Debuggable data: developers can grab the database file and analyze it	SQL queries can get long and complicated
Rock-solid, widely used technology, established since the year 2000	Testability (how to mock a database?)

6.7 Working with Android database

6.7.1 What are SQLite alternatives?

There are plenty of SQLite alternatives. If you simply find it unpleasant to write a lot of SQL and boilerplate code, you can use an object abstraction on top of SQLite. This abstraction is usually an ORM (object/relational mapping). But if you want to replace SQLite completely, there are also quite a few alternative databases: Couchbase Lite, Interbase, LevelDB, Oracle Berkeley DB (formerly Oracle's mobile database was "Oracle Database Lite"), Realm, SnappyDB, Sparksee Mobile (graph database, brand-new at the time of this article), SQL Anywhere, SQL Server Compact (discontinued), and UnQLite.

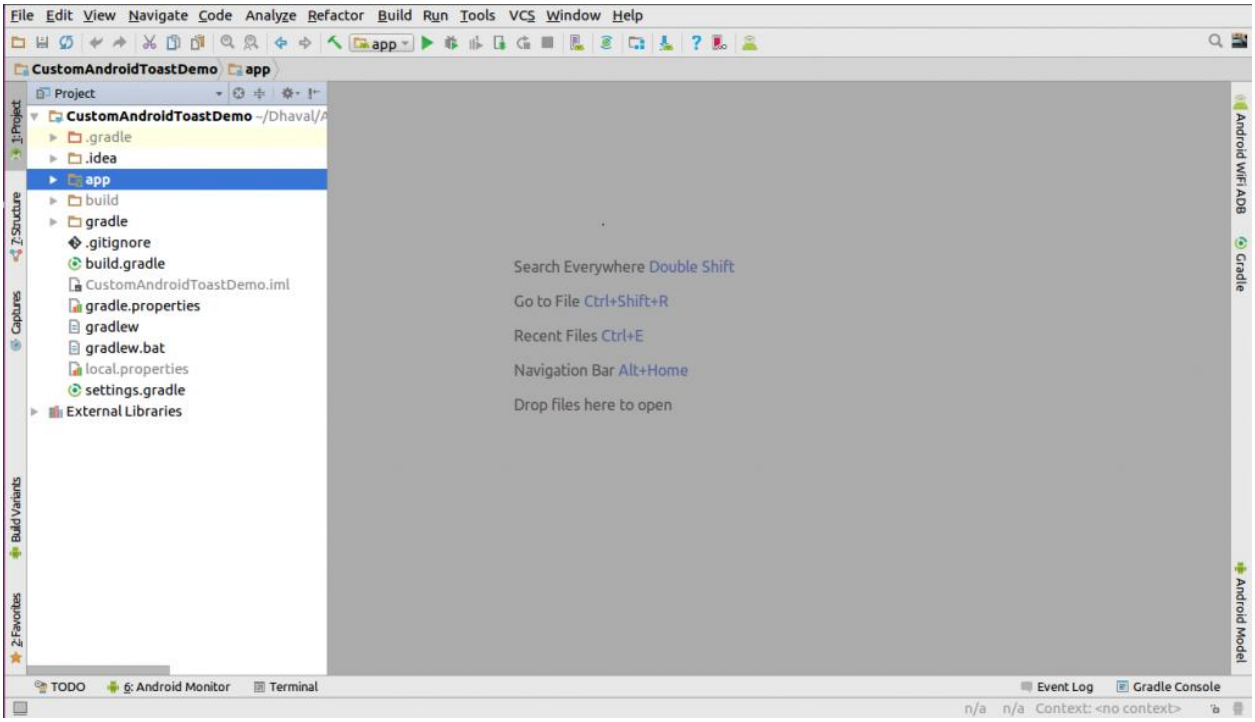
To give you an overview, look at the following comparison table:

Name	Type of DataStored	Datalevel Encryption	Short Description
Couchbase Lite	JSON Documents / NoSQL db	Database encryption with SQLCipher	Embedded / portable db with P2P and central synchronization (sync) support. Secure SSL.
ForestDB	Key-value pairs / NoSQL db	No	Portable lightweight key-value store, NoSQL database.
Interbase	Relational	Depends on version (Lite versus ToGo).	Embeddable SQL database.
LevelDB	Key-value pairs / NoSQL db	No	Portable lightweight key-value store, NoSQL db, doesn't support indexes, very fast for some use cases; earlier available benchmarksfrom 2011 have been removed unfortunately.
Oracle Berkeley DB	Relational and Key-Value-Store	128-bit AES Standard encryption	Embedded / portable db with P2P and central sync support as well as support for sync with SQLite.
Snappy DB	Key-value pairs / NoSQL db	No	Portable lightweight key-value store, NoSQL db based on LevelDB.
Realm	Object Database	Yes	Embedded object db.

6.8 Publish Android Application in Android Market

1. Open your Android Studio.

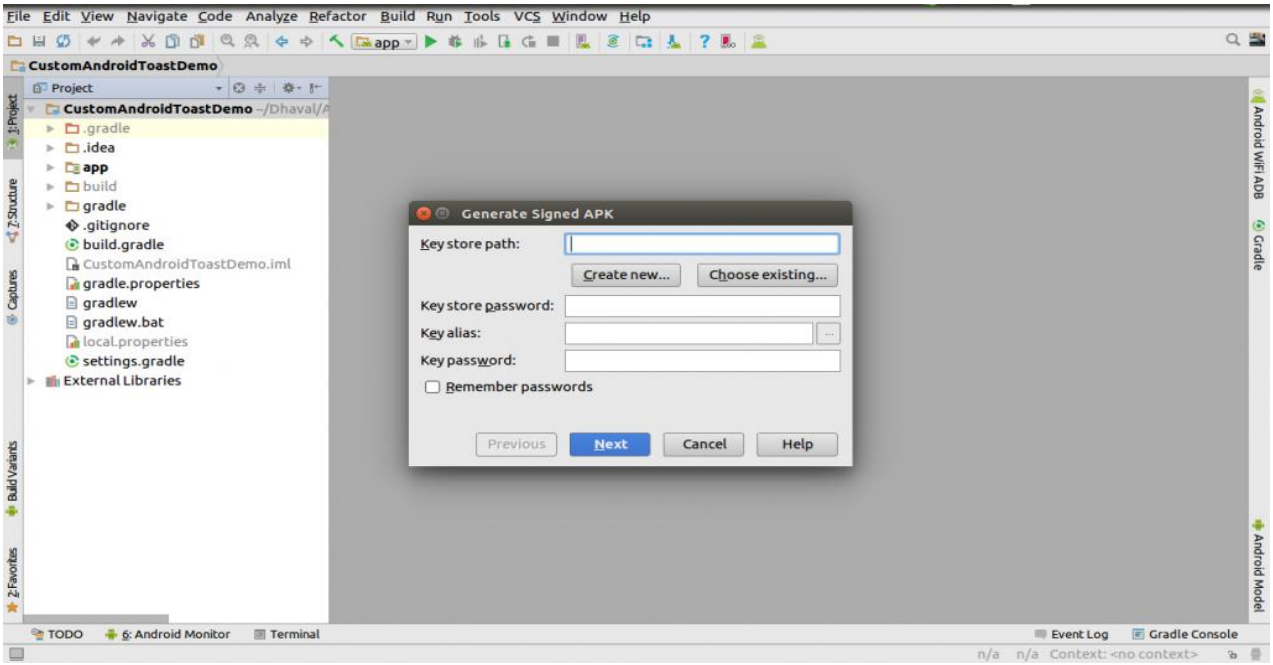
2. Now open the Android app project that you want to upload to Google play store.



3. Click on the 'Build' from toolbar option and select 'Generate Signed APK'

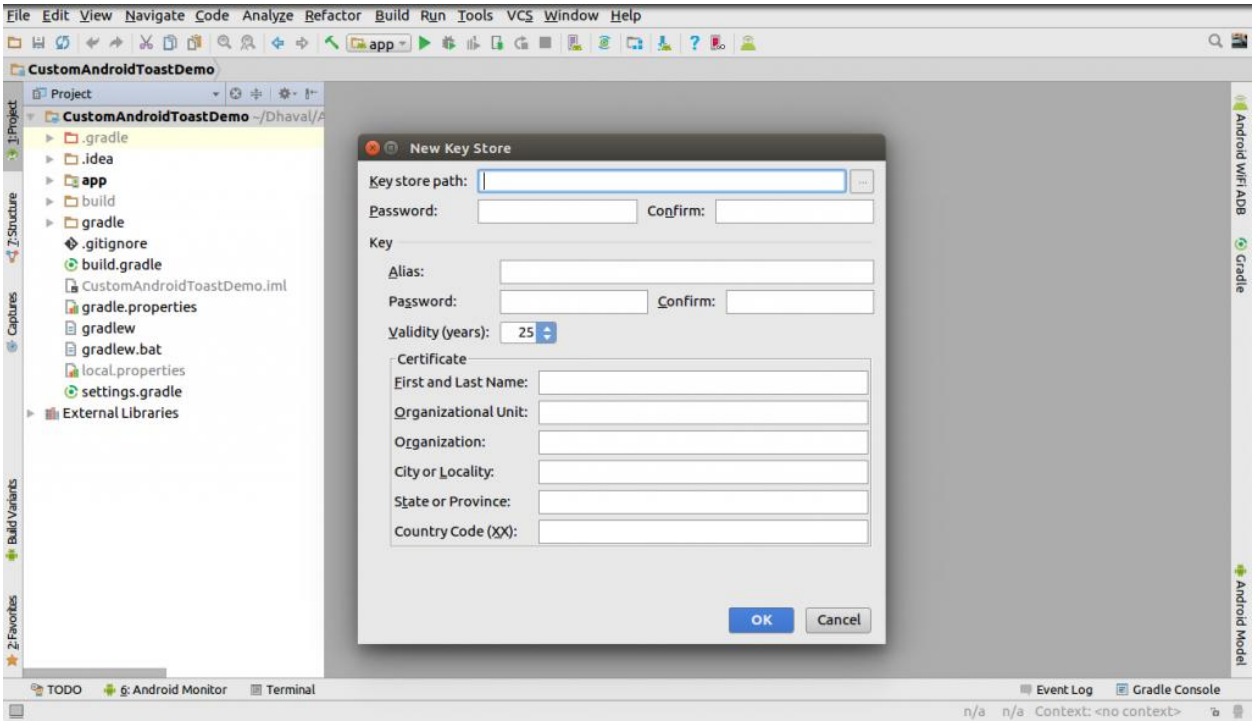
4. Next, enter your keystore details. Here, you'll have two options.

- Create New Keystore
- Choose Existing Keystore



If you've already created keystore file, select that keystore file path and enter keystore password, key alias, and key password information.And, if you haven't created a keystore file, then click on create a new button.

5. Process For Creating New Keystore



Once you click on Create new button, which will ask you to define Keystore path and will also ask enter following mandatory information.

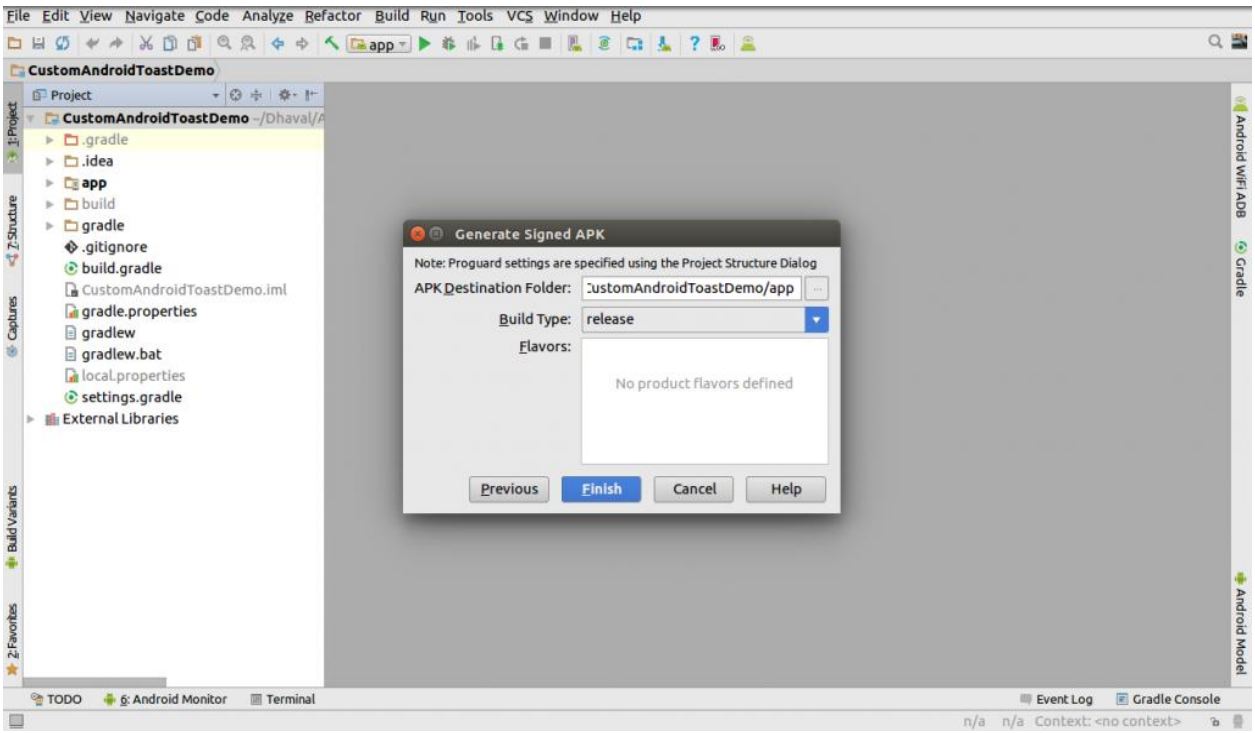
Password and Confirm Password

Key Alias Name

Key Password and Confirm Password

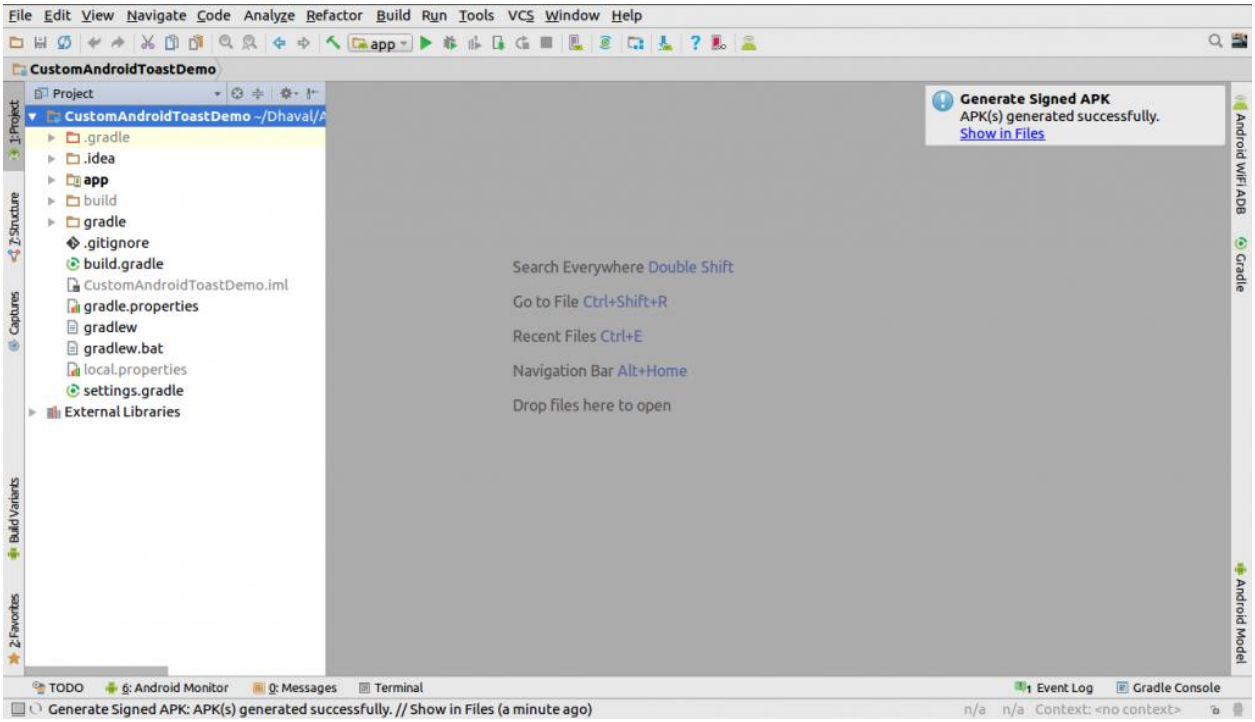
First Name and Last Name of your Android app certificate. (certificate name can be your App name or your organization name.)

6. Generate Signed APK



Once you've successfully created your Keystore, click on Generate Signed APK. it will ask you to define destination folder of APK. And, while uploading your Android app, you need to select 'Release Build Type'.

After completing all process, you'll get a notification as APK Generated Successfully on the top-right corner of Android Studio.



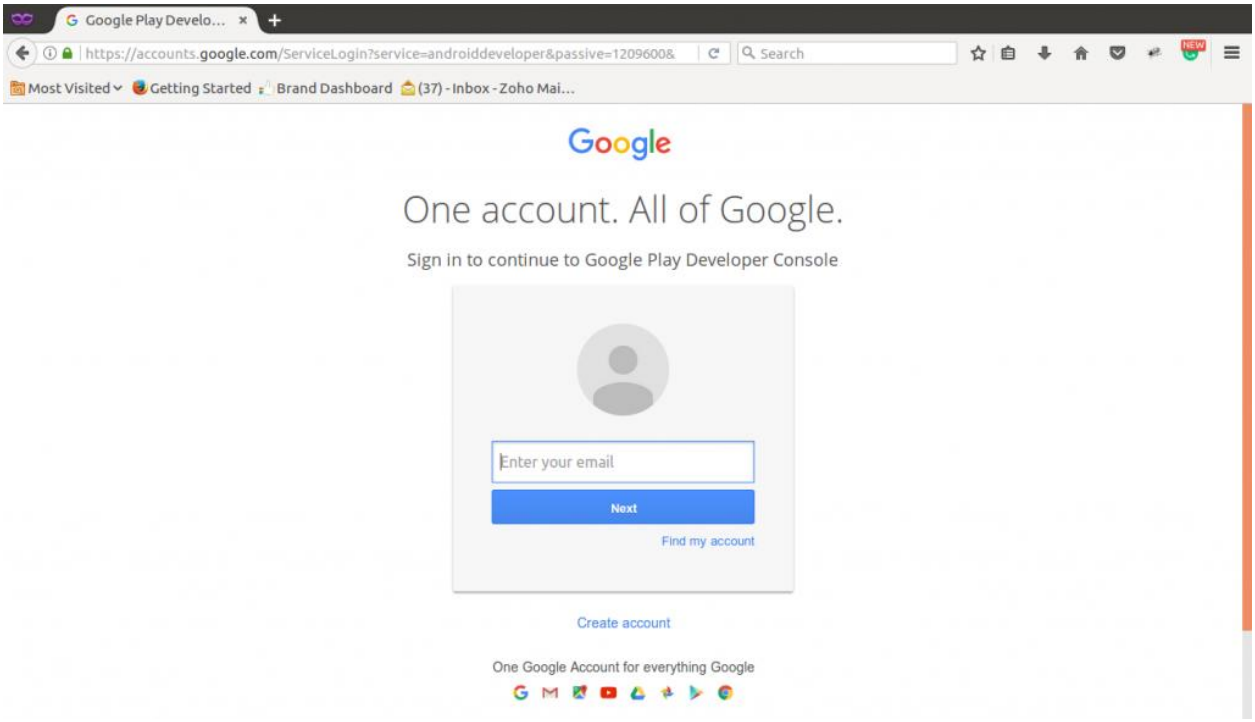
7. Open Google Developer Console

Next, open your web browser and go to Google Play Developer Console from following link

– https://play.google.com/apps/publish/.

Login to your Google developer account.

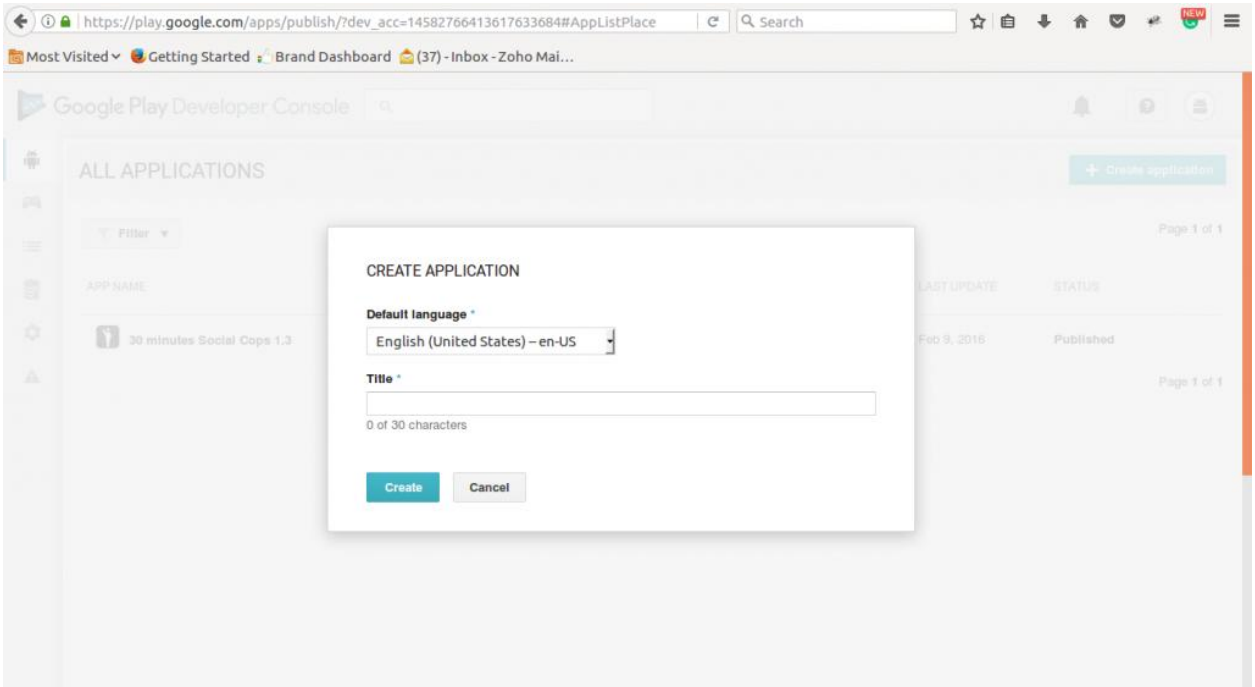
In case you don’t have developer account, click on Sign Up button in the right corner and follow the general steps for creating a new Google account.



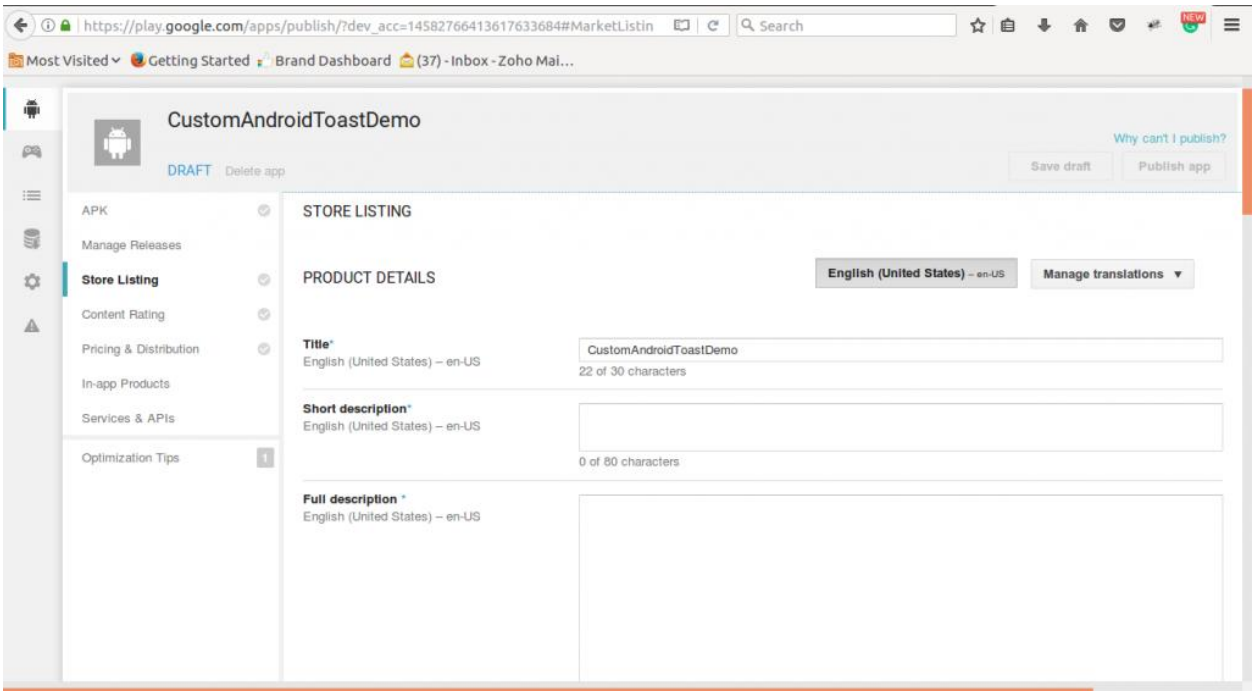
Although, you'll required to pay $25 one-time fee for creating developer account.

Once you pay the fee and create your Google developer account and then you can login to developer console with the same account.Here, click on 'Create New Application' located at top right corner of your screen.

8. Create New Application



In the pop-up screen you have to enter the name of your application, of lenghth up to 30 characters. Next, add a description for your Android application. You can enter app description up to 4000 characters in the description field.

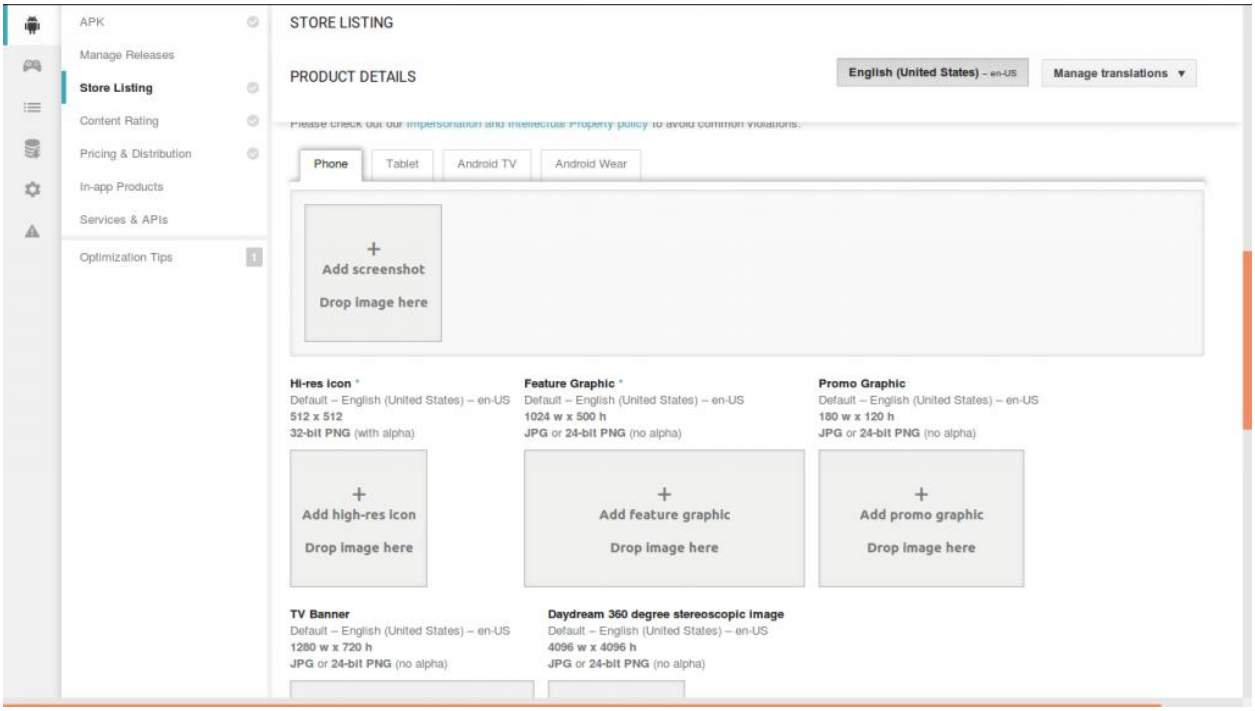


9. Now upload the graphic assets. Here, you'll be asked to upload different images for yourapplication.

Screenshots (minimum 2 screenshots are required. Maximum 8 screenshots).

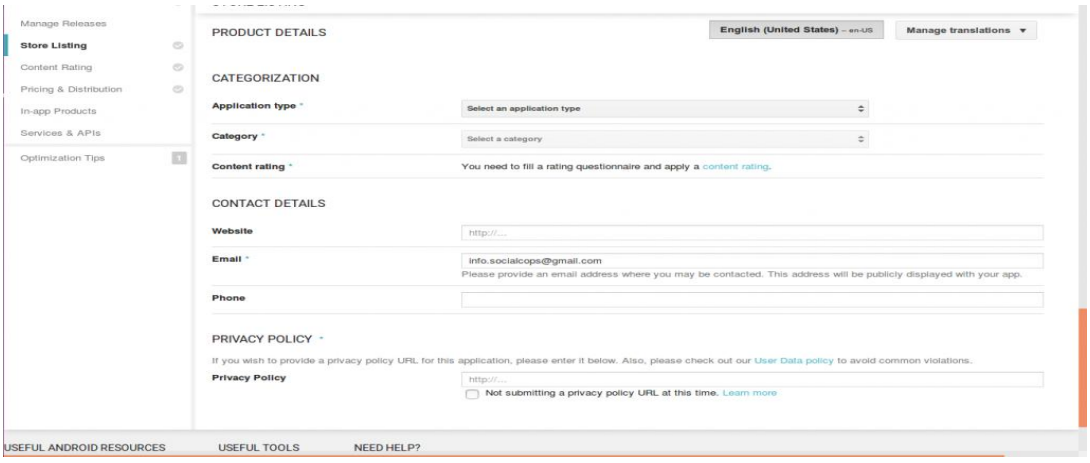
High resolution icon (512 x 512 32-bit PNG (with alpha))

Feature Graphic (1024 x 500 h JPG or 24-bit PNG (no alpha))

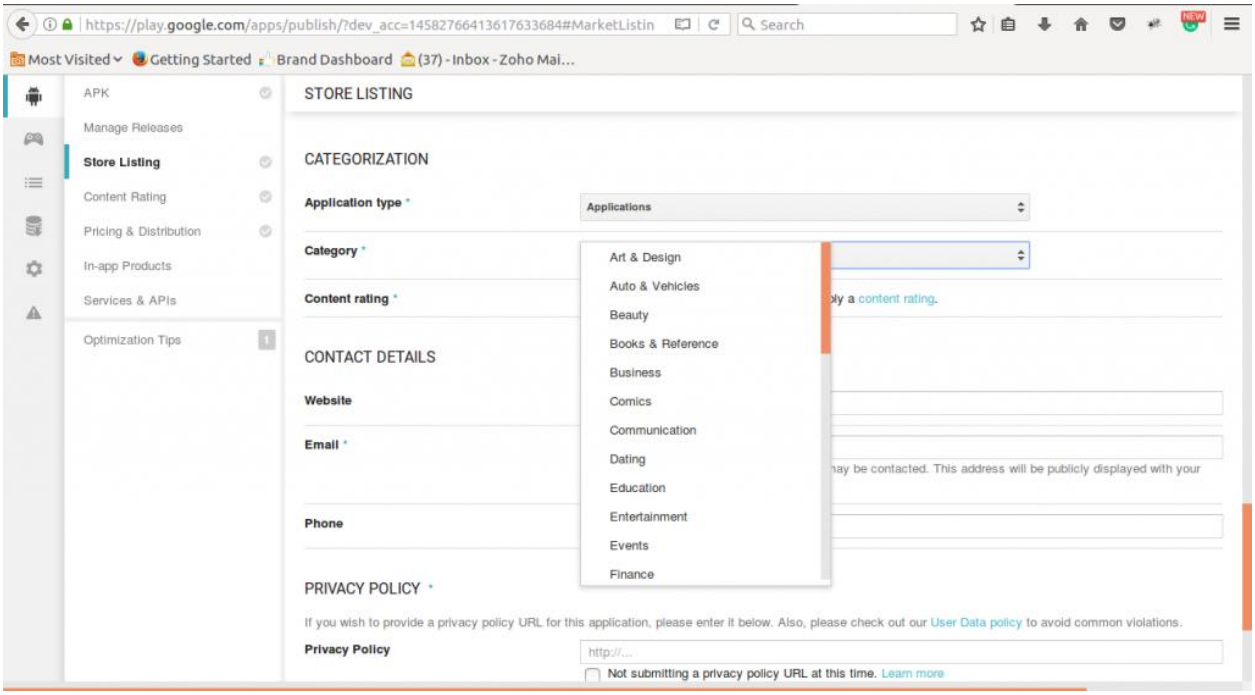


10. After uploading a graphic, select your application category.

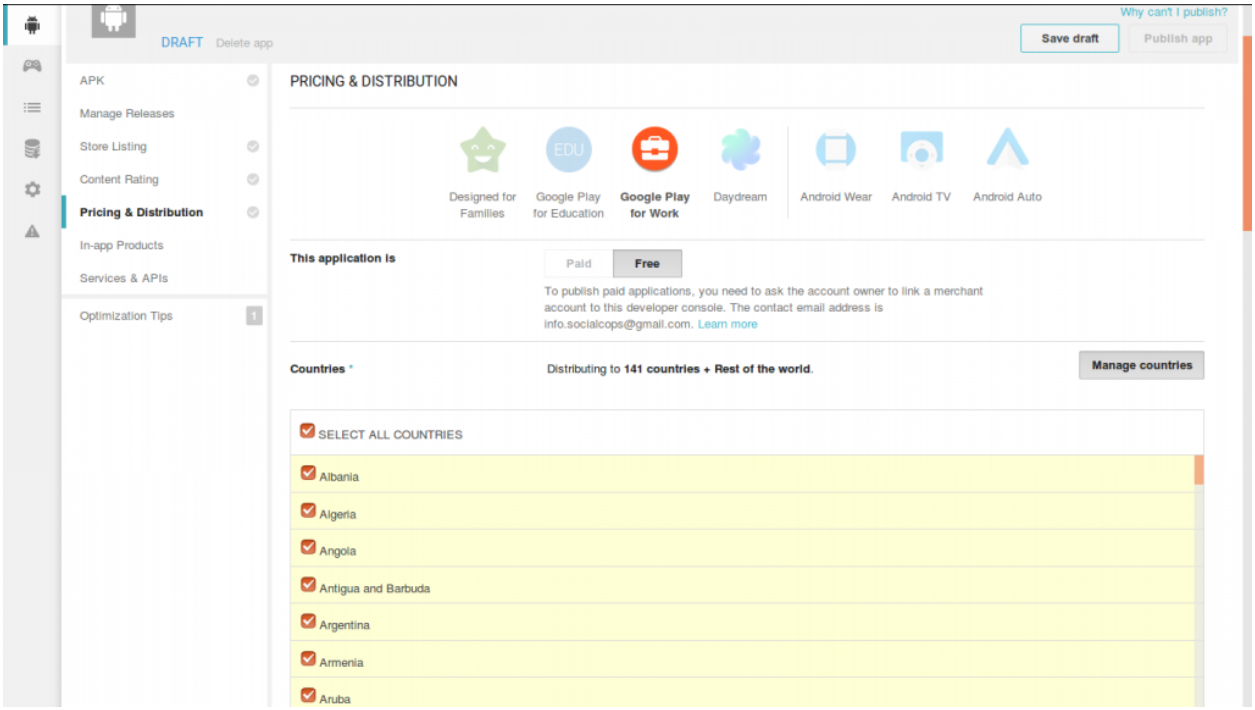
- Application
- Games



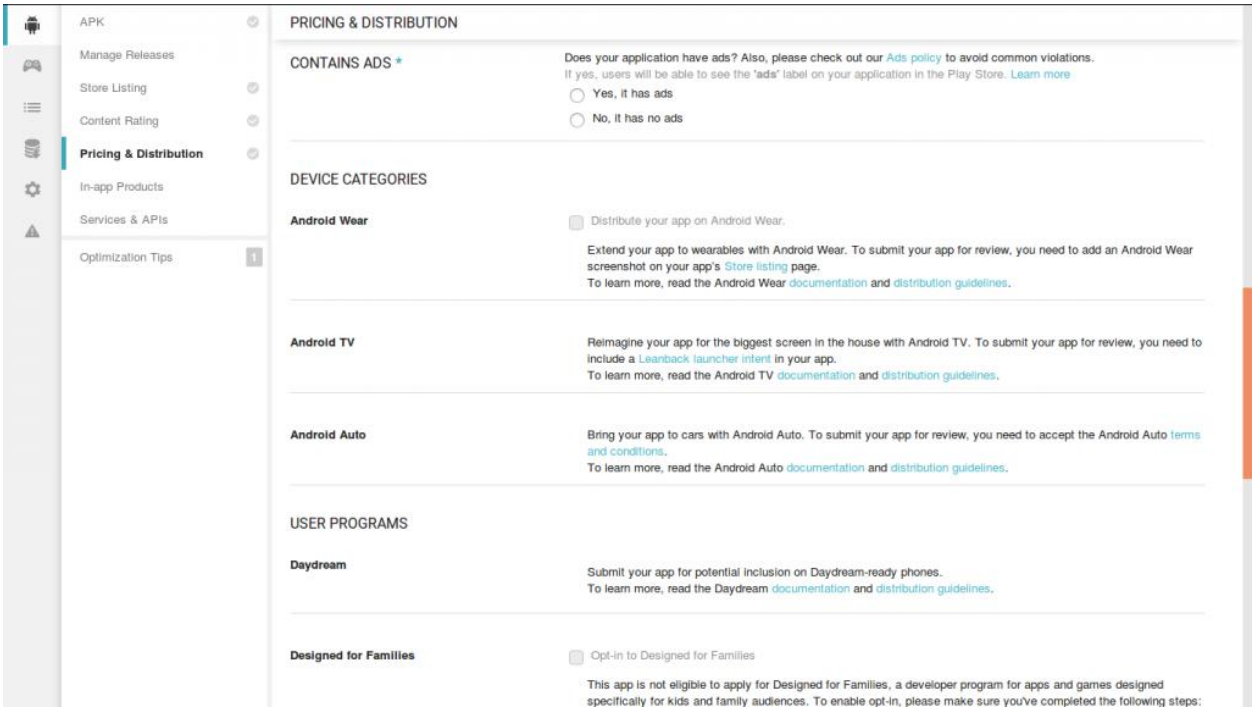
11. In the next drop-down list, select a respective category for your Android application.



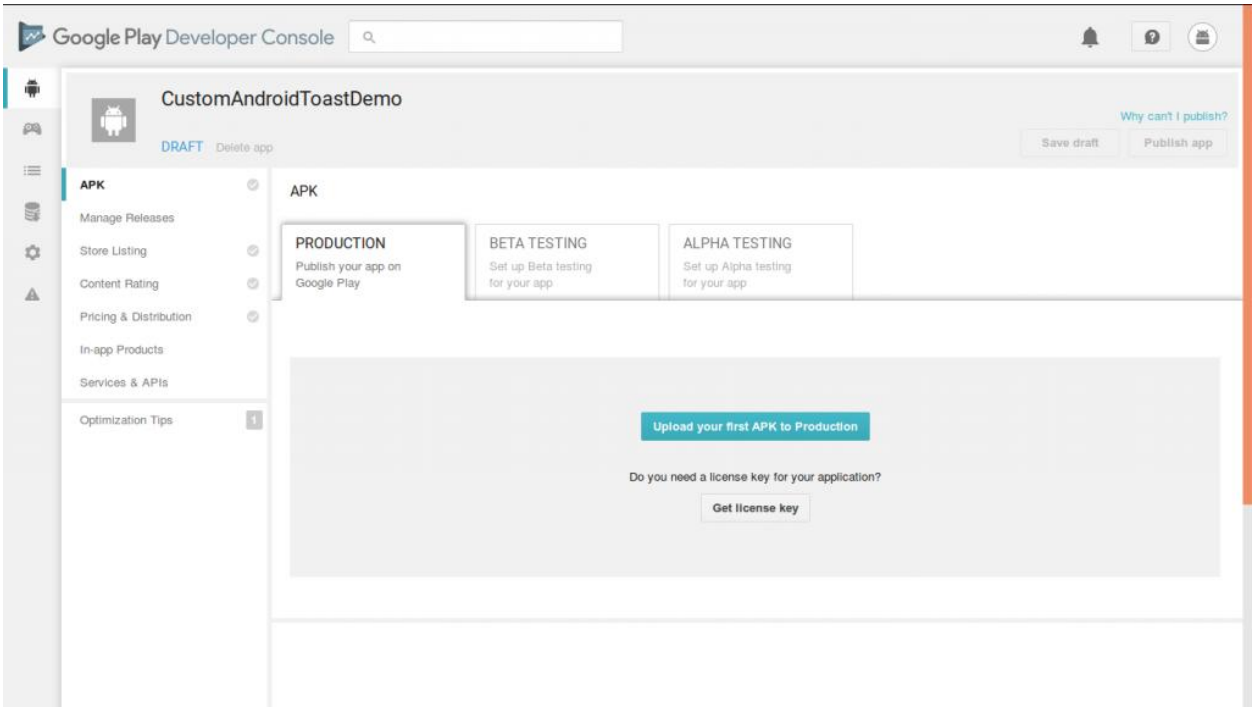
12. Next, add privacy policy URL if you're collecting personal and sensitive information.Now go toPrice & Distribution tab, and choose whether you're uploading your Android app as free or paid.Also, you can select in which countries your app should be distributed.

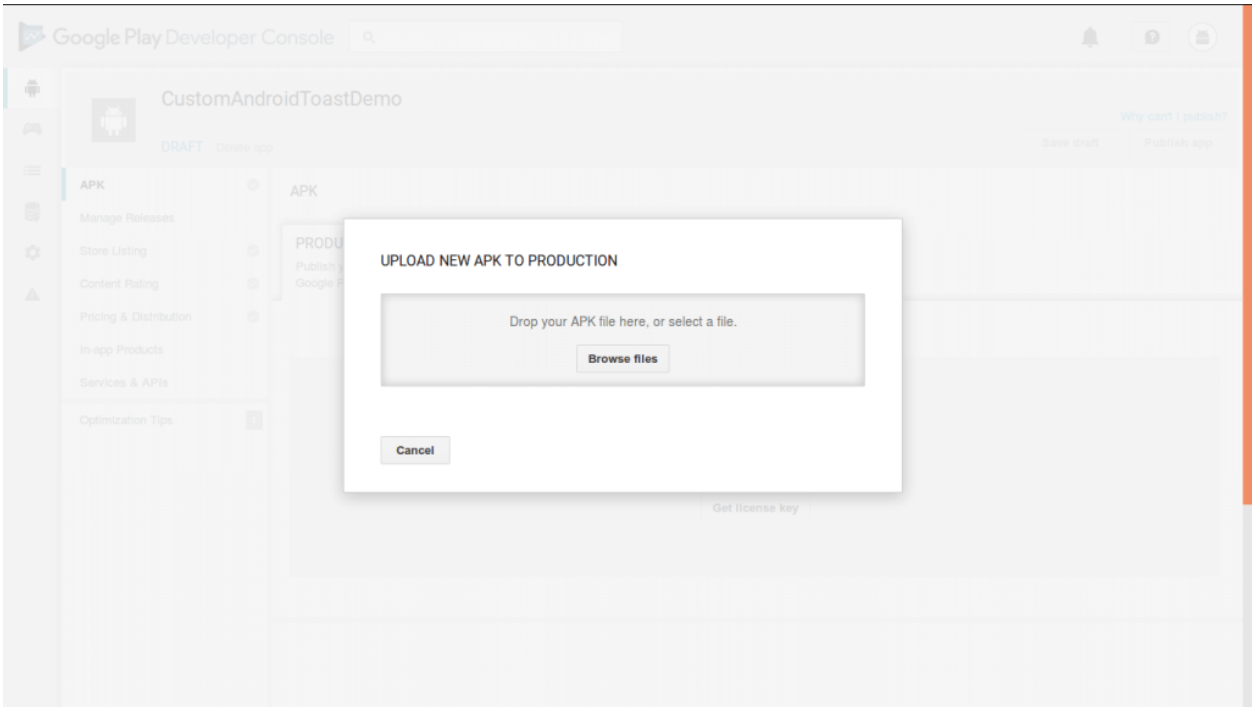


13. Once you add details for price & distribution, you'll have to define whether your app contains adsor not. Click on Yes if it'll have ads.

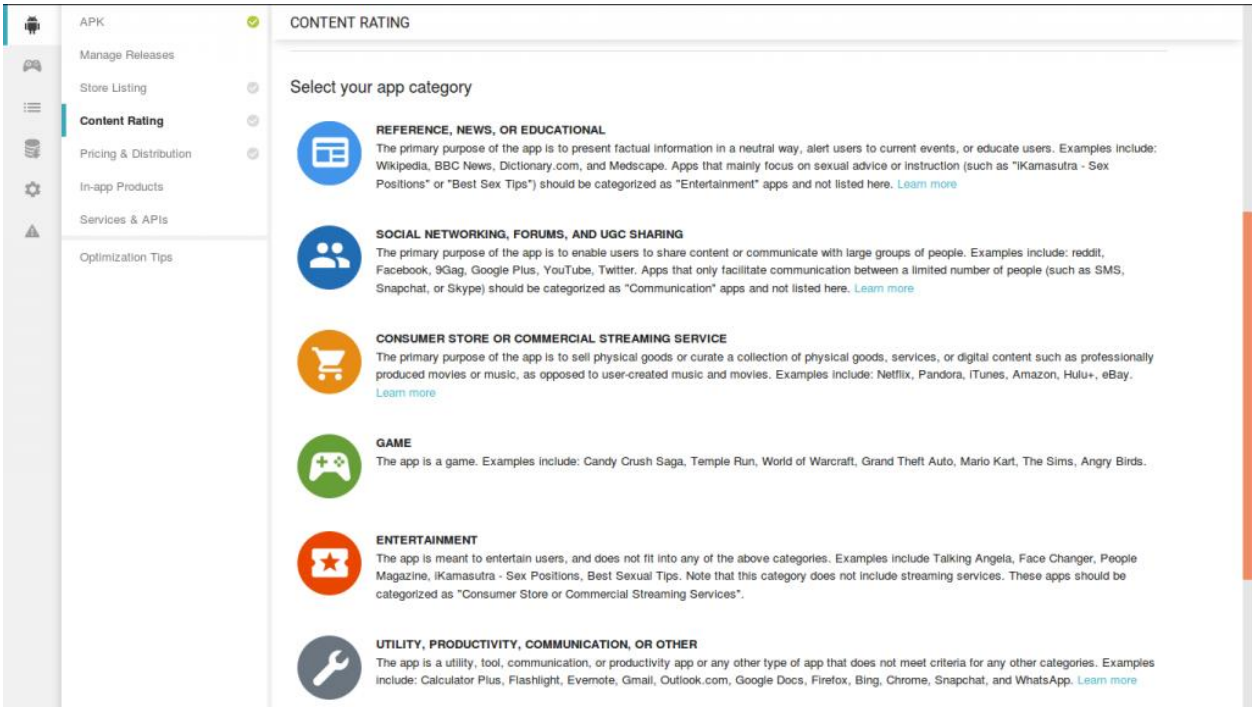


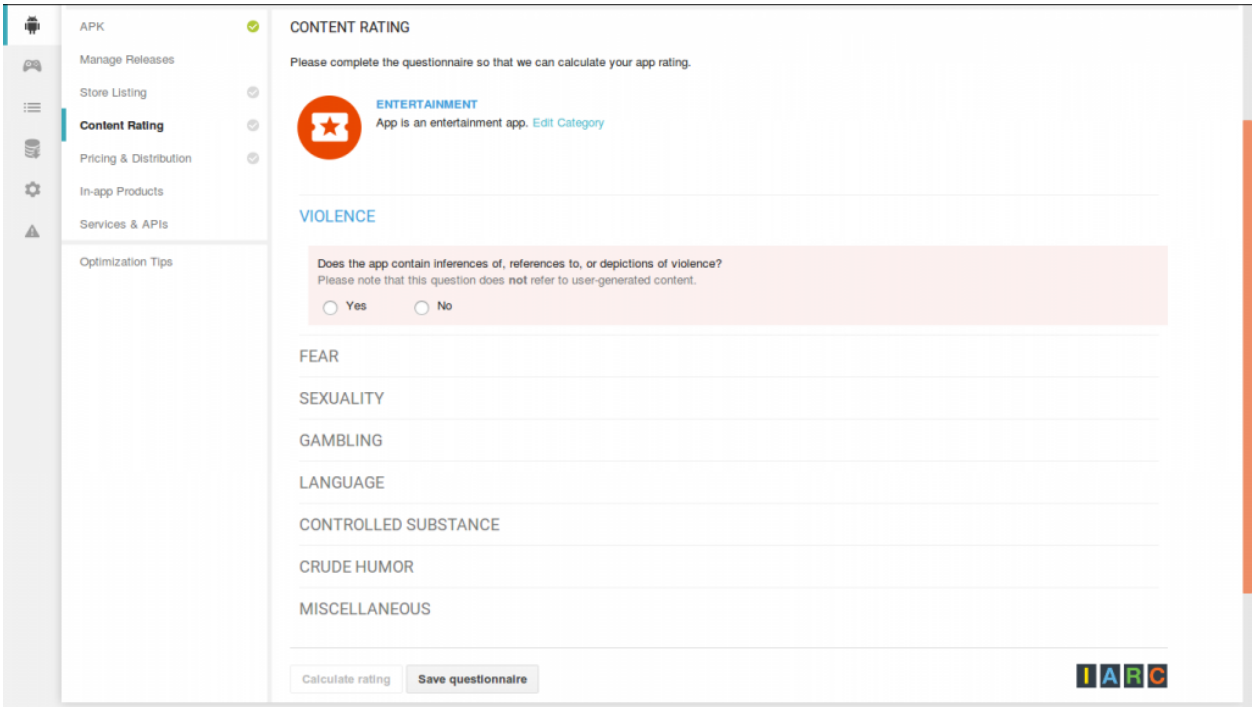
14. Next, upload your Android application. Click on 'Upload your first APK to Production'button.Here, you'll be asked to upload your APK file. Click on Browse files and select your APKfile.

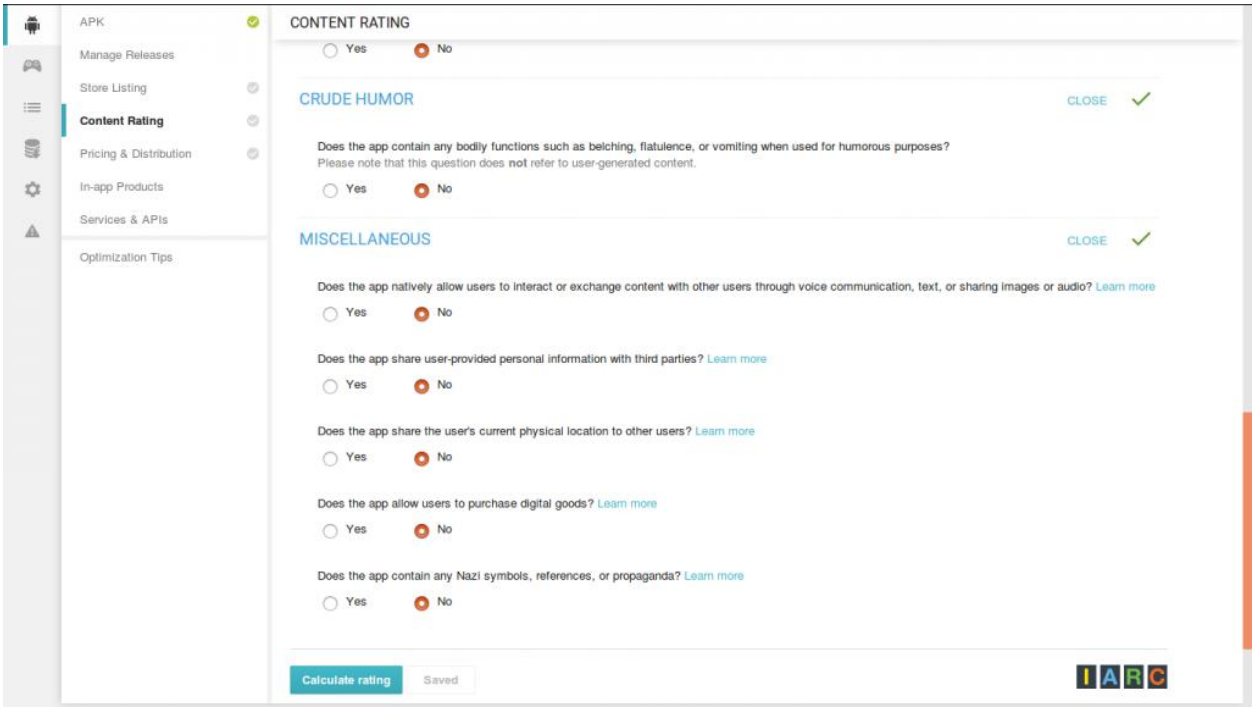


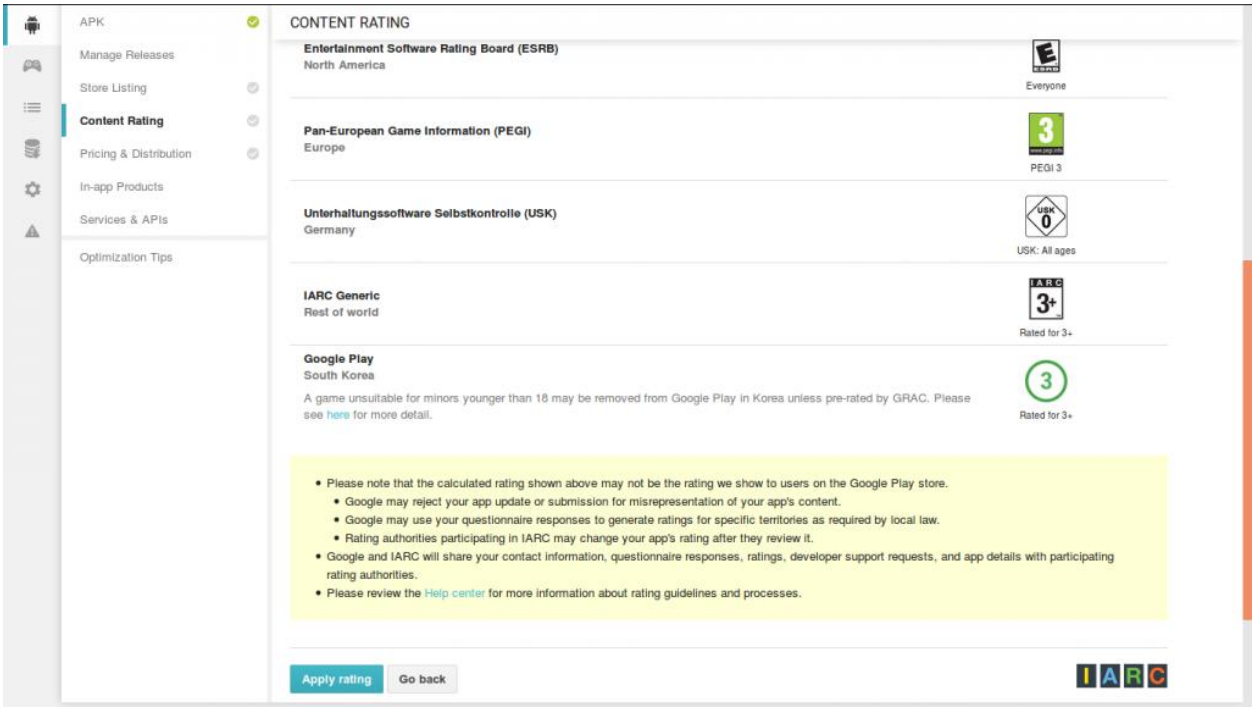


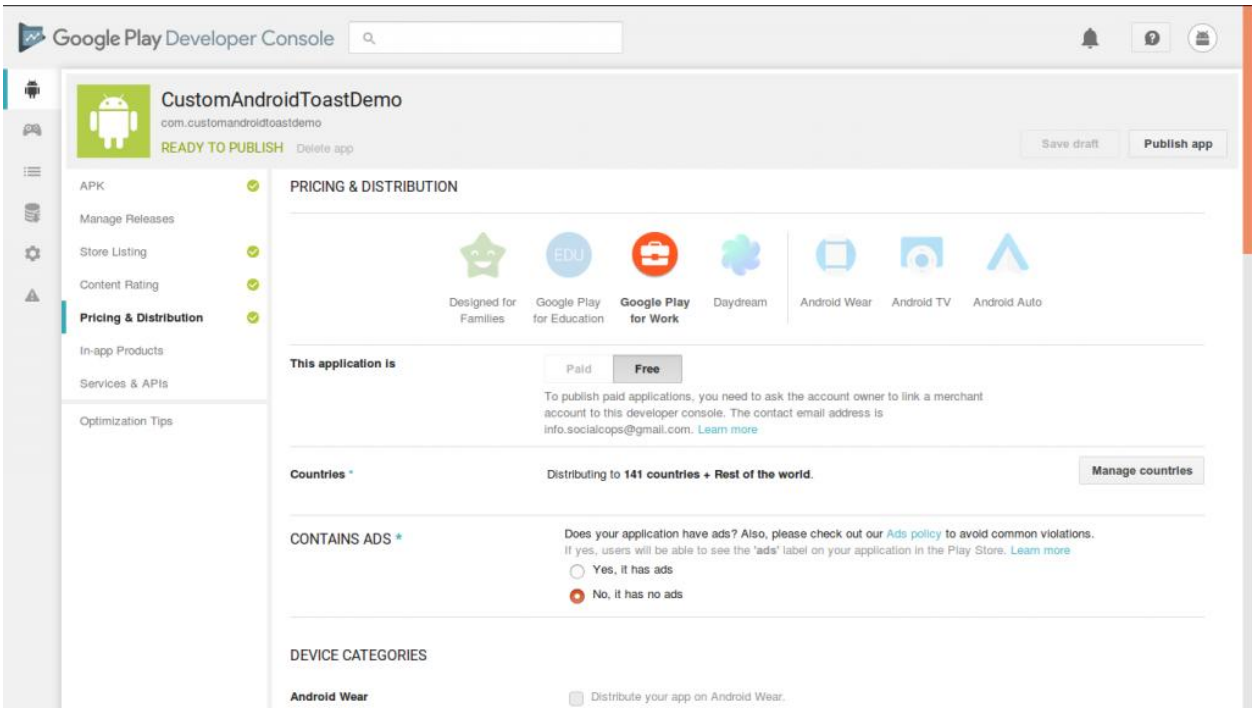
15. For Content Rating, go to its tab. Select your app category. Then you'll be asked to fill one form.Save this questionnaire after filling and click on Calculate rating.




APK
Manage Releases
Store Listing
Content Rating
Pricing & Distribution
In-app Products
Services & APIs
Optimization Tips
CONTENT RATING
Please complete the questionnaire so that we can calculate your app rating.
ENTERTAINMENT
App is an entertainment app. Edit Category
VIOLENCE
Does the app contain inferences of, references to, or depictions of violence?
Please note that this question does not refer to user-generated content.
Yes
No
FEAR
SEXUALITY
GAMBLING
LANGUAGE
CONTROLLED SUBSTANCE
CRUDE HUMOR
MISCELLANEOUS
Calculate rating
Save questionnaire
IARC


APK
Manage Releases
Store Listing
Content Rating
Pricing & Distribution
In-app Products
Services & APIs
Optimization Tips
CONTENT RATING
Yes
No
CRUDE HUMOR
CLOSE
Does the app contain any bodily functions such as belching, flatulence, or vomiting when used for humorous purposes?
Please note that this question does not refer to user-generated content.
Yes
No
MISCELLANEOUS
CLOSE
Does the app natively allow users to interact or exchange content with other users through voice communication, text, or sharing images or audio? Learn more
Yes
No
Does the app share user-provided personal information with third parties? Learn more
Yes
No
Does the app share the user's current physical location to other users? Learn more
Yes
No
Does the app allow users to purchase digital goods? Learn more
Yes
No
Does the app contain any Nazi symbols, references, or propaganda? Learn more
Yes
No
Calculate rating
Saved
IARC





16. Finally, once you've completed all these steps, you'll see 'Ready to Publish' text written on thetop-left corner of your screen. Now, just hit the publish button and your Android app will bepublished.

Summary:

After going through this unit, you will be able to:

- Understand File system in android
- Differentiate Internal and external Storage.
- Create of SQL Database.
- Editing Task with SQL Lite
- Work on Cursor and content values.
- Work with android database.
- Publish android application.

Exercise:

1. Write a short note on File system in Android.
2. Discuss Internal and External Storage in Android.
3. Write a short note on Sqlite Database.
4. What are the most important features of SQLite Database
5. Compare SQL with SQLite Database
6. What is the use of Cursor? Explain with example.
7. Explain Content Values in Details.
8. Write a program to insert and display the database values using SqliteDatabase.
9. Write down the steps to publish the Android App in Android Market.

UNIT 7
PROVIDERS

Learning Objectives:

After going through this unit, you will be able to learn:

- What is Content Providers
- Content Providers fundamentals and types.
- How to create content provider.
- Contact Content Provider
- Other Build in Contact Content Provider
- Creating Custom Contact Content Provider
- Working with Content providers.

7.1 Content Provider

- In android, **Content Provider** acts as a central repository to store the applications data in one place and make that data available for different applications to access whenever it's required.
- Content Providers can be configured to allow other applications securely access and modify our app data based on our requirements.
- **Content Provider** is a part of an android application and it will act as relational database to store the app data. We can perform a multiple operations like insert, update, delete and edit on the data stored in content provider using **insert()**, **update()**, **delete()** and **query()** methods.
- Content provider can be used whenever we want to share our app data with other apps and it allow us to make a modifications to our application data without effecting other applications which depends on our app.
- In android, content provider is offering different ways to store app data. The app data can be stored in in files or in a SQLite database or even over a network based on our requirements. Content providers can manage different data formats such as audio, video, images and personal contact information.
- Different types of access permissions are offered in content provider to share the data. It also allows restricting access permissions in content provider to restrict data access limited to only our application and we can configure different permissions to read or write a data.

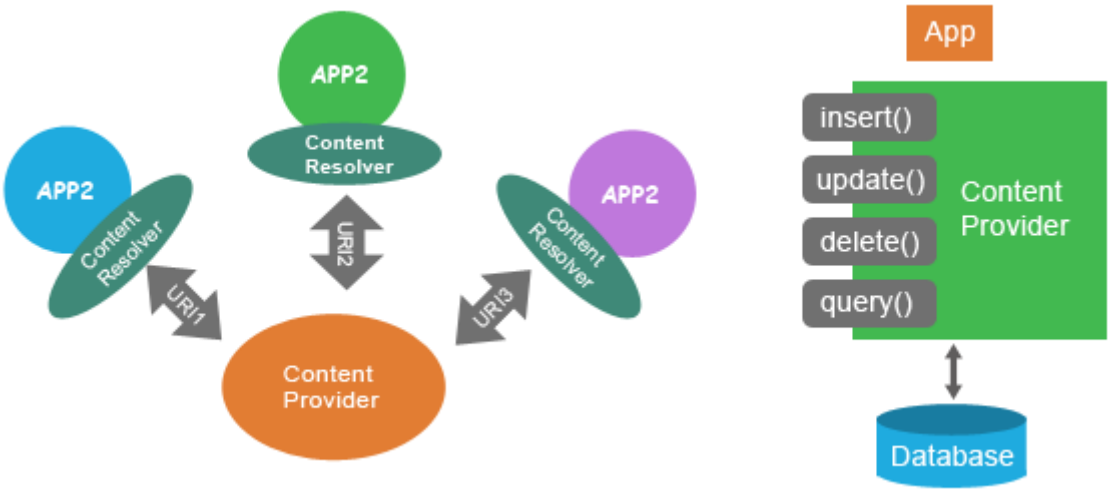


Figure: 7.1 Content Provider in Android

7.2 Content Provider Fundamentals

To create your own content provider we need to extend the ContentProvider class and override some methods defined within it.In order to extract/manipulate data we mainly need two things:

- Uri
- Content Resolver object.

7.2.1 Uri: To get the data from an application we need Uri, the path where actual data is stored in a table.

Synatx:-

<prefix>://<authority>/<data_type>/<id>

Details of Various Parts of the URI is as follows;

Part	Description
Prefix	This is always set to content://
authority	Authorityspecifies the name of the content provider, for example contacts, browser etc. In case of third-party content providers, this could be the fully qualified name, such as com.tutorialspoint.statusprovider
data_type	It indicates the type of data that this particular provider provides. Let us consider with For example, if you are getting all the contacts from the Contacts content provider, then the data path would be people and URI would look like thiscontent://contacts/people
id	id specifies the specific record requested. For example, if you are looking for contact name"abc" in the Contacts content provider then URI would look like this *content://contacts/people/abc*.

Example :

-to get contact details

content: //contacts/people

-to get bookmark details from browser

content: //browser/bookmarks

The general *syntax* for the URI is

<Standard prefix> ://< authority>/< data path>/<id>

In Android every content provider URI starts with

content://

if we want to get the 5th contact from the Contact list then the example would be :

7.2.2 Content Resolver: To extract/get the data provided by the content provider's we use content resolvers. The content resolver job is to dispatch our requests to a content provider, based on the given Uri. so whenever, we try to get data from ContentResolver, the system evaluates the given Uri and passes the request to the ContentProvider.

7.3 How to Create a Content Provider?

- To create your own content provider it involves number of simple steps.
- First step begins with creation of a Content Provider class that extends the ContentProviderbaseclass.
- next, you need to define your content provider URI address which will be used to access the content.
- Next you will need to create your own database to keep the content. Android uses SQLite database and framework which needs to override onCreate() method which will use SQLite Open Helper method to create or open the provider's database. When newly created application is launched, the onCreate() handler of each of its Content Providers is called on the main application thread.
- Next step is to implement Content Provider queries to perform different database specific operations.
- The process ends with registering your Content Provider in your activity file using <provider> tag.

Here is the list of methods which you need to override in Content Provider class to have your Content Provider working −

onCreate()	This method is called when the provider is started.
query()	This method receives a request from a client. The result is returned as a Cursor object.
insert()	This method inserts a new record into the content provider.
delete()	This method deletes an existing record from the content provider.
update()	This method updates an existing record from the content provider.
getType()	This method returns the MIME type of the data at the given URI.

7.4 Contact Content Provider

Here we will see an Android application which will query Android contacts content provider to retrieve the contacts available in the phone and list those contacts in a listview with various details such as name, mobile number, home number, work email id, photo etc.

Step 1:

To access permission of read and write data, and for this, we need to write permission in the Android Manifest file, like below:

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
</manifest>
```

Note- Here is an exception. From Android 6.0 Marshmallow, the application will not be granted any permissions at installation time. Instead, the application has to ask the user for permissions one-by-one at runtime with an alert message. The developer has to call for it manually.

Step 2:

Next, in the main layout, we are giving a simple button and two text views. In this step on a button click event, we will access all contacts on our device and display the contact names with their number on those text views. Here is the layout XML code:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical">
<Button
      android:id="@+id/btnload"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_gravity="center"
      android:gravity="center"
      android:text="View contacts"
      android:textSize="25sp"
      android:layout_marginTop="20dp"/>
```

```
<TextView
        android:id="@+id/txtname"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="10dp"/>
<TextView
        android:id="@+id/txtphno"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="10dp"/>
</LinearLayout>
```

Step 3:

In the MainActivity page, first, we define all the views of the main layout, like below:

```
Button btnview;
TextView  txtname,txtphno;
static final int PICK_CONTACT = 1;
String st;
final private int REQUEST_MULTIPLE_PERMISSIONS = 124;
@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.contact);
   AccessContact();
   btnview = (Button) findViewById(R.id.btnload);
   txtname=(TextView) findViewById(R.id.txtname);
   txtphno=(TextView) findViewById(R.id.txtphno);
```

Here we are calling the AccessContact function for runtime permissions, as discussed above.

```
private void AccessContact()
```

```
{
List<String> permissionsNeeded = new ArrayList<String>();
final List<String> permissionsList = new ArrayList<String>();
   if (!addPermission(permissionsList, Manifest.permission.READ_CONTACTS))
      permissionsNeeded.add("Read Contacts");
   if (!addPermission(permissionsList, Manifest.permission.WRITE_CONTACTS))
      permissionsNeeded.add("Write Contacts");
   if (permissionsList.size() > 0) {
      if (permissionsNeeded.size() > 0) {
        String message = "You need to grant access to " + permissionsNeeded.get(0);
         for (int i = 1; i < permissionsNeeded.size(); i++)
            message = message + ", " + permissionsNeeded.get(i);
         showMessageOKCancel(message,
               new DialogInterface.OnClickListener() {
                  @Override
                  public void onClick(DialogInterface dialog, int which) {
                     requestPermissions(permissionsList.toArray(new String[permissionsList.size()]),
                           REQUEST_MULTIPLE_PERMISSIONS);
                  }
               });
         return;
      }
requestPermissions(permissionsList.toArray(new String[permissionsList.size()]),
            REQUEST_MULTIPLE_PERMISSIONS);
      return;
   }
}
```

Here we have requested read and write contact permissions at runtime. For this, we have added all the permission in a List<String>.

```
private boolean addPermission(List<String> permissionsList, String permission) {
   if (checkSelfPermission(permission) != PackageManager.PERMISSION_GRANTED) {
      permissionsList.add(permission);
```

```
    if (!shouldShowRequestPermissionRationale(permission))
        return false;
  }
  return true;
}
```

If it is termed beyond this approval assumed, the application will quickly crash. If permission has already been granted, then the process will execute directly. Otherwise, request Permissions will be called to launch a permission request dialog, like below.

```
private void showMessageOKCancel(String message, DialogInterface.OnClickListener okListener) {
  new AlertDialog.Builder(Main2Activity.this)
        .setMessage(message)
        .setPositiveButton("OK", okListener)
        .setNegativeButton("Cancel", null)
        .create()
        .show();
}
```

Step 4:

Now, for a button click event, we need to call PICK_CONTACT Intent, like below:

```
btnview.setOnClickListener(new OnClickListener() {
  @Override
  public void onClick(View v) {
    Intent intent = new Intent(Intent.ACTION_PICK, ContactsContract.Contacts.CONTENT_URI);
    startActivityForResult(intent, PICK_CONTACT);
  }
});
```

Step 5:

Next, for the onActivityResult function, we pick a contact and display it in a text view, like below:

```
public void onActivityResult(int reqCode, int resultCode, Intent data) {
```

```
        super.onActivityResult(reqCode, resultCode, data);
        switch (reqCode) {
            case (PICK_CONTACT):
                if (resultCode == Activity.RESULT_OK) {
                    Uri contactData = data.getData();
                    Cursor c = managedQuery(contactData, null, null, null, null);
                    if (c.moveToFirst()) {
String id = c.getString(c.getColumnIndexOrThrow(ContactsContract.Contacts._ID));
 String hasPhone =
c.getString(c.getColumnIndex(ContactsContract.Contacts.HAS_PHONE_NUMBER));
try {
    if (hasPhone.equalsIgnoreCase("1")) {
        Cursor phones = getContentResolver().query(
                ContactsContract.CommonDataKinds.Phone.CONTENT_URI, null,
                ContactsContract.CommonDataKinds.Phone.CONTACT_ID + " = " + id,
                null, null);
        phones.moveToFirst();
        String cNumber = phones.getString(phones.getColumnIndex("data1"));
        System.out.println("number is:" + cNumber);
        txtphno.setText("Phone Number is: "+cNumber);
    }
    String name = c.getString(c.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME));
    txtname.setText("Name is: "+name);
}
catch (Exception ex)
{
  st.getMessage();
}
                    }
                }
                break;
        }
    }
```

7.5 Other Built-in Content Providers

Following are useful Built in Content Providers;

Provider	**Purpose**
AlarmClock	Set Alarm within the alarm clock application
Browser	Browser history and Bookmarks
CalenderContract	Calender and event Information
CallLog	Sent and received calls
ContactsContract	Phone Contact database or phonebook
MediaStore	Audio/Visual data on the phone and external storage
SearchRecentSuggestions	Create search suggestions appropriate to the application
Settings	Systemwide Device settings and preferences
UserDictonary	A dictionary of user-defined words for use with predictive text input
VoicemailContract	A Single unified place for the user to manage voice mail content from different sources

7.6 Creating Custom Content Provider

Example Application:

In this example, we are working on creating an application which creates content provider to share data with another application.

The URI of this content provider is: "content://" + AUTHORITY + "/plates"

Step 1: Create PlatesData.java

Create PlatesData.java file and write the following content in that class.

```
publicclass PlatesData {
   public PlatesData() {
   }

   // A content URI is a URI that identifies data in a provider. Content URIs
   // include the symbolic name of the entire provider (its authority)
   publicstaticfinal String AUTHORITY = "com.tag.custom_contentproviderdemo.Plates";
   publicstaticfinal Uri CONTENT_URI = Uri.parse("content://" + AUTHORITY
         + "/plates");

   publicstaticfinal String DATABASE_NAME = "plates.db";
   publicstaticfinalintDATABASE_VERSION = 1;

   publicstaticfinal String CONTENT_TYPE_PLATES = "vnd.android.cursor.dir/vnd.tag.plates";
   publicstaticfinal String CONTENT_TYPE_PLATE = "vnd.android.cursor.item/vnd.tag.plate";

   publicclass Plates implements BaseColumns {

      private Plates() {
      }

      publicstaticfinal String TABLE_NAME = "plates";
      publicstaticfinal String _ID = "_id";
      publicstaticfinal String _TITLE = "title";
```

```
    publicstaticfinal String _CONTENT = "content";
  }
}
```

Step 2: Create PlatesContentProvider.java file and write the following content in that class.

```
publicclass PlatesContentProvider extends ContentProvider {
  privatestaticfinal UriMatcher sUriMatcher;
  privatestaticfinalintNOTES_ALL = 1;
  privatestaticfinalintNOTES_ONE = 2;

  static {
    sUriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
    sUriMatcher.addURI(PlatesData.AUTHORITY, "plates", NOTES_ALL);
    sUriMatcher.addURI(PlatesData.AUTHORITY, "plates/#", NOTES_ONE);
  }

  // Map table columns
  privatestaticfinal HashMap<String, String>sNotesColumnProjectionMap;
  static {
    sNotesColumnProjectionMap = new HashMap<String, String>();
    sNotesColumnProjectionMap.put(PlatesData.Plates._ID,
        PlatesData.Plates._ID);
    sNotesColumnProjectionMap.put(PlatesData.Plates._TITLE,
        PlatesData.Plates._TITLE);
    sNotesColumnProjectionMap.put(PlatesData.Plates._CONTENT,
        PlatesData.Plates._CONTENT);
  }

  privatestaticclass NotesDBHelper extends SQLiteOpenHelper {

    public NotesDBHelper(Context c) {
```

```
        super(c, PlatesData.DATABASE_NAME, null,
                PlatesData.DATABASE_VERSION);
    }

    privatestaticfinal String SQL_QUERY_CREATE = "CREATE TABLE "
            + PlatesData.Plates.TABLE_NAME + " (" + PlatesData.Plates._ID
            + " INTEGER PRIMARY KEY AUTOINCREMENT, "
            + PlatesData.Plates._TITLE + " TEXT NOT NULL, "
            + PlatesData.Plates._CONTENT + " TEXT NOT NULL" + ");";

    @Override
    publicvoid onCreate(SQLiteDatabase db) {
        db.execSQL(SQL_QUERY_CREATE);
    }

    privatestaticfinal String SQL_QUERY_DROP = "DROP TABLE IF EXISTS "
            + PlatesData.Plates.TABLE_NAME + ";";

    @Override
    publicvoid onUpgrade(SQLiteDatabase db, intoldVer, intnewVer) {
        db.execSQL(SQL_QUERY_DROP);
        onCreate(db);
    }
}
// create a db helper object
private NotesDBHelper mDbHelper;

@Override
publicboolean onCreate() {
    mDbHelper = new NotesDBHelper(getContext());
    returnfalse;
}
```

```
@Override
publicint delete(Uri uri, String where, String[] whereArgs) {
    SQLiteDatabase db = mDbHelper.getWritableDatabase();
    intcount = 0;
    switch (sUriMatcher.match(uri)) {
    caseNOTES_ALL:
        count = db.delete(PlatesData.Plates.TABLE_NAME, where, whereArgs);
        break;

    caseNOTES_ONE:
        String rowId = uri.getPathSegments().get(1);
        count = db.delete(
                PlatesData.Plates.TABLE_NAME,
                PlatesData.Plates._ID
                        + " = "
                        + rowId
                        + (!TextUtils.isEmpty(where) ? " AND (" + where
                                + ")" : ""), whereArgs);
        break;

    default:
        thrownew IllegalArgumentException("Unknown URI: " + uri);
    }

    getContext().getContentResolver().notifyChange(uri, null);
    returncount;
}

@Override
public String getType(Uri uri) {

    switch (sUriMatcher.match(uri)) {
```

```
    caseNOTES_ALL:
        return PlatesData.CONTENT_TYPE_PLATES;

    caseNOTES_ONE:
        return PlatesData.CONTENT_TYPE_PLATE;

    default:
        thrownew IllegalArgumentException("Unknown URI: " + uri);
    }
}

@Override
public Uri insert(Uri uri, ContentValues values) {

    // you cannot insert a bunch of values at once so throw exception
    if (sUriMatcher.match(uri) != NOTES_ALL) {
        thrownew IllegalArgumentException(" Unknown URI: " + uri);
    }

    // Insert once row
    SQLiteDatabase db = mDbHelper.getWritableDatabase();
    longrowId = db.insert(PlatesData.Plates.TABLE_NAME, null, values);
    if (rowId> 0) {
        Uri notesUri = ContentUris.withAppendedId(PlatesData.CONTENT_URI,
                rowId);
        getContext().getContentResolver().notifyChange(notesUri, null);
        returnnotesUri;
    }
    thrownew IllegalArgumentException("<Illegal>Unknown URI: " + uri);
}

// Get values from Content Provider
```

```
@Override
public Cursor query(Uri uri, String[] projection, String selection,
        String[] selectionArgs, String sortOrder) {
    SQLiteQueryBuilder builder = new SQLiteQueryBuilder();
    switch (sUriMatcher.match(uri)) {
    caseNOTES_ALL:
        builder.setTables(PlatesData.Plates.TABLE_NAME);
        builder.setProjectionMap(sNotesColumnProjectionMap);
        break;

    caseNOTES_ONE:
        builder.setTables(PlatesData.Plates.TABLE_NAME);
        builder.setProjectionMap(sNotesColumnProjectionMap);
        builder.appendWhere(PlatesData.Plates._ID + " = "
                + uri.getLastPathSegment());
        break;
    default:
        thrownew IllegalArgumentException("Unknown URI: " + uri);
    }

    SQLiteDatabase db = mDbHelper.getReadableDatabase();
    Cursor queryCursor = builder.query(db, projection, selection,
            selectionArgs, null, null, null);
    queryCursor.setNotificationUri(getContext().getContentResolver(), uri);

    returnqueryCursor;
}

@Override
publicint update(Uri uri, ContentValues values, String where,
        String[] whereArgs) {
```

```
        SQLiteDatabase db = mDbHelper.getWritableDatabase();
        intcount = 0;
        switch (sUriMatcher.match(uri)) {
        caseNOTES_ALL:
            count = db.update(PlatesData.Plates.TABLE_NAME, values, where,
                    whereArgs);
            break;
        caseNOTES_ONE:
            String rowId = uri.getLastPathSegment();
            count = db
                    .update(PlatesData.Plates.TABLE_NAME,
                            values,
                            PlatesData.Plates._ID
                                    + " = "
                                    + rowId
                                    + (!TextUtils.isEmpty(where) ? " AND ("
                                            + ")" : ""), whereArgs);
        default:
            thrownew IllegalArgumentException("Unknown URI: " + uri);
        }
        getContext().getContentResolver().notifyChange(uri, null);
        returncount;
    }
}
```

Step 3: Add_plates.xml file

Take a layout to insert new records to the database table and to delete records from the database table.

```
<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
```

```
android:orientation="vertical">

<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="vertical"
android:padding="@dimen/layout_pad">

<TextView
android:id="@+id/tvAddPlateTitle"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/title"
android:textAppearance="?android:attr/textAppearanceMedium"/>

<EditText
android:id="@+id/etAddPlateTitle"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:background="@drawable/edt_corner"
android:ems="10"
android:padding="@dimen/et_pad">

<requestFocus/>
</EditText>
</LinearLayout>

<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="vertical"
android:padding="@dimen/layout_pad">
```

```
<TextView
android:id="@+id/tvItemContent"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/content"
android:textAppearance="?android:attr/textAppearanceMedium"/>

<EditText
android:id="@+id/etAddPlateContent"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:background="@drawable/edt_corner"
android:ems="10"
android:padding="@dimen/et_pad"/>

</LinearLayout>

<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="vertical">

<Button
android:id="@+id/btnAddPlateSubmit"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="@string/Add">
</Button>
</LinearLayout>
<LinearLayout
android:layout_width="match_parent"
```

```
android:layout_height="wrap_content"
android:orientation="vertical">

<Button
android:id="@+id/btnAddPlateDelete"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="@string/delete"/>

</LinearLayout>
</LinearLayout>
```

Step 4: Create Addplate.java file and write the following content in that class.

```
publicclass AddPlate extends Activity implements OnClickListener {

   private EditText etAddTitle, etAddContent;
   private Button btnAdd, btnDelete;
   private String _ID, _TITLE, _CONTENT;
   HashMap<String, String>map;

   @Override
   protectedvoid onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.add_plate);
      setWidgetReference();
      bindWidgetEvents();
      getDataFromBundle();
   }
   privatevoid setWidgetReference() {
      etAddTitle = (EditText) findViewById(R.id.etAddPlateTitle);
      etAddContent = (EditText) findViewById(R.id.etAddPlateContent);
      btnAdd = (Button) findViewById(R.id.btnAddPlateSubmit);
```

```
    btnDelete = (Button) findViewById(R.id.btnAddPlateDelete);
}

@SuppressWarnings("unchecked")
privatevoid getDataFromBundle() {
    map = (HashMap<String, String>) getIntent().getSerializableExtra(
            Constants.TAG_MAP);
    if (map != null) {
        System.out.println("mapdata" + map.get(Constants.TAG_TITLE));
        etAddTitle.setText(map.get(Constants.TAG_TITLE));
        etAddContent.setText(map.get(Constants.TAG_CONTENT));
        btnAdd.setText("Update");
    }

}

privatevoid bindWidgetEvents() {
    btnAdd.setOnClickListener(this);
    btnDelete.setOnClickListener(this);
}

void updatePlate(String str_id) {
    try {
        intid = Integer.parseInt(str_id);
        ContentValues values = new ContentValues();
        values.put(Plates._TITLE, etAddTitle.getText().toString());
        values.put(Plates._CONTENT, etAddContent.getText().toString());
        getContentResolver().update(PlatesData.CONTENT_URI, values,
                PlatesData.Plates._ID + " = " + id, null);
        startActivity(new Intent(this, PlatesList.class));
        finish();
    } catch (Exception e) {
```

```
            e.printStackTrace();
        }
    }

    privateboolean isValid() {
        if (etAddTitle.getText().toString().length() > 0) {
            if (etAddContent.getText().toString().length() > 0) {
                returntrue;
            } else {
                etAddContent.setError("Enter Content");
            }
        } else {
            etAddContent.setError("Enter Title");
        }
        returnfalse;
    }

    void deletePlate(String str_id) {
        try {
            intid = Integer.parseInt(str_id);
            getContentResolver().delete(PlatesData.CONTENT_URI,
                    PlatesData.Plates._ID + " = " + id, null);
            startActivity(new Intent(this, PlatesList.class));
            finish();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    privatevoid addPlateToDB() {
        if (isValid()) {
            ContentValues values = new ContentValues();
```

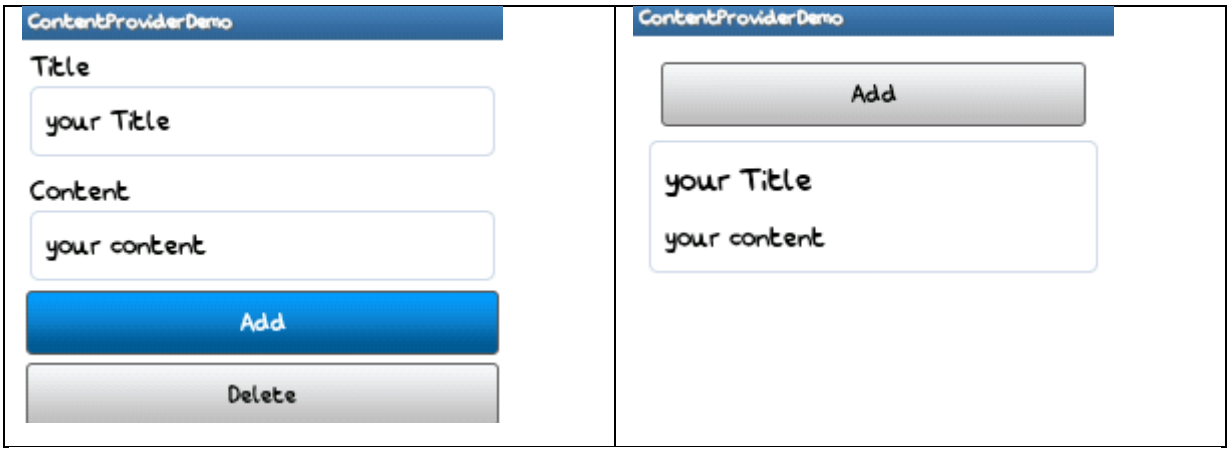
```
            values.put(Plates._TITLE, etAddTitle.getText().toString());
            values.put(Plates._CONTENT, etAddContent.getText().toString());
            getContentResolver().insert(PlatesData.CONTENT_URI, values);
            startActivity(new Intent(this, PlatesList.class));
            finish();
        }
    }
    @Override
    publicvoid onClick(View v) {
        if (v == btnAdd) {
            if (btnAdd.getText().equals("Update")) {
                updatePlate(map.get(Constants.TAG_ID));
            } else {
                addPlateToDB();
            }
        } elseif (v == btnDelete) {
            deletePlate(map.get(Constants.TAG_ID));
        }
    }
}
```

Step 5: Declare ContentProvider in AndroidManifest.xml File

```
<?xmlversion="1.0"encoding="utf-8"?>
<manifestxmlns:android="http://schemas.android.com/apk/res/android"
package="com.tag.custom_contentproviderdemo"
android:versionCode="1"
android:versionName="1.0">
<uses-sdk
android:minSdkVersion="8"
android:targetSdkVersion="17"/>

<application
```

```
android:allowBackup="true"
android:icon="@drawable/ic_launcher"
android:label="@string/app_name"
android:theme="@style/AppTheme">
<activity
android:name="com.tag.custom_contentproviderdemo.PlatesList"
android:label="@string/app_name">
<intent-filter>
<actionandroid:name="android.intent.action.MAIN"/>
<categoryandroid:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
<!-- <activity android:name="com.tag.custom_contentproviderdemo.PlatesList" /> -->
<activityandroid:name="com.tag.custom_contentproviderdemo.AddPlate" />

<provider
android:name="com.tag.custom_contentproviderdemo.PlatesContentProvider"
android:authorities="com.tag.custom_contentproviderdemo.Plates">
</provider>
</application>
</manifest>
```

Output Screen:-



Summary:

After going through this unit, you will be able to:

- Understand Content Providers
- Differentiate Content Providers types.
- Create content provider.
- Understand Contact Content Provider
- Learn types Build in Contact Content Provider
- Create Custom Contact Content Provider
- Implement Content providers.

Exercise:

1. Write a short note on Content Providers.
2. Explain the list of Methods which needs to be override in Content Provider Class.
3. Write a short note on Content URI.
4. List and Explain Built in Content Provider.
5. Write a short note on Content Resolver.

UNIT 8

RECEIVERS

Learning Objectives:

After going through this unit, you will be able to learn:

- What is Broadcast Receiver
- Basics Broadcast Receiver.
- Implement Broadcast Receiver.
- Case Study on SQL lite database.

8.1 Broadcast Receivers

- An Android component Broadcast receiver allows you to send or receive Android system or application events. Android runtime notifies the All the registered applicationonce event happens.
- Broadcast receiver works similar to the publish-subscribe design pattern and it can be used for asynchronous inter-process communication.
- Let us Consider with example, applications register for various system events like boot complete or battery low, and Android system sends broadcast when specific event occur. Custom broadcasts can also BE created by Any application for its own.

8.2 Basics of Broadcast Receiver

8.2.1 Register Broadcast:

Register broadcast can be registered in two different ways, receiver-Manifest-declared(Statically) : In this method receiver can be registered via the *AndroidManifest.xml* file.
Context-registered (Dynamically): : In this method register a receiver dynamically via the Context.registerReceiver

Receive Broadcasts:

Application have to extends the BroadcastReceiver abstract class and override its onReceive() method so that it will be able to receive a broadcast.
The on Receive() method of the receiver is called by the Android systemIf the event for which the broadcast receiver has registered happens,.

8.3 Implementing a broadcast receiver

To implement the Broadcast Receiver in Android Application:

1. Define a Broadcast Register.(It can be defined in two ways i.e one is locally in Activity class and second is Define Custom Broadcast as a class)
2. Register the receiver for particular events.(It can be defined in two ways i.e one is to register receiver in an activity and other is register receiver in Android Manifest file)
3. The receiver gets triggered once the event happens or when a custom broadcast is sent.

Note: – If user register the receiver in AndroidManifest.xml file it will also trigger if the application is killed/ not alive but if user register the receiver in Activity it will only till the application is live.

We can define the receiver locally in a class or can define the Broadcast Receiver class explicitly.

To define the broadcast receiver explicitly.

Here class MyBroadcastReceiver is explicitly defined and showed a toast in the onReceive method.

```
public class MyBroadcastReceiver extends BroadcastReceiver {


   @Override
   public void onReceive(Context context, Intent intent) {
      Toast.makeText(context,"Broadcast Received",Toast.LENGTH_SHORT).show();
   }
}
```

To define the Broadcast receiver in Activity class.

```
public class MainActivity extends AppCompatActivity {


   BroadcastReceiver receiver;


   @Override
   protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);


        receiver=new BroadcastReceiver() {
            @Override
            public void onReceive(Context context, Intent intent) {
                Toast.makeText(context,"Broadcast Received in Activity called ",Toast.LENGTH_SHORT).show();
            }
        };
    }
}
```

To register a Receiver in AndroidManifest.xml file

```
<receiver android:name=".MyBroadcastReceiver">
<intent-filter>
<action android:name="android.intent.action.ACTION_POWER_CONNECTED"/>
</intent-filter>
</receiver>
```

To register a Receiver in Activity class.

to register locally defined receiver.

```
// to register local receiver
  filter = new IntentFilter();
  // specify the action to which receiver will listen
  filter.addAction("com.local.receiver");
  registerReceiver(receiver,filter);
```

to register custom receiver.

```
//to Register custom Broadcast Receiver defined in separate class
        MyBroadcastReceiver myBroadcastReceiver=new MyBroadcastReceiver();
        IntentFilter filter1=new IntentFilter();
        filter1.addAction("Intent.ACTION_POWER_CONNECTED");
```

```
registerReceiver(myBroadcastReceiver,filter1);
```

Note:- Don’t forget to unregister the receiver if not needed.

```
@Override
  protected void onDestroy() {
     super.onDestroy();
     if(receiver!=null)
     {
        unregisterReceiver(receiver);
     }
  }
```

Example:-

Here in this program two receiver’s one is local with the custom action and other is registered in AndroidManifest file with Action power connected which will trigger once the device is connected to power.

1. Android Manifest.XML

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.coderzpassion.broadcastsample">

<application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:roundIcon="@mipmap/ic_launcher_round"
      android:supportsRtl="true"
      android:theme="@style/AppTheme">
<activity android:name=".MainActivity">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
```

```
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>

<receiver android:name=".MyBroadcastReceiver">
<intent-filter>
<action android:name="Intent.ACTION_POWER_CONNECTED"/>
</intent-filter>
</receiver>

</application>

</manifest>
```

2. MainActivity.java

```
public class MainActivity extends AppCompatActivity {
   BroadcastReceiver receiver;
   IntentFilter filter;
   Button sendbroadcast;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      receiver=new BroadcastReceiver() {
         @Override
         public void onReceive(Context context, Intent intent) {
            Toast.makeText(context,"Broadcast Received in Activity called",Toast.LENGTH_SHORT).show();
         }
      };
      // to register local receiver
```

```
        filter = new IntentFilter();
        // specify the action to which receiver will listen
        filter.addAction("com.local.receiver");
        registerReceiver(receiver,filter);
        sendbroadcast=(Button)findViewById(R.id.sendbroadcast);
        sendbroadcast.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent=new Intent("com.local.receiver");
                sendBroadcast(intent);
            }
        });
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        if(receiver!=null)
        {
            unregisterReceiver(receiver);
        }
    }
}
```

3. Activity Main.XML

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.coderzpassion.broadcastsample.MainActivity">
<Button
```

```
        android:id="@+id/sendbroadcast"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
android:text="Send BroadCast!"
        android:layout_centerInParent="true" />
</RelativeLayout>
```

4. MyBordcastReceiver.java

```
public class MyBroadcastReceiver extends BroadcastReceiver {
   @Override
   public void onReceive(Context context, Intent intent) {
      Toast.makeText(context,"Broadcast Received Device is connected to Power",Toast.LENGTH_SHORT).show();
   }
}
```

Output:-

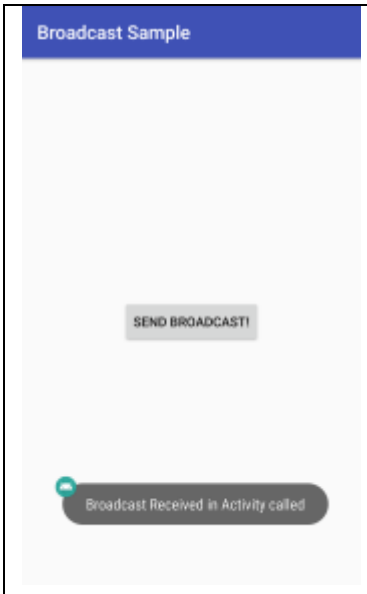


Figure8.1(A) Broadcast Received in Acitivty Called

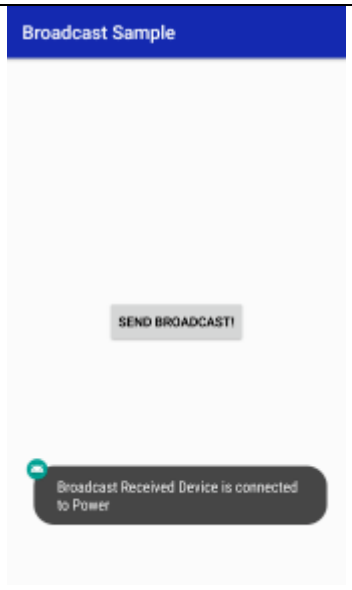


Figure8.1(B) Broadcast Received Device is connected to the Power

8.4 Case Study

Case Study on SQLite Database:-

Here we will discuss a case study to Create Login, Registration page with SQLite Database where user will register first with given username and password. And at the time of registration user has to put same user name and password if that username and password is available in SQLite database then only login will be successful otherwise it will not be successful.

MainActivity.java

```
package com.example.datewithme;

import android.os.Bundle;
import android.view.View;
import android.app.Activity;
import android.content.Intent;

public class MainActivity extends Activity {

	Intent i=null;
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);
	}

	public void login_sigin(View v)
	{
		switch(v.getId())
		{
		case R.id.log_in:
			i=new Intent(this,Login.class);
			startActivityForResult(i, 500);
			overridePendingTransition(R.anim.slide_in_right, R.anim.slide_out_left);
			break;
		case R.id.sign_in:
			i=new Intent(this,Signin.class);
			startActivityForResult(i, 500);
			overridePendingTransition(R.anim.slide_in_right, R.anim.slide_out_left);;
			break;
		}
	}
	@Override
	protected void onActivityResult(int requestCode, int resultCode, Intent data) {
	overridePendingTransition(R.anim.slide_in_left, R.anim.slide_out_right);
	}
}
```

SignIn.java

```
package com.example.datewithme;

import android.app.Activity;
import android.content.Intent;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
```

```
import android.text.InputType;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Toast;

public class Signin extends Activity{
	Intent i=null;
	ImageView im=null;
	EditText tv1,tv2,tv3,tv4;
	boolean flag=false;
	SQLiteDatabase db=null;
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.signin);
		im=(ImageView)findViewById(R.id.show_hide);
		tv1=(EditText)findViewById(R.id.name);
		tv2=(EditText)findViewById(R.id.email_id);
		tv3=(EditText)findViewById(R.id.phone);
		tv4=(EditText)findViewById(R.id.password);
		db=openOrCreateDatabase("mydb", MODE_PRIVATE, null);
		db.execSQL("create table if not exists login(name varchar,mobile_no
varchar,email_id varchar,password varchar,flag varchar)");

		im.setOnClickListener(new View.OnClickListener() {
			@Override
			public void onClick(View arg0) {

				if(flag==false)
				{
					im.setImageResource(R.drawable.hide);

	tv4.setInputType(InputType.TYPE_TEXT_VARIATION_PASSWORD);
					flag=true;
				}
				else
				{
					im.setImageResource(R.drawable.show);
					tv4.setInputType(129);
					flag=false;

				}
			}
		});
	}

	public void action(View v)
	{
	switch(v.getId())
```

```
        {
        case R.id.login:
                i=new Intent(this,Login.class);
                startActivityForResult(i, 500);
                overridePendingTransition(R.anim.slide_in_top, R.anim.slide_out_bottom);
                finish();
                break;
        case R.id.signin:
                String name=tv1.getText().toString();
                String email_id=tv2.getText().toString();
                String mobile_no=tv3.getText().toString();
                String password=tv4.getText().toString();
                if(name==null||name==""||name.length()<3)
                {
                        show("Please Enter Correct Name.");
                }
                else if(mobile_no==null||mobile_no==""||mobile_no.length()<10)
                {
                        show("Please Enter Correct mobile number.");
                }
                else if(email_id==null||email_id==""||email_id.length()<10)
                {
                        show("Please Enter Correct Email id.");
                }
                else if(password==null||password==""||password.length()<6)
                {
                        show("Please Enter Strong Password.");
                }
                else
                {
                        db.execSQL("insert into login
values('"+name+"','"+mobile_no+"','"+email_id+"','"+password+"','nothing')");
                        i=new Intent(this,Welcome.class);
                        startActivityForResult(i, 500);
                        overridePendingTransition(R.anim.slide_in_right,
R.anim.slide_out_left);
                        db.close();
                        finish();
                }
                break;
        }
 }
        @Override
        protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        overridePendingTransition(R.anim.slide_in_left, R.anim.slide_out_right);
        }

        public void show(String str)
        {
        Toast.makeText(this, str, Toast.LENGTH_LONG).show();
```

```
	}
}
```

Login.java

```
package com.example.datewithme;

import android.app.Activity;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.text.InputType;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Toast;

public class Login extends Activity{
	Intent i=null;
	ImageView im=null;
	EditText tv1,tv4;
	boolean flag=false;
	SQLiteDatabase db=null;
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.login);
		im=(ImageView)findViewById(R.id.show_hide2);
		tv1=(EditText)findViewById(R.id.phone2);
		tv4=(EditText)findViewById(R.id.password2);
		db=openOrCreateDatabase("mydb", MODE_PRIVATE, null);
	//	db.execSQL("create table if not exists login(name varchar,mobile_no
varchar,email_id varchar,password varchar,flag varchar)");

		im.setOnClickListener(new View.OnClickListener() {
			@Override
			public void onClick(View arg0) {

				if(flag==false)
				{
					im.setImageResource(R.drawable.hide);

	tv4.setInputType(InputType.TYPE_TEXT_VARIATION_PASSWORD);
					flag=true;
				}
				else
				{
					im.setImageResource(R.drawable.show);
					tv4.setInputType(129);
					flag=false;
```

```
                                }
                        }
                });
        }

        public void action(View v)
        {
        switch(v.getId())
        {
        case R.id.signin2:
                i=new Intent(this,Signin.class);
                startActivityForResult(i, 500);
                overridePendingTransition(R.anim.slide_in_top, R.anim.slide_out_bottom);
                finish();
                break;
         case R.id.start:
                String mobile_no=tv1.getText().toString();
                String password=tv4.getText().toString();
                if(mobile_no==null||mobile_no==""||mobile_no.length()<10)
                {
                        show("Please Enter Correct mobile number.");
                }
                else if(password==null||password==""||password.length()<6)
                {
                        show("Please Enter Correct Password.");
                }
                else
                {
                        Cursor c=db.rawQuery("select * from login where
mobile_no='"+mobile_no+"' and password='"+password+"'",null);
                        c.moveToFirst();
                        if(c.getCount()>0)
                        {
                        i=new Intent(this,Welcome.class);
                        startActivityForResult(i,500);
                        overridePendingTransition(R.anim.slide_in_right,
R.anim.slide_out_left);
                        db.close();
                        finish();
                        }
                        else
                                show("Wrong Password or Mobile number.");

                }
                break;
        }
  }
        @Override
        protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        overridePendingTransition(R.anim.slide_in_left, R.anim.slide_out_right);
```

```
	}

	public void show(String str)
	{
	Toast.makeText(this, str, Toast.LENGTH_LONG).show();
	}

}
```

Welcome.java

```
package com.example.datewithme;

import android.app.Activity;
import android.os.Bundle;

public class Welcome extends Activity{

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.welcome);
	}
}
```

Designing Part

ActivityMain.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:background="#999999" >

<LinearLayout
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:layout_alignParentBottom="true">

<Button
	android:id="@+id/sign_in"
	android:layout_width="wrap_content"
	android:layout_weight="1"
	android:layout_height="wrap_content"
	android:textSize="20sp"
	android:text="Sign In"
	android:onClick="login_sigin" />

<Button
	android:id="@+id/log_in"
	android:layout_width="wrap_content"
	android:layout_height="wrap_content"
```

```
        android:layout_weight="1"
        android:textSize="20sp"
        android:onClick="login_sigin"
        android:text="Log In" />

</LinearLayout>

<TextView
    android:id="@+id/text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentTop="true"
    android:background="#CCCCCC"
    android:text=" Welcome "
    android:gravity="center"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:textColor="#333333"
    android:textSize="25sp" />

</RelativeLayout>
```

Login.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:background="#999999" >

<RelativeLayout
    android:id="@+id/rl"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/phone2"
    android:layout_alignRight="@+id/phone2"
    android:layout_centerVertical="true"
    android:addStatesFromChildren="true"
    android:background="@android:drawable/edit_text"
    android:baselineAligned="false"
    android:gravity="center_vertical" >

<ImageView
        android:id="@+id/show_hide2"
        style="@android:style/Widget.Button.Inset"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignBottom="@+id/rl"
        android:layout_alignParentTop="true"
        android:layout_marginTop="4dp"
        android:background="@drawable/show"
```

```
            />

<EditText
            android:id="@+id/password2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignBottom="@+id/rl"
            android:layout_alignParentLeft="true"
            android:layout_alignParentTop="true"
            android:layout_toLeftOf="@+id/show_hide2"
            android:background="@null"
            android:ems="10"
            android:maxLength="40"
            android:focusable="true"
            android:focusableInTouchMode="true"
            android:hint="Password"
            android:inputType="textPassword"
            android:maxLines="1"
            android:singleLine="true" />

</RelativeLayout>

<EditText
      android:id="@+id/phone2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_above="@+id/rl"
      android:layout_centerHorizontal="true"
      android:ems="10"
android:hint="Mobile No."
      android:maxLength="10"
      android:inputType="phone" >

<requestFocus />
</EditText>

<Button
      android:id="@+id/signin2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentLeft="true"
      android:layout_alignParentRight="true"
      android:layout_alignParentTop="true"
      android:text="Sign in"
      android:onClick="action"
      android:textSize="20sp"/>

<Button
      android:id="@+id/start"
      android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/rl"
        android:layout_alignRight="@+id/rl"
        android:layout_below="@+id/rl"
        android:layout_marginTop="10dp"
        android:textSize="20sp"
        android:onClick="action"
android:text="Continue.." />

</RelativeLayout>
```

Signin.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:background="#999999"
>
<EditText
        android:id="@+id/email_id"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/phone"
        android:layout_centerVertical="true"
        android:ems="10"
        android:hint="Email id"
        android:maxLength="40"
        android:inputType="textEmailAddress" >

</EditText>

<EditText
        android:id="@+id/phone"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/email_id"
        android:layout_centerHorizontal="true"
        android:ems="10"
        android:maxLength="10"
android:hint="Mobile No."
        android:inputType="number" />

<EditText
        android:id="@+id/name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/phone"
        android:layout_centerHorizontal="true"
        android:ems="10"
        android:maxLength="30"
        android:hint="Name"
        android:inputType="textPersonName">
```

```
<requestFocus />
</EditText>

<RelativeLayout
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignLeft="@+id/email_id"
      android:layout_alignRight="@+id/email_id"
      android:layout_below="@+id/email_id"
      android:addStatesFromChildren="true"
      android:background="@android:drawable/edit_text"
      android:baselineAligned="false"
      android:id="@+id/rl"
      android:gravity="center_vertical" >

<ImageView
            android:id="@+id/show_hide"
            style="@android:style/Widget.Button.Inset"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
            android:layout_alignBottom="@+id/rl"
            android:layout_alignParentTop="true"
            android:layout_marginTop="4dp"
            android:background="@drawable/show"
             />

<EditText
            android:id="@+id/password"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignBottom="@+id/rl"
            android:layout_alignParentLeft="true"
            android:layout_alignParentTop="true"
            android:layout_toLeftOf="@+id/show_hide"
            android:background="@null"
            android:ems="10"
            android:maxLength="40"
            android:focusable="true"
            android:focusableInTouchMode="true"
            android:hint="Password"
            android:inputType="textPassword"
            android:maxLines="1"
            android:singleLine="true" />

</RelativeLayout>

<Button
      android:id="@+id/login"
      android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:text="Log in"
        android:onClick="action"
        android:textSize="20sp"/>

<Button
        android:id="@+id/signin"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/rl"
        android:layout_alignRight="@+id/rl"
        android:layout_below="@+id/rl"
        android:layout_marginTop="10dp"
android:text="Continue.."
        android:onClick="action"
        android:textSize="20sp" />

</RelativeLayout>
```

Welcome.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:background="#999999">

<ImageButton
        android:id="@+id/w_image"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:src="@drawable/ic_launcher" />

<TextView
        android:id="@+id/w_welcome"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/w_image"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:layout_toRightOf="@+id/w_image"
        android:text="Welcome Mohsin"
        android:gravity="center"
        android:background="#444444"
        android:textAppearance="?android:attr/textAppearanceLarge" />

</RelativeLayout>
```

Summary:

After going through this unit, you will be able to:

- Understand Broadcast Receiver
- Learn Basics Broadcast Receiver.
- Implement Broadcast Receiver.
- Work on Case Study to use and implement SQL lite database.

Exercise:

1. Write a short note on Broadcast Receiver.
2. Explain the Steps to Implement a Broadcast Receiver.
3. Explain in details how to register a Broadcast Receiver using Android Manifest File.
4. Explain in details how to register a Broadcast Receiver Programmatically
5. Write a short note on Broadcast Receiver.